

# L4Re Operating System Framework

Generated on Sun Sep 28 2025 12:17:26 for L4Re Operating System Framework by Doxygen  
1.15.0

Sun Sep 28 2025 12:17:26



<b>1 Overview</b>	<b>1</b>
<b>2 Introduction</b>	<b>3</b>
2.1 L4Re Microkernel	3
2.1.1 Communication	3
2.1.2 Kernel Objects	4
2.2 L4Re System Structure	4
2.3 L4Re Runtime Environment	6
<b>3 Tutorial</b>	<b>7</b>
3.1 Customizations	8
<b>4 Programming for L4Re</b>	<b>9</b>
4.1 L4 Inter-Process Communication (IPC)	9
4.1.1 IPC mechanism	10
4.1.1.1 IPC Flags	10
4.1.1.2 Partner capability selector	11
4.1.1.3 IPC Label	11
4.1.1.4 IPC Message Tag	11
4.1.1.5 IPC Timeouts	12
4.1.1.6 User-level Thread Control Block	13
4.1.1.7 Transfer of Typed Send Items	15
4.1.2 Examples	15
4.1.2.1 User Thread to Kernel Object	16
4.1.2.2 User Thread to User Thread	16
4.1.2.3 User Thread to User Object	18
4.2 Kernel ABI	18
4.2.1 Capability selector and flags	19
4.2.2 Label	19
4.2.3 Message tag	19
4.2.4 Timeouts	20
4.2.5 User-level thread control block (UTCB)	21
4.2.5.1 Buffer descriptor register	21
4.2.6 Typed message items	22
4.2.6.1 Flexpages	22
4.2.6.2 Send items	23
4.2.6.3 Receive items	23
4.2.6.4 Return items	24
4.3 Capabilities and Naming	25
4.4 Spaces and Mappings	26
4.5 Initial Environment and Application Bootstrapping	27
4.5.1 Configuring an application before startup	28
4.5.2 Connecting clients and servers	28

4.6 Memory management - Data Spaces and the Region Map	29
4.6.1 User-level paging	29
4.6.1.1 Data spaces	29
4.6.1.2 Virtual Memory Handling	29
4.6.1.3 Memory Allocation	29
4.7 Program Input and Output	30
4.8 Initial Memory Allocator and Factory	30
4.9 Application and Server Building Blocks	30
4.9.1 Creating Additional Application Threads	30
4.9.2 Providing a Service	31
4.10 Pthread Support	31
4.11 Interface Definition Language	32
4.11.1 Parameter types for RPC	33
4.11.2 Server Side Interface	34
4.11.3 RPC Return Types	34
4.11.4 RPC Method Declaration	35
4.12 L4Re Build System	35
4.12.1 Building L4Re	36
4.12.2 Writing BID Make Files	37
4.12.3 prog.mk - Application Role	37
4.12.4 include.mk - Header File Role	40
4.12.5 test.mk - Test Application Role	41
4.13 Kernel Factory	48
4.13.1 Passing parameters for the create stream	48
<b>5 L4Re Servers</b>	<b>51</b>
5.1 Sigma0, the Root-Pager	53
5.1.1 Factory	53
5.2 Moe, the Root-Task	53
5.2.1 Moe objects	53
5.2.1.1 Factory	54
5.2.1.2 Namespace	55
5.2.1.3 Dataspace	56
5.2.1.4 Log Subsystem	56
5.2.1.5 DMA Space	56
5.2.1.6 Scheduler subsystem	56
5.2.1.7 Region Map	56
5.2.2 Command Line Options	56
5.3 Ned, the Init Process	57
5.3.1 Lua Bindings for L4Re	57
5.3.1.1 Tutorial	57
5.3.1.2 Capabilities in Lua	58



5.3.1.3 Access to L4Re::Env Capabilities . . . . .	58
5.3.1.4 Constants . . . . .	58
5.3.1.5 Application Startup Details . . . . .	59
5.3.1.6 Reacting on task termination . . . . .	60
5.3.1.7 Control scheduling . . . . .	61
5.3.1.8 Access to the kernel debugger . . . . .	61
5.3.1.9 Using the interactive ned prompt . . . . .	61
5.3.2 Command Line Options . . . . .	62
5.4 Io, the Io Server . . . . .	62
5.5 l4vio_net_p2p, a virtual network point-to-point link . . . . .	68
5.6 Virtio Net Switch, a virtual network switch . . . . .	69
5.7 Uvmm, the virtual machine monitor . . . . .	73
5.7.1 RAM configuration . . . . .	81
5.8 RTC driver . . . . .	82
5.9 NVMe server . . . . .	82
5.10 Mag, the GUI Multiplexer . . . . .	85
5.11 eMMC driver . . . . .	88
5.12 Cons, the Console Multiplexer . . . . .	91
5.13 AHCI driver . . . . .	94
<b>6 uvmm_dtg The device tree generator for Uvmm</b>	<b>97</b>
<b>7 Bootstrap, the L4 kernel bootstrapper</b>	<b>99</b>
<b>8 Deprecated List</b>	<b>103</b>
<b>9 Topic Index</b>	<b>105</b>
9.1 Topics . . . . .	105
<b>10 Namespace Index</b>	<b>109</b>
10.1 Namespace List . . . . .	109
<b>11 Hierarchical Index</b>	<b>111</b>
11.1 Class Hierarchy . . . . .	111
<b>12 Data Structure Index</b>	<b>125</b>
12.1 Data Structures . . . . .	125
<b>13 File Index</b>	<b>143</b>
13.1 File List . . . . .	143
<b>14 Topic Documentation</b>	<b>157</b>
14.1 Base API . . . . .	157
14.1.1 Detailed Description . . . . .	160
14.1.2 Basic Macros . . . . .	160
14.1.2.1 Detailed Description . . . . .	162

14.1.2.2 Macro Definition Documentation	162
14.1.2.3 Function Documentation	164
14.1.3 Fiasco extensions	165
14.1.3.1 Detailed Description	166
14.1.3.2 Function Documentation	166
14.1.3.3 Kernel Debugger	169
14.1.3.4 Kernel Information Dump	178
14.1.3.5 Kernel Tracing	179
14.1.4 Flexpages	182
14.1.4.1 Detailed Description	184
14.1.4.2 Enumeration Type Documentation	184
14.1.4.3 Function Documentation	188
14.1.5 C++ IPC Interface Definition.	199
14.1.5.1 Detailed Description	199
14.1.5.2 Internal Helpers	200
14.1.6 Cache Consistency	200
14.1.6.1 Detailed Description	201
14.1.6.2 Function Documentation	201
14.1.7 Memory related	204
14.1.7.1 Detailed Description	205
14.1.7.2 Macro Definition Documentation	205
14.1.7.3 Enumeration Type Documentation	208
14.1.7.4 Function Documentation	209
14.1.8 Error codes	213
14.1.8.1 Detailed Description	213
14.1.8.2 Enumeration Type Documentation	214
14.1.9 Object Invocation	215
14.1.9.1 Detailed Description	216
14.1.9.2 Timeouts during IPC	217
14.1.9.3 Enumeration Type Documentation	217
14.1.9.4 Function Documentation	218
14.1.9.5 Message Items	236
14.1.9.6 Timeouts	243
14.1.9.7 Error Handling	251
14.1.9.8 Realtime API	257
14.1.9.9 Message Tag	257
14.1.9.10 Virtual Registers (UTCBS)	270
14.1.10 Kernel Objects	284
14.1.10.1 Detailed Description	285
14.1.10.2 IPC-Gate API	286
14.1.10.3 DMA space	291
14.1.10.4 L4 kernel object type information	291

14.1.10.5 Factory	293
14.1.10.6 Virtual Machines	303
14.1.10.7 Interrupt controller	325
14.1.10.8 IRQs	342
14.1.10.9 Platform Control C API	357
14.1.10.10 Scheduler	362
14.1.10.11 Kernel-provided semaphore	370
14.1.10.12 Task	373
14.1.10.13 Thread	386
14.1.10.14 Thread groups	420
14.1.10.15 Virtual Console	422
14.1.11 Kernel Interface Page	436
14.1.11.1 Detailed Description	438
14.1.11.2 Typedef Documentation	438
14.1.11.3 Enumeration Type Documentation	438
14.1.11.4 Function Documentation	439
14.1.11.5 Memory descriptors (C version)	445
14.1.12 Capabilities	450
14.1.12.1 Detailed Description	451
14.1.12.2 Typedef Documentation	451
14.1.12.3 Enumeration Type Documentation	451
14.1.12.4 Function Documentation	453
14.1.13 Memory operations.	454
14.1.13.1 Detailed Description	455
14.1.13.2 Enumeration Type Documentation	455
14.1.13.3 Function Documentation	455
14.1.14 Integer Types	457
14.1.14.1 Detailed Description	458
14.2 EDID parsing functionality	459
14.2.1 Detailed Description	459
14.2.2 Enumeration Type Documentation	459
14.2.2.1 Libedid_consts	459
14.2.3 Function Documentation	459
14.2.3.1 libedid_check_header()	459
14.2.3.2 libedid_checksum()	460
14.2.3.3 libedid_dump()	460
14.2.3.4 libedid_dump_standard_timings()	460
14.2.3.5 libedid_num_ext_blocks()	461
14.2.3.6 libedid_pnp_id()	462
14.2.3.7 libedid_prefered_resolution()	462
14.2.3.8 libedid_revision()	462
14.2.3.9 libedid_version()	463

14.3 IO interface	463
14.3.1 Detailed Description	464
14.3.2 Typedef Documentation	464
14.3.2.1 l4io_resource_t	464
14.3.3 Enumeration Type Documentation	464
14.3.3.1 l4io_device_types_t	464
14.3.3.2 l4io_iomem_flags_t	465
14.3.3.3 l4io_resource_types_t	465
14.3.4 Function Documentation	466
14.3.4.1 l4io_has_resource()	466
14.3.4.2 l4io_lookup_device()	467
14.3.4.3 l4io_lookup_resource()	467
14.3.4.4 l4io_release_iomem()	468
14.3.4.5 l4io_release_ioport()	468
14.3.4.6 l4io_request_iomem()	469
14.3.4.7 l4io_request_iomem_region()	469
14.3.4.8 l4io_request_ioport()	470
14.3.4.9 l4io_request_resource_iomem()	471
14.4 IPC Helpers	472
14.4.1 Detailed Description	472
14.4.2 Function Documentation	472
14.4.2.1 throw_ipc_exception() [1/2]	472
14.4.2.2 throw_ipc_exception() [2/2]	473
14.5 IRQ handling library	474
14.5.1 Detailed Description	474
14.5.2 Interface using direct functionality.	474
14.5.2.1 Detailed Description	475
14.5.2.2 Function Documentation	475
14.5.2.3 Interface using direct functionality.	479
14.5.3 Interface for asynchronous ISR handlers.	481
14.5.3.1 Detailed Description	482
14.5.3.2 Function Documentation	482
14.5.3.3 Interface for asynchronous ISR handlers with a given IRQ capability.	483
14.6 L4 IPC Opcodes	484
14.6.1 Detailed Description	485
14.6.2 Enumeration Type Documentation	485
14.6.2.1 L4_icu_opcode	485
14.6.2.2 L4_ipc_gate_ops	486
14.6.2.3 L4_platform_ctl_ops	486
14.6.2.4 L4_task_ops	487
14.6.2.5 L4_thread_ops	487
14.6.2.6 L4_vcon_ops	488

14.7 L4 VIRTIO Interface	488
14.7.1 Detailed Description	489
14.7.2 L4 VIRTIO Transport Layer	489
14.7.2.1 Detailed Description	491
14.7.2.2 Typedef Documentation	491
14.7.2.3 Enumeration Type Documentation	491
14.7.2.4 Function Documentation	494
14.7.3 L4 VIRTIO Block Device	499
14.7.3.1 Detailed Description	499
14.7.3.2 Enumeration Type Documentation	499
14.7.4 L4 VIRTIO Input Device	500
14.7.4.1 Detailed Description	501
14.7.5 L4 VIRTIO Network Device	501
14.7.5.1 Detailed Description	502
14.8 L4 Vbus functions	502
14.8.1 Detailed Description	503
14.8.2 Enumeration Type Documentation	503
14.8.2.1 L4vbus_dma_domain_assign_flags	503
14.8.3 Function Documentation	504
14.8.3.1 l4vbus_assign_dma_domain()	504
14.8.3.2 l4vbus_get_adr()	505
14.8.3.3 l4vbus_get_device()	505
14.8.3.4 l4vbus_get_device_by_hid()	506
14.8.3.5 l4vbus_get_hid()	507
14.8.3.6 l4vbus_get_next_device()	508
14.8.3.7 l4vbus_get_resource()	509
14.8.3.8 l4vbus_is_compatible()	510
14.8.3.9 l4vbus_release_ioport()	511
14.8.3.10 l4vbus_request_ioport()	511
14.8.3.11 l4vbus_vicu_get_cap()	512
14.8.4 L4vbus GPIO functions	513
14.8.4.1 Detailed Description	514
14.8.4.2 Enumeration Type Documentation	514
14.8.4.3 Function Documentation	514
14.8.5 L4vbus PCI functions	523
14.8.5.1 Detailed Description	524
14.8.5.2 Function Documentation	524
14.8.6 L4vbus power management functions	529
14.8.6.1 Detailed Description	530
14.8.6.2 Function Documentation	530
14.9 L4Re C Interface	531
14.9.1 Detailed Description	533

14.9.2 L4Re Util C Interface . . . . .	534
14.9.3 Dataspace interface . . . . .	534
14.9.3.1 Detailed Description . . . . .	535
14.9.3.2 Enumeration Type Documentation . . . . .	535
14.9.3.3 Function Documentation . . . . .	536
14.9.4 Debug interface . . . . .	539
14.9.4.1 Detailed Description . . . . .	540
14.9.4.2 Function Documentation . . . . .	540
14.9.5 DMA Space Interface . . . . .	540
14.9.5.1 Detailed Description . . . . .	541
14.9.5.2 Typedef Documentation . . . . .	541
14.9.5.3 Function Documentation . . . . .	541
14.9.6 Event interface . . . . .	544
14.9.6.1 Detailed Description . . . . .	545
14.9.6.2 Function Documentation . . . . .	545
14.9.7 Log interface . . . . .	547
14.9.7.1 Detailed Description . . . . .	548
14.9.7.2 Function Documentation . . . . .	548
14.9.8 Memory allocator . . . . .	550
14.9.8.1 Detailed Description . . . . .	551
14.9.8.2 Enumeration Type Documentation . . . . .	551
14.9.8.3 Function Documentation . . . . .	551
14.9.9 Namespace interface . . . . .	556
14.9.9.1 Detailed Description . . . . .	557
14.9.9.2 Enumeration Type Documentation . . . . .	557
14.9.9.3 Function Documentation . . . . .	557
14.9.10 Parent interface . . . . .	560
14.9.11 Region map interface . . . . .	560
14.9.11.1 Detailed Description . . . . .	561
14.9.11.2 Enumeration Type Documentation . . . . .	561
14.9.11.3 Function Documentation . . . . .	562
14.9.12 Capability allocator . . . . .	575
14.9.12.1 Detailed Description . . . . .	576
14.9.12.2 Function Documentation . . . . .	576
14.9.13 Kumem allocator utility . . . . .	576
14.9.14 Video API . . . . .	577
14.9.14.1 Detailed Description . . . . .	578
14.9.14.2 Typedef Documentation . . . . .	578
14.9.14.3 Enumeration Type Documentation . . . . .	578
14.9.14.4 Function Documentation . . . . .	579
14.9.15 Initial Environment . . . . .	584
14.9.15.1 Detailed Description . . . . .	585

14.9.15.2 Function Documentation . . . . .	585
14.10 L4Re C++ Interface . . . . .	589
14.10.1 Detailed Description . . . . .	591
14.10.2 L4Re Util C++ Interface . . . . .	591
14.10.2.1 Detailed Description . . . . .	592
14.10.2.2 L4Re Capability API . . . . .	592
14.10.2.3 Kumem utilities . . . . .	594
14.10.3 Console API . . . . .	596
14.10.3.1 Detailed Description . . . . .	596
14.10.4 Debugging API . . . . .	596
14.10.4.1 Detailed Description . . . . .	597
14.10.5 L4Re ELF Auxiliary Information . . . . .	597
14.10.5.1 Detailed Description . . . . .	598
14.10.5.2 Macro Definition Documentation . . . . .	598
14.10.5.3 Enumeration Type Documentation . . . . .	599
14.10.6 Event API . . . . .	599
14.10.6.1 Detailed Description . . . . .	600
14.10.7 Auxiliary data . . . . .	600
14.10.7.1 Detailed Description . . . . .	601
14.10.8 Logging interface . . . . .	601
14.10.8.1 Detailed Description . . . . .	601
14.10.9 Name-space API . . . . .	601
14.10.9.1 Detailed Description . . . . .	602
14.10.10 Parent API . . . . .	602
14.10.10.1 Detailed Description . . . . .	602
14.10.11 L4Re Protocol identifiers . . . . .	603
14.10.11.1 Detailed Description . . . . .	603
14.10.11.2 Enumeration Type Documentation . . . . .	604
14.10.12 Region map API . . . . .	605
14.10.12.1 Detailed Description . . . . .	605
14.10.13 Video API . . . . .	606
14.10.13.1 Detailed Description . . . . .	606
14.10.14 C++ Exceptions . . . . .	607
14.10.14.1 Detailed Description . . . . .	608
14.10.15 Vbus API . . . . .	608
14.10.15.1 Detailed Description . . . . .	608
14.11 L4SHM-based ring buffer implementation . . . . .	609
14.11.1 Detailed Description . . . . .	609
14.11.2 Sender . . . . .	609
14.11.3 Receiver . . . . .	610
14.11.4 Internal . . . . .	610
14.11.4.1 Detailed Description . . . . .	611

14.11.4.2 Macro Definition Documentation	611
14.12 Shared Memory Library	612
14.12.1 Detailed Description	613
14.12.2 Function Documentation	613
14.12.2.1 l4shmc_area_overhead()	613
14.12.2.2 l4shmc_area_size()	613
14.12.2.3 l4shmc_area_size_free()	614
14.12.2.4 l4shmc_attach()	614
14.12.2.5 l4shmc_chunk_overhead()	615
14.12.2.6 l4shmc_connect_chunk_signal()	615
14.12.2.7 l4shmc_create()	615
14.12.2.8 l4shmc_get_client_nr()	616
14.12.2.9 l4shmc_get_initialized_clients()	616
14.12.2.10 l4shmc_mark_client_initialized()	617
14.12.3 Chunks	617
14.12.3.1 Detailed Description	618
14.12.3.2 Function Documentation	618
14.12.3.3 Producer	621
14.12.3.4 Consumer	625
14.12.4 Signals	630
14.12.4.1 Detailed Description	631
14.12.4.2 Function Documentation	631
14.12.4.3 Producer	633
14.12.4.4 Consumer	634
14.13 Sigma0 API	639
14.13.1 Detailed Description	640
14.13.2 Enumeration Type Documentation	640
14.13.2.1 l4sigma0_return_flags_t	640
14.13.3 Function Documentation	640
14.13.3.1 l4sigma0_debug_dump()	640
14.13.3.2 l4sigma0_map_anypage()	640
14.13.3.3 l4sigma0_map_errstr()	641
14.13.3.4 l4sigma0_map_iomem()	641
14.13.3.5 l4sigma0_map_kip()	642
14.13.3.6 l4sigma0_map_mem()	642
14.13.4 Internal constants	643
14.13.4.1 Detailed Description	644
14.14 Small C++ Template Library	644
14.14.1 Detailed Description	646
14.14.2 Function Documentation	646
14.14.2.1 clamp()	646
14.14.2.2 max() [1/2]	646



14.14.2.3 max() [2/2]	646
14.14.2.4 min() [1/2]	647
14.14.2.5 min() [2/2]	647
14.14.2.6 operator new()	648
14.15 The L4Re IPC Framework	648
14.15.1 Detailed Description	648
14.15.2 Server-Side IPC framework	649
14.15.2.1 Detailed Description	650
14.15.2.2 Enumeration Type Documentation	650
14.16 Utility Functions	650
14.16.1 Detailed Description	652
14.16.2 Function Documentation	653
14.16.2.1 l4_sleep()	653
14.16.2.2 l4_touch_ro()	653
14.16.2.3 l4_touch_rw()	654
14.16.2.4 l4_usleep()	654
14.16.2.5 l4util_micros2l4to()	655
14.16.2.6 l4util_splitlog2_hdl()	655
14.16.2.7 l4util_splitlog2_size()	656
14.16.3 Bitmap graphics and fonts	657
14.16.3.1 Detailed Description	657
14.16.3.2 Functions for rendering bitmap data in frame buffers	657
14.16.3.3 Functions for rendering bitmap fonts to frame buffers	658
14.16.4 CPU related functions	658
14.16.4.1 Detailed Description	658
14.16.4.2 Function Documentation	658
14.16.5 Timestamp Counter	660
14.16.5.1 Detailed Description	661
14.16.5.2 Function Documentation	661
14.16.6 Atomic Instructions	667
14.16.6.1 Detailed Description	669
14.16.6.2 Function Documentation	669
14.16.7 Internal functions	685
14.16.7.1 Detailed Description	685
14.16.8 Bit Manipulation	685
14.16.8.1 Detailed Description	686
14.16.8.2 Function Documentation	686
14.16.9 ELF binary format	693
14.16.9.1 Detailed Description	699
14.16.9.2 Macro Definition Documentation	699
14.16.9.3 Enumeration Type Documentation	700
14.16.10 Kernel Interface Page API	719

14.16.10.1 Detailed Description	719
14.16.10.2 Macro Definition Documentation	719
14.16.10.3 Function Documentation	720
14.16.11 Comfortable Command Line Parsing	721
14.16.11.1 Detailed Description	722
14.16.11.2 Function Documentation	722
14.16.12 Random number support	724
14.16.12.1 Detailed Description	724
14.16.12.2 Function Documentation	724
14.16.13 Low-Level Thread Functions	725
14.16.14 IA32 Port I/O API	725
14.16.14.1 Detailed Description	726
14.16.14.2 Function Documentation	726
14.17 Virtio Net Switch	731
14.17.1 Detailed Description	732
14.18 vCPU Support Library	732
14.18.1 Detailed Description	733
14.18.2 Function Documentation	733
14.18.2.1 l4vcpu_irq_disable()	733
14.18.2.2 l4vcpu_irq_disable_save()	734
14.18.2.3 l4vcpu_irq_enable()	735
14.18.2.4 l4vcpu_irq_restore()	736
14.18.2.5 l4vcpu_is_irq_entry()	737
14.18.2.6 l4vcpu_is_page_fault_entry()	737
14.18.2.7 l4vcpu_print_state()	738
14.18.2.8 l4vcpu_wait_for_event()	738
14.18.3 Extended vCPU support	739
14.18.3.1 Detailed Description	739
14.18.3.2 Function Documentation	739
<b>15 Namespace Documentation</b>	<b>741</b>
15.1 cxx Namespace Reference	741
15.1.1 Detailed Description	743
15.1.2 Function Documentation	743
15.1.2.1 access_once()	743
15.1.2.2 gcd()	745
15.1.2.3 lcm()	746
15.1.2.4 write_now()	747
15.2 cxx::Bits Namespace Reference	747
15.2.1 Detailed Description	748
15.3 L4 Namespace Reference	748
15.3.1 Detailed Description	752

15.3.2 Enumeration Type Documentation . . . . .	752
15.3.2.1 anonymous enum . . . . .	752
15.3.3 Function Documentation . . . . .	752
15.3.3.1 cap_cast() [1/2] . . . . .	752
15.3.3.2 cap_cast() [2/2] . . . . .	754
15.3.3.3 cap_dynamic_cast() . . . . .	754
15.3.3.4 cap_reinterpret_cast() [1/2] . . . . .	755
15.3.3.5 cap_reinterpret_cast() [2/2] . . . . .	756
15.3.3.6 round_order() . . . . .	757
15.3.3.7 trunc_order() . . . . .	757
15.4 L4::lpc Namespace Reference . . . . .	758
15.4.1 Detailed Description . . . . .	760
15.4.2 Function Documentation . . . . .	760
15.4.2.1 buf_cp_in() . . . . .	760
15.4.2.2 buf_cp_out() . . . . .	761
15.4.2.3 buf_in() . . . . .	761
15.4.2.4 make_cap() . . . . .	762
15.4.2.5 make_cap_full() . . . . .	763
15.4.2.6 make_cap_rw() . . . . .	763
15.4.2.7 make_cap_rws() . . . . .	764
15.4.2.8 msg_ptr() . . . . .	765
15.4.2.9 read() . . . . .	765
15.4.2.10 str_cp_in() . . . . .	765
15.5 L4::lpc::Msg Namespace Reference . . . . .	766
15.5.1 Detailed Description . . . . .	767
15.5.2 Enumeration Type Documentation . . . . .	767
15.5.2.1 anonymous enum . . . . .	767
15.5.3 Function Documentation . . . . .	768
15.5.3.1 align_to() [1/2] . . . . .	768
15.5.3.2 align_to() [2/2] . . . . .	769
15.5.3.3 check_size() [1/2] . . . . .	769
15.5.3.4 check_size() [2/2] . . . . .	770
15.5.3.5 msg_add() . . . . .	771
15.5.3.6 msg_get() . . . . .	772
15.6 L4::lpc_svr Namespace Reference . . . . .	773
15.6.1 Detailed Description . . . . .	774
15.7 L4::Typeid Namespace Reference . . . . .	774
15.7.1 Detailed Description . . . . .	774
15.8 L4::Types Namespace Reference . . . . .	774
15.8.1 Detailed Description . . . . .	775
15.9 L4Re Namespace Reference . . . . .	775
15.9.1 Detailed Description . . . . .	777

15.9.2 Typedef Documentation	777
15.9.2.1 Shared_cap	777
15.9.2.2 shared_cap	778
15.9.2.3 Shared_del_cap	778
15.9.2.4 shared_del_cap	779
15.9.2.5 Unique_cap	780
15.9.2.6 unique_cap	780
15.9.2.7 Unique_del_cap	781
15.9.2.8 unique_del_cap	782
15.9.3 Function Documentation	782
15.9.3.1 chkcap()	782
15.9.3.2 chkipc()	784
15.9.3.3 chksys() [1/3]	785
15.9.3.4 chksys() [2/3]	786
15.9.3.5 chksys() [3/3]	787
15.9.3.6 make_shared_cap()	788
15.9.3.7 make_shared_del_cap()	790
15.9.3.8 make_unique_cap()	791
15.9.3.9 make_unique_del_cap()	792
15.9.3.10 throw_error()	792
15.10 L4Re::Util Namespace Reference	794
15.10.1 Detailed Description	797
15.10.2 Typedef Documentation	797
15.10.2.1 Shared_cap	797
15.10.2.2 shared_cap	797
15.10.2.3 Shared_del_cap	798
15.10.2.4 shared_del_cap	799
15.10.2.5 Unique_cap	799
15.10.2.6 unique_cap	800
15.10.2.7 Unique_del_cap	800
15.10.2.8 unique_del_cap	801
15.10.3 Function Documentation	802
15.10.3.1 make_shared_cap()	802
15.10.3.2 make_shared_del_cap()	802
15.10.3.3 make_unique_cap()	802
15.10.3.4 make_unique_del_cap()	803
15.11 L4Re::Vfs Namespace Reference	803
15.11.1 Detailed Description	804
15.12 L4vbus Namespace Reference	804
15.12.1 Detailed Description	805
15.13 L4virtio Namespace Reference	805
15.13.1 Detailed Description	806

<b>16 Data Structure Documentation</b>	<b>807</b>
16.1 Block_device::Device_discard_feature Struct Reference	807
16.1.1 Detailed Description	807
16.2 Block_device::Device_mgr< DEV, FACTORY, SCHEDULER > Class Template Reference	808
16.2.1 Detailed Description	808
16.2.2 Member Function Documentation	809
16.2.2.1 parse_device_name()	809
16.3 Block_device::Device_with_notification_domain< DEV > Struct Template Reference	810
16.3.1 Detailed Description	810
16.4 Block_device::Dma_region_info Struct Reference	811
16.4.1 Detailed Description	811
16.5 Block_device::Errand::Errand Class Reference	811
16.5.1 Detailed Description	814
16.5.2 Member Function Documentation	814
16.5.2.1 expired()	814
16.6 Block_device::Errand::Poll_errand Class Reference	815
16.6.1 Detailed Description	817
16.6.2 Member Function Documentation	817
16.6.2.1 expired()	817
16.7 Block_device::Impl::Partitioned_device_discard_mixin< PART_DEV, BASE_DEV, bool > Class Template Reference	818
16.7.1 Detailed Description	819
16.8 Block_device::Impl::Partitioned_device_discard_mixin< PART_DEV, BASE_DEV, true > Class Template Reference	819
16.8.1 Detailed Description	820
16.9 Block_device::Inout_block Struct Reference	821
16.9.1 Detailed Description	821
16.10 Block_device::Inout_memory< DEV > Class Template Reference	822
16.10.1 Detailed Description	822
16.11 Block_device::Mem_region_info Struct Reference	823
16.11.1 Detailed Description	823
16.12 Block_device::Notification_domain Struct Reference	823
16.12.1 Detailed Description	824
16.13 Block_device::Partition_info Struct Reference	824
16.13.1 Detailed Description	825
16.14 Block_device::Partition_reader< DEV > Class Template Reference	825
16.14.1 Detailed Description	825
16.15 Block_device::Partitioned_device< BASE_DEV > Class Template Reference	826
16.15.1 Detailed Description	827
16.16 Block_device::Pending_request Struct Reference	828
16.16.1 Detailed Description	828
16.16.2 Member Function Documentation	828
16.16.2.1 fail_request()	828

16.16.2.2 <code>handle_request()</code> . . . . .	828
16.17 <code>Block_device::Rr_scheduler&lt; DEV &gt;</code> Struct Template Reference . . . . .	829
16.17.1 Detailed Description . . . . .	830
16.18 <code>Block_device::Scheduler_base&lt; DEV &gt;</code> Class Template Reference . . . . .	831
16.18.1 Detailed Description . . . . .	832
16.19 Buffer Struct Reference . . . . .	832
16.19.1 Detailed Description . . . . .	834
16.20 <code>cx::arith::Ld&lt; V &gt;</code> Struct Template Reference . . . . .	834
16.20.1 Detailed Description . . . . .	834
16.21 <code>cx::Avl_map&lt; KEY_TYPE, DATA_TYPE, COMPARE, ALLOC &gt;</code> Class Template Reference . . . . .	835
16.21.1 Detailed Description . . . . .	838
16.21.2 Constructor & Destructor Documentation . . . . .	838
16.21.2.1 <code>Avl_map()</code> . . . . .	838
16.21.3 Member Function Documentation . . . . .	839
16.21.3.1 <code>insert()</code> . . . . .	839
16.21.3.2 <code>operator[]()</code> [1/2] . . . . .	839
16.21.3.3 <code>operator[]()</code> [2/2] . . . . .	840
16.22 <code>cx::Avl_set&lt; ITEM_TYPE, COMPARE, ALLOC &gt;</code> Class Template Reference . . . . .	840
16.22.1 Detailed Description . . . . .	844
16.23 <code>cx::Avl_tree&lt; Node, Get_key, Compare &gt;</code> Class Template Reference . . . . .	844
16.23.1 Detailed Description . . . . .	849
16.23.2 Member Typedef Documentation . . . . .	850
16.23.2.1 Iterator . . . . .	850
16.23.3 Member Function Documentation . . . . .	850
16.23.3.1 <code>insert()</code> . . . . .	850
16.23.3.2 <code>remove()</code> . . . . .	851
16.24 <code>cx::Avl_tree_node</code> Class Reference . . . . .	853
16.24.1 Detailed Description . . . . .	855
16.25 <code>cx::Base_slab&lt; Obj_size, Slab_size, Max_free, Alloc &gt;</code> Class Template Reference . . . . .	855
16.25.1 Detailed Description . . . . .	857
16.25.2 Member Enumeration Documentation . . . . .	857
16.25.2.1 anonymous enum . . . . .	857
16.25.3 Member Function Documentation . . . . .	858
16.25.3.1 <code>alloc()</code> . . . . .	858
16.25.3.2 <code>free()</code> . . . . .	858
16.25.3.3 <code>free_objects()</code> . . . . .	859
16.25.3.4 <code>total_objects()</code> . . . . .	859
16.26 <code>cx::Base_slab&lt; Obj_size, Slab_size, Max_free, Alloc &gt;::Slab_i</code> Struct Reference . . . . .	859
16.26.1 Detailed Description . . . . .	860
16.27 <code>cx::Base_slab_static&lt; Obj_size, Slab_size, Max_free, Alloc &gt;</code> Class Template Reference . . . . .	860
16.27.1 Detailed Description . . . . .	862
16.27.2 Member Enumeration Documentation . . . . .	863

16.27.2.1 anonymous enum . . . . .	863
16.27.3 Member Function Documentation . . . . .	863
16.27.3.1 alloc() . . . . .	863
16.27.3.2 free() . . . . .	863
16.27.3.3 free_objects() . . . . .	864
16.27.3.4 total_objects() . . . . .	864
16.28 cxx::Bitfield< T, LSB, MSB > Class Template Reference . . . . .	865
16.28.1 Detailed Description . . . . .	866
16.28.2 Member Typedef Documentation . . . . .	867
16.28.2.1 Bits_type . . . . .	867
16.28.2.2 Shift_type . . . . .	867
16.28.3 Member Enumeration Documentation . . . . .	867
16.28.3.1 anonymous enum . . . . .	867
16.28.3.2 Masks . . . . .	867
16.28.4 Member Function Documentation . . . . .	868
16.28.4.1 get() . . . . .	868
16.28.4.2 get_unshifted() . . . . .	868
16.28.4.3 set() . . . . .	868
16.28.4.4 set_dirty() . . . . .	869
16.28.4.5 set_unshifted() . . . . .	870
16.28.4.6 set_unshifted_dirty() . . . . .	870
16.28.4.7 val() . . . . .	871
16.28.4.8 val_dirty() . . . . .	871
16.28.4.9 val_unshifted() . . . . .	872
16.29 cxx::Bitfield< T, LSB, MSB >::Value< TT > Class Template Reference . . . . .	873
16.29.1 Detailed Description . . . . .	874
16.30 cxx::Bitfield< T, LSB, MSB >::Value_base< TT > Class Template Reference . . . . .	874
16.30.1 Detailed Description . . . . .	875
16.31 cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT > Class Template Reference . . . . .	875
16.31.1 Detailed Description . . . . .	876
16.32 cxx::Bitmap< BITS > Class Template Reference . . . . .	876
16.32.1 Detailed Description . . . . .	879
16.32.2 Member Function Documentation . . . . .	880
16.32.2.1 scan_zero() . . . . .	880
16.33 cxx::Bitmap_base Class Reference . . . . .	881
16.33.1 Detailed Description . . . . .	883
16.33.2 Member Enumeration Documentation . . . . .	883
16.33.2.1 anonymous enum . . . . .	883
16.33.3 Member Function Documentation . . . . .	884
16.33.3.1 atomic_clear_bit() . . . . .	884
16.33.3.2 atomic_get_and_clear() . . . . .	884
16.33.3.3 atomic_get_and_set() . . . . .	885

16.33.3.4 <a href="#">atomic_set_bit()</a>	885
16.33.3.5 <a href="#">bit()</a> [1/2]	886
16.33.3.6 <a href="#">bit()</a> [2/2]	887
16.33.3.7 <a href="#">bit_index()</a>	888
16.33.3.8 <a href="#">clear_bit()</a>	889
16.33.3.9 <a href="#">operator[]()</a> [1/2]	889
16.33.3.10 <a href="#">operator[]()</a> [2/2]	890
16.33.3.11 <a href="#">scan_zero()</a>	890
16.33.3.12 <a href="#">set_bit()</a>	891
16.33.3.13 <a href="#">word_index()</a>	892
16.34 <a href="#">cxx::Bitmap_base::Bit</a> Class Reference	893
16.34.1 Detailed Description	894
16.35 <a href="#">cxx::Bitmap_base::Char&lt; BITS &gt;</a> Class Template Reference	894
16.35.1 Detailed Description	894
16.36 <a href="#">cxx::Bitmap_base::Word&lt; BITS &gt;</a> Class Template Reference	895
16.36.1 Detailed Description	895
16.37 <a href="#">cxx::Bits::Avl_map_get_key&lt; KEY_TYPE &gt;</a> Struct Template Reference	896
16.37.1 Detailed Description	896
16.38 <a href="#">cxx::Bits::Avl_set_get_key&lt; KEY_TYPE &gt;</a> Struct Template Reference	896
16.38.1 Detailed Description	897
16.39 <a href="#">cxx::Bits::Base_avl_set&lt; ITEM_TYPE, COMPARE, ALLOC, GET_KEY &gt;</a> Class Template Reference	897
16.39.1 Detailed Description	899
16.39.2 Member Enumeration Documentation	900
16.39.2.1 anonymous enum	900
16.39.3 Constructor & Destructor Documentation	900
16.39.3.1 <a href="#">Base_avl_set()</a> [1/2]	900
16.39.3.2 <a href="#">Base_avl_set()</a> [2/2]	901
16.39.4 Member Function Documentation	902
16.39.4.1 <a href="#">begin()</a> [1/2]	902
16.39.4.2 <a href="#">begin()</a> [2/2]	902
16.39.4.3 <a href="#">end()</a> [1/2]	903
16.39.4.4 <a href="#">end()</a> [2/2]	903
16.39.4.5 <a href="#">erase()</a>	903
16.39.4.6 <a href="#">find_node()</a>	904
16.39.4.7 <a href="#">insert()</a>	904
16.39.4.8 <a href="#">lower_bound_node()</a>	905
16.39.4.9 <a href="#">rbegin()</a> [1/2]	906
16.39.4.10 <a href="#">rbegin()</a> [2/2]	906
16.39.4.11 <a href="#">remove()</a>	906
16.39.4.12 <a href="#">rend()</a> [1/2]	907
16.39.4.13 <a href="#">rend()</a> [2/2]	907
16.40 <a href="#">cxx::Bits::Base_avl_set&lt; ITEM_TYPE, COMPARE, ALLOC, GET_KEY &gt;::Node</a> Class Reference	908



16.40.1 Detailed Description	908
16.40.2 Member Function Documentation	909
16.40.2.1 operator*()	909
16.40.2.2 operator->()	909
16.40.2.3 valid()	909
16.41 cxx::Bits::Basic_list< POLICY > Class Template Reference	910
16.41.1 Detailed Description	911
16.41.2 Member Function Documentation	911
16.41.2.1 clear()	911
16.41.2.2 iter()	912
16.42 cxx::Bits::Bst< Node, Get_key, Compare > Class Template Reference	913
16.42.1 Detailed Description	917
16.42.2 Member Function Documentation	917
16.42.2.1 begin() [1/2]	917
16.42.2.2 begin() [2/2]	918
16.42.2.3 dir() [1/2]	918
16.42.2.4 dir() [2/2]	919
16.42.2.5 end() [1/2]	920
16.42.2.6 end() [2/2]	920
16.42.2.7 find()	921
16.42.2.8 find_node()	921
16.42.2.9 lower_bound_node()	922
16.42.2.10 rbegin() [1/2]	923
16.42.2.11 rbegin() [2/2]	923
16.42.2.12 remove_all()	924
16.42.2.13 remove_tree()	925
16.42.2.14 rend() [1/2]	926
16.42.2.15 rend() [2/2]	926
16.43 cxx::Bits::Bst_node Class Reference	927
16.43.1 Detailed Description	929
16.44 cxx::Bits::Direction Struct Reference	929
16.44.1 Detailed Description	930
16.44.2 Member Enumeration Documentation	930
16.44.2.1 Direction_e	930
16.44.3 Member Function Documentation	931
16.44.3.1 operator"!()	931
16.45 cxx::Bits::Smart_ptr_list< ITEM > Class Template Reference	931
16.45.1 Detailed Description	933
16.45.2 Member Function Documentation	934
16.45.2.1 pop_front()	934
16.46 cxx::Bits::Smart_ptr_list_item< T, STORE_T > Class Template Reference	934
16.46.1 Detailed Description	936

16.47 cxx::H_list< T, POLICY > Class Template Reference . . . . .	936
16.47.1 Detailed Description . . . . .	940
16.47.2 Member Function Documentation . . . . .	940
16.47.2.1 erase() . . . . .	940
16.47.2.2 insert() . . . . .	940
16.47.2.3 insert_after() . . . . .	941
16.47.2.4 insert_before() . . . . .	941
16.47.2.5 iter() . . . . .	942
16.47.2.6 pop_front() . . . . .	943
16.47.2.7 remove() . . . . .	943
16.47.2.8 replace() . . . . .	943
16.48 cxx::H_list_item_t< ELEM_TYPE > Class Template Reference . . . . .	944
16.48.1 Detailed Description . . . . .	945
16.48.2 Constructor & Destructor Documentation . . . . .	945
16.48.2.1 H_list_item_t() . . . . .	945
16.48.2.2 ~H_list_item_t() . . . . .	946
16.49 cxx::H_list_t< T > Struct Template Reference . . . . .	946
16.49.1 Detailed Description . . . . .	950
16.50 cxx::List< D, Alloc > Class Template Reference . . . . .	950
16.50.1 Detailed Description . . . . .	951
16.50.2 Member Function Documentation . . . . .	951
16.50.2.1 operator[]() [1/2] . . . . .	951
16.50.2.2 operator[]() [2/2] . . . . .	952
16.51 cxx::List< D, Alloc >::Iter Class Reference . . . . .	952
16.51.1 Detailed Description . . . . .	952
16.52 cxx::List_alloc Class Reference . . . . .	953
16.52.1 Detailed Description . . . . .	953
16.52.2 Constructor & Destructor Documentation . . . . .	953
16.52.2.1 List_alloc() . . . . .	953
16.52.3 Member Function Documentation . . . . .	954
16.52.3.1 alloc() . . . . .	954
16.52.3.2 alloc_max() . . . . .	954
16.52.3.3 avail() . . . . .	955
16.52.3.4 free() . . . . .	955
16.53 cxx::List_item Class Reference . . . . .	956
16.53.1 Detailed Description . . . . .	957
16.53.2 Member Function Documentation . . . . .	957
16.53.2.1 push_back() . . . . .	957
16.53.2.2 push_front() . . . . .	958
16.53.2.3 remove() . . . . .	959
16.54 cxx::List_item::Iter Class Reference . . . . .	960
16.54.1 Detailed Description . . . . .	961

16.55 <code>cx::List_item::T_iter&lt; T, Poly &gt;</code> Class Template Reference . . . . .	961
16.55.1 Detailed Description . . . . .	962
16.56 <code>cx::Lt_functor&lt; Obj &gt;</code> Struct Template Reference . . . . .	963
16.56.1 Detailed Description . . . . .	963
16.57 <code>cx::New_allocator&lt; _Type &gt;</code> Class Template Reference . . . . .	963
16.57.1 Detailed Description . . . . .	964
16.58 <code>cx::Nothrow</code> Class Reference . . . . .	964
16.58.1 Detailed Description . . . . .	965
16.59 <code>cx::Pair&lt; First, Second &gt;</code> Struct Template Reference . . . . .	965
16.59.1 Detailed Description . . . . .	966
16.59.2 Constructor & Destructor Documentation . . . . .	967
16.59.2.1 <code>Pair()</code> [1/2] . . . . .	967
16.59.2.2 <code>Pair()</code> [2/2] . . . . .	967
16.60 <code>cx::Pair_first_compare&lt; Cmp, Typ &gt;</code> Class Template Reference . . . . .	968
16.60.1 Detailed Description . . . . .	968
16.60.2 Constructor & Destructor Documentation . . . . .	968
16.60.2.1 <code>Pair_first_compare()</code> . . . . .	968
16.60.3 Member Function Documentation . . . . .	969
16.60.3.1 <code>operator()</code> . . . . .	969
16.61 <code>cx::Ref_obj_list_item&lt; T &gt;</code> Struct Template Reference . . . . .	969
16.61.1 Detailed Description . . . . .	971
16.62 <code>cx::Ref_ptr&lt; T, CNT &gt;</code> Class Template Reference . . . . .	971
16.62.1 Detailed Description . . . . .	973
16.62.2 Constructor & Destructor Documentation . . . . .	974
16.62.2.1 <code>Ref_ptr()</code> [1/3] . . . . .	974
16.62.2.2 <code>Ref_ptr()</code> [2/3] . . . . .	974
16.62.2.3 <code>Ref_ptr()</code> [3/3] . . . . .	974
16.62.3 Member Function Documentation . . . . .	975
16.62.3.1 <code>get()</code> . . . . .	975
16.62.3.2 <code>ptr()</code> . . . . .	975
16.62.3.3 <code>release()</code> . . . . .	975
16.63 <code>cx::S_list&lt; T, POLICY &gt;</code> Class Template Reference . . . . .	976
16.63.1 Detailed Description . . . . .	978
16.63.2 Member Function Documentation . . . . .	979
16.63.2.1 <code>pop_front()</code> . . . . .	979
16.64 <code>cx::Slab&lt; Type, Slab_size, Max_free, Alloc &gt;</code> Class Template Reference . . . . .	979
16.64.1 Detailed Description . . . . .	982
16.64.2 Member Function Documentation . . . . .	982
16.64.2.1 <code>alloc()</code> . . . . .	982
16.64.2.2 <code>free()</code> . . . . .	983
16.65 <code>cx::Slab_static&lt; Type, Slab_size, Max_free, Alloc &gt;</code> Class Template Reference . . . . .	984
16.65.1 Detailed Description . . . . .	987

16.65.2 Member Function Documentation	987
16.65.2.1 alloc()	987
16.66 cxx::static_vector< T, IDX > Class Template Reference	988
16.66.1 Detailed Description	989
16.67 cxx::String Class Reference	989
16.67.1 Detailed Description	992
16.67.2 Constructor & Destructor Documentation	993
16.67.2.1 String()	993
16.67.3 Member Function Documentation	993
16.67.3.1 find()	993
16.67.3.2 from_dec()	994
16.67.3.3 from_hex()	995
16.67.3.4 starts_with()	995
16.68 cxx::Weak_ref< T > Class Template Reference	996
16.68.1 Detailed Description	998
16.69 cxx::Weak_ref_base Class Reference	1000
16.69.1 Detailed Description	1003
16.70 cxx::Weak_ref_base::List Struct Reference	1003
16.70.1 Detailed Description	1007
16.71 Elf32_Auxv Struct Reference	1007
16.71.1 Detailed Description	1007
16.71.2 Field Documentation	1008
16.71.2.1 atype	1008
16.72 Elf32_Dyn Struct Reference	1008
16.72.1 Detailed Description	1008
16.72.2 Field Documentation	1009
16.72.2.1 d_tag	1009
16.73 Elf32_Ehdr Struct Reference	1009
16.73.1 Detailed Description	1010
16.73.2 Field Documentation	1010
16.73.2.1 e_flags	1010
16.73.2.2 e_machine	1011
16.73.2.3 e_type	1011
16.73.2.4 e_version	1011
16.74 Elf32_Phdr Struct Reference	1012
16.74.1 Detailed Description	1012
16.74.2 Field Documentation	1013
16.74.2.1 p_flags	1013
16.74.2.2 p_type	1013
16.75 Elf32_Rel Struct Reference	1013
16.75.1 Detailed Description	1014
16.76 Elf32_Rela Struct Reference	1014

16.76.1 Detailed Description	1014
16.77 Elf32_Shdr Struct Reference	1015
16.77.1 Detailed Description	1016
16.77.2 Field Documentation	1016
16.77.2.1 sh_flags	1016
16.77.2.2 sh_type	1016
16.78 Elf32_Sym Struct Reference	1016
16.78.1 Detailed Description	1017
16.79 Elf64_Auxv Struct Reference	1017
16.79.1 Detailed Description	1018
16.79.2 Field Documentation	1018
16.79.2.1 atype	1018
16.80 Elf64_Dyn Struct Reference	1018
16.80.1 Detailed Description	1019
16.80.2 Field Documentation	1019
16.80.2.1 d_tag	1019
16.81 Elf64_Ehdr Struct Reference	1019
16.81.1 Detailed Description	1020
16.81.2 Field Documentation	1020
16.81.2.1 e_flags	1020
16.81.2.2 e_machine	1021
16.81.2.3 e_type	1021
16.81.2.4 e_version	1021
16.82 Elf64_Phdr Struct Reference	1022
16.82.1 Detailed Description	1022
16.82.2 Field Documentation	1023
16.82.2.1 p_flags	1023
16.82.2.2 p_type	1023
16.83 Elf64_Rel Struct Reference	1023
16.83.1 Detailed Description	1024
16.84 Elf64_Rela Struct Reference	1024
16.84.1 Detailed Description	1024
16.85 Elf64_Shdr Struct Reference	1025
16.85.1 Detailed Description	1026
16.85.2 Field Documentation	1026
16.85.2.1 sh_flags	1026
16.85.2.2 sh_type	1026
16.86 Elf64_Sym Struct Reference	1026
16.86.1 Detailed Description	1027
16.87 gfxbitmap_offset Struct Reference	1027
16.87.1 Detailed Description	1028
16.88 L4::Alloc_list Class Reference	1028

16.88.1 Detailed Description	1028
16.89 L4::Arm_smccc Class Reference	1029
16.89.1 Detailed Description	1029
16.89.2 Member Function Documentation	1029
16.89.2.1 call()	1029
16.90 L4::Base_exception Class Reference	1030
16.90.1 Detailed Description	1032
16.91 L4::Basic_registry Class Reference	1033
16.91.1 Detailed Description	1034
16.91.2 Member Function Documentation	1034
16.91.2.1 dispatch()	1034
16.91.2.2 find()	1035
16.92 L4::Bounds_error Class Reference	1035
16.92.1 Detailed Description	1038
16.93 L4::Cap< T > Class Template Reference	1038
16.93.1 Detailed Description	1041
16.93.2 Constructor & Destructor Documentation	1042
16.93.2.1 Cap() [1/4]	1042
16.93.2.2 Cap() [2/4]	1042
16.93.2.3 Cap() [3/4]	1042
16.93.2.4 Cap() [4/4]	1042
16.93.3 Member Function Documentation	1043
16.93.3.1 check_castable_from()	1043
16.93.3.2 check_convertible_from()	1043
16.93.3.3 copy()	1043
16.93.3.4 move()	1044
16.94 L4::Cap_base Class Reference	1044
16.94.1 Detailed Description	1046
16.94.2 Member Enumeration Documentation	1046
16.94.2.1 Cap_type	1046
16.94.2.2 No_init_type	1047
16.94.3 Constructor & Destructor Documentation	1047
16.94.3.1 Cap_base() [1/2]	1047
16.94.3.2 Cap_base() [2/2]	1048
16.94.4 Member Function Documentation	1049
16.94.4.1 cap()	1049
16.94.4.2 copy()	1050
16.94.4.3 fpage()	1052
16.94.4.4 is_valid()	1053
16.94.4.5 move()	1054
16.94.4.6 snd_base()	1055
16.94.4.7 validate() [1/2]	1056

16.94.4.8 validate() [2/2]	1057
16.95 L4::Com_error Class Reference	1058
16.95.1 Detailed Description	1060
16.95.2 Constructor & Destructor Documentation	1060
16.95.2.1 Com_error()	1060
16.96 L4::Debugger Class Reference	1061
16.96.1 Detailed Description	1064
16.96.2 Member Function Documentation	1064
16.96.2.1 add_image_info()	1064
16.96.2.2 get_object_name()	1065
16.96.2.3 global_id()	1065
16.96.2.4 kobj_to_id()	1066
16.96.2.5 query_log_name()	1067
16.96.2.6 query_log_typeid()	1067
16.96.2.7 set_object_name()	1068
16.96.2.8 switch_log()	1069
16.97 L4::Element_already_exists Class Reference	1070
16.97.1 Detailed Description	1072
16.98 L4::Element_not_found Class Reference	1072
16.98.1 Detailed Description	1075
16.99 L4::Epiface Struct Reference	1075
16.99.1 Detailed Description	1077
16.99.2 Member Function Documentation	1077
16.99.2.1 dispatch()	1077
16.99.2.2 get_buffer_demand()	1078
16.99.2.3 obj_cap()	1078
16.99.2.4 server_iface()	1079
16.99.2.5 set_server()	1079
16.100 L4::Epiface_t< Derived, IFACE, BASE, bool > Struct Template Reference	1080
16.100.1 Detailed Description	1083
16.100.2 Member Function Documentation	1084
16.100.2.1 dispatch()	1084
16.101 L4::Epiface_t0< RPC_IFACE, BASE > Struct Template Reference	1084
16.101.1 Detailed Description	1086
16.101.2 Member Function Documentation	1087
16.101.2.1 obj_cap()	1087
16.102 L4::Exception Class Reference	1087
16.102.1 Detailed Description	1088
16.102.2 Member Function Documentation	1088
16.102.2.1 exception()	1088
16.103 L4::Exception_tracer Class Reference	1089
16.103.1 Detailed Description	1090

16.104 L4::Factory Class Reference . . . . .	1090
16.104.1 Detailed Description . . . . .	1094
16.104.2 Member Function Documentation . . . . .	1094
16.104.2.1 create() [1/2] . . . . .	1094
16.104.2.2 create() [2/2] . . . . .	1095
16.104.2.3 create_factory() . . . . .	1096
16.104.2.4 create_gate() . . . . .	1097
16.104.2.5 create_task() . . . . .	1099
16.104.2.6 create_thread_group() . . . . .	1100
16.105 L4::Factory::Lstr Struct Reference . . . . .	1101
16.105.1 Detailed Description . . . . .	1102
16.105.2 Constructor & Destructor Documentation . . . . .	1102
16.105.2.1 Lstr() . . . . .	1102
16.106 L4::Factory::Nil Struct Reference . . . . .	1103
16.106.1 Detailed Description . . . . .	1103
16.107 L4::Factory::S Class Reference . . . . .	1103
16.107.1 Detailed Description . . . . .	1105
16.107.2 Constructor & Destructor Documentation . . . . .	1105
16.107.2.1 S() [1/2] . . . . .	1105
16.107.2.2 S() [2/2] . . . . .	1105
16.107.2.3 ~S() . . . . .	1106
16.107.3 Member Function Documentation . . . . .	1106
16.107.3.1 operator l4_msgtag_t() . . . . .	1106
16.107.3.2 operator<<() [1/2] . . . . .	1106
16.107.3.3 operator<<() [2/2] . . . . .	1107
16.107.3.4 put() [1/5] . . . . .	1108
16.107.3.5 put() [2/5] . . . . .	1108
16.107.3.6 put() [3/5] . . . . .	1108
16.107.3.7 put() [4/5] . . . . .	1109
16.107.3.8 put() [5/5] . . . . .	1109
16.108 L4::lcu Class Reference . . . . .	1109
16.108.1 Detailed Description . . . . .	1113
16.108.2 Member Function Documentation . . . . .	1113
16.108.2.1 bind() . . . . .	1113
16.108.2.2 info() . . . . .	1114
16.108.2.3 mask() . . . . .	1115
16.108.2.4 msi_info() . . . . .	1116
16.108.2.5 set_mode() . . . . .	1117
16.108.2.6 unbind() . . . . .	1118
16.109 L4::lcu::Info Class Reference . . . . .	1119
16.109.1 Detailed Description . . . . .	1120
16.110 L4::Invalid_capability Class Reference . . . . .	1120



16.110.1 Detailed Description . . . . .	1123
16.110.2 Constructor & Destructor Documentation . . . . .	1123
16.110.2.1 Invalid_capability() . . . . .	1123
16.110.3 Member Function Documentation . . . . .	1123
16.110.3.1 cap() . . . . .	1123
16.111 L4::io_pager Class Reference . . . . .	1124
16.111.1 Detailed Description . . . . .	1125
16.111.2 Member Function Documentation . . . . .	1126
16.111.2.1 io_page_fault() . . . . .	1126
16.112 L4::lommu Class Reference . . . . .	1127
16.112.1 Detailed Description . . . . .	1129
16.112.2 Member Function Documentation . . . . .	1130
16.112.2.1 bind() . . . . .	1130
16.112.2.2 unbind() . . . . .	1131
16.113 L4::IOModifier Class Reference . . . . .	1132
16.113.1 Detailed Description . . . . .	1132
16.114 L4::lpc::Array< ELEM_TYPE, LEN_TYPE > Struct Template Reference . . . . .	1132
16.114.1 Detailed Description . . . . .	1134
16.115 L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX > Struct Template Reference . . . . .	1135
16.115.1 Detailed Description . . . . .	1136
16.116 L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE > Struct Template Reference . . . . .	1137
16.116.1 Detailed Description . . . . .	1138
16.117 L4::lpc::As_value< T > Struct Template Reference . . . . .	1138
16.117.1 Detailed Description . . . . .	1138
16.118 L4::lpc::Call Struct Reference . . . . .	1139
16.118.1 Detailed Description . . . . .	1140
16.119 L4::lpc::Call_t< RIGHTS > Struct Template Reference . . . . .	1140
16.119.1 Detailed Description . . . . .	1141
16.120 L4::lpc::Call_zero_send_timeout Struct Reference . . . . .	1141
16.120.1 Detailed Description . . . . .	1142
16.121 L4::lpc::Cap< T > Class Template Reference . . . . .	1143
16.121.1 Detailed Description . . . . .	1145
16.121.2 Member Enumeration Documentation . . . . .	1145
16.121.2.1 anonymous enum . . . . .	1145
16.121.3 Constructor & Destructor Documentation . . . . .	1145
16.121.3.1 Cap() . . . . .	1145
16.121.4 Member Function Documentation . . . . .	1146
16.121.4.1 from_ci() . . . . .	1146
16.122 L4::lpc::Gen_fpage Class Reference . . . . .	1146
16.122.1 Detailed Description . . . . .	1148
16.122.2 Member Enumeration Documentation . . . . .	1148
16.122.2.1 Type . . . . .	1148

16.123 L4::ipc::In_out< T > Struct Template Reference . . . . .	1149
16.123.1 Detailed Description . . . . .	1149
16.124 L4::ipc::lostream Class Reference . . . . .	1150
16.124.1 Detailed Description . . . . .	1153
16.124.2 Constructor & Destructor Documentation . . . . .	1154
16.124.2.1 lostream() . . . . .	1154
16.124.3 Member Function Documentation . . . . .	1155
16.124.3.1 call() . . . . .	1155
16.124.3.2 get() [1/3] . . . . .	1156
16.124.3.3 get() [2/3] . . . . .	1156
16.124.3.4 get() [3/3] . . . . .	1157
16.124.3.5 put() [1/2] . . . . .	1157
16.124.3.6 put() [2/2] . . . . .	1158
16.124.3.7 reply_and_wait() [1/2] . . . . .	1158
16.124.3.8 reply_and_wait() [2/2] . . . . .	1159
16.124.3.9 reset() . . . . .	1160
16.125 L4::ipc::Istream Class Reference . . . . .	1161
16.125.1 Detailed Description . . . . .	1163
16.125.2 Constructor & Destructor Documentation . . . . .	1163
16.125.2.1 Istream() . . . . .	1163
16.125.3 Member Function Documentation . . . . .	1164
16.125.3.1 get() [1/3] . . . . .	1164
16.125.3.2 get() [2/3] . . . . .	1165
16.125.3.3 get() [3/3] . . . . .	1166
16.125.3.4 receive() . . . . .	1167
16.125.3.5 reset() . . . . .	1168
16.125.3.6 skip() . . . . .	1169
16.125.3.7 tag() [1/2] . . . . .	1170
16.125.3.8 tag() [2/2] . . . . .	1170
16.125.3.9 wait() [1/2] . . . . .	1171
16.125.3.10 wait() [2/2] . . . . .	1172
16.126 L4::ipc::Msg::Cls_buffer Struct Reference . . . . .	1173
16.126.1 Detailed Description . . . . .	1174
16.127 L4::ipc::Msg::Cls_data Struct Reference . . . . .	1174
16.127.1 Detailed Description . . . . .	1175
16.128 L4::ipc::Msg::Cls_item Struct Reference . . . . .	1175
16.128.1 Detailed Description . . . . .	1176
16.129 L4::ipc::Msg::Dir_in Struct Reference . . . . .	1176
16.129.1 Detailed Description . . . . .	1177
16.130 L4::ipc::Msg::Dir_out Struct Reference . . . . .	1177
16.130.1 Detailed Description . . . . .	1178
16.131 L4::ipc::Msg::Do_in_data Struct Reference . . . . .	1178

16.131.1 Detailed Description . . . . .	1179
16.132 L4::lpc::Msg::Do_in_items Struct Reference . . . . .	1179
16.132.1 Detailed Description . . . . .	1180
16.133 L4::lpc::Msg::Do_out_data Struct Reference . . . . .	1180
16.133.1 Detailed Description . . . . .	1181
16.134 L4::lpc::Msg::Do_out_items Struct Reference . . . . .	1181
16.134.1 Detailed Description . . . . .	1182
16.135 L4::lpc::Msg::Do_rcv_buffers Struct Reference . . . . .	1182
16.135.1 Detailed Description . . . . .	1183
16.136 L4::lpc::Msg::Elem< Array< A, LEN > & > Struct Template Reference . . . . .	1184
16.136.1 Detailed Description . . . . .	1184
16.137 L4::lpc::Msg::Elem< Array< A, LEN > > Struct Template Reference . . . . .	1185
16.137.1 Detailed Description . . . . .	1185
16.138 L4::lpc::Msg::Elem< Array_ref< A, LEN > & > Struct Template Reference . . . . .	1186
16.138.1 Detailed Description . . . . .	1186
16.139 L4::lpc::Msg::False Struct Reference . . . . .	1187
16.139.1 Detailed Description . . . . .	1188
16.140 L4::lpc::Msg::Is_valid_rpc_type< T > Struct Template Reference . . . . .	1189
16.140.1 Detailed Description . . . . .	1190
16.141 L4::lpc::Msg::Svr_arg_pack< IPC_TYPE > Struct Template Reference . . . . .	1191
16.141.1 Detailed Description . . . . .	1191
16.142 L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS > Struct Template Reference . . . . .	1191
16.142.1 Detailed Description . . . . .	1192
16.143 L4::lpc::Msg::True Struct Reference . . . . .	1193
16.143.1 Detailed Description . . . . .	1195
16.144 L4::lpc::Msg_ptr< T > Class Template Reference . . . . .	1196
16.144.1 Detailed Description . . . . .	1196
16.144.2 Constructor & Destructor Documentation . . . . .	1196
16.144.2.1 Msg_ptr() . . . . .	1196
16.145 L4::lpc::Opt< T > Struct Template Reference . . . . .	1197
16.145.1 Detailed Description . . . . .	1199
16.146 L4::lpc::Ostream Class Reference . . . . .	1199
16.146.1 Detailed Description . . . . .	1202
16.146.2 Member Function Documentation . . . . .	1202
16.146.2.1 put() [1/2] . . . . .	1202
16.146.2.2 put() [2/2] . . . . .	1203
16.146.2.3 send() . . . . .	1203
16.146.2.4 tag() [1/2] . . . . .	1204
16.146.2.5 tag() [2/2] . . . . .	1204
16.147 L4::lpc::Out< T > Struct Template Reference . . . . .	1205
16.147.1 Detailed Description . . . . .	1205
16.148 L4::lpc::Rcv_fpage Class Reference . . . . .	1206

16.148.1 Detailed Description . . . . .	1208
16.148.2 Constructor & Destructor Documentation . . . . .	1208
16.148.2.1 Rcv_fpage() . . . . .	1208
16.148.3 Member Function Documentation . . . . .	1209
16.148.3.1 io() . . . . .	1209
16.148.3.2 mem() . . . . .	1210
16.148.3.3 obj() . . . . .	1210
16.148.3.4 rcv_task() . . . . .	1211
16.149 L4::lpc::Ret_array< T > Struct Template Reference . . . . .	1212
16.149.1 Detailed Description . . . . .	1212
16.150 L4::lpc::Send_only Struct Reference . . . . .	1213
16.150.1 Detailed Description . . . . .	1213
16.151 L4::lpc::Small_buf Class Reference . . . . .	1214
16.151.1 Detailed Description . . . . .	1214
16.151.2 Constructor & Destructor Documentation . . . . .	1214
16.151.2.1 Small_buf() [1/2] . . . . .	1214
16.151.2.2 Small_buf() [2/2] . . . . .	1215
16.152 L4::lpc::Snd_fpage Class Reference . . . . .	1215
16.152.1 Detailed Description . . . . .	1219
16.152.2 Member Enumeration Documentation . . . . .	1219
16.152.2.1 Cacheopt . . . . .	1219
16.152.2.2 Continue . . . . .	1219
16.152.2.3 Map_type . . . . .	1220
16.152.3 Constructor & Destructor Documentation . . . . .	1220
16.152.3.1 Snd_fpage() [1/2] . . . . .	1220
16.152.3.2 Snd_fpage() [2/2] . . . . .	1221
16.152.4 Member Function Documentation . . . . .	1222
16.152.4.1 cap_received() . . . . .	1222
16.152.4.2 id_received() . . . . .	1222
16.152.4.3 io() . . . . .	1222
16.152.4.4 is_compound() . . . . .	1223
16.152.4.5 local_id_received() . . . . .	1223
16.152.4.6 mem() . . . . .	1224
16.152.4.7 obj() . . . . .	1225
16.153 L4::lpc::Str_cp_in< T > Class Template Reference . . . . .	1226
16.153.1 Detailed Description . . . . .	1226
16.153.2 Constructor & Destructor Documentation . . . . .	1226
16.153.2.1 Str_cp_in() . . . . .	1226
16.154 L4::lpc::Varg Class Reference . . . . .	1227
16.154.1 Detailed Description . . . . .	1228
16.154.2 Member Function Documentation . . . . .	1228
16.154.2.1 data() . . . . .	1228

16.154.2.2 <a href="#">get_value()</a>	1229
16.154.2.3 <a href="#">is_nil()</a>	1229
16.154.2.4 <a href="#">is_of()</a>	1230
16.154.2.5 <a href="#">is_of_int()</a>	1231
16.154.2.6 <a href="#">length()</a>	1232
16.154.2.7 <a href="#">tag()</a>	1232
16.154.2.8 <a href="#">type()</a>	1232
16.154.2.9 <a href="#">value()</a>	1233
16.155 <a href="#">L4::lpc::Varg_list&lt; MAX &gt; Class Template Reference</a>	1233
16.155.1 Detailed Description	1235
16.156 <a href="#">L4::lpc::Varg_list_ref Class Reference</a>	1235
16.156.1 Detailed Description	1237
16.156.2 Constructor & Destructor Documentation	1237
16.156.2.1 <a href="#">Varg_list_ref()</a>	1237
16.157 <a href="#">L4::lpc::Varg_list_ref::literator Class Reference</a>	1238
16.157.1 Detailed Description	1239
16.158 <a href="#">L4::lpc_gate Class Reference</a>	1239
16.158.1 Detailed Description	1243
16.158.2 Member Function Documentation	1244
16.158.2.1 <a href="#">get_infos()</a>	1244
16.159 <a href="#">L4::lpc_svr::Br_manager_no_buffers Class Reference</a>	1245
16.159.1 Detailed Description	1248
16.159.2 Member Function Documentation	1248
16.159.2.1 <a href="#">alloc_buffer_demand()</a>	1248
16.160 <a href="#">L4::lpc_svr::Compound_reply Struct Reference</a>	1249
16.160.1 Detailed Description	1251
16.161 <a href="#">L4::lpc_svr::Dbg_dispatch&lt; R, Exc, Printer &gt; Struct Template Reference</a>	1251
16.161.1 Detailed Description	1252
16.162 <a href="#">L4::lpc_svr::Default_loop_hooks Struct Reference</a>	1253
16.162.1 Detailed Description	1255
16.163 <a href="#">L4::lpc_svr::Default_setup_wait Struct Reference</a>	1255
16.163.1 Detailed Description	1256
16.164 <a href="#">L4::lpc_svr::Default_timeout Struct Reference</a>	1256
16.164.1 Detailed Description	1258
16.165 <a href="#">L4::lpc_svr::Direct_dispatch&lt; R &gt; Struct Template Reference</a>	1258
16.165.1 Detailed Description	1259
16.166 <a href="#">L4::lpc_svr::Direct_dispatch&lt; R * &gt; Struct Template Reference</a>	1260
16.166.1 Detailed Description	1261
16.167 <a href="#">L4::lpc_svr::Exc_dispatch&lt; R, Exc &gt; Struct Template Reference</a>	1262
16.167.1 Detailed Description	1263
16.168 <a href="#">L4::lpc_svr::Ignore_errors Struct Reference</a>	1263
16.168.1 Detailed Description	1265

16.169 L4::lpc_svr::Server_iface Class Reference . . . . .	1265
16.169.1 Detailed Description . . . . .	1268
16.169.2 Member Function Documentation . . . . .	1268
16.169.2.1 add_timeout() . . . . .	1268
16.169.2.2 alloc_buffer_demand() . . . . .	1268
16.169.2.3 get_rcv_cap() . . . . .	1269
16.169.2.4 rcv_cap() [1/2] . . . . .	1270
16.169.2.5 rcv_cap() [2/2] . . . . .	1271
16.169.2.6 realloc_rcv_cap() . . . . .	1272
16.169.2.7 remove_timeout() . . . . .	1272
16.170 L4::lpc_svr::Timeout Class Reference . . . . .	1273
16.170.1 Detailed Description . . . . .	1276
16.170.2 Member Function Documentation . . . . .	1276
16.170.2.1 expired() . . . . .	1276
16.170.2.2 timeout() . . . . .	1276
16.171 L4::lpc_svr::Timeout_queue Class Reference . . . . .	1277
16.171.1 Detailed Description . . . . .	1277
16.171.2 Member Function Documentation . . . . .	1278
16.171.2.1 add() . . . . .	1278
16.171.2.2 handle_expired_timeouts() . . . . .	1278
16.171.2.3 next_timeout() . . . . .	1279
16.171.2.4 remove() . . . . .	1279
16.171.2.5 timeout_expired() . . . . .	1279
16.172 L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN > Class Template Reference . . . . .	1280
16.172.1 Detailed Description . . . . .	1283
16.172.2 Member Function Documentation . . . . .	1284
16.172.2.1 add_timeout() . . . . .	1284
16.172.2.2 remove_timeout() . . . . .	1285
16.173 L4::lpc_svr::Irq Class Reference . . . . .	1286
16.173.1 Detailed Description . . . . .	1291
16.173.2 Member Function Documentation . . . . .	1291
16.173.2.1 bind_vcpu() . . . . .	1291
16.173.2.2 detach() . . . . .	1292
16.173.2.3 receive() . . . . .	1293
16.173.2.4 unmask() . . . . .	1294
16.173.2.5 wait() . . . . .	1295
16.174 L4::lpc_svr::Irq_eoi Class Reference . . . . .	1296
16.174.1 Detailed Description . . . . .	1297
16.174.2 Member Function Documentation . . . . .	1297
16.174.2.1 unmask() . . . . .	1297
16.175 L4::lpc_svr::Irq_handler_object Struct Reference . . . . .	1298
16.175.1 Detailed Description . . . . .	1301

16.176 L4::lrqep_t< Derived, BASE, bool > Struct Template Reference . . . . .	1302
16.176.1 Detailed Description . . . . .	1304
16.176.2 Member Function Documentation . . . . .	1305
16.176.2.1 dispatch() . . . . .	1305
16.176.2.2 obj_cap() . . . . .	1306
16.177 L4::Kip::Mem_desc Class Reference . . . . .	1306
16.177.1 Detailed Description . . . . .	1308
16.177.2 Member Enumeration Documentation . . . . .	1308
16.177.2.1 Arch_sub_type_common . . . . .	1308
16.177.2.2 Info_sub_type . . . . .	1309
16.177.2.3 Mem_type . . . . .	1309
16.177.3 Constructor & Destructor Documentation . . . . .	1310
16.177.3.1 Mem_desc() . . . . .	1310
16.177.4 Member Function Documentation . . . . .	1311
16.177.4.1 all() [1/2] . . . . .	1311
16.177.4.2 all() [2/2] . . . . .	1312
16.177.4.3 count() [1/2] . . . . .	1312
16.177.4.4 count() [2/2] . . . . .	1313
16.177.4.5 end() . . . . .	1314
16.177.4.6 first() . . . . .	1314
16.177.4.7 is_virtual() . . . . .	1315
16.177.4.8 set() . . . . .	1315
16.177.4.9 size() . . . . .	1316
16.177.4.10 start() . . . . .	1317
16.177.4.11 sub_type() . . . . .	1317
16.177.4.12 type() . . . . .	1317
16.178 L4::Kobject Class Reference . . . . .	1318
16.178.1 Detailed Description . . . . .	1318
16.178.2 Member Function Documentation . . . . .	1319
16.178.2.1 cap() . . . . .	1319
16.178.2.2 dec_refcnt() . . . . .	1320
16.179 L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND > Class Template Reference . . . . .	1322
16.179.1 Detailed Description . . . . .	1324
16.179.2 Member Typedef Documentation . . . . .	1324
16.179.2.1 __iface . . . . .	1324
16.179.2.2 __iface_list . . . . .	1325
16.179.2.3 Class . . . . .	1325
16.179.3 Member Function Documentation . . . . .	1325
16.179.3.1 __check_protocols__() . . . . .	1325
16.179.3.2 c() . . . . .	1325
16.180 L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND > Struct Template Reference . . . . .	1326
16.180.1 Detailed Description . . . . .	1327

16.180.2 Member Typedef Documentation . . . . .	1328
16.180.2.1 __lface . . . . .	1328
16.180.2.2 __lface_list . . . . .	1328
16.180.2.3 Class . . . . .	1328
16.180.3 Member Function Documentation . . . . .	1329
16.180.3.1 __check_protocols__() . . . . .	1329
16.180.3.2 c() . . . . .	1329
16.181 L4::Kobject_demand< T > Struct Template Reference . . . . .	1329
16.181.1 Detailed Description . . . . .	1329
16.182 L4::Kobject_t< Derived, Base, PROTO, S_DEMAND > Class Template Reference . . . . .	1330
16.182.1 Detailed Description . . . . .	1331
16.183 L4::Kobject_typeid< T > Struct Template Reference . . . . .	1332
16.183.1 Detailed Description . . . . .	1333
16.183.2 Member Typedef Documentation . . . . .	1333
16.183.2.1 Demand . . . . .	1333
16.183.3 Member Function Documentation . . . . .	1333
16.183.3.1 demand() . . . . .	1333
16.183.3.2 id() . . . . .	1334
16.183.3.3 proto_dispatch() . . . . .	1334
16.184 L4::Kobject_typeid< void > Struct Reference . . . . .	1335
16.184.1 Detailed Description . . . . .	1336
16.184.2 Member Function Documentation . . . . .	1336
16.184.2.1 demand() . . . . .	1336
16.184.2.2 id() . . . . .	1337
16.184.2.3 proto_dispatch() . . . . .	1337
16.185 L4::Kobject_x< Derived, ARGS > Struct Template Reference . . . . .	1338
16.185.1 Detailed Description . . . . .	1339
16.186 L4::Lock_guard Class Reference . . . . .	1339
16.186.1 Detailed Description . . . . .	1340
16.186.2 Constructor & Destructor Documentation . . . . .	1340
16.186.2.1 Lock_guard() [1/2] . . . . .	1340
16.186.2.2 Lock_guard() [2/2] . . . . .	1340
16.186.2.3 ~Lock_guard() . . . . .	1341
16.186.3 Member Function Documentation . . . . .	1341
16.186.3.1 operator=() . . . . .	1341
16.186.3.2 status() . . . . .	1341
16.187 L4::Meta Class Reference . . . . .	1342
16.187.1 Detailed Description . . . . .	1344
16.187.2 Member Function Documentation . . . . .	1344
16.187.2.1 interface() . . . . .	1344
16.187.2.2 num_interfaces() . . . . .	1345
16.187.2.3 supports() . . . . .	1346



16.188 L4::Out_of_memory Class Reference . . . . .	1347
16.188.1 Detailed Description . . . . .	1350
16.189 L4::Pager Class Reference . . . . .	1351
16.189.1 Detailed Description . . . . .	1353
16.189.2 Member Function Documentation . . . . .	1353
16.189.2.1 page_fault() . . . . .	1353
16.190 L4::Platform_control Class Reference . . . . .	1354
16.190.1 Detailed Description . . . . .	1357
16.190.2 Member Function Documentation . . . . .	1357
16.190.2.1 cpu_allow_shutdown() . . . . .	1357
16.190.2.2 cpu_disable() . . . . .	1358
16.190.2.3 cpu_enable() . . . . .	1359
16.190.2.4 system_shutdown() . . . . .	1360
16.190.2.5 system_suspend() . . . . .	1361
16.191 L4::Poll_timeout_counter Class Reference . . . . .	1362
16.191.1 Detailed Description . . . . .	1363
16.191.2 Constructor & Destructor Documentation . . . . .	1363
16.191.2.1 Poll_timeout_counter() . . . . .	1363
16.191.3 Member Function Documentation . . . . .	1363
16.191.3.1 set() . . . . .	1363
16.191.3.2 timed_out() . . . . .	1364
16.192 L4::Poll_timeout_kipclock Class Reference . . . . .	1364
16.192.1 Detailed Description . . . . .	1365
16.192.2 Constructor & Destructor Documentation . . . . .	1365
16.192.2.1 Poll_timeout_kipclock() . . . . .	1365
16.192.3 Member Function Documentation . . . . .	1366
16.192.3.1 set() . . . . .	1366
16.192.3.2 test() . . . . .	1366
16.192.3.3 timed_out() . . . . .	1367
16.193 L4::Proto_t< P > Struct Template Reference . . . . .	1368
16.193.1 Detailed Description . . . . .	1368
16.194 L4::Rcv_endpoint Class Reference . . . . .	1368
16.194.1 Detailed Description . . . . .	1371
16.194.2 Member Function Documentation . . . . .	1371
16.194.2.1 bind_snd_destination() . . . . .	1371
16.194.2.2 bind_thread() . . . . .	1372
16.195 L4::Registry_iface Class Reference . . . . .	1374
16.195.1 Detailed Description . . . . .	1376
16.195.2 Member Function Documentation . . . . .	1376
16.195.2.1 register_irq_obj() . . . . .	1376
16.195.2.2 register_obj() [1/3] . . . . .	1376
16.195.2.3 register_obj() [2/3] . . . . .	1377

16.195.2.4 register_obj() [3/3]	1377
16.195.2.5 unregister_obj()	1378
16.196 L4::Runtime_error Class Reference	1378
16.196.1 Detailed Description	1381
16.196.2 Constructor & Destructor Documentation	1381
16.196.2.1 Runtime_error()	1381
16.196.3 Member Function Documentation	1382
16.196.3.1 err_no()	1382
16.196.3.2 extra_str()	1383
16.197 L4::Scheduler Class Reference	1383
16.197.1 Detailed Description	1387
16.197.2 Member Function Documentation	1388
16.197.2.1 idle_time()	1388
16.197.2.2 info()	1389
16.197.2.3 is_online()	1390
16.197.2.4 run_thread()	1391
16.198 L4::Semaphore Struct Reference	1392
16.198.1 Detailed Description	1396
16.198.2 Member Function Documentation	1396
16.198.2.1 down()	1396
16.198.2.2 up()	1397
16.199 L4::Server< LOOP_HOOKS > Class Template Reference	1398
16.199.1 Detailed Description	1401
16.199.2 Constructor & Destructor Documentation	1402
16.199.2.1 Server()	1402
16.199.3 Member Function Documentation	1402
16.199.3.1 internal_loop()	1402
16.199.3.2 loop()	1403
16.199.3.3 loop_dbg()	1404
16.200 L4::Server_object Class Reference	1404
16.200.1 Detailed Description	1407
16.200.2 Member Function Documentation	1407
16.200.2.1 dispatch() [1/2]	1407
16.200.2.2 dispatch() [2/2]	1408
16.201 L4::Server_object_t< IFACE, BASE > Struct Template Reference	1409
16.201.1 Detailed Description	1412
16.201.2 Member Function Documentation	1413
16.201.2.1 dispatch_meta_request()	1413
16.201.2.2 get_buffer_demand()	1413
16.201.2.3 proto_dispatch()	1413
16.202 L4::Server_object_x< Derived, IFACE, BASE > Struct Template Reference	1415
16.202.1 Detailed Description	1417

16.203 L4::Smart_cap< T, SMART > Class Template Reference . . . . .	1418
16.203.1 Detailed Description . . . . .	1422
16.203.2 Constructor & Destructor Documentation . . . . .	1422
16.203.2.1 Smart_cap() . . . . .	1422
16.204 L4::String Class Reference . . . . .	1422
16.204.1 Detailed Description . . . . .	1423
16.205 L4::Task Class Reference . . . . .	1423
16.205.1 Detailed Description . . . . .	1427
16.205.2 Member Function Documentation . . . . .	1427
16.205.2.1 add_ku_mem() . . . . .	1427
16.205.2.2 cap_equal() . . . . .	1428
16.205.2.3 cap_valid() . . . . .	1429
16.205.2.4 delete_obj() . . . . .	1430
16.205.2.5 map() . . . . .	1431
16.205.2.6 release_cap() . . . . .	1432
16.205.2.7 unmap() . . . . .	1433
16.205.2.8 unmap_batch() . . . . .	1434
16.206 L4::Thread Class Reference . . . . .	1435
16.206.1 Detailed Description . . . . .	1439
16.206.2 Member Function Documentation . . . . .	1439
16.206.2.1 control() . . . . .	1439
16.206.2.2 ex_regs() [1/2] . . . . .	1440
16.206.2.3 ex_regs() [2/2] . . . . .	1441
16.206.2.4 modify_senders() . . . . .	1442
16.206.2.5 register_del_irq() . . . . .	1443
16.206.2.6 register_doorbell_irq() . . . . .	1444
16.206.2.7 stats_time() . . . . .	1445
16.206.2.8 switch_to() . . . . .	1446
16.206.2.9 vcpu_control() . . . . .	1447
16.206.2.10 vcpu_control_ext() . . . . .	1447
16.206.2.11 vcpu_resume_commit() . . . . .	1448
16.206.2.12 vcpu_resume_start() . . . . .	1450
16.207 L4::Thread::Attr Class Reference . . . . .	1451
16.207.1 Detailed Description . . . . .	1452
16.207.2 Constructor & Destructor Documentation . . . . .	1452
16.207.2.1 Attr() . . . . .	1452
16.207.3 Member Function Documentation . . . . .	1452
16.207.3.1 alien() . . . . .	1452
16.207.3.2 bind() . . . . .	1453
16.207.3.3 exc_handler() [1/2] . . . . .	1453
16.207.3.4 exc_handler() [2/2] . . . . .	1453
16.207.3.5 pager() [1/2] . . . . .	1454

16.207.3.6 pager() [ 2 / 2 ]	1454
16.208 L4::Thread::Modify_senders Class Reference	1455
16.208.1 Detailed Description	1456
16.208.2 Member Function Documentation	1456
16.208.2.1 add()	1456
16.209 L4::Thread_group Class Reference	1458
16.209.1 Detailed Description	1460
16.209.2 Member Function Documentation	1460
16.209.2.1 add()	1460
16.209.2.2 remove()	1461
16.210 L4::Triggerable Struct Reference	1462
16.210.1 Detailed Description	1465
16.210.2 Member Function Documentation	1465
16.210.2.1 trigger()	1465
16.211 L4::Type_info Struct Reference	1466
16.211.1 Detailed Description	1467
16.212 L4::Type_info::Demand Class Reference	1467
16.212.1 Detailed Description	1469
16.212.2 Constructor & Destructor Documentation	1470
16.212.2.1 Demand()	1470
16.212.3 Member Function Documentation	1470
16.212.3.1 no_demand()	1470
16.213 L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS > Struct Template Reference	1471
16.213.1 Detailed Description	1474
16.213.2 Member Enumeration Documentation	1474
16.213.2.1 anonymous enum	1474
16.214 L4::Type_info::Demand_union_t< D1, D2 > Struct Template Reference	1474
16.214.1 Detailed Description	1477
16.215 L4::Typeid::Detail::_Rpc< OPCODE, O, X > Struct Template Reference	1478
16.215.1 Detailed Description	1478
16.216 L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::Rpc< Y > Struct Template Reference	1479
16.216.1 Detailed Description	1479
16.217 L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... > Struct Template Reference	1479
16.217.1 Detailed Description	1482
16.218 L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >::Rpc< Y > Struct Template Reference	1482
16.218.1 Detailed Description	1482
16.219 L4::Typeid::Detail::Rpc_end Struct Reference	1483
16.219.1 Detailed Description	1483
16.220 L4::Typeid::P_dispatch< LIST > Struct Template Reference	1483
16.220.1 Detailed Description	1484
16.221 L4::Typeid::Raw_ipc< CLASS > Struct Template Reference	1484
16.221.1 Detailed Description	1485

16.222 L4::Typeid::Rpc_nocode< OPERATION > Struct Template Reference . . . . .	1485
16.222.1 Detailed Description . . . . .	1487
16.223 L4::Typeid::Rpc< RPCS > Struct Template Reference . . . . .	1488
16.223.1 Detailed Description . . . . .	1489
16.224 L4::Typeid::Rpc_code< OPCODE_TYPE > Struct Template Reference . . . . .	1489
16.224.1 Detailed Description . . . . .	1490
16.225 L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS > Struct Template Reference . . . . .	1490
16.225.1 Detailed Description . . . . .	1492
16.226 L4::Typeid::Rpc_sys< ARG > Struct Template Reference . . . . .	1493
16.226.1 Detailed Description . . . . .	1494
16.227 L4::Types::Bool< V > Struct Template Reference . . . . .	1495
16.227.1 Detailed Description . . . . .	1496
16.228 L4::Types::False Struct Reference . . . . .	1497
16.228.1 Detailed Description . . . . .	1498
16.229 L4::Types::Flags< BITS_ENUM, UNDERLYING > Class Template Reference . . . . .	1499
16.229.1 Detailed Description . . . . .	1501
16.229.2 Member Enumeration Documentation . . . . .	1502
16.229.2.1 None_type . . . . .	1502
16.229.3 Constructor & Destructor Documentation . . . . .	1502
16.229.3.1 Flags() [1/2] . . . . .	1502
16.229.3.2 Flags() [2/2] . . . . .	1502
16.229.4 Member Function Documentation . . . . .	1503
16.229.4.1 clear() . . . . .	1503
16.229.4.2 from_raw() . . . . .	1503
16.230 L4::Types::Flags_ops_t< DT > Struct Template Reference . . . . .	1503
16.230.1 Detailed Description . . . . .	1505
16.231 L4::Types::Flags_t< DT, T > Struct Template Reference . . . . .	1506
16.231.1 Detailed Description . . . . .	1508
16.232 L4::Types::Int_for_size< SIZE, bool > Struct Template Reference . . . . .	1508
16.232.1 Detailed Description . . . . .	1509
16.233 L4::Types::Int_for_type< T > Struct Template Reference . . . . .	1509
16.233.1 Detailed Description . . . . .	1510
16.234 L4::Types::Same< A, B > Struct Template Reference . . . . .	1510
16.234.1 Detailed Description . . . . .	1512
16.235 L4::Types::True Struct Reference . . . . .	1513
16.235.1 Detailed Description . . . . .	1515
16.236 L4::Uart Class Reference . . . . .	1516
16.236.1 Detailed Description . . . . .	1518
16.236.2 Member Function Documentation . . . . .	1518
16.236.2.1 change_mode() . . . . .	1518
16.236.2.2 char_avail() . . . . .	1518
16.236.2.3 enable_rx_irq() . . . . .	1518

16.236.2.4 generic_write()	1519
16.236.2.5 get_char()	1519
16.236.2.6 mode()	1520
16.236.2.7 rate()	1520
16.236.2.8 shutdown()	1520
16.236.2.9 startup()	1520
16.236.2.10 write()	1521
16.237 L4::Uart_apb Class Reference	1521
16.237.1 Detailed Description	1524
16.237.2 Member Function Documentation	1524
16.237.2.1 change_mode()	1524
16.237.2.2 char_avail()	1525
16.237.2.3 enable_rx_irq()	1525
16.237.2.4 get_char()	1525
16.237.2.5 shutdown()	1526
16.237.2.6 startup()	1526
16.237.2.7 write()	1526
16.238 L4::Unknown_error Class Reference	1527
16.238.1 Detailed Description	1529
16.239 L4::Vcon Class Reference	1529
16.239.1 Detailed Description	1533
16.239.2 Member Function Documentation	1534
16.239.2.1 get_attr()	1534
16.239.2.2 read()	1535
16.239.2.3 read_with_flags()	1536
16.239.2.4 send()	1537
16.239.2.5 set_attr()	1538
16.239.2.6 write()	1539
16.240 L4::Vm Class Reference	1539
16.240.1 Detailed Description	1543
16.240.2 Member Function Documentation	1543
16.240.2.1 vgicc_map()	1543
16.241 l4_buf_regs_t Struct Reference	1544
16.241.1 Detailed Description	1545
16.242 l4_exc_regs_t Struct Reference	1545
16.242.1 Detailed Description	1547
16.242.2 Field Documentation	1547
16.242.2.1 flags	1547
16.242.2.2 ss	1548
16.243 l4_fpage_t Union Reference	1548
16.243.1 Detailed Description	1548
16.244 l4_icu_info_t Struct Reference	1549

16.244.1 Detailed Description . . . . .	1550
16.244.2 Field Documentation . . . . .	1550
16.244.2.1 features . . . . .	1550
16.245 l4_icu_msi_info_t Struct Reference . . . . .	1550
16.245.1 Detailed Description . . . . .	1551
16.246 l4_kernel_info_mem_desc_t Struct Reference . . . . .	1551
16.246.1 Detailed Description . . . . .	1551
16.247 l4_kernel_info_t Struct Reference . . . . .	1552
16.247.1 Detailed Description . . . . .	1553
16.248 l4_msg_regs_t Union Reference . . . . .	1553
16.248.1 Detailed Description . . . . .	1554
16.249 l4_msgtag_t Struct Reference . . . . .	1554
16.249.1 Detailed Description . . . . .	1555
16.249.2 Member Function Documentation . . . . .	1556
16.249.2.1 flags() . . . . .	1556
16.249.2.2 has_error() . . . . .	1556
16.250 l4_sched_cpu_set_t Struct Reference . . . . .	1557
16.250.1 Detailed Description . . . . .	1557
16.250.2 Member Function Documentation . . . . .	1558
16.250.2.1 granularity() . . . . .	1558
16.250.2.2 offset() . . . . .	1558
16.250.2.3 set() . . . . .	1559
16.250.3 Field Documentation . . . . .	1560
16.250.3.1 gran_offset . . . . .	1560
16.251 l4_sched_param_t Struct Reference . . . . .	1561
16.251.1 Detailed Description . . . . .	1561
16.251.2 Field Documentation . . . . .	1562
16.251.2.1 prio . . . . .	1562
16.252 l4_snd_fpage_t Struct Reference . . . . .	1562
16.252.1 Detailed Description . . . . .	1563
16.253 l4_thread_regs_t Struct Reference . . . . .	1563
16.253.1 Detailed Description . . . . .	1563
16.253.2 Field Documentation . . . . .	1564
16.253.2.1 error . . . . .	1564
16.253.2.2 free_marker . . . . .	1564
16.254 l4_timeout_s Struct Reference . . . . .	1564
16.254.1 Detailed Description . . . . .	1565
16.255 l4_timeout_t Union Reference . . . . .	1565
16.255.1 Detailed Description . . . . .	1566
16.256 l4_vcon_attr_t Struct Reference . . . . .	1566
16.256.1 Detailed Description . . . . .	1567
16.256.2 Member Function Documentation . . . . .	1567

16.256.2.1 <a href="#">set_raw()</a>	1567
16.257 <a href="#">l4_vcpu_arch_state_t</a> Struct Reference	1568
16.257.1 Detailed Description	1568
16.258 <a href="#">l4_vcpu_ipc_regs_t</a> Struct Reference	1568
16.258.1 Detailed Description	1569
16.259 <a href="#">l4_vcpu_regs_t</a> Struct Reference	1569
16.259.1 Detailed Description	1571
16.259.2 Field Documentation	1571
16.259.2.1 <a href="#">ax</a>	1571
16.259.2.2 <a href="#">bp</a>	1572
16.259.2.3 <a href="#">bx</a>	1572
16.259.2.4 <a href="#">cx</a>	1572
16.259.2.5 <a href="#">di</a>	1572
16.259.2.6 <a href="#">dx</a>	1572
16.259.2.7 <a href="#">si</a>	1573
16.260 <a href="#">l4_vcpu_state_t</a> Struct Reference	1573
16.260.1 Detailed Description	1576
16.260.2 Field Documentation	1576
16.260.2.1 <a href="#">version</a>	1576
16.261 <a href="#">l4_vm_svm_vmcb_control_area</a> Struct Reference	1577
16.261.1 Detailed Description	1577
16.262 <a href="#">l4_vm_svm_vmcb_state_save_area</a> Struct Reference	1577
16.262.1 Detailed Description	1578
16.263 <a href="#">l4_vm_svm_vmcb_state_save_area_seg</a> Struct Reference	1578
16.263.1 Detailed Description	1579
16.264 <a href="#">l4_vm_svm_vmcb_t</a> Struct Reference	1579
16.264.1 Detailed Description	1580
16.265 <a href="#">l4_vm_tz_state</a> Struct Reference	1581
16.265.1 Detailed Description	1581
16.266 <a href="#">l4_vm_vmx_vcpu_infos_t</a> Struct Reference	1581
16.266.1 Detailed Description	1582
16.266.2 Field Documentation	1582
16.266.2.1 <a href="#">dfli</a>	1582
16.267 <a href="#">l4_vm_vmx_vcpu_state_t</a> Struct Reference	1582
16.267.1 Detailed Description	1583
16.268 <a href="#">l4_vm_vmx_vcpu_vmcs_t</a> Struct Reference	1584
16.268.1 Detailed Description	1584
16.269 <a href="#">l4_vmx_offset_table_t</a> Struct Reference	1585
16.269.1 Detailed Description	1586
16.270 <a href="#">L4drivers::Mmio_register_block&lt; MAX_BITS &gt;</a> Struct Template Reference	1587
16.270.1 Detailed Description	1588
16.271 <a href="#">L4drivers::Register_block&lt; MAX_BITS, BLOCK &gt;</a> Class Template Reference	1589



16.271.1 Detailed Description . . . . .	1589
16.271.2 Member Function Documentation . . . . .	1590
16.271.2.1 operator[]() [1/2] . . . . .	1590
16.271.2.2 operator[]() [2/2] . . . . .	1591
16.271.2.3 r() [1/2] . . . . .	1592
16.271.2.4 r() [2/2] . . . . .	1592
16.272 L4drivers::Register_block_base< MAX_BITS > Struct Template Reference . . . . .	1593
16.272.1 Detailed Description . . . . .	1594
16.273 L4drivers::Register_block_impl< BASE, MAX_BITS > Struct Template Reference . . . . .	1594
16.273.1 Detailed Description . . . . .	1596
16.274 L4drivers::Register_block_tmpl< BLOCK > Class Template Reference . . . . .	1596
16.274.1 Detailed Description . . . . .	1597
16.275 L4drivers::Register_tmpl< BITS, BLOCK > Class Template Reference . . . . .	1597
16.275.1 Detailed Description . . . . .	1600
16.275.2 Member Function Documentation . . . . .	1601
16.275.2.1 clear() . . . . .	1601
16.275.2.2 modify() . . . . .	1601
16.275.2.3 operator=() . . . . .	1602
16.275.2.4 set() . . . . .	1602
16.275.2.5 write() . . . . .	1603
16.276 L4drivers::Ro_register_block< MAX_BITS, BLOCK > Class Template Reference . . . . .	1604
16.276.1 Detailed Description . . . . .	1604
16.276.2 Member Function Documentation . . . . .	1605
16.276.2.1 operator[]() . . . . .	1605
16.276.2.2 r() . . . . .	1605
16.277 L4drivers::Ro_register_tmpl< BITS, BLOCK > Class Template Reference . . . . .	1606
16.277.1 Detailed Description . . . . .	1607
16.277.2 Member Function Documentation . . . . .	1607
16.277.2.1 operator value_type() . . . . .	1607
16.277.2.2 read() . . . . .	1607
16.278 L4Re::Cap_alloc Class Reference . . . . .	1608
16.278.1 Detailed Description . . . . .	1609
16.278.2 Member Function Documentation . . . . .	1609
16.278.2.1 alloc() [1/2] . . . . .	1609
16.278.2.2 alloc() [2/2] . . . . .	1610
16.278.2.3 free() . . . . .	1611
16.279 L4Re::Console Class Reference . . . . .	1611
16.279.1 Detailed Description . . . . .	1614
16.280 L4Re::Core::Ref_ptr< T, CNT > Class Template Reference . . . . .	1614
16.280.1 Detailed Description . . . . .	1615
16.280.2 Constructor & Destructor Documentation . . . . .	1616
16.280.2.1 Ref_ptr() [1/3] . . . . .	1616

16.280.2.2 Ref_ptr() [2/3]	1616
16.280.2.3 Ref_ptr() [3/3]	1616
16.280.3 Member Function Documentation	1617
16.280.3.1 get()	1617
16.280.3.2 ptr()	1617
16.280.3.3 release()	1617
16.281 L4Re::Dataspace Class Reference	1618
16.281.1 Detailed Description	1621
16.281.2 Member Function Documentation	1621
16.281.2.1 allocate()	1621
16.281.2.2 clear()	1622
16.281.2.3 copy_in()	1623
16.281.2.4 flags()	1624
16.281.2.5 info()	1625
16.281.2.6 map()	1626
16.281.2.7 map_info()	1628
16.281.2.8 map_region()	1629
16.281.2.9 size()	1630
16.282 L4Re::Dataspace::F Struct Reference	1631
16.282.1 Detailed Description	1632
16.282.2 Member Enumeration Documentation	1632
16.282.2.1 anonymous enum	1632
16.282.2.2 Flags	1632
16.283 L4Re::Dataspace::Stats Struct Reference	1633
16.283.1 Detailed Description	1633
16.284 L4Re::Debug_obj Class Reference	1634
16.284.1 Detailed Description	1636
16.284.2 Member Function Documentation	1636
16.284.2.1 debug()	1636
16.285 L4Re::Default_event_payload Struct Reference	1637
16.285.1 Detailed Description	1638
16.286 L4Re::Dma_space Class Reference	1638
16.286.1 Detailed Description	1640
16.286.2 Member Typedef Documentation	1640
16.286.2.1 Attributes	1640
16.286.3 Member Enumeration Documentation	1640
16.286.3.1 Attribute	1640
16.286.3.2 Direction	1641
16.286.3.3 Space_attrib	1641
16.286.4 Member Function Documentation	1641
16.286.4.1 associate()	1641
16.286.4.2 disassociate()	1642

16.286.4.3 map()	1643
16.286.4.4 unmap()	1644
16.287 L4Re::Env Class Reference	1645
16.287.1 Detailed Description	1648
16.287.2 Member Function Documentation	1648
16.287.2.1 dbg_events() [1/2]	1648
16.287.2.2 dbg_events() [2/2]	1649
16.287.2.3 env()	1650
16.287.2.4 factory() [1/2]	1651
16.287.2.5 factory() [2/2]	1651
16.287.2.6 first_free_cap() [1/2]	1651
16.287.2.7 first_free_cap() [2/2]	1651
16.287.2.8 first_free_utcb() [1/2]	1652
16.287.2.9 first_free_utcb() [2/2]	1652
16.287.2.10 get()	1652
16.287.2.11 get_cap() [1/2]	1653
16.287.2.12 get_cap() [2/2]	1654
16.287.2.13 initial_caps() [1/2]	1655
16.287.2.14 initial_caps() [2/2]	1655
16.287.2.15 itas() [1/2]	1655
16.287.2.16 itas() [2/2]	1655
16.287.2.17 log() [1/2]	1656
16.287.2.18 log() [2/2]	1656
16.287.2.19 main_thread() [1/2]	1656
16.287.2.20 main_thread() [2/2]	1656
16.287.2.21 mem_alloc() [1/2]	1657
16.287.2.22 mem_alloc() [2/2]	1657
16.287.2.23 parent() [1/2]	1657
16.287.2.24 parent() [2/2]	1657
16.287.2.25 rm() [1/2]	1658
16.287.2.26 rm() [2/2]	1658
16.287.2.27 scheduler() [1/2]	1658
16.287.2.28 scheduler() [2/2]	1658
16.287.2.29 task()	1659
16.287.2.30 utcb_area() [1/2]	1659
16.287.2.31 utcb_area() [2/2]	1659
16.288 L4Re::Event Class Reference	1660
16.288.1 Detailed Description	1663
16.288.2 Member Function Documentation	1663
16.288.2.1 get_axis_info()	1663
16.288.2.2 get_buffer()	1664
16.288.2.3 get_num_streams()	1665

16.288.2.4	get_stream_info()	1666
16.288.2.5	get_stream_info_for_id()	1667
16.288.2.6	get_stream_state_for_id()	1668
16.289	L4Re::Event_buffer_t< PAYLOAD > Class Template Reference	1669
16.289.1	Detailed Description	1670
16.289.2	Constructor & Destructor Documentation	1670
16.289.2.1	Event_buffer_t()	1670
16.289.3	Member Function Documentation	1671
16.289.3.1	next()	1671
16.289.3.2	put()	1671
16.290	L4Re::Event_buffer_t< PAYLOAD >::Event Struct Reference	1672
16.290.1	Detailed Description	1672
16.291	L4Re::Inhibitor Class Reference	1673
16.291.1	Detailed Description	1675
16.291.2	Member Enumeration Documentation	1675
16.291.2.1	anonymous enum	1675
16.291.3	Member Function Documentation	1676
16.291.3.1	acquire()	1676
16.291.3.2	next_lock_info()	1677
16.291.3.3	release()	1677
16.292	L4Re::Itas Class Reference	1678
16.292.1	Detailed Description	1682
16.292.2	Member Enumeration Documentation	1682
16.292.2.1	anonymous enum	1682
16.292.3	Member Function Documentation	1682
16.292.3.1	getitimer()	1682
16.292.3.2	raise()	1683
16.292.3.3	register_thread()	1684
16.292.3.4	setitimer()	1684
16.292.3.5	sigaction()	1685
16.292.3.6	sigaltstack()	1686
16.292.3.7	sigpending()	1686
16.292.3.8	sigprocmask()	1687
16.292.3.9	unregister_thread()	1688
16.293	L4Re::Log Class Reference	1689
16.293.1	Detailed Description	1694
16.293.2	Member Function Documentation	1694
16.293.2.1	print()	1694
16.293.2.2	println()	1694
16.294	L4Re::Mem_alloc Class Reference	1695
16.294.1	Detailed Description	1698
16.294.2	Member Enumeration Documentation	1699

16.294.2.1 Mem_alloc_flags . . . . .	1699
16.294.3 Member Function Documentation . . . . .	1699
16.294.3.1 alloc() . . . . .	1699
16.294.3.2 info() . . . . .	1700
16.295 L4Re::Mem_alloc::Stats Struct Reference . . . . .	1701
16.295.1 Detailed Description . . . . .	1702
16.295.2 Field Documentation . . . . .	1702
16.295.2.1 mem_free . . . . .	1702
16.295.2.2 mem_limit . . . . .	1702
16.295.2.3 mem_used . . . . .	1702
16.295.2.4 quota . . . . .	1703
16.295.2.5 quota_used . . . . .	1703
16.296 L4Re::Mmio_space Struct Reference . . . . .	1703
16.296.1 Detailed Description . . . . .	1706
16.296.2 Member Enumeration Documentation . . . . .	1707
16.296.2.1 Access_width . . . . .	1707
16.296.3 Member Function Documentation . . . . .	1708
16.296.3.1 mmio_read() . . . . .	1708
16.296.3.2 mmio_write() . . . . .	1709
16.297 L4Re::Namespace Class Reference . . . . .	1710
16.297.1 Detailed Description . . . . .	1714
16.297.2 Member Enumeration Documentation . . . . .	1714
16.297.2.1 Query_result_flags . . . . .	1714
16.297.2.2 Query_timeout . . . . .	1714
16.297.2.3 Register_flags . . . . .	1714
16.297.3 Member Function Documentation . . . . .	1715
16.297.3.1 query() [1/2] . . . . .	1715
16.297.3.2 query() [2/2] . . . . .	1716
16.297.3.3 register_obj() . . . . .	1717
16.297.3.4 unlink() . . . . .	1719
16.298 L4Re::Ned::Cmd_control Class Reference . . . . .	1720
16.298.1 Detailed Description . . . . .	1720
16.298.2 Member Function Documentation . . . . .	1720
16.298.2.1 execute() [1/2] . . . . .	1720
16.298.2.2 execute() [2/2] . . . . .	1721
16.299 L4Re::Parent Class Reference . . . . .	1722
16.299.1 Detailed Description . . . . .	1725
16.299.2 Member Function Documentation . . . . .	1725
16.299.2.1 signal() . . . . .	1725
16.300 L4Re::Random Struct Reference . . . . .	1727
16.300.1 Detailed Description . . . . .	1730
16.300.2 Member Function Documentation . . . . .	1730

16.300.2.1 <code>get_random()</code>	1730
16.301 L4Re::Rm Class Reference	1731
16.301.1 Detailed Description	1736
16.301.2 Member Enumeration Documentation	1736
16.301.2.1 <code>Detach_flags</code>	1736
16.301.2.2 <code>Detach_result</code>	1736
16.301.2.3 <code>Region_flag_shifts</code>	1737
16.301.3 Member Function Documentation	1737
16.301.3.1 <code>attach()</code> [1/2]	1737
16.301.3.2 <code>attach()</code> [2/2]	1739
16.301.3.3 <code>detach()</code> [1/3]	1740
16.301.3.4 <code>detach()</code> [2/3]	1742
16.301.3.5 <code>detach()</code> [3/3]	1743
16.301.3.6 <code>find()</code>	1744
16.301.3.7 <code>free_area()</code>	1746
16.301.3.8 <code>get_areas()</code>	1748
16.301.3.9 <code>get_info()</code>	1749
16.301.3.10 <code>get_regions()</code>	1750
16.301.3.11 <code>reserve_area()</code> [1/2]	1751
16.301.3.12 <code>reserve_area()</code> [2/2]	1754
16.302 L4Re::Rm::Area Struct Reference	1755
16.302.1 Detailed Description	1756
16.303 L4Re::Rm::F Struct Reference	1756
16.303.1 Detailed Description	1757
16.303.2 Member Enumeration Documentation	1757
16.303.2.1 <code>Attach_flags</code>	1757
16.303.2.2 <code>Region_flags</code>	1757
16.304 L4Re::Rm::Region Struct Reference	1758
16.304.1 Detailed Description	1759
16.305 L4Re::Rm::Unique_region< T > Class Template Reference	1759
16.305.1 Detailed Description	1761
16.305.2 Constructor & Destructor Documentation	1761
16.305.2.1 <code>Unique_region()</code> [1/3]	1761
16.305.2.2 <code>Unique_region()</code> [2/3]	1761
16.305.2.3 <code>Unique_region()</code> [3/3]	1762
16.305.2.4 <code>~Unique_region()</code>	1762
16.305.3 Member Function Documentation	1762
16.305.3.1 <code>get()</code>	1762
16.305.3.2 <code>is_valid()</code>	1763
16.305.3.3 <code>operator=()</code>	1763
16.305.3.4 <code>release()</code>	1764
16.305.3.5 <code>reset()</code>	1764

16.306 L4Re::Smart_cap_auto< Unmap_flags > Class Template Reference . . . . .	1764
16.306.1 Detailed Description . . . . .	1765
16.307 L4Re::Smart_count_cap< Unmap_flags > Class Template Reference . . . . .	1765
16.307.1 Detailed Description . . . . .	1766
16.308 L4Re::Util::_Cap_alloc Class Reference . . . . .	1766
16.308.1 Detailed Description . . . . .	1767
16.308.2 Member Function Documentation . . . . .	1768
16.308.2.1 alloc() [1/2] . . . . .	1768
16.308.2.2 alloc() [2/2] . . . . .	1768
16.308.2.3 free() . . . . .	1769
16.309 L4Re::Util::Bitmap< BITS > Class Template Reference . . . . .	1769
16.309.1 Detailed Description . . . . .	1772
16.309.2 Member Function Documentation . . . . .	1773
16.309.2.1 scan_zero() . . . . .	1773
16.310 L4Re::Util::Bitmap_base Class Reference . . . . .	1773
16.310.1 Detailed Description . . . . .	1776
16.310.2 Member Enumeration Documentation . . . . .	1776
16.310.2.1 anonymous enum . . . . .	1776
16.310.3 Member Function Documentation . . . . .	1777
16.310.3.1 atomic_clear_bit() . . . . .	1777
16.310.3.2 atomic_get_and_clear() . . . . .	1777
16.310.3.3 atomic_get_and_set() . . . . .	1777
16.310.3.4 atomic_set_bit() . . . . .	1778
16.310.3.5 bit() [1/2] . . . . .	1778
16.310.3.6 bit() [2/2] . . . . .	1778
16.310.3.7 bit_index() . . . . .	1779
16.310.3.8 clear_bit() . . . . .	1779
16.310.3.9 operator[]() [1/2] . . . . .	1779
16.310.3.10 operator[]() [2/2] . . . . .	1780
16.310.3.11 scan_zero() . . . . .	1780
16.310.3.12 set_bit() . . . . .	1781
16.310.3.13 word_index() . . . . .	1781
16.311 L4Re::Util::Bitmap_base::Bit Class Reference . . . . .	1782
16.311.1 Detailed Description . . . . .	1782
16.312 L4Re::Util::Bitmap_base::Char< BITS > Class Template Reference . . . . .	1782
16.312.1 Detailed Description . . . . .	1783
16.313 L4Re::Util::Bitmap_base::Word< BITS > Class Template Reference . . . . .	1783
16.313.1 Detailed Description . . . . .	1784
16.314 L4Re::Util::Br_manager Class Reference . . . . .	1784
16.314.1 Detailed Description . . . . .	1787
16.314.2 Member Function Documentation . . . . .	1787
16.314.2.1 alloc_buffer_demand() . . . . .	1787

16.314.2.2	<a href="#">get_rcv_cap()</a>	1788
16.314.2.3	<a href="#">realloc_rcv_cap()</a>	1788
16.314.2.4	<a href="#">set_rcv_cap_flags()</a>	1789
16.315	<a href="#">L4Re::Util::Br_manager_hooks Struct Reference</a>	1790
16.315.1	<a href="#">Detailed Description</a>	1792
16.316	<a href="#">L4Re::Util::Br_manager_timeout_hooks Struct Reference</a>	1792
16.316.1	<a href="#">Detailed Description</a>	1795
16.317	<a href="#">L4Re::Util::Cap_alloc_base Class Reference</a>	1796
16.317.1	<a href="#">Detailed Description</a>	1796
16.318	<a href="#">L4Re::Util::Counter&lt; COUNTER &gt; Struct Template Reference</a>	1797
16.318.1	<a href="#">Detailed Description</a>	1797
16.318.2	<a href="#">Member Function Documentation</a>	1797
16.318.2.1	<a href="#">dec()</a>	1797
16.318.2.2	<a href="#">inc()</a>	1798
16.319	<a href="#">L4Re::Util::Counter_atomic&lt; COUNTER &gt; Struct Template Reference</a>	1799
16.319.1	<a href="#">Detailed Description</a>	1799
16.319.2	<a href="#">Member Function Documentation</a>	1800
16.319.2.1	<a href="#">dec()</a>	1800
16.319.2.2	<a href="#">inc()</a>	1800
16.320	<a href="#">L4Re::Util::Counting_cap_alloc&lt; COUNTERTYPE, Dbg &gt; Class Template Reference</a>	1800
16.320.1	<a href="#">Detailed Description</a>	1802
16.320.2	<a href="#">Constructor &amp; Destructor Documentation</a>	1802
16.320.2.1	<a href="#">Counting_cap_alloc()</a>	1802
16.320.3	<a href="#">Member Function Documentation</a>	1802
16.320.3.1	<a href="#">alloc() [1/2]</a>	1802
16.320.3.2	<a href="#">alloc() [2/2]</a>	1803
16.320.3.3	<a href="#">free()</a>	1804
16.320.3.4	<a href="#">release()</a>	1804
16.320.3.5	<a href="#">setup()</a>	1805
16.320.3.6	<a href="#">take()</a>	1806
16.321	<a href="#">L4Re::Util::Dataspace_svr Class Reference</a>	1806
16.321.1	<a href="#">Detailed Description</a>	1807
16.321.2	<a href="#">Member Function Documentation</a>	1807
16.321.2.1	<a href="#">allocate()</a>	1807
16.321.2.2	<a href="#">clear()</a>	1808
16.321.2.3	<a href="#">copy()</a>	1808
16.321.2.4	<a href="#">is_static()</a>	1809
16.321.2.5	<a href="#">map()</a>	1809
16.321.2.6	<a href="#">map_hook()</a>	1810
16.321.2.7	<a href="#">map_info()</a>	1811
16.321.2.8	<a href="#">page_shift()</a>	1812
16.321.2.9	<a href="#">release()</a>	1812



16.321.2.10 take()	1812
16.322 L4Re::Util::Event_buffer_consumer_t< PAYLOAD > Class Template Reference	1813
16.322.1 Detailed Description	1815
16.322.2 Member Function Documentation	1815
16.322.2.1 foreach_available_event()	1815
16.322.2.2 process()	1816
16.323 L4Re::Util::Event_buffer_t< PAYLOAD > Class Template Reference	1816
16.323.1 Detailed Description	1819
16.323.2 Member Function Documentation	1819
16.323.2.1 attach()	1819
16.323.2.2 buf()	1819
16.323.2.3 detach()	1819
16.324 L4Re::Util::Event_svr< SVR > Class Template Reference	1820
16.324.1 Detailed Description	1822
16.325 L4Re::Util::Event_t< PAYLOAD > Class Template Reference	1822
16.325.1 Detailed Description	1823
16.325.2 Member Enumeration Documentation	1824
16.325.2.1 Mode	1824
16.325.3 Member Function Documentation	1825
16.325.3.1 buffer()	1825
16.325.3.2 init()	1825
16.325.3.3 init_poll()	1826
16.325.3.4 irq()	1826
16.326 L4Re::Util::Item_alloc_base Class Reference	1827
16.326.1 Detailed Description	1827
16.327 L4Re::Util::Names::Name Class Reference	1827
16.327.1 Detailed Description	1831
16.328 L4Re::Util::Names::Name_space Class Reference	1831
16.328.1 Detailed Description	1832
16.328.2 Member Function Documentation	1832
16.328.2.1 alloc_dynamic_entry()	1832
16.328.2.2 copy_receive_cap()	1832
16.328.2.3 free_capability()	1833
16.328.2.4 free_dynamic_entry()	1833
16.328.2.5 free_epiface()	1834
16.328.2.6 get_epiface()	1835
16.329 L4Re::Util::Object_registry Class Reference	1836
16.329.1 Detailed Description	1838
16.329.2 Constructor & Destructor Documentation	1838
16.329.2.1 Object_registry() [1/2]	1838
16.329.2.2 Object_registry() [2/2]	1838
16.329.3 Member Function Documentation	1839

16.329.3.1 register_irq_obj()	1839
16.329.3.2 register_obj() [1/3]	1839
16.329.3.3 register_obj() [2/3]	1840
16.329.3.4 register_obj() [3/3]	1840
16.329.3.5 unregister_obj()	1841
16.330 L4Re::Util::Ref_cap< T > Struct Template Reference	1842
16.330.1 Detailed Description	1842
16.331 L4Re::Util::Ref_del_cap< T > Struct Template Reference	1843
16.331.1 Detailed Description	1843
16.332 L4Re::Util::Registry_server< LOOP_HOOKS > Class Template Reference	1844
16.332.1 Detailed Description	1848
16.332.2 Constructor & Destructor Documentation	1848
16.332.2.1 Registry_server() [1/3]	1848
16.332.2.2 Registry_server() [2/3]	1848
16.332.2.3 Registry_server() [3/3]	1849
16.332.3 Member Function Documentation	1849
16.332.3.1 loop()	1849
16.332.3.2 loop_dbg()	1850
16.333 L4Re::Util::Smart_cap_auto< Unmap_flags > Class Template Reference	1850
16.333.1 Detailed Description	1851
16.334 L4Re::Util::Smart_count_cap< Unmap_flags > Class Template Reference	1851
16.334.1 Detailed Description	1852
16.335 L4Re::Util::Vcon_svr< SVR > Class Template Reference	1852
16.335.1 Detailed Description	1853
16.336 L4Re::Util::Video::Goos_svr Class Reference	1853
16.336.1 Detailed Description	1855
16.336.2 Member Function Documentation	1855
16.336.2.1 get_fb()	1855
16.336.2.2 init_infos()	1855
16.336.2.3 refresh()	1856
16.336.2.4 screen_info()	1857
16.336.2.5 view_info()	1857
16.337 L4Re::Vfs::Be_file Class Reference	1858
16.337.1 Detailed Description	1861
16.337.2 Member Function Documentation	1861
16.337.2.1 check_ready()	1861
16.337.2.2 data_space()	1861
16.337.2.3 fstat()	1861
16.337.2.4 unlock_all_locks()	1862
16.338 L4Re::Vfs::Be_file_system Class Reference	1863
16.338.1 Detailed Description	1864
16.338.2 Constructor & Destructor Documentation	1864

16.338.2.1 Be_file_system()	1864
16.338.2.2 ~Be_file_system()	1865
16.338.3 Member Function Documentation	1865
16.338.3.1 type()	1865
16.339 L4Re::Vfs::Directory Class Reference	1865
16.339.1 Detailed Description	1867
16.339.2 Member Function Documentation	1868
16.339.2.1 faccessat()	1868
16.339.2.2 link()	1868
16.339.2.3 mkdir()	1868
16.339.2.4 rename()	1869
16.339.2.5 rmdir()	1869
16.339.2.6 symlink()	1870
16.339.2.7 unlink()	1870
16.340 L4Re::Vfs::File Class Reference	1871
16.340.1 Detailed Description	1873
16.341 L4Re::Vfs::File_system Class Reference	1874
16.341.1 Detailed Description	1875
16.341.2 Member Function Documentation	1875
16.341.2.1 mount()	1875
16.341.2.2 type()	1876
16.342 L4Re::Vfs::Fs Class Reference	1876
16.342.1 Detailed Description	1878
16.342.2 Member Function Documentation	1878
16.342.2.1 alloc_fd()	1878
16.342.2.2 free_fd()	1879
16.342.2.3 get_file()	1879
16.342.2.4 mount()	1879
16.342.2.5 set_fd()	1880
16.343 L4Re::Vfs::Generic_file Class Reference	1881
16.343.1 Detailed Description	1883
16.343.2 Member Enumeration Documentation	1884
16.343.2.1 Ready_type	1884
16.343.3 Member Function Documentation	1884
16.343.3.1 check_ready()	1884
16.343.3.2 fchmod()	1885
16.343.3.3 fstat()	1886
16.343.3.4 get_status_flags()	1887
16.343.3.5 set_status_flags()	1887
16.343.3.6 unlock_all_locks()	1889
16.344 L4Re::Vfs::Mman Class Reference	1890
16.344.1 Detailed Description	1891

16.345 L4Re::Vfs::Ops Class Reference . . . . .	1891
16.345.1 Detailed Description . . . . .	1894
16.346 L4Re::Vfs::Regular_file Class Reference . . . . .	1894
16.346.1 Detailed Description . . . . .	1896
16.346.2 Member Function Documentation . . . . .	1897
16.346.2.1 data_space() . . . . .	1897
16.346.2.2 fdatsync() . . . . .	1897
16.346.2.3 fsync() . . . . .	1898
16.346.2.4 ftruncate() . . . . .	1898
16.346.2.5 get_lock() . . . . .	1899
16.346.2.6 lseek() . . . . .	1900
16.346.2.7 readv() . . . . .	1901
16.346.2.8 set_lock() . . . . .	1902
16.346.2.9 writev() . . . . .	1903
16.347 L4Re::Vfs::Special_file Class Reference . . . . .	1904
16.347.1 Detailed Description . . . . .	1906
16.347.2 Member Function Documentation . . . . .	1906
16.347.2.1 ioctl() . . . . .	1906
16.348 L4Re::Video::Color_component Class Reference . . . . .	1907
16.348.1 Detailed Description . . . . .	1907
16.348.2 Constructor & Destructor Documentation . . . . .	1908
16.348.2.1 Color_component() . . . . .	1908
16.348.3 Member Function Documentation . . . . .	1908
16.348.3.1 dump() . . . . .	1908
16.348.3.2 get() . . . . .	1909
16.348.3.3 operator==( ) . . . . .	1909
16.348.3.4 set() . . . . .	1910
16.348.3.5 shift() . . . . .	1910
16.348.3.6 size() . . . . .	1911
16.349 L4Re::Video::Goos Class Reference . . . . .	1912
16.349.1 Detailed Description . . . . .	1915
16.349.2 Member Enumeration Documentation . . . . .	1915
16.349.2.1 Flags . . . . .	1915
16.349.3 Member Function Documentation . . . . .	1915
16.349.3.1 create_buffer() . . . . .	1915
16.349.3.2 create_view() . . . . .	1916
16.349.3.3 delete_buffer() . . . . .	1917
16.349.3.4 delete_view() . . . . .	1918
16.349.3.5 get_static_buffer() . . . . .	1919
16.349.3.6 info() . . . . .	1920
16.349.3.7 view() . . . . .	1921
16.350 L4Re::Video::Goos::Info Struct Reference . . . . .	1922

16.350.1 Detailed Description . . . . .	1924
16.351 L4Re::Video::Pixel_info Class Reference . . . . .	1924
16.351.1 Detailed Description . . . . .	1925
16.351.2 Constructor & Destructor Documentation . . . . .	1926
16.351.2.1 Pixel_info() [1/2] . . . . .	1926
16.351.2.2 Pixel_info() [2/2] . . . . .	1927
16.351.3 Member Function Documentation . . . . .	1928
16.351.3.1 a() [1/2] . . . . .	1928
16.351.3.2 a() [2/2] . . . . .	1928
16.351.3.3 b() [1/2] . . . . .	1929
16.351.3.4 b() [2/2] . . . . .	1929
16.351.3.5 bits_per_pixel() . . . . .	1930
16.351.3.6 bytes_per_pixel() [1/2] . . . . .	1930
16.351.3.7 bytes_per_pixel() [2/2] . . . . .	1930
16.351.3.8 dump() . . . . .	1931
16.351.3.9 g() [1/2] . . . . .	1932
16.351.3.10 g() [2/2] . . . . .	1932
16.351.3.11 has_alpha() . . . . .	1932
16.351.3.12 operator==( ) . . . . .	1933
16.351.3.13 padding() . . . . .	1933
16.351.3.14 r() [1/2] . . . . .	1934
16.351.3.15 r() [2/2] . . . . .	1934
16.352 L4Re::Video::View Class Reference . . . . .	1935
16.352.1 Detailed Description . . . . .	1936
16.352.2 Member Enumeration Documentation . . . . .	1936
16.352.2.1 Flags . . . . .	1936
16.352.2.2 V_flags . . . . .	1937
16.352.3 Member Function Documentation . . . . .	1937
16.352.3.1 info() . . . . .	1937
16.352.3.2 refresh() . . . . .	1938
16.352.3.3 set_info() . . . . .	1939
16.352.3.4 set_viewport() . . . . .	1940
16.352.3.5 stack() . . . . .	1940
16.353 L4Re::Video::View::Info Struct Reference . . . . .	1941
16.353.1 Detailed Description . . . . .	1943
16.354 l4re_aux_t Struct Reference . . . . .	1944
16.354.1 Detailed Description . . . . .	1944
16.355 l4re_ds_stats_t Struct Reference . . . . .	1945
16.355.1 Detailed Description . . . . .	1945
16.356 l4re_elf_aux_mword_t Struct Reference . . . . .	1945
16.356.1 Detailed Description . . . . .	1946
16.357 l4re_elf_aux_t Struct Reference . . . . .	1946

16.357.1 Detailed Description . . . . .	1946
16.358 l4re_elf_aux_vma_t Struct Reference . . . . .	1946
16.358.1 Detailed Description . . . . .	1947
16.359 l4re_env_cap_entry_t Struct Reference . . . . .	1947
16.359.1 Detailed Description . . . . .	1948
16.359.2 Constructor & Destructor Documentation . . . . .	1948
16.359.2.1 l4re_env_cap_entry_t() . . . . .	1948
16.359.3 Field Documentation . . . . .	1948
16.359.3.1 flags . . . . .	1948
16.360 l4re_env_t Struct Reference . . . . .	1949
16.360.1 Detailed Description . . . . .	1950
16.360.2 Field Documentation . . . . .	1950
16.360.2.1 caps . . . . .	1950
16.361 l4re_event_t Struct Reference . . . . .	1951
16.361.1 Detailed Description . . . . .	1951
16.362 l4re_video_color_component_t Struct Reference . . . . .	1952
16.362.1 Detailed Description . . . . .	1952
16.363 l4re_video_goos_info_t Struct Reference . . . . .	1952
16.363.1 Detailed Description . . . . .	1954
16.364 l4re_video_pixel_info_t Struct Reference . . . . .	1954
16.364.1 Detailed Description . . . . .	1955
16.365 l4re_video_view_info_t Struct Reference . . . . .	1955
16.365.1 Detailed Description . . . . .	1956
16.366 l4re_video_view_t Struct Reference . . . . .	1956
16.366.1 Detailed Description . . . . .	1957
16.367 l4shmc_ringbuf_head_t Struct Reference . . . . .	1957
16.367.1 Detailed Description . . . . .	1958
16.368 l4shmc_ringbuf_t Struct Reference . . . . .	1958
16.368.1 Detailed Description . . . . .	1959
16.369 l4util_l4mod_info Struct Reference . . . . .	1959
16.369.1 Detailed Description . . . . .	1960
16.369.2 Field Documentation . . . . .	1960
16.369.2.1 vbe_ctrl_info . . . . .	1960
16.370 l4util_l4mod_mod Struct Reference . . . . .	1960
16.370.1 Detailed Description . . . . .	1961
16.371 l4util_mb_addr_range_t Struct Reference . . . . .	1961
16.371.1 Detailed Description . . . . .	1962
16.372 l4util_mb_apm_t Struct Reference . . . . .	1962
16.372.1 Detailed Description . . . . .	1962
16.373 l4util_mb_drive_t Struct Reference . . . . .	1963
16.373.1 Detailed Description . . . . .	1963
16.374 l4util_mb_info_t Struct Reference . . . . .	1964

16.374.1 Detailed Description . . . . .	1965
16.375 lutil_mb_mod_t Struct Reference . . . . .	1965
16.375.1 Detailed Description . . . . .	1966
16.376 lutil_mb_vbe_ctrl_t Struct Reference . . . . .	1966
16.376.1 Detailed Description . . . . .	1967
16.377 lutil_mb_vbe_mode_t Struct Reference . . . . .	1967
16.377.1 Detailed Description . . . . .	1970
16.378 L4vbus::Device Class Reference . . . . .	1971
16.378.1 Detailed Description . . . . .	1973
16.378.2 Constructor & Destructor Documentation . . . . .	1973
16.378.2.1 Device() . . . . .	1973
16.378.3 Member Function Documentation . . . . .	1974
16.378.3.1 bus_cap() . . . . .	1974
16.378.3.2 dev_handle() . . . . .	1975
16.378.3.3 device() . . . . .	1975
16.378.3.4 device_by_hid() . . . . .	1976
16.378.3.5 get_resource() . . . . .	1977
16.378.3.6 is_compatible() . . . . .	1977
16.378.3.7 next_device() . . . . .	1978
16.378.3.8 operator!=(()) . . . . .	1979
16.378.3.9 operator==(()) . . . . .	1979
16.379 L4vbus::Gpio_module Class Reference . . . . .	1980
16.379.1 Detailed Description . . . . .	1984
16.379.2 Member Function Documentation . . . . .	1984
16.379.2.1 config_pad() . . . . .	1984
16.379.2.2 get() . . . . .	1984
16.379.2.3 pin() . . . . .	1985
16.379.2.4 set() . . . . .	1986
16.379.2.5 setup() . . . . .	1987
16.380 L4vbus::Gpio_module::Pin_slice Struct Reference . . . . .	1988
16.380.1 Detailed Description . . . . .	1988
16.381 L4vbus::Gpio_pin Class Reference . . . . .	1988
16.381.1 Detailed Description . . . . .	1992
16.381.2 Member Function Documentation . . . . .	1992
16.381.2.1 config_get() . . . . .	1992
16.381.2.2 config_pad() . . . . .	1993
16.381.2.3 config_pull() . . . . .	1994
16.381.2.4 get() . . . . .	1995
16.381.2.5 pin() . . . . .	1995
16.381.2.6 set() . . . . .	1995
16.381.2.7 setup() . . . . .	1996
16.381.2.8 to_irq() . . . . .	1997

16.382 L4vbus::Icu Class Reference . . . . .	1997
16.382.1 Detailed Description . . . . .	2001
16.382.2 Member Enumeration Documentation . . . . .	2001
16.382.2.1 Src_types . . . . .	2001
16.382.3 Member Function Documentation . . . . .	2001
16.382.3.1 vicu() . . . . .	2001
16.383 L4vbus::Pci_dev Class Reference . . . . .	2002
16.383.1 Detailed Description . . . . .	2005
16.383.2 Member Function Documentation . . . . .	2006
16.383.2.1 cfg_read() . . . . .	2006
16.383.2.2 cfg_write() . . . . .	2006
16.383.2.3 irq_enable() . . . . .	2007
16.384 L4vbus::Pci_host_bridge Class Reference . . . . .	2008
16.384.1 Detailed Description . . . . .	2012
16.384.2 Member Function Documentation . . . . .	2012
16.384.2.1 cfg_read() . . . . .	2012
16.384.2.2 cfg_write() . . . . .	2013
16.384.2.3 irq_enable() . . . . .	2013
16.385 L4vbus::Pm< DEC > Class Template Reference . . . . .	2014
16.385.1 Detailed Description . . . . .	2016
16.385.2 Member Function Documentation . . . . .	2016
16.385.2.1 pm_resume() . . . . .	2016
16.385.2.2 pm_suspend() . . . . .	2016
16.386 L4vbus::Vbus Class Reference . . . . .	2017
16.386.1 Detailed Description . . . . .	2024
16.386.2 Member Function Documentation . . . . .	2024
16.386.2.1 assign_dma_domain() [1/2] . . . . .	2024
16.386.2.2 assign_dma_domain() [2/2] . . . . .	2025
16.386.2.3 release_ioport() . . . . .	2026
16.386.2.4 request_ioport() . . . . .	2026
16.386.2.5 root() . . . . .	2027
16.387 l4vbus_device_t Struct Reference . . . . .	2028
16.387.1 Detailed Description . . . . .	2028
16.388 l4vbus_resource_t Struct Reference . . . . .	2029
16.388.1 Detailed Description . . . . .	2029
16.389 L4vcpu::State Class Reference . . . . .	2030
16.389.1 Detailed Description . . . . .	2030
16.389.2 Constructor & Destructor Documentation . . . . .	2030
16.389.2.1 State() . . . . .	2030
16.389.3 Member Function Documentation . . . . .	2031
16.389.3.1 add() . . . . .	2031
16.389.3.2 clear() . . . . .	2031



16.389.3.3 set()	2031
16.390 L4vcpu::Vcpu Class Reference	2032
16.390.1 Detailed Description	2035
16.390.2 Member Function Documentation	2035
16.390.2.1 cast() [1/2]	2035
16.390.2.2 cast() [2/2]	2035
16.390.2.3 entry_ip()	2036
16.390.2.4 entry_sp()	2037
16.390.2.5 ext_alloc()	2038
16.390.2.6 i() [1/2]	2039
16.390.2.7 i() [2/2]	2039
16.390.2.8 irq_disable_save()	2039
16.390.2.9 irq_enable()	2039
16.390.2.10 irq_restore()	2040
16.390.2.11 is_irq_entry()	2041
16.390.2.12 is_page_fault_entry()	2041
16.390.2.13 r() [1/2]	2042
16.390.2.14 r() [2/2]	2042
16.390.2.15 saved_state() [1/2]	2042
16.390.2.16 saved_state() [2/2]	2042
16.390.2.17 state() [1/2]	2043
16.390.2.18 state() [2/2]	2043
16.390.2.19 task()	2043
16.390.2.20 wait_for_event()	2044
16.391 L4virtio::Device Class Reference	2045
16.391.1 Detailed Description	2049
16.391.2 Member Function Documentation	2049
16.391.2.1 config_queue()	2049
16.391.2.2 device_config()	2050
16.391.2.3 device_notification_irq()	2051
16.391.2.4 register_ds()	2052
16.391.2.5 set_status()	2053
16.392 L4virtio::Driver::Block_device Class Reference	2054
16.392.1 Detailed Description	2058
16.392.2 Member Function Documentation	2058
16.392.2.1 add_block()	2058
16.392.2.2 process_request()	2059
16.392.2.3 process_used_queue()	2059
16.392.2.4 send_request()	2060
16.392.2.5 setup_device()	2061
16.392.2.6 start_request()	2062
16.393 L4virtio::Driver::Block_device::Handle Class Reference	2063

16.393.1 Detailed Description . . . . .	2063
16.394 L4virtio::Driver::Device Class Reference . . . . .	2063
16.394.1 Detailed Description . . . . .	2066
16.394.2 Member Function Documentation . . . . .	2066
16.394.2.1 bind_notification_irq() . . . . .	2066
16.394.2.2 config_queue() . . . . .	2067
16.394.2.3 driver_acknowledge() . . . . .	2068
16.394.2.4 driver_connect() . . . . .	2068
16.394.2.5 feature_negotiated() . . . . .	2070
16.394.2.6 max_queue_size() . . . . .	2071
16.394.2.7 register_ds() . . . . .	2071
16.394.2.8 send() . . . . .	2072
16.394.2.9 send_and_wait() . . . . .	2073
16.394.2.10 wait() . . . . .	2074
16.394.2.11 wait_for_next_used() . . . . .	2075
16.395 L4virtio::Driver::Virtio_net_device Class Reference . . . . .	2076
16.395.1 Detailed Description . . . . .	2080
16.395.2 Member Function Documentation . . . . .	2080
16.395.2.1 bind_rx_notification_irq() . . . . .	2080
16.395.2.2 finish_rx() . . . . .	2080
16.395.2.3 rx_pkt() . . . . .	2081
16.395.2.4 rx_queue_size() . . . . .	2081
16.395.2.5 setup_device() . . . . .	2082
16.395.2.6 tx() . . . . .	2083
16.395.2.7 tx_queue_size() . . . . .	2084
16.395.2.8 wait_rx() . . . . .	2085
16.396 L4virtio::Driver::Virtio_net_device::Packet Struct Reference . . . . .	2086
16.396.1 Detailed Description . . . . .	2086
16.397 L4virtio::Driver::Virtqueue Class Reference . . . . .	2086
16.397.1 Detailed Description . . . . .	2090
16.397.2 Member Function Documentation . . . . .	2091
16.397.2.1 alloc_descriptor() . . . . .	2091
16.397.2.2 desc() . . . . .	2091
16.397.2.3 enqueue_descriptor() . . . . .	2091
16.397.2.4 find_next_used() . . . . .	2092
16.397.2.5 free_descriptor() . . . . .	2093
16.397.2.6 init_queue() [1/2] . . . . .	2094
16.397.2.7 init_queue() [2/2] . . . . .	2095
16.397.2.8 initialize_rings() . . . . .	2096
16.398 L4virtio::Ptr< T > Class Template Reference . . . . .	2097
16.398.1 Detailed Description . . . . .	2099
16.398.2 Member Enumeration Documentation . . . . .	2099

16.398.2.1 Invalid_type . . . . .	2099
16.398.3 Member Function Documentation . . . . .	2099
16.398.3.1 get() . . . . .	2099
16.398.3.2 is_valid() . . . . .	2100
16.399 L4virtio::Svr::Bad_descriptor Struct Reference . . . . .	2101
16.399.1 Detailed Description . . . . .	2102
16.399.2 Member Enumeration Documentation . . . . .	2102
16.399.2.1 Error . . . . .	2102
16.399.3 Constructor & Destructor Documentation . . . . .	2102
16.399.3.1 Bad_descriptor() . . . . .	2102
16.399.4 Member Function Documentation . . . . .	2103
16.399.4.1 message() . . . . .	2103
16.400 L4virtio::Svr::Block_dev_base< Ds_data > Class Template Reference . . . . .	2103
16.400.1 Detailed Description . . . . .	2107
16.400.2 Constructor & Destructor Documentation . . . . .	2107
16.400.2.1 Block_dev_base() . . . . .	2107
16.400.3 Member Function Documentation . . . . .	2108
16.400.3.1 finalize_request() . . . . .	2108
16.400.3.2 get_writeback() . . . . .	2109
16.400.3.3 set_blk_size() . . . . .	2109
16.400.3.4 set_config_wce() . . . . .	2109
16.400.3.5 set_discard() . . . . .	2110
16.400.3.6 set_size_max() . . . . .	2110
16.400.3.7 set_topology() . . . . .	2110
16.400.3.8 set_write_zeroes() . . . . .	2111
16.401 L4virtio::Svr::Block_request< Ds_data > Class Template Reference . . . . .	2111
16.401.1 Detailed Description . . . . .	2112
16.401.2 Member Function Documentation . . . . .	2112
16.401.2.1 data_size() . . . . .	2112
16.401.2.2 next_block() . . . . .	2113
16.402 L4virtio::Svr::Console::Control_message Struct Reference . . . . .	2114
16.402.1 Detailed Description . . . . .	2114
16.402.2 Member Enumeration Documentation . . . . .	2114
16.402.2.1 Events . . . . .	2114
16.403 L4virtio::Svr::Console::Control_request Struct Reference . . . . .	2115
16.403.1 Detailed Description . . . . .	2117
16.404 L4virtio::Svr::Console::Device Class Reference . . . . .	2117
16.404.1 Detailed Description . . . . .	2122
16.404.2 Constructor & Destructor Documentation . . . . .	2122
16.404.2.1 Device() [1/3] . . . . .	2122
16.404.2.2 Device() [2/3] . . . . .	2123
16.404.2.3 Device() [3/3] . . . . .	2123

16.404.3 Member Function Documentation . . . . .	2124
16.404.3.1 notify_queue() . . . . .	2124
16.404.3.2 port() . . . . .	2125
16.404.3.3 port_read() . . . . .	2126
16.404.3.4 port_write() . . . . .	2128
16.404.3.5 process_device_ready() . . . . .	2129
16.404.3.6 process_port_open() . . . . .	2130
16.404.3.7 process_port_ready() . . . . .	2130
16.405 L4virtio::Svr::Console::Device_port Struct Reference . . . . .	2131
16.405.1 Detailed Description . . . . .	2135
16.406 L4virtio::Svr::Console::Features Struct Reference . . . . .	2135
16.406.1 Detailed Description . . . . .	2137
16.406.2 Member Typedef Documentation . . . . .	2138
16.406.2.1 console_multiport_bfm_t . . . . .	2138
16.406.2.2 console_size_bfm_t . . . . .	2138
16.406.2.3 emerg_write_bfm_t . . . . .	2138
16.407 L4virtio::Svr::Console::Port Struct Reference . . . . .	2139
16.407.1 Detailed Description . . . . .	2142
16.407.2 Member Enumeration Documentation . . . . .	2142
16.407.2.1 Port_status . . . . .	2142
16.407.3 Field Documentation . . . . .	2143
16.407.3.1 state_transitions . . . . .	2143
16.408 L4virtio::Svr::Console::Port::Transition Struct Reference . . . . .	2143
16.408.1 Detailed Description . . . . .	2144
16.409 L4virtio::Svr::Console::Virtio_con Class Reference . . . . .	2144
16.409.1 Detailed Description . . . . .	2148
16.409.2 Constructor & Destructor Documentation . . . . .	2148
16.409.2.1 Virtio_con() . . . . .	2148
16.409.3 Member Function Documentation . . . . .	2149
16.409.3.1 handle_control_message() . . . . .	2149
16.409.3.2 notify_queue() . . . . .	2150
16.409.3.3 port() . . . . .	2151
16.409.3.4 port_add() . . . . .	2152
16.409.3.5 port_name() . . . . .	2153
16.409.3.6 port_open() . . . . .	2154
16.409.3.7 port_remove() . . . . .	2155
16.409.3.8 process_device_ready() . . . . .	2156
16.409.3.9 process_port_open() . . . . .	2157
16.409.3.10 process_port_ready() . . . . .	2158
16.409.3.11 reset_device() . . . . .	2159
16.409.3.12 send_control_message() . . . . .	2159
16.410 L4virtio::Svr::Data_buffer Struct Reference . . . . .	2161

16.410.1 Detailed Description . . . . .	2162
16.410.2 Constructor & Destructor Documentation . . . . .	2162
16.410.2.1 Data_buffer() . . . . .	2162
16.410.3 Member Function Documentation . . . . .	2163
16.410.3.1 copy_to() . . . . .	2163
16.410.3.2 done() . . . . .	2164
16.410.3.3 set() . . . . .	2164
16.410.3.4 skip() . . . . .	2164
16.411 L4virtio::Svr::Dev_config Class Reference . . . . .	2165
16.411.1 Detailed Description . . . . .	2167
16.411.2 Constructor & Destructor Documentation . . . . .	2167
16.411.2.1 Dev_config() [1/2] . . . . .	2167
16.411.2.2 Dev_config() [2/2] . . . . .	2168
16.411.3 Member Function Documentation . . . . .	2169
16.411.3.1 add_irq_status() . . . . .	2169
16.411.3.2 change_queue_config() . . . . .	2169
16.411.3.3 ds() . . . . .	2170
16.411.3.4 get_cmd() . . . . .	2170
16.411.3.5 guest_features() . . . . .	2171
16.411.3.6 hdr() . . . . .	2171
16.411.3.7 negotiated_features() . . . . .	2172
16.411.3.8 qconfig() . . . . .	2173
16.411.3.9 reset_cmd() . . . . .	2173
16.411.3.10 reset_queue() . . . . .	2174
16.411.3.11 set_device_needs_reset() . . . . .	2174
16.411.3.12 set_status() . . . . .	2175
16.411.3.13 status() . . . . .	2176
16.412 L4virtio::Svr::Dev_features Struct Reference . . . . .	2176
16.412.1 Detailed Description . . . . .	2179
16.413 L4virtio::Svr::Dev_status Struct Reference . . . . .	2179
16.413.1 Detailed Description . . . . .	2181
16.413.2 Member Function Documentation . . . . .	2181
16.413.2.1 running() . . . . .	2181
16.414 L4virtio::Svr::Device_t< DATA > Class Template Reference . . . . .	2182
16.414.1 Detailed Description . . . . .	2184
16.414.2 Member Function Documentation . . . . .	2184
16.414.2.1 add_trusted_dataspaces() . . . . .	2184
16.414.2.2 device_error() . . . . .	2185
16.414.2.3 device_notify_irq() . . . . .	2185
16.414.2.4 handle_mem_cmd_write() . . . . .	2185
16.414.2.5 init_mem_info() . . . . .	2186
16.414.2.6 register_driver_irq() . . . . .	2186

16.414.2.7 reset_queue_config()	2186
16.414.2.8 setup_queue()	2187
16.415 L4virtio::Svr::Driver_mem_list_t< DATA > Class Template Reference	2188
16.415.1 Detailed Description	2190
16.415.2 Member Function Documentation	2190
16.415.2.1 add()	2190
16.415.2.2 find()	2190
16.415.2.3 full()	2191
16.415.2.4 init()	2192
16.415.2.5 load_desc() [1/3]	2192
16.415.2.6 load_desc() [2/3]	2193
16.415.2.7 load_desc() [3/3]	2193
16.415.2.8 remove()	2194
16.416 L4virtio::Svr::Driver_mem_region_t< DATA > Class Template Reference	2194
16.416.1 Detailed Description	2196
16.416.2 Constructor & Destructor Documentation	2198
16.416.2.1 Driver_mem_region_t()	2198
16.416.3 Member Function Documentation	2198
16.416.3.1 contains()	2198
16.416.3.2 drv_base()	2199
16.416.3.3 ds()	2199
16.416.3.4 ds_offset()	2199
16.416.3.5 empty()	2199
16.416.3.6 flags()	2200
16.416.3.7 is_writable()	2200
16.416.3.8 local()	2200
16.416.3.9 local_base()	2201
16.416.3.10 size()	2201
16.417 L4virtio::Svr::Request_processor Class Reference	2201
16.417.1 Detailed Description	2202
16.417.2 Member Function Documentation	2203
16.417.2.1 current_flags()	2203
16.417.2.2 has_more()	2203
16.417.2.3 next()	2204
16.417.2.4 start() [1/2]	2205
16.417.2.5 start() [2/2]	2207
16.418 L4virtio::Svr::Scmi::Base_attr_t Struct Reference	2208
16.418.1 Detailed Description	2209
16.419 L4virtio::Svr::Scmi::Base_proto Class Reference	2209
16.419.1 Detailed Description	2211
16.420 L4virtio::Svr::Scmi::Perf_proto Class Reference	2211
16.420.1 Detailed Description	2213

16.421 L4virtio::Svr::Scmi::Performance_attr_t Struct Reference . . . . .	2214
16.421.1 Detailed Description . . . . .	2214
16.422 L4virtio::Svr::Scmi::Performance_describe_level_t Struct Reference . . . . .	2215
16.422.1 Detailed Description . . . . .	2215
16.423 L4virtio::Svr::Scmi::Performance_describe_levels_n_t Struct Reference . . . . .	2215
16.423.1 Detailed Description . . . . .	2216
16.424 L4virtio::Svr::Scmi::Performance_domain_attr_t Struct Reference . . . . .	2217
16.424.1 Detailed Description . . . . .	2218
16.425 L4virtio::Svr::Scmi::Proto< OBSERV > Struct Template Reference . . . . .	2219
16.425.1 Detailed Description . . . . .	2220
16.426 L4virtio::Svr::Scmi::Scmi_dev Class Reference . . . . .	2220
16.426.1 Detailed Description . . . . .	2223
16.427 L4virtio::Svr::Scmi::Scmi_hdr_t Struct Reference . . . . .	2224
16.427.1 Detailed Description . . . . .	2225
16.428 L4virtio::Svr::Virtio_gpio< Request_handler, Epiface > Class Template Reference . . . . .	2225
16.428.1 Detailed Description . . . . .	2229
16.429 L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::Host_irq Struct Reference . . . . .	2230
16.429.1 Detailed Description . . . . .	2233
16.430 L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::Irq_handler Struct Reference . . . . .	2234
16.430.1 Detailed Description . . . . .	2234
16.431 L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::Request_processor Struct Reference . . . . .	2235
16.431.1 Detailed Description . . . . .	2237
16.431.2 Member Function Documentation . . . . .	2237
16.431.2.1 get_request() . . . . .	2237
16.432 L4virtio::Svr::Virtio_i2c< Request_handler, Epiface > Class Template Reference . . . . .	2238
16.432.1 Detailed Description . . . . .	2241
16.433 L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >::Host_irq Class Reference . . . . .	2242
16.433.1 Detailed Description . . . . .	2245
16.434 L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >::Request_processor Class Reference . . . . .	2246
16.434.1 Detailed Description . . . . .	2248
16.434.2 Member Function Documentation . . . . .	2248
16.434.2.1 get_request() . . . . .	2248
16.435 L4virtio::Svr::Virtio_rng< Rnd_state, Epiface > Class Template Reference . . . . .	2249
16.435.1 Detailed Description . . . . .	2252
16.436 L4virtio::Svr::Virtio_rng< Rnd_state, Epiface >::Host_irq Class Reference . . . . .	2253
16.436.1 Detailed Description . . . . .	2256
16.437 L4virtio::Svr::Virtio_rng< Rnd_state, Epiface >::Request_processor Class Reference . . . . .	2257
16.437.1 Detailed Description . . . . .	2258
16.438 L4virtio::Svr::Virtqueue Class Reference . . . . .	2259
16.438.1 Detailed Description . . . . .	2263
16.438.2 Member Function Documentation . . . . .	2264
16.438.2.1 consumed() [1/2] . . . . .	2264

16.438.2.2 consumed() [2/2]	2264
16.438.2.3 desc()	2265
16.438.2.4 desc_avail()	2266
16.438.2.5 disable_notify()	2267
16.438.2.6 enable_notify()	2267
16.438.2.7 finish() [1/2]	2268
16.438.2.8 finish() [2/2]	2269
16.438.2.9 next_avail()	2270
16.438.2.10 rewind_avail()	2271
16.439 L4virtio::Svr::Virtqueue::Head_desc Class Reference	2272
16.439.1 Detailed Description	2272
16.439.2 Member Function Documentation	2272
16.439.2.1 desc()	2272
16.439.2.2 operator bool()	2273
16.439.2.3 valid()	2274
16.440 L4virtio::Virtqueue Class Reference	2274
16.440.1 Detailed Description	2278
16.440.2 Member Function Documentation	2278
16.440.2.1 avail_align()	2278
16.440.2.2 avail_size()	2278
16.440.2.3 desc_align()	2280
16.440.2.4 desc_size()	2280
16.440.2.5 disable()	2281
16.440.2.6 dump()	2282
16.440.2.7 get_avail_idx()	2282
16.440.2.8 get_tail_avail_idx()	2283
16.440.2.9 no_notify_guest()	2283
16.440.2.10 no_notify_host() [1/2]	2284
16.440.2.11 no_notify_host() [2/2]	2284
16.440.2.12 num()	2284
16.440.2.13 ready()	2285
16.440.2.14 setup()	2286
16.440.2.15 setup_simple()	2287
16.440.2.16 total_size() [1/2]	2288
16.440.2.17 total_size() [2/2]	2289
16.440.2.18 used_align()	2290
16.440.2.19 used_size()	2290
16.441 L4virtio::Virtqueue::Avail Class Reference	2291
16.441.1 Detailed Description	2292
16.442 L4virtio::Virtqueue::Avail::Flags Struct Reference	2293
16.442.1 Detailed Description	2293
16.442.2 Member Typedef Documentation	2294



16.442.2.1 no_irq_bfm_t . . . . .	2294
16.443 L4virtio::Virtqueue::Desc Class Reference . . . . .	2294
16.443.1 Detailed Description . . . . .	2296
16.444 L4virtio::Virtqueue::Desc::Flags Struct Reference . . . . .	2296
16.444.1 Detailed Description . . . . .	2297
16.444.2 Member Typedef Documentation . . . . .	2297
16.444.2.1 indirect_bfm_t . . . . .	2297
16.444.2.2 next_bfm_t . . . . .	2298
16.444.2.3 write_bfm_t . . . . .	2298
16.445 L4virtio::Virtqueue::Used Class Reference . . . . .	2298
16.445.1 Detailed Description . . . . .	2299
16.446 L4virtio::Virtqueue::Used::Flags Struct Reference . . . . .	2299
16.446.1 Detailed Description . . . . .	2300
16.446.2 Member Typedef Documentation . . . . .	2300
16.446.2.1 no_notify_bfm_t . . . . .	2300
16.447 L4virtio::Virtqueue::Used_elem Struct Reference . . . . .	2301
16.447.1 Detailed Description . . . . .	2301
16.447.2 Constructor & Destructor Documentation . . . . .	2301
16.447.2.1 Used_elem() . . . . .	2301
16.448 l4virtio_block_config_t Struct Reference . . . . .	2302
16.448.1 Detailed Description . . . . .	2303
16.449 l4virtio_block_discard_t Struct Reference . . . . .	2303
16.449.1 Detailed Description . . . . .	2303
16.450 l4virtio_block_header_t Struct Reference . . . . .	2304
16.450.1 Detailed Description . . . . .	2304
16.451 l4virtio_config_hdr_t Struct Reference . . . . .	2305
16.451.1 Detailed Description . . . . .	2305
16.452 l4virtio_config_queue_t Struct Reference . . . . .	2306
16.452.1 Detailed Description . . . . .	2307
16.453 l4virtio_input_absinfo_t Struct Reference . . . . .	2307
16.453.1 Detailed Description . . . . .	2307
16.454 l4virtio_input_config_t Struct Reference . . . . .	2308
16.454.1 Detailed Description . . . . .	2308
16.455 l4virtio_input_devids_t Struct Reference . . . . .	2308
16.455.1 Detailed Description . . . . .	2309
16.456 l4virtio_input_event_t Struct Reference . . . . .	2309
16.456.1 Detailed Description . . . . .	2309
16.457 l4virtio_net_config_t Struct Reference . . . . .	2309
16.457.1 Detailed Description . . . . .	2310
16.458 l4virtio_net_header_t Struct Reference . . . . .	2310
16.458.1 Detailed Description . . . . .	2310
16.459 L4virtio_port Class Reference . . . . .	2311

16.459.1 Detailed Description . . . . .	2315
16.459.2 Member Function Documentation . . . . .	2315
16.459.2.1 drop_requests() . . . . .	2315
16.460 Mac_addr Class Reference . . . . .	2316
16.460.1 Detailed Description . . . . .	2316
16.461 Mac_table< Size > Class Template Reference . . . . .	2316
16.461.1 Detailed Description . . . . .	2317
16.461.2 Member Function Documentation . . . . .	2317
16.461.2.1 flush() . . . . .	2317
16.461.2.2 learn() . . . . .	2318
16.461.2.3 lookup() . . . . .	2318
16.462 Net_transfer Class Reference . . . . .	2319
16.462.1 Detailed Description . . . . .	2320
16.462.2 Member Function Documentation . . . . .	2321
16.462.2.1 cur_buf() . . . . .	2321
16.462.2.2 done() . . . . .	2321
16.463 Rm Class Reference . . . . .	2322
16.463.1 Detailed Description . . . . .	2326
16.463.2 Member Enumeration Documentation . . . . .	2326
16.463.2.1 Detach_flags . . . . .	2326
16.463.2.2 Detach_result . . . . .	2326
16.463.2.3 Region_flag_shifts . . . . .	2327
16.463.3 Member Function Documentation . . . . .	2327
16.463.3.1 attach() [1/2] . . . . .	2327
16.463.3.2 attach() [2/2] . . . . .	2328
16.463.3.3 detach() [1/3] . . . . .	2330
16.463.3.4 detach() [2/3] . . . . .	2331
16.463.3.5 detach() [3/3] . . . . .	2332
16.463.3.6 find() . . . . .	2332
16.463.3.7 free_area() . . . . .	2333
16.463.3.8 get_areas() . . . . .	2334
16.463.3.9 get_info() . . . . .	2334
16.463.3.10 get_regions() . . . . .	2335
16.463.3.11 reserve_area() [1/2] . . . . .	2335
16.463.3.12 reserve_area() [2/2] . . . . .	2336
16.464 Rm::Area Struct Reference . . . . .	2337
16.464.1 Detailed Description . . . . .	2337
16.465 Rm::F Struct Reference . . . . .	2338
16.465.1 Detailed Description . . . . .	2338
16.465.2 Member Enumeration Documentation . . . . .	2338
16.465.2.1 Attach_flags . . . . .	2338
16.465.2.2 Region_flags . . . . .	2339

16.466 Rm::Region Struct Reference . . . . .	2340
16.466.1 Detailed Description . . . . .	2340
16.467 Rm::Unique_region< T > Class Template Reference . . . . .	2340
16.467.1 Detailed Description . . . . .	2342
16.467.2 Constructor & Destructor Documentation . . . . .	2342
16.467.2.1 Unique_region() [1/3] . . . . .	2342
16.467.2.2 Unique_region() [2/3] . . . . .	2342
16.467.2.3 Unique_region() [3/3] . . . . .	2343
16.467.2.4 ~Unique_region() . . . . .	2343
16.467.3 Member Function Documentation . . . . .	2343
16.467.3.1 get() . . . . .	2343
16.467.3.2 is_valid() . . . . .	2344
16.467.3.3 operator=() . . . . .	2344
16.467.3.4 release() . . . . .	2344
16.467.3.5 reset() . . . . .	2344
16.468 Switch_factory Class Reference . . . . .	2345
16.468.1 Detailed Description . . . . .	2348
16.468.2 Member Function Documentation . . . . .	2349
16.468.2.1 op_create() . . . . .	2349
16.469 Virtio_net Class Reference . . . . .	2349
16.469.1 Detailed Description . . . . .	2354
16.469.2 Member Function Documentation . . . . .	2354
16.469.2.1 notify_queue() . . . . .	2354
16.470 Virtio_net_request Class Reference . . . . .	2354
16.470.1 Detailed Description . . . . .	2355
16.470.2 Member Function Documentation . . . . .	2355
16.470.2.1 drop_requests() . . . . .	2355
16.470.2.2 get_request() . . . . .	2356
16.471 Virtio_switch Class Reference . . . . .	2357
16.471.1 Detailed Description . . . . .	2358
16.471.2 Constructor & Destructor Documentation . . . . .	2358
16.471.2.1 Virtio_switch() . . . . .	2358
16.471.3 Member Function Documentation . . . . .	2359
16.471.3.1 add_monitor_port() . . . . .	2359
16.471.3.2 add_port() . . . . .	2359
16.471.3.3 check_ports() . . . . .	2360
16.471.3.4 handle_l4virtio_port_tx() . . . . .	2360
16.471.3.5 port_available() . . . . .	2361
16.472 Virtio_vlan_mangle Class Reference . . . . .	2362
16.472.1 Detailed Description . . . . .	2362
16.472.2 Constructor & Destructor Documentation . . . . .	2363
16.472.2.1 Virtio_vlan_mangle() . . . . .	2363

16.472.3 Member Function Documentation	2363
16.472.3.1 add()	2363
16.472.3.2 copy_pkt()	2364
16.472.3.3 remove()	2364
16.472.3.4 rewrite_hdr()	2365
<b>17 File Documentation</b>	<b>2367</b>
17.1 asm_access.h	2367
17.2 asm_access.h	2367
17.3 asm_access.h	2368
17.4 asm_access.h	2369
17.5 asm_access.h	2369
17.6 asm_access.h	2370
17.7 asm_access.h	2371
17.8 asm_access.h	2371
17.9 asm_access_gen.h	2372
17.10 hw_mmio_register_block	2372
17.11 hw_register_block	2373
17.12 io_regblock.h	2376
17.13 io_regblock_port.h	2379
17.14 poll_timeout_counter.h	2379
17.15 uart_16550.h	2380
17.16 uart_16550_dw.h	2381
17.17 uart_apb.h	2382
17.18 uart_base.h	2382
17.19 uart_cadence.h	2383
17.20 uart_dcc-v6.h	2384
17.21 uart_dm.h	2384
17.22 uart_dummy.h	2385
17.23 uart_geni.h	2385
17.24 uart_imx.h	2386
17.25 uart_leon3.h	2387
17.26 uart_linflex.h	2387
17.27 uart_lpuart.h	2388
17.28 uart_mvebu.h	2388
17.29 uart_of.h	2389
17.30 uart_omap35x.h	2389
17.31 uart_pl011.h	2390
17.32 uart_s3c2410.h	2390
17.33 uart_sa1000.h	2391
17.34 uart_sbi.h	2392
17.35 uart_sh.h	2392

17.36	uart_sifive.h	2393
17.37	uart_tegra-tcu.h	2393
17.38	tutorial.lua	2394
17.39	cmd_control	2397
17.40	debug.h	2397
17.41	debug.h	2398
17.42	l4/re/c/debug.h File Reference	2399
17.42.1	Detailed Description	2399
17.43	debug.h	2400
17.44	filter.cc	2400
17.45	filter.h	2400
17.46	mac_addr.h	2401
17.47	mac_table.h	2402
17.48	main.cc	2404
17.49	Makefile	2412
17.50	Makefile	2412
17.51	Makefile	2412
17.52	Makefile	2412
17.53	options.cc	2413
17.54	options.h	2415
17.55	port.h	2416
17.56	port_ixl.h	2418
17.57	port_l4virtio.h	2420
17.58	request.h	2424
17.59	request.h	2425
17.60	request_ixl.h	2425
17.61	request_l4virtio.h	2427
17.62	stats.h	2430
17.63	switch.cc	2431
17.64	switch.h	2434
17.65	virtio_net.h	2435
17.66	virtio_net.h	2438
17.67	virtio_net_buffer.h	2439
17.68	vlan.h	2440
17.69	amd64/l4/util/perform.h File Reference	2441
17.69.1	Detailed Description	2441
17.70	perform.h	2442
17.71	x86/l4/util/perform.h File Reference	2447
17.71.1	Detailed Description	2447
17.72	perform.h	2448
17.73	amd64/l4/util/rdtsc.h File Reference	2453
17.73.1	Detailed Description	2454

17.74 rdtsc.h . . . . .	2454
17.75 x86/i4/util/rdtsc.h File Reference . . . . .	2457
17.75.1 Detailed Description . . . . .	2458
17.76 rdtsc.h . . . . .	2458
17.77 amd64/i4/util/spin.h File Reference . . . . .	2461
17.77.1 Detailed Description . . . . .	2461
17.78 spin.h . . . . .	2462
17.79 x86/i4/util/spin.h File Reference . . . . .	2462
17.79.1 Detailed Description . . . . .	2463
17.80 spin.h . . . . .	2463
17.81 amd64/i4/sys/segment.h File Reference . . . . .	2463
17.81.1 Detailed Description . . . . .	2465
17.81.2 Enumeration Type Documentation . . . . .	2465
17.81.2.1 L4_sys_segment . . . . .	2465
17.81.2.2 L4_task_ldt_x86_consts . . . . .	2465
17.81.3 Function Documentation . . . . .	2465
17.81.3.1 fiasco_amd64_segment_info() . . . . .	2465
17.81.3.2 fiasco_amd64_set_fs() . . . . .	2466
17.81.3.3 fiasco_amd64_set_segment_base() . . . . .	2467
17.82 segment.h . . . . .	2468
17.83 amd64/i4f/i4/sys/segment.h File Reference . . . . .	2469
17.83.1 Detailed Description . . . . .	2470
17.83.2 Function Documentation . . . . .	2470
17.83.2.1 fiasco_amd64_set_fs() . . . . .	2470
17.83.2.2 fiasco_amd64_set_segment_base() . . . . .	2471
17.84 segment.h . . . . .	2472
17.85 x86/i4/sys/segment.h File Reference . . . . .	2473
17.85.1 Detailed Description . . . . .	2474
17.85.2 Enumeration Type Documentation . . . . .	2474
17.85.2.1 L4_task_ldt_x86_consts . . . . .	2474
17.86 segment.h . . . . .	2474
17.87 x86/i4f/i4/sys/segment.h File Reference . . . . .	2475
17.87.1 Detailed Description . . . . .	2476
17.88 segment.h . . . . .	2476
17.89 amd64/i4/util/port_io.h File Reference . . . . .	2477
17.89.1 Detailed Description . . . . .	2477
17.90 port_io.h . . . . .	2477
17.91 amd64/i4f/i4/util/port_io.h File Reference . . . . .	2478
17.91.1 Detailed Description . . . . .	2478
17.92 port_io.h . . . . .	2478
17.93 x86/i4/util/port_io.h File Reference . . . . .	2478
17.93.1 Detailed Description . . . . .	2480

17.94 port_io.h . . . . .	2480
17.95 x86/l4/l4/util/port_io.h File Reference . . . . .	2482
17.95.1 Detailed Description . . . . .	2483
17.96 port_io.h . . . . .	2484
17.97 __kip-arch.h . . . . .	2484
17.98 __kip-arch.h . . . . .	2485
17.99 __kip-arch.h . . . . .	2485
17.100 __kip-arch.h . . . . .	2485
17.101 amd64/l4/sys/__vcpu-arch.h File Reference . . . . .	2485
17.101.1 Detailed Description . . . . .	2487
17.101.2 Enumeration Type Documentation . . . . .	2487
17.101.2.1 anonymous enum . . . . .	2487
17.102 __vcpu-arch.h . . . . .	2487
17.103 arm/l4/sys/__vcpu-arch.h File Reference . . . . .	2488
17.103.1 Detailed Description . . . . .	2489
17.103.2 Enumeration Type Documentation . . . . .	2489
17.103.2.1 anonymous enum . . . . .	2489
17.103.2.2 L4_vcpu_e_field_ids . . . . .	2490
17.104 __vcpu-arch.h . . . . .	2490
17.105 arm64/l4/sys/__vcpu-arch.h File Reference . . . . .	2491
17.105.1 Detailed Description . . . . .	2493
17.105.2 Enumeration Type Documentation . . . . .	2493
17.105.2.1 anonymous enum . . . . .	2493
17.105.2.2 L4_vcpu_e_field_ids . . . . .	2493
17.106 __vcpu-arch.h . . . . .	2494
17.107 x86/l4/sys/__vcpu-arch.h File Reference . . . . .	2495
17.107.1 Detailed Description . . . . .	2496
17.107.2 Enumeration Type Documentation . . . . .	2496
17.107.2.1 anonymous enum . . . . .	2496
17.108 __vcpu-arch.h . . . . .	2496
17.109 ktrace_events.h . . . . .	2497
17.110 ktrace_events.h . . . . .	2500
17.111 ktrace_events.h . . . . .	2503
17.112 ktrace_events.h . . . . .	2506
17.113 amd64/l4/sys/linkage.h File Reference . . . . .	2509
17.113.1 Detailed Description . . . . .	2510
17.114 linkage.h . . . . .	2510
17.115 arm/l4/sys/linkage.h File Reference . . . . .	2510
17.115.1 Detailed Description . . . . .	2510
17.116 linkage.h . . . . .	2511
17.117 linkage.h . . . . .	2511
17.118 x86/l4/sys/linkage.h File Reference . . . . .	2511

17.118.1 Detailed Description . . . . .	2511
17.119 linkage.h . . . . .	2512
17.120 arm/l4/sys/mem_op.h File Reference . . . . .	2512
17.120.1 Detailed Description . . . . .	2513
17.121 mem_op.h . . . . .	2513
17.122 vm.h . . . . .	2514
17.123 arm/l4/sys/vm.h File Reference . . . . .	2514
17.123.1 Detailed Description . . . . .	2515
17.124 vm.h . . . . .	2515
17.125 vm.h . . . . .	2516
17.126 vm.h . . . . .	2516
17.127 amd64/l4/util/bitops_arch.h File Reference . . . . .	2516
17.127.1 Detailed Description . . . . .	2517
17.128 bitops_arch.h . . . . .	2517
17.129 arm/l4/util/bitops_arch.h File Reference . . . . .	2520
17.129.1 Detailed Description . . . . .	2520
17.130 bitops_arch.h . . . . .	2521
17.131 x86/l4/util/bitops_arch.h File Reference . . . . .	2521
17.131.1 Detailed Description . . . . .	2521
17.132 bitops_arch.h . . . . .	2521
17.133 amd64/l4/util/cpu.h File Reference . . . . .	2524
17.133.1 Detailed Description . . . . .	2525
17.134 cpu.h . . . . .	2525
17.135 arm/l4/util/cpu.h File Reference . . . . .	2526
17.135.1 Detailed Description . . . . .	2526
17.136 cpu.h . . . . .	2526
17.137 x86/l4/util/cpu.h File Reference . . . . .	2527
17.137.1 Detailed Description . . . . .	2527
17.138 cpu.h . . . . .	2528
17.139 amd64/l4/util/l4_macros.h File Reference . . . . .	2529
17.139.1 Detailed Description . . . . .	2529
17.140 l4_macros.h . . . . .	2529
17.141 arm/l4/util/l4_macros.h File Reference . . . . .	2529
17.141.1 Detailed Description . . . . .	2530
17.142 l4_macros.h . . . . .	2530
17.143 l4/util/l4_macros.h File Reference . . . . .	2530
17.143.1 Detailed Description . . . . .	2530
17.144 l4_macros.h . . . . .	2531
17.145 x86/l4/util/l4_macros.h File Reference . . . . .	2531
17.145.1 Detailed Description . . . . .	2531
17.146 l4_macros.h . . . . .	2531
17.147 amd64/l4/util/mbi_argv.h File Reference . . . . .	2532



17.147.1 Detailed Description . . . . .	2532
17.148 mbi_argv.h . . . . .	2533
17.149 arm/l4/util/mbi_argv.h File Reference . . . . .	2533
17.149.1 Detailed Description . . . . .	2534
17.150 mbi_argv.h . . . . .	2534
17.151 x86/l4/util/mbi_argv.h File Reference . . . . .	2534
17.151.1 Detailed Description . . . . .	2535
17.152 mbi_argv.h . . . . .	2535
17.153 arm/l4/l4/sys/syscall_defs.h File Reference . . . . .	2536
17.153.1 Detailed Description . . . . .	2536
17.154 syscall_defs.h . . . . .	2536
17.155 contrib/libio-io/l4/io/io.h File Reference . . . . .	2536
17.155.1 Function Documentation . . . . .	2538
17.155.1.1 l4io_get_root_device() . . . . .	2538
17.155.1.2 l4io_iterate_devices() . . . . .	2538
17.155.1.3 l4io_request_all_ioports() . . . . .	2538
17.155.1.4 l4io_request_icu() . . . . .	2539
17.156 io.h . . . . .	2539
17.157 l4/cxx/alloc.h File Reference . . . . .	2540
17.157.1 Detailed Description . . . . .	2540
17.158 alloc.h . . . . .	2541
17.159 arith . . . . .	2541
17.160 l4/cxx/avl_map File Reference . . . . .	2542
17.160.1 Detailed Description . . . . .	2543
17.161 avl_map . . . . .	2543
17.162 l4/cxx/avl_set File Reference . . . . .	2544
17.162.1 Detailed Description . . . . .	2546
17.163 avl_set . . . . .	2546
17.164 l4/cxx/avl_tree File Reference . . . . .	2549
17.164.1 Detailed Description . . . . .	2551
17.165 avl_tree . . . . .	2552
17.166 l4/cxx/basic_ostream File Reference . . . . .	2556
17.166.1 Detailed Description . . . . .	2556
17.167 basic_ostream . . . . .	2557
17.168 l4/cxx/basic_vector.h File Reference . . . . .	2560
17.168.1 Detailed Description . . . . .	2560
17.169 basic_vector.h . . . . .	2560
17.170 bitfield . . . . .	2561
17.171 bitmap . . . . .	2563
17.172 l4/cxx/bits/bst.h File Reference . . . . .	2566
17.172.1 Detailed Description . . . . .	2567
17.173 bst.h . . . . .	2568

17.174 l4/cxx/bits/bst_base.h File Reference . . . . .	2570
17.174.1 Detailed Description . . . . .	2572
17.175 bst_base.h . . . . .	2572
17.176 l4/cxx/bits/bst_iter.h File Reference . . . . .	2573
17.176.1 Detailed Description . . . . .	2574
17.177 bst_iter.h . . . . .	2575
17.178 list_basics.h . . . . .	2576
17.179 l4/cxx/bits/smart_ptr_list.h File Reference . . . . .	2578
17.179.1 Detailed Description . . . . .	2579
17.180 smart_ptr_list.h . . . . .	2580
17.181 type_traits.h . . . . .	2581
17.182 dlist . . . . .	2584
17.183 l4/cxx/exceptions File Reference . . . . .	2589
17.183.1 Detailed Description . . . . .	2591
17.184 exceptions . . . . .	2591
17.185 hlist . . . . .	2593
17.186 l4/cxx/iostream File Reference . . . . .	2595
17.186.1 Detailed Description . . . . .	2596
17.187 iostream . . . . .	2596
17.188 l4/cxx/ipc_helper File Reference . . . . .	2596
17.188.1 Detailed Description . . . . .	2597
17.189 ipc_helper . . . . .	2598
17.190 l4/cxx/ipc_stream File Reference . . . . .	2598
17.190.1 Detailed Description . . . . .	2601
17.190.2 Function Documentation . . . . .	2601
17.190.2.1 operator<<() [1/4] . . . . .	2601
17.190.2.2 operator<<() [2/4] . . . . .	2601
17.190.2.3 operator<<() [3/4] . . . . .	2602
17.190.2.4 operator<<() [4/4] . . . . .	2603
17.190.2.5 operator>>() [1/6] . . . . .	2604
17.190.2.6 operator>>() [2/6] . . . . .	2604
17.190.2.7 operator>>() [3/6] . . . . .	2605
17.190.2.8 operator>>() [4/6] . . . . .	2606
17.190.2.9 operator>>() [5/6] . . . . .	2607
17.190.2.10 operator>>() [6/6] . . . . .	2607
17.191 ipc_stream . . . . .	2608
17.192 ipc_timeout_queue . . . . .	2616
17.193 l4/cxx/l4iostream File Reference . . . . .	2618
17.193.1 Detailed Description . . . . .	2618
17.194 l4iostream . . . . .	2619
17.195 l4/cxx/l4types.h File Reference . . . . .	2619
17.195.1 Detailed Description . . . . .	2620

17.196 l4types.h . . . . .	2620
17.197 list . . . . .	2620
17.198 list_alloc . . . . .	2624
17.199 l4/cxx/lock_guard.h File Reference . . . . .	2630
17.199.1 Detailed Description . . . . .	2630
17.200 lock_guard.h . . . . .	2631
17.201 l4/cxx/main_thread File Reference . . . . .	2632
17.201.1 Detailed Description . . . . .	2632
17.202 main_thread . . . . .	2633
17.203 minmax . . . . .	2633
17.204 numeric . . . . .	2634
17.205 observer . . . . .	2634
17.206 l4/cxx/pair File Reference . . . . .	2635
17.206.1 Detailed Description . . . . .	2636
17.207 pair . . . . .	2636
17.208 ref_ptr . . . . .	2637
17.209 l4/cxx/ref_ptr_list File Reference . . . . .	2640
17.209.1 Detailed Description . . . . .	2642
17.210 ref_ptr_list . . . . .	2642
17.211 slab_alloc . . . . .	2642
17.212 slist . . . . .	2646
17.213 static_container . . . . .	2649
17.214 static_vector . . . . .	2649
17.215 std_alloc . . . . .	2650
17.216 std_ops . . . . .	2650
17.217 string . . . . .	2651
17.218 l4/cxx/string.h File Reference . . . . .	2654
17.218.1 Detailed Description . . . . .	2655
17.219 string.h . . . . .	2655
17.220 type_list . . . . .	2655
17.221 type_traits . . . . .	2656
17.222 unique_ptr . . . . .	2660
17.223 l4/cxx/unique_ptr_list File Reference . . . . .	2662
17.223.1 Detailed Description . . . . .	2664
17.224 unique_ptr_list . . . . .	2664
17.225 utils . . . . .	2664
17.226 weak_ref . . . . .	2665
17.227 amd64/l4/util/irq.h File Reference . . . . .	2666
17.227.1 Detailed Description . . . . .	2667
17.228 irq.h . . . . .	2667
17.229 arm/l4/util/irq.h File Reference . . . . .	2668
17.229.1 Detailed Description . . . . .	2668

17.230 irq.h	2668
17.231 l4/irq/irq.h File Reference	2669
17.231.1 Detailed Description	2670
17.232 irq.h	2671
17.233 l4/sys/irq.h File Reference	2672
17.233.1 Detailed Description	2673
17.234 irq.h	2673
17.235 x86/l4/util/irq.h File Reference	2675
17.235.1 Detailed Description	2676
17.236 irq.h	2676
17.237 backend	2677
17.238 default_ops_impl.h	2680
17.239 fd_store.h	2681
17.240 fd_store_impl.h	2682
17.241 ns_fs.h	2682
17.242 ns_fs_impl.h	2683
17.243 ro_file.h	2688
17.244 ro_file_impl.h	2688
17.245 vcon_stream.h	2690
17.246 vcon_stream_impl.h	2690
17.247 vfs_impl.h	2692
17.248 vfs.h	2705
17.249 virtio-net	2714
17.250 l4virtio	2716
17.251 l4virtio	2719
17.252 l4virtio	2721
17.253 virtio	2732
17.254 virtio-block	2736
17.255 virtio-block	2739
17.256 virtio-console	2746
17.257 virtio-console-device	2752
17.258 virtio-gpio-device	2755
17.259 virtio-i2c-device	2761
17.260 virtio-rng-device	2765
17.261 virtio-scmi-device	2768
17.262 virtio.h	2777
17.263 virtio_block.h	2780
17.264 virtio_input.h	2781
17.265 virtqueue	2782
17.266 block_device_mgr.h	2786
17.267 device.h	2792
17.268 errand.h	2793

17.269 gpt.h . . . . .	2795
17.270 inout_memory.h . . . . .	2795
17.271 part_device.h . . . . .	2797
17.272 partition.h . . . . .	2799
17.273 scheduler.h . . . . .	2802
17.274 l4/sys/scheduler.h File Reference . . . . .	2805
17.274.1 Detailed Description . . . . .	2807
17.275 scheduler.h . . . . .	2808
17.276 types.h . . . . .	2810
17.277 types.h . . . . .	2811
17.278 l4/sys/types.h File Reference . . . . .	2812
17.278.1 Detailed Description . . . . .	2814
17.278.2 Function Documentation . . . . .	2814
17.278.2.1 l4_capability_next() . . . . .	2814
17.279 types.h . . . . .	2815
17.280 virtio_client.h . . . . .	2817
17.281 l4/libedid/edid.h File Reference . . . . .	2826
17.282 edid.h . . . . .	2827
17.283 l4/libgfxbitmap/bitmap.h File Reference . . . . .	2827
17.283.1 Detailed Description . . . . .	2829
17.283.2 Typedef Documentation . . . . .	2829
17.283.2.1 gfxbitmap_color_pix_t . . . . .	2829
17.283.2.2 gfxbitmap_color_t . . . . .	2829
17.283.3 Function Documentation . . . . .	2830
17.283.3.1 gfxbitmap_bmap() . . . . .	2830
17.283.3.2 gfxbitmap_convert_color() . . . . .	2830
17.283.3.3 gfxbitmap_copy() . . . . .	2831
17.283.3.4 gfxbitmap_fill() . . . . .	2831
17.283.3.5 gfxbitmap_set() . . . . .	2832
17.284 bitmap.h . . . . .	2832
17.285 l4/libgfxbitmap/font.h File Reference . . . . .	2833
17.285.1 Detailed Description . . . . .	2835
17.285.2 Enumeration Type Documentation . . . . .	2835
17.285.2.1 anonymous enum . . . . .	2835
17.285.3 Function Documentation . . . . .	2835
17.285.3.1 gfxbitmap_font_data() . . . . .	2835
17.285.3.2 gfxbitmap_font_get() . . . . .	2835
17.285.3.3 gfxbitmap_font_height() . . . . .	2836
17.285.3.4 gfxbitmap_font_init() . . . . .	2836
17.285.3.5 gfxbitmap_font_text() . . . . .	2836
17.285.3.6 gfxbitmap_font_text_scale() . . . . .	2837
17.285.3.7 gfxbitmap_font_width() . . . . .	2838

17.286 font.h . . . . .	2838
17.287 l4/libgfxbitmap/support File Reference . . . . .	2839
17.287.1 Detailed Description . . . . .	2839
17.288 support . . . . .	2840
17.289 l4/re/c/dataspace.h File Reference . . . . .	2840
17.289.1 Detailed Description . . . . .	2841
17.290 dataspace.h . . . . .	2842
17.291 l4/re/c/dma_space.h File Reference . . . . .	2843
17.291.1 Detailed Description . . . . .	2844
17.291.2 Enumeration Type Documentation . . . . .	2844
17.291.2.1 l4re_dma_space_direction . . . . .	2844
17.291.2.2 l4re_dma_space_space_attrbs . . . . .	2844
17.292 dma_space.h . . . . .	2845
17.293 l4/re/c/event.h File Reference . . . . .	2845
17.293.1 Detailed Description . . . . .	2847
17.294 event.h . . . . .	2847
17.295 l4/re/event.h File Reference . . . . .	2848
17.295.1 Detailed Description . . . . .	2848
17.296 event.h . . . . .	2848
17.297 event_buffer.h . . . . .	2849
17.298 l4/re/c/inhibitor.h File Reference . . . . .	2850
17.298.1 Detailed Description . . . . .	2851
17.298.2 Function Documentation . . . . .	2851
17.298.2.1 l4re_inhibitor_acquire() . . . . .	2851
17.298.2.2 l4re_inhibitor_next_lock_info() . . . . .	2851
17.298.2.3 l4re_inhibitor_release() . . . . .	2852
17.299 inhibitor.h . . . . .	2853
17.300 l4/re/c/log.h File Reference . . . . .	2853
17.300.1 Detailed Description . . . . .	2854
17.301 log.h . . . . .	2854
17.302 l4/re/c/mem_alloc.h File Reference . . . . .	2855
17.302.1 Detailed Description . . . . .	2855
17.303 mem_alloc.h . . . . .	2856
17.304 l4/re/c/namespace.h File Reference . . . . .	2856
17.304.1 Detailed Description . . . . .	2857
17.305 namespace.h . . . . .	2858
17.306 l4/re/c/parent.h File Reference . . . . .	2858
17.306.1 Detailed Description . . . . .	2859
17.306.2 Function Documentation . . . . .	2859
17.306.2.1 l4re_parent_signal() . . . . .	2859
17.307 parent.h . . . . .	2860
17.308 l4/re/c/rm.h File Reference . . . . .	2860

17.308.1 Detailed Description . . . . .	2862
17.309 rm.h . . . . .	2862
17.310 l4/re/c/util/cap_alloc.h File Reference . . . . .	2864
17.310.1 Detailed Description . . . . .	2865
17.311 cap_alloc.h . . . . .	2865
17.312 l4/re/c/util/kumem_alloc.h File Reference . . . . .	2866
17.312.1 Detailed Description . . . . .	2866
17.312.2 Function Documentation . . . . .	2867
17.312.2.1 l4re_util_kumem_alloc() . . . . .	2867
17.313 kumem_alloc.h . . . . .	2867
17.314 l4/re/c/util/video/goos_fb.h File Reference . . . . .	2868
17.314.1 Detailed Description . . . . .	2868
17.315 goos_fb.h . . . . .	2868
17.316 l4/re/c/video/colors.h File Reference . . . . .	2869
17.316.1 Detailed Description . . . . .	2870
17.317 colors.h . . . . .	2871
17.318 l4/re/c/video/goos.h File Reference . . . . .	2871
17.318.1 Detailed Description . . . . .	2873
17.319 goos.h . . . . .	2873
17.320 l4/re/c/video/view.h File Reference . . . . .	2874
17.320.1 Detailed Description . . . . .	2876
17.321 view.h . . . . .	2876
17.322 console . . . . .	2877
17.323 l4/re/dataspace File Reference . . . . .	2878
17.323.1 Detailed Description . . . . .	2879
17.324 dataspace . . . . .	2879
17.325 l4/re/dataspace-sys.h File Reference . . . . .	2881
17.325.1 Detailed Description . . . . .	2881
17.326 dataspace-sys.h . . . . .	2882
17.327 dbg_events . . . . .	2882
17.328 l4/re/dma_space File Reference . . . . .	2882
17.329 dma_space . . . . .	2884
17.330 l4/re/elf_aux.h File Reference . . . . .	2885
17.330.1 Detailed Description . . . . .	2886
17.331 elf_aux.h . . . . .	2886
17.332 l4/re/env File Reference . . . . .	2887
17.332.1 Detailed Description . . . . .	2888
17.333 env . . . . .	2888
17.334 l4/re/env.h File Reference . . . . .	2889
17.334.1 Detailed Description . . . . .	2891
17.334.2 Typedef Documentation . . . . .	2891
17.334.2.1 l4re_env_t . . . . .	2891

17.335 env.h . . . . .	2891
17.336 l4/re/error_helper File Reference . . . . .	2893
17.336.1 Detailed Description . . . . .	2894
17.337 error_helper . . . . .	2894
17.338 event-sys.h . . . . .	2896
17.339 event_enums.h . . . . .	2896
17.340 l4/re/impl/dataspace_impl.h File Reference . . . . .	2902
17.340.1 Detailed Description . . . . .	2903
17.341 dataspace_impl.h . . . . .	2903
17.342 l4/re/impl/mem_alloc_impl.h File Reference . . . . .	2905
17.342.1 Detailed Description . . . . .	2905
17.343 mem_alloc_impl.h . . . . .	2906
17.344 l4/re/impl/namespace_impl.h File Reference . . . . .	2906
17.344.1 Detailed Description . . . . .	2907
17.345 namespace_impl.h . . . . .	2907
17.346 l4/re/impl/rm_impl.h File Reference . . . . .	2909
17.346.1 Detailed Description . . . . .	2909
17.347 rm_impl.h . . . . .	2910
17.348 inhibitor . . . . .	2911
17.349 inhibitor-sys.h . . . . .	2911
17.350 itas . . . . .	2912
17.351 l4/re/l4aux.h File Reference . . . . .	2913
17.351.1 Detailed Description . . . . .	2913
17.352 l4aux.h . . . . .	2914
17.353 l4/re/log File Reference . . . . .	2914
17.353.1 Detailed Description . . . . .	2915
17.354 log . . . . .	2915
17.355 l4/re/log-sys.h File Reference . . . . .	2916
17.355.1 Detailed Description . . . . .	2916
17.356 log-sys.h . . . . .	2916
17.357 l4/re/mem_alloc File Reference . . . . .	2916
17.357.1 Detailed Description . . . . .	2918
17.358 mem_alloc . . . . .	2918
17.359 l4/re/mem_alloc-sys.h File Reference . . . . .	2919
17.359.1 Detailed Description . . . . .	2919
17.360 mem_alloc-sys.h . . . . .	2919
17.361 l4/re/mmio_space File Reference . . . . .	2920
17.361.1 Detailed Description . . . . .	2920
17.362 mmio_space . . . . .	2921
17.363 l4/re/namespace File Reference . . . . .	2921
17.363.1 Detailed Description . . . . .	2923
17.364 namespace . . . . .	2923



17.365 l4/re/namespace-sys.h File Reference	2924
17.365.1 Detailed Description	2924
17.366 namespace-sys.h	2924
17.367 l4/re/parent File Reference	2925
17.367.1 Detailed Description	2926
17.368 parent	2926
17.369 l4/re/parent-sys.h File Reference	2926
17.369.1 Detailed Description	2927
17.370 parent-sys.h	2927
17.371 l4/re/protocols.h File Reference	2927
17.371.1 Detailed Description	2927
17.372 protocols.h	2928
17.373 l4/re/random File Reference	2928
17.373.1 Detailed Description	2929
17.374 random	2930
17.375 remote_access	2930
17.376 l4/re/rm File Reference	2931
17.376.1 Detailed Description	2932
17.377 rm	2932
17.378 l4/re/rm-sys.h File Reference	2936
17.378.1 Detailed Description	2937
17.379 rm-sys.h	2937
17.380 l4/re/util/bitmap_cap_alloc File Reference	2937
17.380.1 Detailed Description	2938
17.381 bitmap_cap_alloc	2938
17.382 br_manager	2939
17.383 l4/re/util/cap File Reference	2941
17.383.1 Detailed Description	2942
17.384 cap	2943
17.385 l4/re/cap_alloc File Reference	2943
17.385.1 Detailed Description	2944
17.386 cap_alloc	2944
17.387 l4/re/util/cap_alloc File Reference	2946
17.387.1 Detailed Description	2948
17.388 cap_alloc	2948
17.389 l4/re/util/cap_alloc_impl.h File Reference	2949
17.389.1 Detailed Description	2950
17.390 cap_alloc_impl.h	2950
17.391 l4/re/util/counting_cap_alloc File Reference	2951
17.391.1 Detailed Description	2953
17.392 counting_cap_alloc	2953
17.393 dataspace_svr	2956

17.394 I4/re/debug File Reference . . . . .	2959
17.394.1 Detailed Description . . . . .	2960
17.395 debug . . . . .	2961
17.396 debug . . . . .	2961
17.397 env_ns . . . . .	2963
17.398 event . . . . .	2964
17.399 I4/re/util/event File Reference . . . . .	2966
17.400 event . . . . .	2967
17.401 event_buffer . . . . .	2969
17.402 event_svr . . . . .	2970
17.403 icu_svr . . . . .	2971
17.404 I4/re/util/item_alloc File Reference . . . . .	2973
17.404.1 Detailed Description . . . . .	2975
17.405 item_alloc . . . . .	2975
17.406 I4/re/util/kumem_alloc File Reference . . . . .	2976
17.406.1 Detailed Description . . . . .	2977
17.407 kumem_alloc . . . . .	2977
17.408 name_space_svr . . . . .	2977
17.409 object_registry . . . . .	2983
17.410 poll_timeout_kipclock . . . . .	2985
17.411 I4/re/util/region_mapping File Reference . . . . .	2986
17.411.1 Detailed Description . . . . .	2987
17.412 region_mapping . . . . .	2987
17.413 region_mapping_svr . . . . .	2993
17.414 I4/re/shared_cap File Reference . . . . .	2996
17.414.1 Detailed Description . . . . .	2998
17.415 shared_cap . . . . .	2998
17.416 I4/re/util/shared_cap File Reference . . . . .	2999
17.416.1 Detailed Description . . . . .	3000
17.417 shared_cap . . . . .	3000
17.418 I4/re/unique_cap File Reference . . . . .	3001
17.418.1 Detailed Description . . . . .	3003
17.419 unique_cap . . . . .	3003
17.420 I4/re/util/unique_cap File Reference . . . . .	3003
17.420.1 Detailed Description . . . . .	3005
17.421 unique_cap . . . . .	3005
17.422 vcon_svr . . . . .	3006
17.423 goos_fb . . . . .	3007
17.424 goos_svr . . . . .	3009
17.425 colors . . . . .	3010
17.426 goos . . . . .	3012
17.427 I4/re/video/goos-sys.h File Reference . . . . .	3015

17.427.1 Detailed Description . . . . .	3015
17.428 goos-sys.h . . . . .	3016
17.429 view . . . . .	3016
17.430 l4/shmc/ringbuf.h File Reference . . . . .	3016
17.430.1 Function Documentation . . . . .	3018
17.430.1.1 l4shmc_rb_attach_receiver() . . . . .	3018
17.430.1.2 l4shmc_rb_attach_sender() . . . . .	3018
17.430.1.3 l4shmc_rb_deinit_buffer() . . . . .	3019
17.430.1.4 l4shmc_rb_init_buffer() . . . . .	3019
17.430.1.5 l4shmc_rb_init_receiver() . . . . .	3020
17.430.1.6 l4shmc_rb_receiver_copy_out() . . . . .	3020
17.430.1.7 l4shmc_rb_receiver_notify_done() . . . . .	3021
17.430.1.8 l4shmc_rb_receiver_read_next_size() . . . . .	3021
17.430.1.9 l4shmc_rb_receiver_wait_for_data() . . . . .	3021
17.430.1.10 l4shmc_rb_sender_alloc_packet() . . . . .	3022
17.430.1.11 l4shmc_rb_sender_commit_packet() . . . . .	3022
17.430.1.12 l4shmc_rb_sender_next_copy_in() . . . . .	3023
17.430.1.13 l4shmc_rb_sender_put_data() . . . . .	3023
17.431 ringbuf.h . . . . .	3024
17.432 l4/shmc/shmc.h File Reference . . . . .	3026
17.432.1 Detailed Description . . . . .	3028
17.433 shmc.h . . . . .	3029
17.434 l4/sigma0/sigma0.h File Reference . . . . .	3031
17.434.1 Detailed Description . . . . .	3032
17.435 sigma0.h . . . . .	3033
17.436 __kernel_object_impl.h . . . . .	3034
17.437 l4/sys/__ktrace-impl.h File Reference . . . . .	3034
17.437.1 Detailed Description . . . . .	3036
17.438 __ktrace-impl.h . . . . .	3036
17.439 __l4_fpage.h . . . . .	3036
17.440 __platform_control-arm.h . . . . .	3041
17.441 __task-arm.h . . . . .	3041
17.442 __timeout.h . . . . .	3042
17.443 l4/sys/__typeinfo.h File Reference . . . . .	3044
17.443.1 Detailed Description . . . . .	3046
17.444 __typeinfo.h . . . . .	3046
17.445 __vcpu-arm.h . . . . .	3056
17.446 l4/sys/__vm-arm.h File Reference . . . . .	3057
17.446.1 Detailed Description . . . . .	3059
17.447 __vm-arm.h . . . . .	3059
17.448 __vm-svm.h . . . . .	3059
17.449 __vm-vmx.h . . . . .	3061

17.450 l4/sys/arm_smccc File Reference . . . . .	3067
17.450.1 Detailed Description . . . . .	3068
17.451 arm_smccc . . . . .	3069
17.452 l4/sys/arm_smccc.h File Reference . . . . .	3069
17.452.1 Detailed Description . . . . .	3070
17.452.2 Function Documentation . . . . .	3070
17.452.2.1 l4_arm_smccc_call() . . . . .	3070
17.453 arm_smccc.h . . . . .	3071
17.454 amd64/l4/sys/cache.h File Reference . . . . .	3072
17.454.1 Detailed Description . . . . .	3073
17.455 cache.h . . . . .	3073
17.456 arm/l4/sys/cache.h File Reference . . . . .	3074
17.456.1 Detailed Description . . . . .	3075
17.457 cache.h . . . . .	3075
17.458 arm64/l4/sys/cache.h File Reference . . . . .	3076
17.458.1 Detailed Description . . . . .	3077
17.459 cache.h . . . . .	3078
17.460 l4/sys/cache.h File Reference . . . . .	3079
17.460.1 Detailed Description . . . . .	3080
17.461 cache.h . . . . .	3081
17.462 x86/l4/sys/cache.h File Reference . . . . .	3081
17.462.1 Detailed Description . . . . .	3082
17.463 cache.h . . . . .	3082
17.464 l4/sys/capability File Reference . . . . .	3082
17.464.1 Detailed Description . . . . .	3084
17.464.2 Macro Definition Documentation . . . . .	3084
17.464.2.1 L4_DISABLE_COPY . . . . .	3084
17.465 capability . . . . .	3085
17.466 l4/sys/compiler.h File Reference . . . . .	3086
17.466.1 Detailed Description . . . . .	3087
17.467 compiler.h . . . . .	3088
17.468 amd64/l4/sys/consts.h File Reference . . . . .	3090
17.468.1 Detailed Description . . . . .	3090
17.469 consts.h . . . . .	3091
17.470 arm/l4/sys/consts.h File Reference . . . . .	3091
17.470.1 Detailed Description . . . . .	3091
17.471 consts.h . . . . .	3092
17.472 arm64/l4/sys/consts.h File Reference . . . . .	3092
17.472.1 Detailed Description . . . . .	3092
17.473 consts.h . . . . .	3093
17.474 l4/re/consts.h File Reference . . . . .	3093
17.474.1 Detailed Description . . . . .	3094

17.474.2 Enumeration Type Documentation	3094
17.474.2.1 anonymous enum	3094
17.475 consts.h	3094
17.476 l4/sys/consts.h File Reference	3095
17.476.1 Detailed Description	3096
17.477 consts.h	3097
17.478 x86/l4/sys/consts.h File Reference	3098
17.478.1 Detailed Description	3099
17.479 consts.h	3099
17.480 capability.h	3099
17.481 l4/re/consts File Reference	3102
17.481.1 Detailed Description	3103
17.482 consts	3103
17.483 consts	3103
17.484 ipc_array	3104
17.485 ipc_basics	3107
17.486 l4/sys/cxx/ipc_client File Reference	3111
17.486.1 Macro Definition Documentation	3113
17.486.1.1 L4_RPC_DEF	3113
17.487 ipc_client	3113
17.488 ipc_epiface	3114
17.489 l4/sys/cxx/ipc_iface File Reference	3118
17.489.1 Detailed Description	3119
17.489.2 Macro Definition Documentation	3120
17.489.2.1 L4_INLINE_RPC	3120
17.489.2.2 L4_INLINE_RPC_NF	3120
17.489.2.3 L4_INLINE_RPC_NF_OP	3121
17.489.2.4 L4_INLINE_RPC_OP	3121
17.489.2.5 L4_RPC	3122
17.489.2.6 L4_RPC_NF	3122
17.489.2.7 L4_RPC_NF_OP	3123
17.489.2.8 L4_RPC_OP	3123
17.490 ipc_iface	3124
17.491 ipc_legacy	3129
17.492 ipc_ret_array	3129
17.493 l4/cxx/ipc_server File Reference	3130
17.493.1 Detailed Description	3131
17.494 ipc_server	3132
17.495 ipc_server	3132
17.496 ipc_server_loop	3136
17.497 ipc_string	3139
17.498 l4/sys/cxx/ipc_types File Reference	3141

17.499 ipc_types	3142
17.500 ipc_varg	3149
17.501 l4/sys/cxx/smart_capability_1x File Reference	3155
17.502 smart_capability_1x	3156
17.503 l4/sys/cxx/types File Reference	3158
17.503.1 Macro Definition Documentation	3159
17.503.1.1 L4_TYPES_FLAGS_OPS_DEF	3159
17.504 types	3159
17.505 l4/sys/debugger File Reference	3162
17.505.1 Detailed Description	3162
17.506 debugger	3163
17.507 l4/sys/debugger.h File Reference	3163
17.507.1 Detailed Description	3165
17.508 debugger.h	3165
17.509 l4/sys/err.h File Reference	3168
17.509.1 Detailed Description	3169
17.510 err.h	3170
17.511 l4/sys/exception File Reference	3170
17.511.1 Detailed Description	3171
17.512 exception	3172
17.513 l4/sys/factory File Reference	3172
17.513.1 Detailed Description	3174
17.514 factory	3174
17.515 l4/sys/factory.h File Reference	3176
17.515.1 Detailed Description	3177
17.516 factory.h	3178
17.517 l4/sys/icu File Reference	3182
17.517.1 Detailed Description	3183
17.518 icu	3184
17.519 l4/sys/icu.h File Reference	3184
17.519.1 Detailed Description	3186
17.520 icu.h	3187
17.521 iommu	3190
17.522 ipc.h	3190
17.523 ipc.h	3191
17.524 ipc.h	3192
17.525 l4/sys/ipc.h File Reference	3192
17.525.1 Detailed Description	3194
17.525.2 Function Documentation	3194
17.525.2.1 l4_ipc_to_errno()	3194
17.526 ipc.h	3195
17.527 x86/l4f/l4/sys/ipc.h File Reference	3198

17.527.1 Detailed Description . . . . .	3199
17.528 ipc.h . . . . .	3199
17.529 l4/sys/ipc_gate File Reference . . . . .	3200
17.529.1 Detailed Description . . . . .	3201
17.530 ipc_gate . . . . .	3201
17.531 l4/sys/ipc_gate.h File Reference . . . . .	3201
17.531.1 Detailed Description . . . . .	3203
17.532 ipc_gate.h . . . . .	3204
17.533 l4/sys/irq File Reference . . . . .	3205
17.533.1 Detailed Description . . . . .	3206
17.534 irq . . . . .	3206
17.535 l4/sys/kdebug.h File Reference . . . . .	3208
17.535.1 Detailed Description . . . . .	3210
17.535.2 Enumeration Type Documentation . . . . .	3210
17.535.2.1 l4_kdebug_ops_t . . . . .	3210
17.535.3 Function Documentation . . . . .	3210
17.535.3.1 __kdebug_3_text() . . . . .	3210
17.535.3.2 __kdebug_op() . . . . .	3212
17.535.3.3 __kdebug_op_1() . . . . .	3213
17.535.3.4 __kdebug_text() . . . . .	3215
17.535.3.5 enter_kdebug() . . . . .	3216
17.535.3.6 outchar() . . . . .	3217
17.535.3.7 outdec() . . . . .	3217
17.535.3.8 outhex12() . . . . .	3218
17.535.3.9 outhex16() . . . . .	3218
17.535.3.10 outhex20() . . . . .	3219
17.535.3.11 outhex32() . . . . .	3220
17.535.3.12 outhex64() . . . . .	3221
17.535.3.13 outhex8() . . . . .	3222
17.535.3.14 outnstring() . . . . .	3222
17.535.3.15 outstring() . . . . .	3223
17.535.3.16 outumword() . . . . .	3224
17.536 kdebug.h . . . . .	3224
17.537 l4/sys/kdump.h File Reference . . . . .	3227
17.537.1 Detailed Description . . . . .	3227
17.537.2 Function Documentation . . . . .	3227
17.537.2.1 fiasco_dump_kmem_stats() . . . . .	3227
17.538 kdump.h . . . . .	3228
17.539 l4/sys/kernel_object.h File Reference . . . . .	3228
17.539.1 Detailed Description . . . . .	3229
17.540 kernel_object.h . . . . .	3229
17.541 l4/sys/kip File Reference . . . . .	3230

17.541.1 Detailed Description . . . . .	3231
17.542 kip . . . . .	3231
17.543 kobject . . . . .	3233
17.544 l4/sys/ktrace.h File Reference . . . . .	3233
17.544.1 Detailed Description . . . . .	3234
17.545 ktrace.h . . . . .	3235
17.546 amd64/l4/sys/l4int.h File Reference . . . . .	3235
17.546.1 Detailed Description . . . . .	3235
17.547 l4int.h . . . . .	3236
17.548 arm/l4/sys/l4int.h File Reference . . . . .	3236
17.548.1 Detailed Description . . . . .	3236
17.549 l4int.h . . . . .	3236
17.550 arm64/l4/sys/l4int.h File Reference . . . . .	3237
17.550.1 Detailed Description . . . . .	3237
17.551 l4int.h . . . . .	3237
17.552 l4/sys/l4int.h File Reference . . . . .	3237
17.552.1 Detailed Description . . . . .	3238
17.553 l4int.h . . . . .	3238
17.554 x86/l4/sys/l4int.h File Reference . . . . .	3239
17.554.1 Detailed Description . . . . .	3239
17.555 l4int.h . . . . .	3239
17.556 l4/sys/memdesc.h File Reference . . . . .	3240
17.556.1 Detailed Description . . . . .	3241
17.557 memdesc.h . . . . .	3241
17.558 meta . . . . .	3243
17.559 l4/sys/meta File Reference . . . . .	3243
17.559.1 Detailed Description . . . . .	3244
17.560 meta . . . . .	3245
17.561 l4/sys/obj_info.h File Reference . . . . .	3245
17.561.1 Detailed Description . . . . .	3246
17.561.2 Function Documentation . . . . .	3246
17.561.2.1 l4_debugger_query_obj_infos() . . . . .	3246
17.562 obj_info.h . . . . .	3247
17.563 l4/sys/pager File Reference . . . . .	3249
17.563.1 Detailed Description . . . . .	3251
17.564 pager . . . . .	3251
17.565 l4/sys/platform_control File Reference . . . . .	3251
17.565.1 Detailed Description . . . . .	3252
17.566 platform_control . . . . .	3253
17.567 platform_control.h . . . . .	3253
17.568 platform_control.h . . . . .	3253
17.569 l4/sys/platform_control.h File Reference . . . . .	3254



17.569.1 Detailed Description . . . . .	3255
17.570 platform_control.h . . . . .	3255
17.571 I4/sys/rcv_endpoint File Reference . . . . .	3257
17.571.1 Detailed Description . . . . .	3259
17.572 rcv_endpoint . . . . .	3259
17.573 I4/sys/rcv_endpoint.h File Reference . . . . .	3259
17.573.1 Detailed Description . . . . .	3261
17.573.2 Enumeration Type Documentation . . . . .	3261
17.573.2.1 L4_rcv_ep_ops . . . . .	3261
17.574 rcv_endpoint.h . . . . .	3261
17.575 I4/sys/scheduler File Reference . . . . .	3262
17.575.1 Detailed Description . . . . .	3263
17.576 scheduler . . . . .	3264
17.577 I4/sys/semaphore File Reference . . . . .	3264
17.577.1 Detailed Description . . . . .	3266
17.578 semaphore . . . . .	3266
17.579 I4/sys/semaphore.h File Reference . . . . .	3266
17.579.1 Detailed Description . . . . .	3268
17.580 semaphore.h . . . . .	3268
17.581 I4/sys/smart_capability File Reference . . . . .	3268
17.581.1 Detailed Description . . . . .	3270
17.582 smart_capability . . . . .	3270
17.583 I4/sys/snd_destination File Reference . . . . .	3272
17.583.1 Detailed Description . . . . .	3273
17.584 snd_destination . . . . .	3274
17.585 I4/sys/snd_destination.h File Reference . . . . .	3274
17.585.1 Detailed Description . . . . .	3275
17.586 snd_destination.h . . . . .	3275
17.587 I4/sys/task File Reference . . . . .	3275
17.587.1 Detailed Description . . . . .	3277
17.588 task . . . . .	3277
17.589 task.h . . . . .	3278
17.590 task.h . . . . .	3278
17.591 I4/sys/task.h File Reference . . . . .	3278
17.591.1 Detailed Description . . . . .	3279
17.592 task.h . . . . .	3280
17.593 I4/cxx/thread File Reference . . . . .	3282
17.593.1 Detailed Description . . . . .	3283
17.594 thread . . . . .	3284
17.595 I4/sys/thread File Reference . . . . .	3284
17.595.1 Detailed Description . . . . .	3286
17.596 thread . . . . .	3286

17.597 l4/sys/thread_group File Reference . . . . .	3288
17.598 thread_group . . . . .	3288
17.599 l4/sys/thread_group.h File Reference . . . . .	3289
17.600 thread_group.h . . . . .	3290
17.601 l4/sys/typeinfo_svr File Reference . . . . .	3291
17.601.1 Detailed Description . . . . .	3292
17.602 typeinfo_svr . . . . .	3292
17.603 amd64/l4/sys/utcb.h File Reference . . . . .	3293
17.603.1 Detailed Description . . . . .	3294
17.604 utcb.h . . . . .	3294
17.605 arm/l4/sys/utcb.h File Reference . . . . .	3295
17.605.1 Detailed Description . . . . .	3297
17.606 utcb.h . . . . .	3297
17.607 arm64/l4/sys/utcb.h File Reference . . . . .	3298
17.607.1 Detailed Description . . . . .	3299
17.608 utcb.h . . . . .	3299
17.609 l4/sys/utcb.h File Reference . . . . .	3300
17.609.1 Detailed Description . . . . .	3302
17.610 utcb.h . . . . .	3302
17.611 x86/l4/sys/utcb.h File Reference . . . . .	3304
17.611.1 Detailed Description . . . . .	3306
17.612 utcb.h . . . . .	3306
17.613 l4/sys/vcon File Reference . . . . .	3307
17.613.1 Detailed Description . . . . .	3309
17.614 vcon . . . . .	3309
17.615 l4/sys/vcon.h File Reference . . . . .	3309
17.615.1 Detailed Description . . . . .	3312
17.615.2 Enumeration Type Documentation . . . . .	3312
17.615.2.1 L4_vcon_read_flags . . . . .	3312
17.616 vcon.h . . . . .	3312
17.617 l4/sys/vcpu_context File Reference . . . . .	3315
17.617.1 Detailed Description . . . . .	3316
17.618 vcpu_context . . . . .	3316
17.619 vcpu_context.h . . . . .	3316
17.620 vm . . . . .	3316
17.621 vm . . . . .	3316
17.622 l4/sys/vm File Reference . . . . .	3317
17.622.1 Detailed Description . . . . .	3318
17.623 vm . . . . .	3318
17.624 l4/sys/assert.h File Reference . . . . .	3318
17.624.1 Detailed Description . . . . .	3319
17.624.2 Macro Definition Documentation . . . . .	3320

17.624.2.1 <code>l4_assert</code> . . . . .	3320
17.625 <code>assert.h</code> . . . . .	3320
17.626 <code>l4/util/assert.h</code> File Reference . . . . .	3321
17.626.1 Detailed Description . . . . .	3322
17.627 <code>assert.h</code> . . . . .	3322
17.628 <code>atomic.h</code> . . . . .	3324
17.629 <code>l4/cxx/atomic.h</code> File Reference . . . . .	3324
17.629.1 Detailed Description . . . . .	3325
17.630 <code>atomic.h</code> . . . . .	3325
17.631 <code>l4/util/atomic.h</code> File Reference . . . . .	3326
17.631.1 Detailed Description . . . . .	3328
17.632 <code>atomic.h</code> . . . . .	3329
17.633 <code>l4/util/backtrace.h</code> File Reference . . . . .	3333
17.633.1 Detailed Description . . . . .	3334
17.633.2 Function Documentation . . . . .	3334
17.633.2.1 <code>l4util_backtrace()</code> . . . . .	3334
17.634 <code>backtrace.h</code> . . . . .	3335
17.635 <code>l4/util/base64.h</code> File Reference . . . . .	3335
17.635.1 Detailed Description . . . . .	3336
17.636 <code>base64.h</code> . . . . .	3337
17.637 <code>l4/util/bitops.h</code> File Reference . . . . .	3337
17.637.1 Detailed Description . . . . .	3339
17.638 <code>bitops.h</code> . . . . .	3339
17.639 <code>l4/util/elf.h</code> File Reference . . . . .	3342
17.639.1 Detailed Description . . . . .	3348
17.640 <code>elf.h</code> . . . . .	3348
17.641 <code>l4/util/getopt.h</code> File Reference . . . . .	3359
17.641.1 Detailed Description . . . . .	3359
17.642 <code>getopt.h</code> . . . . .	3360
17.643 <code>l4/util/keymap.h</code> File Reference . . . . .	3361
17.643.1 Detailed Description . . . . .	3361
17.644 <code>keymap.h</code> . . . . .	3362
17.645 <code>l4/sys/kip.h</code> File Reference . . . . .	3362
17.645.1 Detailed Description . . . . .	3363
17.645.2 Macro Definition Documentation . . . . .	3364
17.645.2.1 <code>l4_kip_for_each_feature</code> . . . . .	3364
17.645.3 Function Documentation . . . . .	3364
17.645.3.1 <code>l4_kip_kernel_has_feature()</code> . . . . .	3364
17.646 <code>kip.h</code> . . . . .	3365
17.647 <code>l4/util/kip.h</code> File Reference . . . . .	3367
17.648 <code>kip.h</code> . . . . .	3367
17.649 <code>l4/util/kprintf.h</code> File Reference . . . . .	3368

17.649.1 Detailed Description . . . . .	3368
17.650 kprintf.h . . . . .	3369
17.651 l4/util/l4mod.h File Reference . . . . .	3369
17.651.1 Detailed Description . . . . .	3370
17.651.2 Enumeration Type Documentation . . . . .	3370
17.651.2.1 l4util_l4mod_mod_info_flag . . . . .	3370
17.652 l4mod.h . . . . .	3370
17.653 l4/util/list_alloc.h File Reference . . . . .	3371
17.653.1 Detailed Description . . . . .	3371
17.653.2 Function Documentation . . . . .	3372
17.653.2.1 l4la_alloc() . . . . .	3372
17.653.2.2 l4la_avail() . . . . .	3372
17.653.2.3 l4la_dump() . . . . .	3372
17.653.2.4 l4la_free() . . . . .	3372
17.653.2.5 l4la_init() . . . . .	3373
17.654 list_alloc.h . . . . .	3373
17.655 l4/util/lock.h File Reference . . . . .	3374
17.655.1 Detailed Description . . . . .	3374
17.656 lock.h . . . . .	3375
17.657 l4/util/mb_info.h File Reference . . . . .	3375
17.657.1 Detailed Description . . . . .	3378
17.657.2 Macro Definition Documentation . . . . .	3378
17.657.2.1 l4util_mb_for_each_mmap_entry . . . . .	3378
17.657.2.2 L4UTIL_MB_MEMORY . . . . .	3378
17.657.2.3 MB_ART_MEMORY . . . . .	3378
17.658 mb_info.h . . . . .	3379
17.659 l4/util/parse_cmd.h File Reference . . . . .	3383
17.659.1 Detailed Description . . . . .	3384
17.660 parse_cmd.h . . . . .	3384
17.661 printf_helpers.h . . . . .	3385
17.662 l4/util/rand.h File Reference . . . . .	3385
17.662.1 Detailed Description . . . . .	3386
17.663 rand.h . . . . .	3386
17.664 l4/util/splitlog2.h File Reference . . . . .	3386
17.664.1 Detailed Description . . . . .	3387
17.665 splitlog2.h . . . . .	3387
17.666 arm/l4/sys/thread.h File Reference . . . . .	3388
17.666.1 Detailed Description . . . . .	3388
17.667 thread.h . . . . .	3389
17.668 arm64/l4/sys/thread.h File Reference . . . . .	3389
17.668.1 Detailed Description . . . . .	3390
17.669 thread.h . . . . .	3390

17.670 l4/sys/thread.h File Reference . . . . .	3390
17.670.1 Detailed Description . . . . .	3393
17.671 thread.h . . . . .	3393
17.672 l4/util/thread.h File Reference . . . . .	3400
17.672.1 Detailed Description . . . . .	3401
17.672.2 Macro Definition Documentation . . . . .	3401
17.672.2.1 __L4UTIL_THREAD_FUNC . . . . .	3401
17.673 thread.h . . . . .	3402
17.674 util.h . . . . .	3402
17.675 vbus . . . . .	3403
17.676 l4/vbus/vbus.h File Reference . . . . .	3405
17.676.1 Detailed Description . . . . .	3407
17.676.2 Enumeration Type Documentation . . . . .	3407
17.676.2.1 anonymous enum . . . . .	3407
17.676.2.2 l4vbus_icu_src_types . . . . .	3407
17.677 vbus.h . . . . .	3407
17.678 vbus_generic . . . . .	3408
17.679 vbus_gpio . . . . .	3409
17.680 vbus_gpio-ops.h . . . . .	3410
17.681 vbus_gpio.h . . . . .	3410
17.682 vbus_i2c.h . . . . .	3411
17.683 vbus_inhibitor.h . . . . .	3412
17.684 l4/vbus/vbus_interfaces.h File Reference . . . . .	3412
17.684.1 Detailed Description . . . . .	3413
17.684.2 Typedef Documentation . . . . .	3413
17.684.2.1 l4vbus_iface_type_t . . . . .	3413
17.684.3 Enumeration Type Documentation . . . . .	3414
17.684.3.1 anonymous enum . . . . .	3414
17.684.3.2 l4vbus_iface_type_t . . . . .	3414
17.684.4 Function Documentation . . . . .	3414
17.684.4.1 l4vbus_subinterface_supported() . . . . .	3414
17.685 vbus_interfaces.h . . . . .	3415
17.686 vbus_mcspi.h . . . . .	3415
17.687 vbus_pci . . . . .	3416
17.688 vbus_pci-ops.h . . . . .	3417
17.689 vbus_pci.h . . . . .	3417
17.690 vbus_pm-ops.h . . . . .	3418
17.691 vbus_pm.h . . . . .	3418
17.692 l4/vbus/vbus_types.h File Reference . . . . .	3418
17.692.1 Detailed Description . . . . .	3420
17.692.2 Enumeration Type Documentation . . . . .	3420
17.692.2.1 l4vbus_device_flags_t . . . . .	3420

17.692.2.2 l4vbus_resource_flags_t	3420
17.692.2.3 l4vbus_resource_type_t	3421
17.693 vbus_types.h	3421
17.694 vdevice-ops.h	3422
17.695 l4/vcpu/vcpu File Reference	3423
17.695.1 Detailed Description	3423
17.696 vcpu	3423
17.697 l4/sys/vcpu.h File Reference	3425
17.697.1 Detailed Description	3426
17.697.2 Function Documentation	3427
17.697.2.1 l4_vcpu_check_version()	3427
17.698 vcpu.h	3428
17.699 l4/vcpu/vcpu.h File Reference	3429
17.699.1 Detailed Description	3430
17.700 vcpu.h	3430
17.701 ipc-invoke.h	3432
17.702 ipc-l42-gcc3-nopic.h	3433
<b>18 Examples</b>	<b>3435</b>
18.1 hello/server/src/main.c	3435
18.2 examples/sys/ipc/ipc_example.c	3435
18.3 examples/sys/ipc/ipc.cfg	3437
18.4 examples/sys/start-with-exc/main.c	3437
18.5 examples/sys/singlestep/main.c	3439
18.6 examples/sys/aliens/main.c	3441
18.7 examples/sys/utcb-ipc/main.c	3445
18.8 examples/sys/isr/main.c	3447
18.9 examples/clntsrv/src/server.cc	3449
18.10 examples/clntsrv/src/client.cc	3449
18.11 examples/clntsrv/src/shared.h	3450
18.12 examples/clntsrv/configs/clntsrv.cfg	3451
18.13 examples/libs/l4re/c/ma+rm.c	3451
18.14 examples/libs/l4re/c++/mem_alloc/ma+rm.cc	3452
18.15 examples/libs/l4re/c++/shared_ds/ds_clnt.cc	3453
18.16 examples/libs/l4re/c++/shared_ds/ds_srv.cc	3455
18.17 examples/libs/l4re/c++/shared_ds/shared_ds.cfg	3457
18.18 examples/libs/l4re/streammap/server.cc	3457
18.19 examples/libs/l4re/streammap/client.cc	3458
18.20 examples/libs/l4re/streammap/streammap.cfg	3459
18.21 examples/libs/libirq/loop.c	3460
18.22 examples/libs/libirq/async_isr.c	3461
18.23 examples/sys/migrate/thread_migrate.cc	3461

18.24 <a href="#">examples/sys/migrate/thread_migrate.cfg</a> . . . . .	3463
18.25 <a href="#">tmpfs/lib/src/fs.cc</a> . . . . .	3463
18.26 <a href="#">examples/libs/shmc/prodcons.c</a> . . . . .	3470
<b>Index</b>	<b>3473</b>





# Chapter 1

## Overview

Welcome to the documentation of the L4Re Operating System Framework, or L4Re for short. There are two parts to this documentation: a user manual, which provides a birds eye view of L4Re and its environment, and a reference section which documents the complete programming API.

### User Manual

1. [Introduction](#) shortly explains the concept of microkernels and introduces the basic terminology.
2. [Tutorial](#) helps you getting started with setting up the development environment and writing your own first L4Re application.
3. [Programming for L4Re](#) explains in detail the most important programming concepts.
4. [L4Re Servers](#) provides a quick overview over standard services running on the L4Re operating system.

### Reference

The second part provides the complete reference of all classes and functions of the L4Re Operating System Framework as well as a list of example code.



# Chapter 2

## Introduction

The intention of this section is to provide a short overview about the [L4Re](#) Operating System Framework.

The general structure of a microkernel-based system will be introduced and the principal functionality of the servers in the basic environment outlined.

### 2.1 L4Re Microkernel

The [L4Re](#) Microkernel is the lowest-level component of software running in an L4Re-based system. The microkernel is the only component that runs in privileged processor mode. It does not include complex services such as program loading, device drivers, or file systems; those are implemented in user-level programs on top of it (a basic set of these services and abstractions is provided by the [L4](#) Runtime Environment).

Microkernel services are implemented in kernel objects. Tasks hold references to kernel objects in their respective *"object space"*, which is a kernel-protected table. These references are called *capabilities*. System calls to the microkernel are function invocations on kernel objects through the corresponding capabilities. These can be thought of as function invocations on object references in an object-oriented programming environment. Furthermore, if a task owns a capability, it may grant other tasks the same (or fewer) rights on this object by passing the capability from its own to the other task's object space.

From a design perspective, capabilities are a concept that enables flexibility in the system structure. A thread that invokes an object through a capability does not need to care about where this object is implemented. In fact, it is possible to implement all objects either in the kernel or in a user-level server and replace one implementation with the other transparently for clients.

#### 2.1.1 Communication

The basic communication mechanism in L4-based systems is called *"Inter Process Communication (IPC)"*. It is always synchronous, i.e. both communication partners need to actively rendezvous for IPC. In addition to transmitting arbitrary data between threads, IPC is also used to resolve hardware exceptions, faults and for virtual memory management.

### 2.1.2 Kernel Objects

The following list gives a short overview of the kernel objects provided by the [L4Re](#) Microkernel:

- **Task** A task comprises a memory address space (represented by the task's page table), an object space (holding the kernel protected capabilities), and on x86 an IO-port address space.
- **Thread** A thread is bound to a task and executes code. Multiple threads can coexist in one task and are scheduled by the microkernel's scheduler.
- **Factory** A factory is used by applications to create new kernel objects. Access to a factory is required to create any new kernel object. Factories can control and restrict object creation.
- **IPC Gate** An IPC gate is used to create a secure communication channel between different tasks. It embeds a label (kernel protected payload) that securely identifies the gate through which a message is received. The gate label is not visible to and cannot be altered by the sender.
- **IRQ** IRQ objects provide access to hardware interrupts. Additionally, programs can create new virtual interrupt objects and trigger them. This allows to implement a signaling mechanism. The receiver cannot decide whether the interrupt is a physical or virtual one.
- **Vcon** Provides access to the in-kernel debugging console (input and output). There is only one such object in the kernel and it is only available, if the kernel is built with debugging enabled. This object is typically interposed through a user-level service or without debugging in the kernel can be completely based on user-level services.
- **Scheduler** Implements scheduling policy and assignment of threads to CPUs, including CPU statistics.

## 2.2 L4Re System Structure

The system has a multi-tier architecture consisting of the following layers depicted in the figure below:

- **Microkernel** The microkernel is the component at the lowest level of the software stack. It is the only piece of software that is running in the privileged mode of the processor.
- **Tasks** Tasks are the basic containers (address spaces) in which system services and applications are executed. They run in the processor's deprivileged user mode.

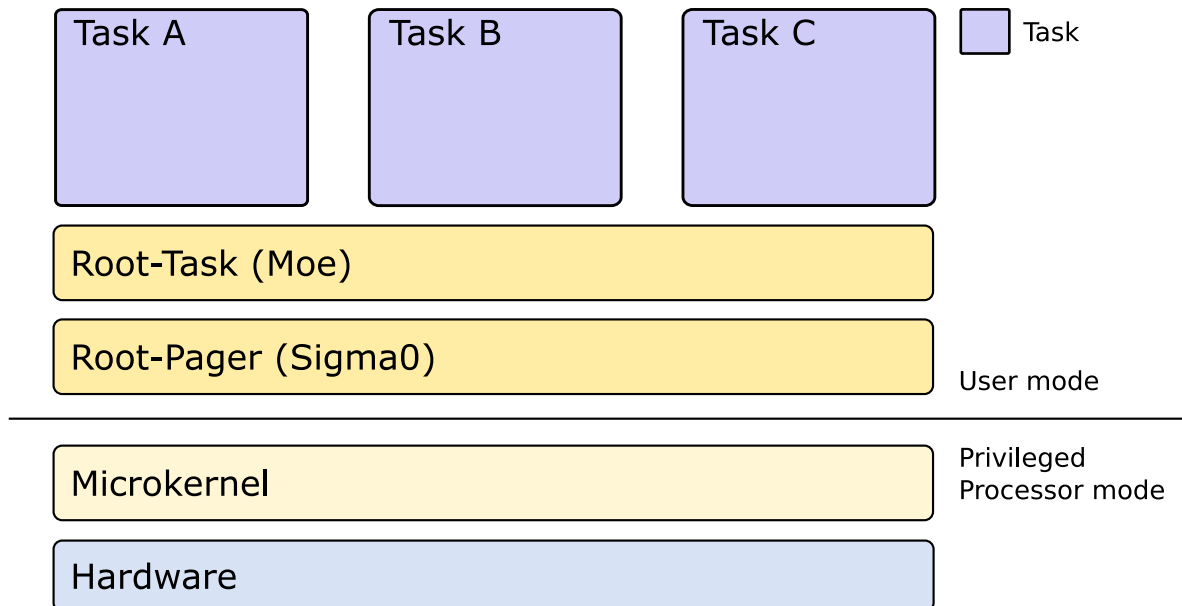


Figure 2.1 Basic Structure of an L4Re based system

In terms of functionality, the system is structured as follows:

- Microkernel** The kernel provides primitives to execute programs in tasks, to enforce isolation among them, and to provide means of secure communication in order to let them cooperate. As the kernel is the most privileged, security-critical software component in the system, it is a general design goal to make it as small as possible in order to reduce its attack surface. It provides only a minimal set of mechanisms that are necessary to support applications.
- Runtime Environment** The small kernel offers a concise set of interfaces, but these are not necessarily suited for building applications directly on top of it. The [L4Re](#) Runtime Environment aims at providing more convenient abstractions for application development. It comprises low-level software components that interface directly with the microkernel. The root pager *sigma0* and the root task *Moe* are the most basic components of the [L4Re](#) Runtime Environment. Other services (e.g., for device enumeration) use interfaces provided by them.
- Applications** Applications run on top of the system and use services provided by the runtime environment – or by other applications. There may be several types of applications in the system and even virtual machine monitors and device drivers are considered applications in the terminology used in this document. They are running alongside other applications on the system.

Lending terminology from the distributed systems area, applications offering services to other applications are usually called *servers*, whereas applications using those services are named *clients*. Being in both roles is also common, for instance, a file system server may be viewed as a server with respect to clients using the file system, while the server itself may also act as a client of a hard disk driver.

## 2.3 L4Re Runtime Environment

The [L4Re](#) Runtime Environment provides a basic set of services and abstractions, which are useful to implement and run user-level applications on top of the [L4Re](#) Microkernel. They form the [L4Re](#) Operating System Framework.

The [L4Re](#) Operating System Framework consists of a set of libraries and servers. [L4Re](#) follows an object-oriented design. Server interfaces are object-oriented, and the implementation is also object-oriented.

A minimal L4Re-based application needs 3 components to be booted beforehand: the [L4Re](#) Microkernel, the root pager (Sigma0), and the root task (Moe). The Sigma0 root pager initially owns all system resources, but is usually used only to resolve page faults for the Moe root task. Moe provides the essential services to normal user applications such as an initial program loader, a region-map service for virtual memory management, and a memory (data space) allocator.

# Chapter 3

## Tutorial

This tutorial assumes that the reader is familiar with the basic L4 concepts that were discussed in the [Introduction](#) section.

Here you can find the first steps to boot a very simple setup. The setup consists of the following components:

- [L4Re](#) Microkernel
- Sigma0 — Root Pager
- Moe — Root Task
- Ned — Init Process
- hello — The classical 'Hello World' Application

The guide assumes that you already compiled the base components and describes how to generate an ISO image, with GRUB as a boot loader, that can for example be booted within QEMU.

First you need a `modules.list` file that contains an entry for the scenario.

```
modaddr 0x002000000

entry hello
  kernel fiasco -serial_esc
  roottask moe rom/hello.cfg
  module l4re
  module ned
  module hello.cfg
  module hello
```

This file describes all the binaries and scripts to put into the ISO image, and also describes the GRUB `menu.lst` contents. What you need to do is to set the `make` variable `MODULE_SEARCH_PATH` to contain the path to your [L4Re](#) Microkernel's build directory and the directory containing your `hello.cfg` script.

The `hello.cfg` script should look like the following. A ready to use version can be found in `I4/conf/examples`.

```
local L4 = require("L4");
L4.default_loader:start({}, "rom/hello");
```

The first line of this script ensures that the [L4](#) package is available for the script. The second line uses the default loader object defined in that package and starts the binary `rom/hello`.

**Note**

All modules defined in `modules.list` are available as data spaces ([L4Re::Dataspace](#)) and registered in a name space ([L4Re::Namespace](#)). This name space is in turn available as 'rom' to the init process ([Ned](#)).

Now you can go to your [L4Re](#) build directory and run the following command.

**Note**

The example assumes that you have created the `modules.list` and `hello.cfg` files in the `/tmp` directory. Adapt if you created them somewhere else.

```
make grub2iso E=hello MODULES_LIST=/tmp/modules.list MODULE_SEARCH_PATH=/tmp:<path_to_fiasco_builddir>
```

Now you should be able to boot the image in QEMU by running:

```
qemu-system-x86_64 -m 1024 -cdrom images/hello.iso -serial stdio
```

If you press `<ESC>` in the terminal that shows you the serial output you enter the microkernel's debugger... Have fun.

## 3.1 Customizations

A basic set of bootable entries can be found in `l4/conf/modules.list`. This file is the default for any image creation as shown above. It is recommended that local modification regarding image creation are done in `conf/Makeconf.boot`. Initially you may copy `Makeconf.boot.example` to `Makeconf.boot`. You can overwrite `MODULES_LIST` to set your own modules-list file. Set `MODULE_SEARCH_PATH` to your setup according to the examples given in the file. When configured a `make` call is reduced to:

```
make grub2iso E=hello
```

All other local configuration can be done in a `Makeconf.local` file located in the `l4` directory.



## Chapter 4

# Programming for L4Re

This part of the documentation discusses the concept of microkernel-based programming in more detail.

You should already have a basic understanding of the [L4Re](#) programming environment from the tutorial.

- [L4 Inter-Process Communication \(IPC\)](#)
- [Kernel ABI](#)
- [Capabilities and Naming](#)
- [Spaces and Mappings](#)
- [Initial Environment and Application Bootstrapping](#)
- [Memory management - Data Spaces and the Region Map](#)
- [Program Input and Output](#)
- [Initial Memory Allocator and Factory](#)
- [Application and Server Building Blocks](#)
- [Pthread Support](#)
- tasks and threads
- communication channels
- server loops
- [Interface Definition Language](#)
- hardware access
- [L4Re Build System](#)
- [Kernel Factory](#)

### 4.1 L4 Inter-Process Communication (IPC)

Inter-process communication (IPC) is the fundamental communication mechanism in the [L4Re](#) Microkernel.

Basically IPC invokes a subroutine in a different context using input and output parameters. It is used to communicate between userland threads and, as a special case, to communicate between a userland thread and a kernel object. IPC provides the only (non-debugging) way of doing system calls.

### 4.1.1 IPC mechanism

When using this API, an IPC operation can be conducted using the `l4_ipc()` function (or one of its related [helpers](#)). In general it causes a method to be invoked on the called kernel object. An IPC operation consists of a send and receive phase, but either of them can be skipped, that is, an IPC operation can consist of only either a send or a receive phase. IPC is always synchronous and blocking and can be given a timeout. Timeouts can be specified separately for each phase. Invoking the IPC syscall without any phase is deprecated.

On the lowest abstraction level, IPC operations need the following arguments:

- [flags](#) describing the IPC mode,
- the [capability selector](#) of the communication partner,
- a [label](#),
- a [message tag](#), and
- a pair of [timeout](#) values.

During an IPC operation the kernel accesses the UTCB of the current thread to read parameters which are not passed as direct arguments.

As result of an IPC operation the kernel returns a message tag and a label and the kernel also changes UTCB content. For the detailed call signature, refer to `l4_ipc()`. Furthermore, depending on the IPC parameters, the kernel might have transferred the FPU state and capabilities (memory, I/O ports, and/or object capabilities) from the sender to the receiving thread.

The transition between the IPC send phase and the IPC receive phase is atomic, that is, as soon as the send phase has finished, the thread receive phase starts. A relative receive timeout does not start before the send phase has finished (see also below) and a thread which received an IPC call from another thread can assume that the other thread is ready to receive the reply message and the replying thread can therefore reply with a timeout of zero, see [IPC Timeouts](#).

For performance optimization and under certain conditions, the kernel may perform a context switch from the IPC sender to the IPC receiver without consulting the scheduler after the send phase finished. For instance, a client performing an IPC call to a server has to wait for the server anyway. Hence, after the client request was sent to the server, the kernel switches directly to the server thread. This behavior can be disabled by setting the `L4_MSGTAG_SCHEDULE` flag in the sender message tag (see below).

#### 4.1.1.1 IPC Flags

The *flags* defined in `l4_syscall_flags_t` are used by the invoking thread to define the intended IPC operation. The variants of `l4_ipc()` (see [Object Invocation](#)) use the flags

- to request the IPC phases (send-only IPC, receive-only IPC, IPC with send and receive phase), and
- to decide, if the reply capability (see [below](#)) should be used instead of the capability of a dedicated kernel object as target for the send phase (*reply*), and
- to decide, if receiving should wait for an incoming message from any possible sender (*open wait*) instead of a message from a dedicated sender (*closed wait*).

#### 4.1.1.2 Partner capability selector

The *partner capability selector* defines a kernel object as partner of the IPC operation. Some kernel objects forward IPC to a userland thread.

Basically an object capability is represented by `l4_cap_idx_t` where the bits starting from `L4_CAP_SHIFT` are used as index into the local capability table of the current address space.

Specifying `L4_INVALID_CAP` as target for an IPC operation is equivalent to specifying a thread capability of the current thread with full permissions. As a result, the userland thread either communicates with its corresponding kernel thread object (if `L4_PROTO_THREAD` is specified as protocol value, see the description of the message tag [below](#)) or the IPC target is the current userland thread. In the latter case, no IPC will be performed and the send phase and the receive phase will block until the corresponding timeout has expired, see [below](#).

A special partner is defined by the *reply capability*. Since it would be impractical (and a security flaw) to always pass an explicit object capability to reply to for each IPC, the kernel generates an implicit one that can be used for just that purpose if the IPC contains an **open wait** phase. The reply capability is valid after a receive phase and points to the kernel object that sent the IPC just received. It can be used only once. The reply capability is selected by setting the `L4_SYSF_REPLY` flag, see [above](#).

#### 4.1.1.3 IPC Label

The IPC label is a machine word which is transferred unchanged from the IPC sender to the IPC receiver when directly sending to a userland thread. However, the primary purpose of the label is to create a relationship between an `L4::Rcv_endpoint` (`L4::lpc_gate` or `L4::lrq`) and the bound thread.

During `L4::Rcv_endpoint::bind_thread()`, a label is specified. If the thread receives an IPC message through the endpoint, that label is delivered to the receiving thread as output parameter of `l4_ipc()` instead of the label specified during the corresponding IPC send operation (see the detailed description of `L4::lpc_gate` for more details on the label with IPC gates). The label is usually used by the receiving thread to invoke the object which is responsible for handling the IPC request of the corresponding endpoint. This mechanism is used by the `L4::Epiface` mechanism for server loops.

#### 4.1.1.4 IPC Message Tag

The *message tag* (`l4_msgtag_t`) describes the payload of the IPC and can also be used to enable certain features. It contains:

- a *protocol value* (cf. `l4_msgtag_t::label()`, also called the tag's *label*),
- the number of items in [UTCB message registers](#) to transfer (cg. `l4_msgtag_t::words()` and `l4_msgtag_t::items()`), and
- flags (cf. `l4_msgtag_t::flags()` and `L4_msgtag_flags`, may be 0).

The information from the message tag is required by the kernel to transfer the message. The IPC system call returns a message tag as result of the IPC operation. On success, a copy of the message tag specified by the sender is returned if there is a receive phase. Without receive phase, a successful IPC syscall returns the message tag specified as input parameter.

Failures during IPC are signalled using the `L4_MSGTAG_ERROR` flag in the message tag. If this bit is set by the kernel, the content of the returned message tag apart from that bit is undefined and the kernel wrote the actual error code into the `l4_thread_regs_t::error` register of the UTCB (see also [IPC Thread Control Registers](#)). When an IPC error occurs after the rendezvous of the IPC partners, both partners observe the same error information. If, for

instance, the IPC was aborted using `L4::Thread::ex_regs()`, the sender gets an `L4_IPC_SECANCELED` error while the receiver gets an `L4_IPC_RECANCELED` error. The function `L4Re::chkipc()` can be used to verify that an IPC operation finished successfully: It throws an error if the IPC failed.

The *protocol value* is usually used to distinguish between different protocols of the same interface. Certain protocol IDs are pre-defined when talking to kernel objects, see `L4_msgtag_protocol`. From IPC point of view, the protocol value is just payload that is transferred from sender to receiver and hence doesn't have a dedicated meaning.

By convention, during IPC calls, the protocol value is used for return values, where negative values signify errors. See the [section](#) about L4 RPC return types for further information. The function `L4Re::chksys()` can be used to verify that an RPC call using IPC was successful: It throws an error if the IPC failed or if the returned protocol value is negative.

#### 4.1.1.5 IPC Timeouts

As written above, IPC *timeouts* are specified separately for the send phase (IPC send timeout) and the receive phase (IPC receive timeout). The timeout of one phase is encapsulated by `l4_timeout_s`. Both timeouts are combined into a single `l4_timeout_t` parameter. Timeouts are either relative to the current time of the invoking thread or absolute. In the latter case, the absolute time of the deadline of the respective phase is written into a UTCB buffer register (see `l4_timeout_abs()`). The relative timeout of the receive phase starts immediately after the send phase has finished.

Two specific timeout values are sufficient for most IPC operations:

- `L4_IPC_TIMEOUT_NEVER` is used if the IPC partner might not yet be ready. Usually a client trusting a server will perform an IPC call with an infinite timeout for both phases. The send phase will take some time if the IPC receiver is currently not ready. The receive phase will take some time as the server needs time to serve the request. Also, a server will usually wait with an infinite receive timeout for the next request (see below for a possible exception).
- `L4_IPC_TIMEOUT_0` is used when it is either certain that the IPC receiver is currently ready or if the IPC sender doesn't want to wait if the IPC receiver is not ready. The former case applies to a thread which was called with an IPC call, for example a server got a client request. The reply to the IPC call should use a timeout of zero to ensure that a client doesn't block a server (server could not deliver the result of the request). See also `L4::ipc_srv::Default_timeout`. Another case is an IPC send operation for waking up another thread from an IPC receive operation. If the other thread is not ready to receive, then it might be already woken up and it does not make sense to wait any longer. Also triggering an IRQ is usually done with a send timeout of 0, see `L4::Triggerable::trigger()`.

In certain cases it also makes sense to specify an IPC timeout different from "never" or "zero":

- A server might leave the server loop after some time to perform idle work (see `L4::ipc_srv::Server_iface::add_timeout()`).
- A thread does not want to wait for an infinite time if the partner is not ready. This could be also some safety measure.
- A thread wants to block for a certain amount of time without consuming CPU time. The `l4_ipc_sleep()` function specifies the `L4_INVALID_CAP` as target for an IPC receive operation and specifies the intended relative waiting period as IPC timeout. Waiting for an absolute timeout would be possible with similar code.

#### Note

The kernel IPC path is optimized for the two special cases using `L4_IPC_TIMEOUT_NEVER` and `L4_IPC_TIMEOUT_0`. Specifying a different timeout causes more maintenance effort for the kernel.

Special care is required if a finite timeout for the receive phase of an IPC call is specified: The IPC receive operation could abort before the partner was able to send the reply message. Under certain circumstances the partner may still have the temporary reply capability to the calling thread and may use this capability to reply to the caller at a later, unexpected time specifying an arbitrary IPC label. This case is relevant for servers which call another, possibly untrusted, server while serving a client request.

#### 4.1.1.6 User-level Thread Control Block

The **UTCB** is located on a power-of-2-sized and power-of-2-aligned memory area shared between userland and the kernel (`L4::Task::add_ku_mem()`). The UTCB is bound to a thread during the `L4::Thread::control()` operation with the `L4::Thread::Attr` parameters set up using `L4::Thread::Attr::bind()`. The UTCB is used for IPC-related data exchange and is set up before invoking `l4_ipc()`. To access certain parts of the UTCB, the corresponding functions have to be used (there is no data type `L4_utcb` or similar). The UTCB consists of:

- the **message registers** `MR[0]`, `MR[1]`, ..., `MR[n-1]` with  $n = \text{L4\_UTCB\_GENERIC\_DATA\_SIZE}$  (access using `l4_utcb_mr()`),
- the **buffer descriptor register** `BDR` (access using `l4_utcb_br()`, see `l4_buf_regs_t::bdr`),
- the **buffer registers** `BR[0]`, `BR[1]`, ..., `BR[m-1]` with  $m = \text{L4\_UTCB\_GENERIC\_BUFFERS\_SIZE}$  (access using `l4_utcb_br()`),
- the **thread control registers** (access using `l4_utcb_tcr()`, includes the IPC error code), and
- in case of an exception IPC, the register state of the thread which triggered the exception (access using `l4_utcb_exc()`).

IPC to a kernel object requires the setup of the UTCB of the corresponding userlevel thread. IPC between userlevel threads requires the setup of UTCBs of both partners.

The kernel changes only the following UTCB content:

- The message registers of the UTCB of the receiver of an IPC, and
- the IPC error field of the thread invoking `l4_ipc()` if there was an error during IPC.

##### 4.1.1.6.1 IPC Message registers

The *message registers* contain *untyped items* and *typed items*. The sender's typed items are interpreted by the kernel in conjunction with the receiver's *receive items*. Each typed send item occupies two message registers. The untyped items, on the other hand, are free to be used by the communication partners to exchange data: The content of all message registers used for untyped items (`l4_msgtag_t::words()`) is copied from the UTCB of the IPC sender to the UTCB of the IPC receiver.

A typed send item consists of a *flexpage* (see `l4_fpage()`, `l4_obj_fpage()`, and `l4_iofpage()`) of the to be transferred capabilities (*flexpage word*) and a *message word*. There are two types of send items: *map items* and *void items*. For a void item, the message word is all zero. For a map item, the message word contains:

- the *compound bit* allowing to use the same receive buffer for the subsequent typed send item (scatter-gather behavior, see `L4_ITEM_CONT` of `l4_msg_item_consts_t`),
- the *type bit* defining this typed send item as a *map item*,
- the *grant flag* for delegating the access to the flexpage from the sender to the receiver and atomically removing the rights from the sender (see `l4_msg_item_consts_t`),
- *attributes* with semantics depending on the item type; for memory mappings, they contain cacheability information (see `l4_fpage_cacheability_opt_t`); for objects, they contain additional rights (see `L4_obj_fpage_ctl`),
- the *send base* (also called *hot spot*) defining what is actually mapped when send and receive flexpages have a different size.

A typed send item can be created using `l4_sndfpage_add()`. This function sets the compound bit unconditionally. Alternatively, the functions `l4_map_control()` and `l4_map_obj_control()` can be used to set up the message word of a map item for a memory flexpage respective for objects.

See [below](#) for a description how typed items are transferred.

#### 4.1.1.6.2 IPC Buffer Registers

The *buffer registers* and *buffer descriptor register* are interpreted by the kernel during the receive phase (if any). [Buffer](#) registers define *receive items* which are required to receive typed send items (memory, I/O ports or object capabilities). To specify a receive item, up to three buffer registers are required:

- A *small receive item* ([L4::ipc::Small\\_buf](#)) occupying one buffer register is sufficient to receive one object capability.
- A *receive item* ([L4::ipc::Rcv\\_fpage](#)) occupying two or three buffer registers (*message word*, a *flexpage*, and an optional destination task capability index) is required to receive memory flexpages, I/O ports, or multiple object capabilities.

#### 4.1.1.6.3 IPC Buffer Descriptor Register

The buffer descriptor register defines indices of buffer registers used to receive dedicated types of send items and also contains a flag:

- 5 bits starting at [L4\\_BDR\\_MEM\\_SHIFT](#) define the index of the first receive item used for memory flexpages.
- 5 bits starting at [L4\\_BDR\\_IO\\_SHIFT](#) define the index of the first receive item used for I/O flexpages.
- 5 bits starting at [L4\\_BDR\\_OBJ\\_SHIFT](#) define the index of the first receive item used for object flexpages.
- The [L4\\_UTCB\\_INHERIT\\_FPU](#) can be set using [l4\\_utcb\\_inherit\\_fpu\(\)](#) and allows to receive the FPU state from the IPC sender. This is only relevant if the sender set [L4\\_MSGTAG\\_TRANSFER\\_FPU](#) in the message tag.

For most use cases, a BDR value of zero is sufficient. In that case, if `BR[0]` contains a void item, no capabilities are received. Otherwise, only one type of capabilities (memory, I/O ports or objects) can be received even if there are several receive items. For more complex setups that require receiving different types of capabilities in one receive operation, other BDR values are necessary.

The BDR of the receiving thread is only used by the kernel if at least one typed item is transferred during the IPC or if [L4\\_MSGTAG\\_TRANSFER\\_FPU](#) is set in the UTCB of the sending thread.

#### 4.1.1.6.4 IPC Thread Control Registers

The [l4\\_thread\\_regs\\_t::error](#) register contains the IPC error code in case the [L4\\_MSGTAG\\_ERROR](#) flag is set in the message tag returned by an IPC syscall. Otherwise this register is not touched by the kernel. See [l4\\_ipc\\_tcr\\_error\\_t](#) for a detailed enumeration of all possible error codes during an IPC operation.

The [l4\\_thread\\_regs\\_t::free\\_marker](#) is set by the kernel to zero immediately before a thread is destroyed. This value indicates that the kernel does not longer use the UTCB and it can be re-used by other threads.

The other members of [l4\\_thread\\_regs\\_t](#) are never touched by the kernel.

#### 4.1.1.7 Transfer of Typed Send Items

The kernel interprets all typed items in the sender UTCB ([l4\\_msgtag\\_t::items\(\)](#)) and performs the following operations while modifying the corresponding typed items in the receiver UTCB:

- If the message item of the sender is void, this item is ignored and the message word of the corresponding typed item in the receiver UTCB is set to zero (making it a void item). The flexpage word of this item is not changed.
- Otherwise, if the type bit of the message item of the sender is not set, the IPC is aborted with [L4\\_IPC\\_SEMSGCUT](#) / [L4\\_IPC\\_REMSGCUT](#).
- Otherwise, if there is a receive item corresponding to the flexpage type of the send item available (see [IPC Buffer Descriptor Register](#)), information described by the flexpage is transferred to the receiver.

In the last case, the message word of the typed item in the receiver UTCB is modified to contain the send base, the type and the size of the transferred flexpage, as well as a copy of the compound bit and the type bit of the send item. If the receiver ordered a local ID in the corresponding receive item, the kernel attempts to apply special rules, see [L4\\_RCV\\_ITEM\\_LOCAL\\_ID](#). Otherwise, regular mappings as described by the flexpage of the send item are created in the receiver space.

A receive item forms a *receive window* of a specific address and size in the receiver space. Each typed send item that is a map item requires a corresponding receive item. By default, there is a one-to-one mapping (one receive item per typed send item) but it is also possible to use one receive item to receive several typed send items: The compound bit (see [l4\\_msg\\_item\\_consts\\_t](#)) of a send item defines if the following typed send item shall use the same receive item as the current one for receiving the flexpage. If the compound bit is set, proper values of the send base shall be used to prevent overlapping of addresses in the receiver space.

The send base is relevant when the size of the receive flexpage differs from the size of the transferred resource. As a typical example, triggering a memory page fault opens a receive window covering the entire memory address space of the faulting thread. The pager will usually reply a memory flexpage smaller than the entire address space of the faulting thread. Hence, the pager has to specify a proper base which is used as offset of the sent object in the receive flexpage in the *receiver space*. If the sender flexpage is bigger than the receive window, a flexpage of the size of the receive window starting at the send base in the *sender space* is transferred to the receiver.

The kernel will stop transmission of typed items before [l4\\_msgtag\\_t::items\(\)](#) is reached either if it finds a void item as receive buffer or if the flexpage type of the send item does not match the flexpage type of the corresponding receive item. Under both conditions, the IPC is aborted with [L4\\_IPC\\_SEMSGCUT](#) / [L4\\_IPC\\_REMSGCUT](#).

#### Note

The kernel ignores the flexpage access rights of the receive items. The actual rights for a transferred resource in the target address space are defined by the access rights to that resource of the IPC sender address space and the flexpage access rights in the typed send item. Additionally, when transferring object capabilities, the transferred rights also depend on the sender's rights on the capability invoked for IPC.

The kernel does not unmap capabilities in the receive window when there is no capability present at the corresponding index at the sender. Further, the receiver cannot reliably detect at which capability indices it received capabilities in its receive windows. Therefore, before receiving from an untrusted source, all receive windows should be cleared. Otherwise the receiver may erroneously associate a capability in one of its receive windows with his last IPC partner although it was actually received in an earlier IPC.

However, the kernel indicates if at least one object capability was received or not, see [L4::lpc::Snd\\_fpage::cap\\_received\(\)](#).

## 4.1.2 Examples

A number of examples show the interplay of the concepts introduced so far.

#### 4.1.2.1 User Thread to Kernel Object

The `L4::Scheduler` kernel object has a method `L4::Scheduler.idle_time()`. It takes a set of CPUs to query, represented by two machine words.

In user space, the function `L4::Cap<L4::Scheduler>->idle_time()` is called, which does the following:

- Fill `MR[0]` with a constant identifying the method being called (`L4_SCHEDULER_IDLE_TIME_OP`).
- Fill `MR[1]` and `MR[2]` with the two words making up the CPU set.
- Set up the message tag with the protocol value (`L4_PROTO_SCHEDULER`), the number of untyped and typed items (3 and 0), and some flags.
- Call `l4_ipc()` with the partner capability ID, the tag, the pointer to the filled UTCB, infinite timeouts, and with flags indicating a send and receive phase. (The label does not matter in this case.)

This function traps into kernel space using standard platform specific mechanisms. The kernel reads the protocol value on the message tag, checks that the partner capability ID refers to a valid object that speaks that protocol, and dispatches the message to the appropriate handler. The handler fills the first 64 bits of the message registers with the computed time value. This will cover `MR[0]` on 64-bit architectures and `MR[0]` and `MR[1]` on 32-bit architectures. It then sets up the return message tag:

- The number of untyped items is 1 or 2.
- The number of typed items is 0.
- The protocol value contains the return value and is set to 0.
- An error would be signalled as a negative protocol value. Under certain conditions (e.g. wrong kernel object capability specified), the error is signalled as IPC error (see `L4_MSGTAG_ERROR` in the description of the `IPC Message Tag`).
- (The return label is as irrelevant in this case as the send label.)

This reply is received by the receive phase of the original `l4_ipc()` call from userland. Finally the `l4_ipc()` function copies the time value out of the message registers and forwards it with a possible error from the message tag flags to its caller.

#### 4.1.2.2 User Thread to User Thread

When the target kernel object is of type `L4::Thread` (or `L4::ipc_gate`, but we will cover this in the example below) and the message tag's protocol value is not `L4_PROTO_THREAD`, the kernel will forward the IPC message to the userland thread represented by the kernel object. There it can be received by a call to `l4_ipc()`. The restriction of the protocol number is necessary because one also wants to invoke `L4::Thread`'s control methods such as `L4::Thread.switch_to()` or `L4::Thread.ex_regs()`. Besides this restriction, the interpretation of all the IPC parameters and the untyped items of the UTCB is up to the communication partners.



#### 4.1.2.2.1 Simple Messages

A simple example is a client calling a server to have a computation performed on a value: Similar to IPC from a userland thread to a kernel object, the client writes the value to `MR[0]`. It sets up the message tag with some agreed upon protocol value (which, as explained above, must be different from `L4_PROTO_THREAD`), number of untyped items to 1, typed items to 0, and no flags set. The client may want to pass a label that identifies itself, as many clients can use the server. In this context, the identifier might also be passed via the message registers, but the label is the proper place for this, as it gets a special treatment by the kernel for IPC gates (covered by the example below). The client then calls `l4_ipc()` with the tag, label and flags indicating it wants a send phase and a receive phase (as it wants a result back). The target is the capability index referring to a capability for the `L4::Thread` kernel object of the server.

To be able to receive an IPC message, the server has set up a UTCB of its own and called `l4_ipc()` with flags indicating it only wants to enter a receive phase and it accepts IPCs from any partner. This is called an **open wait**. (The alternative would be to specify a capability index referring to a `L4::Thread` capability to exclusively receive from.)

Both system calls (the send IPC initiated by the client and the receive IPC initiated by the server) may be seen by the kernel in any order but the IPC will not start before sender and receiver are ready. In that case the kernel will copy the relevant message registers the client specified in the message tag from the client's UTCB to the server's UTCB, in the current example just `MR[0]`. It will then switch the client to the receive phase of its call (which cannot yet be executed) and return the server's call with the message tag and label.

The server inspects the tag for the correct protocol value and number of untyped items passed, inspect the label to decide whether it maybe wants special treatment of this particular client, performs the computation on `MR[0]` and writes the result back to `MR[0]` (or to more words, depending on what exactly it computes). It sets up the tag in the usual way, but probably needs to pass no label, as the client knows who it is talking to.

For the reply, the server makes use the reply capability (see above). Since the client sent the last IPC to the server, the reply capability will point to it. So when the server calls `l4_ipc()` with the computed result in the message registers and using the reply capability as target, the kernel knows to forward this to the client's thread. The kernel copies the message registers from the server's UTCB to the client's UTCB, and the client's `l4_ipc()` system call, which is still stuck in the receive phase, is returned with the tag.

The client looks at the tag and then the message registers for its wanted result and the example is concluded.

#### 4.1.2.2.2 Send Items

IPC between userland threads is also used to transfer typed items: Memory, I/O ports, and objects, all represented as flexpages. Typed items and untyped items can be part of the same IPC. As general rule, the sender specifies what he wants to send, the receiver where and how much it wants to receive, and the kernel checks the required permissions before doing the actual transfer. As written before, this mechanism is synchronous and the receiver cannot be transferred items against its will.

See also

[Flexpages](#)

Suppose a client wants a server to have read only access to a page of its memory. The client sets up a flexpage covering the page and with only the `L4_FPAGE_RO` right set. The server sets up a flexpage of a memory region where it will receive the mapping. This may be larger than one page, suppose for our case four pages, in which case the exact position of the mapping will be resolved by the send base provided by the sender. The client combines the hot spot and some flags into a machine word and writes it to `MR[0]` (see also `l4_map_control()`). At `MR[1]` follows the flexpage it wants to send (see also `l4_fpage()`). The server does almost the same but writes the words to `BR[0]` and `BR[1]`. (The server could also specify a hot spot, but it is currently ignored by the kernel.) The

client specifies 1 typed and 0 untyped items in the message tag. The server writes 0 to `BDR` to specify that the first receive item starts at the first buffer register.

Both client and server initiate their IPC, the client with only a send phase to the server, and the server with an open wait receive phase. The kernel checks the compatibility of the send items and the receive buffers (e.g. that no object capability flexpage is sent to a receive buffer describing a memory mapping flexpage) and updates its internal structures to reflect the change. In our case, the sender's hot spot indicates to which of the four pages that make up the receive buffer the sent page should be mapped. The kernel also translates the typed send item to the server's address space and stores it in the server's UTCB at `MR[0]` and `MR[1]`.

Once the server returns from its syscall, it will have read access to the client's shared page.

#### 4.1.2.3 User Thread to User Object

A common use case for thread to thread communication is when a server implements a number of object interfaces and a client wants to invoke methods on them. For security reasons, the server does not want to hand out its thread capability to everyone it nonetheless wants to serve. It also may not want to allow every client access to everyone of its interfaces. For this purpose, IPC gates implemented by the kernel object `L4::ipc_gate` can be used. An IPC gate can be bound to a thread and forwards IPC to it. In doing so it applies two transformations

1. It sets the label to a predefined value.
2. It changes the rights of transferred items (see `L4_CAP_FPAGE_S`).

For each object of every interface the server implements, it creates an IPC gate and binds it to itself (see `L4::ipc_gate::bind_thread()`). It sets the gate's label to a unique value identifying the object. Then it hands the gate out to clients. Several clients can be handed the same gate and will all end up invoking methods on the same object.

Instead of setting the target as the server's thread kernel object, the client uses the IPC gate's instead. The label the client sends is irrelevant, as the gate will overwrite it with the value the server has set during the bind operation. The server executes an open wait, and the kernel performs the same operation as in the above [example](#) with the transformed IPC finally ending up in the server's thread.

The server checks that the received label refers to one of its objects. It then checks if the protocol value in the message tag matches the interface the object implements. Then it invokes the method specified in `MR[0]` with the rest of the arguments. Finally it returns the results via UTCB and message tag to the reply capability and waits for the next IPC.

## 4.2 Kernel ABI

This section details the binary representation of the IPC interface of the kernel.

It accompanies the [L4 Inter-Process Communication \(IPC\)](#) section. The details presented here are usually not relevant when developing L4Re applications and can therefore be skipped by many readers.

#### Note

The kernel ABI is subject to change. Please use the API instead of relying on particular binary representations.

The following notation is used to indicate how particular data fields are used:

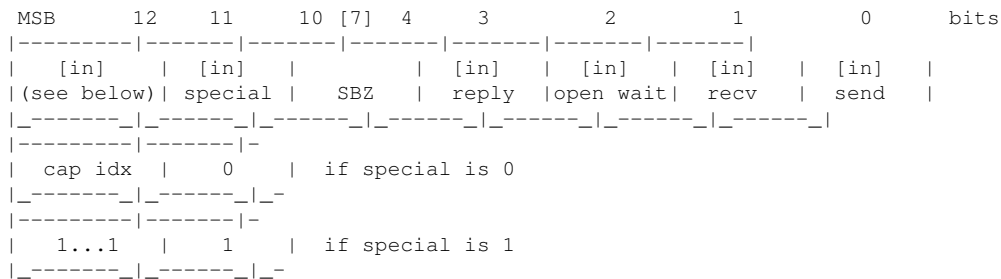
- `[in]`: The kernel reads and interprets this field.
- `[out]`: The kernel writes this field with information provided by the kernel.
- `[cpy]`: The kernel copies this field from sender to receiver (without interpretation if `[in]` is not listed as well).

The above indications may be combined.

### 4.2.1 Capability selector and flags

See [partner capability selector](#) and [IPC flags](#).

The kernel reads and interprets all the fields ([in]).



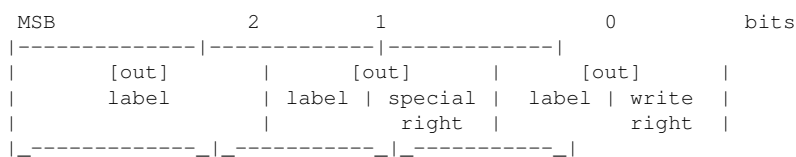
- Bits 0...3 [in]: These bits correspond to the flags defined in [l4\\_syscall\\_flags\\_t](#). The individual bits correspond to [L4\\_SYSF\\_REPLY](#), [L4\\_SYSF\\_OPEN\\_WAIT](#), [L4\\_SYSF\\_RECV](#), [L4\\_SYSF\\_SEND](#). Note that not all combinations of those bits are defined; see [l4\\_syscall\\_flags\\_t](#).
- Bits 4...10 [in] SBZ: should be zero
- Bit 11 [in] special: Set when using [L4\\_INVALID\\_CAP](#), otherwise unset.
- Bits 12...MSB [in]: Capability index if special is unset, otherwise all those bits should be one (see [L4\\_INVALID\\_CAP](#), [partner capability selector](#) and [l4\\_cap\\_idx\\_t](#)).

### 4.2.2 Label

See [IPC label](#).

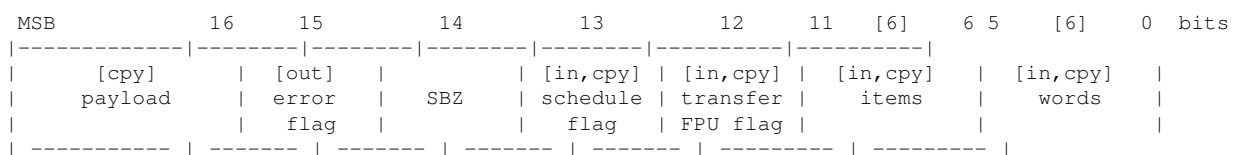
When IPC is sent via a thread capability, the label is copied to the receiver unchanged ([cpy]).

When IPC is sent via an IPC gate, the sent label is ignored and the kernel provides the bitwise OR (|) of the IPC gate label and the sender's write and special permissions (see [L4\\_CAP\\_FPAGE\\_W](#) and [L4\\_CAP\\_FPAGE\\_S](#)) of the used capability ([out]):



### 4.2.3 Message tag

See [IPC message tag](#). Note that, for a message tag returned by the kernel, if the error flag is set, all other contents of the message tag is undefined.



- Bits 0...5 [in,cpy] `words`: Number of (untyped) message words in the UTCB's message registers. See [l4\\_msgtag\\_words\(\)](#) and [l4\\_msgtag\\_t::words\(\)](#).
- Bits 6...11 [in,cpy] `items`: Number of typed message items in the UTCB's message registers. See [l4\\_msgtag\\_items\(\)](#) and [l4\\_msgtag\\_t::items\(\)](#).
- Bit 12 [in,cpy] `transfer` FPU flag: See [L4\\_MSGTAG\\_TRANSFER\\_FPU](#).
- Bit 13 [in,cpy] `schedule` flag: See [L4\\_MSGTAG\\_SCHEDULE](#).
- Bit 14 SBZ: should be zero
- Bit 15 [out] `error`: See [L4\\_MSGTAG\\_ERROR](#), [l4\\_msgtag\\_has\\_error\(\)](#) and [l4\\_msgtag\\_t::has\\_error\(\)](#).
- Bits 16...MSB [cpy] `payload`: Transferred to receiver unchanged; not interpreted by kernel (unless it is the communication partner). For IPC calls or send-only IPC, this is usually the protocol. For replies, this is usually used for return values and server error signaling. See [l4\\_msgtag\\_label\(\)](#) and [l4\\_msgtag\\_t::label\(\)](#).

## 4.2.4 Timeouts

See [IPC timeouts](#) and [l4\\_timeout\\_t](#).

The kernel reads and interprets all the fields ([in]).

```

31      [16]      16 15      [16]      0  bits
|-----|-----|
|      [in]      |      [in]      |
| send timeout   | receive timeout |
|_-----|_-----|

```

A timeout has the following format. There are two special timeout values:

- *Zero timeout*: Only bit 10 is set. See [L4\\_IPC\\_TIMEOUT\\_0](#).

```

15  [5]  11  10   9      [10]      0  bits
|-----|-----|-----|
|      0      |  1  |      0      |
|_-----|_-----|_-----|

```

- *Infinite timeout*: All bits are unset. See [L4\\_IPC\\_TIMEOUT\\_NEVER](#).

```

15      [16]      0  bits
|-----|
|      0      |
|_-----|

```

Otherwise, the timeout is either relative or absolute, which is specified by bit 15.

- *Relative timeout*: If bit 15 is unset, the timeout is `mantissa * 2 ^ exponent` micro seconds relative to the current time. The `mantissa` must not be zero:

```

15  14  [5]  10  9      [10]      0  bits
|----|-----|-----|
|  0  | exponent | mantissa ≠ 0 |
|_---|_-----|_-----|

```

- *Absolute timeout*: If bit 15 is set, an absolute timeout is specified in the UTCB's buffer registers starting at `buf reg idx` (the particular number of registers depends on the architecture; see [l4\\_timeout\\_s](#)):

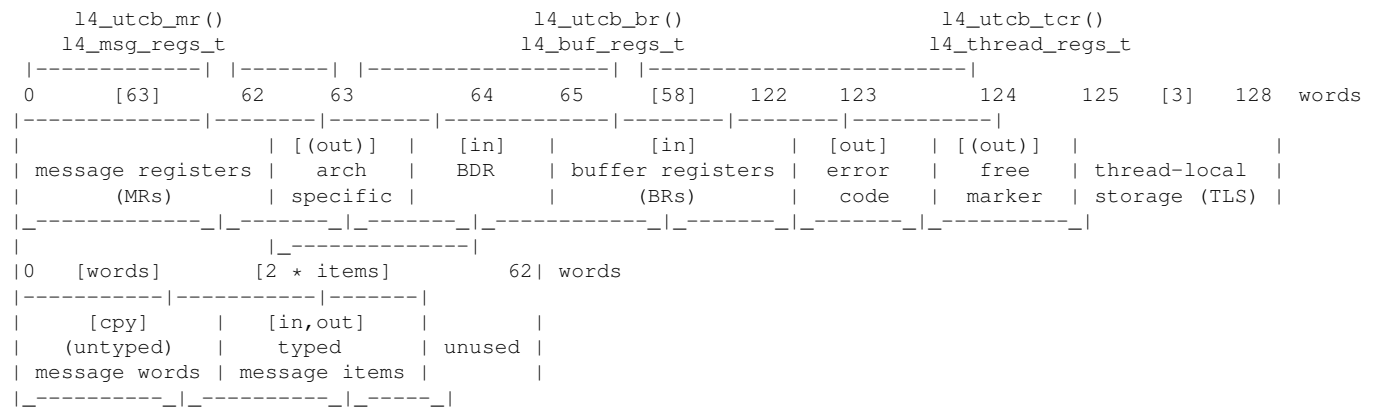
```

15  14      [9]      6 5      [6]      0  bits
|----|-----|-----|
|  1  |      SBZ      | buf reg idx |
|_---|_-----|_-----|

```

## 4.2.5 User-level thread control block (UTCB)

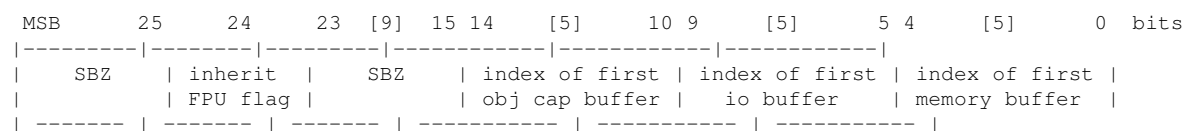
See [User-level thread control block \(UTCB\)](#).



- Words 0...62 MRs: See [IPC Message registers](#) and [l4\\_utcb\\_mr\(\)](#). The number of message registers is defined by [L4\\_UTCB\\_GENERIC\\_DATA\\_SIZE](#). The actually used message registers are defined by `words` and `items` in the [message tag](#). The layout of a typed message item varies depending on being an input or output value, see [typed message items](#).
- Word 63 [(out)]: Depending on the architecture, this word may be used by the kernel to signify the position of a thread's UTCB in memory. See architecture-specific implementation of [l4\\_utcb\(\)](#). If at all, the kernel writes this word when kernel-user memory is set up as UTCB while binding a thread to a task; see [l4\\_thread\\_control\\_bind\(\)](#), [L4::Thread::Attr::bind\(\)](#).
- Word 64 [in] BDR: See [buffer descriptor register](#).
- Words 65...122 [in] BRs: See [IPC Buffer Registers](#), [receive items](#) and [l4\\_utcb\\_br\(\)->br](#). The number of buffer registers is defined by [L4\\_UTCB\\_GENERIC\\_BUFFERS\\_SIZE](#).
- Word 123 [out] `error code`: See [IPC Thread Control Registers](#) and [l4\\_utcb\\_tcr\(\)->error](#).
- Word 124 [(out)] `free marker`: Written by the kernel, but not necessarily during IPC. See [IPC Thread Control Registers](#) and [l4\\_utcb\\_tcr\(\)->free\\_marker](#).
- Word 125...128 TLS: Ignored and left untouched by the kernel. See [IPC Thread Control Registers](#) and [l4\\_utcb\\_tcr\(\)->user](#).

### 4.2.5.1 Buffer descriptor register

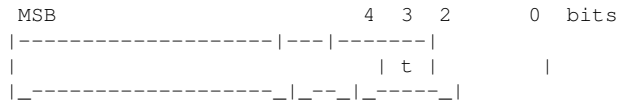
See [IPC Buffer Descriptor Register](#) and [l4\\_utcb\\_br\(\)->bdr](#).



## 4.2.6 Typed message items

The number of words in a typed message item varies depending on the particular kind of item. However, for the first word, the following properties are shared:

- *Void item*: If all bits of the first word of a typed message item are zero, then it is a void item.
- *Non-void item*: The first word of a non-void typed message item has the following binary layout:



Bit 3 ( $t$ ) is the type bit. If  $t$  is set, the item is a map item. Currently, map item is the only supported type. Hence, this bit must be set for all items except for void items.

There are three sub-types of typed message items: *send items*, *receive items*, and *return items*; see [Message Items](#).

Many typed items make use of flexpages, therefore, these are described before the various kinds of typed items. Note that flexpages are also used outside of typed message items, e.g., for [L4::Task::unmap\(\)](#).

### 4.2.6.1 Flexpages

A flexpage consists of a single word and, except for some special values, describes a range in an address space, see [flex pages](#).

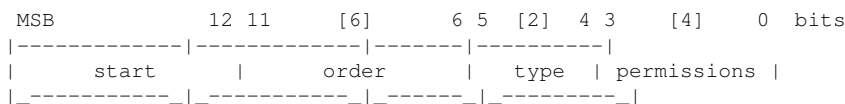
The general layout is defined as follows:



- Bits 4...5 type: See [l4\\_fpage\\_type\(\)](#) and [L4\\_fpage\\_type](#).

The type [L4\\_FPAGE\\_SPECIAL](#) only supports some selected values, which are only supported for selected interfaces; see [L4\\_FPAGE\\_SPECIAL](#).

The other types share the same layout:



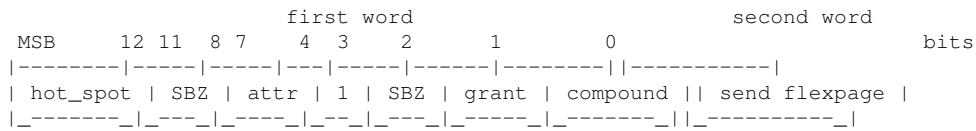
- Bits 0...3 permissions: See [l4\\_fpage\\_rights\(\)](#), [L4\\_fpage\\_rights](#) (memory space) and [L4\\_cap\\_fpage\\_rights](#) (object space). Should be zero for I/O port space.
- Bits 6...11 order: The  $\log_2$  size of the flexpage. See [l4\\_fpage\\_size](#).
- Bits 12...MSB start: The starting page number / I/O port number / capability index of the flexpage. Must be aligned to the flexpage size. See [l4\\_fpage\\_page\(\)](#), [l4\\_fpage\\_memaddr\(\)](#), [l4\\_fpage\\_ioport\(\)](#) and [l4\\_fpage\\_obj\(\)](#).

Also see [l4\\_fpage\(\)](#) (memory space), [l4\\_iofpage\(\)](#) (I/O port space) and [l4\\_fpage\\_obj\(\)](#) (object space).

### 4.2.6.2 Send items

A send item consists of two words. The second word of a non-void send item is a [flexpage](#). The type of the flexpage determines the interpretation of the `attr` bits in the first word (see below).

If not void, the layout of the first word is defined as follows:



SBZ means “should be zero”.

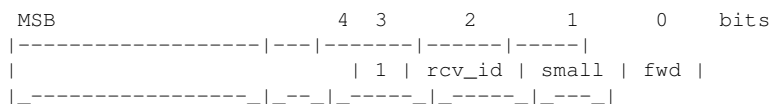
- Bit 0 (compound): Compound bit. See [L4\\_ITEM\\_CONT](#) and [L4::lpc::Snd\\_fpage::is\\_compound\(\)](#).
- Bit 1 (grant): Grant flag. See [L4\\_ITEM\\_MAP](#), [L4\\_MAP\\_ITEM\\_GRANT](#) and [L4::lpc::Snd\\_fpage::Map\\_type](#).
- Bits 7..4 (attr): Attributes. See [L4\\_obj\\_fpage\\_ctl](#) and [l4\\_fpage\\_cacheability\\_opt\\_t](#), [L4::lpc::Snd\\_fpage::Cacheopt](#).
- Bits MSB..12 (hot\_spot): Send base (also called hot spot). See [L4::lpc::Snd\\_fpage::snd\\_base\(\)](#).

For details, see [IPC Message registers](#).

### 4.2.6.3 Receive items

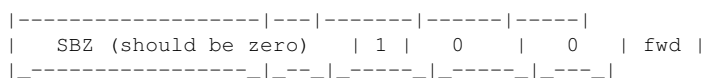
A non-void receive item consists of up to three words.

If not void, the general layout of the first word is defined as follows:

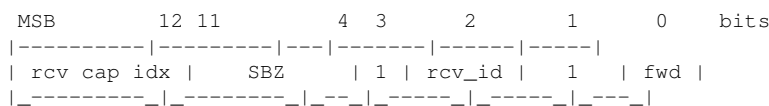


The `small` and `fwd` bits determine the details of the layout of the whole message item.

If `small` is unset, then also `rcv_id` must be unset, and the most significant bits should be zero:



If `small` is set, the most significant bits are layouted as follows:







```

|-----|-----|-----|---|-----|---|-----|
| hot_spot | order | type | 1 | 01 | c | undefined |
|_-----|_-----|_-----|_--|_-----|_--|_-----|

```

10: Used if the receive item's `rcv_id` bit was set and the conditions for transferring an IPC gate label were fulfilled. In that case, no mapping is done for this item and the payload consists of the bitwise OR (|) of the IPC gate label and the write and special permissions (see [L4\\_CAP\\_FPAGE\\_W](#) and [L4\\_CAP\\_FPAGE\\_S](#)) that would have been mapped (also see [L4::lpc::Snd\\_fpage::id\\_received\(\)](#)):

```

                                     2 1      0 bits
|-----|-----|-----|---|-----|---|-----|
| hot_spot | order | type | 1 | 10 | c | label | rights |
|_-----|_-----|_-----|_--|_-----|_--|_-----|

```

11: Used if the receive item's `rcv_id` bit was set and the conditions for transferring the sender's flexpage word were fulfilled. In that case, no mapping is done for this item and the payload is a copy of the sender's flexpage word (also see [L4::lpc::Snd\\_fpage::local\\_id\\_received\(\)](#)):

```

|-----|-----|-----|---|-----|---|-----|
| hot_spot | order | type | 1 | 11 | c | send flexpage |
|_-----|_-----|_-----|_--|_-----|_--|_-----|

```

## 4.3 Capabilities and Naming

The [L4Re](#) system is a capability based system which uses and offers capabilities to implement fine-grained access control.

Generally, owning a capability means to be allowed to communicate with the object the capability points to. All user-visible kernel objects, such as tasks, threads, and IRQs, can only be accessed through a capability. Please refer to the [Kernel Objects](#) documentation for details. Capabilities are stored in per-task capability tables (the object space) and are referenced by capability selectors or object flexpages. In a simplified view, a capability selector is a natural number indexing into the capability table of the current task.

As a matter of fact, a system designed solely based on capabilities uses so-called 'local names' because each task can only access those objects made available to this task. Other objects are not visible to and accessible by the task.

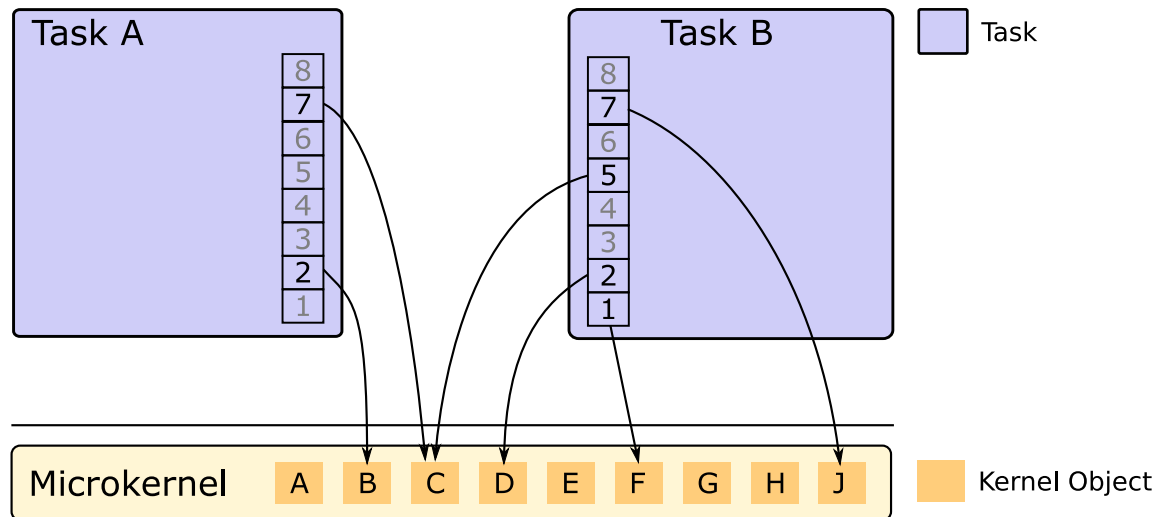


Figure 4.1 Capabilities and Local Naming in L4

So how does an application get access to a service? In general all applications are started with an initial set of available objects. This set of objects is predetermined by the creator of a new application process and granted directly to the new task before starting the first application thread. The application can then use these initial objects to request access to further objects or to transfer capabilities to its own objects to other applications. A central L4Re object for exchanging capabilities at runtime is the name-space object, implementing a store of named capabilities.

From a security perspective, the set of initial capabilities (access rights to objects) completely define the execution environment of an application. Mandatory security policies can be defined by well known properties of the initial objects and carefully handled access rights to them.

## 4.4 Spaces and Mappings

Each task in the L4Re system has access to two resource spaces (three on IA32) which are maintained by the kernel.

These are the

1. object space,
2. memory space, and
3. IO-port space (only on IA32).

The entities addressed in each space are capabilities to objects, virtual memory pages, and IO ports. The addresses are unsigned integers and the largest valid address depends on which space is referenced, the hardware, and the configuration of the kernel. Although a program can access memory at byte granularity, from the kernel's point of view the address granularity in the memory space is not bytes but pages, as determined by the hardware. The address of a capability is also called its "capability slot".

Flexpages describe a range in any of the spaces that has a power-of-two length and is also aligned to this length. They additionally hold access rights information and further space specific information.

When a resource is present at some address in a task's corresponding resource space, then we say that resource is mapped to that task. For example, a capability to the task's main thread may be mapped to capability slot 5, or the first page of the code segment a thread executes is mapped to virtual memory page 12345. However, there need not be any resource mapped to an address.

Tasks can exchange resources through a process called "mapping" during IPC and using the [L4::Task::map\(\)](#) method. The sending task specifies a send flexpage and the receiving task a receive flexpage. The resources mapped to the send flexpage will then be mapped to the receive flexpage by the kernel.

Memory mappings and IO port mappings are hierarchical: If a resource of such a type is subject of a map operation, the received mapping is a child mapping of the corresponding mapping in the sending task (parent mapping). The kernel usually respects the relationship between these two mappings (granting is an exception; see below): If rights of a parent mapping are revoked using [L4::Task::unmap\(\)](#), these rights are also removed from its child mappings. Also, if a mapping is completely removed (via [L4::Task::unmap\(\)](#) or by mapping something else at its place), then also all child mappings are removed. In contrast, revoking rights of a child mapping leaves the rights of its parent mapping untouched.

The mapping of a resource can be performed as *grant* operation (see [L4\\_MAP\\_ITEM\\_GRANT](#)): Such an operation includes the removal of all involved mappings from the send flexpage (basically a move operation). While with a map operation without grant the mapping in the send flexpage remains the parent of all child mappings (including the new child mapping in the receive flexpage), a grant operation moves the mappings covered by the send flexpage to the corresponding addresses from the receive flexpage while leaving the parent/child relationship of the moved mappings with other mappings untouched.

During a map operation at most the access rights of the source mapping(s) can be transferred but no additional rights can be added. So only rights that are present in the source mapping and that are specified in the send item/flexpage are transferred. This also holds for grant mappings, however, rights revocation is *not* guaranteed to be applied to descendant mappings in case of grant.

There are cases where a grant operation is not or cannot be performed as requested; see [L4\\_MAP\\_ITEM\\_GRANT](#) for details.

Object capabilities are not hierarchical – they have no children. The result of the map operation on an object capability is a copy of that capability in the object space of the destination task.

## 4.5 Initial Environment and Application Bootstrapping

New applications that are started by a loader conforming to [L4Re](#) get provided an [Initial Environment](#).

This environment comprises a set of capabilities to initial [L4Re](#) objects that are required to bootstrap and run this application. These capabilities include:

- A capability to an initial memory allocator for obtaining memory in the form of data spaces
- A capability to a factory which can be used to create additional kernel objects
- A capability to a Vcon object for debugging output and maybe input

- A set of named capabilities to application specific objects

During the bootstrapping of the application, the loader establishes data spaces for each individual region in the ELF binary. These include data spaces for the code and data sections, and a data space backed with RAM for the stack of the program's first thread.

One loader implementation is the `moe` root task. Moe usually starts an `init` process that is responsible for coordinating the further boot process. The default `init` process is `ned`, which implements a script-based configuration and startup of other processes. Ned uses Lua (<http://www.lua.org>) as its scripting language, see [Ned Script example](#) for more details.

### 4.5.1 Configuring an application before startup

The default L4Re init process (Ned) provides a Lua script based configuration of initial capabilities and application startup. Ned itself also has a set of initial objects available that can be used to create the environment for an application. The most important object is a kernel object factory that allows creation of kernel objects such as IPC gates (communication channels), tasks, threads, etc. Ned uses Lua tables (associative arrays) to represent sets of capabilities that shall be granted to application processes.

```
local caps = {
    name = some_capability
}
```

The L4 Lua package in Ned also has support functions to create application tasks, region-map objects, etc. to start an ELF binary in a new task. The package also contains Lua bindings for basic L4Re objects, for example, to generic factory objects, which are used to create kernel objects and also user-level objects provided by user-level servers.

```
L4.default_loader:start({ caps = { some_service = service } }, "rom/program --arg");
```

### 4.5.2 Connecting clients and servers

In general, a connection between a client and a server is represented by a communication channel (IPC gate) that is available to both of them. You can see the simplest connection between a client and a server in the following example.

```
local loader = L4.default_loader; -- which is Moe
local svc = loader:new_channel(); -- create an IPC gate
loader:start({ caps = { service = svc:svr() } }, "rom/my_server");
loader:start({ caps = { service = svc:m("rw") } }, "rom/my_client");
```

As you can see in the snippet, the first action is to create a new channel (IPC gate) using `loader:new_channel()`. The capability to the gate is stored in the variable `svc`. Then the binary `my_server` is started in a new task, and full (`:svr()`) access to the IPC gate is granted to the server as initial object. The gate is accessible to the server application as "service" in the set of its initial capabilities. Virtually in parallel a second task, running the client application, is started and also given access to the IPC gate with less rights (`:m("rw")`, note, this is essential). The server can now receive messages via the IPC gate and provide some service and the client can call operations on the IPC gate to communicate with the server.

Services that keep client specific state need to implement per-client server objects. Usually it is the responsibility of some authority (e.g., Ned) to request such an object from the service via a generic factory object that the service provides initially.

```
local loader = L4.default_loader; -- which is Moe
local svc = loader:new_channel():m("rws"); -- create an IPC gate with rws rights
loader:start({ caps = { service = svc:svr() } }, "rom/my-service");
loader:start({ caps = { foo_service = svc:create(object_to_create, "param") } }, "rom/client");
```

This example is quite similar to the first one, however, the difference is that Ned itself calls the `create` method on the factory object provided by the server and passes the returned capability of that request as "foo\_service" to the client process.

#### Note

The `svc:create(...)` call blocks on the server. This means the script execution blocks until the my-service application handles the create request.

## 4.6 Memory management - Data Spaces and the Region Map

### 4.6.1 User-level paging

Memory management in L4-based systems is done by user-level applications, the role is usually called *pager*. Tasks can give other tasks full or restricted access rights to parts of their own memory. The kernel offers means to give access to memory in a secure way, often referred to as *memory mapping*.

The mapping mechanism allows one task to resolve page faults of another: A thread usually has a pager assigned to it. When the thread causes a page fault, the kernel sends an IPC message to the pager with information about the page fault. The pager answers this IPC by either providing a backing page, or with an error. The kernel will map the backing page into the address space of the faulting thread's task.

These mechanisms can be used to construct a memory and paging hierarchy among tasks. The root of the hierarchy is `sigma0`, which initially gets all system resources and hands them out once on a first-come-first-served basis. Memory resources can be mapped between tasks at a page-size granularity. This size is predetermined by the CPU's memory management unit and is commonly set to 4 kB.

#### 4.6.1.1 Data spaces

A data space is the [L4Re](#) abstraction for objects which may be accessed in a memory mapped fashion (i.e., using normal memory read and write instructions). Examples include the sections of a binary which the loader attaches to the application's address space, files in the ROM or on disk provided by a file server, the registers of memory-mapped devices and anonymous memory such as the heap or the stack.

Anonymous memory data spaces in particular (but in general all data spaces except memory mapped IO) can either be constructed entirely from a portion of the RAM or the current working set may be multiplexed on some portion of the RAM. In the first case it is possible to eagerly insert all pages (more precisely page-frame capabilities) into the application's address space such that no further page faults occur when this data space is accessed. In general, however, only the pages for some portion are provided and further pages are inserted by the pager as a result of page faults.

#### 4.6.1.2 Virtual Memory Handling

The virtual memory of each task is constructed from data spaces backing virtual memory regions (VMRs). The management of the VMRs is provided by an object called *region map*. A dedicated region-map object is associated with each task; it allows attaching and detaching data spaces to an address space as well as reserving areas of virtual memory. Since the region-map object possesses all knowledge about the virtual memory layout of a task, it also serves as an application's default pager.

#### 4.6.1.3 Memory Allocation

Operating systems commonly use anonymous memory for implementing dynamic memory allocation (e.g., using `malloc` or `new`). In an L4Re-based system, each task gets assigned a memory allocator providing anonymous memory using data spaces.

See also

[L4Re::Dataspace](#) and [L4Re::Rm](#).

## 4.7 Program Input and Output

The initial environment provides a Vcon capability used as the standard input/output stream.

Output is usually connected to the parent of the program and displayed as debugging output. The standard output is also used as a back end to the C-style printf functions and the C++ streams.

Vcon services are implemented in Moe and the loader as well as by the [L4Re](#) Microkernel and connected either to the serial line or to the screen if available.

See also

[Virtual Console](#)

## 4.8 Initial Memory Allocator and Factory

The purpose of the memory allocator and of the factory is to provide the application with the means to allocate memory (in the form of data spaces) and kernel objects respectively.

An initial memory allocator and an initial factory are accessible via the initial [L4Re](#) environment.

See also

[L4Re::Mem\\_alloc](#)

The factory is a kernel object that provides the ability to create new kernel objects dynamically. A factory imposes a resource limit for kernel memory, and is thus a means to prevent denial of service attacks on kernel resources. A factory can also be used to create new factory objects.

See also

[Factory](#)

## 4.9 Application and Server Building Blocks

So far we have discussed the environment of applications in which a single thread runs and which may invoke services provided through their initial objects.

In the following we describe some building blocks to extend the application in various dimensions and to eventually implement a server which implements user-level objects that may in turn be accessed by other applications and servers.

### 4.9.1 Creating Additional Application Threads

To create application threads, one must allocate a stack on which this thread may execute, create a thread kernel object and setup the information required at startup time (instruction pointer, stack pointer, etc.). In [L4Re](#) this functionality is encapsulated in the pthread library.

### 4.9.2 Providing a Service

In capability systems, services are typically provided by transferring a capability to those applications that are authorised to access the object to which the capability refers to.

Let us discuss an example to illustrate how two parties can communicate with each other: Assume a simple file server, which implements an interface for accessing individual files: `read(pos, buf, length)` and `write(pos, data, length)`.

L4Re provides support for building servers based on the class `L4::Server_object`. `L4::Server_object` provides an abstract interface to be used with the `L4::Server` class. Specific server objects such as, in our case, files inherit from `L4::Server_object`. Let us call this class `File_object`. When invoked upon receiving a message, the `L4::Server` will automatically identify the corresponding server object based on the capability that has been provided to its clients and invoke this object's `dispatch` function with the incoming message as a parameter. Based on this message, the server must then decide which of the protocols it implements was invoked (if any). Usually, it will evaluate a protocol specific opcode that clients are required to transmit as one of the first words in the message. For example, assume our server assigns the following opcodes: `Read = 0` and `Write = 1`. The `dispatch` function calls the corresponding server function (i.e., `File_object::read()` or `File_object::write()`), which will in turn parse additional parameters given to the function. In our case, this would be the position and the amount of data to be read or written. In case the write function was called the server will now update the contents of the file with the data supplied. In case of a read it will store the requested part of the file in the message buffer. A reply to the client finishes the client request.

## 4.10 Pthread Support

L4Re supports the standard pthread library functionality.

Therefore L4Re itself does not contain any documentation for pthreads itself. Please refer to the standard pthread documentation instead.

The L4Re specific parts will be described herein.

- Include pthread-l4.h header file:

```
#include <pthread-l4.h>
```

- Return the local thread capability of a pthread thread:

Use `pthread_l4_cap(pthread_t t)` to get the capability index of the pthread `t`.

For example:

```
pthread_l4_cap(pthread_self());
```

- Setting the L4 priority of an L4 thread works with a special scheduling policy (other policies do not affect the L4 thread priority):

```
pthread_t t;
pthread_attr_t a;
struct sched_param sp;

pthread_attr_init(&a);
sp.sched_priority = l4_priority;
pthread_attr_setschedpolicy(&a, SCHED_L4);
pthread_attr_setschedparam(&a, &sp);
pthread_attr_setinheritsched(&a, PTHREAD_EXPLICIT_SCHED);

if (pthread_create(&t, &a, pthread_func, NULL))
    // failure...

pthread_attr_destroy(&a);
```

- You can prevent your pthread from running immediately after the call to `pthread_create(..)` by adding `PTHREAD_L4_ATTR_NO_START` to the `create_flags` of the pthread attributes. To finally start the thread you need to call `scheduler()->run_thread()` passing the capability of the pthread and scheduling parameters.

```
pthread_t t;
pthread_attr_t attr;

pthread_attr_init(&attr);
attr.create_flags |= PTHREAD_L4_ATTR_NO_START;

if (pthread_create(&t, &attr, pthread_func, nullptr))
    // failure...

pthread_attr_destroy(&attr);

// do stuff

auto ret = L4Re::Env::env()->scheduler()->run_thread(pthread_l4_cap(t),
                                                    l4_sched_param(2));
if (l4_error(ret))
    // failure...
```

### Constraints on pthread\_t, user-land capability slot, and kernel thread-object

- `pthread_l4_cap()` is guaranteed to return the valid capability slot of the pthread (A) until `pthread_join()` or `pthread_detach()` is invoked on (A)'s `pthread_t`.
- `pthread_l4_cap()` exposes internal state of the pthread management, take the necessary precautions as you would for any shared data in concurrent environments. If you use `pthread_l4_cap()` guarding against concurrency issues is your duty.
- There is no guarantee that a valid capability slot points to a present capability.

#### • Example

It is possible to obtain a valid thread capability slot and for `l4_task_cap_valid()` to return the capability as not present. The following example showcases a possible sequence of events.

```
// Assume: void some_func(void *)
pthread_t pthread = nullptr;
pthread_create(&pthread, nullptr, some_func, nullptr);

// pthread running some_func()
l4_cap_idx_t cap_idx = pthread_l4_cap(pthread);
l4_is_valid_cap(cap_idx); // --> true

long valid = l4_task_cap_valid(L4RE_THIS_TASK_CAP, cap_idx).label();
// valid == 1 --> capability object is present (refers to a kernel object).

// some_func() exits

cap_idx = pthread_l4_cap(pthread);
l4_is_valid_cap(cap_idx); // --> true

valid = l4_task_cap_valid(L4RE_THIS_TASK_CAP, cap_idx).label();
// valid == 0 --> capability object is not present (refers to no kernel object).

pthread_join(pthread, nullptr); // invalidates the cap slot and frees
                                // the pthread's data structures

// using cap_idx here is undefined behavior.
```

## 4.11 Interface Definition Language

An interface definition in L4Re is normally declared as a class derived from `L4::Kobject_t`.

For example, the [simple calculator example](#) declares its interface like that:

```
struct Calc : L4::Kobject_t<Calc, L4::Kobject>
{
```



```

L4_INLINE_RPC(long, sub, (l4_uint32_t a, l4_uint32_t b, l4_uint32_t *res));
L4_INLINE_RPC(long, neg, (l4_uint32_t a, l4_uint32_t *res));

typedef L4::Typeid::Rpc<sub_t, neg_t> Rpc;
};

```

The signature of each function is first declared using one of the RPC macros (see below) and then all the functions need to be listed in the `Rpcs` type.

Clients invoke these functions with the name given to the RPC macros, `sub` and `neg` above. Servers implement them by defining functions with an `op_` prepended, `op_sub` and `op_neg`. The types of the parameters in the macro definition, on the server side, and on the client side are not the same. The following section describes how they are related to each other.

### 4.11.1 Parameter types for RPC

Generally all value parameters, const reference parameters, and const pointer parameters to an RPC interface are considered as input parameters for the RPC and are transmitted from the client to the server.

#### Note

This means that `char const *` is treated as an input `char` and not as a zero terminated string value, for strings see `L4::lpc::String<>`.

Parameters that are non-const references or non-const pointers are treated as output parameters going from the server to the client.

There are special data types that appear on only one side (client or server) when used, see the following table for details.

```
L4_RPC(long, test, (int arg1, char const *arg2, unsigned *ret1));
```

The example shows the declaration of a method called `test` with `long` as return type, `arg1` is an `int` input, `arg2` a `char` input, and `ret1` an unsigned output parameter.

Type	Direction	Client-Type	Server-Type
T	Input	T	T
T const &	Input	T const &	T const &
T const *	Input	T const *	T const &
T &	Output	T &	T &
T *	Output	T *	T &
<a href="#">L4::Ipc::In_out&lt;T &amp;&gt;</a>	In/Out	T &	T &
<a href="#">L4::Ipc::In_out&lt;T *&gt;</a>	In/Out	T *	T &
<a href="#">L4::Ipc::Cap&lt;T&gt;</a>	Input	<a href="#">L4::Ipc::Cap&lt;T&gt;</a>	<a href="#">L4::Ipc::Snd_fpage</a>
<a href="#">L4::Ipc::Out&lt;L4::Cap&lt;T&gt;&gt;</a>	Output	<a href="#">L4::Cap&lt;T&gt;</a>	<a href="#">L4::Ipc::Cap&lt;T&gt; &amp;</a>
<a href="#">L4::Ipc::Rcv_fpage</a>	Input	<a href="#">L4::Ipc::Rcv_fpage</a>	void
<a href="#">L4::Ipc::Small_buf</a>	Input	<a href="#">L4::Ipc::Small_buf</a>	void

Array types can be used to transmit arrays of variable length. They can either be stored in a client-provided buffer ([L4::lpc::Array](#)), copied into a server-provided buffer ([L4::lpc::Array\\_in\\_buf](#)) or directly read and written into the UTCB ([L4::lpc::Array\\_ref](#)).

Constraints on [L4::lpc::Array\\_ref](#):

- the start position of this array type needs to be known in advance.
- it must be the last parameter of a message.
- the size of the array type is not transmitted to the server, only the client side knows the intended size of the input array. The server assumes the rest of the UTCB as the actual array. Different sever-side behavior must be steered otherwise, e.g. through another parameter.

Type	Direction	Client-Type	Server-Type
<code>L4::Ipc::Array&lt;const T&gt;</code>	Input	<code>L4::Ipc::Array&lt;const T&gt;</code>	<code>L4::Ipc::Array_ref&lt;const T&gt;</code>
<code>L4::Ipc::Array&lt;const T&gt;</code>	Input	<code>L4::Ipc::Array&lt;const T&gt;</code>	<code>L4::Ipc::Array_in_buf&lt;const T&gt;</code>
<code>L4::Ipc::Array&lt;T&gt; &amp;</code>	Output	<code>L4::Ipc::Array&lt;T&gt; &amp;</code>	<code>L4::Ipc::Array_ref&lt;T&gt; &amp;</code>
<code>L4::Ipc::Array_ref&lt;T&gt; &amp;</code>	Output	<code>L4::Ipc::Array_ref&lt;T&gt; &amp;</code>	<code>L4::Ipc::Array_ref&lt;T&gt; &amp;</code>

Finally, there are some optional types where the sender can choose if the parameter should be included in the message. These types are for the implementation of some legacy message formats and should generally not be needed for the definition of ordinary interfaces.

Type	Direction	Client-Type	Server-Type
<code>L4::Ipc::Opt&lt;T&gt;</code>	Input	<code>L4::Ipc::Opt&lt;T&gt;</code>	<code>T</code>
<code>L4::Ipc::Opt&lt;const T*&gt;</code>	Input	<code>L4::Ipc::Opt&lt;const T*&gt;</code>	<code>T</code>
<code>L4::Ipc::Opt&lt;T &amp;&gt;</code>	Output	<code>T &amp;</code>	<code>L4::Ipc::Opt&lt;T&gt; &amp;</code>
<code>L4::Ipc::Opt&lt;T *&gt;</code>	Output	<code>T *</code>	<code>L4::Ipc::Opt&lt;T&gt; &amp;</code>
<code>L4::Ipc::Opt&lt;Array↔_ref&lt;T&gt; &amp;&gt;</code>	Output	<code>Array_ref&lt;T&gt; &amp;</code>	<code>L4::Ipc::Opt&lt;Array↔_ref&lt;T&gt;&gt; &amp;</code>

### 4.11.2 Server Side Interface

The server side function signature for the Calc example above is

```
long op_sub(Calc::Rights, 14_uint32_t a, 14_uint32_t b, 14_uint32_t &res);
```

The first rights parameter is a bitfield encoding the rights the client has on the used capability in the lower two bits. Currently, the W-right and S-right are supported. The rest of the Rights-bitfield is reserved.

The second and third arguments are input parameters as can be deduced from the tables above. The fourth parameter is an output parameter, delivered to the client with the return value of type long.

### 4.11.3 RPC Return Types

On the server side, the return type of an RPC handling function is always long. The return value is transmitted via the label field in `l4_msgtag_t` and is therefore restricted to its length. Per convention, a negative return value is interpreted as an error condition. If the return value is negative, output parameters are not transmitted back to the client.

**Attention**

The client must never rely on the content of output parameters when the return value is negative.

On the client-side, the return value of the RPC is set as defined in the RPC macro. If `l4_msgtag_t` is given, then the client has access to the full message tag, otherwise the return type should be `long`. Note that the client might not only receive the server return value in response but also an IPC error code.

**4.11.4 RPC Method Declaration**

RPC member functions can be declared using one of the following C++ macros.

For inline RPC stubs, where the RPC stub code itself is `inline`:

- `L4_INLINE_RPC(res, name, (args...), flags)`  
Define an inline RPC call (type and callable).
- `L4_INLINE_RPC_OP(op, res, name, (args...), flags)`  
Define an inline RPC call with specific opcode (type and callable).
- `L4_INLINE_RPC_NF(res, name, (args...), flags)`  
Define an inline RPC call type (the type only, no callable).
- `L4_INLINE_RPC_NF_OP(opcode, Ret_type, func_name, (args...), flags)`  
Define an inline RPC call type with specific opcode (the type only, no callable).

For external RPC stubs, where the RPC stub code must be defined in a separate compile unit (usually a `.cc` file):

- `L4_RPC(Ret_type, func_name, (args...), flags)`  
Define an RPC call (type and callable).
- `L4_RPC_OP(opcode, Ret_type, func_name, (args...), flags)`  
Define an RPC call with specific opcode (type and callable).
- `L4_RPC_NF(Ret_type, func_name, (args...), flags)`  
Define an RPC call type (the type only, no callable).
- `L4_RPC_NF_OP(opcode, Ret_type, func_name, (args...), flags)`  
Define an RPC call type with specific opcode (the type only, no callable).

To generate the implementation of an external RPC stub:

- `L4_RPC_DEF(class_name::func_name)`  
Generate the definition of an RPC stub.

The NF versions of the macro generally do not generate a callable member function named `<name>` but do only generate the type `<name>_t`. This data type can be used to call the RPC stub explicitly using `<name>_t::call(L4::Cap<Iface_class> cap, args...)`.

**4.12 L4Re Build System**

[L4Re](#) uses a custom make-based build system, often simply referred to as *BID*.

This section explains how to use BID when writing applications and libraries for [L4Re](#).

### 4.12.1 Building L4Re

#### Setting up the Build Directory

L4Re must be built out-of-source. Therefore the first mandatory step is creating and populating a build directory. From the root of the L4Re source tree run

```
make B=<builddir>
```

Other targets that can be executed in the source directory are

#### update

Update the source directory from svn. Only makes sense when you have downloaded L4Re from the official subversion repository.

#### help

Show a short help with the most important targets.

#### Invoking Make

Once the build directory is set up, BID make can be invoked in one of two ways:

1. Go to the build directory and invoke make without special options.
2. Go to a source directory with a BID make file and invoke `make O=<builddir> . . .`

The default target builds the source (as you would expect), other targets that are available in build mode are

#### cleanfast

Quickly cleans the build directory by removing all subdirectories that contain generated files. The configuration will remain untouched.

#### clean

Remove generated files. Slower than `make cleanfast` but can be used on selected packages. Use `S= . . .` to select the target package.

In addition to these targets, there are a number of targets to generate images which are explained elsewhere.

### 4.12.2 Writing BID Make Files

The BID build system exports different roles that define what should be done in the subdirectory. So a BID make file essentially consists of defining the role and a number of role-dependent make variables. The basic layout should look like this:

```
PKGDIR  ?= <path to package's root directory> # e.g., '.' or '..'
L4DIR   ?= <path to L4Re source directory>    # e.g. '$(PKGDIR)/../..'

<various definitions>

include $(L4DIR)/mk/<role>.mk
```

PKGDIR in the first line defines the root directory of the current package. L4DIR in the next line must be pointed to the root of the L4Re source tree the package should be built against. After this custom variable definitions for the role follow. In the final line of the file, the make file with the role-specific rules must be sourced.

The following roles are currently defined:

- project.mk - Sub-project Role
- subdir.mk - Directory Role
- [prog.mk - Application Role](#)
- lib.mk - Library Role
- [include.mk - Header File Role](#)
- doc.mk - Documentation Role
- [test.mk - Test Application Role](#)
- idl.mk - IDL File Role (currently unused)
- runux.mk - Tests in FiascoUX Role

### BID-global Variables

This section lists variables that configure how the BID build system behaves. They are applicable for all roles.

Variable	Description
CC	C compiler for target
CXX	C++ compiler for target
HOST_CC	C compiler for host
HOST_CXX	C++ compiler for host

### 4.12.3 prog.mk - Application Role

The prog role is used to build executable programs.

## General Configuration Variables

The following variables can only be set globally for the Makefile:

MODE

Kind of target to build for. The following values are possible:

- `static` - build a statically linked binary (default)
- `shared` - build a dynamically linked binary
- `l4linux` - build a binary for running on L4Linux on the target platform
- `host` - build for host system
- `targetsys` - build a binary for the target platform with the compiler's default settings

SYSTEMS

List of architectures the target can be built for. The entries must be space-separated entries either naming an architecture (e.g. `amd64`) or an architecture and ABI (e.g. `arm-l4f`). When not defined, the target will be built for all possible platforms.

TARGET

Name or names of the binaries to compile. This variable may also be postfixed with a specific architecture.

## Target-specific Configuration Variables

The following variables may either be used with or without a description suffix. Without suffix they will be used for all operations. With a specific description their use is restricted to a subset. These specifications include a target file and the architecture, both optional but in this order, separated by underscores. The specific variables will be used in addition to the more general ones.

`SRC_C / SRC_CC / SRC_F / SRC_S`

`.c, .cc, .f90, .S` source files.

`REQUIRES_LIBS`

List of libraries the binary depends on. This works only with libraries that export a `pkg_config` configuration file. Automatically adds any required include and link options.

DEPENDS\_PKGS

List of packages this binary depends on. If one these packages is missing then building of the binary will be skipped.

CPPFLAGS / CFLAGS / CXXFLAGS / FFLAGS / ASFLAGS

Options for the C preprocessor, C compiler, C++ compiler, Fortran compiler and assembler. When used with suffix, the referred element is the source file, not the target file.

LDFLAGS

Options for the linker ld.

LIBS

Additional libraries to link against (with -l).

PRIVATE\_LIBDIR

Additional directories to search for libraries.

CRT0 / CRTN

(expert use only) Files containing custom startup and finish code.

LDSCRIPT

(expert use only) Custom link script to use.

#### 4.12.4 include.mk - Header File Role

The header file role is responsible for installing header file at the appropriate location.

The following variables can be used for customizing the process:

INCSRC\_DIR

Source directory where the headers can be found. Default is the directory where the Makefile resides.

TARGET

List of header files to install. If left undefined, then INCSRC\_DIR will be scanned for files with suffix .h or .i.

Supports the specification of special filenames to allow for different source and target filenames to be installed. The syntax is TARGET<SRC, where a filename including the path of SRC is installed as TARGET. An example is

```
libfoo.h<contrib/libfoo_linux.h
```

which installs the header from the contrib directory under the name without that contrib directory and without the platform specific suffix.

EXTRA\_TARGET

When TARGET is undefined, then add these files to the headers found by scanning the source directory. Has no effect if TARGET has been defined.

The filenames specified allow for the same rule specifications as supported by TARGET.

CONTRIB\_HEADERS

When set, the headers will be installed in \${BUILDDIR}/include/contrib/\${PKGNAME} rather than \${BUILDDIR}/include/l4/\${PKGNAME}.

INSTALL\_INC\_PREFIX

Base directory where to install the headers. Overwrites CONTRIB\_HEADERS. The headers will then be found under \${BUILDDIR}/include/\${INSTALL\_INC\_PREFIX}.

PC\_FILENAME

When set, a pkg\_config configuration file is created with the given name.



### 4.12.5 test.mk - Test Application Role

The test role is very similar to the application role, it also builds an executable binary.

The difference is that it also builds for each target a test script that executes the test target either on the host (MODE=host) or a target platform (currently only qemu).

The role accepts all make variables that are accepted by the application role. The only difference is that the `TARGET` variable is not required. If it is missing, the source directory will be scanned for source files that fit the pattern `test_*.c[c]` and create one target for each of them.

#### Note

It is possible to still use `SRC_C[C]` when targets are determined automatically. In that case the specified sources will be used in addition\* to the main `test_*.c[c]` source.

In addition to the variables above, there are a number of variables that control how the test is executed. All these variables may be used as a global variable that applies to all test or, if the target name is added as a suffix, set for a specific target only.

#### TEST\_TARGET

Name of binary containing the test (default: same as `TARGET`).

#### TARGET\_\$ (ARCH)

When `TARGET` is undefined, these targets are added to the list of targets for the specified architecture. For all targets `SRC_C[C]` files must be defined separately.

#### TEST\_KERNEL\_ARGS

Arguments to append to the kernel command line. These are also appended when specifying custom ones via a `.t`-file's `-f` parameter or when using `-d`.

#### TEST\_EXPECTED

File containing expected output. By default the variable is empty, which means the test binary is expected to produce TAP test output, that can be directly processed. When the `TEST_TAP_PLUGINS` variable is given, `TEST_EXPECTED` is ignored.

#### TEST\_EXPECTED\_REPEAT

Number of times the expected output should be repeated, by default 1. When set to 0 then output is expected to repeat forever. This is particularly useful to make sure that stress tests that are meant to run in an endless loop are still alive. Note that such endless tests can only be run by directly executing the test script. They will be skipped when run in a test harness like `prove`.

**TEST\_TAP\_PLUGINS**

Specify the plugins that are used to process the output of the test run. The syntax is of the values is:

plugin1:arg1=a,arg2=b plugin2:arg=foo

Multiple plugins separated by a space are loaded in order. Spaces are not allowed inside a plugin specification. One or more arguments are optionally passed to the plugin separated by commas and delimited by a colon.

If the variable is not specified the plugins for TAPOutput and OutputMatching (depending on the TEST\_EXPECTED variable) are automatically loaded.

For the supported plugins and their options please refer to their in-line documentation in tool/lib/L4/TapWrapper/↵ Plugin/. The plugin name corresponds to the file stem name in that directory.

**TEST\_TIMEOUT**

Non-standard timeout after which the test run is aborted (useful for tests involving sleep).

**NED\_CFG**

LUA configuration file for startup to give to Ned

**REQUIRED\_MODULES**

Additional modules needed to run the test. By adding [opts] to the name of a module you can add module options that are reflected in the generated modules.list.

**BOOTSTRAP\_ARGS**

Additional parameters to supply to bootstrap.

**QEMU\_ARGS**

Additional parameters to supply to QEMU.

**MOE\_ARGS**

Additional parameters to supply to moe.

**TEST\_ARGS**

Additional arguments for the TEST\_STARTER (tapper-wrapper per default).

**TEST\_ROOT\_TASK**

Alternative root task to be used during a test instead of moe.

**TEST\_ROOT\_TASK\_ARGS**

Arguments passed to TEST\_ROOT\_TASK if TEST\_ROOT\_TASK is different from moe.

`KERNEL_CONF`

Features the [L4Re](#) Microkernel must have been compiled with. A space-separated list of config options as used by Kconfig. `run_test` looks for a `globalconfig.out` file in the same directory as the kernel and checks that all options are enabled. If not, the test is skipped. Has only an effect if the `globalconfig.out` file is present.

`L4RE_CONF`

Features the [L4Re](#) userland must have been compiled with. A space-separated list of config options as used by Kconfig. `run_test` will look for these in the `.kconfig` file in the [L4Re](#) build directory.

`L4LINUX_CONF`

Features the L4Linux kernel must have been compiled with. Similar to `KERNEL_CONF` but checks for a `.config` file in the directory of the L4Linux kernel.

`TEST_SETUP`

Command to execute before the test is run. The test will only be executed if the command returns 0. If the exit code is 69, the test is marked as skipped with the reason provided in the final line of stdout.

`TEST_LOGFILE`

Append output of test execution to the given file unless `TEST_WORKDIR` is given.

`TEST_WORKDIR`

Create logs, temp and other files below the given directory. That directory is taken as base dir for more automatically created subdir levels using the current test path, in order to guarantee conflict-free usage when running many different tests with a common workdir. When `TEST_WORKDIR` is provided then `TEST_LOGFILE` is ignored as it is organized below workdir.

`TEST_TAGS`

List of conditions for tags provided during execution of a test. A tag can be set to 1, set to 0 or be unspecified via `TEST_RUN_TAGS` during execution. Therefore there are 4 possible conditions for a tag that can be specified in `TEST_TAGS`: `tag`, `!tag`, `+tag` and `-tag`. The following table shows the conditions they represent.

<code>TEST_RUN_TAGS \ TEST_TAGS</code>	<code>tag</code>	<code>!tag</code>	<code>+tag</code>	<code>-tag</code>
<code>tag</code> or <code>tag=1</code>	y		y	
unspecified		y	y	
<code>tag=0</code>		y		y

*Example usage:*

The tag `long-running` is used by tests which take a long time and should be skipped by default. These tests are marked with the tag `long-running` unprefixd.

The tag `hardware` is set to 1 at runtime when the tests will run on real hardware. Tests that must not run on real hardware are marked with `!hardware`.

The tag `+impl-def` is used by tests that test implementation details. Due to the nature of this flag we require the "+" prefix to be used, so they are run by default but can be excluded from execution by setting `TEST_RUN_TAGS` to `impl-def=0` at runtime.

If you want to specify multiple tag conditions they need to be separated with a comma.

**TEST\_PLATFORM\_ALLOW and TEST\_PLATFORM\_DENY**

Deny and allow lists of platforms a test is banned from or limited to. If you list platforms in the `TEST_PLATFORM_ALLOW` variable the test will only be run on these listed platforms and will be skipped on any other platform. If you list platforms in the `TEST_PLATFORM_DENY` variable the test will be skipped on the listed platforms and will be run on any other platform. You can only use one of these variables per test, not both. See `mk/platforms/` for the various identifiers.

*Example usage:*

```
# Do not run this test on the Raspberry Pi platform
TEST_PLATFORM_DENY_test_xyz := rpi

# Only run this test on this test on the RCar3 platform.
TEST_PLATFORM_ALLOW_test_abc := rcar3
```

**TAPARCHIVE**

Filename for an archive file to store the resulting TAP output.

In addition to compiled tests, it is also possible to create tests where the test binary or script comes from a different source. These tests must be listed in `EXTRA_TARGET` and for each target a custom `TEST_TARGET` must be provided.

**Running Tests**

The make role creates a test script which can be found in `<builddir>/test/t/<arch>/<api>`. It is possible to organise the tests further in subdirectories below by specifying a `TEST_GROUP`.

To be able to execute the test, a minimal test environment needs to be set up by exporting the following environment variables:

`KERNEL_<arch>, KERNEL`

**L4Re** Microkernel binary to use. The test runner is able to check if the kernel has all features necessary for the test and skip tests accordingly. In order for this to work, the `globalconfig.out` config file from the build directory needs to be available in the same directory as the kernel.

`L4LX_KERNEL_<arch>, L4LX_KERNEL`

L4Linux binary to use. This is only required to run tests in `mode=l4linux`. If no L4Linux kernel is set then these tests will simply be skipped. The test runner is also able to check if the kernel has all features compiled in that are required to run the test successfully (see make variable `L4LINUX_CONF` above). For this to work, the `.config` configuration file from the build directory needs to be available in the same directory as the kernel.

`LINUX_RAMDISK_<arch>, LINUX_RAMDISK`

Ramdisk to mount as root in L4Linux. This is only required to run tests in `mode=l4linux`. If not supplied, L4Linux tests will be skipped. The ramdisk must be set up to start the test directly after the initial startup is finished. The name of the test binary is supplied via the kernel command line option `l4re_testprog`. The `tool/test` directory contains an example script `launch-l4linux-test`, which can be copied onto the ramdisk and started by the init script.

## TEST\_HWCONFIG and TEST\_FIASCOCONFIG

Some userland tests rely on external information about the underlying platform and the configuration of the [L4Re](#) Microkernel to decide whether or not to test specific features or to determine which and how much resources are available. Some examples for this are whether or not virtualization is supported by the platform, how many cores the platform has, how many cores the kernel supports or how much memory the platform provides. To convey this information to these tests you can set the two environment variables `TEST_HWCONFIG` and `TEST_FIASCOCONFIG`.

Using `TEST_HWCONFIG` requires a plain text document containing key-value pairs separated by a `=` symbol. On top of that comment lines starting with `#` are supported. Simply create a plain text file such as the following and set `TEST_HWCONFIG` to its absolute path.

```
VIRTUALIZATION=y
MP=y
NUM_CORES=4
MEMORY=2048
```

Using `TEST_FIASCOCONFIG` is easier since it only needs to contain the absolute path of the `globalconfig.out` file in the [L4Re](#) Microkernel's build directory. The build system will then extract the information when a test is started.

When starting a test the build system will read both files and provide their content as a lua table to the test. A lua script can then make decisions based on them. To simplify some decisions the build system merges some information by itself, e.g. virtualization is only available if both the platform and the [L4Re](#) Microkernel support this feature. More details can be obtained from the perl module in `tool/lib/L4/TestEnvLua.pm`.

In addition to these variables, the following BID variables can be overwritten at runtime: `PT` (for the platform type) and `TEST_TIMEOUT`. You may also supply `QEMU_ARGS` and `MOE_ARGS` which will be appended to the parameters specified in the BID test make file.

Once the environment is set up, the tests can be run either by simply executing all of them from the build directory with

```
make test
```

or executing them directly, like

```
test/t/amd64_amdfam10/14f/14re-core/moe/test_namespace.t
```

or running one or more tests through the test harness `prove`, like

```
prove test/t/amd64_amdfam10/14f/14re-core/moe/test_namespace.t
prove -r test/t/amd64_amdfam10/14f/14re-core/
prove -rv test/t/amd64_amdfam10/14f/14re-core/
```

`TEST_TAGS` allow for a way to include or exclude whole groups of tests during execution, primarily with `prove`. You can specify which tests to run at runtime using one of the following ways:

```
$ test/t/amd64_amdfam10/14f/14re-core/test_one.t --run-tags slow,gtest-shuffle=0
$ test/t/amd64_amdfam10/14f/14re-core/test_one.t -T slow,gtest-shuffle=0
$ prove -r test/t/amd64_amdfam10/14f/14re-core/ : -T slow,gtest-shuffle=0
$ TEST_RUN_TAGS=slow,gtest-shuffle=0 prove -r test/t/amd64_amdfam10/14f/14re-core/
```

For each test tag requirements defined in the corresponding `TEST_TAGS` Makefile variable are tested. If the requirements for tags do not match the test is skipped. The `SKIP` message will provide insight why the test was skipped:

```
$ make test
...
test/t/amd64_amdfam10/test_one.t .... ok
test/t/amd64_amdfam10/test_two.t .... skipped: Running this test requires tag slow to be set to 1.
test/t/amd64_amdfam10/test_three.t .. ok
```

When tags are provided, the tests requiring those tags are now also executed while the tests that forbid them are skipped:

```
$ TEST_RUN_TAGS=slow,gtest-shuffle
$ make test
...
test/t/amd64_amdfam10/test_one.t .... ok
test/t/amd64_amdfam10/test_two.t .... ok
test/t/amd64_amdfam10/test_three.t .. skipped: Running this test requires tag gtest-shuffle to be set to 0 or
```

For further details on how values in `TEST_TAGS` and `TEST_RUN_TAGS` interact, see the help text for `TEST_TAGS`.

## Running Tests in External Programs

You can hand-over test execution to an external program by setting the environment variable `EXTERNAL_TEST_STARTER` to the full path of that program:

```
export EXTERNAL_TEST_STARTER=/path/to/external/test-starter
make test
```

`EXTERNAL_TEST_STARTER`

This variable is evaluated by `tool/bin/run_test` (the backend behind `make test`) and contains the full path to the tool which actually starts the test instead of the test itself.

The `EXTERNAL_TEST_STARTER` can be any program instead of the default execution via `make qemu E=maketest`. Its output is taken by `run_test` as the test output.

Usually it is just a bridge to prepare the test execution, e.g., it could create the test as image and start that image via a simulator.

## Running Tests in a Simulator

Based on above mechanism there is a dedicated external test starter `tool/bin/teststarter-image-telnet.pl` shipped in BID which assumes an image to be started with another program which provides test execution output on a network port.

This can be used to execute tests in a simulator, like this:

```
export EXTERNAL_TEST_STARTER=$L4RE_SRC/tool/bin/teststarter-image-telnet.pl
export SIMULATOR_START=/path/to/configured/simulator-exe
make test
```

After building the image and starting the simulator it contacts the simulator via a network port (sometimes called "telnet" port) to pass-through its execution output as its own output so it gets captured by `run_test` as usual.

The following variables control `teststarter-image-telnet.pl`:

#### `SIMULATOR_START`

This points to the full path of the program that actually starts the prepared test image. Most often this is the frontend script of your simulator environment which is pre-configured so that it actually works in the way that `teststarter-image-telnet.pl` expects from the following settings.

#### `SIMULATOR_IMAGETYPE`

The image type to be generated via `make $SIMULATOR_IMAGETYPE E=maketest`. Default is `elfimage`.

#### `SIMULATOR_HOST`

The simulator will be contacted via socket on that host to read its output. Default is `localhost`.

#### `SIMULATOR_PORT`

The simulator will be contacted via socket on that port to read its output. Default is `11111`.

#### `SIMULATOR_START_SLEEPTIME`

After starting the simulator it waits that many seconds before reading from the port. Default is `1` (second).

### Running tests without `tapper-wrapper`

In case you want to replace the `tapper-wrapper` test starter, you can replace the default one by setting the environment variable `TEST_STARTER` to the path of your test starter. Then your test starter can use the same environment which is normally set up for the default starter, which includes environment variables provided by the build system as well as the test itself. Among these are `SEARCHPATH`, `MODE`, `ARCH`, `MOE_CFG`, `MOE_ARGS`, `TEST_TIMEOUT`, `TEST_TARGET`, `TEST_EXPECTED`, `QEMU_ARGS` and many more.

### Debugging Tests

The test script is only a thin wrapper that sets up the test environment as it was defined in the make file and then executes two scripts: `tapper-wrapper` and `run_test`.

The main work horse of the two is `tool/bin/run_test`. It collects the necessary files and starts `qemu` to execute the test. This script is always required.

There is then a second script wrapped around the test runner: `tool/bin/tapper-wrapper`. This tool inspects the output of the test runner and reformats it, so that it can be read by tools like `prove`. If the test produces tap output, then the script scans for this output and filters away all the debug output. If `TEST_EXPECTED` was defined, then the script scans the output for the expected lines and prints a suitable TAP message with success or failure. It also makes sure that `qemu` is killed as soon as the test is finished.

There are a number of command-line parameters that allow to quickly change test parameters for debugging purposes. Run the test with `'-help'` for more information about available parameters.

## 4.13 Kernel Factory

The kernel factory is a kernel object that provides the ability to create new kernel objects dynamically.

The kernel factory enforces a memory quota. This quota defines the maximum amount of kernel memory the factory service can use to construct the requested objects. When the quota is depleted, the factory refuses the creation of new objects.

The quota may be higher than the amount of available kernel memory; ultimately, the amount of available kernel memory is the strict limit for the factory to remain operational.

The kernel factory creates the following kinds of objects:

- [DMA space](#)
- [L4::Factory](#)
- [L4::lpc\\_gate](#) ([L4\\_PROTO\\_NONE](#), [L4\\_PROTO\\_KOBJECT](#))
- [L4::Irq](#) ([L4\\_PROTO\\_IRQ\\_SENDER](#))
- [L4::Semaphore](#)
- [L4::Task](#)
- [L4::Thread](#)
- [L4::Vm](#)
- [L4::Vcpu\\_context](#)

The protocol IDs for objects in this list are given in [L4\\_msgtag\\_protocol](#). Kernel objects whose protocol ID is not immediately clear from the documentation of [L4\\_msgtag\\_protocol](#) have their protocol IDs stated within parenthesis. As an exception, [L4::lpc\\_gate](#) can be identified by more than one protocol IDs. The protocol ID shall be used as the second argument for [L4::Factory.create](#)([Cap<void>](#), long, [l4\\_utcb\\_t \\*](#)).

For the C++ interface see [L4::Factory](#), for the C interface see [Factory](#).

### 4.13.1 Passing parameters for the create stream

[L4::Factory.create\(\)](#) returns a [create stream](#) that allows arguments to be forwarded to the constructor of the object to be created.

Objects that support additional parameters on their creation are presented with a non-empty list of parameters. The parameters are listed in the order they should be provided to a create stream returned by [L4::Factory.create\(\)](#).

- [Dmar\\_space\(\)](#)
- [L4::Factory\(l4\\_umword\\_t\)](#)
  - Argument: factory quota (in bytes).
  - See [L4::Factory.create\\_factory\(\)](#) for details.
- [L4::lpc\\_gate\(\)](#)
  - Creates an unbound IPC gate.
  - Alternatively, an IPC gate can be immediately bound to a thread upon creation using [L4::Factory.create\\_gate\(\)](#).



- [L4::Irq\(\)](#)
- [L4::Semaphore\(\)](#)
- [L4::Task\(l4\\_fpage\\_t\)](#)
  - Argument: utcb\_area
  - See [L4::Factory.create\\_task\(\)](#) for details.
- [L4::Thread\(\)](#)
- [L4::Vm\(\)](#)
- [L4::Vcpu\\_context\(\)](#)



## Chapter 5

# L4Re Servers

Here you shall find a quick overview over the standard services running on the [L4Re](#) Microkernel.

### Sigma0, the Root Pager

Sigma0 is a special server running on [L4](#) because it is responsible of resolving page faults for the root task, the first useful task on [L4Re](#). Sigma0 can be seen as part of the kernel, however it runs in unprivileged mode. To run something useful on the [L4Re](#) Microkernel you usually need to run Sigma0, nevertheless it is possible to replace Sigma0 by a different implementation.

For more details see [Sigma0, the Root-Pager](#)

### Moe, the Root Task

Moe is our implementation of the [L4](#) root task that is responsible for bootstrapping the system, and to provide basic resource management services to the applications on top. Therefore Moe provides [L4Re](#) resource management and multiplexing services:

- **Memory** in the form of memory allocators ([L4Re::Mem\\_alloc](#), [L4::Factory](#)) and data spaces ([L4Re::Dataspace](#))
- **Cpu** in the form of basic scheduler objects ([L4::Scheduler](#))
- **Vcon** multiplexing for debug output (output only)
- **Virtual memory management** for applications, [L4Re::Rm](#)

Moe further provides an implementation of [L4Re](#) name spaces ([L4Re::Namespace](#)), which are for example used to provide a read only directory of all multi-boot modules. In the case of a boot loader, like grub that enables a VESA frame buffer, there is also a single instance of an [L4Re](#) graphics session ([L4Re::Goos](#)).

To start the system Moe starts a single ELF program, the init process. The init process (usually Ned, see the next section) gets access to all resources managed by Moe and to the Sigma0 root pager interface.

For more details see [Moe, the Root-Task](#).

## Ned, the Default Init Process

To keep the root task free from complicated scripting engines and to avoid circular dependencies in application startup (that could lead to dead locks) the configuration and startup of the real system is managed by an extra task, the init process.

Ned is such an init process that allows system configuration via Lua scripts.

For more information see [Ned](#).

## Io, the Platform and Device Resource Manager

Because all peripheral management in [L4Re](#) is done in user-level applications, there is the need to have a centralized management of the resources belonging to the platform and to peripheral devices.

This is the job of Io. Io provides portable abstractions for iterating and accessing devices and their resources (IRQ's, IO Memory...), as well as delegating access to those resources to other applications (e.g., device drivers).

For more details see [Io, the Io Server](#).

## Other Servers

The following additional server package are available on top of the core [L4Re](#) environment.

- [Rtc, the Real-Time Clock Server](#)  
is a simple multiplexer for real-time clock hardware on your platform.
- [fb-drv, the Low-Level Graphics Driver](#)  
provides low-level access and initialization of various graphics hardware. It has support for running VESA BIOS calls on Intel x86 platforms, as well as support for various ARM display controllers. `fb-drv` provides a single instance of the `L4Re::Goos` interface and can serve as a back end for the Mag server, in particular, if there is no graphics support in the boot loader.
- [l4vio\\_net\\_p2p, a virtual network point-to-point link](#)
- [Virtio Net Switch, a virtual network switch](#)
- [Uvmm, the virtual machine monitor](#)
- [RTC driver](#)
- [NVMe server](#)
- [Mag, the GUI Multiplexer](#)
- [Sigma0, the Root-Pager](#)
- [eMMC driver](#)
- [Cons, the Console Multiplexer](#)
- [AHCI driver](#)

## 5.1 Sigma0, the Root-Pager

Sigma0 is a special [L4](#) server that serves as the origin for mapping memory.

It is started by Fiasco.OC on the system boot and gets full access to all userland RAM and device memory. It functions as the pager (main memory provider) for Moe and as the provider for device memory for Io. Moe and Io are trusted and usually the only applications besides Ned that get a capability for Sigma0. Memory can be requested from Sigma0 directly via an IPC, or indirectly by causing page faults and having them resolved by Sigma0.

### 5.1.1 Factory

There is only one instance of Sigma0 in an [L4Re](#) system, which is made accessible to Moe via an IPC gate capability. Using this capability, Moe can request Sigma0 to create new communication channels to itself by creating additional IPC gate capabilities. This request is done using the [L4::Factory](#) interface. This is the only kind of object that can be created by the factory in Sigma0.

List of objects that the Sigma0 Factory can create:

- Sigma0 ()
  - Use protocol id [L4\\_PROTO\\_SIGMA0](#) for creation
  - No arguments supported

See also

[Sigma0 API](#)

## 5.2 Moe, the Root-Task

Moe is the default root-task implementation for L4Re-based systems.

*Moe* is the first task which is usually started in L4Re-based systems. The micro kernel starts *Moe* as the Root-Task.

### 5.2.1 Moe objects

Moe provides a default implementation for the basic [L4Re](#) abstractions, such as data spaces ([L4Re::Dataspace](#)), region maps ([L4Re::Rm](#)), memory allocators ([L4::Factory](#), [L4Re::Mem\\_alloc](#)), name spaces ([L4Re::Namespace](#)) and so on (see [L4Re Interface](#)). These are described in the following subsections.

### 5.2.1.1 Factory

The factory in Moe is responsible for all kinds of dynamic object allocation.

Moe's factory allows allocation of the following objects:

- [L4Re::Namespace](#)
- [L4Re::Dataspace](#), RAM allocation
- [L4Re::Dma\\_space](#), memory management for DMA-capable devices
- [L4Re::Rm](#), virtual memory management for application tasks
- [L4::Vcon](#) (output only)
- [L4::Scheduler](#), to provide a restricted priority / CPU range for clients
- [L4::Factory](#), to provide a quota limited allocation for clients

#### Note

[L4::Scheduler](#) objects can be only created through the user factory provided by Moe to the initial application. Other factory instances cannot create this object.

#### 5.2.1.1.1 Passing parameters to the create stream

[L4::Factory.create\(\)](#) returns a [create stream](#) that allows arguments to be forwarded to the object creation in Moe.

Objects that support additional parameters on their creation are presented next with a non-empty list of parameters. The parameters are listed in the order they should be provided to a create stream. Optional parameters are identified by their default values. Multiple entries in the next list denote different ways of initializing an object.

- [L4Re::Namespace](#) ()
  - For more details see [Namespace](#)
- [L4Re::Dataspace](#) (`l4_mword_t size`, `l4_umword_t flags = 0`, `l4_umword_t align = 0`)
  - Argument `size`: size in bytes (mandatory)
  - Argument `flags`: special dataspace properties, see [L4Re::Mem\\_alloc::Mem\\_alloc\\_flags](#)
  - Argument `align`: Log2 alignment of dataspace if supported by allocator
  - See detailed description of the parameters in [L4Re::Mem\\_alloc::alloc\(\)](#)
  - For details on the types of dataspace provided by Moe, see [Dataspace](#)
- [L4Re::Dma\\_space](#) ()
  - For more details see [DMA Space](#)
- [L4Re::Rm](#) ()
  - For more details see [Region Map](#)
- [L4::Vcon](#) (`char const *label`, `l4_mword_t color = 7`)
  - Argument `label`: label used as prefix for the console output
  - Argument `color`: color code 0..15

- For more details see [Log Subsystem](#)
- **L4::Vcon** (`char const *label, char const *color = "w"`)
  - Argument `label`: label used as prefix for the console output
  - Argument `color`: color code
    - \* The color is identified by a single character
    - \* Supported colors: N, n, R, r, G, g, Y, y, B, b, M, m, C, c, W, w
  - For more details see [Log Subsystem](#)
- **L4::Scheduler** (`l4_mword_t limit, l4_mword_t offset, l4_umword_t bitmap = ~0UL, ...`)
  - Argument `limit`: maximum priority
  - Argument `offset`: priority offset
  - Argument `bitmap`: bitmap of CPUs - can be repeated to address higher order CPUs
  - Argument `limit` must be greater than `offset`
  - For more details see [Scheduler subsystem](#)
- **L4::Factory** (`l4_mword_t quota`)
  - Argument `quota`: limit in bytes (not zero)
  - The limit is deducted from the limit of the factory that creates the new factory

### 5.2.1.2 Namespace

Moe provides a name space conforming to the [L4Re::Namespace](#) interface (see [Name-space API](#)). Per default Moe creates a single name space for the [Boot FS](#). That is available as `rom` in the initial objects of the init process.

#### 5.2.1.2.1 Boot FS

The Boot FS subsystem provides access to the files loaded during the platform boot (or available in ROM). These files are either linked into the boot image or loaded via a flexible boot loader, such as GRUB.

The subsystem provides an [L4Re::Namespace](#) object as directory and an [L4Re::Dataspace](#) object for each file.

By default all files are read only and visible in the namespace `rom`. As an option, files can be supplied with the argument `:rw` to mark them as writable modules. Moe will allow read and write access to these dataspace and make them visible in a different namespace called ``rwfs``.

An example entry in 'modules.list' would look like this:

```
module somemodule :rw
```

#### Note

In order for a client to receive write permissions to the dataspace, the corresponding cap also needs write permissions.

### 5.2.1.3 Dataspace

Dataspaces can be allocated with an arbitrary size. The granularity for memory allocation however is the machine page size ([L4\\_PAGESIZE](#)). A dataspace user must be aware that, as a result of this page-size granularity, there may be padding memory at the end of a dataspace which is accessible to each client. Moe currently allows most dataspace operations on this padding area. Nonetheless, the client must not make any assumptions about the size or content of the padding area, as this behaviour might change in the future.

The provided data spaces can have different characteristics:

- Physically contiguous and pre-allocated
- Non contiguous and on-demand allocated with possible copy on write (COW)

Dataspaces allocated via the Moe's factory allow mappings with any combination of the read-write-execute (RWX) rights, subject to a possible restriction of the writable right for client capabilities lacking the 'W' right.

### 5.2.1.4 Log Subsystem

The logging facility of Moe provides per application tagged and synchronized log output.

### 5.2.1.5 DMA Space

### 5.2.1.6 Scheduler subsystem

The scheduler subsystem provides a simple scheduler proxy for scheduling policy enforcement.

The priority offset provided on the creation of a scheduler proxy defines the minimum priority assigned to threads which are scheduled by that instance of the scheduler proxy. The offset is implicitly added to priorities provided to [L4::Scheduler.run\\_thread\(\)](#).

### 5.2.1.7 Region Map

## 5.2.2 Command Line Options

Moe's command-line syntax is:

```
moe [--debug=<flags>] [--init=<binary>] [--l4re-dbg=<flags>] [--ldr-flags=<flags>] [-- <init options>]
```

```
--debug=<debug flags>
```

This option enables debug messages from Moe itself, the `<debug flags>` values are a combination of `info`, `warn`, `boot`, `server`, `loader`, `exceptions`, and `ns` (or all for full verbosity).

```
--init=<init process>
```

This options allows to override the default init process binary, which is 'rom/ned'.



```
--l4re-dbg=<debug flags>
```

This option allows to set the debug options for the [L4Re](#) runtime environment of the init process. The flags are the same as for `--debug=`.

```
--ldr-flags=<loader flags>
```

This option allows setting some loader options for the [L4Re](#) runtime environment. The flags are `pre_alloc`, `all_segs_cow`, and `pinned_segs`.

```
--brk=<address>
```

This option is only present on systems without MMU. It restricts dynamic allocations to addresses equal or higher than `<address>`. The argument is parsed as hexadecimal number without any `0x` prefix. Use it to prevent moe from allocating memory in regions that shall later be used by other applications or virtual machines.

```
-- <init options>
```

All command-line parameters after the special `--` option are passed directly to the init process.

## 5.3 Ned, the Init Process

Ned's job is to bootstrap the system running on [L4Re](#).

The main thing to do here is to coordinate the startup of services and applications as well as to provide the communication channels for them. The central facility in Ned is the Lua (<http://www.lua.org>) script interpreter with the [L4Re](#) and ELF-loader bindings.

The boot process is based on the execution of one or more Lua scripts that create communication channels (IPC gates), instantiate other [L4Re](#) objects, organize capabilities to these objects in sets, and start application processes with access to those objects (or based on those objects).

For starting applications, Ned depends on the services of [Moe, the Root-Task](#) or another *loader*, which must provide data spaces and region maps. Ned also uses the 'rom' capability as source for Lua scripts and at least the 'l4re' binary (the runtime environment core) running in each application.

Each application Ned starts is equipped with an [L4Re::Env](#) environment that provides information about all the initial objects made accessible to this application.

### 5.3.1 Lua Bindings for L4Re

Ned provides various bindings for [L4Re](#) abstractions. These bindings are located in the '[L4](#)' package (`require "L4"`).

#### 5.3.1.1 Tutorial

For a verbose example using the Ned Lua bindings, see [tutorial.lua](#).

### 5.3.1.2 Capabilities in Lua

Capabilities are handled as normal values in Lua. They can be stored in normal variables or Lua compound structures (tables). A capability in Lua possesses additional information about the access rights that shall be transferred to other tasks when the capability is transferred. To support implementation of the Principle of Least Privilege, minimal rights are assigned by default. Extended rights can be added using the method `mode("...")` (short `m("...")`) that returns a new reference to the capability with the given rights.

#### Note

It is generally impossible to elevate the real access rights to an object. This means that if Ned has only restricted rights to an object it is not possible to upgrade the access rights with the `mode` method.

The capabilities in Lua also carry dynamic type information about the referenced objects. They thereby provide type-specific operations on the objects, such as the `create` operation on a generic factory or the `query` and `register` operations on a name space.

### 5.3.1.3 Access to L4Re::Env Capabilities

The initial objects provided to Ned itself are accessible via the table `L4.Env`. The default (usually unnamed) capabilities are accessible as `factory`, `log`, `mem_alloc`, `parent`, `rm`, and `scheduler` in the `L4.Env` table.

### 5.3.1.4 Constants

#### Protocols

The protocol constants are defined by default in the [L4](#) package's table `L4.Proto`. The definition is not complete and only covers what is usually needed to configure and start applications. The protocols are for example used as first argument to the `Factory:create` method.

```
Proto = {
  Dataspace = 0x4000,
  Namespace = 0x4001,
  Goos      = 0x4003,
  Mem_alloc = 0x4004,
  Rm        = 0x4005,
  Event     = 0x4006,
  Inhibitor = 0x4007,
  Sigma0    = -6,
  Log       = -13,
  Scheduler = -14,
  Factory   = -15,
  Vm        = -16,
  Dma_space = -17,
  Irq_sender = -18,
  Semaphore = -20,
  Iommu     = -22,
  Ipc_gate  = 0,
}
```

#### Rights

The rights of a Lua capability can be defined in two ways via the `:mode()` interface. Either via a string representation of the rights or via an integer value. An example for the former is `:mode("rsnc")` while the latter equivalent is `:mode(L4.Rights.r | L4.Rights.s | L4.Rights.n | L4.Rights.c)`. The following listing shows the integer constants. The constant names can be used in the string parameter to `:mode()`.

```
Rights = {
  s = 2,
  w = 1,
  r = 4,
  d = 8,
```

```

n    = 16,
c    = 32,
ro   = 4,
rw   = 5,
rws  = 7,
}

```

## Debugging Flags

Debugging flags used for the applications [L4Re](#) core:

```

Dbg = {
    Info      = 1,
    Warn      = 2,
    Boot      = 4,
    Server     = 0x10,
    Exceptions = 0x20,
    Cmd_line   = 0x40,
    Loader     = 0x80,
    Name_space = 0x400,
    All       = 0xffffffff,
}

```

## Loader Flags

Flags for configuring the loading process of an application.

```

Ldr_flags = {
    eager_map      = 0x1, -- L4RE_AUX_LDR_FLAG_EAGER_MAP
    all_segs_cow  = 0x2, -- L4RE_AUX_LDR_FLAG_ALL_SEGS_COW
    pinned_segs   = 0x4, -- L4RE_AUX_LDR_FLAG_PINNED_SEGS
}

```

### 5.3.1.5 Application Startup Details

The central facility for starting a new task with Ned is the class `L4.Loader`. This class provides interfaces for conveniently configuring and starting programs. It provides three operations:

- `new_channel()` Returns a new IPC gate that can be used to connect two applications
- `start()` and `startv()` Start a new application process and return a process object

The `new_channel()` call is used to provide a service application with a communication channel to bind its initial service to. The concrete behavior of the object and the number of IPC gates required by a server depends on the server implementation. The channel can be passed to client applications as well or can be used for operations within the script itself.

`start()` and `startv()` always require at least two arguments. The first one is a table that contains information about the initial objects an application shall get. The second argument is a string, which for `start()` is the program name plus a white-space-separated list of program arguments (`argv`). For `startv()` the second argument is just the program binary name – which may contain spaces –, and the program arguments are provided as separate string arguments following the binary name (allowing spaces in arguments, too). The last optional argument is a table containing the POSIX environment variables for the program.

The Loader class uses reasonable defaults for most of the initial objects. However, you can override any initial object with some user-defined values. The main elements of the initial object table are:

- `factory` The factory used by the new process to create new kernel objects, such as threads etc. This must be a capability to an object implementing the [L4::Factory](#) protocol and defaults to the factory object provided to Ned.
- `mem` The memory allocator provided to the application and used by Ned allocates data spaces for the process. This defaults to Ned's memory allocator object (see [L4Re::Mem\\_alloc](#)).

- `rm_fab` The generic factory object used to allocate the region-map object for the process. (defaults to Ned's memory allocator).
- `log_fab` The generic factory to create the [L4Re::Log](#) object for the application's output (defaults to Ned's memory allocator). The `create` method of the `log_fab` object is called with `log_tag` and `log_color`, from this table, as arguments.
- `log` A table with parameters passed to the `log_fab`:
  - The first item is a short string defining the tag used for tagging log output of this process (defaults to the program name).
  - The second item is a string defining the color used for the log tag and the log string (defaults to "white").
  - Further parameters might be evaluated by certain implementations of the [L4Re::Log](#) interface.
- `scheduler` The scheduler object used for the process' threads (defaults to Ned's own scheduler).
- `caps` The table with application-specific named capabilities (default is an empty table). If the table does not contain a capability with the name 'rom', the 'rom' capability from Ned's initial caps is inserted into the table.

Less frequently used parameters:

- `l4re_dbg` An integer value overriding the debug level of the ITAS used for this application. Default is 2 (Warn). See *Debugging Flags* above.
- `ldr_flags` An integer value for setting additional flags for attaching regions, see *Loader Flags* above.

The `start()` and `startv()` calls return a task object that supports a number of operations.

- `state()` returns a string with the current task state: "running" or "zombie" if the task has terminated. If the task was already reaped (`wait()` returned) or if `kill()` was called, then the function will return `nil`.
- `exit_code()` returns the exit code if the task has terminated or `nil` if it was either killed or has been reaped.
- `kill()` forcefully terminates the task. Returns "killed" if the task was terminated or the exit code if the task was already gone. Returns `nil` if the task was already reaped.
- `exit_handler(function)` registers a Lua function that is invoked when the task terminates. If the task has already terminated, the function is called immediately. Returns `true` if the callback is pending, otherwise `false`. The callback function gets the exit code (`nil` if killed) of the task as its only parameter. The return value of the function is ignored. Only one callback can be registered.
- `wait()` suspends execution until the task has terminated. It's better to use `exit_handler()` instead. While the Lua code executes `wait()`, no `exit_handler()` will be dispatched.

### 5.3.1.6 Reacting on task termination

Ned can react on the termination of child tasks. The preferred mechanism is to register an `exit_handler()` for a task:

```
task = L4.default_loader:start(...)
task:exit_handler(function(exit_code)
  if exit_code == nil then
    print("Task was killed")
  else
    print("Task has terminated w/ code [" .. exit_code .. "]")
  end
end)
```

If the task did already terminate then the callback is invoked immediately. It is also possible to suspend execution of the Ned script until a task has terminated:

```
task = L4.default_loader:start(...)
task:wait()
```

This method should be used with caution, though. The Ned script will wait until the child task has terminated. Neither will any of the registered `exit_handler()` will be called during this time, nor will the [remote command interface](#) be able to execute commands either.

### 5.3.1.7 Control scheduling

Scheduling of [L4Re](#) applications is controlled by creating scheduler proxies. The proxy restricts the threads of an application to run on a subset of the available CPUs and to set their minimum and maximum priority. Use the loader factory function to create the proxy and pass it to the application:

```
sched_proxy = L4.default_loader:create_sched_proxy{ cpus=L4.Cpu_set:new("0-3") }
L4.default_loader:start({ scheduler = sched_proxy }, ...)
```

The `create_sched_proxy` function takes the following table arguments. All arguments are optional:

`cpus`: Set of allowed CPUs. Default: all CPUs. `prio_offset`: Base priority of all threads. Default: 0. `prio_limit`: Maximum priority of all threads. Default: `prio_offset + 10`. `fab`: Scheduler proxy factory capability. Default: `loader sched_fab`.

The `Cpu_set` constructor takes any number of CPU numbers or ranges:

```
Cpu_set:new()           -- all CPUs (because no argument was passed)
Cpu_set:new{}          -- empty CPU set (because an empty table was passed)
Cpu_set:new(42)         -- single CPU: 42
Cpu_set:new("0-3")     -- 4 CPUs: 0..3
Cpu_set:new{"0-3", 42} -- 5 CPUs: 0..3, 42
```

A limited number of operations are defined on CPU sets. To compute the union of two sets (all CPUs of both sets), use the `|` operator. To compute the intersection of two sets (all CPUs common to both sets), use the `&` operator.

### 5.3.1.8 Access to the kernel debugger

Applications can enrich the kernel debugger with information using the API defined in [l4/sys/debugger](#). In order to do so, the developer has to assign access to the kernel debugger kernel object to the application. This can be done like this:

```
L4.default_loader:start({ caps = { jdb = L4.Env.jdb; }}, "rom/example")
```

### 5.3.1.9 Using the interactive ned prompt

Ned can be used in interactive mode by connecting the small ned-prompt helper tool to the command capability. Add the following code snippet at the end of your ned script:

```
local L4 = require("L4");
local l = L4.default_loader;

cmd = l:new_channel()

l:start({ log = L4.Env.log, caps = { svr = cmd } }, "rom/ned-prompt")

L4.server_loop(cmd)
```

The script hands in ned's own log capability to `ned-prompt`. This ensures that input and output of ned and the prompt appear on the same console.

`ned-prompt` needs to be added to your modules list.

### 5.3.2 Command Line Options

Ned's command line syntax is:

```
[--exit|--wait-and-exit] [--disable-backtracer-autoload] [--execute|-e STATEMENT] <lua script> [options passed
```

Ned interprets the first non-option argument `<lua script>` as the Lua script which it should load and run. All arguments following the first non-option argument are passed as arguments to the Lua script via Lua's global `arg` table.

- Exit Options: **exit**, **wait-and-exit** (must be first if used)
  - **exit**: terminates the application after the script has run through even if there are still tasks running. `wait` for tasks at the end of the script to ensure they do not die forcefully.
  - **exit-and-wait**: terminates the application after the script has run through and all tasks started by ned have signaled their exit.
- Execute Statement Option: **execute**, **e**  
Execute the Lua statement `STATEMENT`.
- **disable-backtracer-autoload**: Do not load the backtracer automatically.

## 5.4 Io, the Io Server

The Io server handles all platform devices and resources such as I/O memory, ports (on x86) and interrupts, and grants access to those to clients.

Upon startup Io discovers all platform devices using available means on the system, e.g. on x86 the PCI bus is scanned and the ACPI subsystem initialised. Available I/O resource can also be configured via configuration scripts.

Io's configuration consists of two parts:

- the description of the real hardware
- the description of virtual buses

Both descriptions represent a hierarchical (tree) structure of device nodes. Where each device has a set of resources attached to it. And a device that has child devices can be considered a bus.

### Hardware Description

The hardware description represents the devices that are available on the particular platform including their resource descriptions, such as MMIO regions, IO-Port regions, IRQs, bus numbers etc.

The root of the hardware devices is formed by a system bus device (accessible in the configuration via `Io.system↔_bus()`). As mentioned before, platforms that support methods for device discovery may populate the hardware description automatically, for example from ACPI. On platforms that do not have support for such methods you have to specify the hardware description by hand. A simple example for this is `x86-legacy.devs`.

## Virtual Bus Description

Each Io server client is provided with its own virtual bus which it can iterate to find devices. A virtual PCI bus may be a part of this virtual bus.

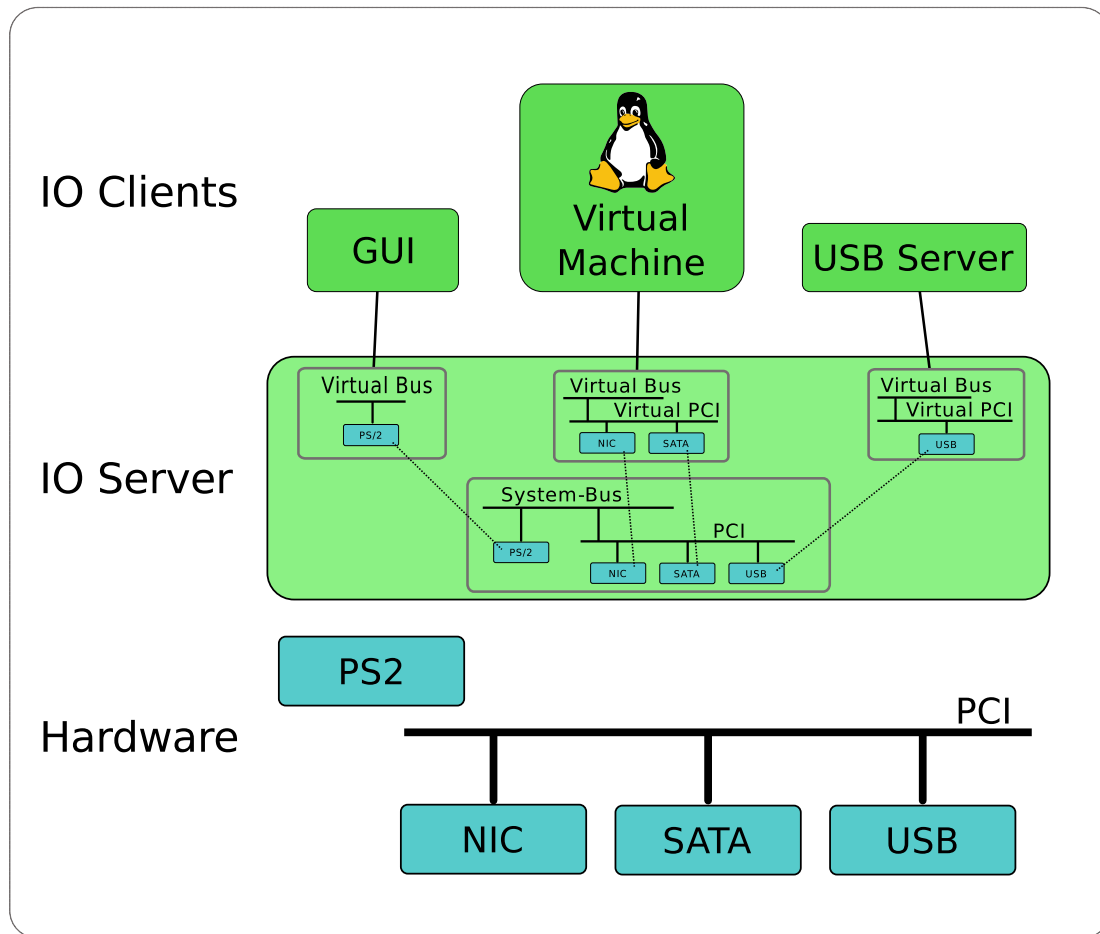


Figure 5.1 IO Service Architecture Overview

The Io server must be configured to create virtual buses for its clients.

This is done with at least one configuration file specifying static resources as well as virtual buses for clients. The configuration may be split across several configuration files passed to Io through the command line.

To allow clients access to available devices, a virtual system bus needs to be created that lists the devices and their resources that should be available to that client. The names of the buses correspond to the capabilities given to Io in its launch configuration.

A very simple configuration for Io could look like this:

```
-- vim:ft=lua
-- Example configuration for io

-- Configure two platform devices to be known to io
Io.Dt.add_children(Io.system_bus(), function()

    -- create a new hardware device called "FOODEVICE"
    FOODEVICE = Io.Hw.Device(function()
        -- set the compatibility IDs for this device
        -- a client tries to match against these IDs and configures
        -- itself accordingly
        -- the list should be sorted from specific to less specific IDs
        compatible = {"dev-foo,mmio", "dev-foo"};

        -- set the 'hid' property of the device, the hid can also be used
```

```

-- as a compatible ID when matching clients
Property.hid = "dev-foo,Example";

-- note: names for resources are truncated to 4 letters and a client
-- can determine the name from the ID field of a l4vbus_resource_t
-- add two resources 'irq0' and 'reg0' to the device
Resource.irq0 = Io.Res.irq(17);
Resource.reg0 = Io.Res.mmio(0x6f000000, 0x6f007fff);
end);

-- create a new hardware device called "BARDEVICE"
BARDEVICE = Io.Hw.Device(function()
  -- set the compatibility IDs for this device
  -- a client tries to match against these IDs and configures
  -- itself accordingly
  -- the list should be sorted from specific to less specific IDs
  compatible = {"dev-bar,mmio", "dev-bar"};

  -- set the 'hid' property of the device, the hid can also be used
  -- as a compatible ID when matching clients
  Property.hid = "dev-bar,Example";

  -- Specify that this device is able to use direct memory access (DMA).
  -- This is needed to allow clients to gain access to DMA addresses
  -- used by this device to directly access memory.
  Property.flags = Io.Hw_device_DF_dma_supported;

  -- note: names for resources are truncated to 4 letters and a client
  -- can determine the name from the ID field of a l4vbus_resource_t
  -- add three resources 'irq0', 'irq1', and 'reg0' to the device
  Resource.irq0 = Io.Res.irq(19);
  Resource.irq1 = Io.Res.irq(20);
  Resource.reg0 = Io.Res.mmio(0x6f100000, 0x6f100fff);
end);
end);

Io.add_vbusses
{
  -- Create a virtual bus for a client and give access to FOODEVICE
  client1 = Io.Vi.System_bus(function ()
    dev = wrap(Io.system_bus():match("dev-foo,mmio"));
  end);

  -- Create a virtual bus for another client and give it access to BARDEVICE
  client2 = Io.Vi.System_bus(function ()
    dev = wrap(Io.system_bus():match("dev-bar,Example"));
  end);
}

```

Each device supports a 'compatible' property. It is a list of compatibility strings. A client matches itself against one (or multiple) compatibility IDs and configures itself accordingly. All other device members are handled according to their type. If the type is a resource (Io.Res) it is added as a named resource. Note that resource names are truncated to 4 letters and are stored in the ID field of a [l4vbus\\_resource\\_t](#). If the type is a device it is added as a child device to the current one. All other types are treated as a device property which can be used to configure a device driver. Right now, device properties are internal to Io only.

## Matching and Assigning PCI Devices

Assigning clients PCI devices could look like this:

```

-- This is a configuration snippet for PCI device selection

local hw = Io.system_bus();

Io.add_vbusses
{
  pciclient = Io.Vi.System_bus(function ()
    PCI = Io.Vi.PCI_bus(function ()
      pci_mm      = wrap(hw:match("PCI/CC_04"));
      pci_net     = wrap(hw:match("PCI/CC_02"));
      pci_storage = wrap(hw:match("PCI/CC_01"));
    end)
  end)
}

```

The "PCI/" is followed by a bus-specific ID string. The format of the PCI ID string may be one of the following:



- PCI/CC\_cc
- PCI/CC\_ccss
- PCI/CC\_ccssp
- PCI/VEN\_vvvv
- PCI/DEV\_dddd
- PCI/SUBSYS\_sssssss
- PCI/REV\_rr
- PCI/ADR\_xxxx:xx:xx.x

Where:

- cc is the hexadecimal representation of the class code byte
- ss is the hexadecimal representation of the subclass code byte
- pp is the hexadecimal representation of the programming interface byte
- vvvv is the hexadecimal representation of the vendor ID
- dddd is the hexadecimal representation of the device ID
- sssssss is the hexadecimal representation of the subsystem ID
- rr is the hexadecimal representation of the revision byte
- xxxx:xx:xx.x is the bus address in PCI nomenclature

As a special extension lo supports replacing the ID string with a human-readable common PCI class name. The following table gives an overview of the names known to lo and their respective PCI class and subclass.

Common Name	Description	PCI ID string
storage	Mass storage controller	CC_01
scsi	SCSI storage controller	CC_0100
ide	IDE interface	CC_0101
floppy	Floppy disk controller	CC_0102
raid	RAID bus controller	CC_0104
ata	ATA controller	CC_0105
sata	SATA controller	CC_0106
sas	Serial attached SCSI controller	CC_0107
nvm	Non-volatile memory controller	CC_0108
-	-	-
network	Network controller	CC_02
ethernet	Ethernet controller	CC_0200
token_ring	Token ring network controller	CC_0201
fddi	FDDI network controller	CC_0202
atm	ATM network controller	CC_0203
isdn	ISDN controller	CC_0204

Common Name	Description	PCI ID string
picmg	PICMG controller	CC_0206
net_infiniband	Infiniband controller	CC_0207
fabric	Fabric controller	CC_0208
network_nw	Network controller e.g. Wifi	CC_0280
-	-	-
display	Display controller	CC_03
vga	VGA compatible controller	CC_0300
xga	XGA compatible controller	CC_0301
-	-	-
media	Multimedia controller	CC_04
mm_video	Multimedia video controller	CC_0400
mm_audio	Multimedia audio controller	CC_0401
telephony	Computer telephony device	CC_0402
audio	Audio device	CC_0403
-	-	-
bridge	Bridge	CC_06
br_host	Host bridge	CC_0600
br_isa	ISA bridge	CC_0601
br_eisa	EISA bridge	CC_0602
br_microchannel	MicroChannel bridge	CC_0603
br_pci	PCI bridge	CC_0604
br_pcmcia	PCMCIA bridge	CC_0605
br_nubus	NuBus bridge	CC_0606
br_cardbus	CardBus bridge	CC_0607
br_raceway	RACEway bridge	CC_0608
br_semi_pci	Semi-transparent PCI-to-PCI bridge	CC_0609
br_infiniband_to_pci	InfiniBand to PCI host bridge	CC_060a
-	-	-
com	Communication controller	CC_07
com_serial	Serial controller	CC_0700
com_parallel	Parallel controller	CC_0701
com_multiport_ser	Multiport serial controller	CC_0702
com_modem	Modem	CC_0703
com_gpib	GPiB controller	CC_0704
com_smart_card	Smart card controller	CC_0705
-	-	-
serial_bus	Serial bus controller	CC_0c
firewire	FireWire (IEEE 1394)	CC_0c00
access_bus	ACCESS bus	CC_0c01
ssa	SSA	CC_0c02
usb	USB controller	CC_0c03

Common Name	Description	PCI ID string
fibre_channel	Fibre channel	CC_0c04
smbus	SMBus	CC_0c05
bus_infiniband	InfiniBand	CC_0c06
ipmi_smic	IPMI SMIC interface	CC_0c07
sercos	SERCOS interface	CC_0c08
canbus	CAN bus	CC_0c09
-	-	-
wireless	Wireless controller	CC_0d
bluetooth	Bluetooth	CC_0d11
w_8021a	802.1a controller	CC_0d20
w_8021b	802.1b controller	CC_0d21

### Strong Matching of PCI Devices

If more specific matching of PCI devices is required it is possible to concatenate multiple ID strings using `&`. An example where a specific device from a specific vendor at a fixed bus address is matched would use the string `PCI/VEN_vvvv&DEV_dddd&ADR_xxxx:xx:xx.x`.

### Isolation of PCIe devices

PCIe encodes device communication with a network-like protocol with destination headers and packet fragmentation allowing a devices to talk directly to other devices. This potentially works against security boundaries for a system. E.g. two network cards could exchange packets and thereby leak information from one security domain to the other without involvement of the OS.

PCIe introduced an optional capability named PCI Access Control Services (PCI/ACS) to control communication between PCIe devices.

With PCI/ACS it is possible to restrict inter-device communication between PCIe devices.

PCI/ACS is optional and for Intel chipsets, it is usually only implemented on high-end PCI platform controller hubs (PCHs), and is missing on low-end and mobile PCHs. On some Intel-PCHs there exist facilities that allow for similar isolation.

If IO encounters a supported PCH, it will enable those facilities in order to enforce device isolation.

### Command Line Options

The Io Server supports the following optional parameters:

```
[--verbose|v] [--transparent-msi] [--trace <trace_mask>] [--acpi-debug-level <debug_level>] [config_files]
```

- **verbose|v**

By default, error debug messages are enabled. This option increments the verbosity level, and can be applied multiple times to reach the desired debug level. The available debug levels are ordered as: `DBG_ERR` (default, level 1), `DBG_WARN`, `DBG_INFO`, `DBG_DEBUG`, `DBG_DEBUG2` and `DBG_ALL` (level 6).

- **transparent-msi**

Enable MSI on PCI devices which support this feature. This is transparent to clients, as there are no changes in the API used to interact with PCI device via interrupts.

- **acpi-debug-level** <level\_mask>

Set the ACPI debug level. The <level\_mask> is a mask that selects components of interest for debugging. It can be constructed from the ACPI debug constants defined in the linux kernel, see [ACPI Debug Output](#) for details. By default, the ACPI debug level is set to `ACPI_LV_INIT | ACPI_LV_TABLES | ACPI_LV_VERBOSE_INFO`.

- **trace** <trace\_mask>

Enable tracing of events matching `trace_mask`. The only supported trace mask is 1 and this matches ACPI events.

- **config\_files**

Space separated list of Lua configuration files specifying real hardware and virtual buses. See example on [Virtual Bus Description](#).

## 5.5 l4vio\_net\_p2p, a virtual network point-to-point link

The virtual network point-to-point server (p2p) connects two clients with a virtual network connection. It uses virtio as the transport mechanism. Each virtual network p2p endpoint implements the device-side of a virtio network device. Each client can access its endpoint using the driver-side semantics of a virtio network device.

### Building and Configuration

The virtual network p2p server can be built using the [L4Re](#) build system by placing this project into the `pkg` directory.

### Starting the service

The virtual network p2p server can be started with Lua like this:

```
local p2p = L4.default_loader:new_channel();
L4.default_loader:start(
{
  caps = {
    svr = p2p:svr(),
  },
},
"rom/l4vio_net_p2p [<options>]");
```

First an IPC gate (p2p) is created which is used between the virtual network p2p server and a client to create new virtual ports. The server-side is assigned to the mandatory `svr` capability of the virtual network p2p server. See the section below on how to create a new virtual port and connect a client to it.

## Options

The following command line options are supported:

- `-p <num_usec>, --poll <num_usec>`  
Enable polling mode and set the poll interval. IRQ notification is disabled for queues while in polling mode. Must be a positive integer specified in microseconds.
- `-s <num>, --size <num>`  
Set the maximum queue size for the device-side virtio queues. Must be a power of 2 in the range of 1 to 32768 inclusive.

## Connecting a client

Prior to connecting a client to a virtual network p2p server port it has to be created using the following Lua function. It has to be called on the client side of the IPC gate capability whose server side is bound to the virtual network p2p server.

The "key=value" pairs passed to `create()` can be omitted and their order is not important.

```
create(obj_type, ["ds-max=<max>" , "mac=<mac_address>"])
```

- `obj_type`  
The type of object that should be created by the server. The type must be a positive integer. Currently the following objects are supported:
  - 0: Virtual p2p port
- `"ds-max=<max>"`  
Specifies the upper limit of the number of dataspace the client is allowed to register with the server for virtio DMA. Must be in the range of 1 to 80 inclusive. The default value is 2.
- `"mac=<mac_address>"`  
Specify the MAC address of the endpoint where `<mac_address>` is of the form X:XX:Xx:x:xx:XX.

If the `create()` call is successful a new capability which references a virtual network p2p server port is returned. A client uses this capability to talk to the virtual network p2p server using the virtio network protocol.

A couple of examples on how to create ports with different properties are listed below.

```
-- two normal ports with at most 4 data spaces
net0 = p2p:create(0, "ds-max=4")
net1 = p2p:create(0, "ds-max=4")
-- normal port with 4 data spaces and MAC address
net0 = p2p:create(0, "ds-max=4", "mac=11:22:33:44:55:66")
```

## 5.6 Virtio Net Switch, a virtual network switch

The virtual network switch connects multiple clients with a virtual network connection. It uses Virtio as the transport mechanism. Each virtual switch port implements the host-side of a Virtio network device (virtio-net).

## Capabilities

- `dataspace`  
Trusted dataspace  
Multiple capability names can be provided by the `--register-ds` command line parameter.
- `svr`  
Server Capability of application. Endpoint for IPC calls  
Mandatory capability.

## Command Line Options

In the example above the virtual network switch is started in its default configuration with a maximum of 5 virtual ports. To customize the configuration the virtual network switch accepts the following command line options:

- `-D <component=level>, --debug <component=level>`  
Configure individual debug levels per component. Verbosity increases from `quiet` up to `trace`.  
Can be used multiple times.  
Possible values for component are | `core, virtio, port, request, queue, packet`.  
Possible values for level are | `quiet, warn, info, debug, trace`.
- `-m, --mac`  
Ignored. Provided for compatibility with older switch versions.  
Flag. True if provided.
- `-M`  
Do not assign a random MAC address to ports by default. It is always possible to set an explicit MAC address by passing the `mac=` to the factory call, regardless of this option. If `-M` is passed and no `mac=` address was given, it is the responsibility of the Virtio driver to choose an appropriate address.  
Flag. True if provided.
- `-p <num>, --ports <num>`  
Set the maximum number of virtual ports.  
Numerical value.  
Default: 5
- `-q, --quiet`  
Silence all output except for error messages.  
Flag. True if provided.
- `-s <num>, --size <num>`  
Set the maximum queue size for the device-side Virtio queues.  
Numerical value.  
– Must be a power of 2 in the range of 1 to 32768 inclusive.  
Default: 256
- `-v, --verbose`  
Increase the global verbosity level. Individual levels per component can be set using the `-D` option.  
Can be used up to 3 times.  
Flag. True if provided.

- `-d <cap_name>,--register-ds <cap_name>`

Register a trusted dataspace capability. If this option gets used, it is not possible to communicate with the server via dataspace other than the registered ones. Can be used multiple times for multiple dataspace.

The option's parameter is the name of a dataspace capability.

Can be used multiple times.

Name of a provided capability that adheres to the dataspace protocol.

The virtual network switch can be setup to feature exactly one monitor port. All traffic passing through the switch is mirrored to the monitor port. The monitor port is read-only, and has no TX capability. An optional packet filter can be configured and implemented to filter data sent to the monitor port.

## Configuration

Certain features of the virtual network switch are configurable at compile-time. Configuration is done through the build-time configuration of the [L4Re](#) build tree.

## Starting the service

The virtual network switch can be started in Ned like this:

```
local switch = L4.default_loader:new_channel();
L4.default_loader:start(
{
  caps = {
    svr = switch:svr(),
  },
},
"rom/l4vio_switch");
```

First a communication channel (`switch`) is created which is used to create virtual network ports. It is connected to the switch component via its mandatory `svr` capability. See the section below on how to create a new virtual port and connect a client to it.

## Hardware devices

To plug hardware devices into the switch, provide a Vbus capability with the name `vbus` when starting the switch. To use this feature, you have to enable the `VNS_IXL` config option.

## Virtual switch port

First, a virtual network port has to be created using the following Ned-Lua function. It has to be called on the communication channel called `switch`, which has been created earlier.

Call: `create(0 [, "ds-max=<max>", "name=<name>", "type=<port type>", "vlan=(access=<vlan id>|trunk=<vlan id>[,<vlan id>]*)", "mac=<addr>"])`

- `"ds-max=<max>"`

Specifies the upper limit of the number of dataspace the client is allowed to register with the virtual network switch for Virtio DMA.

Numerical value.

- In the range of 1 to 80 inclusive
- "name=<name>"
 

Sets the name of port in debug messages to <name>. A name may consist of at most 19 characters, all other characters are dropped. If there is enough space left, the name will get a postfix of "[<port number>]", e.g. "name=foo" -> foo[1].

String value.
- "type=<port type>"
 

Optionally specify the port type.

Possible values for <port type> are

  - monitor: Monitor Port
  - normal: Normal Port

Default: normal
- "vlan=(access=<vlan id>|trunk=<vlan id>[,<vlan id>]\*) "
 

Configure the port to participate in an IEEE 802.1Q compatible VLAN. Fundamentally there are two types of ports: access ports and trunk ports.

The Value for this parameter can be one of the following:

  - "access=<vlan id>"
 

Configures the port as access port for VLAN <vlan id> where the id must be a decimal number greater than 0 and less than 4095 in accordance to the standard. Packets on an access port belong to the configured VLAN and are only forwarded to ports that belong to the same VLAN or trunk ports that participate in the particular VLAN. The packets on this port will not have a VLAN tag attached to them so that a guest connected to this port does not see that the port is part of a VLAN.

An optional monitor port will see packets from an access port as VLAN tagged packets with the <vlan id> given for the port.

Numerical value.

    - \* In the range of 0 to 4095 exclusive
  - "trunk=<vlan id>[, <vlan id>]\*"
 

Configures the port as trunk port. It participates either in all VLANs, if specified by the keyword 'all', or in the list of VLANs given as comma separated list. There must be no whitespace in the list. Each id must be a decimal number greater than 0 and less than 4095 in accordance to the standard. Outgoing packets on this port will be tagged with an IEEE 802.1Q compatible tag. Incoming packets must be tagged with a VLAN tag from the given list. Packets that have no tag or a tag not in the vlan id list are dropped silently. They are not forwarded to the monitor port either. Currently there is no support for IEEE 802.1p. The PCP and DEI sub-fields in the TCI field will be set to zero on outgoing packets and are ignored for incoming packets.

Numerical value.
- "mac=<addr>"
 

Explicitly sets the MAC address of the port. It will be checked that no other port on the switch has the same address. It is the responsibility of the user to ensure the validity of the address and its global uniqueness, though.

String value.

  - In form xx:xx:xx:xx:xx:xx

If the `create()` call is successful a new capability which references a virtual switch port is returned. A client uses this capability to talk to the virtual network switch using the Virtio network protocol.



## Examples

Here are couple of examples on how to create ports with different properties:

```
-- normal port with at most 4 data spaces
net0 = switch:create(0, "ds-max=4")
-- like the previous but with name foo
net0 = switch:create(0, "ds-max=4", "name=foo")
-- like the previous but the port is a monitor port
net0 = switch:create(0, "ds-max=4", "name=foo", "type=monitor")
-- normal port with 4 data spaces as access port to VLAN 1
net0 = switch:create(0, "ds-max=4", "name=v11", "vlan=access=1")
-- normal port with 4 data spaces as trunk port participating in VLAN 1 & 2
net0 = switch:create(0, "ds-max=4", "name=v11", "vlan=trunk=1,2")
```

## 5.7 Uvmm, the virtual machine monitor

Uvmm provides a virtual machine for running an unmodified guest in non- privileged mode.

### Capabilities

- `ram`

The memory for the guest provided via a simple dataspace.

- `pfc`

Uvmm can be instructed to inform a PM manager of PM events through the [L4::Platform\\_control](#) interface. To that end, uvmm may be equipped with a `pfc` cap. On suspend, uvmm will call [l4\\_platform\\_ctl\\_system\\_suspend\(\)](#).

The `pfc` cap can also be implemented by IO. In that case the guest can start a machine suspend/shutdown/reboot.

### Command Line Options

- `--mon <option>`

Control the handling of guest reads/writes to non-existing memory.

Possible values for `<option>` are

- `true`: The monitor interface is enabled but no server implementing the client side of the monitor interface is started. The monitor interface can still be utilized via cons but no readline functionality will be available.
- `some_binary`: If a string is passed as the value of `mon`, the monitor interface is enabled and the string is interpreted as the name of a server binary which implements the client side of the monitor interface. This server is automatically started and has access to a `vcon` capability named `mon` at startup through which it can make use of the monitor interface. Unless you have written your own server you should specify `'uvmm_cli'` which is a server implementing a simple readline interface.
- `false`: The monitor interface is disabled at runtime.

Default: `true`

- `-c <guest command line>,--cmdline <guest command line>`

Command line that is passed to the guest on boot.

String value.

- `-k <kernel image name>,--kernel <kernel image name>`  
The name of the guest-kernel image file present in the ROM namespace.  
File path in ROM namespace.
- `-d <DTB overlay>,--dtb <DTB overlay>`  
The name of the device tree file present in the ROM namespace. The device tree will be placed in the upmost region of guest memory. Optionally, a user may use an additional parameter in the form of "`<DTB overlay>:limit=0xffffffff`" to set an upper limit for the device tree location.  
File path in ROM namespace.
- `-r <RAM disk name>,--ramdisk <RAM disk name>`  
The name of the RAM disk file present in the ROM namespace  
File path in ROM namespace.
- `-b <Base address of the guest RAM>,--rambase <Base address of the guest RAM>`  
Physical start address for the guest RAM. This value is platform specific.  
Numerical value.
- `-D <component=level>,--debug <component=level>`  
Control the verbosity level of the uvmm.  
Using the `component` prefix, the verbosity level of each uvmm component is configurable.  
For example, the following command line sets the verbosity of all uvmm components to `info` except for IRQ handling, which is set to `trace`:  

```
uvmm -D info -D irq=trace
```

  
Can be used multiple times.  
Possible values for component are | `core,cpu,mmio,irq,dev,pm,vbus_event`.  
Possible values for level are | `quiet,warn,info,trace`.  
Options `-q, --quiet,-v, --verbose` and `-D, --debug` cancel each other out.
- `-f,--fault-mode`  
Control the handling of guest reads/writes to non-existing memory.  
Possible values are
  - `ignore`: Invalid writes are ignored. Invalid reads either return 0 or are skipped. The guest may experience undefined behaviour.
  - `halt`: Halt the VM on the first invalid memory access.
  - `inject`: Try to forward the invalid access to the guest. This is not supported on all architectures. Falls back to `halt` if the error could not be forwarded to the guest.
 Default: `ignore`
- `-q,--quiet`  
Silence all uvmm output.  
Flag. True if provided.  
Options `-q, --quiet,-v, --verbose` and `-D, --debug` cancel each other out.
- `-v,--verbose`  
Increase the verbosity of the uvmm. Repeating the option increases the verbosity by another level.  
Can be used multiple times.  
Flag. True if provided.  
Options `-q, --quiet,-v, --verbose` and `-D, --debug` cancel each other out.

- `-W, --wakeup-on-system-resume`

When set, the uvmm resumes when the host system resumes after a suspend call.

Flag. True if provided.

- `-i, --ram-identity-mapping`

When set, the option forces the guest RAM to be mapped to its corresponding host-physical addresses.

Flag. True if provided.

## Setting up guest memory

In the most simple setup, memory for the guest can be provided via a simple dataspace. In your ned script, create a new dataspace of the required size and hand it into uvmm as the `ram` capability:

```
local ramds = L4.Env.user_factory:create(L4.Proto.Dataspace, 0x4000000)

L4.default_loader::startv({caps = {ram = ramds:m("rw")}}, "rom/uvmm")
```

The memory will be mapped to the most appropriate place and a memory node added to the device tree, so that the guest can find the memory.

For a more complex setup, the memory can be configured via the device tree. uvmm scans for memory nodes and tries to set up the memory from them. A memory device node should look like this:

```
memory@0 {
    device_type = "memory";
    reg = <0x00000000 0x00100000
          0x00200000 0xffffffff>;
    l4vmm,dscap = "memcap";
    dma-ranges = <>;
};
```

The `device_type` property is mandatory and needs to be set to `memory`.

`l4vmm,dscap` contains the name of the capability containing the dataspace to be used for the RAM. `reg` describe the memory regions to use for the memory. The regions will be filled up to the size of the supplied dataspace. If they are larger, then the remaining area will be cut.

If the optional `dma-ranges` property is given, the host-physical address ranges for the memory regions will be added here. Note that the property is not cleared first, so it should be left empty.

For more details see [RAM configuration](#).

## Memory layout

uvmm populates the RAM with the following data:

- kernel binary
- (optional) ramdisk
- (optional) device tree

The kernel binary is put at the predefined address. For ELF binaries, this is an absolute physical address. If the binary supports relative addressing, the binary is put to the requested offset relative to beginning of the first 'memory' region defined in the device tree.

The ramdisk and device tree are placed as far as possible to the end of the regions defined in the first 'memory' node.

If there is a part of RAM that must remain empty, then define an extra memory node for it in the device tree. uvmm only writes to memory in the first memory node it finds.

Warning: uvmm does not touch any unpopulated memory. In particular, it does not ensure that the memory is cleared. It is the responsibility of the provider of the RAM dataspace to make sure that no data leakage can happen. Normally this is not an issue because dataspaces are guaranteed to be cleaned when they are newly created but users should be careful when reusing memory or dataspaces, for example, when restarting the uvmm.

## Forwarding hardware resources to the guest

Hardware resources must be specified in two places: the device tree contains the description of all hardware devices the guest could see and the Vbus describes which resources are actually available to the uvmm.

The vbus allows the uvmm access to hardware resources in the same way as any other [L4](#) application. uvmm expects a capability named 'vbus' where it can access its hardware resources. It is possible to leave out the capability for purely virtual guests (Note that this is not actually practical on some architectures. On ARM, for example, the guest needs hardware access to the interrupt controller. Without a 'vbus' capability, interrupts will not work.) For information on how to configure a vbus, see the [IO documentation](#).

The device tree needs to contain the hardware description the guest should see. For hardware devices this usually means to use a device tree that would also be used when running the guest directly on hardware.

On startup, uvmm scans the device tree for any devices that require memory or interrupt resources and compares the required resources with the ones available from its vbus. When all resources are available, it sets up the appropriate forwarding, so that the guest now has direct access to the hardware. If the resources are not available, the device will be marked as 'disabled'. This mechanism allows to work with a standard device tree for all guests in the system while handling the actual resource allocation in a flexible manner via the vbus configuration.

The default mechanism assigns all resources 1:1, i.e. with the same memory address and interrupt number as on hardware. It is also possible to map a hardware device to a different location. In this case, the assignment between vbus device and device tree device must be known in advance and marked in the device tree using the `l4vmm, vbus-dev` property.

The following device will for example be bound with the vbus device with the HID 'l4-test,dev':

```
test@e0000000 {
    compatible = "memdev,bar";
    reg = <0 0xe0000000 0 0x50000>,
        <0 0xe1000000 0 0x50000>;
    l4vmm,vbus-dev = "l4-test,dev";
    interrupts-extended = <&gic 0 139 4>;
};
```

Resources are then matched by name. Memory resources in the vbus must be named `reg0` to `reg9` to match against the address ranges in the device tree `reg` property. Interrupts must be called `irq0` to `irq9` and will be matched against `interrupts` or `interrupts-extended` entries in the device tree. The vbus must expose resources for all resources defined in the device tree entry or the initialisation will fail.

An appropriate IO entry for the above device would thus be:

```
MEM = Io.Hw.Device(function()
    Property.hid = "l4-test,dev"
    Resource.reg0 = Io.Res.mmio(0x41000000, 0x4104ffff)
    Resource.reg1 = Io.Res.mmio(0x42000000, 0x4204ffff)
    Resource.irq0 = Io.Res.irq(134);
end)
```

Please note that HIDs on the vbus are not necessarily unique. If multiple devices with the HID given in `l4vmm, vbus-dev` are available on the vbus, then one device is chosen at random.

If no vbus device with the given HID is available, the device is disabled.

## How to enable guest suspend/resume

### Note

Currently only supported on ARM. It should work fine with Linux version 4.4 or newer.

Uvmm (partially) implements the power state coordination interface (PSCI), which is the standard ARM power management interface. To make use of this interface, you have to announce its availability to the guest operating system via the device tree like so:

```
psci {
    compatible = "arm,psci-0.2";
    method = "hvc";
};
```

The Linux guest must be configured with at least these options:

```
CONFIG_SUSPEND=y
CONFIG_ARM_PSCI=y
```

## How to communicate power management (PM) events

Uvmm can be instructed to inform a PM manager of PM events through the [L4::Platform\\_control](#) interface. To that end, uvmm may be equipped with a `pf_c` cap. On suspend, uvmm will call `l4_platform_ctl_system_suspend()`.

The `pf_c` cap can also be implemented by IO. In that case the guest can start a machine suspend/shutdown/reboot.

## Ram block device support

The example ramdisk works by loading a file system into RAM, which needs RAM block device support to work. In the Linux kernel configuration add: `CONFIG_BLK_DEV_RAM=y`

## Framebuffer support for uvmm/amd64 guests

Uvmm can be instructed to pass along a framebuffer to the Linux guest. To enable this three things need to be done:

1. Configure Linux to support a simple framebuffer by enabling

```
CONFIG_FB_SIMPLE=y
CONFIG_X86_SYSFB=y
```

2. Configure a simple framebuffer device in the device tree (currently only read by uvmm, linearer framebuffer at [0xf0000000 - 0xf1000000])

```
simplefb {
    compatible = "simple-framebuffer";
    reg = <0x0 0xf0000000 0x0 0x1000000>;
    l4vmm,fbcap = "fb";
};
```

3. Start a framebuffer instance and connect it to uvmm e.g.

```
-- Start fb-drv (but only if we need to)
local fbdrv_fb = L4.Env.vesa;
if (not fbdrv_fb) then
    fbdrv_fb = l:new_channel();
    l:start({
        caps = {
            vbus = io_busses.fbdrv,
            fb    = fbdrv_fb:svr(),
        },
        log = { "fbdrv", "r" },
    },
    "rom/fb-drv");
end
vmm.start_vm{
    ext_caps = { fb = fbdrv_fb },
    -- ...
```

## Requirements on the Fiasco.OC configuration on amd64

The kernel configuration must feature `CONFIG_SYNC_TSC=y` in order for the emulated timers to reach a sufficiently high resolution.

## Recommended Linux configuration options for uvmm/amd64 guests

The following options are recommended in addition to the amd64 defaults provided by a `make defconfig`:

Virtio support is required to access virtual devices provided by uvmm:

```
CONFIG_VIRTIO=y
CONFIG_VIRTIO_PCI=y
CONFIG_VIRTIO_BLK=y
CONFIG_BLK_MQ_VIRTIO=y
CONFIG_VIRTIO_CONSOLE=y
CONFIG_VIRTIO_INPUT=y
CONFIG_VIRTIO_NET=y
```

It is highly recommended to use the X2APIC, which needs virtualization awareness to work under uvmm:

```
CONFIG_X86_X2APIC=y
CONFIG_PARAVIRT=y
CONFIG_PARAVIRT_SPINLOCKS=y
```

## KVM clock for uvmm/amd64 guests

When executing [L4Re](#) + uvmm on QEMU, the PIT as clock source is not reliable. The paravirtualized KVM clock provides the guest with a stable clock source.

A KVM clock device is available to the guest, if the device tree contains the corresponding entry:

```
kvm_clock {
    compatible = "kvm-clock";
    reg = <0x0 0x0 0x0 0x0>;
};
```

To make use of this clock, the Linux guest must be built with the following configuration options:

```
CONFIG_HYPERVISOR_GUEST=y
CONFIG_KVM_GUEST=y
CONFIG_PTP_1588_CLOCK_KVM is not set
```

### Note

KVM calls besides the KVM clock are unhandled and lead to failure in the uvmm, e.g. `vmcall 0x9` for the `PTP_1588_CLOCK_KVM`.

This is considered a development feature. The KVM clock is not required when running on physical hardware as TSC calibration via the PIT works as expected.

## Development notes for amd64

When you are developing on Linux using QEMU please note that nested virtualization support is necessary on your host system to run uvmm guests. Your host Linux version should be 4.12 or greater, **excluding 4.20**.

Check if your KVM module has nested virtualization enabled via:

```
cat /sys/module/kvm_intel/parameters/nested
Y
```

In case it shows N instead of Y enable nested virtualization support via:

```
modprobe kvm_intel nested=1
```

On AMD platforms the module name is `kvm_amd`.

## QEMU network setup for a uvmm guest on amd64

```
qemu-system-x86_64 -M q35 -cpu host -enable-kvm -device intel-iommu
                  -device e1000e,netdev=net0 -netdev bridge,id=net0,br=virbr0
```

where 'virbr0' is the name of the host's bridge device. The line 'allow virbr0' needs to be present in `/etc/qemu/bridge.conf`. The bridge can either be created via the network manager or via the command line:

```
brctl addbr virbr0
ip addr add 192.168.124.1/24 dev virbr0
ip link set up dev virbr0
```

In the guest linux with eth0 as network device:

```
ip a a 192.168.124.5/24 dev eth0
ip li se up dev eth0
```

Now the host and guest can ping each other using their respective IPs.

Of course, uvmm needs to be connected to io and io needs a vbus configuration for the uvmm client like this:

```
Io.add_vbusses
{
    vm_pci = Io.Vi.System_bus(function ()
        Property.num_msis = 6
        PCI = Io.Vi.PCI_bus(function ()
            pci_net = wrap(Io.system_bus():match("PCI/CC_0200"))
        end)
    end)
}
```

## QEMU emulated VirtIO devices and IO-MMU on amd64

QEMU does not route VirtIO devices through the IO-MMU per default. To use QEMU emulated VirtIO devices add the `disable-legacy=on,disable-modern=off,iommu_platform=on` flags to the option list of the device. The e1000e card in the network example above can be replaced with an virtio-net-pci card like this:

```
-device virtio-net-pci,disable-legacy=on,disable-modern=off,
      iommu_platform=on,netdev=net0
```

For more information on VirtIO devices and their options see <https://wiki.qemu.org/Features/VT-d>.

## Using the uvmm monitor interface

Uvmm implements an interface with which parts of the guest's state can be queried and manipulated at runtime. This monitor interface needs to be enabled during compilation as well as during startup of uvmm. This is described in detail below.

### Compiling uvmm with monitor interface support

To compile uvmm with monitor interface support pass the `CONFIG_MONITOR=y`, option during the `make` step (or set in in the `Makefile.config`). This option is available on all architectures but note that the set of available monitor interface features may vary significantly between them. Also note that the monitor interface will always be disabled in release mode, i.e. if `CONFIG_RELEASE_MODE=y`.

## Enabling the monitor interface at runtime

When starting a uvmm instance from inside a `ned` script using the `uvmm.start_vm` function, the `mon` argument controls whether the monitor interface is enabled at runtime. There are three cases to distinguish:

- `mon=true` (default): The monitor interface is enabled but no server implementing the client side of the monitor interface is started. The monitor interface can still be utilized via `cons` but no readline functionality will be available.
- `mon='some_binary'`: If a string is passed as the value of `mon`, the monitor interface is enabled and the string is interpreted as the name of a server binary which implements the client side of the monitor interface. This server is automatically started and has access to a `vcon` capability named `mon` at startup through which it can make use of the monitor interface. Unless you have written your own server you should specify `'uvmm_cli'` which is a server implementing a simple readline interface.
- `mon=false`: The monitor interface is disabled at runtime.

## Using the monitor interface

If the monitor interface was enabled you can connect to it via `cons` under the name `mon<n>` where `<n>` is a unique integer for every uvmm instance that is started with the monitor interface enabled (numbered starting from one in order of corresponding `uvmm.start_vm` calls). If `mon='uvmm_cli'` was specified, readline functionality such as command completion and history will be available. Enter a command followed by enter to run that command. To obtain a list of all available commands issue the `help` command, to obtain usage information for a specific command `foo` issue `help foo`.

### Note

Some commands will modify the guests state. Since it should be obvious to which ones this applies this is usually not specifically highlighted. Exercise reasonable caution.

## Using the guest debugger

The guest debugger provides monitoring functionality akin to a very bare-bone GDB interface, e.g. guest RAM and page table dumping, breakpointing and single stepping. Additional functionality might be added in the future.

### Note

The guest debugger is currently still under development. The guest debugger may also not be available on all architectures. To check whether the guest debugger is available check if `help dbg` returns usage information.

If the guest debugger is available, you have to manually load it at runtime using the monitor interface. This saves resources if the guest debugger is not used. To enable the guest debugger, issue the `dbg on` monitor command. Once enabled, the guest debugger can not be disabled again.

To list available guest debugger subcommands, issue `dbg help` after `dbg on`.

### Note

When using SMP, most guest debugger subcommands require you to explicitly specify a guest vcpu using an index starting from zero.



### 5.7.1 RAM configuration

#### RAM configuration for uvmm

##### Without a memory node in the device tree

- setup default RAM for guest VM.
- RAM starts either
  - at base-address which defaults to 0x0 or the base address value set via the -b cmdline option or
  - in case of identity mapping at the host-physical address of the dataspace allocated for the RAM

##### With a memory node in the device tree

The memory node needs at least the properties `device_type` and `l4vmm,dscap`:

```
memory@0 {  
    device_type = "memory";  
    l4vmm,dscap = "ram";  
}
```

Where the given `l4vmm,dscap` name is accessible in the capability namespace of the uvmm. If the capability is invalid, the memory node is disabled.

If memory nodes are given, but none provides valid RAM the configuration is invalid and uvmm refuses to boot.

Additional properties of the memory node are `reg` and `dma-ranges`.

The `reg` property describes the location in the guest's address space that should be backed by RAM.

The `dma-ranges` property describes the offset between guest-physical and host-physical addresses. The guest can evaluate this non-standard property to derive the correct DMA addresses to program into passed-through devices. Usage of this property **requires** modification of guest code.

##### Without `reg` and `dma-ranges` properties

The `reg` property is optional only in case the uvmm maps the guest's RAM into the VM under the host-physical addresses of the backing memory (`l4vmm,dscap`).

This case can be forced via the cmdline parameter `-i` and is the default for platforms without IOMMU, but with DMA capable devices on the configured vBus.

##### Without a `reg` property, but with a `dma-ranges` property

If the `-i` cmdline parameter is given, identity mapping is forced and the behavior is the same as in the case above. Additionally, the `dma-ranges` property is written

In case no `-i` cmdline parameter is given, the configuration is invalid and uvmm refuses to boot.

### With a reg property

uvmm parses the reg property of the memory node and maps the memory into the VM to the given range(s).

If the -i cmdline parameter is set, the reg property is ignored and the memory is mapped into the VM under the corresponding host-physical addresses of the backing memory (l4vmm,dscap)

### With a reg and dma-ranges property

uvmm parses the reg property of the memory node and maps the memory into the VM to the given range(s).

The dma-ranges property is filled with the corresponding host-physical addresses of the backing memory (l4vmm,dscap).

## 5.8 RTC driver

The RTC driver can drive various real-time clocks and provides an interface for other components, e.g., uvmm, to read and write the time.

It needs access to the hardware, depending on the clock, either via a vbus or via an I2C device.

### Command Line Options

There are no command line options.

### Environment

Several capabilities can be used to interact with the environment:

- 'rtc' (server)  
The capability with which clients talk to the service. Note that writing to the RTC is only possible when having the rtc capability with write rights, read-only clients must have the capability with read rights only.
- 'vbus'  
The vbus used to access hardware for port-based clock on X86 and pl031 on arm.  
The vbus is also needed for receiving the inhibitor signal from IO.

## 5.9 NVMe server

The NVMe server is a driver for PCI Express NVMe controllers.

The NVMe server is capable of exposing entire disks (i.e. NVMe namespaces) (by serial number and namespace identifier) or individual partitions (by their partition UUID) of a hard drive to clients via the Virtio block interface.

The server consists of two parts. The first one is the hardware driver itself that takes care of the communication with the underlying hardware and interrupt handling. The second part implements a virtual block device and is responsible to communicate with clients. The virtual block device translates commands it receives into NVMe requests and issues them to the hardware driver.

The NVMe server allows both statically and dynamically configured clients. A static configuration is given priority over dynamically connecting clients and configured while the service starts. Dynamic clients can connect and disconnect during runtime of the NVMe server.

## Capabilities

- `vbus`  
Virtual bus capability  
Mandatory capability.
- `dataspace`  
Trusted dataspace  
Multiple capability names can be provided by the `--register-ds` command line parameter.
- `client`  
Static client  
Multiple capability names can be provided by the `--client` command line parameter.
- `svr`  
Server Capability of application. Endpoint for IPC calls  
Mandatory capability.

## Command Line Options

In the example above the NVMe server is started in its default configuration. To customize the configuration of the NVMe-server it accepts the following command line options:

- `-v, --verbose`  
Enable verbose mode. You can repeat this option to increase verbosity up to trace level.  
Can be used up to 3 times.  
Flag. True if provided.
- `-q, --quiet`  
This option enables the quiet mode. All output is silenced.  
Flag. True if provided.
- `--client <cap_name>`  
Connect a static client.  
Can be used multiple times.  
Name of a provided capability with server rights that adheres to the ipc protocol.  
This parameter opens a scope for the following subparameters:
  - `--device <UUID | <SN>:n<NAMESPACE_ID>>`  
This option denotes the partition UUID or serial number of the preceding `client` option followed by a colon, letter 'n' and the identifier of the requested NVMe namespace.  
String value.
  - `--ds-max <max>`  
This option sets the upper limit of the number of dataspace the client is able to register with the NVMe server for virtio DMA.  
Numerical value.
  - `--readonly`  
This option sets the access to disks or partitions to read only for the preceding `client` option.  
Flag. True if provided.

- `--nosgl`  
This option disables support for SGLs.  
Flag. True if provided.
- `--nomsi`  
This option disables support for MSI interrupts.  
Flag. True if provided.
- `--nomsix`  
This option disables support for MSI-X interrupts.  
Flag. True if provided.
- `-d <cap_name>,--register-ds <cap_name>`  
This option registers a trusted dataspace capability. If this option gets used, it is not possible to communicate to the driver via dataspace other than the registered ones. Can be used multiple times for multiple dataspace.  
The option's parameter is the name of a dataspace capability.  
Can be used multiple times.  
Name of a provided capability that adheres to the dataspace protocol.

## Building and Configuration

The NVMe server can be built using the [L4Re](#) build system. Just place this project into your `pkg` directory. The resulting binary is called `nvme-drv`

## Starting the service

The NVMe server can be started with Lua like this:

```
local nvme_bus = L4.default_loader:new_channel();
L4.default_loader:start({
  caps = {
    vbus = vbus_nvme,
    svr = nvme_bus:svr(),
  },
},
"rom/nvme-drv");
```

First an IPC gate (`nvme_bus`) is created which is used between the NVMe server and a client to request access to a particular disk or partition. The server- side is assigned to the mandatory `svr` capability of the NVMe server. See the section below on how to configure access to a disk or partition.

The NVMe server needs access to a virtual bus capability (`vbus`). On the virtual bus the NVMe server searches for NVMe compliant storage controllers. Please see `io`'s documentation about how to setup a virtual bus.

## Virtio block host

Prior to connecting a client to a virtual block session it has to be created using the following Lua function. It has to be called on the client side of the IPC gate capability whose server side is bound to the NVMe server.

Call: `create(0, "device=<UUID | <SN>:n<NAMESPACE_ID>>" [, "ds-max=<max>", "read-only"])`

- "device=<UUID | <SN>:n<NAMESPACE\_ID>>"

This string denotes either a partition UUID, or a disk serial number the client wants to be exported via the Virtio block interface followed by a colon, letter 'n' and the identifier of the requested NVMe namespace.

String value.

- "ds-max=<max>"

Specifies the upper limit of the number of dataspace the client is allowed to register with the NVMe server for virtio DMA.

Numerical value.

- In the range of 1 to 256 inclusive

Default: 2

- "read-only"

This string sets the access to disks or partitions to read only for the client.

Flag. True if provided.

If the `create()` call is successful a new capability which references an NVMe virtio device is returned. A client uses this capability to communicate with the NVMe server using the Virtio block protocol.

## Examples

A couple of examples on how to request different disks or partitions are listed below.

- Request a partition with the given UUID

```
vda1 = nvme_bus:create(0, "ds-max=5", "device=88E59675-4DC8-469A-98E4-B7B021DC7FBE")
```

- Request complete namespace with the given serial number

```
vda = nvme_bus:create(0, "ds-max=4", "device=1234:n1")
```

- A more elaborate example with a static client. The client uses the client side of the `nvme_c11` capability to communicate with the NVMe server.

```
local nvme_c11 = L4.default_loader:new_channel();
local nvme_bus = L4.default_loader:new_channel();
L4.default_loader:start({
  caps = {
    vbus = vbus_nvme,
    svr = nvme_bus:svr(),
    c11 = nvme_c11:svr(),
  },
},
"rom/nvme-drv --client c11 --device 88E59675-4DC8-469A-98E4-B7B021DC7FBE --ds-max 5");
```

## 5.10 Mag, the GUI Multiplexer

Mag is the default multiplexer for graphics hardware. Mag is a Nitpicker derivate that allows secure multiplexing of the graphics and input hardware among multiple applications and multiple complete windowing environments. It is not, and does not attempt to be, a fully-fledged window manager.

## Capabilities

### Command Line Options

The following command line options are supported:

```
<$FILE.lua | libmag-$LIBNAME.so>
```

As Mag's only command line option it supports loading additional plugins via the application's command line.

Can be used multiple times.

File path in ROM namespace.

- Plugins must be either a Lua file or a shared library. Shared libraries must be named libmag-\$LIBNAME.so.

### Session Factory

As a simple nitpicker clone Mag supports the so-called Xray mode. This mode displays all session labels and draws a colored frame around them. The session that currently has the input focus is highlighted. The Xray mode is activated via the special keys Scroll or NEXTSONG.

```
Call: create(0 [, "label|l=<LABEL>", "col=(string|hex) "])
```

- "label|l=<LABEL>"

Set the session's text label to LABEL.

String value.

- The label is restricted to a length of 256 characters.

- "col=(string|hex) "

Set the session's color which is used in Xray mode to tint the session's screen area and the border drawn around it. The argument can be either one of the following letters or a hexadecimal representation of the RGB values.

The Value for this parameter can be one of the following:

- string

Define the color by the given character

Possible values are

- \* r, R: Red color
- \* g, G: Green color
- \* b, B: Blue color
- \* w, W: White color
- \* y, Y: Yellow color
- \* v, V: Magenta color

- hex

Define the color by its RGB values in hexadecimal representation.

Hexadecimal number without '0x' prefix.

## Mag Client Session

A client with a mag client session gets access to the whole screen. The client has to allocate and manage its own buffers and has to position them on the screen on its own.

Call: `create(L4.Proto.Goos [, "default-background|dfl-bg"])`

- `"default-background|dfl-bg"`  
Marks this session as the default background.  
Flag. True if provided.

## Mag Client Framebuffer Session

For a client framebuffer session mag allocates a view of the requested size and displays it at the requested coordinates on the screen.

Call: `create(L4.Proto.Goos [, "geometry|g=<GEOMETRY>", "focus", "shaded", "fixed", "barheight=<X>"])`

- `"geometry|g=<GEOMETRY>"`  
Set the session's geometry and position on the screen. GEOMETRY is provided in an X11-style format: WIDTHxHEIGHT+X\_OFFSET+Y\_OFFSET.  
String value.
- `"focus"`  
Set the focus to this session.  
Flag. True if provided.
- `"shaded"`  
The window is collapsed and only the title bar is visible. The window can be expanded by clicking into the title bar with the middle mouse button. Collapsing and expanding works also independently of this option.  
Flag. True if provided.
- `"fixed"`  
The window cannot be moved on the screen.  
Flag. True if provided.
- `"barheight=<X>"`  
Set the height of the title bar in pixels.  
Numerical value.

## Examples

### Creating a session

```
-- set label to "Linux" and use a light blue color
fb = mag_client:create(L4.Proto.Goos, "l=Linux", "col=98d9ff");
```

### Requesting a framebuffer via a client framebuffer session

```
-- create a window of 640x480 pixels at position (100,100) on the screen.
fb = mag_fb:create(L4.Proto.Goos, "g=640x480+100+100");
```

## 5.11 eMMC driver

The eMMC driver is a driver for PCI Express eMMC controllers.

### Starting the service

The eMMC driver can be started with Lua like this:

```
local emmc_bus = L4.default_loader:new_channel();
L4.default_loader:start({
  caps = {
    vbus = vbus_emmc,
    svr = emmc_bus:svr(),
  },
}, "rom/emmc-drv");
```

First, an IPC gate (`emmc_bus`) is created which is used between the eMMC driver and a client to request access to a particular disk or partition. The server side is assigned to the mandatory `svr` capability of the eMMC driver. See the section below on how to configure access to a disk or partition.

The eMMC driver needs access to a virtual bus capability (`vbus`). On the virtual bus the eMMC driver searches for eMMC compliant storage controllers. Please see io's documentation about how to setup a virtual bus.

### Supported devices

The eMMC driver supports SDHCI and SDHI controllers, in particular

- SDHI interfaces found on RCar3 r8a7795 boards
- SDHCI interfaces found on RPI4
- uSDHCI interfaces found on i.MX8 boards
- uSDHCI interfaces found on the S32G SoC
- the QEMU SD card emulation (SDHCI, see `doc/pcie-ecam.io`),
- the QEMU eMMC emulation (provided by extending the QEMU SD card emulation by `doc/qemu-patch.↔diff`).



## Capabilities

- `vbus`  
Required for finding the device which should be driven by this driver.  
Mandatory capability.
- `client`  
Static client  
Multiple capability names can be provided by the `--client` command line parameter.
- `bbds`  
**Only used by the SDHCI driver.** Certain SDHCI devices cannot handle DMA requests with DMA buffers beyond 4GiB. The provided dataspace is used as bounce buffer if the driver detects that a certain request needs it. **Note:** The bounce buffer needs to be able to hold the memory for an entire read/write request. That means that the buffer is divided into the number of maximum segments (see `--max-seg` parameter).
- `sdhci_adma_buf`  
**Only used by the SDHCI driver.** Page (4096 bytes) for storing DMA descriptors for the SDHCI driver. If this capability is not provided, the driver will allocate an arbitrary page.
- `bcm2835_mbox_mem`  
**Only used by the SDHCI driver when attaching to an bcm2711-compatible device.** Page (4096 bytes) for storing bcm2835 mbox messages. The firmware mbox is used to perform voltage switching for certain SD card configurations. If this capability is not provided, the driver will allocate an arbitrary page.
- `svr`  
Server Capability of application. Endpoint for IPC calls  
Mandatory capability.

## Command Line Options

In the example above the eMMC driver is started in its default configuration. To customize the configuration of the eMMC driver it accepts the following command line options:

- `-v, --verbose`  
Enable verbose mode. You can repeat this option to increase verbosity up to trace level.  
Can be used up to 3 times.  
Flag. True if provided.
- `-q, --quiet`  
This option enables the quiet mode. All output is silenced.  
Flag. True if provided.
- `--disable-mode <mode>`  
This option allows to disable certain eMMC/SD card modes from autodetection. Modes with the `hs-` prefix are determined for eMMC devices, others are for SD card devices.  
Can be used multiple times.  
Possible values for `<mode>` are `hs26`, `hs52`, `hs52_ddr`, `hs200`, `hs400`, `sdr12`, `sdr25`, `sdr50`, `sdr104`, `ddr50`

- `--max-seg <max>`

Maximum number of segments per request. This number is announced to the virtio interface and is also relevant for the required bounce buffer size, see below.

Numerical value.

Default: 64

- `--client <cap_name>`

Connect a static client.

Can be used multiple times.

Name of a provided capability with server rights that adheres to the ipc protocol.

This parameter opens a scope for the following subparameters:

- `--device <UUID>`

This option denotes the partition UUID of the partition to be exported for the client specified in the preceding `client` option.

String value.

- `--ds-max <max>`

This option sets the upper limit of the number of dataspace the client is able to register with the eMMC driver for virtio DMA.

Numerical value.

Default: 2

- `--readonly`

This option sets the access to disks or partitions to read only for the preceding `client` option.

Flag. True if provided.

- `--dma-map-all`

Map the entire client dataspace into the DMA space at the first I/O request and never unmap the dataspace until the client is destroyed. The default behavior is to map the relevant part of the dataspace before an I/O request and unmap it after the request.

Flag. True if provided.

## Virtio block host

Prior to connecting a client to a virtual block session it has to be created using the following Lua function. It has to be called on the client side of the IPC gate capability whose server side is bound to the eMMC driver.

Call: `create(0, "device=<UUID>" [, "ds-max=<max>", "readonly", "dma-map-all"])`

- `"device=<UUID>"`

This string denotes a partition UUID the client wants to be exported via the Virtio block interface.

String value.

- `"ds-max=<max>"`

Specifies the upper limit of the number of dataspace the client is allowed to register with the eMMC driver for virtio DMA.

Numerical value.

- Must be in the range of 1 to 256 inclusive.

Default: 2

- "readonly"

This option sets the access to disks or partitions to read only for this client connection.

Flag. True if provided.

- "dma-map-all"

Map the entire client dataspace into the DMA space at the first I/O request and never unmap the dataspace until the client is destroyed. The default behavior is to map the relevant part of the dataspace before an I/O request and unmap it after the request.

Flag. True if provided.

If the `create()` call is successful, a new capability which references an eMMC virtio driver is returned. A client uses this capability to communicate with the eMMC driver using the Virtio block protocol.

## Examples

A couple of examples on how to request different disks or partitions are listed below.

- Request a partition with the given UUID

```
vda1 = emmc_bus:create(0, "ds-max=5", "device=AFFA05B0-9379-480E-B9C6-5FF57FB1D194")
```

- A more elaborate example with a static client. The client uses the client side of the `emmc_c11` capability to communicate with the eMMC driver.

```
local emmc_c11 = L4.default_loader:new_channel();
local emmc_bus = L4.default_loader:new_channel();
L4.default_loader:start({
  caps = {
    vbus = vbus_emmc,
    svr = emmc_bus:svr(),
    c11 = emmc_c11:svr(),
  },
}, "rom/emmc-drv --client c11 --device 88E59675-4DC8-469A-98E4-B7B021DC7FBE --ds-max 5");
```

- Accessing a device from QEMU:

The file `pcie-ecam.io` contains an IO config file which is able to use the QEMU PCI controller to search for attached eMMC devices.

- eMMC emulation with QEMU:

The attached patch extends QEMU SD card emulation to emulate eMMC devices. After applying the patch and recompiling QEMU, attach the following parameters to your QEMU command line (assuming that `$HOME/foobar.img` is the eMMC medium):

```
-drive id=sd_disk,file=$(HOME)/foobar.img,if=none,format=raw \
-device sdhci-pci,id=sdhci \
-device sd-card,drive=sd_disk,spec_version=3,emmc=on
```

## 5.12 Cons, the Console Multiplexer

`cons` is an interactive multiplexer for console input and output. It buffers the output from different L4 clients and allows to switch between them to redirect input.

## Multiplexers and Frontends

cons is able to connect multiple clients to multiple console I/O servers. All clients are connected to all configured multiplexers

Multiplexers and frontends come in pairs where the actual I/O is handled by the frontend. From the point-of-view of cons, a frontend consists of an IPC channel to a server that speaks an appropriate server protocol. By default the L4.Env.log capability is used. Only frontends that speak the L4::Vcon protocol are supported.

A multiplexer handles and routes the I/O of each client to and from its respective frontend.

Each client's console settings (e.g. color, visibility) apply to all multiplexers and cannot be changed individually. A user can connect to all clients through all multiplexers. It is not possible to assign individual clients to distinct multiplexers.

## Client sessions

For clients cons implements the L4::Vcon and the Virtio console interface. A client session can be requested through a `create` call to cons' factory. cons binds its factory capability to the `cons` capability. See the example below on how this can be set up.

## Starting the service

The cons server can be started with Lua like this:

```
local log_server = L4.default_loader:new_channel()
L4.default_loader:start({
  caps = {
    cons = log_server:svr(),
  },
  log = L4.Env.log,
},
"rom/cons")
```

First an IPC gate (`log_server`) is created which is used between the cons server and a client to request a new session. The server side is assigned to the mandatory `cons` capability of cons. This example explicitly assigns the kernel's log capability (`L4.Env.log`) to cons' `log` capability in order to allow cons to provide input and output for its clients. The default log factory (usually provided by `moe`) doesn't provide input capabilities.

## Command Line Options

cons accepts the following command line switches:

- `-a, --show-all`  
Initially show output from all clients.
- `-B <size>, --defaultbufsize <size>`  
Default buffer size per client in bytes. Default: 40960

- `-c <client>, --autoconnect <client>`  
Automatically connect to the client with the given name. That means that output of this client will be visible and input will be routed to it.
- `-f <cap>, --frontend <cap>`  
Set the frontend for the current multiplexer. Output for the multiplexer is then sent to the capability with the given name `<cap>`. The server connected to the capability needs to understand the [L4::Vcon](#) protocol.
- `-k, --keep`  
Keep the console buffer when a client disconnects.
- `-l, --no-line-buffering`  
By default, merge the client output to entire lines. If the client writes characters without a final newline, the following client output is merged with the current line content. Specifying this switch disables the line buffered mode by default.
- `--line-buffering-ms <timeout>`  
Timeout in milliseconds before buffered client output is written even without a newline. Default value is 50.
- `-m <prompt name>, --mux <prompt name>`  
Add a new multiplexer named `<prompt name>`. This is necessary if output should be sent to different frontends. This option must be used in conjunction with the `-f` frontend option
- `-n, --defaultname`  
Default name for the multiplexer prompt. Default: `cons`.
- `-t, --timestamp`  
Prefix the output with timestamps.

## Connecting a client

```
create(backend_type, ["client_name"], ["color"], ["option"] [, "option"] ...)
```

- `backend_type`  
The type of backend that should be created for the client. The type is a positive integer and currently the following types are supported:
  - `L4.Proto.Log`: [L4::Vcon](#) client
  - `1`: Virtio console client

`cons` accepts the following per-client options:

- `bufsz=n`  
Use a buffer of `n` bytes for this client, deviating from the default buffer size.
- `keep / no-keep`  
The console buffer is kept / thrown away when the client disconnects.
- `key=<key>`  
Assign `<key>` as keyboard shortcut to this client.
- `line-buffering / no-linux-buffering`  
Line buffering is enabled / disabled for this client.
- `show / hide`  
Output from this client is initially shown / hidden.
- `timestamp / no-timestamp`  
Do / do not prefix the output of this client with timestamps.

## 5.13 AHCI driver

The AHCI driver is a driver for PCI serial ATA host controllers.

The AHCI driver is capable of exposing entire disks (by serial number) or individual partitions (by their partition UUID) of a hard drive to clients via the Virtio block interface.

The driver consists of two parts. The first one is the hardware driver itself that takes care of the communication with the underlying hardware and interrupt handling. The second part implements a virtual block device and is responsible to communicate with clients. The virtual block device translates commands it receives into AHCI requests and issues them to the hardware driver.

The AHCI driver allows both statically and dynamically configured clients. A static configuration is given priority over dynamically connecting clients and configured while the service starts. Dynamic clients can connect and disconnect during runtime of the AHCI driver.

### Capabilities

- `vbus`  
Virtual bus capability  
Mandatory capability.
- `client`  
Static client  
Multiple capability names can be provided by the `--client` command line parameter.
- `svr`  
Server Capability of application. Endpoint for IPC calls  
Mandatory capability.

### Command Line Options

In the example above the ahci driver is started in its default configuration. To customize the configuration of the ahci-driver it accepts the following command line options:

- `-A, --check-address`  
Disable check for address width of the device. Only do this if all physical memory is guaranteed to be below 4GB.  
Flag. True if provided.
- `-v, --verbose`  
Enable verbose mode. You can repeat this option up to three times to increase verbosity up to trace level.  
Can be used up to 3 times.  
Flag. True if provided.
- `-q, --quiet`  
This option enables the quiet mode. All output is silenced.  
Flag. True if provided.

- `--client <cap_name>`

Connect a static client.

Can be used multiple times.

Name of a provided capability with server rights that adheres to the ipc protocol.

This parameter opens a scope for the following subparameters:

- `--device <UUID | SN>`

This option denotes the partition UUID or serial number of the preceding `client` option.

String value.

- `--ds-max <max>`

This option sets the upper limit of the number of dataspace the client is able to register with the AHCI driver for virtio DMA.

Numerical value.

Default: 2

- `--slot-max <max>`

This option defines the maximum number of requests a single client may have in parallel running on the device. If a positive number is given, then this is considered the absolute number of slots to be used. If a negative number is given, then the client may use all available slots except the number given. In any case, a client gets at least 1 slot and at most the number of slots available in hardware. This parameter is only valid when a client accesses a partition and ignored otherwise.

Numerical value.

- `--readonly`

This option sets the access to disks or partitions to read only for the preceding `client` option.

Flag. True if provided.

## Building and Configuration

The AHCI driver can be built using the [L4Re](#) build system. Just place this project into your `pkg` directory. The resulting binary is called `ahci-drv`

## Starting the service

The AHCI driver can be started with Lua like this:

```
local ahci_bus = L4.default_loader:new_channel();
L4.default_loader:start({
  caps = {
    vbus = vbus_ahci,
    svr = ahci_bus:svr(),
  },
},
"rom/ahci-drv");
```

First an IPC gate (`ahci_bus`) is created which is used between the AHCI driver and a client to request access to a particular disk or partition. The server- side is assigned to the mandatory `svr` capability of the AHCI driver. See the section below on how to configure access to a disk or partition.

The ahci driver needs access to a virtual bus capability (`vbus`). On the virtual bus the AHCI driver searches for AHCI 1.0 compliant storage controllers. Please see io's documentation about how to setup a virtual bus.

## Virtio block host

Prior to connecting a client to a virtual block session it has to be created using the following Lua function. It has to be called on the client side of the IPC gate capability whose server side is bound to the ahci driver.

Call: `create(0, "device=<UUID | SN>" [, "ds-max=<max>", "slot-max=<max>"])`

- "device=<UUID | SN>"

This string denotes either a partition UUID or a disk serial number the client wants to be exported via the Virtio block interface.

String value.

- "ds-max=<max>"

Specifies the upper limit of the number of dataspace the client is allowed to register with the AHCI driver for virtio DMA.

Numerical value.

- In the range of 1 to 256 inclusive

Default: 2

- "slot-max=<max>"

Specifies the maximum number of requests that will be processed in parallel by the AHCI device. See `--slot-max` option above for details.

Numerical value.

If the `create()` call is successful a new capability which references an AHCI virtio driver is returned. A client uses this capability to communicate with the AHCI driver using the Virtio block protocol.

## Examples

A couple of examples on how to request different disks or partitions are listed below.

- Request a partition with the given UUID

```
vda1 = ahci_bus:create(0, "ds-max=5", "device=88E59675-4DC8-469A-98E4-B7B021DC7FBE")
```

- Request complete disk with the given serial number

```
vda = ahci_bus:create(0, "ds-max=4", "device=QM00005")
```

- A more elaborate example with a static client. The client uses the client side of the `ahci_c11` capability to communicate with the AHCI driver.

```
local ahci_c11 = L4.default_loader:new_channel();
local ahci_bus = L4.default_loader:new_channel();
L4.default_loader:start({
  caps = {
    vbus = vbus_ahci,
    svr = ahci_bus:svr(),
    c11 = ahci_c11:svr(),
  },
},
"rom/ahci-drv --client c11 --device 88E59675-4DC8-469A-98E4-B7B021DC7FBE --ds-max 5");
```



## Chapter 6

# uvmm\_dtg The device tree generator for Uvmm

A virtual machine in Uvmm is configured with a device tree that contains information about the VMs resources, memory layout, virtual CPUs and peripheral devices.

Uvmm\_dtg is a tool to generate such a device tree at runtime according to its command line.

### Capabilities

- dt

The dataspace that the device tree is put into.

### Command Line Options

<file | -->

-- prints to stdout. On [L4Re](#), the string given as <file> is interpreted as a named capability which needs to be backed by a sufficiently large Dataspace. On Linux, a file with the given name is created. In both cases, uvmm\_dtg will output into the named file.

String value.

- -h, --help

Show help.

Flag. True if provided.

- --arch <target architecture>

Select the target architecture.

Possible values for <target architecture> are x86, x86\_64, arm32, arm64, mips32, mips64

- --format <format>

Select the output format.

Possible values for <format> are

- txt: The device tree will be printed as plain text (dts).
- bin: The device tree will be output as binary (dtb).
- --mem-base <membase>  
Configure the start of the memory distribution. membase can be defined in both decimal and hex notations. uvmm\_dtg rounds the given base up to the platforms page size.  
This value can be overridden by memory devices with fixed addresses.  
Numerical value.
- --device <devicename:[Option1,Option2=value,Option3=value,...]>  
This configures a device.  
To get a list of supported devices, use --device help.  
To get help for a specific device, use --device devicename:help.  
String value.

## Examples

### Usage in L4Re

Example lua script for Ned:

```
-- Create DS holding device tree
local dt = L4.Env.user_factory:create(L4.Proto.Dataspace, 4 * 1024):m("rw");

-- Start the generator
L4.default_loader:start(
{
  caps = { dt = dt },
}, "rom/uvmm_dtg dt"):wait();

-- Start uvmm
vmm.start_vm
{
  ...
  ext_caps = { dt = dt },
  fdt = "dt",
  ...
}
```

Please notice the :wait() when starting uvmm\_dtg. This makes Ned pause until uvmm\_dtg has exited and put the device tree into the dataspace such that Uvmm can commence.

## Chapter 7

# Bootstrap, the L4 kernel bootstrapper

### Bootstrap Command Line Options

`bootstrap` and the kernel can be configured through command line switches. `bootstrap` is responsible for parsing both command lines: `bootstrap` options are the ones directly given to `bootstrap`, whereas kernel options are those directly given to the kernel, respectively.

When using Multiboot boot, the first module directly after `bootstrap` is considered the kernel: An example using GRUB2 might look like this:

```
multiboot /path/to/bootstrap bootstrap -bs-boolean-opt -bs-opt-with-argument=foo
module /path/to/fiasco fiasco -kernel-opt-with-argument=bar -kernel-boolean-opt
```

#### Note

The exact way to provide the command line to `bootstrap` is platform-dependent. On platforms supporting booting via Multiboot Specification the command line can be specified in the boot configuration (currently x86/amd64 only, e.g. using GRUB) whereas other platforms need the command line to be compiled into the `bootstrap` binary.

Platforms utilising flattened device trees usually also allow specifying a command line through the DT. This mechanism is **not** currently supported in `bootstrap`!

#### Note

`bootstrap` will ignore options it does not understand. For most cases, this also holds true if an option's arguments cannot be understood.

## bootstrap options

Command line options directly understood by `bootstrap` itself are as follows (passed via `bootstrap` command line):

- `-comirq=<irqno>` (x86/amd64 only)

If serial logging is enabled (default on), `<irqno>` defines which IRQ to use for serial port communication. This option is ignored if `-noserial` is also specified (see below).

- `-comport=<portspec>` (x86/amd64 only)

If serial logging is enabled (default on), `<portspec>` defines which serial port to use, being one of:

- `<number>`  
Use legacy port `<number>`, e.g. use `-comport=1` for the port commonly known as *COM1*, or
- `pci:<card>:<port>`  
Use serial port number `<port>` at PCI card number `<card>`, e.g. use `-comport=pci:0:1` for the second port on the first PCI card.
- `pci:probe`  
Use this to have `bootstrap` autodiscover all PCI serial lines. On each discovered line a message will be displayed, telling the correct `<portspec>` to be given to use the respective line.

### Note

`bootstrap` does not support specifying the serial port's baudrate and always uses 115200 bps.

- `-noserial`

Disable serial logging.

- `-wait`

Wait for key press at early startup.

### Note

This is not to be confused with the kernel's `-wait` option, see below.

- `-maxmem=<mbytes>`

Limit the available memory to at most `<mbytes>` MiB.

- `-mem=<size>@<offset>`

Add a region of memory of `<size>` at `<offset>` to the system's memory map. Both `<size>` and `<offset>` may be suffixed by either G, M, or K/k to denote GiB, MiB, or KiB, respectively.

This option may be specified multiple times. If the option is not given, a platform-specific method for determining the memory layout will be used, if available.

- `-presetmem=<intval>`

Initialise memory regions with `<intval>` before starting the kernel.

- `-modaddr=<paddr>`

Relocate modules to the physical address `<paddr>`. Use this when utilising a version of GRUB that lacks support for the `modaddr` command.

## Kernel Options

Command line options for the kernel (passed on kernel command line, i.e. to first module) `bootstrap` understands are as follows.

### Note

Availability of individual options might depend on used platform and/or actual kernel configuration.

- `-wait`  
Enter debugger directly after startup, prior to executing any task.
- `-serial_esc`  
Enable entering the debugger over serial line by pressing `Esc`.
- `-noserial`  
Disable serial logging.

### Note

If this is given as kernel command line argument, it does not affect `bootstrap` but only the kernel.

- `-noscreen`  
Disable VGA console.
- `-esc`  
Enable entering the debugger by pressing `Esc` on attached keyboard.
- `-nojdb`  
Disable the kernel debugger.
- `-nohlt`  
Enable quirk for broken HLT instruction.
- `-apic`  
Use Advanced Programmable Interrupt Controller (APIC) if available and known to be well-behaving.
- `-loadcnt`  
Use load counter for performance counting.
- `-watchdog`  
Enable watchdog timer.
- `-irq0`  
Allow IRQ 0 to be used by userland. This enables some sanity checks to ensure IRQ 0 is not used by the kernel, e.g. for profiling or scheduling purposes. This only has an effect on x86.
- `-nosfn`  
Disable SFN (special fully nested) mode of interrupt controller. This only has an effect on x86 with PIC8259 interrupt controller.
- `-jdb_never_stop`  
Prevent system from stopping to enter JDB. This only has an effect on x86.
- `-kmemsize=<KB>`  
Reserve <KB> KiB of memory for the kernel.

- `-tbuf_entries=<number>`  
Specify the `<number>` of trace buffer entries.
- `-out_buf=<length>`  
Specify length of console buffer to be `<length>` bytes.
- `-jdb_cmd=<ctrlseq>`  
Execute JDB command sequence `<ctrlseq>` right after start-up. If `-wait` is also given, `<ctrlseq>` is executed right before entering JDB.

## Module options

Bootstrap supports module attributes for `sigma0` and the `roottask`. They need to be specified in `modules.list`, e.g.:

```
sigma0[attr:nodes=4-7] ...
```

Attributes are not supported when using multi-boot on platforms that support it. The following attributes are supported:

- `nodes`

This is a colon separated list of AMP node ranges. A range can also be a single number. Examples:

- `nodes=1` – Only node 1
- `nodes=1-3` – Nodes 1 to 3 (inclusive)
- `nodes=0:2-3` – Nodes 0, 2 and 3

If not present, the `sigma0/roottask` module is applicable to all AMP nodes.

- `reloc`

Normally the `sigma0` or `roottask` images are loaded at the preferred load address if the RAM is available at the desired location. If this is not possible, they will be relocated to some free RAM region. Setting the "reloc" module attribute to a non-empty string will always request the dynamic relocation.

This attribute can be used on no-MMU systems to maximize the size of contiguous free RAM regions.

## Chapter 8

# Deprecated List

Global `L4::Rcv_endpoint::bind_thread (Ipc::Cap< Thread > t, I4_umword_t label)`

Use `bind_snd_destination()` instead.

Global `L4_CAP_SIZE`

Superseded by `L4_CAP_OFFSET`.

Global `I4_kip_clock_lw (I4_kernel_info_t const *kip) L4_NOTHROW`

Use `I4_kip_clock()` instead.

Global `L4Re::Util::Registry_server< LOOP_HOOKS >::Registry_server (I4_utcb_t *, L4::Cap< L4::Thread > server, L4::Cap< L4::Factory > factory)`

Note that this variant of the constructor is deprecated, please do not supply the UTCB pointer, it's not used.

Global `I4util_kip_for_each_feature (s)`

Use `I4_kip_for_each_feature()`.

Global `I4util_kip_kernel_has_feature (I4_kernel_info_t const *k, char const *str)`

Use `I4_kip_kernel_has_feature()`.

Global `I4util_micros2I4to (I4_uint64_t us) L4_NOTHROW`

Use `I4_timeout_from_us()`.





# Chapter 9

## Topic Index

### 9.1 Topics

Here is a list of all topics with brief descriptions:

Base API	157
Basic Macros	160
Fiasco extensions	165
Kernel Debugger	169
Kernel Information Dump	178
Kernel Tracing	179
Flexpages	182
C++ IPC Interface Definition.	199
Internal Helpers	200
Cache Consistency	200
Memory related	204
Error codes	213
Object Invocation	215
Message Items	236
Timeouts	243
Error Handling	251
Realtime API	257
Message Tag	257
Virtual Registers (UTCBs)	270
Message Registers (MRs)	275
Exception registers	276
Buffer Registers (BRs)	279
Thread Control Registers (TCRs)	280
ARM Virtual Registers (UTCB)	280
ARM64 Virtual Registers (UTCB)	281
AMD64 Virtual Registers (UTCB)	282
x86 Virtual Registers (UTCB)	282
Kernel Objects	284
IPC-Gate API	286
DMA space	291
L4 kernel object type information	291
Factory	293
Virtual Machines	303
VM API for SVM	304

VM API for VMX . . . . .	305
VM API for TZ . . . . .	325
Interrupt controller . . . . .	325
IRQs . . . . .	342
Platform Control C API . . . . .	357
Scheduler . . . . .	362
Kernel-provided semaphore . . . . .	370
Task . . . . .	373
Thread . . . . .	386
Thread control . . . . .	409
vCPU API . . . . .	415
Thread groups . . . . .	420
Virtual Console . . . . .	422
Kernel Interface Page . . . . .	436
Memory descriptors (C version) . . . . .	445
Capabilities . . . . .	450
Memory operations. . . . .	454
Integer Types . . . . .	457
EDID parsing functionality . . . . .	459
IO interface . . . . .	463
IPC Helpers . . . . .	472
IRQ handling library . . . . .	474
Interface using direct functionality. . . . .	474
Interface using direct functionality. . . . .	479
Interface for asynchronous ISR handlers. . . . .	481
Interface for asynchronous ISR handlers with a given IRQ capability. . . . .	483
L4 IPC Opcodes . . . . .	484
L4 VIRTIO Interface . . . . .	488
L4 VIRTIO Transport Layer . . . . .	489
L4 VIRTIO Block Device . . . . .	499
L4 VIRTIO Input Device . . . . .	500
L4 VIRTIO Network Device . . . . .	501
L4 Vbus functions . . . . .	502
L4vbus GPIO functions . . . . .	513
L4vbus PCI functions . . . . .	523
L4vbus power management functions . . . . .	529
L4Re C Interface . . . . .	531
L4Re Util C Interface . . . . .	534
Dataspace interface . . . . .	534
Debug interface . . . . .	539
DMA Space Interface . . . . .	540
Event interface . . . . .	544
Log interface . . . . .	547
Memory allocator . . . . .	550
Namespace interface . . . . .	556
Parent interface . . . . .	560
Region map interface . . . . .	560
Capability allocator . . . . .	575
Kumem allocator utility . . . . .	576
Video API . . . . .	577
Initial Environment . . . . .	584
L4Re C++ Interface . . . . .	589
L4Re Util C++ Interface . . . . .	591
L4Re Capability API . . . . .	592
Kumem utilities . . . . .	594
Console API . . . . .	596

Debugging API . . . . .	596
L4Re ELF Auxiliary Information . . . . .	597
Event API . . . . .	599
Auxiliary data . . . . .	600
Logging interface . . . . .	601
Name-space API . . . . .	601
Parent API . . . . .	602
L4Re Protocol identifiers . . . . .	603
Region map API . . . . .	605
Video API . . . . .	606
C++ Exceptions . . . . .	607
Vbus API . . . . .	608
L4SHM-based ring buffer implementation . . . . .	609
Sender . . . . .	609
Receiver . . . . .	610
Internal . . . . .	610
Shared Memory Library . . . . .	612
Chunks . . . . .	617
Producer . . . . .	621
Consumer . . . . .	625
Signals . . . . .	630
Producer . . . . .	633
Consumer . . . . .	634
Sigma0 API . . . . .	639
Internal constants . . . . .	643
Small C++ Template Library . . . . .	644
The L4Re IPC Framework . . . . .	648
Server-Side IPC framework . . . . .	649
Utility Functions . . . . .	650
Bitmap graphics and fonts . . . . .	657
Functions for rendering bitmap data in frame buffers . . . . .	657
Functions for rendering bitmap fonts to frame buffers . . . . .	658
CPU related functions . . . . .	658
Timestamp Counter . . . . .	660
Atomic Instructions . . . . .	667
Internal functions . . . . .	685
Bit Manipulation . . . . .	685
ELF binary format . . . . .	693
Kernel Interface Page API . . . . .	719
Comfortable Command Line Parsing . . . . .	721
Random number support . . . . .	724
Low-Level Thread Functions . . . . .	725
IA32 Port I/O API . . . . .	725
Virtio Net Switch . . . . .	731
vCPU Support Library . . . . .	732
Extended vCPU support . . . . .	739



## Chapter 10

# Namespace Index

### 10.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

<a href="#">cxx</a>	Our C++ library . . . . .	741
<a href="#">cxx::Bits</a>	Internal helpers for the cxx package . . . . .	747
<a href="#">L4</a>	<a href="#">L4</a> low-level kernel interface . . . . .	748
<a href="#">L4::lpc</a>	IPC related functionality . . . . .	758
<a href="#">L4::lpc::Msg</a>	IPC Message related functionality . . . . .	766
<a href="#">L4::lpc_svr</a>	Helper classes for <a href="#">L4::Server</a> instantiation . . . . .	773
<a href="#">L4::Typeid</a>	Definition of interface data-type helpers . . . . .	774
<a href="#">L4::Types</a>	<a href="#">L4</a> basic type helpers for C++ . . . . .	774
<a href="#">L4Re</a>	<a href="#">L4Re</a> C++ Interfaces . . . . .	775
<a href="#">L4Re::Util</a>	Documentation of the <a href="#">L4</a> Runtime Environment utility functionality in C++ . . . . .	794
<a href="#">L4Re::Vfs</a>	Virtual file system for interfaces in POSIX libc . . . . .	803
<a href="#">L4vbus</a>	C++ interface of the <a href="#">Vbus</a> API. . . . .	804
<a href="#">L4virtio</a>	L4-VIRTIO Transport C++ API . . . . .	805



# Chapter 11

## Hierarchical Index

### 11.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Block_device::Device_discard_feature	807
Block_device::Device_mgr< DEV, FACTORY, SCHEDULER >	808
Block_device::Device_with_notification_domain< DEV >	810
Block_device::Dma_region_info	811
Block_device::Impl::Partitioned_device_discard_mixin< PART_DEV, BASE_DEV, bool >	818
Block_device::Impl::Partitioned_device_discard_mixin< PART_DEV, BASE_DEV, true >	819
Block_device::Impl::Partitioned_device_discard_mixin< Partitioned_device< Device >, Device >	818
Block_device::Partitioned_device< BASE_DEV >	826
Block_device::Inout_block	821
Block_device::Inout_memory< DEV >	822
Block_device::Inout_memory< Device_type >	822
Block_device::Mem_region_info	823
Block_device::Notification_domain	823
Block_device::Partition_info	824
Block_device::Partition_reader< DEV >	825
Block_device::Pending_request	828
Block_device::Scheduler_base< DEV >	831
Block_device::Rr_scheduler< DEV >	829
cxx::arith::Ld< V >	834
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >	855
cxx::Base_slab< sizeof(Type), L4_PAGESIZE, 2, New_allocator >	855
cxx::Slab< Type, Slab_size, Max_free, Alloc >	979
cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >	860
cxx::Base_slab_static< sizeof(Type), L4_PAGESIZE, 2, New_allocator >	860
cxx::Slab_static< Type, Slab_size, Max_free, Alloc >	984
cxx::Bitfield< T, LSB, MSB >	865
cxx::Bitfield< T, LSB, MSB >::Value_base< TT >	874
cxx::Bitfield< T, LSB, MSB >::Value< Base_type & >	873
cxx::Bitfield< T, LSB, MSB >::Value< Base_type volatile & >	873
cxx::Bitfield< T, LSB, MSB >::Value< Base_type const >	873
cxx::Bitfield< T, LSB, MSB >::Value_unshifted< Base_type & >	875
cxx::Bitfield< T, LSB, MSB >::Value_unshifted< Base_type volatile & >	875
cxx::Bitfield< T, LSB, MSB >::Value_unshifted< Base_type const >	875
cxx::Bitfield< T, LSB, MSB >::Value< TT >	873

cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT > . . . . .	875
cxx::Bitmap_base . . . . .	881
cxx::Bitmap< BITS > . . . . .	876
cxx::Bitmap_base::Bit . . . . .	893
cxx::Bitmap_base::Char< BITS > . . . . .	894
cxx::Bitmap_base::Word< BITS > . . . . .	895
cxx::Bits::Avl_map_get_key< KEY_TYPE > . . . . .	896
cxx::Bits::Avl_set_get_key< KEY_TYPE > . . . . .	896
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY > . . . . .	897
cxx::Avl_map< Region, Hdlr, cxx::Lt_functor, Alloc > . . . . .	835
cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC > . . . . .	840
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node . . . . .	908
cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, Lt_functor< KEY_TYPE >, New_allocator, Bits::Avl_map_get_key< KEY_TYPE > > . . . . .	897
cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC > . . . . .	835
cxx::Bits::Basic_list< POLICY > . . . . .	910
cxx::H_list< Timeout > . . . . .	936
cxx::H_list< Observer > . . . . .	936
cxx::H_list< cxx::Base_slab::Slab_i > . . . . .	936
cxx::S_list< T, POLICY > . . . . .	976
cxx::H_list< T, POLICY > . . . . .	936
cxx::Bits::Basic_list< Bits::Basic_list_policy< T, H_list_item_t< T > > > . . . . .	910
cxx::H_list< T, Bits::Basic_list_policy< T, H_list_item_t< T > > > . . . . .	936
cxx::H_list_t< T > . . . . .	946
cxx::Bits::Basic_list< Bits::Basic_list_policy< T, S_list_item > > . . . . .	910
cxx::S_list< T, Bits::Basic_list_policy< T, S_list_item > > . . . . .	976
cxx::S_list< T, POLICY > . . . . .	976
cxx::Bits::Basic_list< Bits::Basic_list_policy< Weak_ref_base, H_list_item_t< Weak_ref_base > > > . . . . .	910
cxx::H_list< Weak_ref_base, Bits::Basic_list_policy< Weak_ref_base, H_list_item_t< Weak_ref_base > > > . . . . .	936
cxx::H_list_t< Weak_ref_base > . . . . .	946
cxx::Weak_ref_base::List . . . . .	1003
cxx::Bits::Bst< Node, Get_key, Compare > . . . . .	913
cxx::Avl_tree< Entry, Names_get_key > . . . . .	844
cxx::Avl_tree< _Node, GET_KEY, COMPARE > . . . . .	844
cxx::Avl_tree< Node, Get_key, Compare > . . . . .	844
cxx::Bits::Bst< _Node, Bits::Avl_map_get_key< KEY_TYPE >, Lt_functor< KEY_TYPE > > > . . . . .	913
cxx::Avl_tree< _Node, Bits::Avl_map_get_key< KEY_TYPE >, Lt_functor< KEY_TYPE > > > . . . . .	844
cxx::Bits::Bst< _Node, Bits::Avl_set_get_key< ITEM_TYPE >, Lt_functor< ITEM_TYPE > > > . . . . .	913
cxx::Avl_tree< _Node, Bits::Avl_set_get_key< ITEM_TYPE >, Lt_functor< ITEM_TYPE > > > . . . . .	844
cxx::Bits::Bst_node . . . . .	927
cxx::Avl_tree_node . . . . .	853
cxx::Bits::Direction . . . . .	929
cxx::Bits::Smart_ptr_list< ITEM > . . . . .	931
cxx::Bits::Smart_ptr_list_item< T, STORE_T > . . . . .	934
cxx::Bits::Smart_ptr_list_item< T, cxx::Ref_ptr< T > > . . . . .	934
cxx::Ref_obj_list_item< Connection > . . . . .	969
cxx::Ref_obj_list_item< T > . . . . .	969
cxx::H_list_item_t< ELEM_TYPE > . . . . .	944
cxx::H_list_item_t< void > . . . . .	944
L4::lpc_svr::Timeout . . . . .	1273
Block_device::Errand::Errand . . . . .	811
Block_device::Errand::Poll_errand . . . . .	815
cxx::H_list_item_t< Weak_ref_base > . . . . .	944



cxx::Weak_ref_base	1000
cxx::Weak_ref< T >	996
cxx::List< D, Alloc >	950
cxx::List< D, Alloc >::Iter	952
cxx::List_alloc	953
cxx::List_item	956
cxx::List_item::Iter	960
cxx::List_item::T_iter< E >	961
cxx::List_item::T_iter< T, Poly >	961
cxx::Lt_functor< Obj >	963
cxx::New_allocator< _Type >	963
cxx::Nothrow	964
cxx::Pair< First, Second >	965
cxx::Pair_first_compare< Cmp, Typ >	968
cxx::Ref_ptr< T, CNT >	971
cxx::Ref_ptr< Device_type >	971
cxx::Ref_ptr< L4Re::Vfs::File >	971
cxx::Ref_ptr< Mount_tree >	971
cxx::Ref_ptr< T, CNT >	971
cxx::static_vector< T, IDX >	988
cxx::String	989
L4Re::Util::Names::Name	1827
Elf32_Auxv	1007
Elf32_Dyn	1008
Elf32_Ehdr	1009
Elf32_Phdr	1012
Elf32_Rel	1013
Elf32_Rela	1014
Elf32_Shdr	1015
Elf32_Sym	1016
Elf64_Auxv	1017
Elf64_Dyn	1018
Elf64_Ehdr	1019
Elf64_Phdr	1022
Elf64_Rel	1023
Elf64_Rela	1024
Elf64_Shdr	1025
Elf64_Sym	1026
gfxbitmap_offset	1027
L4::Alloc_list	1028
L4::Basic_registry	1033
L4Re::Util::Object_registry	1836
L4::Cap_base	1044
L4::Cap< A >	1038
L4::Cap< L4::Rcv_endpoint >	1038
L4::Cap< L4Re::Rm >	1038
L4::Cap< L4::Irq >	1038
L4::Cap< L4Re::Namespace >	1038
L4::Cap< L4Re::Dataspace >	1038
L4::Cap< L4::Vcon >	1038
L4::Cap< L4::Semaphore >	1038
L4::Cap< L4::Thread >	1038
L4::Cap< L4::Factory >	1038
L4::Cap< L4Re::Video::Goos >	1038
L4::Cap< L4vbus::Vbus >	1038
L4::Cap< L4virtio::Device >	1038
L4::Smart_cap< T, Smart_count_cap< L4_FP_ALL_SPACES > >	1418

L4::Smart_cap< T, Smart_count_cap< L4_FP_DELETE_OBJ > > . . . . .	1418
L4::Cap< T > . . . . .	1038
L4::Smart_cap< T, SMART > . . . . .	1418
L4::Epiface . . . . .	1075
L4::Epiface_t0< void, Epiface > . . . . .	1084
L4::Irqep_t< Irq_object > . . . . .	1302
L4::Irqep_t< Host_irq > . . . . .	1302
L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::Host_irq . . . . .	2230
L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >::Host_irq . . . . .	2242
L4virtio::Svr::Virtio_rng< Rnd_state, Epiface >::Host_irq . . . . .	2253
L4::Irqep_t< Del_cap_irq > . . . . .	1302
L4::Irqep_t< Kick_irq > . . . . .	1302
L4::Irqep_t< Derived, BASE, bool > . . . . .	1302
L4::Epiface_t0< L4virtio::Device, L4::Epiface > . . . . .	1084
L4::Epiface_t< Virtio_client< DEV >, L4virtio::Device > . . . . .	1080
L4::Epiface_t< Block_dev< Ds_data >, L4virtio::Device > . . . . .	1080
L4::Epiface_t< Virtio_gpio< Request_handler, L4virtio::Device >, L4virtio::Device > . . . . .	1080
L4virtio::Svr::Virtio_gpio< Request_handler, Epiface > . . . . .	2225
L4::Epiface_t< Virtio_i2c< Request_handler, L4virtio::Device >, L4virtio::Device > . . . . .	1080
L4virtio::Svr::Virtio_i2c< Request_handler, Epiface > . . . . .	2238
L4::Epiface_t< Virtio_rng< Rnd_state >, L4virtio::Device > . . . . .	1080
L4virtio::Svr::Virtio_rng< Rnd_state, Epiface > . . . . .	2249
L4::Epiface_t< Virtio_net, L4virtio::Device > . . . . .	1080
Virtio_net . . . . .	2349
L4virtio_port . . . . .	2311
L4::Epiface_t0< IFACE, L4::Epiface > . . . . .	1084
L4::Epiface_t< Derived, IFACE, BASE, bool > . . . . .	1080
L4::Epiface_t0< L4::Kobject, L4::Epiface > . . . . .	1084
L4::Epiface_t< Null_handler, L4::Kobject > . . . . .	1080
L4::Epiface_t0< L4::Factory, L4::Epiface > . . . . .	1084
L4::Epiface_t< Switch_factory, L4::Factory > . . . . .	1080
Switch_factory . . . . .	2345
L4::Epiface_t0< Virtio_net_switch::Statistics_if, L4::Epiface > . . . . .	1084
L4::Epiface_t< Stats_reader, Virtio_net_switch::Statistics_if > . . . . .	1080
L4::Epiface_t0< RPC_IFACE, BASE > . . . . .	1084
L4::Server_object . . . . .	1404
L4::Server_object_t< Kobject > . . . . .	1409
L4::Irq_handler_object . . . . .	1298
L4::Server_object_t< IFACE, L4::Server_object > . . . . .	1409
L4::Server_object_x< Derived, IFACE, BASE > . . . . .	1415
L4::Server_object_t< IFACE, BASE > . . . . .	1409
L4::Server_object_x< Derived, IFACE, BASE > . . . . .	1415
L4::Epiface_t0< IFACE, BASE > . . . . .	1084
L4::Epiface_t0< void, BASE > . . . . .	1084
L4::Exception_tracer . . . . .	1089
L4::Base_exception . . . . .	1030
L4::Invalid_capability . . . . .	1120
L4::Runtime_error . . . . .	1378
L4::Bounds_error . . . . .	1035
L4::Com_error . . . . .	1058
L4::Element_already_exists . . . . .	1070
L4::Element_not_found . . . . .	1072
L4::Out_of_memory . . . . .	1347
L4::Unknown_error . . . . .	1527
L4::Factory::Lstr . . . . .	1101
L4::Factory::Nil . . . . .	1103

L4::Factory::S	1103
L4::IOModifier	1132
L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX >	1135
L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE >	1137
L4::lpc::Array_ref< A, LEN >	1137
L4::lpc::Array< A, LEN >	1132
L4::lpc::Array< A, LEN > &	1132
L4::lpc::Array_ref< char const, unsigned long >	1137
L4::lpc::Array< char const, unsigned long >	1132
L4::lpc::Array_ref< ELEM_TYPE, Array_len_default >	1137
L4::lpc::Array< ELEM_TYPE, LEN_TYPE >	1132
L4::lpc::Array_ref< X, LEN_TYPE >	1137
L4::lpc::Array< X, LEN_TYPE >	1132
L4::lpc::As_value< T >	1138
L4::lpc::Call	1139
L4::lpc::Call_t< RIGHTS >	1140
L4::lpc::Call_zero_send_timeout	1141
L4::lpc::Cap< T >	1143
L4::lpc::Gen_fpage	1146
L4::lpc::Rcv_fpage	1206
L4::lpc::Snd_fpage	1215
L4::lpc::In_out< T >	1149
L4::lpc::Istream	1161
L4::lpc::Iostream	1150
L4::lpc::Msg::Cls_buffer	1173
L4::lpc::Msg::Do_rcv_buffers	1182
L4::lpc::Msg::Cls_data	1174
L4::lpc::Msg::Do_in_data	1178
L4::lpc::Msg::Do_out_data	1180
L4::lpc::Msg::Cls_item	1175
L4::lpc::Msg::Do_in_items	1179
L4::lpc::Msg::Do_out_items	1181
L4::lpc::Msg::Dir_in	1176
L4::lpc::Msg::Do_in_data	1178
L4::lpc::Msg::Do_in_items	1179
L4::lpc::Msg::Do_rcv_buffers	1182
L4::lpc::Msg::Dir_out	1177
L4::lpc::Msg::Do_out_data	1180
L4::lpc::Msg::Do_out_items	1181
L4::lpc::Msg::Elem< Array< A, LEN > & >	1184
L4::lpc::Msg::Elem< Array< A, LEN > >	1185
L4::lpc::Msg::Elem< Array_ref< A, LEN > & >	1186
L4::lpc::Msg::Svr_arg_pack< IPC_TYPE >	1191
L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS >	1191
L4::lpc::Msg::Svr_val_ops< L4::lpc::Snd_fpage, Dir_in, CLASS >	1191
L4::lpc::Msg_ptr< T >	1196
L4::lpc::Opt< T >	1197
L4::lpc::Ostream	1199
L4::lpc::Iostream	1150
L4::lpc::Out< T >	1205
L4::lpc::Ret_array< T >	1212
L4::lpc::Send_only	1213
L4::lpc::Small_buf	1214
L4::lpc::Str_cp_in< T >	1226

L4::lpc::Varg	1227
L4::lpc::Varg_list_ref	1235
L4::lpc::Varg_list< MAX >	1233
L4::lpc::Varg_list_ref::iterator	1238
L4::lpc_svr::Compound_reply	1249
L4::lpc_svr::Default_loop_hooks	1253
L4::Server< L4::lpc_svr::Default_loop_hooks >	1398
L4Re::Util::Registry_server< LOOP_HOOKS >	1844
L4::Server< LOOP_HOOKS >	1398
L4Re::Util::Registry_server< Loop_hooks >	1844
L4Re::Util::Br_manager_hooks	1790
L4::lpc_svr::Default_setup_wait	1255
L4::lpc_svr::Default_timeout	1256
L4::lpc_svr::Default_loop_hooks	1253
L4Re::Util::Br_manager_hooks	1790
L4::lpc_svr::Direct_dispatch< R >	1258
L4::lpc_svr::Dbg_dispatch< R, Exc, Printer >	1251
L4::lpc_svr::Exc_dispatch< R, Exc >	1262
L4::lpc_svr::Direct_dispatch< R * >	1260
L4::lpc_svr::Ignore_errors	1263
L4::lpc_svr::Default_loop_hooks	1253
L4Re::Util::Br_manager_hooks	1790
L4Re::Util::Br_manager_timeout_hooks	1792
L4::lpc_svr::Server_iface	1265
L4::lpc_svr::Timeout_queue_hooks< Loop_hooks, L4Re::Util::Br_manager >	1280
L4::lpc_svr::Timeout_queue_hooks< Br_manager_timeout_hooks, Br_manager >	1280
L4Re::Util::Br_manager_timeout_hooks	1792
L4::lpc_svr::Br_manager_no_buffers	1245
L4::lpc_svr::Default_loop_hooks	1253
L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >	1280
L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >	1280
L4Re::Util::Br_manager	1784
L4::lpc_svr::Timeout_queue_hooks< Loop_hooks, L4Re::Util::Br_manager >	1280
L4Re::Util::Br_manager_hooks	1790
L4::lpc_svr::Timeout_queue	1277
L4::Kip::Mem_desc	1306
L4::Kobject	1318
L4::Kobject_t< Arm_smccc, L4::Kobject, PROTO, Type_info::Demand_t<> >	1330
L4::Kobject_t< Debugger, Kobject, L4_PROTO_DEBUGGER >	1330
L4::Debugger	1061
L4::Kobject_t< Exception, L4::Kobject, PROTO, Type_info::Demand_t<> >	1330
L4::Kobject_t< Factory, Kobject, L4_PROTO_FACTORY >	1330
L4::Factory	1090
L4::Kobject_t< Mem_alloc, L4::Factory, L4RE_PROTO_MEM_ALLOC >	1330
L4Re::Mem_alloc	1695
L4::Kobject_t< Io_pager, L4::Kobject, PROTO, Type_info::Demand_t<> >	1330
L4::Kobject_t< Irq_eoi, L4::Kobject, PROTO, Type_info::Demand_t<> >	1330
L4::Kobject_t< Derived, L4::Kobject, L4::PROTO_ANY, Type_info::Demand_t<> >	1330
L4::Kobject_t< Meta, Kobject, L4_PROTO_META >	1330
L4::Meta	1342
L4::Kobject_t< Platform_control, Kobject, L4_PROTO_PLATFORM_CTL >	1330
L4::Platform_control	1354
L4::Kobject_t< Rcv_endpoint, Kobject, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > >	1330
L4::Rcv_endpoint	1368
L4::Kobject_2t< Irq, Triggerable, Rcv_endpoint, L4_PROTO_IRQ_SENDER >	1322

L4::Irq	1286
L4::Kobject_t< Ipc_gate, Rcv_endpoint, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > >	1330
L4::Ipc_gate	1239
L4::Kobject_t< Snd_destination, Kobject, L4_PROTO_KOBJECT >	1330
L4::Kobject_t< Task, Kobject, L4_PROTO_TASK, Type_info::Demand_t< 2 > >	1330
L4::Task	1423
L4::Kobject_t< Vm, Task, L4_PROTO_VM >	1330
L4::Vm	1539
L4::Vm	1539
L4::Kobject_t< Vcpu_context, Kobject, L4_PROTO_VCPU_CONTEXT >	1330
L4::Kobject_t< Dataspace, L4::Kobject, L4RE_PROTO_DATASPACE, L4::Type_info::Demand_t< 1 > >	1330
L4Re::Dataspace	1618
L4::Kobject_3t< Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event >	1326
L4vbus::Vbus	2017
L4::Kobject_t< Dbg_events, L4::Kobject, 0, L4::Type_info::Demand_t< 2 > >	1330
L4::Kobject_t< Debug_obj, L4::Kobject, L4RE_PROTO_DEBUG >	1330
L4Re::Debug_obj	1634
L4::Kobject_t< Dma_space, L4::Kobject, PROTO, L4::Type_info::Demand_t< 1 > >	1330
L4::Kobject_t< Inhibitor, L4::Kobject, L4RE_PROTO_INHIBITOR >	1330
L4Re::Inhibitor	1673
L4::Kobject_3t< Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event >	1326
L4::Kobject_t< Itas, L4::Kobject, L4RE_PROTO_ITAS, L4::Type_info::Demand_t< 2 > >	1330
L4Re::Itas	1678
L4::Kobject_t< Mmio_space, L4::Kobject, L4RE_PROTO_MMIO_SPACE >	1330
L4Re::Mmio_space	1703
L4::Kobject_t< Namespace, L4::Kobject, L4RE_PROTO_NAMESPACE, L4::Type_info::Demand_t< 1 > >	1330
L4Re::Namespace	1710
L4::Kobject_t< Cmd_control, L4::Kobject, L4::PROTO_ANY, Type_info::Demand_t<> >	1330
L4::Kobject_t< Parent, L4::Kobject, L4RE_PROTO_PARENT >	1330
L4Re::Parent	1722
L4::Kobject_t< Goos, L4::Kobject, L4RE_PROTO_GOOS >	1330
L4Re::Video::Goos	1912
L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >	1322
L4::Kobject_2t< Console, Video::Goos, Event, L4::PROTO_EMPTY >	1322
L4Re::Console	1611
L4::Kobject_2t< Debug_obj_t< BASE >, BASE, Debug_obj, L4::PROTO_EMPTY >	1322
L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >	1326
L4::Kobject_demand< T >	1329
L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >	1330
L4::Kobject_t< Remote_access, Dataspace, L4RE_PROTO_REMOTE_ACCESS >	1330
L4::Kobject_typeid< T >	1332
L4::Kobject_typeid< void >	1335
L4::Kobject_x< Derived, ARGS >	1338
L4::Kobject_x< Iommu, Proto_t< L4_PROTO_IOMMU >, Type_info::Demand_t< 1 > >	1338
L4::Iommu	1127
L4::Lock_guard	1339
L4::Poll_timeout_counter	1362
L4::Poll_timeout_kipclock	1364
L4::Proto_t< P >	1368
L4::Registry_iface	1374
L4Re::Util::Object_registry	1836
L4::Server< LOOP_HOOKS >	1398
L4::String	1422
L4::Thread::Attr	1451

L4::Thread::Modify_senders	1455
L4::Type_info	1466
L4::Type_info::Demand	1467
L4::Type_info::Demand_t< __l::Max< Kobject_typeid< T1 >::Demand::Caps, Kobject_demand< T2... >::Caps >Res, Kobject_typeid< T1 >::Demand::Flags Kobject_demand< T2... >Flags, __l::Max< Kobject_typeid< T1 >::Demand::Mem, Kobject_demand< T2... >::Mem >Res, __l::Max< Kobject_typeid< T1 >::Demand::Ports, Kobject_demand< T2... >::Ports >Res >	1471
L4::Type_info::Demand_union_t< Kobject_typeid< T1 >::Demand, Kobject_demand< T2... >	1474
L4::Type_info::Demand_t< __l::Max< D1::Caps, D2::Caps >::Res, D1::Flags D2Flags, __l::Max< D1::Mem, D2::Mem >::Res, __l::Max< D1::Ports, D2::Ports >::Res >	1471
L4::Type_info::Demand_union_t< D1, D2 >	1474
L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >	1471
L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >Rpc< Y >	1479
L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >::Rpc< Y >	1482
L4::Typeid::Detail::Rpc_end	1483
L4::Typeid::Detail::type< OPCODE, 1,... >	1478
L4::Typeid::Detail::_Rpc< void, 0, OPERATION >	1478
L4::Typeid::Rpc_nocode< OPERATION >	1485
L4::Typeid::Detail::_Rpc< L4::Opcode, 0, RPCS... >	1478
L4::Typeid::Rpc< set_status_t, config_queue_t, register_ds_t, device_config_t, device_← notification_irq_t >	1488
L4::Typeid::Rpc< execute_t >	1488
L4::Typeid::Rpc< num_interfaces_t, interface_t, supports_t >	1488
L4::Typeid::Rpc< map_t, clear_t, info_t, copy_in_t, allocate_t, map_info_t >	1488
L4::Typeid::Rpc< request_backtrace_t >	1488
L4::Typeid::Rpc< map_t, unmap_t, associate_t, disassociate_t >	1488
L4::Typeid::Rpc< get_buffer_t, get_num_streams_t, get_stream_info_t, get_stream_info_for_id← t, get_axis_info_t, get_stream_state_for_id_t >	1488
L4::Typeid::Rpc< acquire_t, release_t, next_lock_info_t >	1488
L4::Typeid::Rpc< register_thread_t, unregister_thread_t, sigaction_t, sigaltstack_t, sigprocmask← t, sigpending_t, setitimer_t, getitimer_t, raise_t >	1488
L4::Typeid::Rpc< info_t >	1488
L4::Typeid::Rpc< mmio_read_t, mmio_write_t >	1488
L4::Typeid::Rpc< query_t, register_obj_t, unlink_t >	1488
L4::Typeid::Rpc< signal_t >	1488
L4::Typeid::Rpc< get_random_t >	1488
L4::Typeid::Rpc< read_mem_t, write_mem_t, terminate_t >	1488
L4::Typeid::Rpc< attach_t, detach_t, find_t, reserve_area_t, free_area_t, get_regions_t, get_← areas_t, get_info_t >	1488
L4::Typeid::Rpc< info_t, get_static_buffer_t, create_buffer_t, create_view_t, delete_buffer← t, delete_view_t, view_info_t, set_view_info_t, view_stack_t, view_refresh_t, refresh_t >	1488
L4::Typeid::Rpc< RPCS >	1488
L4::Typeid::Detail::_Rpc< OPCODE_TYPE, 0, RPCS... >	1478
L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS >	1490
L4::Typeid::Detail::_Rpc< l4_umword_t, 0, ARG... >	1478
L4::Typeid::Rpc_sys< bind_thread_t, get_infos_t >	1493
L4::Typeid::Rpc_sys< bind_t, unbind_t, info_t, msi_info_t, unmask_t, mask_t, set_mode_t >	1493
L4::Typeid::Rpc_sys< system_suspend_t, system_shutdown_t, cpu_allow_shutdown_t, cpu_← enable_t, cpu_disable_t >	1493
L4::Typeid::Rpc_sys< bind_thread_t >	1493
L4::Typeid::Rpc_sys< info_t, run_thread_t, idle_time_t >	1493
L4::Typeid::Rpc_sys< ARG >	1493
L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >	1479
L4::Typeid::Detail::_Rpc< OPCODE, O, X >	1478
L4::Typeid::Rpc_nocode< call_t >	1485
L4::Typeid::Rpc_nocode< exception_t >	1485

L4::Typeid::Rpc_nocode< create_t > . . . . .	1485
L4::Typeid::Rpc_nocode< io_page_fault_t > . . . . .	1485
L4::Typeid::Rpc_nocode< page_fault_t > . . . . .	1485
L4::Typeid::Rpc_nocode< debug_t > . . . . .	1485
L4::Typeid::P_dispatch< LIST > . . . . .	1483
L4::Typeid::Raw_ipc< CLASS > . . . . .	1484
L4::Typeid::Rpcs_code< OPCODE_TYPE > . . . . .	1489
L4::Types::Bool< V > . . . . .	1495
L4::Types::Bool< false > . . . . .	1495
L4::Types::False . . . . .	1497
L4::Types::Same< A, B > . . . . .	1510
L4::Types::Bool< I1::Proto !=PROTO_EMPTY &&I1::Proto==I2::Proto > . . . . .	1495
L4::Types::Bool< true > . . . . .	1495
L4::Types::True . . . . .	1513
L4::Ipc::Msg::ls_valid_rpc_type< A * > . . . . .	1189
L4::Ipc::Msg::ls_valid_rpc_type< T > . . . . .	1189
L4::Types::Flags< BITS_ENUM, UNDERLYING > . . . . .	1499
L4::Types::Flags_ops_t< DT > . . . . .	1503
L4::Types::Flags_ops_t< Flags > . . . . .	1503
L4::Types::Flags_ops_t< Flags_t< DT, T > > . . . . .	1503
L4::Types::Flags_t< DT, T > . . . . .	1506
L4::Types::Int_for_size< SIZE, bool > . . . . .	1508
L4::Types::Int_for_type< T > . . . . .	1509
L4::Uart . . . . .	1516
L4::Uart_apb . . . . .	1521
l4_buf_regs_t . . . . .	1544
l4_exc_regs_t . . . . .	1545
l4_fpage_t . . . . .	1548
l4_icu_info_t . . . . .	1549
L4::Icu::Info . . . . .	1119
l4_icu_msi_info_t . . . . .	1550
l4_kernel_info_mem_desc_t . . . . .	1551
l4_kernel_info_t . . . . .	1552
l4_msg_regs_t . . . . .	1553
l4_msgtag_t . . . . .	1554
l4_sched_cpu_set_t . . . . .	1557
l4_sched_param_t . . . . .	1561
l4_snd_fpage_t . . . . .	1562
l4_thread_regs_t . . . . .	1563
l4_timeout_s . . . . .	1564
l4_timeout_t . . . . .	1565
l4_vcon_attr_t . . . . .	1566
l4_vcpu_arch_state_t . . . . .	1568
l4_vcpu_ipc_regs_t . . . . .	1568
l4_vcpu_regs_t . . . . .	1569
l4_vcpu_state_t . . . . .	1573
L4vcpu::Vcpu . . . . .	2032
l4_vm_svm_vmcb_control_area . . . . .	1577
l4_vm_svm_vmcb_state_save_area . . . . .	1577
l4_vm_svm_vmcb_state_save_area_seg . . . . .	1578
l4_vm_svm_vmcb_t . . . . .	1579
l4_vm_tz_state . . . . .	1581
l4_vm_vmx_vcpu_infos_t . . . . .	1581
l4_vm_vmx_vcpu_state_t . . . . .	1582
l4_vm_vmx_vcpu_vmcs_t . . . . .	1584
l4_vmx_offset_table_t . . . . .	1585

L4drivers::Register_block< MAX_BITS, BLOCK > . . . . .	1589
L4drivers::Register_block_base< MAX_BITS > . . . . .	1593
L4drivers::Register_block_impl< BASE, MAX_BITS > . . . . .	1594
L4drivers::Register_block_impl< Mmio_register_block< 32 >, 32 > . . . . .	1594
L4drivers::Mmio_register_block< MAX_BITS > . . . . .	1587
L4drivers::Register_block_tmpl< BLOCK > . . . . .	1596
L4drivers::Register_block_tmpl< Register_block_base< MAX_BITS > > . . . . .	1596
L4drivers::Register_block_tmpl< Register_block_base< MAX_BITS > const > . . . . .	1596
L4drivers::Ro_register_block< MAX_BITS, BLOCK > . . . . .	1604
L4drivers::Ro_register_tmpl< BITS, BLOCK > . . . . .	1606
L4drivers::Register_tmpl< MAX_BITS, Block > . . . . .	1597
L4drivers::Register_tmpl< BITS, BLOCK > . . . . .	1597
L4Re::Cap_alloc . . . . .	1608
L4Re::Util::_Cap_alloc . . . . .	1766
L4Re::Core::Ref_ptr< T, CNT > . . . . .	1614
L4Re::Dataspace::F . . . . .	1631
L4Re::Dataspace::Stats . . . . .	1633
L4Re::Default_event_payload . . . . .	1637
L4Re::Env . . . . .	1645
L4Re::Event_buffer_t< PAYLOAD > . . . . .	1669
L4Re::Event_buffer_t< PAYLOAD >::Event . . . . .	1672
L4Re::Event_buffer_t< Default_event_payload > . . . . .	1669
L4Re::Util::Event_buffer_t< Default_event_payload > . . . . .	1816
L4Re::Util::Event_buffer_consumer_t< Default_event_payload > . . . . .	1813
L4Re::Event_buffer_t< PAYLOAD > . . . . .	1669
L4Re::Util::Event_buffer_t< PAYLOAD > . . . . .	1816
L4Re::Util::Event_buffer_consumer_t< PAYLOAD > . . . . .	1813
L4Re::Mem_alloc::Stats . . . . .	1701
L4Re::Rm::Area . . . . .	1755
L4Re::Rm::F . . . . .	1756
L4Re::Rm::Region . . . . .	1758
L4Re::Rm::Unique_region< T > . . . . .	1759
L4Re::Smart_cap_auto< Unmap_flags > . . . . .	1764
L4Re::Smart_count_cap< Unmap_flags > . . . . .	1765
L4Re::Util::Bitmap_base . . . . .	1773
cxx::Bitmap< BITS > . . . . .	876
L4Re::Util::Bitmap_base::Bit . . . . .	1782
L4Re::Util::Bitmap_base::Char< BITS > . . . . .	1782
L4Re::Util::Bitmap_base::Word< BITS > . . . . .	1783
L4Re::Util::Cap_alloc_base . . . . .	1796
L4Re::Util::Counter< COUNTER > . . . . .	1797
L4Re::Util::Counter_atomic< COUNTER > . . . . .	1799
L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg > . . . . .	1800
L4Re::Util::Dataspace_svr . . . . .	1806
L4Re::Util::Event_svr< SVR > . . . . .	1820
L4Re::Util::Event_t< PAYLOAD > . . . . .	1822
L4Re::Util::Item_alloc_base . . . . .	1827
L4Re::Util::Names::Name_space . . . . .	1831
L4Re::Util::Ref_cap< T > . . . . .	1842
L4Re::Util::Ref_del_cap< T > . . . . .	1843
L4Re::Util::Smart_cap_auto< Unmap_flags > . . . . .	1850
L4Re::Util::Smart_count_cap< Unmap_flags > . . . . .	1851
L4Re::Util::Vcon_svr< SVR > . . . . .	1852
L4Re::Util::Video::Goos_svr . . . . .	1853
L4Re::Vfs::Directory . . . . .	1865
L4Re::Vfs::File . . . . .	1871



L4Re::Vfs::Be_file	1858
L4Re::Vfs::File_system	1874
L4Re::Vfs::Be_file_system	1863
L4Re::Vfs::Fs	1876
L4Re::Vfs::Ops	1891
L4Re::Vfs::Generic_file	1881
L4Re::Vfs::File	1871
L4Re::Vfs::Mman	1890
L4Re::Vfs::Ops	1891
L4Re::Vfs::Regular_file	1894
L4Re::Vfs::File	1871
L4Re::Vfs::Special_file	1904
L4Re::Vfs::File	1871
L4Re::Video::Color_component	1907
L4Re::Video::Goos::Info	1922
L4Re::Video::Pixel_info	1924
L4Re::Video::View	1935
L4Re::Video::View::Info	1941
l4re_aux_t	1944
l4re_ds_stats_t	1945
l4re_elf_aux_mword_t	1945
l4re_elf_aux_t	1946
l4re_elf_aux_vma_t	1946
l4re_env_cap_entry_t	1947
l4re_env_t	1949
l4re_event_t	1951
l4re_video_color_component_t	1952
l4re_video_goos_info_t	1952
l4re_video_pixel_info_t	1954
l4re_video_view_info_t	1955
l4re_video_view_t	1956
l4shmc_ringbuf_head_t	1957
l4shmc_ringbuf_t	1958
l4util_l4mod_info	1959
l4util_l4mod_mod	1960
l4util_mb_addr_range_t	1961
l4util_mb_apm_t	1962
l4util_mb_drive_t	1963
l4util_mb_info_t	1964
l4util_mb_mod_t	1965
l4util_mb_vbe_ctrl_t	1966
l4util_mb_vbe_mode_t	1967
L4vbus::Gpio_module::Pin_slice	1988
L4vbus::Pm< DEC >	2014
L4vbus::Pm< Device >	2014
L4vbus::Device	1971
L4vbus::Gpio_module	1980
L4vbus::Gpio_pin	1988
L4vbus::Icu	1997
L4vbus::Pci_dev	2002
L4vbus::Pci_host_bridge	2008
l4vbus_device_t	2028
l4vbus_resource_t	2029
L4vcpu::State	2030
L4virtio::Driver::Block_device::Handle	2063
L4virtio::Driver::Device	2063

L4virtio::Driver::Block_device	2054
L4virtio::Driver::Virtio_net_device	2076
L4virtio::Driver::Virtio_net_device::Packet	2086
L4virtio::Ptr< T >	2097
L4virtio::Svr::Bad_descriptor	2101
L4virtio::Svr::Block_request< Ds_data >	2111
L4virtio::Svr::Console::Control_message	2114
L4virtio::Svr::Console::Control_request	2115
L4virtio::Svr::Console::Port	2139
L4virtio::Svr::Console::Device_port	2131
L4virtio::Svr::Console::Port::Transition	2143
L4virtio::Svr::Data_buffer	2161
Buffer	832
L4virtio::Svr::Dev_config	2165
L4virtio::Svr::Dev_features	2176
L4virtio::Svr::Console::Features	2135
L4virtio::Svr::Dev_status	2179
L4virtio::Svr::Device_t< DATA >	2182
L4virtio::Svr::Device_t< Ds_data >	2182
L4virtio::Svr::Block_dev_base< Ds_data >	2103
L4virtio::Svr::Device_t< Mem_region_info >	2182
L4virtio::Svr::Block_dev_base< Mem_region_info >	2103
L4virtio::Svr::Device_t< No_custom_data >	2182
L4virtio::Svr::Console::Virtio_con	2144
L4virtio::Svr::Console::Device	2117
L4virtio::Svr::Scmi::Scmi_dev	2220
L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >	2225
L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >	2238
L4virtio::Svr::Virtio_rng< Rnd_state, Epiface >	2249
Virtio_net	2349
L4virtio::Svr::Driver_mem_list_t< DATA >	2188
L4virtio::Svr::Driver_mem_region_t< DATA >	2194
L4virtio::Svr::Driver_mem_region_t< Ds_data >	2194
L4virtio::Svr::Driver_mem_region_t< Mem_region_info >	2194
L4virtio::Svr::Driver_mem_region_t< No_custom_data >	2194
L4virtio::Svr::Request_processor	2201
L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::Request_processor	2235
L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >::Request_processor	2246
L4virtio::Svr::Virtio_rng< Rnd_state, Epiface >::Request_processor	2257
L4virtio::Svr::Scmi::Base_attr_t	2208
L4virtio::Svr::Scmi::Performance_attr_t	2214
L4virtio::Svr::Scmi::Performance_describe_level_t	2215
L4virtio::Svr::Scmi::Performance_describe_levels_n_t	2215
L4virtio::Svr::Scmi::Performance_domain_attr_t	2217
L4virtio::Svr::Scmi::Proto< OBSERV >	2219
L4virtio::Svr::Scmi::Proto< Scmi_dev >	2219
L4virtio::Svr::Scmi::Base_proto	2209
L4virtio::Svr::Scmi::Perf_proto	2211
L4virtio::Svr::Scmi::Scmi_hdr_t	2224
L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::Irq_handler	2234
L4virtio::Svr::Virtqueue::Head_desc	2272
L4virtio::Virtqueue	2274
L4virtio::Driver::Virtqueue	2086
L4virtio::Svr::Virtqueue	2259
L4virtio::Virtqueue::Avail	2291

L4virtio::Virtqueue::Avail::Flags . . . . .	2293
L4virtio::Virtqueue::Desc . . . . .	2294
L4virtio::Virtqueue::Desc::Flags . . . . .	2296
L4virtio::Virtqueue::Used . . . . .	2298
L4virtio::Virtqueue::Used::Flags . . . . .	2299
L4virtio::Virtqueue::Used_elem . . . . .	2301
l4virtio_block_config_t . . . . .	2302
l4virtio_block_discard_t . . . . .	2303
l4virtio_block_header_t . . . . .	2304
l4virtio_config_hdr_t . . . . .	2305
l4virtio_config_queue_t . . . . .	2306
l4virtio_input_absinfo_t . . . . .	2307
l4virtio_input_config_t . . . . .	2308
l4virtio_input_devids_t . . . . .	2308
l4virtio_input_event_t . . . . .	2309
l4virtio_net_config_t . . . . .	2309
l4virtio_net_header_t . . . . .	2310
Mac_addr . . . . .	2316
Mac_table< Size > . . . . .	2316
Net_transfer . . . . .	2319
Rm::Area . . . . .	2337
Rm::F . . . . .	2338
Rm::Region . . . . .	2340
Rm::Unique_region< T > . . . . .	2340
Virtio_net_request . . . . .	2354
Virtio_switch . . . . .	2357
Virtio_vlan_mangle . . . . .	2362



## Chapter 12

# Data Structure Index

### 12.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">Block_device::Device_discard_feature</a>	807
Partial interface for devices that offer discard functionality . . . . .	
<a href="#">Block_device::Device_mgr&lt; DEV, FACTORY, SCHEDULER &gt;</a>	808
Basic class that scans devices and handles client connections . . . . .	
<a href="#">Block_device::Device_with_notification_domain&lt; DEV &gt;</a>	810
Device with a per-device notification domain . . . . .	
<a href="#">Block_device::Dma_region_info</a>	811
Base class used by the driver implementation to derive its own DMA mapping tracking structure	
<a href="#">Block_device::Errand::Errand</a>	811
Wrapper for a small task executed asynchronously in the server loop . . . . .	
<a href="#">Block_device::Errand::Poll_errand</a>	815
Wrapper for a regularly repeated task . . . . .	
<a href="#">Block_device::Impl::Partitioned_device_discard_mixin&lt; PART_DEV, BASE_DEV, bool &gt;</a>	818
Dummy class used when the device class is not derived from <a href="#">Device_discard_feature</a> . . . . .	
<a href="#">Block_device::Impl::Partitioned_device_discard_mixin&lt; PART_DEV, BASE_DEV, true &gt;</a>	819
Mixin implementing discard for partition devices . . . . .	
<a href="#">Block_device::Inout_block</a>	821
Description of an inout block to be sent to the device . . . . .	
<a href="#">Block_device::Inout_memory&lt; DEV &gt;</a>	822
Helper class that temporarily allocates memory that can be used for in/out operations with the device . . . . .	
<a href="#">Block_device::Mem_region_info</a>	823
Additional info stored in each <a href="#">L4virtio::Svr::Driver_mem_region_t</a> used for tracking dataspace-wide DMA mappings . . . . .	
<a href="#">Block_device::Notification_domain</a>	823
Opaque type for representing a notification domain . . . . .	
<a href="#">Block_device::Partition_info</a>	824
Information about a single partition . . . . .	
<a href="#">Block_device::Partition_reader&lt; DEV &gt;</a>	825
Partition table reader for block devices . . . . .	
<a href="#">Block_device::Partitioned_device&lt; BASE_DEV &gt;</a>	826
A partition device for the given device interface . . . . .	
<a href="#">Block_device::Pending_request</a>	828
Interface for pending requests . . . . .	
<a href="#">Block_device::Rr_scheduler&lt; DEV &gt;</a>	829
Round Robin scheduler class . . . . .	

<a href="#">Block_device::Scheduler_base&lt; DEV &gt;</a>	
Scheduler base class	831
<a href="#">Buffer</a>	
Data buffer used to transfer packets	832
<a href="#">cxx::arith::Ld&lt; V &gt;</a>	
Computes the binary logarithm of the given number at compile time	834
<a href="#">cxx::Avl_map&lt; KEY_TYPE, DATA_TYPE, COMPARE, ALLOC &gt;</a>	
AVL tree based associative container	835
<a href="#">cxx::Avl_set&lt; ITEM_TYPE, COMPARE, ALLOC &gt;</a>	
AVL set for simple comparable items	840
<a href="#">cxx::Avl_tree&lt; Node, Get_key, Compare &gt;</a>	
A generic AVL tree	844
<a href="#">cxx::Avl_tree_node</a>	
Node of an AVL tree	853
<a href="#">cxx::Base_slab&lt; Obj_size, Slab_size, Max_free, Alloc &gt;</a>	
Basic slab allocator	855
<a href="#">cxx::Base_slab&lt; Obj_size, Slab_size, Max_free, Alloc &gt;::Slab_i</a>	
Type of a slab	859
<a href="#">cxx::Base_slab_static&lt; Obj_size, Slab_size, Max_free, Alloc &gt;</a>	
Merged slab allocator (allocators for objects of the same size are merged together)	860
<a href="#">cxx::Bitfield&lt; T, LSB, MSB &gt;</a>	
Definition for a member (part) of a bit field	865
<a href="#">cxx::Bitfield&lt; T, LSB, MSB &gt;::Value&lt; TT &gt;</a>	
Internal helper type	873
<a href="#">cxx::Bitfield&lt; T, LSB, MSB &gt;::Value_base&lt; TT &gt;</a>	
Internal helper type	874
<a href="#">cxx::Bitfield&lt; T, LSB, MSB &gt;::Value_unshifted&lt; TT &gt;</a>	
Internal helper type	875
<a href="#">cxx::Bitmap&lt; BITS &gt;</a>	
A static bitmap	876
<a href="#">cxx::Bitmap_base</a>	
Basic bitmap abstraction	881
<a href="#">cxx::Bitmap_base::Bit</a>	
A writable bit in a bitmap	893
<a href="#">cxx::Bitmap_base::Char&lt; BITS &gt;</a>	
Helper abstraction for a byte contained in the bitmap	894
<a href="#">cxx::Bitmap_base::Word&lt; BITS &gt;</a>	
Helper abstraction for a word contained in the bitmap	895
<a href="#">cxx::Bits::Avl_map_get_key&lt; KEY_TYPE &gt;</a>	
Key-getter for <a href="#">Avl_map</a>	896
<a href="#">cxx::Bits::Avl_set_get_key&lt; KEY_TYPE &gt;</a>	
Internal, key-getter for <a href="#">Avl_set</a> nodes	896
<a href="#">cxx::Bits::Base_avl_set&lt; ITEM_TYPE, COMPARE, ALLOC, GET_KEY &gt;</a>	
Internal: AVL set with internally managed nodes	897
<a href="#">cxx::Bits::Base_avl_set&lt; ITEM_TYPE, COMPARE, ALLOC, GET_KEY &gt;::Node</a>	
A smart pointer to a tree item	908
<a href="#">cxx::Bits::Basic_list&lt; POLICY &gt;</a>	
Internal: Common functions for all head-based list implementations	910
<a href="#">cxx::Bits::Bst&lt; Node, Get_key, Compare &gt;</a>	
Basic binary search tree (BST)	913
<a href="#">cxx::Bits::Bst_node</a>	
Basic type of a node in a binary search tree (BST)	927
<a href="#">cxx::Bits::Direction</a>	
The direction to go in a binary search tree	929
<a href="#">cxx::Bits::Smart_ptr_list&lt; ITEM &gt;</a>	
List of smart-pointer-managed objects	931
<a href="#">cxx::Bits::Smart_ptr_list_item&lt; T, STORE_T &gt;</a>	
List item for an arbitrary item in a <a href="#">Smart_ptr_list</a>	934

<a href="#">cxx::H_list&lt; T, POLICY &gt;</a>	General double-linked list of unspecified <a href="#">cxx::H_list_item</a> elements . . . . .	936
<a href="#">cxx::H_list_item_t&lt; ELEM_TYPE &gt;</a>	Basic element type for a double-linked <a href="#">H_list</a> . . . . .	944
<a href="#">cxx::H_list_t&lt; T &gt;</a>	Double-linked list of typed <a href="#">H_list_item_t</a> elements . . . . .	946
<a href="#">cxx::List&lt; D, Alloc &gt;</a>	Doubly linked list, with internal allocation . . . . .	950
<a href="#">cxx::List&lt; D, Alloc &gt;::Iter</a>	Iterator . . . . .	952
<a href="#">cxx::List_alloc</a>	Standard list-based allocator . . . . .	953
<a href="#">cxx::List_item</a>	Basic list item . . . . .	956
<a href="#">cxx::List_item::Iter</a>	Iterator for a list of <a href="#">ListItem</a> -s . . . . .	960
<a href="#">cxx::List_item::T_iter&lt; T, Poly &gt;</a>	Iterator for derived classes from <a href="#">ListItem</a> . . . . .	961
<a href="#">cxx::Lt_func&lt; Obj &gt;</a>	Generic comparator class that defaults to the less-than operator . . . . .	963
<a href="#">cxx::New_allocator&lt; _Type &gt;</a>	Standard allocator based on <code>operator new ()</code> . . . . .	963
<a href="#">cxx::Nothrow</a>	Helper type to distinguish the <code>operator new</code> version that does not throw exceptions . . . . .	964
<a href="#">cxx::Pair&lt; First, Second &gt;</a>	Pair of two values . . . . .	965
<a href="#">cxx::Pair_first_compare&lt; Cmp, Typ &gt;</a>	Comparison functor for <a href="#">Pair</a> . . . . .	968
<a href="#">cxx::Ref_obj_list_item&lt; T &gt;</a>	Item for list linked via <a href="#">cxx::Ref_ptr</a> with default reference counting . . . . .	969
<a href="#">cxx::Ref_ptr&lt; T, CNT &gt;</a>	A reference-counting pointer with automatic cleanup . . . . .	971
<a href="#">cxx::S_list&lt; T, POLICY &gt;</a>	Simple single-linked list . . . . .	976
<a href="#">cxx::Slab&lt; Type, Slab_size, Max_free, Alloc &gt;</a>	Slab allocator for object of type <code>Type</code> . . . . .	979
<a href="#">cxx::Slab_static&lt; Type, Slab_size, Max_free, Alloc &gt;</a>	Merged slab allocator (allocators for objects of the same size are merged together) . . . . .	984
<a href="#">cxx::static_vector&lt; T, IDX &gt;</a>	Simple encapsulation for a dynamically allocated array . . . . .	988
<a href="#">cxx::String</a>	Allocation free string class with explicit length field . . . . .	989
<a href="#">cxx::Weak_ref&lt; T &gt;</a>	Typed weak reference to an object of type <code>T</code> . . . . .	996
<a href="#">cxx::Weak_ref_base</a>	Generic (base) weak reference to some object . . . . .	1000
<a href="#">cxx::Weak_ref_base::List</a>	The list type for keeping all weak references to an object . . . . .	1003
<a href="#">Elf32_Auxv</a>	Auxiliary vector (32-bit) . . . . .	1007
<a href="#">Elf32_Dyn</a>	ELF32 dynamic entry . . . . .	1008
<a href="#">Elf32_Ehdr</a>	ELF32 header . . . . .	1009
<a href="#">Elf32_Phdr</a>	ELF32 program header . . . . .	1012
<a href="#">Elf32_Rel</a>	ELF32 relocation entry w/o addend . . . . .	1013

<a href="#">Elf32_Rela</a>	ELF32 relocation entry w/ addend . . . . .	1014
<a href="#">Elf32_Shdr</a>	ELF32 section header . . . . .	1015
<a href="#">Elf32_Sym</a>	ELF32 symbol table entry . . . . .	1016
<a href="#">Elf64_Auxv</a>	Auxiliary vector (64-bit) . . . . .	1017
<a href="#">Elf64_Dyn</a>	ELF64 dynamic entry . . . . .	1018
<a href="#">Elf64_Ehdr</a>	ELF64 header . . . . .	1019
<a href="#">Elf64_Phdr</a>	ELF64 program header . . . . .	1022
<a href="#">Elf64_Rel</a>	ELF64 relocation entry w/o addend . . . . .	1023
<a href="#">Elf64_Rela</a>	ELF64 relocation entry w/ addend . . . . .	1024
<a href="#">Elf64_Shdr</a>	ELF64 section header . . . . .	1025
<a href="#">Elf64_Sym</a>	ELF64 symbol table entry . . . . .	1026
<a href="#">gfxbitmap_offset</a>	Offsets in pmap[] and bmap[] . . . . .	1027
<a href="#">L4::Alloc_list</a>	A simple list-based allocator . . . . .	1028
<a href="#">L4::Arm_smccc</a>	Wrapper for function calls that follow the ARM SMC/HVC calling convention . . . . .	1029
<a href="#">L4::Base_exception</a>	Base class for all exceptions, thrown by the <a href="#">L4Re</a> framework . . . . .	1030
<a href="#">L4::Basic_registry</a>	This registry returns the corresponding server object based on the label of an <a href="#">lpc_gate</a> . . . . .	1033
<a href="#">L4::Bounds_error</a>	Access out of bounds . . . . .	1035
<a href="#">L4::Cap&lt; T &gt;</a>	C++ interface for capabilities . . . . .	1038
<a href="#">L4::Cap_base</a>	Base class for all kinds of capabilities . . . . .	1044
<a href="#">L4::Com_error</a>	Error conditions during IPC . . . . .	1058
<a href="#">L4::Debugger</a>	C++ kernel debugger API . . . . .	1061
<a href="#">L4::Element_already_exists</a>	<a href="#">Exception</a> for duplicate element insertions . . . . .	1070
<a href="#">L4::Element_not_found</a>	<a href="#">Exception</a> for a failed lookup (element not found) . . . . .	1072
<a href="#">L4::Epiface</a>	Base class for interface implementations . . . . .	1075
<a href="#">L4::Epiface_t&lt; Derived, IFACE, BASE, bool &gt;</a>	<a href="#">Epiface</a> implementation for Kobject-based interface implementations . . . . .	1080
<a href="#">L4::Epiface_t0&lt; RPC_IFACE, BASE &gt;</a>	<a href="#">Epiface</a> mixin for generic Kobject-based interfaces . . . . .	1084
<a href="#">L4::Exception</a>	<a href="#">Exception</a> interface . . . . .	1087
<a href="#">L4::Exception_tracer</a>	Back-trace support for exceptions . . . . .	1089
<a href="#">L4::Factory</a>	C++ Factory interface, see <a href="#">Factory</a> for the C interface . . . . .	1090



<a href="#">L4::Factory::Lstr</a>	Special type to add a pascal string into the factory create stream . . . . .	1101
<a href="#">L4::Factory::Nil</a>	Special type to add a void argument into the factory create stream . . . . .	1103
<a href="#">L4::Factory::S</a>	Stream class for the <a href="#">create()</a> argument stream . . . . .	1103
<a href="#">L4::lcu</a>	C++ <a href="#">lcu</a> interface, see <a href="#">Interrupt controller</a> for the C interface . . . . .	1109
<a href="#">L4::lcu::Info</a>	This class encapsulates information about an ICU . . . . .	1119
<a href="#">L4::Invalid_capability</a>	Indicates that an invalid object was invoked . . . . .	1120
<a href="#">L4::lo_pager</a>	<a href="#">lo_pager</a> interface . . . . .	1124
<a href="#">L4::lommu</a>	Interface for IO-MMUs used for DMA remapping . . . . .	1127
<a href="#">L4::IOModifier</a>	Modifier class for the IO stream . . . . .	1132
<a href="#">L4::lpc::Array&lt; ELEM_TYPE, LEN_TYPE &gt;</a>	Array data type for dynamically sized arrays in RPCs . . . . .	1132
<a href="#">L4::lpc::Array_in_buf&lt; ELEM_TYPE, LEN_TYPE, MAX &gt;</a>	Server-side copy in buffer for <a href="#">Array</a> . . . . .	1135
<a href="#">L4::lpc::Array_ref&lt; ELEM_TYPE, LEN_TYPE &gt;</a>	Array reference data type for arrays located in the message . . . . .	1137
<a href="#">L4::lpc::As_value&lt; T &gt;</a>	Pass the argument as plain data value . . . . .	1138
<a href="#">L4::lpc::Call</a>	RPC attribute for a standard RPC call . . . . .	1139
<a href="#">L4::lpc::Call_t&lt; RIGHTS &gt;</a>	RPC attribute for an RPC call with required rights . . . . .	1140
<a href="#">L4::lpc::Call_zero_send_timeout</a>	RPC attribute for an RPC call, with zero send timeout . . . . .	1141
<a href="#">L4::lpc::Cap&lt; T &gt;</a>	Capability type for RPC interfaces (see <a href="#">L4::Cap&lt;T&gt;</a> ) . . . . .	1143
<a href="#">L4::lpc::Gen_fpage</a>	Generic RPC base for typed message items . . . . .	1146
<a href="#">L4::lpc::In_out&lt; T &gt;</a>	Mark an argument as in-out argument . . . . .	1149
<a href="#">L4::lpc::lostream</a>	Input/Output stream for IPC [un]marshalling . . . . .	1150
<a href="#">L4::lpc::lstream</a>	Input stream for IPC unmarshalling . . . . .	1161
<a href="#">L4::lpc::Msg::Cls_buffer</a>	Marker type for receive buffer values . . . . .	1173
<a href="#">L4::lpc::Msg::Cls_data</a>	Marker type for data values . . . . .	1174
<a href="#">L4::lpc::Msg::Cls_item</a>	Marker type for item values . . . . .	1175
<a href="#">L4::lpc::Msg::Dir_in</a>	Marker type for input values . . . . .	1176
<a href="#">L4::lpc::Msg::Dir_out</a>	Marker type for output values . . . . .	1177
<a href="#">L4::lpc::Msg::Do_in_data</a>	Marker for Input data . . . . .	1178
<a href="#">L4::lpc::Msg::Do_in_items</a>	Marker for Input items . . . . .	1179
<a href="#">L4::lpc::Msg::Do_out_data</a>	Marker for Output data . . . . .	1180

<a href="#">L4::lpc::Msg::Do_out_items</a>	
Marker for Output items . . . . .	1181
<a href="#">L4::lpc::Msg::Do_rcv_buffers</a>	
Marker for receive buffers . . . . .	1182
<a href="#">L4::lpc::Msg::Elem&lt; Array&lt; A, LEN &gt; &amp; &gt;</a>	
Array as output argument . . . . .	1184
<a href="#">L4::lpc::Msg::Elem&lt; Array&lt; A, LEN &gt; &gt;</a>	
Array as input arguments . . . . .	1185
<a href="#">L4::lpc::Msg::Elem&lt; Array_ref&lt; A, LEN &gt; &amp; &gt;</a>	
Array_ref as output argument . . . . .	1186
<a href="#">L4::lpc::Msg::False</a>	
False meta value . . . . .	1187
<a href="#">L4::lpc::Msg::Is_valid_rpc_type&lt; T &gt;</a>	
Type trait defining a valid RPC parameter type . . . . .	1189
<a href="#">L4::lpc::Msg::Svr_arg_pack&lt; IPC_TYPE &gt;</a>	
Server-side RPC arguments data structure used to provide arguments to the server-side implementation of an RPC function . . . . .	1191
<a href="#">L4::lpc::Msg::Svr_val_ops&lt; MTYPE, DIR, CLASS &gt;</a>	
Defines client-side handling of 'MTYPE' as RPC argument . . . . .	1191
<a href="#">L4::lpc::Msg::True</a>	
True meta value . . . . .	1193
<a href="#">L4::lpc::Msg_ptr&lt; T &gt;</a>	
Pointer to an element of type T in an <a href="#">lpc::lstream</a> . . . . .	1196
<a href="#">L4::lpc::Opt&lt; T &gt;</a>	
Attribute for defining an optional RPC argument . . . . .	1197
<a href="#">L4::lpc::Ostream</a>	
Output stream for IPC marshalling . . . . .	1199
<a href="#">L4::lpc::Out&lt; T &gt;</a>	
Mark an argument as a output value in an RPC signature . . . . .	1205
<a href="#">L4::lpc::Rcv_fpage</a>	
Non-small receive item . . . . .	1206
<a href="#">L4::lpc::Ret_array&lt; T &gt;</a>	
Dynamically sized output array of type T . . . . .	1212
<a href="#">L4::lpc::Send_only</a>	
RPC attribute for a send-only RPC . . . . .	1213
<a href="#">L4::lpc::Small_buf</a>	
A receive item for receiving a single object capability . . . . .	1214
<a href="#">L4::lpc::Snd_fpage</a>	
Send item or return item . . . . .	1215
<a href="#">L4::lpc::Str_cp_in&lt; T &gt;</a>	
Abstraction for extracting a zero-terminated string from an <a href="#">lpc::lstream</a> . . . . .	1226
<a href="#">L4::lpc::Varg</a>	
Variably sized RPC argument . . . . .	1227
<a href="#">L4::lpc::Varg_list&lt; MAX &gt;</a>	
Self-contained list of variable-sized RPC parameters . . . . .	1233
<a href="#">L4::lpc::Varg_list_ref</a>	
List of variable-sized RPC parameters as received by the server . . . . .	1235
<a href="#">L4::lpc::Varg_list_ref::Iterator</a>	
Iterator for Valists . . . . .	1238
<a href="#">L4::lpc_gate</a>	
The C++ IPC gate interface, see <a href="#">IPC-Gate API</a> for the C interface . . . . .	1239
<a href="#">L4::lpc_svr::Br_manager_no_buffers</a>	
Empty implementation of <a href="#">Server_iface</a> . . . . .	1245
<a href="#">L4::lpc_svr::Compound_reply</a>	
Mix in for LOOP_HOOKS to always use compound reply and wait . . . . .	1249
<a href="#">L4::lpc_svr::Dbg_dispatch&lt; R, Exc, Printer &gt;</a>	
Dispatch helper that, in addition to what <a href="#">Exc_dispatch</a> does, prints exception messages . . . . .	1251

<a href="#">L4::lpc_svr::Default_loop_hooks</a>	
Default LOOP_HOOKS . . . . .	1253
<a href="#">L4::lpc_svr::Default_setup_wait</a>	
Mix in for LOOP_HOOKS for setup_wait no op . . . . .	1255
<a href="#">L4::lpc_svr::Default_timeout</a>	
Mix in for LOOP_HOOKS to use a 0 send and a infinite receive timeout . . . . .	1256
<a href="#">L4::lpc_svr::Direct_dispatch&lt; R &gt;</a>	
Direct dispatch helper, for forwarding dispatch calls to a registry <i>R</i> . . . . .	1258
<a href="#">L4::lpc_svr::Direct_dispatch&lt; R * &gt;</a>	
Direct dispatch helper, for forwarding dispatch calls via a pointer to a registry <i>R</i> . . . . .	1260
<a href="#">L4::lpc_svr::Exc_dispatch&lt; R, Exc &gt;</a>	
Dispatch helper wrapping try {} catch {} around the dispatch call . . . . .	1262
<a href="#">L4::lpc_svr::Ignore_errors</a>	
Mix in for LOOP_HOOKS to ignore IPC errors . . . . .	1263
<a href="#">L4::lpc_svr::Server_iface</a>	
Interface for server-loop related functions . . . . .	1265
<a href="#">L4::lpc_svr::Timeout</a>	
Callback interface for <a href="#">Timeout_queue</a> . . . . .	1273
<a href="#">L4::lpc_svr::Timeout_queue</a>	
Timeout queue to be used in <a href="#">L4re</a> server loop . . . . .	1277
<a href="#">L4::lpc_svr::Timeout_queue_hooks&lt; HOOKS, BR_MAN &gt;</a>	
Loop hooks mixin for integrating a timeout queue into the server loop . . . . .	1280
<a href="#">L4::lrq</a>	
C++ <a href="#">lrq</a> interface, see <a href="#">IRQs</a> for the C interface . . . . .	1286
<a href="#">L4::lrq_eoi</a>	
Interface for sending an unmask message to an object . . . . .	1296
<a href="#">L4::lrq_handler_object</a>	
Server object base class for handling IRQ messages . . . . .	1298
<a href="#">L4::lrqep_t&lt; Derived, BASE, bool &gt;</a>	
Epiface implementation for interrupt handlers . . . . .	1302
<a href="#">L4::Kip::Mem_desc</a>	
Memory descriptors stored in the kernel interface page . . . . .	1306
<a href="#">L4::Kobject</a>	
Base class for all kinds of kernel objects and remote objects, referenced by capabilities . . . . .	1318
<a href="#">L4::Kobject_2t&lt; Derived, Base1, Base2, PROTO, S_DEMAND &gt;</a>	
Helper class to create an <a href="#">L4Re</a> interface class that is derived from two base classes (see <a href="#">L4::Kobject_t</a> ) . . . . .	1322
<a href="#">L4::Kobject_3t&lt; Derived, Base1, Base2, Base3, PROTO, S_DEMAND &gt;</a>	
Helper class to create an <a href="#">L4Re</a> interface class that is derived from three base classes (see <a href="#">L4::Kobject_t</a> ) . . . . .	1326
<a href="#">L4::Kobject_demand&lt; T &gt;</a>	
Get the combined server-side resource requirements for all type <i>T</i> . . . . .	1329
<a href="#">L4::Kobject_t&lt; Derived, Base, PROTO, S_DEMAND &gt;</a>	
Helper class to create an <a href="#">L4Re</a> interface class that is derived from a single base class . . . . .	1330
<a href="#">L4::Kobject_typeid&lt; T &gt;</a>	
Meta object for handling access to type information of Kobjects . . . . .	1332
<a href="#">L4::Kobject_typeid&lt; void &gt;</a>	
Minimalistic ID for <code>void</code> interface . . . . .	1335
<a href="#">L4::Kobject_x&lt; Derived, ARGS &gt;</a>	
Generic <a href="#">Kobject</a> inheritance template . . . . .	1338
<a href="#">L4::Lock_guard</a>	
Basic lock guard implementation that prevents forgotten unlocks on exit paths from a method or a block of code . . . . .	1339
<a href="#">L4::Meta</a>	
Meta interface that shall be implemented by each <a href="#">L4Re</a> object and gives access to the dynamic type information for <a href="#">L4Re</a> objects . . . . .	1342
<a href="#">L4::Out_of_memory</a>	
Exception signalling insufficient memory . . . . .	1347

<a href="#">L4::Pager</a>	
<a href="#">Pager</a> interface including the <a href="#">lo_pager</a> interface	1351
<a href="#">L4::Platform_control</a>	
C++ interface for controlling platform-wide properties, see <a href="#">Platform Control C API</a> for the C interface	1354
<a href="#">L4::Poll_timeout_counter</a>	
Evaluate an expression for a maximum number of times	1362
<a href="#">L4::Poll_timeout_kipclock</a>	
A polling timeout based on the <a href="#">L4Re</a> clock	1364
<a href="#">L4::Proto_t&lt; P &gt;</a>	
Data type for defining protocol numbers	1368
<a href="#">L4::Rcv_endpoint</a>	
Interface for kernel objects that allow to receive IPC from them	1368
<a href="#">L4::Registry_iface</a>	
Abstract interface for object registries	1374
<a href="#">L4::Runtime_error</a>	
Exception for an abstract runtime error	1378
<a href="#">L4::Scheduler</a>	
C++ interface of the <a href="#">Scheduler</a> kernel object, see <a href="#">Scheduler</a> for the C interface	1383
<a href="#">L4::Semaphore</a>	
C++ Kernel-provided semaphore interface, see <a href="#">Kernel-provided semaphore</a> for the C interface	1392
<a href="#">L4::Server&lt; LOOP_HOOKS &gt;</a>	
Basic server loop for handling client requests	1398
<a href="#">L4::Server_object</a>	
Abstract server object to be used with <a href="#">L4::Server</a> and <a href="#">L4::Basic_registry</a>	1404
<a href="#">L4::Server_object_t&lt; IFACE, BASE &gt;</a>	
Base class (template) for server implementing server objects	1409
<a href="#">L4::Server_object_x&lt; Derived, IFACE, BASE &gt;</a>	
Helper class to implement p_dispatch based server objects	1415
<a href="#">L4::Smart_cap&lt; T, SMART &gt;</a>	
Smart capability class	1418
<a href="#">L4::String</a>	
A null-terminated string container class	1422
<a href="#">L4::Task</a>	
C++ interface of the <a href="#">Task</a> kernel object, see <a href="#">Task</a> for the C interface	1423
<a href="#">L4::Thread</a>	
C++ <a href="#">L4</a> kernel thread interface, see <a href="#">Thread</a> for the C interface	1435
<a href="#">L4::Thread::Attr</a>	
<a href="#">Thread</a> attributes used for <a href="#">control()</a>	1451
<a href="#">L4::Thread::Modify_senders</a>	
Class wrapping a list of rules which modify the sender label of IPC messages inbound to this thread	1455
<a href="#">L4::Thread_group</a>	
C++ <a href="#">L4</a> kernel thread group interface, see <a href="#">Thread groups</a> for the C interface	1458
<a href="#">L4::Triggerable</a>	
Interface that allows an object to be triggered by some source	1462
<a href="#">L4::Type_info</a>	
Dynamic Type Information for <a href="#">L4Re</a> Interfaces	1466
<a href="#">L4::Type_info::Demand</a>	
Data type for expressing the needed receive buffers at the server-side of an interface	1467
<a href="#">L4::Type_info::Demand_t&lt; CAPS, FLAGS, MEM, PORTS &gt;</a>	
Template type statically describing demand of receive buffers	1471
<a href="#">L4::Type_info::Demand_union_t&lt; D1, D2 &gt;</a>	
Template type statically describing the combination of two <a href="#">Demand</a> object	1474
<a href="#">L4::Typeid::Detail::_Rpc&lt; OPCODE, O, X &gt;</a>	
Empty list of RPCs	1478
<a href="#">L4::Typeid::Detail::_Rpc&lt; OPCODE, O, Default_op&lt; R &gt; &gt;::Rpc&lt; Y &gt;</a>	
Find the given RPC in the list	1479

<a href="#">L4::Typeid::Detail::_Rpc&lt; OPCODE, O, R, X... &gt;</a>	
Non-empty list of RPCs	1479
<a href="#">L4::Typeid::Detail::_Rpc&lt; OPCODE, O, R, X... &gt;::Rpc&lt; Y &gt;</a>	
Find the given RPC in the list	1482
<a href="#">L4::Typeid::Detail::Rpc_end</a>	
Internal end-of-list marker	1483
<a href="#">L4::Typeid::P_dispatch&lt; LIST &gt;</a>	
Use for protocol based dispatch stage	1483
<a href="#">L4::Typeid::Raw_ipc&lt; CLASS &gt;</a>	
RPCs list for passing raw incoming IPC to the server object	1484
<a href="#">L4::Typeid::Rpc_nocode&lt; OPERATION &gt;</a>	
List of RPCs of an interface using a single operation without an opcode	1485
<a href="#">L4::Typeid::Rpc&lt; RPCS &gt;</a>	
Standard list of RPCs of an interface	1488
<a href="#">L4::Typeid::Rpc_code&lt; OPCODE_TYPE &gt;</a>	
List of RPCs of an interface using a special opcode type	1489
<a href="#">L4::Typeid::Rpc_code&lt; OPCODE_TYPE &gt;::F&lt; RPCS &gt;</a>	1490
<a href="#">L4::Typeid::Rpc_sys&lt; ARG &gt;</a>	
List of RPCs typically used for kernel interfaces	1493
<a href="#">L4::Types::Bool&lt; V &gt;</a>	
Boolean meta type	1495
<a href="#">L4::Types::False</a>	
False meta value	1497
<a href="#">L4::Types::Flags&lt; BITS_ENUM, UNDERLYING &gt;</a>	
Template for defining typical <a href="#">Flags</a> bitmaps	1499
<a href="#">L4::Types::Flags_ops_t&lt; DT &gt;</a>	
Mixin class to define a set of friend bitwise operators on DT	1503
<a href="#">L4::Types::Flags_t&lt; DT, T &gt;</a>	
Template type to define a flags type with bitwise operations	1506
<a href="#">L4::Types::Int_for_size&lt; SIZE, bool &gt;</a>	
Metafunction to get an unsigned integral type for the given size	1508
<a href="#">L4::Types::Int_for_type&lt; T &gt;</a>	
Metafunction to get an integral type of the same size as T	1509
<a href="#">L4::Types::Same&lt; A, B &gt;</a>	
Compare two data types for equality	1510
<a href="#">L4::Types::True</a>	
True meta value	1513
<a href="#">L4::Uart</a>	
Uart driver abstraction	1516
<a href="#">L4::Uart_apb</a>	
Driver for the Advanced Peripheral Bus (APB) UART from the Cortex-M System Design Kit (CMSDK)	1521
<a href="#">L4::Unknown_error</a>	
Exception for an unknown condition	1527
<a href="#">L4::Vcon</a>	
C++ <a href="#">L4 Vcon</a> interface, see <a href="#">Virtual Console</a> for the C interface	1529
<a href="#">L4::Vm</a>	
Virtual machine host address space	1539
<a href="#">l4_buf_regs_t</a>	
Encapsulation of the buffer-registers block in the UTCB	1544
<a href="#">l4_exc_regs_t</a>	
UTCB structure for exceptions	1545
<a href="#">l4_fpage_t</a>	
<a href="#">L4</a> flexpage type	1548
<a href="#">l4_icu_info_t</a>	
Info structure for an ICU	1549
<a href="#">l4_icu_msi_info_t</a>	
Info to use for a specific MSI	1550

<a href="#">l4_kernel_info_mem_desc_t</a>	Memory descriptor data structure . . . . .	1551
<a href="#">l4_kernel_info_t</a>	L4 Kernel Interface Page . . . . .	1552
<a href="#">l4_msg_regs_t</a>	Encapsulation of the message-register block in the UTCB . . . . .	1553
<a href="#">l4_msgtag_t</a>	Message tag data structure . . . . .	1554
<a href="#">l4_sched_cpu_set_t</a>	CPU sets . . . . .	1557
<a href="#">l4_sched_param_t</a>	Scheduler parameter set . . . . .	1561
<a href="#">l4_snd_fpage_t</a>	Send-flexpage types . . . . .	1562
<a href="#">l4_thread_regs_t</a>	Encapsulation of the thread-control-register block of the UTCB . . . . .	1563
<a href="#">l4_timeout_s</a>	Basic timeout specification . . . . .	1564
<a href="#">l4_timeout_t</a>	Timeout pair . . . . .	1565
<a href="#">l4_vcon_attr_t</a>	Vcon attribute structure . . . . .	1566
<a href="#">l4_vcpu_arch_state_t</a>	Architecture-specific vCPU state . . . . .	1568
<a href="#">l4_vcpu_ipc_regs_t</a>	vCPU message registers . . . . .	1568
<a href="#">l4_vcpu_regs_t</a>	vCPU registers . . . . .	1569
<a href="#">l4_vcpu_state_t</a>	State of a vCPU . . . . .	1573
<a href="#">l4_vm_svm_vmcb_control_area</a>	VMCB structure for SVM VMs . . . . .	1577
<a href="#">l4_vm_svm_vmcb_state_save_area</a>	State save area structure for SVM VMs . . . . .	1577
<a href="#">l4_vm_svm_vmcb_state_save_area_seg</a>	State save area segment selector struct . . . . .	1578
<a href="#">l4_vm_svm_vmcb_t</a>	Control structure for SVM VMs . . . . .	1579
<a href="#">l4_vm_tz_state</a>	State structure for TrustZone VMs . . . . .	1581
<a href="#">l4_vm_vmx_vcpu_infos_t</a>	VMX information members . . . . .	1581
<a href="#">l4_vm_vmx_vcpu_state_t</a>	VMX vCPU state . . . . .	1582
<a href="#">l4_vm_vmx_vcpu_vmcs_t</a>	VMX software VMCS . . . . .	1584
<a href="#">l4_vmx_offset_table_t</a>	Software VMCS field offset table . . . . .	1585
<a href="#">L4drivers::Mmio_register_block&lt; MAX_BITS &gt;</a>	An MMIO block with up to 64-bit register access (32-bit default) and little endian byte order . . . . .	1587
<a href="#">L4drivers::Register_block&lt; MAX_BITS, BLOCK &gt;</a>	Handles a reference to a register block of the given maximum access width . . . . .	1589
<a href="#">L4drivers::Register_block_base&lt; MAX_BITS &gt;</a>	Abstract register block interface . . . . .	1593
<a href="#">L4drivers::Register_block_impl&lt; BASE, MAX_BITS &gt;</a>	Implementation helper for register blocks . . . . .	1594
<a href="#">L4drivers::Register_block_tmpl&lt; BLOCK &gt;</a>	Helper template that translates to the <a href="#">Register_block_base</a> interface . . . . .	1596

<a href="#">L4drivers::Register_tmpl&lt; BITS, BLOCK &gt;</a>	
Single hardware register inside a <a href="#">Register_block_base</a> interface	1597
<a href="#">L4drivers::Ro_register_block&lt; MAX_BITS, BLOCK &gt;</a>	
Handles a reference to a read only register block of the given maximum access width	1604
<a href="#">L4drivers::Ro_register_tmpl&lt; BITS, BLOCK &gt;</a>	
Single read only register inside a <a href="#">Register_block_base</a> interface	1606
<a href="#">L4Re::Cap_alloc</a>	
Capability allocator interface	1608
<a href="#">L4Re::Console</a>	
Console class	1611
<a href="#">L4Re::Core::Ref_ptr&lt; T, CNT &gt;</a>	
A reference-counting pointer with automatic cleanup	1614
<a href="#">L4Re::Dataspace</a>	
Interface for memory-like objects	1618
<a href="#">L4Re::Dataspace::F</a>	
Dataspace flags definitions	1631
<a href="#">L4Re::Dataspace::Stats</a>	
Information about the dataspace	1633
<a href="#">L4Re::Debug_obj</a>	
Debug interface	1634
<a href="#">L4Re::Default_event_payload</a>	
Default event stream payload	1637
<a href="#">L4Re::Dma_space</a>	
Managed DMA Address Space	1638
<a href="#">L4Re::Env</a>	
C++ interface of the initial environment that is provided to an <a href="#">L4</a> task	1645
<a href="#">L4Re::Event</a>	
Event class	1660
<a href="#">L4Re::Event_buffer_t&lt; PAYLOAD &gt;</a>	
Event buffer class	1669
<a href="#">L4Re::Event_buffer_t&lt; PAYLOAD &gt;::Event</a>	
Event structure used in buffer	1672
<a href="#">L4Re::Inhibitor</a>	
Set of inhibitor locks, which inhibit specific actions when held	1673
<a href="#">L4Re::Itas</a>	
Interface to the ITAS	1678
<a href="#">L4Re::Log</a>	
Log interface class	1689
<a href="#">L4Re::Mem_alloc</a>	
Memory allocation interface	1695
<a href="#">L4Re::Mem_alloc::Stats</a>	
Statistics about memory-allocator	1701
<a href="#">L4Re::Mmio_space</a>	
Interface for memory-like address space accessible via IPC	1703
<a href="#">L4Re::Namespace</a>	
Name-space interface	1710
<a href="#">L4Re::Ned::Cmd_control</a>	
Direct control interface for Ned	1720
<a href="#">L4Re::Parent</a>	
Parent interface	1722
<a href="#">L4Re::Random</a>	
Low-bandwidth interface for random number generators	1727
<a href="#">L4Re::Rm</a>	
Region map	1731
<a href="#">L4Re::Rm::Area</a>	
An area is a range of virtual addresses which is reserved, see <a href="#">L4Re::Rm::reserve_area()</a>	1755
<a href="#">L4Re::Rm::F</a>	
Rm flags definitions	1756

<a href="#">L4Re::Rm::Region</a>	
A region is a range of virtual addresses which is backed by content . . . . .	1758
<a href="#">L4Re::Rm::Unique_region&lt; T &gt;</a>	
Unique region . . . . .	1759
<a href="#">L4Re::Smart_cap_auto&lt; Unmap_flags &gt;</a>	
Helper for <a href="#">Unique_cap</a> and <a href="#">Unique_del_cap</a> . . . . .	1764
<a href="#">L4Re::Smart_count_cap&lt; Unmap_flags &gt;</a>	
Helper for <a href="#">Ref_cap</a> and <a href="#">Ref_del_cap</a> . . . . .	1765
<a href="#">L4Re::Util::_Cap_alloc</a>	
Adapter to expose the cap allocator implementation as <a href="#">L4Re::Cap_alloc</a> compatible class . . .	1766
<a href="#">L4Re::Util::Bitmap&lt; BITS &gt;</a>	
A static bitmap . . . . .	1769
<a href="#">L4Re::Util::Bitmap_base</a>	
Basic bitmap abstraction . . . . .	1773
<a href="#">L4Re::Util::Bitmap_base::Bit</a>	
A writable bit in a bitmap . . . . .	1782
<a href="#">L4Re::Util::Bitmap_base::Char&lt; BITS &gt;</a>	
Helper abstraction for a byte contained in the bitmap . . . . .	1782
<a href="#">L4Re::Util::Bitmap_base::Word&lt; BITS &gt;</a>	
Helper abstraction for a word contained in the bitmap . . . . .	1783
<a href="#">L4Re::Util::Br_manager</a>	
Buffer-register (BR) manager for <a href="#">L4::Server</a> . . . . .	1784
<a href="#">L4Re::Util::Br_manager_hooks</a>	
Predefined server-loop hooks for a server loop using the <a href="#">Br_manager</a> . . . . .	1790
<a href="#">L4Re::Util::Br_manager_timeout_hooks</a>	
Predefined server-loop hooks for a server with using the <a href="#">Br_manager</a> and a timeout queue . .	1792
<a href="#">L4Re::Util::Cap_alloc_base</a>	
Capability allocator . . . . .	1796
<a href="#">L4Re::Util::Counter&lt; COUNTER &gt;</a>	
Counter for <a href="#">Counting_cap_alloc</a> with variable data width . . . . .	1797
<a href="#">L4Re::Util::Counter_atomic&lt; COUNTER &gt;</a>	
Thread safe version of counter for <a href="#">Counting_cap_alloc</a> . . . . .	1799
<a href="#">L4Re::Util::Counting_cap_alloc&lt; COUNTERTYPE, Dbg &gt;</a>	
Internal reference-counting cap allocator . . . . .	1800
<a href="#">L4Re::Util::Dataspace_svr</a>	
Dataspace server class . . . . .	1806
<a href="#">L4Re::Util::Event_buffer_consumer_t&lt; PAYLOAD &gt;</a>	
An event buffer consumer . . . . .	1813
<a href="#">L4Re::Util::Event_buffer_t&lt; PAYLOAD &gt;</a>	
Event_buffer utility class . . . . .	1816
<a href="#">L4Re::Util::Event_svr&lt; SVR &gt;</a>	
Convenience wrapper for implementing an event server . . . . .	1820
<a href="#">L4Re::Util::Event_t&lt; PAYLOAD &gt;</a>	
Convenience wrapper for getting access to an event object . . . . .	1822
<a href="#">L4Re::Util::Item_alloc_base</a>	
Item allocator . . . . .	1827
<a href="#">L4Re::Util::Names::Name</a>	
Name class . . . . .	1827
<a href="#">L4Re::Util::Names::Name_space</a>	
Abstract server-side implementation of the <a href="#">L4::Namespace</a> interface . . . . .	1831
<a href="#">L4Re::Util::Object_registry</a>	
A registry that manages server objects and their attached IPC gates for a single server loop for a specific thread . . . . .	1836
<a href="#">L4Re::Util::Ref_cap&lt; T &gt;</a>	
Automatic capability that implements automatic free and unmap of the capability selector . . .	1842
<a href="#">L4Re::Util::Ref_del_cap&lt; T &gt;</a>	
Automatic capability that implements automatic free and unmap+delete of the capability selector	1843



<a href="#">L4Re::Util::Registry_server&lt; LOOP_HOOKS &gt;</a>	
A server loop object which has a <a href="#">Object_registry</a> included	1844
<a href="#">L4Re::Util::Smart_cap_auto&lt; Unmap_flags &gt;</a>	
Helper for <a href="#">Unique_cap</a> and <a href="#">Unique_del_cap</a>	1850
<a href="#">L4Re::Util::Smart_count_cap&lt; Unmap_flags &gt;</a>	
Helper for <a href="#">Ref_cap</a> and <a href="#">Ref_del_cap</a>	1851
<a href="#">L4Re::Util::Vcon_svr&lt; SVR &gt;</a>	
Console server template class	1852
<a href="#">L4Re::Util::Video::Goos_svr</a>	
Goos server class	1853
<a href="#">L4Re::Vfs::Be_file</a>	
Boiler plate class for implementing an open file for <a href="#">L4Re::Vfs</a>	1858
<a href="#">L4Re::Vfs::Be_file_system</a>	
Boilerplate class for implementing a <a href="#">L4Re::Vfs::File_system</a>	1863
<a href="#">L4Re::Vfs::Directory</a>	
Interface for a POSIX file that is a directory	1865
<a href="#">L4Re::Vfs::File</a>	
The basic interface for an open POSIX file	1871
<a href="#">L4Re::Vfs::File_system</a>	
Basic interface for an <a href="#">L4Re::Vfs</a> file system	1874
<a href="#">L4Re::Vfs::Fs</a>	
POSIX File-system related functionality	1876
<a href="#">L4Re::Vfs::Generic_file</a>	
The common interface for an open POSIX file	1881
<a href="#">L4Re::Vfs::Mman</a>	
Interface for POSIX memory management	1890
<a href="#">L4Re::Vfs::Ops</a>	
Interface for the POSIX backends of an application	1891
<a href="#">L4Re::Vfs::Regular_file</a>	
Interface for a POSIX file that provides regular file semantics	1894
<a href="#">L4Re::Vfs::Special_file</a>	
Interface for a POSIX file that provides special file semantics	1904
<a href="#">L4Re::Video::Color_component</a>	
A color component	1907
<a href="#">L4Re::Video::Goos</a>	
Class that abstracts framebuffers	1912
<a href="#">L4Re::Video::Goos::Info</a>	
Information structure of a <a href="#">Goos</a>	1922
<a href="#">L4Re::Video::Pixel_info</a>	
Pixel information	1924
<a href="#">L4Re::Video::View</a>	
View of a framebuffer	1935
<a href="#">L4Re::Video::View::Info</a>	
Information structure of a view	1941
<a href="#">l4re_aux_t</a>	
Auxiliary descriptor	1944
<a href="#">l4re_ds_stats_t</a>	
Information about the data space	1945
<a href="#">l4re_elf_aux_mword_t</a>	
Auxiliary vector element for a single unsigned data word	1945
<a href="#">l4re_elf_aux_t</a>	
Generic header for each auxiliary vector element	1946
<a href="#">l4re_elf_aux_vma_t</a>	
Auxiliary vector element for a reserved virtual memory area	1946
<a href="#">l4re_env_cap_entry_t</a>	
Entry in the <a href="#">L4Re</a> environment array for the named initial objects	1947
<a href="#">l4re_env_t</a>	
Initial environment data structure	1949

<a href="#">l4re_event_t</a>	Event structure used in buffer . . . . .	1951
<a href="#">l4re_video_color_component_t</a>	Color component structure . . . . .	1952
<a href="#">l4re_video_goos_info_t</a>	Goos information structure . . . . .	1952
<a href="#">l4re_video_pixel_info_t</a>	Pixel_info structure . . . . .	1954
<a href="#">l4re_video_view_info_t</a>	View information structure . . . . .	1955
<a href="#">l4re_video_view_t</a>	C representation of a goos view . . . . .	1956
<a href="#">l4shmc_ringbuf_head_t</a>	Head field of a ring buffer . . . . .	1957
<a href="#">l4shmc_ringbuf_t</a>	Ring buffer . . . . .	1958
<a href="#">l4util_l4mod_info</a>	Base module structure . . . . .	1959
<a href="#">l4util_l4mod_mod</a>	A single module . . . . .	1960
<a href="#">l4util_mb_addr_range_t</a>	INT-15, AX=E820 style "AddressRangeDescriptor" ...with a "size" parameter on the front which is the structure size - 4, pointing to the next one, up until the full buffer length of the memory map has been reached . . . . .	1961
<a href="#">l4util_mb_apm_t</a>	APM BIOS info . . . . .	1962
<a href="#">l4util_mb_drive_t</a>	Drive Info structure . . . . .	1963
<a href="#">l4util_mb_info_t</a>	MultiBoot Info description . . . . .	1964
<a href="#">l4util_mb_mod_t</a>	The structure type "mod_list" is used by the <a href="#">multiboot_info</a> structure . . . . .	1965
<a href="#">l4util_mb_vbe_ctrl_t</a>	VBE controller information . . . . .	1966
<a href="#">l4util_mb_vbe_mode_t</a>	VBE mode information . . . . .	1967
<a href="#">L4vbus::Device</a>	Device on a <a href="#">L4vbus::Vbus</a> . . . . .	1971
<a href="#">L4vbus::Gpio_module</a>	A <a href="#">Gpio_module</a> groups multiple GPIO pins together . . . . .	1980
<a href="#">L4vbus::Gpio_module::Pin_slice</a>	A slice of the pins provided by this module . . . . .	1988
<a href="#">L4vbus::Gpio_pin</a>	A GPIO pin . . . . .	1988
<a href="#">L4vbus::lcu</a>	<a href="#">Vbus</a> Interrupt controller API . . . . .	1997
<a href="#">L4vbus::Pci_dev</a>	A PCI device . . . . .	2002
<a href="#">L4vbus::Pci_host_bridge</a>	A Pci host bridge . . . . .	2008
<a href="#">L4vbus::Pm&lt; DEC &gt;</a>	Power-management API mixin . . . . .	2014
<a href="#">L4vbus::Vbus</a>	The virtual bus ( <a href="#">Vbus</a> ) interface . . . . .	2017
<a href="#">l4vbus_device_t</a>	Detailed information about a vbus device . . . . .	2028
<a href="#">l4vbus_resource_t</a>	Description of a single vbus resource . . . . .	2029

<a href="#">L4vcpu::State</a>	C++ implementation of state word in the vCPU area . . . . .	2030
<a href="#">L4vcpu::Vcpu</a>	C++ implementation of the vCPU save state area . . . . .	2032
<a href="#">L4virtio::Device</a>	IPC interface for virtio over <a href="#">L4 IPC</a> . . . . .	2045
<a href="#">L4virtio::Driver::Block_device</a>	Simple class for accessing a virtio block device synchronously . . . . .	2054
<a href="#">L4virtio::Driver::Block_device::Handle</a>	<a href="#">Handle</a> to an ongoing request . . . . .	2063
<a href="#">L4virtio::Driver::Device</a>	Client-side implementation for a general virtio device . . . . .	2063
<a href="#">L4virtio::Driver::Virtio_net_device</a>	Simple class for accessing a virtio net device . . . . .	2076
<a href="#">L4virtio::Driver::Virtio_net_device::Packet</a>	Structure for a network packet (header including data) with maximum size, assuming that no extra features have been negotiated . . . . .	2086
<a href="#">L4virtio::Driver::Virtqueue</a>	Driver-side implementation of a <a href="#">Virtqueue</a> . . . . .	2086
<a href="#">L4virtio::Ptr&lt; T &gt;</a>	Pointer used in virtio descriptors . . . . .	2097
<a href="#">L4virtio::Svr::Bad_descriptor</a>	Exception used by Queue to indicate descriptor errors . . . . .	2101
<a href="#">L4virtio::Svr::Block_dev_base&lt; Ds_data &gt;</a>	Base class for virtio block devices . . . . .	2103
<a href="#">L4virtio::Svr::Block_request&lt; Ds_data &gt;</a>	A request to read or write data . . . . .	2111
<a href="#">L4virtio::Svr::Console::Control_message</a>	Virtio console control message . . . . .	2114
<a href="#">L4virtio::Svr::Console::Control_request</a>	Specialised <a href="#">Virtqueue::Request</a> providing access to control message payload . . . . .	2115
<a href="#">L4virtio::Svr::Console::Device</a>	Base class implementing a virtio console device with L4Re-based notification handling . . . . .	2117
<a href="#">L4virtio::Svr::Console::Device_port</a>	A console port with associated read/write state . . . . .	2131
<a href="#">L4virtio::Svr::Console::Features</a>	Virtio console specific feature bits . . . . .	2135
<a href="#">L4virtio::Svr::Console::Port</a>	Representation of a Virtio console port . . . . .	2139
<a href="#">L4virtio::Svr::Console::Port::Transition</a>	State transition from last report state to current state . . . . .	2143
<a href="#">L4virtio::Svr::Console::Virtio_con</a>	Base class implementing a virtio console functionality . . . . .	2144
<a href="#">L4virtio::Svr::Data_buffer</a>	Abstract data buffer . . . . .	2161
<a href="#">L4virtio::Svr::Dev_config</a>	Abstraction for L4-Virtio device config memory . . . . .	2165
<a href="#">L4virtio::Svr::Dev_features</a>	Type for device feature bitmap . . . . .	2176
<a href="#">L4virtio::Svr::Dev_status</a>	Type of the device status register . . . . .	2179
<a href="#">L4virtio::Svr::Device_t&lt; DATA &gt;</a>	Server-side L4-VIRTIO device stub . . . . .	2182
<a href="#">L4virtio::Svr::Driver_mem_list_t&lt; DATA &gt;</a>	List of driver memory regions assigned to a single L4-VIRTIO transport instance . . . . .	2188
<a href="#">L4virtio::Svr::Driver_mem_region_t&lt; DATA &gt;</a>	Region of driver memory, that shall be managed locally . . . . .	2194

<a href="#">L4virtio::Svr::Request_processor</a>	Encapsulate the state for processing a VIRTIO request . . . . .	2201
<a href="#">L4virtio::Svr::Scmi::Base_attr_t</a>	SCMI base protocol attributes . . . . .	2208
<a href="#">L4virtio::Svr::Scmi::Base_proto</a>	Base class for the SCMI base protocol . . . . .	2209
<a href="#">L4virtio::Svr::Scmi::Perf_proto</a>	Base class for the SCMI performance protocol . . . . .	2211
<a href="#">L4virtio::Svr::Scmi::Performance_attr_t</a>	SCMI performance protocol attributes . . . . .	2214
<a href="#">L4virtio::Svr::Scmi::Performance_describe_level_t</a>	SCMI performance describe level . . . . .	2215
<a href="#">L4virtio::Svr::Scmi::Performance_describe_levels_n_t</a>	SCMI performance describe levels numbers . . . . .	2215
<a href="#">L4virtio::Svr::Scmi::Performance_domain_attr_t</a>	SCMI performance domain protocol attributes . . . . .	2217
<a href="#">L4virtio::Svr::Scmi::Proto&lt; OBSERV &gt;</a>	Base class for all protocols . . . . .	2219
<a href="#">L4virtio::Svr::Scmi::Scmi_dev</a>	A server implementation of the virtio-scmi protocol . . . . .	2220
<a href="#">L4virtio::Svr::Scmi::Scmi_hdr_t</a>	SCMI header . . . . .	2224
<a href="#">L4virtio::Svr::Virtio_gpio&lt; Request_handler, Epiface &gt;</a>	A server implementation of the virtio-gpio protocol . . . . .	2225
<a href="#">L4virtio::Svr::Virtio_gpio&lt; Request_handler, Epiface &gt;::Host_irq</a>	Handler for the host irq . . . . .	2230
<a href="#">L4virtio::Svr::Virtio_gpio&lt; Request_handler, Epiface &gt;::Irq_handler</a>	Handler for an gpio pin irq . . . . .	2234
<a href="#">L4virtio::Svr::Virtio_gpio&lt; Request_handler, Epiface &gt;::Request_processor</a>	Generic handler for the Virtio requests . . . . .	2235
<a href="#">L4virtio::Svr::Virtio_i2c&lt; Request_handler, Epiface &gt;</a>	A server implementation of the virtio-i2c protocol . . . . .	2238
<a href="#">L4virtio::Svr::Virtio_i2c&lt; Request_handler, Epiface &gt;::Host_irq</a>	Handler for the host irq . . . . .	2242
<a href="#">L4virtio::Svr::Virtio_i2c&lt; Request_handler, Epiface &gt;::Request_processor</a>	Handler for the Virtio requests . . . . .	2246
<a href="#">L4virtio::Svr::Virtio_rng&lt; Rnd_state, Epiface &gt;</a>	A server implementation of the virtio-rng protocol . . . . .	2249
<a href="#">L4virtio::Svr::Virtio_rng&lt; Rnd_state, Epiface &gt;::Host_irq</a>	Handler for the host irq . . . . .	2253
<a href="#">L4virtio::Svr::Virtio_rng&lt; Rnd_state, Epiface &gt;::Request_processor</a>	Handler for the Virtio requests . . . . .	2257
<a href="#">L4virtio::Svr::Virtqueue</a>	Virtqueue implementation for the device . . . . .	2259
<a href="#">L4virtio::Svr::Virtqueue::Head_desc</a>	VIRTIO request, essentially a descriptor from the available ring . . . . .	2272
<a href="#">L4virtio::Virtqueue</a>	Low-level Virtqueue . . . . .	2274
<a href="#">L4virtio::Virtqueue::Avail</a>	Type of available ring, this is read-only for the host . . . . .	2291
<a href="#">L4virtio::Virtqueue::Avail::Flags</a>	Flags of the available ring . . . . .	2293
<a href="#">L4virtio::Virtqueue::Desc</a>	Descriptor in the descriptor table . . . . .	2294
<a href="#">L4virtio::Virtqueue::Desc::Flags</a>	Type for descriptor flags . . . . .	2296
<a href="#">L4virtio::Virtqueue::Used</a>	Used ring . . . . .	2298

<a href="#">L4virtio::Virtqueue::Used::Flags</a>	
Flags for the used ring	2299
<a href="#">L4virtio::Virtqueue::Used_elem</a>	
Type of an element of the used ring	2301
<a href="#">l4virtio_block_config_t</a>	
Device configuration for block devices	2302
<a href="#">l4virtio_block_discard_t</a>	
Structure used for the write zeroes and discard commands	2303
<a href="#">l4virtio_block_header_t</a>	
Header structure of a request for a block device	2304
<a href="#">l4virtio_config_hdr_t</a>	
L4-VIRTIO config header, provided in shared data space	2305
<a href="#">l4virtio_config_queue_t</a>	
Queue configuration entry	2306
<a href="#">l4virtio_input_absinfo_t</a>	
Information about the absolute axis in the underlying evdev implementation	2307
<a href="#">l4virtio_input_config_t</a>	
Device configuration for input devices	2308
<a href="#">l4virtio_input_devids_t</a>	
Device ID information for the device	2308
<a href="#">l4virtio_input_event_t</a>	
Single event in event or status queue	2309
<a href="#">l4virtio_net_config_t</a>	
Device configuration for network devices	2309
<a href="#">l4virtio_net_header_t</a>	
Header structure of a request for a network device	2310
<a href="#">L4virtio_port</a>	
A Port on the Virtio Net Switch	2311
<a href="#">Mac_addr</a>	
A wrapper class around the value of a MAC address	2316
<a href="#">Mac_table&lt; Size &gt;</a>	
<a href="#">Mac_table</a> manages a 1:n association between ports and MAC addresses	2316
<a href="#">Net_transfer</a>	
A network request to only a single destination	2319
<a href="#">Rm</a>	
<a href="#">Region</a> map	2322
<a href="#">Rm::Area</a>	
An area is a range of virtual addresses which is reserved, see <a href="#">L4Re::Rm::reserve_area()</a>	2337
<a href="#">Rm::F</a>	
<a href="#">Rm</a> flags definitions	2338
<a href="#">Rm::Region</a>	
A region is a range of virtual addresses which is backed by content	2340
<a href="#">Rm::Unique_region&lt; T &gt;</a>	
Unique region	2340
<a href="#">Switch_factory</a>	
The IPC interface for creating ports	2345
<a href="#">Virtio_net</a>	
The Base class of a Port	2349
<a href="#">Virtio_net_request</a>	
Abstraction for a network request	2354
<a href="#">Virtio_switch</a>	
The Virtio switch contains all ports and processes network requests	2357
<a href="#">Virtio_vlan_mangle</a>	
Class for VLAN packet rewriting	2362



# Chapter 13

## File Index

### 13.1 File List

Here is a list of all documented files with brief descriptions:

pkg/drivers-frst/include/asm_access_gen.h . . . . .	2372
pkg/drivers-frst/include/hw_mmio_register_block . . . . .	2372
pkg/drivers-frst/include/hw_register_block . . . . .	2373
pkg/drivers-frst/include/io_regblock.h . . . . .	2376
pkg/drivers-frst/include/io_regblock_port.h . . . . .	2379
pkg/drivers-frst/include/Makefile . . . . .	2412
pkg/drivers-frst/include/poll_timeout_counter.h . . . . .	2379
pkg/drivers-frst/include/ARCH-amd64/asm_access.h . . . . .	2367
pkg/drivers-frst/include/ARCH-arm/asm_access.h . . . . .	2367
pkg/drivers-frst/include/ARCH-arm64/asm_access.h . . . . .	2368
pkg/drivers-frst/include/ARCH-mips/asm_access.h . . . . .	2369
pkg/drivers-frst/include/ARCH-ppc32/asm_access.h . . . . .	2369
pkg/drivers-frst/include/ARCH-riscv/asm_access.h . . . . .	2370
pkg/drivers-frst/include/ARCH-sparc/asm_access.h . . . . .	2371
pkg/drivers-frst/include/ARCH-x86/asm_access.h . . . . .	2371
pkg/drivers-frst/uart/include/Makefile . . . . .	2412
pkg/drivers-frst/uart/include/uart_16550.h . . . . .	2380
pkg/drivers-frst/uart/include/uart_16550_dw.h . . . . .	2381
pkg/drivers-frst/uart/include/uart_apb.h . . . . .	2382
pkg/drivers-frst/uart/include/uart_base.h . . . . .	2382
pkg/drivers-frst/uart/include/uart_cadence.h . . . . .	2383
pkg/drivers-frst/uart/include/uart_dcc-v6.h . . . . .	2384
pkg/drivers-frst/uart/include/uart_dm.h . . . . .	2384
pkg/drivers-frst/uart/include/uart_dummy.h . . . . .	2385
pkg/drivers-frst/uart/include/uart_geni.h . . . . .	2385
pkg/drivers-frst/uart/include/uart_imx.h . . . . .	2386
pkg/drivers-frst/uart/include/uart_leon3.h . . . . .	2387
pkg/drivers-frst/uart/include/uart_linflex.h . . . . .	2387
pkg/drivers-frst/uart/include/uart_lpuart.h . . . . .	2388
pkg/drivers-frst/uart/include/uart_mvebu.h . . . . .	2388
pkg/drivers-frst/uart/include/uart_of.h . . . . .	2389
pkg/drivers-frst/uart/include/uart_omap35x.h . . . . .	2389
pkg/drivers-frst/uart/include/uart_pl011.h . . . . .	2390
pkg/drivers-frst/uart/include/uart_s3c2410.h . . . . .	2390
pkg/drivers-frst/uart/include/uart_sa1000.h . . . . .	2391

pkg/drivers-frst/uart/include/uart_sbi.h	2392
pkg/drivers-frst/uart/include/uart_sh.h	2392
pkg/drivers-frst/uart/include/uart_sifive.h	2393
pkg/drivers-frst/uart/include/uart_tegra-tcu.h	2393
pkg/l4re-core/ned/doc/tutorial.lua	2394
pkg/l4re-core/ned/lib/include/cmd_control	2397
pkg/l4re-core/ned/lib/include/Makefile	2412
pkg/virtio-net-switch/server/switch/debug.h	2397
pkg/virtio-net-switch/server/switch/filter.cc	2400
pkg/virtio-net-switch/server/switch/filter.h	2400
pkg/virtio-net-switch/server/switch/mac_addr.h	2401
pkg/virtio-net-switch/server/switch/mac_table.h	2402
pkg/virtio-net-switch/server/switch/main.cc	2404
pkg/virtio-net-switch/server/switch/Makefile	2412
pkg/virtio-net-switch/server/switch/options.cc	2413
pkg/virtio-net-switch/server/switch/options.h	2415
pkg/virtio-net-switch/server/switch/port.h	2416
pkg/virtio-net-switch/server/switch/port_ixl.h	2418
pkg/virtio-net-switch/server/switch/port_l4virtio.h	2420
pkg/virtio-net-switch/server/switch/request.h	2424
pkg/virtio-net-switch/server/switch/request_ixl.h	2425
pkg/virtio-net-switch/server/switch/request_l4virtio.h	2427
pkg/virtio-net-switch/server/switch/stats.h	2430
pkg/virtio-net-switch/server/switch/switch.cc	2431
pkg/virtio-net-switch/server/switch/switch.h	2434
pkg/virtio-net-switch/server/switch/virtio_net.h	2435
pkg/virtio-net-switch/server/switch/virtio_net_buffer.h	2439
pkg/virtio-net-switch/server/switch/vlan.h	2440
amd64/l4/sys/__kip-arch.h	2484
amd64/l4/sys/__vcpu-arch.h	
AMD64-specific vCPU interface	2485
amd64/l4/sys/cache.h	
Cache functions	3072
amd64/l4/sys/consts.h	
Common L4 constants, AMD64 version	3090
amd64/l4/sys/ktrace_events.h	2497
amd64/l4/sys/l4int.h	
Fixed sized integer types, AMD64 version	3235
amd64/l4/sys/linkage.h	
Linkage	2509
amd64/l4/sys/segment.h	
Segment handling (AMD64)	2463
amd64/l4/sys/utcb.h	
UTCB definitions for AMD64	3293
amd64/l4/sys/vm.h	2514
amd64/l4/util/bitops_arch.h	
Amd64 bit manipulation functions	2516
amd64/l4/util/cpu.h	
CPU related functions	2524
amd64/l4/util/irq.h	
Some PIC and hardware interrupt related functions	2666
amd64/l4/util/l4_macros.h	
Main function	2529
amd64/l4/util/mbi_argv.h	
Command line handling	2532
amd64/l4/util/perform.h	
Performance Monitoring using P5/P6 Measurement Counters	2441



amd64/l4/util/ <a href="#">port_io.h</a>	
Port I/O functions	2477
amd64/l4/util/ <a href="#">rdtsc.h</a>	
Timestamp counter related functions	2453
amd64/l4/util/ <a href="#">spin.h</a>	
Spinning for amd64	2461
amd64/l4f/l4/sys/ <a href="#">ipc.h</a>	3190
amd64/l4f/l4/sys/ <a href="#">segment.h</a>	
L4f-specific fs/gs manipulation (AMD64)	2469
amd64/l4f/l4/util/ <a href="#">port_io.h</a>	
Port I/O functions	2478
arm/l4/sys/ <a href="#">__kip-arch.h</a>	2485
arm/l4/sys/ <a href="#">__vcpu-arch.h</a>	
ARM-specific vCPU interface	2488
arm/l4/sys/ <a href="#">atomic.h</a>	3324
arm/l4/sys/ <a href="#">cache.h</a>	
Cache functions	3074
arm/l4/sys/ <a href="#">consts.h</a>	
Common L4 constants, arm version	3091
arm/l4/sys/ <a href="#">ktrace_events.h</a>	2500
arm/l4/sys/ <a href="#">l4int.h</a>	
Fixed sized integer types, arm version	3236
arm/l4/sys/ <a href="#">linkage.h</a>	
Linkage	2510
arm/l4/sys/ <a href="#">mem_op.h</a>	
Memory access functions (ARM specific)	2512
arm/l4/sys/ <a href="#">platform_control.h</a>	3253
arm/l4/sys/ <a href="#">task.h</a>	3278
arm/l4/sys/ <a href="#">thread.h</a>	
ARM-specific thread related definitions	3388
arm/l4/sys/ <a href="#">utcb.h</a>	
UTCB definitions for ARM	3295
arm/l4/sys/ <a href="#">vm</a>	3316
arm/l4/sys/ <a href="#">vm.h</a>	
ARM virtualization interface	2514
arm/l4/util/ <a href="#">bitops_arch.h</a>	
ARM specific implementation of bitops functions	2520
arm/l4/util/ <a href="#">cpu.h</a>	
CPU related functions	2526
arm/l4/util/ <a href="#">irq.h</a>	
ARM specific implementation of irq functions	2668
arm/l4/util/ <a href="#">l4_macros.h</a>	
Main function	2529
arm/l4/util/ <a href="#">mbi_argv.h</a>	
Multiboot	2533
arm/l4f/l4/sys/ <a href="#">ipc.h</a>	3191
arm/l4f/l4/sys/ <a href="#">syscall_defs.h</a>	
Syscall entry definitions	2536
arm64/l4/sys/ <a href="#">__kip-arch.h</a>	2485
arm64/l4/sys/ <a href="#">__vcpu-arch.h</a>	
ARM64-specific vCPU interface	2491
arm64/l4/sys/ <a href="#">cache.h</a>	
Cache functions	3076
arm64/l4/sys/ <a href="#">consts.h</a>	
Common L4 constants, arm version	3092
arm64/l4/sys/ <a href="#">ktrace_events.h</a>	2503
arm64/l4/sys/ <a href="#">l4int.h</a>	
Fixed sized integer types, arm version	3237

arm64/l4/sys/linkage.h	2511
arm64/l4/sys/platform_control.h	3253
arm64/l4/sys/task.h	3278
arm64/l4/sys/thread.h	
ARM64-specific thread related definitions	3389
arm64/l4/sys/utcb.h	
UTCB definitions for ARM64	3298
arm64/l4/sys/vm	3316
arm64/l4/sys/vm.h	2516
arm64/l4f/l4/sys/ipc.h	3192
contrib/libio-io/l4/io/io.h	2536
contrib/libio-io/l4/io/types.h	2810
l4/cxx/alloc.h	
Alloc list	2540
l4/cxx/arith	2541
l4/cxx/atomic.h	
Atomic template	3324
l4/cxx/avl_map	
AVL map	2542
l4/cxx/avl_set	
AVL set	2544
l4/cxx/avl_tree	
AVL tree	2549
l4/cxx/basic_ostream	
Basic IO stream	2556
l4/cxx/basic_vector.h	
Basic vector	2560
l4/cxx/bitfield	2561
l4/cxx/bitmap	2563
l4/cxx/dlist	2584
l4/cxx/exceptions	
Base exceptions	2589
l4/cxx/hlist	2593
l4/cxx/iostream	
IO Stream	2595
l4/cxx/ipc_helper	
IPC helper	2596
l4/cxx/ipc_server	
IPC server loop	3130
l4/cxx/ipc_stream	
IPC stream	2598
l4/cxx/ipc_timeout_queue	2616
l4/cxx/l4iostream	
L4 IO stream	2618
l4/cxx/l4types.h	
L4 Types	2619
l4/cxx/list	2620
l4/cxx/list_alloc	2624
l4/cxx/lock_guard.h	
Lock guard implementation	2630
l4/cxx/main_thread	
Main thread	2632
l4/cxx/minmax	2633
l4/cxx/numeric	2634
l4/cxx/observer	2634
l4/cxx/pair	
Pair implementation	2635
l4/cxx/ref_ptr	2637

l4/cxx/ref_ptr_list	
Implementation of a list of ref-ptr-managed objects	2640
l4/cxx/slab_alloc	2642
l4/cxx/slist	2646
l4/cxx/static_container	2649
l4/cxx/static_vector	2649
l4/cxx/std_alloc	2650
l4/cxx/std_ops	2650
l4/cxx/string	2651
l4/cxx/string.h	
String	2654
l4/cxx/thread	
Thread implementation	3282
l4/cxx/type_list	2655
l4/cxx/type_traits	2656
l4/cxx/unique_ptr	2660
l4/cxx/unique_ptr_list	
Implementation of a list of unique-ptr-managed objects	2662
l4/cxx/utils	2664
l4/cxx/weak_ref	2665
l4/cxx/bits/bst.h	
AVL tree	2566
l4/cxx/bits/bst_base.h	
AVL tree	2570
l4/cxx/bits/bst_iter.h	
AVL tree	2573
l4/cxx/bits/list_basics.h	2576
l4/cxx/bits/smart_ptr_list.h	
Implementation of a list of smart-pointer-managed objects	2578
l4/cxx/bits/type_traits.h	2581
l4/irq/irq.h	
IRQ handling routines	2669
l4/l4re_vfs/backend	2677
l4/l4re_vfs/vfs.h	2705
l4/l4re_vfs/impl/default_ops_impl.h	2680
l4/l4re_vfs/impl/fd_store.h	2681
l4/l4re_vfs/impl/fd_store_impl.h	2682
l4/l4re_vfs/impl/ns_fs.h	2682
l4/l4re_vfs/impl/ns_fs_impl.h	2683
l4/l4re_vfs/impl/ro_file.h	2688
l4/l4re_vfs/impl/ro_file_impl.h	2688
l4/l4re_vfs/impl/vcon_stream.h	2690
l4/l4re_vfs/impl/vcon_stream_impl.h	2690
l4/l4re_vfs/impl/vfs_impl.h	2692
l4/l4virtio/l4virtio	2719
l4/l4virtio/virtio.h	2777
l4/l4virtio/virtio_block.h	2780
l4/l4virtio/virtio_input.h	2781
l4/l4virtio/virtio_net.h	2438
l4/l4virtio/virtqueue	2782
l4/l4virtio/client/l4virtio	2716
l4/l4virtio/client/virtio-block	2736
l4/l4virtio/client/virtio-net	2714
l4/l4virtio/server/l4virtio	2721
l4/l4virtio/server/virtio	2732
l4/l4virtio/server/virtio-block	2739
l4/l4virtio/server/virtio-console	2746
l4/l4virtio/server/virtio-console-device	2752

l4/l4virtio/server/virtio-gpio-device	2755
l4/l4virtio/server/virtio-i2c-device	2761
l4/l4virtio/server/virtio-rng-device	2765
l4/l4virtio/server/virtio-scmi-device	2768
l4/libblock-device/block_device_mgr.h	2786
l4/libblock-device/debug.h	2398
l4/libblock-device/device.h	2792
l4/libblock-device/errand.h	2793
l4/libblock-device/gpt.h	2795
l4/libblock-device/inout_memory.h	2795
l4/libblock-device/part_device.h	2797
l4/libblock-device/partition.h	2799
l4/libblock-device/request.h	2425
l4/libblock-device/scheduler.h	2802
l4/libblock-device/types.h	2811
l4/libblock-device/virtio_client.h	2817
l4/libedid/edid.h	2826
l4/libgfxbitmap/bitmap.h	
Bitmap renderer header file	2827
l4/libgfxbitmap/font.h	
Bitmap font renderer header file	2833
l4/libgfxbitmap/support	
Terminal support functionality	2839
l4/re/cap_alloc	
Abstract capability-allocator interface	2943
l4/re/console	2877
l4/re/consts	
Constants	3102
l4/re/consts.h	
Constants	3093
l4/re/dataspace	
Dataspace interface	2878
l4/re/dataspace-sys.h	
Dataspace protocol definition	2881
l4/re/dbg_events	2882
l4/re/debug	
Debug interface	2959
l4/re/dma_space	2882
l4/re/elf_aux.h	
Auxiliary information for binaries	2885
l4/re/env	
Environment interface	2887
l4/re/env.h	
Environment interface	2889
l4/re/error_helper	
Error helper	2893
l4/re/event	2964
l4/re/event-sys.h	2896
l4/re/event.h	
Events	2848
l4/re/event_enums.h	2896
l4/re/inhibitor	2911
l4/re/inhibitor-sys.h	2911
l4/re/itas	2912
l4/re/l4aux.h	
Auxiliary definitions	2913
l4/re/log	
Log interface	2914

<a href="#">l4/re/log-sys.h</a>	
Log protocol definition	2916
<a href="#">l4/re/mem_alloc</a>	
Memory allocator interface	2916
<a href="#">l4/re/mem_alloc-sys.h</a>	
Memory allocator protocol definitions	2919
<a href="#">l4/re/mmio_space</a>	
Interface definition to emit MMIO-like accesses via IPC	2920
<a href="#">l4/re/namespace</a>	
Namespace interface	2921
<a href="#">l4/re/namespace-sys.h</a>	
Namespace protocol definitions	2924
<a href="#">l4/re/parent</a>	
Parent interface	2925
<a href="#">l4/re/parent-sys.h</a>	
Parent protocol definition	2926
<a href="#">l4/re/protocols.h</a>	
L4Re Protocol Constants (C version)	2927
<a href="#">l4/re/random</a>	
Random number generator interface definition	2928
<a href="#">l4/re/remote_access</a>	2930
<a href="#">l4/re/rm</a>	
Region mapper interface	2931
<a href="#">l4/re/rm-sys.h</a>	
Region mapper protocol definitions	2936
<a href="#">l4/re/shared_cap</a>	
Shared_cap / Shared_del_cap	2996
<a href="#">l4/re/unique_cap</a>	
Unique_cap / Unique_del_cap	3001
<a href="#">l4/re/c/dataspace.h</a>	
Data space C interface	2840
<a href="#">l4/re/c/debug.h</a>	
Debug C interface	2399
<a href="#">l4/re/c/dma_space.h</a>	
DMA space C interface	2843
<a href="#">l4/re/c/event.h</a>	
Event C interface	2845
<a href="#">l4/re/c/event_buffer.h</a>	2849
<a href="#">l4/re/c/inhibitor.h</a>	
Inhibitor C interface	2850
<a href="#">l4/re/c/log.h</a>	
Log C interface	2853
<a href="#">l4/re/c/mem_alloc.h</a>	
Memory allocator C interface	2855
<a href="#">l4/re/c/namespace.h</a>	
Namespace functions, C interface	2856
<a href="#">l4/re/c/parent.h</a>	
Parent C interface	2858
<a href="#">l4/re/c/rm.h</a>	
Region map interface, C interface	2860
<a href="#">l4/re/c/util/cap_alloc.h</a>	
Capability allocator C interface	2864
<a href="#">l4/re/c/util/kumem_alloc.h</a>	
Kumem allocator utility C interface	2866
<a href="#">l4/re/c/util/video/goos_fb.h</a>	
Framebuffer utility functionality	2868
<a href="#">l4/re/c/video/colors.h</a>	2869
<a href="#">l4/re/c/video/goos.h</a>	2871

I4/re/c/video/view.h	2874
I4/re/impl/dataspace_impl.h	
Dataspace client stub implementation	2902
I4/re/impl/mem_alloc_impl.h	
Memory allocator client stub implementation	2905
I4/re/impl/namespace_impl.h	
Namespace client stub implementation	2906
I4/re/impl/rm_impl.h	
Region map client stub implementation	2909
I4/re/util/bitmap_cap_alloc	
Bitmap capability allocator	2937
I4/re/util/br_manager	2939
I4/re/util/cap	
Capability utility functions	2941
I4/re/util/cap_alloc	
Capability allocator	2946
I4/re/util/cap_alloc_impl.h	
Capability allocator implementation	2949
I4/re/util/counting_cap_alloc	
Reference-counting capability allocator	2951
I4/re/util/dataspace_svr	2956
I4/re/util/debug	2961
I4/re/util/env_ns	2963
I4/re/util/event	2966
I4/re/util/event_buffer	2969
I4/re/util/event_svr	2970
I4/re/util/icu_svr	2971
I4/re/util/item_alloc	
Item allocator	2973
I4/re/util/kumem_alloc	
Kumem allocator helper	2976
I4/re/util/meta	3243
I4/re/util/name_space_svr	2977
I4/re/util/object_registry	2983
I4/re/util/poll_timeout_kipclock	2985
I4/re/util/region_mapping	
Region handling	2986
I4/re/util/region_mapping_svr	2993
I4/re/util/shared_cap	
Shared_cap / Shared_del_cap	2999
I4/re/util/unique_cap	
Unique_cap / Unique_del_cap	3003
I4/re/util/vcon_svr	3006
I4/re/util/video/goos_fb	3007
I4/re/util/video/goos_svr	3009
I4/re/video/colors	3010
I4/re/video/goos	3012
I4/re/video/goos-sys.h	
Goos protocol definition	3015
I4/re/video/view	3016
I4/shmc/ringbuf.h	3016
I4/shmc/shmc.h	
Shared memory library header file	3026
I4/sigma0/sigma0.h	
Sigma0 interface	3031
I4/sys/_kernel_object_impl.h	3034
I4/sys/_ktrace-impl.h	
L4 kernel event tracing	3034

<a href="#">l4/sys/_l4_fpage.h</a>	3036
<a href="#">l4/sys/_platform_control-arm.h</a>	3041
<a href="#">l4/sys/_task-arm.h</a>	3041
<a href="#">l4/sys/_timeout.h</a>	3042
<a href="#">l4/sys/_typeinfo.h</a>	
Type information handling	3044
<a href="#">l4/sys/_vcpu-arm.h</a>	3056
<a href="#">l4/sys/_vm-arm.h</a>	
Virtualization interface	3057
<a href="#">l4/sys/_vm-svm.h</a>	3059
<a href="#">l4/sys/_vm-vmx.h</a>	3061
<a href="#">l4/sys/arm_smccc</a>	
ARM secure monitor call functions	3067
<a href="#">l4/sys/arm_smccc.h</a>	
ARM secure monitor call functions	3069
<a href="#">l4/sys/assert.h</a>	
Low-level assert implementation	3318
<a href="#">l4/sys/cache.h</a>	
Cache-consistency functions	3079
<a href="#">l4/sys/capability</a>	
L4::Cap related definitions	3082
<a href="#">l4/sys/compiler.h</a>	
L4 compiler related defines	3086
<a href="#">l4/sys/consts.h</a>	
Common constants	3095
<a href="#">l4/sys/debugger</a>	
The debugger interface specifies common debugging related definitions	3162
<a href="#">l4/sys/debugger.h</a>	
Debugger related definitions	3163
<a href="#">l4/sys/err.h</a>	
Error codes	3168
<a href="#">l4/sys/exception</a>	
Exception C++ interface	3170
<a href="#">l4/sys/factory</a>	
Common factory related definitions	3172
<a href="#">l4/sys/factory.h</a>	
Common factory related definitions	3176
<a href="#">l4/sys/icu</a>	
Interrupt controller	3182
<a href="#">l4/sys/icu.h</a>	
Interrupt controller	3184
<a href="#">l4/sys/iommu</a>	3190
<a href="#">l4/sys/ipc.h</a>	
Common IPC interface	3192
<a href="#">l4/sys/ipc_gate</a>	
The C++ IPC gate interface	3200
<a href="#">l4/sys/ipc_gate.h</a>	
The C IPC gate interface, see <a href="#">L4::lpc_gate</a> for the C++ interface	3201
<a href="#">l4/sys/irq</a>	
C++ Irq interface	3205
<a href="#">l4/sys/irq.h</a>	
C Irq interface	2672
<a href="#">l4/sys/kdebug.h</a>	
Functionality for invoking the kernel debugger	3208
<a href="#">l4/sys/kdump.h</a>	
Functionality for dumping kernel information	3227
<a href="#">l4/sys/kernel_object.h</a>	
Kernel object system calls	3228

<a href="#">l4/sys/kip</a>	3230
<a href="#">l4/sys/kip.h</a>	
Kernel Info Page access functions	3362
<a href="#">l4/sys/kobject</a>	3233
<a href="#">l4/sys/ktrace.h</a>	
L4 kernel event tracing	3233
<a href="#">l4/sys/l4int.h</a>	
Fixed sized integer types, generic version	3237
<a href="#">l4/sys/memdesc.h</a>	
Memory description functions	3240
<a href="#">l4/sys/meta</a>	
Meta interface for getting dynamic type information about objects behind capabilities	3243
<a href="#">l4/sys/obj_info.h</a>	
Debugger related functions	3245
<a href="#">l4/sys/pager</a>	
Pager and lo_pager C++ interface	3249
<a href="#">l4/sys/platform_control</a>	
Platform control object	3251
<a href="#">l4/sys/platform_control.h</a>	
Platform control object	3254
<a href="#">l4/sys/rcv_endpoint</a>	
The C++ Receive endpoint interface	3257
<a href="#">l4/sys/rcv_endpoint.h</a>	
Receive endpoint C interface	3259
<a href="#">l4/sys/scheduler</a>	
Scheduler object functions	3262
<a href="#">l4/sys/scheduler.h</a>	
Scheduler object functions	2805
<a href="#">l4/sys/semaphore</a>	
Semaphore class definition	3264
<a href="#">l4/sys/semaphore.h</a>	
C semaphore interface	3266
<a href="#">l4/sys/smart_capability</a>	
L4::Capability class	3268
<a href="#">l4/sys/snd_destination</a>	
The C++ Sender destination interface	3272
<a href="#">l4/sys/snd_destination.h</a>	
Sender destination endpoint C interface	3274
<a href="#">l4/sys/task</a>	
Common task related definitions	3275
<a href="#">l4/sys/task.h</a>	
Common task related definitions	3278
<a href="#">l4/sys/thread</a>	
Common thread related definitions	3284
<a href="#">l4/sys/thread.h</a>	
Common thread related definitions	3390
<a href="#">l4/sys/thread_group</a>	3288
<a href="#">l4/sys/thread_group.h</a>	3289
<a href="#">l4/sys/typeinfo_svr</a>	
Type information server template	3291
<a href="#">l4/sys/types.h</a>	
Common L4 ABI Data Types	2812
<a href="#">l4/sys/utcb.h</a>	
UTCB definitions	3300
<a href="#">l4/sys/vcon</a>	
C++ Virtual console interface	3307
<a href="#">l4/sys/vcon.h</a>	
Virtual console interface	3309



<a href="#">l4/sys/vcpu.h</a>	
VCPU API . . . . .	3425
<a href="#">l4/sys/vcpu_context</a>	
Hardware vCPU context interface . . . . .	3315
<a href="#">l4/sys/vcpu_context.h</a> . . . . .	3316
<a href="#">l4/sys/vm</a>	
Virtualization interface . . . . .	3317
<a href="#">l4/sys/cxx/capability.h</a> . . . . .	3099
<a href="#">l4/sys/cxx/consts</a> . . . . .	3103
<a href="#">l4/sys/cxx/ipc_array</a> . . . . .	3104
<a href="#">l4/sys/cxx/ipc_basics</a> . . . . .	3107
<a href="#">l4/sys/cxx/ipc_client</a> . . . . .	3111
<a href="#">l4/sys/cxx/ipc_epiface</a> . . . . .	3114
<a href="#">l4/sys/cxx/ipc_iface</a>	
Interface Definition Language . . . . .	3118
<a href="#">l4/sys/cxx/ipc_legacy</a> . . . . .	3129
<a href="#">l4/sys/cxx/ipc_ret_array</a> . . . . .	3129
<a href="#">l4/sys/cxx/ipc_server</a> . . . . .	3132
<a href="#">l4/sys/cxx/ipc_server_loop</a> . . . . .	3136
<a href="#">l4/sys/cxx/ipc_string</a> . . . . .	3139
<a href="#">l4/sys/cxx/ipc_types</a> . . . . .	3141
<a href="#">l4/sys/cxx/ipc_varg</a> . . . . .	3149
<a href="#">l4/sys/cxx/smart_capability_1x</a> . . . . .	3155
<a href="#">l4/sys/cxx/types</a> . . . . .	3158
<a href="#">l4/util/assert.h</a>	
Some useful assert-style macros . . . . .	3321
<a href="#">l4/util/atomic.h</a>	
Atomic operations header and generic implementations . . . . .	3326
<a href="#">l4/util/backtrace.h</a>	
Backtrace . . . . .	3333
<a href="#">l4/util/base64.h</a>	
Base 64 encoding and decoding functions adapted from Bob Trower 08/04/01 . . . . .	3335
<a href="#">l4/util/bitops.h</a>	
Bit manipulation functions . . . . .	3337
<a href="#">l4/util/elf.h</a>	
ELF definition . . . . .	3342
<a href="#">l4/util/getopt.h</a>	
Getopt . . . . .	3359
<a href="#">l4/util/keymap.h</a>	
Event to ASCII key mapping . . . . .	3361
<a href="#">l4/util/kip.h</a> . . . . .	3367
<a href="#">l4/util/kprintf.h</a>	
Printf using the kernel debugger . . . . .	3368
<a href="#">l4/util/l4_macros.h</a>	
Some useful generic macros, L4f version . . . . .	2530
<a href="#">l4/util/l4mod.h</a>	
L4mod structures and constants . . . . .	3369
<a href="#">l4/util/list_alloc.h</a>	
Simple list-based allocator . . . . .	3371
<a href="#">l4/util/lock.h</a>	
Simple lock implementation . . . . .	3374
<a href="#">l4/util/mb_info.h</a>	
Multiboot info structure as defined by GRUB . . . . .	3375
<a href="#">l4/util/parse_cmd.h</a>	
Comfortable command-line parsing . . . . .	3383
<a href="#">l4/util/printf_helpers.h</a> . . . . .	3385
<a href="#">l4/util/rand.h</a>	
Simple Pseudo-Random Number Generator . . . . .	3385

l4/util/ <a href="#">splitlog2.h</a>	
Split a range in log2 aligned and size-aligned chunks	3386
l4/util/ <a href="#">thread.h</a>	
Low-level Thread Functions	3400
l4/util/ <a href="#">util.h</a>	3402
l4/vbus/ <a href="#">vbus</a>	3403
l4/vbus/ <a href="#">vbus.h</a>	
Description of the vbus C API	3405
l4/vbus/ <a href="#">vbus_generic</a>	3408
l4/vbus/ <a href="#">vbus_gpio</a>	3409
l4/vbus/ <a href="#">vbus_gpio-ops.h</a>	3410
l4/vbus/ <a href="#">vbus_gpio.h</a>	3410
l4/vbus/ <a href="#">vbus_i2c.h</a>	3411
l4/vbus/ <a href="#">vbus_inhibitor.h</a>	3412
l4/vbus/ <a href="#">vbus_interfaces.h</a>	
This header contains the definition of VBUS sub-interfaces and convenience functions to work with the interface IDs	3412
l4/vbus/ <a href="#">vbus_mcspi.h</a>	3415
l4/vbus/ <a href="#">vbus_pci</a>	3416
l4/vbus/ <a href="#">vbus_pci-ops.h</a>	3417
l4/vbus/ <a href="#">vbus_pci.h</a>	3417
l4/vbus/ <a href="#">vbus_pm-ops.h</a>	3418
l4/vbus/ <a href="#">vbus_pm.h</a>	3418
l4/vbus/ <a href="#">vbus_types.h</a>	
This header file contains descriptions of vbus related data types and constants	3418
l4/vbus/ <a href="#">vdevice-ops.h</a>	3422
l4/vcpu/ <a href="#">vcpu</a>	
VCPU support library (C++ interface)	3423
l4/vcpu/ <a href="#">vcpu.h</a>	
VCPU support library (C interface)	3429
x86/l4/sys/ <a href="#">__kip-arch.h</a>	2485
x86/l4/sys/ <a href="#">__vcpu-arch.h</a>	
X86-specific vCPU interface	2495
x86/l4/sys/ <a href="#">cache.h</a>	
Cache functions	3081
x86/l4/sys/ <a href="#">consts.h</a>	
Common L4 constants, x86 version	3098
x86/l4/sys/ <a href="#">ipc-invoke.h</a>	3432
x86/l4/sys/ <a href="#">ktrace_events.h</a>	2506
x86/l4/sys/ <a href="#">l4int.h</a>	
Fixed sized integer types, x86 version	3239
x86/l4/sys/ <a href="#">linkage.h</a>	
Linkage	2511
x86/l4/sys/ <a href="#">segment.h</a>	
Segment handling (x86)	2473
x86/l4/sys/ <a href="#">utcb.h</a>	
UTCB definitions for x86	3304
x86/l4/sys/ <a href="#">vm.h</a>	2516
x86/l4/util/ <a href="#">bitops_arch.h</a>	
X86 bit manipulation functions	2521
x86/l4/util/ <a href="#">cpu.h</a>	
CPU related functions	2527
x86/l4/util/ <a href="#">irq.h</a>	
Some PIC and hardware interrupt related functions	2675
x86/l4/util/ <a href="#">l4_macros.h</a>	
Main function	2531
x86/l4/util/ <a href="#">mbi_argv.h</a>	
Command line handling	2534

x86/l4/util/ <a href="#">perform.h</a>	
Performance Monitoring using P5/P6 Measurement Counters . . . . .	2447
x86/l4/util/ <a href="#">port_io.h</a>	
X86 port I/O . . . . .	2478
x86/l4/util/ <a href="#">rdtsc.h</a>	
Timestamp counter related functions . . . . .	2457
x86/l4/util/ <a href="#">spin.h</a>	
Spinning for x86 . . . . .	2462
x86/l4f/l4/sys/ <a href="#">ipc-l42-gcc3-nopic.h</a> . . . . .	3433
x86/l4f/l4/sys/ <a href="#">ipc.h</a>	
L4 IPC System Calls, x86 . . . . .	3198
x86/l4f/l4/sys/ <a href="#">segment.h</a>	
L4f-specific segment manipulation (x86) . . . . .	2475
x86/l4f/l4/util/ <a href="#">port_io.h</a>	
Port I/O functions . . . . .	2482



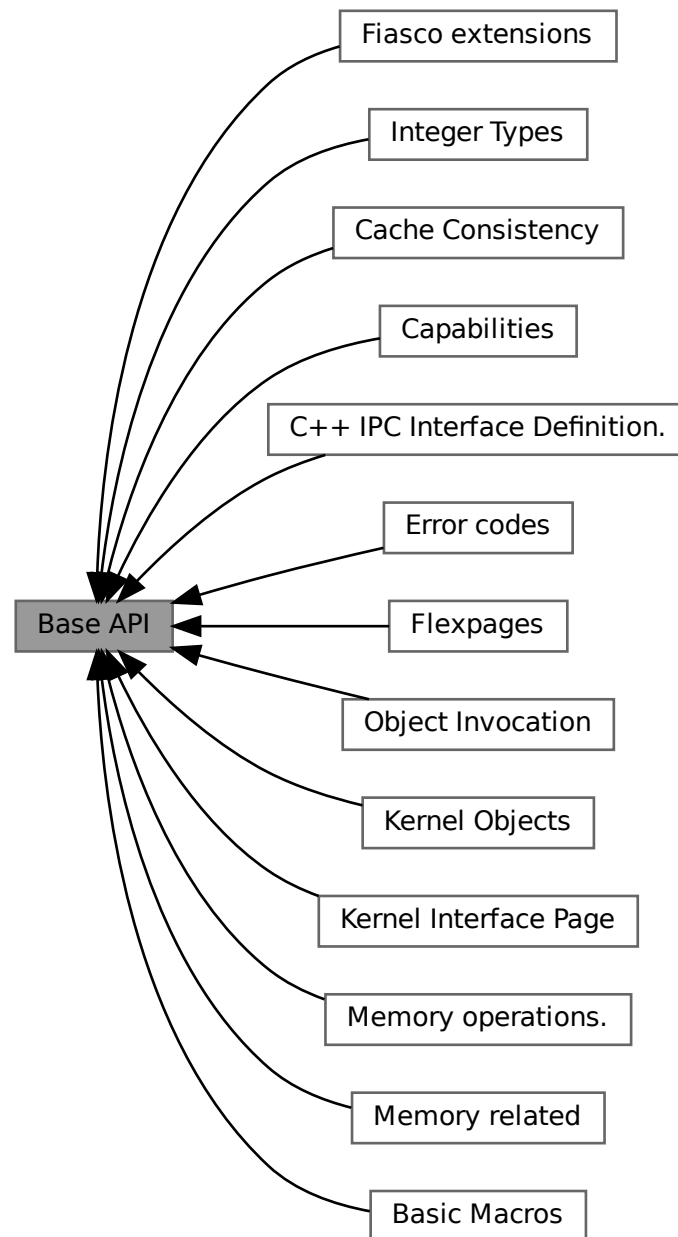
## Chapter 14

# Topic Documentation

### 14.1 Base API

Interfaces for all kinds of base functionality.

Collaboration diagram for Base API:



## Topics

- Basic Macros . . . . . 160  
*L4 standard macros for header files, function definitions, and public APIs etc.*
- Fiasco extensions . . . . . 165

<i>Extensions of the Fiasco L4 implementation.</i>	
• Flexpages . . . . .	182
<i>Flexpage-related API.</i>	
• C++ IPC Interface Definition. . . . .	199
<i>APIs for defining IPC interfaces using C++ as language.</i>	
• Cache Consistency . . . . .	200
<i>Various functions for cache consistency.</i>	
• Memory related . . . . .	204
<i>Memory related constants, data types and functions.</i>	
• Error codes . . . . .	213
<i>Common error codes.</i>	
• Object Invocation . . . . .	215
<i>API for L4 object invocation.</i>	
• Kernel Objects . . . . .	284
<i>API of kernel objects.</i>	
• Kernel Interface Page . . . . .	436
<i>Kernel Interface Page.</i>	
• Capabilities . . . . .	450
<i>C interface for capabilities.</i>	
• Memory operations. . . . .	454
<i>Operations for memory access.</i>	
• Integer Types . . . . .	457

## Files

- file [cache.h](#)  
*Cache-consistency functions.*
- file [compiler.h](#)  
*L4 compiler related defines.*
- file [consts.h](#)  
*Common constants.*
- file [debugger.h](#)  
*Debugger related definitions.*
- file [factory.h](#)  
*Common factory related definitions.*
- file [icu](#)  
*Interrupt controller.*
- file [icu.h](#)  
*Interrupt controller.*

- file [ipc.h](#)  
*Common IPC interface.*
- file [irq.h](#)  
*C Irq interface.*
- file [kip](#)
- file [kip.h](#)  
*Kernel Info Page access functions.*
- file [memdesc.h](#)  
*Memory description functions.*
- file [semaphore.h](#)  
*C semaphore interface.*
- file [types.h](#)  
*Common L4 ABI Data Types.*
- file [consts.h](#)  
*Common L4 constants, arm version.*
- file [consts.h](#)  
*Common L4 constants, arm version.*
- file [consts.h](#)  
*Common L4 constants, AMD64 version.*
- file [ipc.h](#)  
*L4 IPC System Calls, x86.*
- file [consts.h](#)  
*Common L4 constants, x86 version.*

### 14.1.1 Detailed Description

Interfaces for all kinds of base functionality.

Some notes on Inter Process Communication (IPC)

IPC in L4 is always synchronous and unbuffered: a message is transferred from the sender to the recipient if and only if the recipient has invoked a corresponding IPC operation. The sender blocks until this happens or a timeout specified by the sender elapsed without the destination becoming ready to receive.

### 14.1.2 Basic Macros

L4 standard macros for header files, function definitions, and public APIs etc.

Collaboration diagram for Basic Macros:





## Files

- file [linkage.h](#)  
*Linkage.*
- file [linkage.h](#)  
*Linkage.*
- file [linkage.h](#)  
*Linkage.*

## Macros

- **#define L4\_INLINE**  
*L4 Inline function attribute.*
- **#define L4\_ALWAYS\_INLINE**  
*Always inline a function.*
- **#define L4\_BEGIN\_DECLS**  
*Start section with C types and functions.*
- **#define L4\_END\_DECLS**  
*End section with C types and functions.*
- **#define L4\_NOTHROW**  
*Mark a function declaration and definition as never throwing an exception.*
- **#define L4\_EXPORT**  
*Attribute to mark functions, variables, and data types as being exported from a library.*
- **#define L4\_HIDDEN**  
*Attribute to mark functions, variables, and data types as being explicitly hidden from users of a library.*
- **#define L4\_CONSTEXPR**  
*Constexpr function attribute.*
- **#define L4\_NORETURN**  
*Noreturn function attribute.*
- **#define L4\_NOINSTRUMENT**  
*No instrumentation function attribute.*
- **#define L4\_LIKELY(x)**  
*Expression is likely to execute.*
- **#define L4\_UNLIKELY(x)**  
*Expression is unlikely to execute.*
- **#define L4\_STICKY(x)**  
*Mark symbol sticky (even not there).*
- **#define L4\_DEPRECATED(s)**  
*Mark symbol deprecated.*
- **#define L4\_stringify\_helper(x)**  
*stringify helper.*
- **#define L4\_stringify(x)**  
*stringify.*
- **#define L4\_CV**  
*Define calling convention.*
- **#define L4\_CV**  
*Define calling convention.*
- **#define L4\_CV**  
*Define calling convention.*
- **#define L4\_CV**  
*Define calling convention.*

## Functions

- unsigned long [l4\\_align\\_stack\\_for\\_direct\\_fncall](#) (unsigned long stack)  
*Specify the desired alignment of the stack pointer.*
- void [l4\\_barrier](#) (void)  
*Memory barrier.*
- void [l4\\_mb](#) (void)  
*Memory barrier.*
- void [l4\\_wmb](#) (void)  
*Write memory barrier.*
- [L4\\_NORETURN](#) void [l4\\_infinite\\_loop](#) (void)  
*Infinite loop.*

### 14.1.2.1 Detailed Description

[L4](#) standard macros for header files, function definitions, and public APIs etc.

#### Include File

```
#include <l4/sys/compiler.h>
```

### 14.1.2.2 Macro Definition Documentation

#### 14.1.2.2.1 L4\_EXPORT

```
#define L4_EXPORT
```

Attribute to mark functions, variables, and data types as being exported from a library.

All data types, functions, and global variables that shall be exported from a library shall be marked with this attribute. The default may become to hide everything that is not marked as `L4_EXPORT` from the users of a library and provide the possibility for aggressive optimization of all those internal functionality of a library.

#### Usage:

```
class L4_EXPORT My_class
{
    ...
};

int L4_EXPORT function(void);

int L4_EXPORT global_data; // global data is not recommended
```

Definition at line 214 of file [compiler.h](#).

Referenced by [l4io\\_iterate\\_devices\(\)](#), [l4io\\_lookup\\_device\(\)](#), [l4io\\_lookup\\_resource\(\)](#), [l4io\\_release\\_iomem\(\)](#), [l4io\\_release\\_ioport\(\)](#), [l4io\\_request\\_all\\_ioports\(\)](#), [l4io\\_request\\_icu\(\)](#), [l4io\\_request\\_iomem\(\)](#), [l4io\\_request\\_iomem\\_region\(\)](#), [l4io\\_request\\_ioport\(\)](#), [l4io\\_request\\_resource\\_iomem\(\)](#), [l4re\\_inhibitor\\_acquire\(\)](#), and [l4re\\_inhibitor\\_release\(\)](#).

### 14.1.2.2.2 L4\_HIDDEN

```
#define L4_HIDDEN
```

Attribute to mark functions, variables, and data types as being explicitly hidden from users of a library.

This attribute is intended for functions, data, and data types that shall never be visible outside of a library. In particular, for shared libraries this may result in much faster code within the library and short linking times.

```
class L4_HIDDEN My_class
{
    ...
};

int L4_HIDDEN function(void);

int L4_HIDDEN global_data; // global data is not recommended
```

Definition at line 211 of file [compiler.h](#).

### 14.1.2.2.3 L4\_NOTHROW

```
#define L4_NOTHROW
```

Mark a function declaration and definition as never throwing an exception.

(Also for C code).

This macro shall be used to mark C and C++ functions that never throw any exception. Note that also C functions may throw exceptions according to the compilers ABI and shall be marked with L4\_NOTHROW if they never do. In C++ this is equivalent to `throw()`.

```
int foo() L4_NOTHROW;
...
int foo() L4_NOTHROW
{
    ...
    return result;
}
```

Definition at line 161 of file [compiler.h](#).

Referenced by [\\_\\_kdebug\\_3\\_text\(\)](#), [\\_\\_kdebug\\_op\(\)](#), [\\_\\_kdebug\\_op\\_1\(\)](#), [\\_\\_kdebug\\_text\(\)](#), [enter\\_kdebug\(\)](#), [l4\\_msgtag\\_t::flags\(\)](#), [l4\\_msgtag\\_t::has\\_error\(\)](#), [l4\\_msgtag\\_t::is\\_exception\(\)](#), [l4\\_msgtag\\_t::is\\_io\\_page\\_fault\(\)](#), [l4\\_msgtag\\_t::is\\_page\\_fault\(\)](#), [l4\\_msgtag\\_t::is\\_sigma0\(\)](#), [l4\\_msgtag\\_t::items\(\)](#), [l4\\_arm\\_smccc\\_call\(\)](#), [l4\\_bytes\\_to\\_mwords\(\)](#), [l4\\_cache\\_clean\\_data\(\)](#), [l4\\_cache\\_coherent\(\)](#), [l4\\_cache\\_dma\\_coherent\(\)](#), [l4\\_cache\\_dma\\_coherent\\_full\(\)](#), [l4\\_cache\\_flush\\_data\(\)](#), [l4\\_cache\\_inv\\_data\(\)](#), [l4\\_capability\\_equal\(\)](#), [l4\\_capability\\_next\(\)](#), [l4\\_debugger\\_add\\_image\\_info\(\)](#), [l4\\_debugger\\_get\\_object\\_name\(\)](#), [l4\\_debugger\\_global\\_id\(\)](#), [l4\\_debugger\\_kobj\\_to\\_id\(\)](#), [l4\\_debugger\\_query\\_log\\_name\(\)](#), [l4\\_debugger\\_query\\_log\\_typeid\(\)](#), [l4\\_debugger\\_query\\_obj\\_infos\(\)](#), [l4\\_debugger\\_set\\_object\\_name\(\)](#), [l4\\_debugger\\_switch\\_log\(\)](#), [l4\\_error\(\)](#), [l4\\_factory\\_create\(\)](#), [l4\\_factory\\_create\\_factory\(\)](#), [l4\\_factory\\_create\\_gate\(\)](#), [l4\\_factory\\_create\\_irq\(\)](#), [l4\\_factory\\_create\\_task\(\)](#), [l4\\_factory\\_create\\_thread\(\)](#), [l4\\_factory\\_create\\_thread\\_group\(\)](#), [l4\\_factory\\_create\\_vcpu\\_context\(\)](#), [l4\\_factory\\_create\\_vm\(\)](#), [l4\\_fpage\(\)](#), [l4\\_fpage\\_all\(\)](#), [l4\\_fpage\\_contains\(\)](#), [l4\\_fpage\\_invalid\(\)](#), [l4\\_fpage\\_ioport\(\)](#), [l4\\_fpage\\_memaddr\(\)](#), [l4\\_fpage\\_obj\(\)](#), [l4\\_fpage\\_page\(\)](#), [l4\\_fpage\\_rights\(\)](#), [l4\\_fpage\\_set\\_rights\(\)](#), [l4\\_fpage\\_size\(\)](#), [l4\\_fpage\\_type\(\)](#), [l4\\_icu\\_bind\(\)](#), [l4\\_icu\\_bind\\_u\(\)](#), [l4\\_icu\\_info\(\)](#), [l4\\_icu\\_info\\_u\(\)](#), [l4\\_icu\\_mask\(\)](#), [l4\\_icu\\_mask\\_u\(\)](#), [l4\\_icu\\_msi\\_info\(\)](#), [l4\\_icu\\_msi\\_info\\_u\(\)](#), [l4\\_icu\\_set\\_mode\(\)](#), [l4\\_icu\\_set\\_mode\\_u\(\)](#), [l4\\_icu\\_unbind\(\)](#), [l4\\_icu\\_unbind\\_u\(\)](#), [l4\\_icu\\_unmask\(\)](#), [l4\\_icu\\_unmask\\_u\(\)](#), [l4\\_iofpage\(\)](#), [l4\\_ipc\(\)](#), [l4\\_ipc\\_call\(\)](#), [l4\\_ipc\\_error\(\)](#), [l4\\_ipc\\_error\\_code\(\)](#), [l4\\_ipc\\_is\\_rcv\\_error\(\)](#), [l4\\_ipc\\_is\\_snd\\_error\(\)](#), [l4\\_ipc\\_receive\(\)](#), [l4\\_ipc\\_reply\\_and\\_wait\(\)](#), [l4\\_ipc\\_send\(\)](#), [l4\\_ipc\\_send\\_and\\_wait\(\)](#), [l4\\_ipc\\_sleep\(\)](#), [l4\\_ipc\\_sleep\\_ms\(\)](#), [l4\\_ipc\\_sleep\\_us\(\)](#), [l4\\_ipc\\_timeout\(\)](#), [l4\\_ipc\\_to\\_errno\(\)](#), [l4\\_ipc\\_wait\(\)](#), [l4\\_irq\\_bind\\_vcpu\(\)](#), [l4\\_irq\\_bind\\_vcpu\\_u\(\)](#), [l4\\_irq\\_detach\(\)](#), [l4\\_irq\\_detach\\_u\(\)](#), [l4\\_irq\\_receive\(\)](#), [l4\\_irq\\_receive\\_u\(\)](#), [l4\\_irq\\_trigger\(\)](#), [l4\\_irq\\_trigger\\_u\(\)](#), [l4\\_irq\\_unmask\(\)](#), [l4\\_irq\\_unmask\\_u\(\)](#), [l4\\_irq\\_wait\(\)](#), [l4\\_irq\\_wait\\_u\(\)](#), [l4\\_is\\_fpage\\_valid\(\)](#), [l4\\_is\\_fpage\\_writable\(\)](#), [l4\\_is\\_invalid\\_cap\(\)](#), [l4\\_is\\_valid\\_cap\(\)](#), [l4\\_kernel\\_info\\_get\\_mem\\_desc\\_end\(\)](#), [l4\\_kernel\\_info\\_get\\_mem\\_desc\\_info\(\)](#), [l4\\_kernel\\_info\\_get\\_mem\\_desc\\_start\(\)](#), [l4\\_kernel\\_info\\_get\\_mem\\_desc\\_subtype\(\)](#), [l4\\_kernel\\_info\\_get\\_mem\\_desc\\_type\(\)](#), [l4\\_kernel\\_info\\_get\\_mem\\_descs\(\)](#), [l4\\_kernel\\_info\\_get\\_num\\_mem\\_descs\(\)](#), [l4\\_kernel\\_info\\_set\\_mem\\_desc\(\)](#),

[l4\\_kernel\\_info\\_version\\_offset\(\)](#), [l4\\_kip\(\)](#), [l4\\_kip\\_clock\(\)](#), [l4\\_kip\\_clock\\_lw\(\)](#), [l4\\_kip\\_clock\\_ns\(\)](#), [l4\\_kip\\_version\(\)](#),  
[l4\\_kip\\_version\\_string\(\)](#), [l4\\_map\\_control\(\)](#), [l4\\_map\\_obj\\_control\(\)](#), [l4\\_msgtag\(\)](#), [l4\\_msgtag\\_flags\(\)](#), [l4\\_msgtag\\_has\\_error\(\)](#),  
[l4\\_msgtag\\_is\\_exception\(\)](#), [l4\\_msgtag\\_is\\_io\\_page\\_fault\(\)](#), [l4\\_msgtag\\_is\\_page\\_fault\(\)](#), [l4\\_msgtag\\_is\\_sigma0\(\)](#),  
[l4\\_msgtag\\_items\(\)](#), [l4\\_msgtag\\_label\(\)](#), [l4\\_msgtag\\_words\(\)](#), [l4\\_obj\\_fpage\(\)](#), [l4\\_platform\\_ctl\\_cpu\\_allow\\_shutdown\(\)](#),  
[l4\\_platform\\_ctl\\_cpu\\_disable\(\)](#), [l4\\_platform\\_ctl\\_cpu\\_enable\(\)](#), [l4\\_platform\\_ctl\\_set\\_task\\_asid\(\)](#), [l4\\_platform\\_ctl\\_system\\_shutdown\(\)](#),  
[l4\\_platform\\_ctl\\_system\\_suspend\(\)](#), [l4\\_rcv\\_timeout\(\)](#), [l4\\_round\\_page\(\)](#), [l4\\_round\\_size\(\)](#), [l4\\_sched\\_cpu\\_set\(\)](#),  
[l4\\_sched\\_param\(\)](#), [l4\\_scheduler\\_idle\\_time\(\)](#), [l4\\_scheduler\\_info\(\)](#), [l4\\_scheduler\\_info\\_with\\_classes\(\)](#), [l4\\_scheduler\\_is\\_online\(\)](#),  
[l4\\_scheduler\\_run\\_thread\(\)](#), [l4\\_semaphore\\_down\(\)](#), [l4\\_semaphore\\_up\(\)](#), [l4\\_sleep\(\)](#), [l4\\_sleep\\_forever\(\)](#), [l4\\_snd\\_timeout\(\)](#),  
[l4\\_sndpage\\_add\(\)](#), [l4\\_task\\_add\\_ku\\_mem\(\)](#), [l4\\_task\\_cap\\_equal\(\)](#), [l4\\_task\\_cap\\_valid\(\)](#), [l4\\_task\\_delete\\_obj\(\)](#),  
[l4\\_task\\_map\(\)](#), [l4\\_task\\_release\\_cap\(\)](#), [l4\\_task\\_unmap\(\)](#), [l4\\_task\\_unmap\\_batch\(\)](#), [l4\\_task\\_vgicc\\_map\(\)](#),  
[l4\\_thread\\_arm\\_set\\_tpidruro\(\)](#), [l4\\_thread\\_control\\_alien\(\)](#), [l4\\_thread\\_control\\_bind\(\)](#), [l4\\_thread\\_control\\_commit\(\)](#),  
[l4\\_thread\\_control\\_exc\\_handler\(\)](#), [l4\\_thread\\_control\\_pager\(\)](#), [l4\\_thread\\_control\\_start\(\)](#), [l4\\_thread\\_ex\\_regs\(\)](#),  
[l4\\_thread\\_ex\\_regs\\_ret\(\)](#), [l4\\_thread\\_ex\\_regs\\_ret\\_u\(\)](#), [l4\\_thread\\_ex\\_regs\\_u\(\)](#), [l4\\_thread\\_group\\_add\(\)](#), [l4\\_thread\\_group\\_remove\(\)](#),  
[l4\\_thread\\_modify\\_sender\\_add\(\)](#), [l4\\_thread\\_modify\\_sender\\_commit\(\)](#), [l4\\_thread\\_modify\\_sender\\_start\(\)](#), [l4\\_thread\\_register\\_del\\_irq\(\)](#),  
[l4\\_thread\\_register\\_doorbell\\_irq\(\)](#), [l4\\_thread\\_stats\\_time\(\)](#), [l4\\_thread\\_switch\(\)](#), [l4\\_thread\\_vcpu\\_control\(\)](#), [l4\\_thread\\_vcpu\\_control\\_ext\(\)](#),  
[l4\\_thread\\_vcpu\\_control\\_ext\\_u\(\)](#), [l4\\_thread\\_vcpu\\_control\\_u\(\)](#), [l4\\_thread\\_vcpu\\_resume\\_commit\(\)](#), [l4\\_thread\\_vcpu\\_resume\\_start\(\)](#),  
[l4\\_thread\\_yield\(\)](#), [l4\\_timeout\(\)](#), [l4\\_timeout\\_abs\(\)](#), [l4\\_timeout\\_get\(\)](#), [l4\\_timeout\\_is\\_absolute\(\)](#), [l4\\_timeout\\_rel\(\)](#),  
[l4\\_timeout\\_rel\\_get\(\)](#), [l4\\_touch\\_ro\(\)](#), [l4\\_touch\\_rw\(\)](#), [l4\\_trunc\\_page\(\)](#), [l4\\_trunc\\_size\(\)](#), [l4\\_usleep\(\)](#), [l4\\_utcb\(\)](#),  
[l4\\_utcb\\_br\(\)](#), [l4\\_utcb\\_exc\(\)](#), [l4\\_utcb\\_exc\\_is\\_ex\\_regs\\_exception\(\)](#), [l4\\_utcb\\_exc\\_is\\_ex\\_regs\\_exception\(\)](#), [l4\\_utcb\\_exc\\_is\\_pf\(\)](#),  
[l4\\_utcb\\_exc\\_is\\_pf\(\)](#), [l4\\_utcb\\_exc\\_pc\(\)](#), [l4\\_utcb\\_exc\\_pc\(\)](#), [l4\\_utcb\\_exc\\_pc\\_set\(\)](#), [l4\\_utcb\\_exc\\_pfa\(\)](#), [l4\\_utcb\\_exc\\_pfa\(\)](#),  
[l4\\_utcb\\_exc\\_typeval\(\)](#), [l4\\_utcb\\_exc\\_typeval\(\)](#), [l4\\_utcb\\_inherit\\_fpu\(\)](#), [l4\\_utcb\\_mr\(\)](#), [l4\\_utcb\\_mr64\\_idx\(\)](#), [l4\\_utcb\\_tcr\(\)](#),  
[l4\\_vcon\\_get\\_attr\(\)](#), [l4\\_vcon\\_get\\_attr\\_u\(\)](#), [l4\\_vcon\\_read\(\)](#), [l4\\_vcon\\_read\\_u\(\)](#), [l4\\_vcon\\_read\\_with\\_flags\(\)](#),  
[l4\\_vcon\\_send\(\)](#), [l4\\_vcon\\_send\\_u\(\)](#), [l4\\_vcon\\_set\\_attr\(\)](#), [l4\\_vcon\\_set\\_attr\\_raw\(\)](#), [l4\\_vcon\\_set\\_attr\\_u\(\)](#), [l4\\_vcon\\_write\(\)](#),  
[l4\\_vcon\\_write\\_u\(\)](#), [l4\\_vcpu\\_check\\_version\(\)](#), [l4\\_vm\\_vmx\\_clear\(\)](#), [l4\\_vm\\_vmx\\_field\\_len\(\)](#), [l4\\_vm\\_vmx\\_field\\_order\(\)](#),  
[l4\\_vm\\_vmx\\_get\\_caps\(\)](#), [l4\\_vm\\_vmx\\_get\\_caps\\_default1\(\)](#), [l4\\_vm\\_vmx\\_get\\_cr2\\_index\(\)](#), [l4\\_vm\\_vmx\\_get\\_hw\\_vmcs\(\)](#),  
[l4\\_vm\\_vmx\\_ptr\\_load\(\)](#), [l4\\_vm\\_vmx\\_read\(\)](#), [l4\\_vm\\_vmx\\_read\\_16\(\)](#), [l4\\_vm\\_vmx\\_read\\_32\(\)](#), [l4\\_vm\\_vmx\\_read\\_64\(\)](#),  
[l4\\_vm\\_vmx\\_read\\_nat\(\)](#), [l4\\_vm\\_vmx\\_set\\_hw\\_vmcs\(\)](#), [l4\\_vm\\_vmx\\_write\(\)](#), [l4\\_vm\\_vmx\\_write\\_16\(\)](#), [l4\\_vm\\_vmx\\_write\\_32\(\)](#),  
[l4\\_vm\\_vmx\\_write\\_64\(\)](#), [l4\\_vm\\_vmx\\_write\\_nat\(\)](#), [l4re\\_debug\\_obj\\_debug\(\)](#), [l4re\\_dma\\_space\\_associate\(\)](#), [l4re\\_dma\\_space\\_map\(\)](#),  
[l4re\\_dma\\_space\\_unmap\(\)](#), [l4re\\_ds\\_allocate\(\)](#), [l4re\\_ds\\_clear\(\)](#), [l4re\\_ds\\_copy\\_in\(\)](#), [l4re\\_ds\\_flags\(\)](#), [l4re\\_ds\\_info\(\)](#),  
[l4re\\_ds\\_map\\_info\(\)](#), [l4re\\_ds\\_size\(\)](#), [l4re\\_env\(\)](#), [l4re\\_env\\_cap\\_entry\\_t::l4re\\_env\\_cap\\_entry\\_t\(\)](#), [l4re\\_env\\_cap\\_entry\\_t::l4re\\_env\\_cap](#),  
[l4re\\_env\\_get\\_cap\(\)](#), [l4re\\_env\\_get\\_cap\\_e\(\)](#), [l4re\\_env\\_get\\_cap\\_l\(\)](#), [l4re\\_event\\_get\\_axis\\_info\(\)](#), [l4re\\_event\\_get\\_buffer\(\)](#),  
[l4re\\_event\\_get\\_num\\_streams\(\)](#), [l4re\\_event\\_get\\_stream\\_info\(\)](#), [l4re\\_event\\_get\\_stream\\_info\\_for\\_id\(\)](#), [l4re\\_kip\(\)](#),  
[l4re\\_log\\_print\(\)](#), [l4re\\_log\\_print\\_srv\(\)](#), [l4re\\_log\\_printn\(\)](#), [l4re\\_log\\_printn\\_srv\(\)](#), [l4re\\_ma\\_alloc\(\)](#), [l4re\\_ma\\_alloc\\_align\(\)](#),  
[l4re\\_ma\\_alloc\\_align\\_srv\(\)](#), [l4re\\_ns\\_query\\_srv\(\)](#), [l4re\\_ns\\_query\\_to\\_srv\(\)](#), [l4re\\_ns\\_register\\_obj\\_srv\(\)](#), [l4re\\_rm\\_attach\(\)](#),  
[l4re\\_rm\\_attach\\_srv\(\)](#), [l4re\\_rm\\_detach\(\)](#), [l4re\\_rm\\_detach\\_ds\(\)](#), [l4re\\_rm\\_detach\\_ds\\_unmap\(\)](#), [l4re\\_rm\\_detach\\_srv\(\)](#),  
[l4re\\_rm\\_detach\\_unmap\(\)](#), [l4re\\_rm\\_find\(\)](#), [l4re\\_rm\\_find\\_srv\(\)](#), [l4re\\_rm\\_free\\_area\(\)](#), [l4re\\_rm\\_free\\_area\\_srv\(\)](#),  
[l4re\\_rm\\_get\\_info\(\)](#), [l4re\\_rm\\_get\\_info\\_srv\(\)](#), [l4re\\_rm\\_reserve\\_area\(\)](#), [l4re\\_rm\\_reserve\\_area\\_srv\(\)](#), [l4re\\_rm\\_show\\_lists\(\)](#),  
[l4re\\_rm\\_show\\_lists\\_srv\(\)](#), [l4re\\_util\\_cap\\_alloc\(\)](#), [l4re\\_util\\_cap\\_free\(\)](#), [l4re\\_util\\_cap\\_free\\_um\(\)](#), [l4re\\_util\\_cap\\_last\(\)](#),  
[l4re\\_util\\_kumem\\_alloc\(\)](#), [l4re\\_video\\_goos\\_create\\_buffer\(\)](#), [l4re\\_video\\_goos\\_create\\_view\(\)](#), [l4re\\_video\\_goos\\_delete\\_buffer\(\)](#),  
[l4re\\_video\\_goos\\_delete\\_view\(\)](#), [l4re\\_video\\_goos\\_get\\_static\\_buffer\(\)](#), [l4re\\_video\\_goos\\_get\\_view\(\)](#), [l4re\\_video\\_goos\\_info\(\)](#),  
[l4re\\_video\\_goos\\_refresh\(\)](#), [l4re\\_video\\_view\\_get\\_info\(\)](#), [l4re\\_video\\_view\\_refresh\(\)](#), [l4re\\_video\\_view\\_set\\_info\(\)](#),  
[l4re\\_video\\_view\\_set\\_viewport\(\)](#), [l4re\\_video\\_view\\_stack\(\)](#), [l4util\\_micros2l4to\(\)](#), [l4vcpu\\_ext\\_alloc\(\)](#), [l4vcpu\\_irq\\_disable\(\)](#),  
[l4vcpu\\_irq\\_disable\\_save\(\)](#), [l4vcpu\\_irq\\_enable\(\)](#), [l4vcpu\\_irq\\_restore\(\)](#), [l4vcpu\\_is\\_irq\\_entry\(\)](#), [l4vcpu\\_is\\_page\\_fault\\_entry\(\)](#),  
[l4vcpu\\_print\\_state\(\)](#), [l4vcpu\\_wait\\_for\\_event\(\)](#), [l4virtio\\_config\\_queue\(\)](#), [l4virtio\\_device\\_config\\_ds\(\)](#), [l4virtio\\_device\\_notification\\_irq\(\)](#),  
[l4virtio\\_register\\_ds\(\)](#), [l4virtio\\_set\\_status\(\)](#), [l4\\_msgtag\\_t::label\(\)](#), [l4\\_msgtag\\_t::label\(\)](#), and [l4\\_msgtag\\_t::words\(\)](#).

### 14.1.2.3 Function Documentation

#### 14.1.2.3.1 l4\_align\_stack\_for\_direct\_fncall()

```

unsigned long l4_align_stack_for_direct_fncall (
    unsigned long stack) [inline]
  
```

Specify the desired alignment of the stack pointer.

**BIGGEST\_ALIGNMENT** provides the largest alignment ever used for any data type on the target machine. This is normally identical to desired stack alignment. Align stack pointer for directly invoked functions.

The stack needs to be aligned to `L4_STACK_ALIGN` for being able to access certain data on the stack. On x86/AMD64, a function call is performed using the 'call' instruction decrementing the stack pointer and writing the return address onto the stack. The called function considers this when adapting the stack pointer after function entry. If the called function was not invoked by a 'call' instruction, the stack pointer is actually off by a machine word leading to stack alignment issues when executing SSE instructions.

This function fixes the stack pointer for directly invoked functions. For architectures not automatically pushing the stack pointer during a function call, just enforce the `L4_STACK_ALIGN` alignment.

Definition at line 273 of file [compiler.h](#).

References [L4\\_INLINE](#).

#### 14.1.2.3.2 l4\_infinite\_loop()

```
L4_NORETURN void l4_infinite_loop (
    void ) [inline]
```

Infinite loop.

Will never return. Use [l4\\_sleep\\_forever\(\)](#) if at all possible.

Definition at line 347 of file [compiler.h](#).

References [l4\\_barrier\(\)](#), [L4\\_INLINE](#), and [L4\\_NORETURN](#).

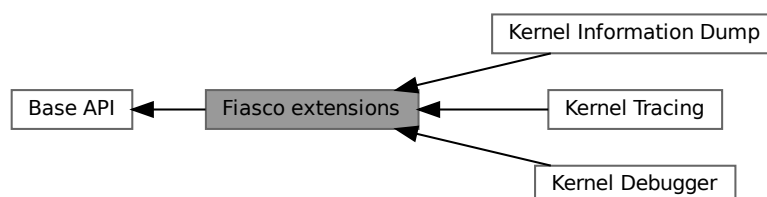
Here is the call graph for this function:



#### 14.1.3 Fiasco extensions

Extensions of the Fiasco [L4](#) implementation.

Collaboration diagram for Fiasco extensions:



## Topics

- Kernel Debugger . . . . . 169  
*Kernel debugger related functionality.*
- Kernel Information Dump . . . . . 178  
*Kernel information dumping related functionality.*
- Kernel Tracing . . . . . 179  
*Kernel tracing related functionality.*

## Files

- file [\\_\\_ktrace-impl.h](#)  
*L4 kernel event tracing.*
- file [ktrace.h](#)  
*L4 kernel event tracing.*
- file [obj\\_info.h](#)  
*Debugger related functions.*
- file [segment.h](#)  
*I4f-specific fs/gs manipulation (AMD64).*
- file [segment.h](#)  
*Segment handling (AMD64).*
- file [segment.h](#)  
*I4f-specific segment manipulation (x86).*
- file [segment.h](#)  
*Segment handling (x86).*

## Functions

- long [fiasco\\_ldt\\_set](#) ([l4\\_cap\\_idx\\_t](#) task, void \*ldt, unsigned int num\_desc, unsigned int entry\_number\_start, [l4\\_utcb\\_t](#) \*utcb)  
*Set LDT segments descriptors.*
- long [fiasco\\_gdt\\_set](#) ([l4\\_cap\\_idx\\_t](#) thread, void \*desc, unsigned int size, unsigned int entry\_number\_start, [l4\\_utcb\\_t](#) \*utcb)  
*Set GDT segment descriptors.*
- unsigned [fiasco\\_gdt\\_get\\_entry\\_offset](#) ([l4\\_cap\\_idx\\_t](#) thread, [l4\\_utcb\\_t](#) \*utcb)  
*Return the offset of the entry in the GDT.*

### 14.1.3.1 Detailed Description

Extensions of the Fiasco L4 implementation.

### 14.1.3.2 Function Documentation

#### 14.1.3.2.1 fiasco\_gdt\_get\_entry\_offset()

```
unsigned fiasco_gdt_get_entry_offset (
    l4\_cap\_idx\_t thread,
    l4\_utcb\_t * utcb) [inline]
```

Return the offset of the entry in the GDT.

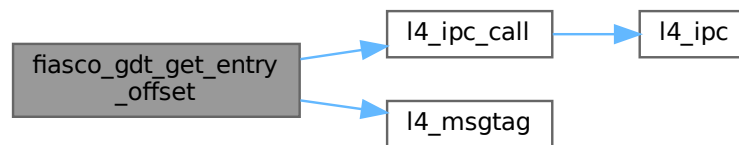
## Parameters

<i>thread</i>	Thread to get info from.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

Definition at line 166 of file [segment.h](#).

References [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_THREAD](#), and [L4\\_THREAD\\_X86\\_GDT\\_OP](#).

Here is the call graph for this function:



#### 14.1.3.2.2 fiasco\_gdt\_set()

```

long fiasco_gdt_set (
    l4_cap_idx_t thread,
    void * desc,
    unsigned int size,
    unsigned int entry_number_start,
    l4_utcb_t * utcb) [inline]
  
```

Set GDT segment descriptors.

Fiasco supports 4 consecutive entries, starting at the value returned by [fiasco\\_gdt\\_get\\_entry\\_offset\(\)](#).

#### Parameters

<i>thread</i>	Thread to set the GDT entry for.
<i>desc</i>	Pointer to GDT descriptors.
<i>size</i>	Size of the descriptors in bytes (multiple of 8).
<i>entry_number_start</i>	Entry number to start (valid values: 0-3).
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

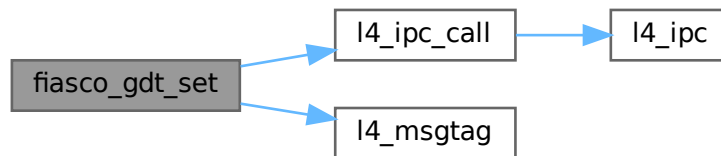
#### Return values

<0	At least one provided GDT descriptor is considered unsafe by the kernel, and not all selected GDT descriptors have been updated.
<i>L4_EOK</i>	Success.

Definition at line 41 of file [segment.h](#).

References [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_THREAD](#), and [L4\\_THREAD\\_X86\\_GDT\\_OP](#).

Here is the call graph for this function:



#### 14.1.3.2.3 fiasco\_ldt\_set()

```

long fiasco_ldt_set (
    l4_cap_idx_t task,
    void * ldt,
    unsigned int num_desc,
    unsigned int entry_number_start,
    l4_utcb_t * utcb) [inline]
  
```

Set LDT segments descriptors.

##### Parameters

<i>task</i>	Task to set the segment for.
<i>ldt</i>	Pointer to LDT hardware descriptors.
<i>num_desc</i>	Number of descriptors.
<i>entry_number_start</i>	Entry number to start.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

##### Return values

<code>-L4_ENOSYS</code>	The kernel configuration doesn't support this feature.
<code>-L4_EINVAL</code>	Invalid descriptor or invalid entry number.
<code>L4_EOK</code>	Success.



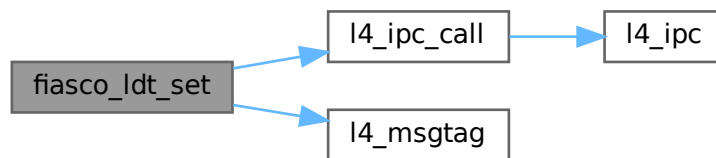
**Note**

This feature is not available if the kernel is configured with page table isolation.

Definition at line 153 of file [segment.h](#).

References [L4\\_EINVAL](#), [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_TASK](#), [L4\\_TASK\\_LDT\\_SET\\_X86\\_OP](#), [L4\\_TASK\\_LDT\\_X86\\_ENTRY\\_SIZE](#), and [L4\\_TASK\\_LDT\\_X86\\_MAX\\_ENTRIES](#).

Here is the call graph for this function:

**14.1.3.3 Kernel Debugger**

Kernel debugger related functionality.

Collaboration diagram for Kernel Debugger:

**Files**

- file [kdebug.h](#)

*Functionality for invoking the kernel debugger.*

## Functions

- `l4_msgtag_t l4_debugger_set_object_name (l4_cap_idx_t cap, const char *name) L4_NOTHROW`  
*Set the name of a kernel object.*
- `l4_msgtag_t l4_debugger_get_object_name (l4_cap_idx_t cap, unsigned id, char *name, unsigned size) L4_NOTHROW`  
*Get name of the kernel object with id `id`.*
- `unsigned long l4_debugger_global_id (l4_cap_idx_t cap) L4_NOTHROW`  
*Get the globally unique ID of the object behind a capability.*
- `unsigned long l4_debugger_kobj_to_id (l4_cap_idx_t cap, l4_addr_t kobjp) L4_NOTHROW`  
*Get the globally unique ID of the object behind the kobject pointer.*
- `long l4_debugger_query_log_typeid (l4_cap_idx_t cap, const char *name, unsigned idx) L4_NOTHROW`  
*Query the log-id for a log type.*
- `long l4_debugger_query_log_name (l4_cap_idx_t cap, unsigned idx, char *name, unsigned namelen, char *shortname, unsigned shortnamelen) L4_NOTHROW`  
*Query the name of a log type given the ID.*
- `l4_msgtag_t l4_debugger_switch_log (l4_cap_idx_t cap, const char *name, int on_off) L4_NOTHROW`  
*Set or unset log.*
- `l4_msgtag_t l4_debugger_add_image_info (l4_cap_idx_t cap, l4_addr_t base, const char *name) L4_NOTHROW`  
*Add loaded image information for a task.*

### 14.1.3.3.1 Detailed Description

Kernel debugger related functionality.

#### Attention

This API is subject to change!

This is a debugging facility, any call to any function might be invalid. Do not rely on it in any real code.

#### Include File

```
#include <l4/sys/debugger.h>
```

### 14.1.3.3.2 Function Documentation

#### 14.1.3.3.2.1 l4\_debugger\_add\_image\_info()

```
l4_msgtag_t l4_debugger_add_image_info (
    l4_cap_idx_t cap,
    l4_addr_t base,
    const char * name) [inline]
```

Add loaded image information for a task.

#### Parameters

---

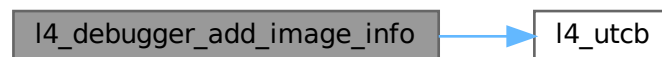
<i>cap</i>	Capability which refers to the task object.
<i>base</i>	Load base address of image.
<i>name</i>	Image base name.

This is a debugging facility, the call might be invalid.

Definition at line 417 of file [debugger.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.3.3.2.2 l4\_debugger\_get\_object\_name()

```

l4_msgtag_t l4_debugger_get_object_name (
    l4_cap_idx_t cap,
    unsigned id,
    char * name,
    unsigned size) [inline]
  
```

Get name of the kernel object with Id *id*.

##### Parameters

	<i>cap</i>	Capability of the debugger object.
	<i>id</i>	Global id of the object whose name is asked.
out	<i>name</i>	<a href="#">Buffer</a> to copy the name into. The buffer must be allocated by the caller.
	<i>size</i>	Length of the <i>name</i> buffer.

##### Returns

Syscall return tag

Definition at line 410 of file [debugger.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.3.3.2.3 l4\_debugger\_global\_id()

```
unsigned long l4_debugger_global_id (
    l4_cap_idx_t cap) [inline]
```

Get the globally unique ID of the object behind a capability.

##### Parameters

<i>cap</i>	Capability
------------	------------

##### Return values

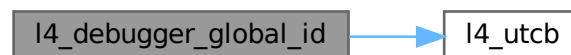
<i>~0UL</i>	Capability is not valid.
<i>otherwise</i>	Global debugger id.

This is a debugging facility, the call might be invalid.

Definition at line 375 of file [debugger.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.3.3.2.4 l4\_debugger\_kobj\_to\_id()

```
unsigned long l4_debugger_kobj_to_id (  
    l4_cap_idx_t cap,  
    l4_addr_t kobjp) [inline]
```

Get the globally unique ID of the object behind the kobject pointer.

#### Parameters

---

<i>cap</i>	Capability
<i>kobjp</i>	Kobject pointer

### Return values

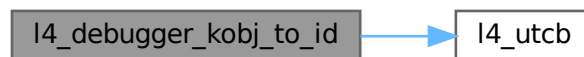
<i>~0UL</i>	The capability or the kobject pointer are invalid.
<i>otherwise</i>	The globally unique id.

This is a debugging facility, the call might be invalid.

Definition at line 381 of file [debugger.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.3.3.2.5 l4\_debugger\_query\_log\_name()

```

long l4_debugger_query_log_name (
    l4_cap_idx_t cap,
    unsigned idx,
    char * name,
    unsigned namelen,
    char * shortname,
    unsigned shortnamelen) [inline]
  
```

Query the name of a log type given the ID.

### Parameters

<i>cap</i>	Debugger capability.
<i>idx</i>	ID to query.
<i>name</i>	<a href="#">Buffer</a> to copy name to.
<i>namelen</i>	<a href="#">Buffer</a> length of name.
<i>shortname</i>	<a href="#">Buffer</a> to copy shortname to.
<i>shortnamelen</i>	<a href="#">Buffer</a> length of shortname.

### Return values

0	Success
<0	Error

This is a debugging facility, the call might be invalid.

Definition at line 394 of file [debugger.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.3.3.2.6 l4\_debugger\_query\_log\_typeid()

```

long l4_debugger_query_log_typeid (
    l4_cap_idx_t cap,
    const char * name,
    unsigned idx) [inline]
  
```

Query the log-id for a log type.

##### Parameters

<i>cap</i>	Debugger capability
<i>name</i>	Name to query for.
<i>idx</i>	Idx to start searching, start with 0

##### Returns

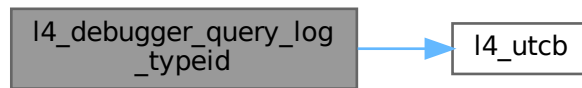
positive ID, or negative error code

This is a debugging facility, the call might be invalid.

Definition at line 387 of file [debugger.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.3.3.2.7 l4\_debugger\_set\_object\_name()

```

l4_msgtag_t l4_debugger_set_object_name (
    l4_cap_idx_t cap,
    const char * name) [inline]
  
```

Set the name of a kernel object.

##### Parameters

<i>cap</i>	Capability which refers to the kernel object.
<i>name</i>	Name of the kernel object that is e.g. displayed in the kernel debugger.

This is a debugging facility, the call might be invalid.

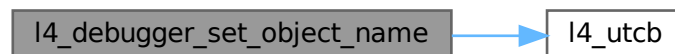
##### Examples

[examples/libs/shmc/prodcons.c](#), and [examples/sys/aliens/main.c](#).

Definition at line 368 of file [debugger.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:





#### 14.1.3.3.2.8 l4\_debugger\_switch\_log()

```
l4_msgtag_t l4_debugger_switch_log (  
    l4_cap_idx_t cap,  
    const char * name,  
    int on_off) [inline]
```

Set or unset log.

#### Parameters

---

<i>cap</i>	Debugger object.
<i>name</i>	Name of the log type.
<i>on_off</i>	1: turn log on, 0: turn log off

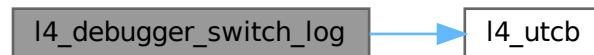
#### Returns

Syscall return tag

Definition at line 403 of file [debugger.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.3.4 Kernel Information Dump

Kernel information dumping related functionality.

Collaboration diagram for Kernel Information Dump:



Kernel information dumping related functionality.

Functions that dump various kernel internal information to the console. Probably only present in kernel debug builds.

#### Include File

```
#include <l4/sys/kdump.h>
```

### 14.1.3.5 Kernel Tracing

Kernel tracing related functionality.

Collaboration diagram for Kernel Tracing:



#### Functions

- `l4_umword_t fiasco_tbuf_log` (const char \*text)  
*Create new trace-buffer entry with describing <text>.*
- `l4_umword_t fiasco_tbuf_log_3val` (const char \*text, `l4_umword_t` v1, `l4_umword_t` v2, `l4_umword_t` v3)  
*Create new trace-buffer entry with describing <text> and three additional values.*
- `l4_umword_t fiasco_tbuf_log_binary` (const unsigned char \*data)  
*Create new trace-buffer entry with binary data.*
- void `fiasco_tbuf_clear` (void)  
*Clear trace-buffer.*
- void `fiasco_tbuf_dump` (void)  
*Dump trace-buffer to kernel console.*

#### 14.1.3.5.1 Detailed Description

Kernel tracing related functionality.

##### Attention

This API is subject to change!

This is a tracing facility for the Fiasco kernel trace buffer. Any call to any function might be invalid. Do not rely on it in any real code.

##### Include File

```
#include <l4/sys/ktrace.h>
```

#### 14.1.3.5.2 Function Documentation

##### 14.1.3.5.2.1 fiasco\_tbuf\_log()

```
l4_umword_t fiasco_tbuf_log (
    const char * text) [inline]
```

Create new trace-buffer entry with describing <text>.

##### Parameters

---

<i>text</i>	Logging text
-------------	--------------

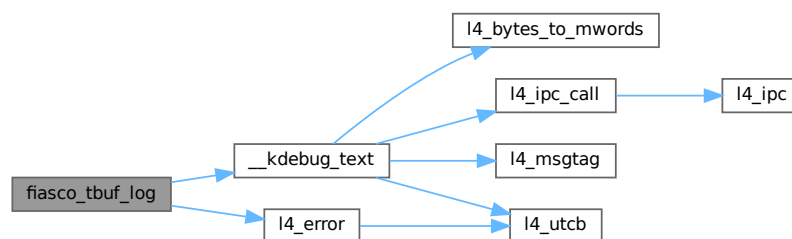
### Returns

Pointer to trace-buffer entry

Definition at line 24 of file [\\_\\_ktrace-impl.h](#).

References [\\_\\_kdebug\\_text\(\)](#), and [l4\\_error\(\)](#).

Here is the call graph for this function:



#### 14.1.3.5.2.2 fiasco\_tbuf\_log\_3val()

```

l4_umword_t fiasco_tbuf_log_3val (
    const char * text,
    l4_umword_t v1,
    l4_umword_t v2,
    l4_umword_t v3) [inline]
  
```

Create new trace-buffer entry with describing <text> and three additional values.

### Parameters

<i>text</i>	Logging text
<i>v1</i>	first value
<i>v2</i>	second value
<i>v3</i>	third value

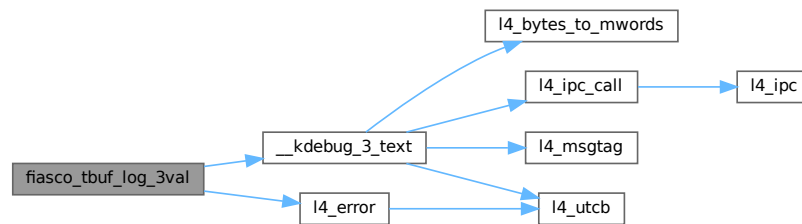
**Returns**

Pointer to trace-buffer entry

Definition at line 31 of file [\\_\\_ktrace-impl.h](#).

References [\\_\\_kdebug\\_3\\_text\(\)](#), and [l4\\_error\(\)](#).

Here is the call graph for this function:

**14.1.3.5.2.3 fiasco\_tbuf\_log\_binary()**

```
l4_umword_t fiasco_tbuf_log_binary (
    const unsigned char * data) [inline]
```

Create new trace-buffer entry with binary data.

**Parameters**

<i>data</i>	binary data
-------------	-------------

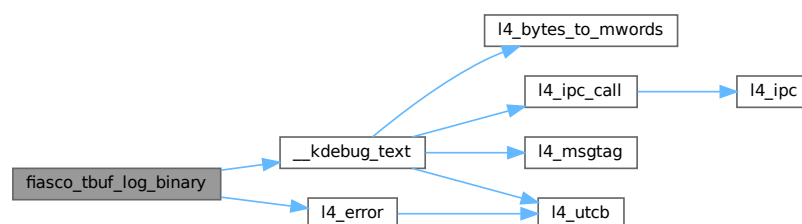
**Returns**

Pointer to trace-buffer entry

Definition at line 54 of file [\\_\\_ktrace-impl.h](#).

References [\\_\\_kdebug\\_text\(\)](#), and [l4\\_error\(\)](#).

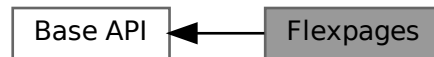
Here is the call graph for this function:



### 14.1.4 Flexpages

Flexpage-related API.

Collaboration diagram for Flexpages:



#### Data Structures

- union `l4_fpage_t`  
*L4 flexpage type.*

#### Enumerations

- enum `L4_fpage_consts` {  
`L4_FPAGE_RIGHTS_SHIFT` = 0 , `L4_FPAGE_TYPE_SHIFT` = 4 , `L4_FPAGE_SIZE_SHIFT` = 6 ,  
`L4_FPAGE_ADDR_SHIFT` = 12 ,  
`L4_FPAGE_RIGHTS_BITS` = 4 , `L4_FPAGE_TYPE_BITS` = 2 , `L4_FPAGE_SIZE_BITS` = 6 , `L4_FPAGE_ADDR_BITS`  
= `L4_MWORD_BITS` - `L4_FPAGE_ADDR_SHIFT` ,  
`L4_FPAGE_RIGHTS_MASK` , `L4_FPAGE_TYPE_MASK` , `L4_FPAGE_SIZE_MASK` , `L4_FPAGE_ADDR_`  
`_MASK` = `~0UL << L4_FPAGE_ADDR_SHIFT` ,  
`L4_FPAGE_RIGHTS_ALL` = `L4_FPAGE_RIGHTS_MASK` }  
*L4 flexpage structure.*
- enum { `L4_WHOLE_ADDRESS_SPACE` = 63 }  
*Constants for flexpages.*
- enum `L4_fpage_rights` {  
`L4_FPAGE_X` = 1 , `L4_FPAGE_W` = 2 , `L4_FPAGE_RO` = 4 , `L4_FPAGE_RW` = `L4_FPAGE_RO` | `L4_`  
`FPAGE_W` ,  
`L4_FPAGE_RX` = `L4_FPAGE_RO` | `L4_FPAGE_X` , `L4_FPAGE_RWX` = `L4_FPAGE_RW` | `L4_FPAGE_X` }  
*Memory and IO port flexpage rights.*
- enum `L4_cap_fpage_rights` {  
`L4_CAP_FPAGE_W` = 0x1 , `L4_CAP_FPAGE_S` = 0x2 , `L4_CAP_FPAGE_R` = 0x4 , `L4_CAP_FPAGE_RO` =  
0x4 ,  
`L4_CAP_FPAGE_D` = 0x8 , `L4_CAP_FPAGE_RW` = `L4_CAP_FPAGE_R` | `L4_CAP_FPAGE_W` ,  
`L4_CAP_FPAGE_RS` = `L4_CAP_FPAGE_R` | `L4_CAP_FPAGE_S` , `L4_CAP_FPAGE_RWS` = `L4_CAP_`  
`_FPAGE_RW` | `L4_CAP_FPAGE_S` ,  
`L4_CAP_FPAGE_RWSD` = `L4_CAP_FPAGE_RWS` | `L4_CAP_FPAGE_D` , `L4_CAP_FPAGE_RWD` = `L4_`  
`_CAP_FPAGE_RW` | `L4_CAP_FPAGE_D` , `L4_CAP_FPAGE_RSD` = `L4_CAP_FPAGE_RS` | `L4_CAP_`  
`FPAGE_D` }  
*Object flexpage rights.*
- enum `L4_fpage_type` { `L4_FPAGE_SPECIAL` = 0 , `L4_FPAGE_MEMORY` = 1 , `L4_FPAGE_IO` = 2 ,  
`L4_FPAGE_OBJ` = 3 }  
*Flexpage type.*

- enum `L4_fpage_control` { `L4_FPAGE_CONTROL_OFFSET_SHIFT` = 12 , `L4_FPAGE_CONTROL_MASK` = `~0UL << L4_FPAGE_CONTROL_OFFSET_SHIFT` }  
*Flexpage map control flags.*
- enum { `L4_WHOLE_IOADDRESS_SPACE` = 16 , `L4_IOPORT_MAX` = (1L << `L4_WHOLE_IOADDRESS_SPACE`) }  
*Special constants for IO flexpages.*

## Functions

- `l4_fpage_t l4_fpage` (`l4_addr_t` address, unsigned int order, unsigned char rights) `L4_NOTHROW`  
*Create a memory flexpage.*
- `l4_fpage_t l4_fpage_all` (void) `L4_NOTHROW`  
*Get a flexpage, describing all address spaces at once.*
- `l4_fpage_t l4_fpage_invalid` (void) `L4_NOTHROW`  
*Get an invalid flexpage.*
- `l4_fpage_t l4_iofpage` (unsigned long port, unsigned int order) `L4_NOTHROW`  
*Create an IO-port flexpage.*
- `l4_fpage_t l4_obj_fpage` (`l4_cap_idx_t` obj, unsigned int order, unsigned char rights) `L4_NOTHROW`  
*Create a kernel-object flexpage.*
- int `l4_is_fpage_writable` (`l4_fpage_t` fp) `L4_NOTHROW`  
*Test if the flexpage is writable.*
- unsigned `l4_fpage_rights` (`l4_fpage_t` f) `L4_NOTHROW`  
*Return rights from a flexpage.*
- unsigned `l4_fpage_type` (`l4_fpage_t` f) `L4_NOTHROW`  
*Return type from a flexpage.*
- unsigned `l4_fpage_size` (`l4_fpage_t` f) `L4_NOTHROW`  
*Return size (log2) from a flexpage.*
- unsigned long `l4_fpage_page` (`l4_fpage_t` f) `L4_NOTHROW`  
*Return the page part from a flexpage.*
- `l4_addr_t l4_fpage_memaddr` (`l4_fpage_t` f) `L4_NOTHROW`  
*Return the memory address from the memory flexpage.*
- `l4_cap_idx_t l4_fpage_obj` (`l4_fpage_t` f) `L4_NOTHROW`  
*Return the capability index from the object flexpage.*
- unsigned long `l4_fpage_ioport` (`l4_fpage_t` f) `L4_NOTHROW`  
*Return the IO port number from the IO flexpage.*
- `l4_fpage_t l4_fpage_set_rights` (`l4_fpage_t` src, unsigned char new\_rights) `L4_NOTHROW`  
*Set new right in a flexpage.*
- int `l4_fpage_contains` (`l4_fpage_t` fpage, `l4_addr_t` addr, unsigned order) `L4_NOTHROW`  
*Test whether a given range is completely within an fpage.*
- unsigned char `l4_fpage_max_order` (unsigned char order, `l4_addr_t` addr, `l4_addr_t` min\_addr, `l4_addr_t` max\_addr, `l4_addr_t` hotspot=0)  
*Determine maximum flexpage size of a region.*
- int `l4_is_fpage_valid` (`l4_fpage_t` fp) `L4_NOTHROW`  
*Test if the flexpage is valid.*

#### 14.1.4.1 Detailed Description

Flexpage-related API.

A flexpage is a page with a variable size, that can describe memory, IO-Ports (IA32 only), and sets of kernel objects.

A flexpage describes an always size aligned region of an address space. The size is given in a log2 scale. This means the size in elements (bytes for memory, ports for IO-Ports, and capabilities for kernel objects) is always a power of two.

A flexpage also carries type and access right information for the described region. The type information selects the address space in which the flexpage is valid. Access rights have a meaning depending on the specific address space (type).

There exists a special type for defining *receive windows* or for the [l4\\_task\\_unmap\(\)](#) method, that can be used to describe all address spaces (all types) with a single flexpage.

##### Include File

```
#include <l4/sys/types.h>
```

#### 14.1.4.2 Enumeration Type Documentation

##### 14.1.4.2.1 anonymous enum

anonymous enum

Constants for flexpages.

##### Enumerator

L4_WHOLE_ADDRESS_SPACE	Whole address space size. This value does not only specify the log2 size of the biggest possible memory flexpage. It can be also used as size for a special flexpage to define a flexpage which completely covers all spaces.
------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Definition at line 84 of file [\\_\\_l4\\_fpage.h](#).

##### 14.1.4.2.2 anonymous enum

anonymous enum

Special constants for IO flexpages.

##### Enumerator

L4_WHOLE_IOADDRESS_SPACE	Whole I/O address space size. In contrast to <a href="#">L4_WHOLE_ADDRESS_SPACE</a> , this value forms the log2 size of the biggest possible I/O flexpage.
--------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------



L4_IOPORT_MAX	Maximum I/O port address plus 1.
---------------	----------------------------------

Definition at line 314 of file [\\_\\_l4\\_fpage.h](#).

#### 14.1.4.2.3 L4\_cap\_fpage\_rights

```
enum L4_cap_fpage_rights
```

Object flexpage rights.

Capabilities are modified or transferred with map and unmap operations. For that, capabilities are wrapped into flexpage objects. The flexpage carries a set of rights the sender wants to hand over to the receiver along with the capability.

For the user only the 'S' and the 'W' right are visible. Other rights such as the 'D' right are internal to the corresponding kernel object and cannot be evaluated by the receiver.

Note that additional object attributes and permissions can be specified in a send item, see [L4\\_obj\\_fpage\\_ctl](#).

#### Note

A thread can also map a capability from its task's capability table with a reduced set of rights into another slot of its own capability table.

#### Enumerator

L4_CAP_FPAGE_W	Interface specific 'W' right for capability flexpages. The semantics of the 'W' right is defined by the protocol. For example in case of a dataspace cap, the 'W' right is needed to get a writable dataspace.
L4_CAP_FPAGE_S	Interface specific 'S' right for capability flexpages. The semantics of the 'S' right is defined by the interface. When transferring object capabilities via IPC, the kernel masks this right with the 'S' right of the capability used to address the IPC partner. Thus, the 'S' right of sent capabilities is only transferred if both the flexpage and the IPC gate or thread capability specifying the IPC partner have the 'S' right. For <a href="#">L4::Task::map()</a> , the 'S' right is only transferred if the flexpage, the source and destination task capabilities have the 'S' right.
L4_CAP_FPAGE_R	Read right for capability flexpages. This is always required, otherwise no capability is mapped.
L4_CAP_FPAGE_RO	Read right for capability flexpages. This is always required, otherwise no capability is mapped.
L4_CAP_FPAGE_D	Delete right for capability flexpages. This allows the receiver to delete the corresponding kernel object using unmap() regardless of other tasks still holding a capability to the kernel object. Such capabilities are set to an empty capability if the object is deleted.
L4_CAP_FPAGE_RW	Read and interface specific 'W' right for capability flexpages. The semantics of the 'W' right is defined by the interface.  See also  <a href="#">L4_CAP_FPAGE_W</a>

L4_CAP_FPAGE_RS	Read and interface specific 'S' right for capability flexpages. The semantics of the 'S' right is defined by the interface.  See also <a href="#">L4_CAP_FPAGE_S</a>
L4_CAP_FPAGE_RWS	Read, interface specific 'W', and 'S' rights for capability flexpages. The semantics of the 'W' and 'S' right are defined by the interface.  See also <a href="#">L4_CAP_FPAGE_R</a> , <a href="#">L4_CAP_FPAGE_W</a> , and <a href="#">L4_CAP_FPAGE_S</a>
L4_CAP_FPAGE_RWSD	Full rights for capability flexpages.  See also <a href="#">L4_CAP_FPAGE_R</a> , <a href="#">L4_CAP_FPAGE_W</a> , <a href="#">L4_CAP_FPAGE_S</a> , and <a href="#">L4_CAP_FPAGE_D</a>
L4_CAP_FPAGE_RWD	Read, write, and delete right for capability flexpages.  See also <a href="#">L4_CAP_FPAGE_R</a> , <a href="#">L4_CAP_FPAGE_W</a> , and <a href="#">L4_CAP_FPAGE_D</a>
L4_CAP_FPAGE_RSD	Read, 'S', and delete right for capability flexpages.  See also <a href="#">L4_CAP_FPAGE_R</a> , <a href="#">L4_CAP_FPAGE_S</a> , and <a href="#">L4_CAP_FPAGE_D</a>

Definition at line 148 of file [\\_\\_l4\\_fpage.h](#).

#### 14.1.4.2.4 L4\_fpage\_consts

enum [L4\\_fpage\\_consts](#)

[L4](#) flexpage structure.

##### Enumerator

L4_FPAGE_RIGHTS_SHIFT	Access permissions shift.
L4_FPAGE_TYPE_SHIFT	Flexpage type shift (memory, IO port, obj...).
L4_FPAGE_SIZE_SHIFT	Flexpage size shift (log2-based).
L4_FPAGE_ADDR_SHIFT	Page address shift.
L4_FPAGE_RIGHTS_BITS	Access permissions size.
L4_FPAGE_TYPE_BITS	Flexpage type size (memory, IO port, obj...).
L4_FPAGE_SIZE_BITS	Flexpage size size (log2-based).
L4_FPAGE_ADDR_BITS	Page address size.
L4_FPAGE_RIGHTS_MASK	Mask to get the flexpage rights.

L4_FPAGE_RIGHTS_ALL	Specify as flexpage rights during grant.
---------------------	------------------------------------------

Definition at line 48 of file [\\_\\_l4\\_fpage.h](#).

#### 14.1.4.2.5 L4\_fpage\_control

```
enum L4_fpage_control
```

Flexpage map control flags.

##### Enumerator

L4_FPAGE_CONTROL_OFFSET_SHIFT	Number of bits an index must be shifted or an address must be aligned to in the control word.
L4_FPAGE_CONTROL_MASK	Mask for truncating the lower bits of the send base or the index of the control word.

Definition at line 243 of file [\\_\\_l4\\_fpage.h](#).

#### 14.1.4.2.6 L4\_fpage\_rights

```
enum L4_fpage_rights
```

Memory and IO port flexpage rights.

For IO flexpages, bit 1 and bit 2 are a combined read/write right. In a map operation, the receiver receives the IO port capability when the sender possesses it and at least one of these bits is present. For an unmap operation, the absence of one of those bits is sufficient to unmap the IO port capability.

Note that more memory attributes can be specified in a send item, see [l4\\_fpage\\_cacheability\\_opt\\_t](#).

##### Enumerator

L4_FPAGE_X	Executable flexpage.
L4_FPAGE_W	Writable flexpage.
L4_FPAGE_RO	Read-only flexpage.
L4_FPAGE_RW	Read-write flexpage.
L4_FPAGE_RX	Read-execute flexpage.
L4_FPAGE_RWX	Read-write-execute flexpage.

Definition at line 118 of file [\\_\\_l4\\_fpage.h](#).

#### 14.1.4.2.7 L4\_fpage\_type

```
enum L4_fpage_type
```

Flexpage type.

##### Enumerator

L4_FPAGE_SPECIAL	Special flexpage, either <a href="#">l4_fpage_invalid()</a> or <a href="#">l4_fpage_all()</a> ; only supported by selected interfaces.
L4_FPAGE_MEMORY	Flexpage for memory spaces.
L4_FPAGE_IO	Flexpage for I/O port spaces.
L4_FPAGE_OBJ	Flexpage for object spaces.

Definition at line 230 of file [\\_\\_l4\\_fpage.h](#).

### 14.1.4.3 Function Documentation

#### 14.1.4.3.1 l4\_fpage()

```
l4_fpage_t l4_fpage (
    l4_addr_t address,
    unsigned int order,
    unsigned char rights) [inline]
```

Create a memory flexpage.

#### Parameters

<i>address</i>	Flexpage start address
<i>order</i>	Flexpage size (log2), <a href="#">L4_WHOLE_ADDRESS_SPACE</a> to specify the whole address space (with <i>address</i> 0). The minimum log2 size of a memory flexpage is defined by <a href="#">L4_LOG2_PAGESIZE</a> according to the size of the smallest virtual page supported by the MMU.
<i>rights</i>	Access rights, see <a href="#">L4_fpage_rights</a>

#### Returns

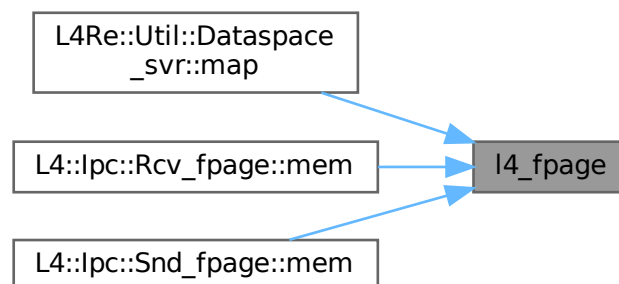
Memory flexpage

Definition at line 703 of file [\\_\\_l4\\_fpage.h](#).

References [L4\\_FPAGE\\_MEMORY](#), and [L4\\_NOTHROW](#).

Referenced by [L4Re::Util::Dataspace\\_svr::map\(\)](#), [L4::lpc::Rcv\\_fpage::mem\(\)](#), and [L4::lpc::Snd\\_fpage::mem\(\)](#).

Here is the caller graph for this function:



#### 14.1.4.3.2 l4\_fpage\_all()

```
l4_fpage_t l4_fpage_all (
    void ) [inline]
```

Get a flexpage, describing all address spaces at once.

##### Returns

Special *all-spaces* flexpage.

##### Note

This flexpage can be used to define a receive window where the sender can send objects of any type, or for an unmap item completely covering all spaces of the target task. It does not make sense to use this flexpage as send item.

Definition at line 723 of file [\\_\\_l4\\_fpage.h](#).

References [L4\\_FPAGE\\_SPECIAL](#), [L4\\_NOTHROW](#), and [L4\\_WHOLE\\_ADDRESS\\_SPACE](#).

#### 14.1.4.3.3 l4\_fpage\_contains()

```
int l4_fpage_contains (
    l4_fpage_t fpage,
    l4_addr_t addr,
    unsigned order) [inline]
```

Test whether a given range is completely within an fpage.

##### Parameters

<i>fpage</i>	Flexpage
<i>addr</i>	Address
<i>order</i>	Size of range in log2.

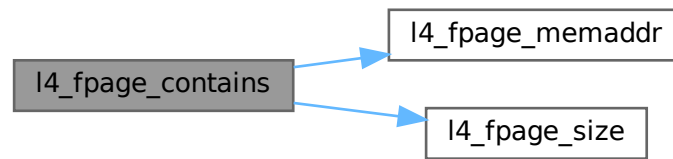
##### Return values

<code>==0</code>	The range is not completely in the fpage.
<code>!=0</code>	The range is within the fpage.

Definition at line 755 of file [\\_\\_l4\\_fpage.h](#).

References [l4\\_fpage\\_memaddr\(\)](#), [l4\\_fpage\\_size\(\)](#), and [L4\\_NOTHROW](#).

Here is the call graph for this function:



#### 14.1.4.3.4 l4\_fpage\_invalid()

```
l4_fpage_t l4_fpage_invalid (
    void ) [inline]
```

Get an invalid flexpage.

##### Returns

Special *invalid* flexpage.

Definition at line 729 of file `__l4_fpage.h`.

References [L4\\_FPAGE\\_SPECIAL](#), and [L4\\_NOTHROW](#).

#### 14.1.4.3.5 l4\_fpage\_ioport()

```
unsigned long l4_fpage_ioport (
    l4_fpage_t f) [inline]
```

Return the IO port number from the IO flexpage.

##### Parameters

<i>f</i>	Flexpage
----------	----------

##### Returns

IO port number from the given IO flexpage.

##### Precondition

*f* must be an IO flexpage (`l4_fpage_type(f) == L4\_FPAGE\_IO`) and

The function does not enforce size alignment of the read memory address. The caller must ensure the input fpage is correct.

Definition at line 659 of file `__l4_fpage.h`.

References [L4\\_FPAGE\\_ADDR\\_SHIFT](#), and [L4\\_NOTHROW](#).

**14.1.4.3.6 l4\_fpage\_max\_order()**

```

unsigned char l4_fpage_max_order (
    unsigned char order,
    l4_addr_t addr,
    l4_addr_t min_addr,
    l4_addr_t max_addr,
    l4_addr_t hotspot = 0) [inline]

```

Determine maximum flexpage size of a region.

**Parameters**

<i>order</i>	Order value to start with (e.g. for memory L4_LOG2_PAGESIZE would be used)
<i>addr</i>	Address to be covered by the flexpage.
<i>min_addr</i>	Start of region / minimal address (including).
<i>max_addr</i>	End of region / maximal address (excluding).
<i>hotspot</i>	(Optional) hot spot.

**Returns**

Maximum order (log2-size) possible.

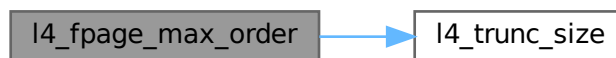
**Note**

The start address of the flexpage can be determined with `l4_trunc_size(addr, returnvalue)`

Definition at line 763 of file `__l4_fpage.h`.

References `l4_trunc_size()`.

Here is the call graph for this function:

**14.1.4.3.7 l4\_fpage\_memaddr()**

```

l4_addr_t l4_fpage_memaddr (
    l4_fpage_t f) [inline]

```

Return the memory address from the memory flexpage.

**Parameters**

<i>f</i>	Flexpage
----------	----------

**Returns**

Page address from the given memory flexpage.

**Precondition**

*f* must be a memory flexpage (`l4_fpage_type(f) == L4_FPAGE_MEMORY`).

The function does not enforce size alignment of the read memory address. The caller must ensure the input fpage is correct.

Definition at line 665 of file `__l4_fpage.h`.

References [L4\\_NOTHROW](#).

Referenced by [l4\\_fpage\\_contains\(\)](#).

Here is the caller graph for this function:

**14.1.4.3.8 l4\_fpage\_obj()**

```

l4_cap_idx_t l4_fpage_obj (
    l4_fpage_t f) [inline]
  
```

Return the capability index from the object flexpage.

**Parameters**

<i>f</i>	Flexpage
----------	----------

**Returns**

Capability index from the given object flexpage.

**Precondition**

*f* must be an object flexpage (`l4_fpage_type(f) == L4_FPAGE_OBJ`).

The function does not enforce size alignment of the read memory address. The caller must ensure the input fpage is correct.

Definition at line 671 of file `__l4_fpage.h`.

References [L4\\_NOTHROW](#).



#### 14.1.4.3.9 l4\_fpage\_page()

```
unsigned long l4_fpage_page (  
    l4_fpage_t f) [inline]
```

Return the page part from a flexpage.

##### Parameters

<i>f</i>	Flexpage
----------	----------

##### Returns

Page part of the given flexpage.

##### Note

The meaning of the page part depends on the flexpage type.

Definition at line 653 of file [\\_\\_l4\\_fpage.h](#).

References [L4\\_FPAGE\\_ADDR\\_SHIFT](#), and [L4\\_NOTHROW](#).

#### 14.1.4.3.10 l4\_fpage\_rights()

```
unsigned l4_fpage_rights (  
    l4_fpage_t f) [inline]
```

Return rights from a flexpage.

##### Parameters

<i>f</i>	Flexpage
----------	----------

##### Returns

Size part of the given flexpage.

Definition at line 635 of file [\\_\\_l4\\_fpage.h](#).

References [L4\\_FPAGE\\_RIGHTS\\_MASK](#), [L4\\_FPAGE\\_RIGHTS\\_SHIFT](#), and [L4\\_NOTHROW](#).

Referenced by [l4\\_is\\_fpage\\_writable\(\)](#).

Here is the caller graph for this function:



#### 14.1.4.3.11 l4\_fpage\_set\_rights()

```
l4_fpage_t l4_fpage_set_rights (
    l4_fpage_t src,
    unsigned char new_rights) [inline]
```

Set new right in a flexpage.

##### Parameters

<i>src</i>	Flexpage
<i>new_rights</i>	New rights

##### Returns

Modified flexpage with new rights.

Definition at line 694 of file [\\_\\_l4\\_fpage.h](#).

References [L4\\_FPAGE\\_RIGHTS\\_MASK](#), [L4\\_FPAGE\\_RIGHTS\\_SHIFT](#), [L4\\_NOTHROW](#), and [l4\\_fpage\\_t::raw](#).

Referenced by [L4::lpc::Snd\\_fpage::io\(\)](#).

Here is the caller graph for this function:



#### 14.1.4.3.12 l4\_fpage\_size()

```
unsigned l4_fpage_size (
    l4_fpage_t f) [inline]
```

Return size (log2) from a flexpage.

##### Parameters

<i>f</i>	Flexpage
----------	----------

##### Returns

Size part of the given flexpage.

See also

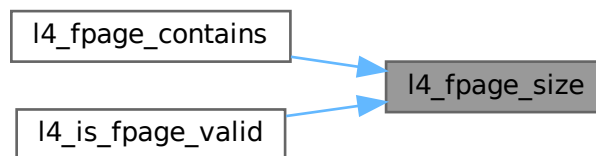
[l4\\_fpage\\_memaddr\(\)](#), [l4\\_fpage\\_obj\(\)](#), [l4\\_fpage\\_ioport\(\)](#)

Definition at line 647 of file [\\_\\_l4\\_fpage.h](#).

References [L4\\_FPAGE\\_SIZE\\_SHIFT](#), and [L4\\_NOTHROW](#).

Referenced by [l4\\_fpage\\_contains\(\)](#), and [l4\\_is\\_fpage\\_valid\(\)](#).

Here is the caller graph for this function:



#### 14.1.4.3.13 l4\_fpage\_type()

```

unsigned l4_fpage_type (
    l4_fpage_t f) [inline]
  
```

Return type from a flexpage.

##### Parameters

<i>f</i>	Flexpage
----------	----------

##### Returns

Type part of the given flexpage.

Definition at line 641 of file [\\_\\_l4\\_fpage.h](#).

References [L4\\_FPAGE\\_TYPE\\_SHIFT](#), and [L4\\_NOTHROW](#).

Referenced by [l4\\_is\\_fpage\\_valid\(\)](#).

Here is the caller graph for this function:



#### 14.1.4.3.14 l4\_iofpage()

```
l4_fpage_t l4_iofpage (
    unsigned long port,
    unsigned int order) [inline]
```

Create an IO-port flexpage.

##### Parameters

<i>port</i>	I/O-flexpage port base
<i>order</i>	I/O-flexpage size (log2), <a href="#">L4_WHOLE_IOADDRESS_SPACE</a> to specify the whole I/O address space (with <code>port 0</code> )

##### Returns

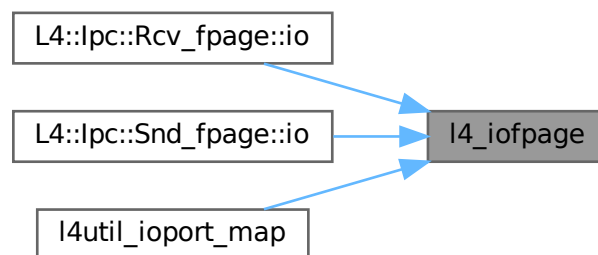
I/O flexpage

Definition at line 709 of file [\\_\\_l4\\_fpage.h](#).

References [L4\\_FPAGE\\_ADDR\\_SHIFT](#), [L4\\_FPAGE\\_IO](#), [L4\\_FPAGE\\_RW](#), and [L4\\_NOTHROW](#).

Referenced by [L4::lpc::Rcv\\_fpage::io\(\)](#), [L4::lpc::Snd\\_fpage::io\(\)](#), and [l4util\\_ioport\\_map\(\)](#).

Here is the caller graph for this function:



#### 14.1.4.3.15 l4\_is\_fpage\_valid()

```
int l4_is_fpage_valid (
    l4_fpage_t fp) [inline]
```

Test if the flexpage is valid.

##### Parameters

<i>fp</i>	Flexpage.
-----------	-----------

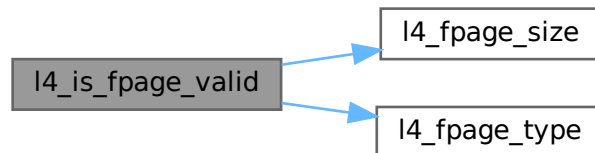
**Return values**

<i>!=0</i>	if flexpage is valid.
<i>==0</i>	if flexpage is not valid.

Definition at line 788 of file [\\_\\_l4\\_fpage.h](#).

References [l4\\_fpage\\_size\(\)](#), [L4\\_FPAGE\\_SPECIAL](#), [l4\\_fpage\\_type\(\)](#), and [L4\\_NOTHROW](#).

Here is the call graph for this function:

**14.1.4.3.16 l4\_is\_fpage\_writable()**

```
int l4_is_fpage_writable (
    l4_fpage_t fp) [inline]
```

Test if the flexpage is writable.

**Parameters**

<i>fp</i>	Flexpage.
-----------	-----------

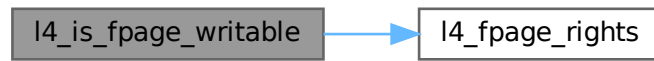
**Return values**

<i>!=0</i>	if flexpage is writable.
<i>==0</i>	if flexpage is not writable.

Definition at line 736 of file [\\_\\_l4\\_fpage.h](#).

References [l4\\_fpage\\_rights\(\)](#), [L4\\_FPAGE\\_W](#), and [L4\\_NOTHROW](#).

Here is the call graph for this function:



#### 14.1.4.3.17 l4\_obj\_fpage()

```

l4_fpage_t l4_obj_fpage (
    l4_cap_idx_t obj,
    unsigned int order,
    unsigned char rights) [inline]
  
```

Create a kernel-object flexpage.

##### Parameters

<i>obj</i>	Base capability selector.
<i>order</i>	Log2 size (number of capabilities).
<i>rights</i>	Access rights, see <a href="#">L4_cap_fpage_rights</a>

##### Returns

Flexpage for a set of kernel objects.

##### Note

[L4\\_CAP\\_FPAGE\\_R](#) is always required, otherwise no capability is mapped.

##### Examples

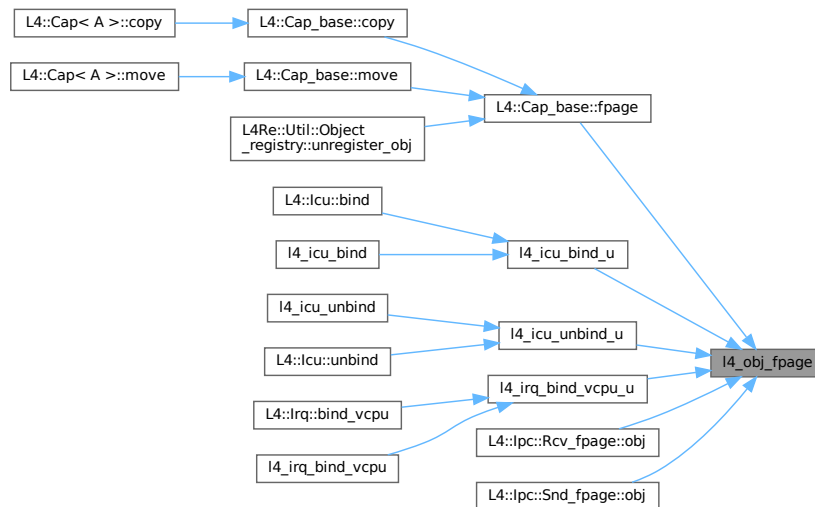
[examples/sys/utcb-ipc/main.c](#).

Definition at line 715 of file [\\_\\_l4\\_fpage.h](#).

References [L4\\_CAP\\_SHIFT](#), [L4\\_FPAGE\\_ADDR\\_SHIFT](#), [L4\\_FPAGE\\_OBJ](#), and [L4\\_NOTHROW](#).

Referenced by [L4::Cap\\_base::fpage\(\)](#), [l4\\_icu\\_bind\\_u\(\)](#), [l4\\_icu\\_unbind\\_u\(\)](#), [l4\\_irq\\_bind\\_vcpu\\_u\(\)](#), [L4::lpc::Rcv\\_fpage::obj\(\)](#), and [L4::lpc::Snd\\_fpage::obj\(\)](#).

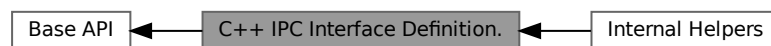
Here is the caller graph for this function:



### 14.1.5 C++ IPC Interface Definition.

APIs for defining IPC interfaces using C++ as language.

Collaboration diagram for C++ IPC Interface Definition.:



### Topics

- Internal Helpers . . . . . 200

### Namespaces

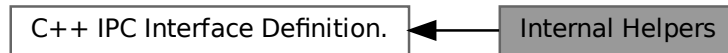
- namespace [L4::Typeid](#)  
Definition of interface data-type helpers.

#### 14.1.5.1 Detailed Description

APIs for defining IPC interfaces using C++ as language.

### 14.1.5.2 Internal Helpers

Collaboration diagram for Internal Helpers:



### Data Structures

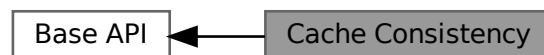
- struct [L4::Types::Bool< V >](#)  
*Boolean meta type.*
- struct [L4::Types::False](#)  
*False meta value.*
- struct [L4::Types::True](#)  
*True meta value.*
- struct [L4::Types::Same< A, B >](#)  
*Compare two data types for equality.*

#### 14.1.5.2.1 Detailed Description

### 14.1.6 Cache Consistency

Various functions for cache consistency.

Collaboration diagram for Cache Consistency:



### Functions

- [L4\\_BEGIN\\_DECLS](#) int [l4\\_cache\\_clean\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache clean a range in D-cache; writes back to PoC.*
- int [l4\\_cache\\_flush\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache flush a range; writes back to PoC.*
- int [l4\\_cache\\_inv\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache invalidate a range; might write back to PoC.*
- int [l4\\_cache\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent between I-cache and D-cache; writes back to PoU.*
- int [l4\\_cache\\_dma\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent for use with external memory; writes back to PoC.*
- int [l4\\_cache\\_dma\\_coherent\\_full](#) (void) [L4\\_NOTHROW](#)  
*Make memory coherent for use with external memory; writes back to PoC.*



### 14.1.6.1 Detailed Description

Various functions for cache consistency.

These functions shall be used to ensure that

- all blocks (e.g. CPU cores, devices, DMA engines) are guaranteed to see the same copy of a memory location (Point of Coherency – PoC),
- instruction and data caches of a core are guaranteed to see the same copy of a memory location (Point of Unification – PoU).

Certain functions are NOPs on certain architectures, for example on Intel it's not necessary to explicitly make caches coherent to PoU.

### 14.1.6.2 Function Documentation

#### 14.1.6.2.1 `l4_cache_clean_data()`

```
L4_BEGIN_DECLS int l4_cache_clean_data (
    unsigned long start,
    unsigned long end) [inline]
```

Cache clean a range in D-cache; writes back to PoC.

#### Parameters

<i>start</i>	Start of range (inclusive)
<i>end</i>	End of range (exclusive)

#### Return values

<i>0</i>	on success
<i>-EFAULT</i>	in the case of an unresolved page fault in the given area

Writes back any dirty cache lines in the range but leaves them in the cache and marks the cached copies clean.

#### Examples

[examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#).

Definition at line 70 of file [cache.h](#).

References [L4\\_NOTHROW](#).

#### 14.1.6.2.2 `l4_cache_coherent()`

```
int l4_cache_coherent (
    unsigned long start,
    unsigned long end) [inline]
```

Make memory coherent between I-cache and D-cache; writes back to PoU.

#### Parameters

<i>start</i>	Start of range (inclusive)
<i>end</i>	End of range (exclusive)

### Return values

<i>0</i>	on success
<i>-EFAULT</i>	in the case of an unresolved page fault in the given area

Definition at line 94 of file [cache.h](#).

References [L4\\_NOTHROW](#).

#### 14.1.6.2.3 `l4_cache_dma_coherent()`

```
int l4_cache_dma_coherent (  
    unsigned long start,  
    unsigned long end) [inline]
```

Make memory coherent for use with external memory; writes back to PoC.

### Parameters

<i>start</i>	Start of range (inclusive)
<i>end</i>	End of range (exclusive)

### Return values

<i>0</i>	on success
<i>-EFAULT</i>	in the case of an unresolved page fault in the given area

Definition at line 102 of file [cache.h](#).

References [L4\\_NOTHROW](#).

#### 14.1.6.2.4 `l4_cache_flush_data()`

```
int l4_cache_flush_data (  
    unsigned long start,  
    unsigned long end) [inline]
```

Cache flush a range; writes back to PoC.

### Parameters

---

<i>start</i>	Start of range (inclusive)
<i>end</i>	End of range (exclusive)

### Return values

0	on success
-EFAULT	in the case of an unresolved page fault in the given area

Writes back any dirty cache lines and invalidates all cache entries in the range.

Definition at line 78 of file [cache.h](#).

References [L4\\_NOTHROW](#).

#### 14.1.6.2.5 l4\_cache\_inv\_data()

```
int l4_cache_inv_data (
    unsigned long start,
    unsigned long end) [inline]
```

Cache invalidate a range; might write back to PoC.

### Parameters

<i>start</i>	Start of range (inclusive)
<i>end</i>	End of range (exclusive)

### Return values

0	on success
-EFAULT	in the case of an unresolved page fault in the given area

Invalidates all cache entries in the range but does not necessarily write back dirty cache lines.

### Note

Implementations may choose to write back dirty lines nonetheless if this is more efficient.

Definition at line 86 of file [cache.h](#).

References [L4\\_NOTHROW](#).

### 14.1.7 Memory related

Memory related constants, data types and functions.

Collaboration diagram for Memory related:



#### Macros

- **#define L4\_PAGESIZE**  
*Minimal page size (in bytes).*
- **#define L4\_PAGEMASK**  
*Mask for the page number.*
- **#define L4\_LOG2\_PAGESIZE**  
*Number of bits used for page offset.*
- **#define L4\_SUPERPAGESIZE**  
*Size of a large page.*
- **#define L4\_SUPERPAGEMASK**  
*Mask for the number of a large page.*
- **#define L4\_LOG2\_SUPERPAGESIZE**  
*Number of bits used as offset for a large page.*
- **#define L4\_INVALID\_PTR** ((void \*)L4\_INVALID\_ADDR)  
*Invalid address as pointer type.*
- **#define L4\_PAGESHIFT** 12  
*Size of a page, log2-based.*
- **#define L4\_SUPERPAGESHIFT** 21  
*Size of a large page, log2-based.*
- **#define L4\_PAGESHIFT** 12  
*Size of a page, log2-based.*
- **#define L4\_SUPERPAGESHIFT** 21  
*Size of a large page, log2-based.*
- **#define L4\_PAGESHIFT** 12  
*Size of a page, log2-based.*
- **#define L4\_SUPERPAGESHIFT** 21  
*Size of a large page, log2-based.*
- **#define L4\_PAGESHIFT** 12  
*Size of a page log2-based.*
- **#define L4\_SUPERPAGESHIFT** 22  
*Size of a large page log2-based.*

## Enumerations

- enum [l4\\_addr\\_consts\\_t](#) { [L4\\_INVALID\\_ADDR](#) = ~0UL }
- Address related constants.*

## Functions

- [l4\\_addr\\_t l4\\_trunc\\_page](#) ([l4\\_addr\\_t](#) address) [L4\\_NOTHROW](#)  
*Round an address down to the next lower page boundary.*
- [l4\\_addr\\_t l4\\_trunc\\_size](#) ([l4\\_addr\\_t](#) address, unsigned char bits) [L4\\_NOTHROW](#)  
*Round an address down to the next lower flexpage with size bits.*
- [l4\\_addr\\_t l4\\_round\\_page](#) ([l4\\_addr\\_t](#) address) [L4\\_NOTHROW](#)  
*Round address up to the next page.*
- [l4\\_addr\\_t l4\\_round\\_size](#) ([l4\\_addr\\_t](#) value, unsigned char bits) [L4\\_NOTHROW](#)  
*Round value up to the next alignment with bits size.*
- unsigned [l4\\_bytes\\_to\\_mwords](#) (unsigned size) [L4\\_NOTHROW](#)  
*Determine how many machine words ([l4\\_umword\\_t](#)) are required to store a buffer of 'size' bytes.*

### 14.1.7.1 Detailed Description

Memory related constants, data types and functions.

### 14.1.7.2 Macro Definition Documentation

#### 14.1.7.2.1 L4\_LOG2\_PAGESIZE

```
#define L4_LOG2_PAGESIZE
```

Number of bits used for page offset.

Size of page in log2.

Definition at line 409 of file [consts.h](#).

Referenced by [L4Re::Dataspace::map\(\)](#), [L4Re::Dataspace::map\\_region\(\)](#), and [L4Re::Util::Dataspace\\_svr::page\\_shift\(\)](#).

#### 14.1.7.2.2 L4\_LOG2\_SUPERPAGESIZE

```
#define L4_LOG2_SUPERPAGESIZE
```

Number of bits used as offset for a large page.

Size of large page in log2

Definition at line 435 of file [consts.h](#).

**14.1.7.2.3 L4\_PAGEMASK**

```
#define L4_PAGEMASK
```

Mask for the page number.

**Note**

The most significant bits are set.

Definition at line 400 of file [consts.h](#).

Referenced by [L4virtio::Driver::Device::driver\\_connect\(\)](#), [l4\\_round\\_page\(\)](#), and [l4\\_trunc\\_page\(\)](#).

**14.1.7.2.4 L4\_PAGESHIFT [1/3]**

```
#define L4_PAGESHIFT 12
```

Size of a page, log2-based.

Size of a page log2-based.

Definition at line 24 of file [consts.h](#).

**14.1.7.2.5 L4\_PAGESHIFT [2/3]**

```
#define L4_PAGESHIFT 12
```

Size of a page, log2-based.

Size of a page log2-based.

**Examples**

[examples/libs/l4re/c++/mem\\_alloc/ma+rm.cc](#), [examples/libs/l4re/c/ma+rm.c](#), [examples/libs/l4re/streammap/client.cc](#),  
and [examples/libs/l4re/streammap/server.cc](#).

Definition at line 26 of file [consts.h](#).

Referenced by [L4Re::Rm::attach\(\)](#), [L4virtio::Driver::Device::driver\\_connect\(\)](#), [L4Re::Rm::free\\_area\(\)](#), [L4Re::Util::Dataspace\\_svr::map](#)  
[L4Re::Rm::reserve\\_area\(\)](#), [L4Re::Rm::reserve\\_area\(\)](#), [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#), and  
[L4virtio::Driver::Virtio\\_net\\_device::setup\\_device\(\)](#).

**14.1.7.2.6 L4\_PAGESHIFT [3/3]**

```
#define L4_PAGESHIFT 12
```

Size of a page, log2-based.

Size of a page log2-based.

Definition at line 26 of file [consts.h](#).

#### 14.1.7.2.7 L4\_SUPERPAGEMASK

```
#define L4_SUPERPAGEMASK
```

Mask for the number of a large page.

##### Note

The most significant bits are set.

Definition at line 427 of file [consts.h](#).

#### 14.1.7.2.8 L4\_SUPERPAGESHIFT [1/3]

```
#define L4_SUPERPAGESHIFT 21
```

Size of a large page, log2-based.

Size of a large page log2-based.

Definition at line 30 of file [consts.h](#).

#### 14.1.7.2.9 L4\_SUPERPAGESHIFT [2/3]

```
#define L4_SUPERPAGESHIFT 21
```

Size of a large page, log2-based.

Size of a large page log2-based.

##### Examples

[examples/libs/l4re/c++/mem\\_alloc/ma+rm.cc](#), and [examples/libs/l4re/c/ma+rm.c](#).

Definition at line 31 of file [consts.h](#).

Referenced by [L4virtio::Svr::Driver\\_mem\\_region\\_t< Mem\\_region\\_info >::Driver\\_mem\\_region\\_t\(\)](#).

#### 14.1.7.2.10 L4\_SUPERPAGESHIFT [3/3]

```
#define L4_SUPERPAGESHIFT 21
```

Size of a large page, log2-based.

Size of a large page log2-based.

Definition at line 31 of file [consts.h](#).

#### 14.1.7.2.11 L4\_SUPERPAGESIZE

```
#define L4_SUPERPAGESIZE
```

Size of a large page.

A large page is a *super page* on IA32 or a *section* on ARM.

Definition at line 418 of file [consts.h](#).

Referenced by [L4virtio::Driver::Device::driver\\_connect\(\)](#).

### 14.1.7.3 Enumeration Type Documentation

#### 14.1.7.3.1 l4\_addr\_consts\_t

```
enum l4_addr_consts_t
```

Address related constants.

##### Enumerator

---



L4_INVALID_ADDR	Invalid address.
-----------------	------------------

Definition at line 503 of file [consts.h](#).

#### 14.1.7.4 Function Documentation

##### 14.1.7.4.1 l4\_bytes\_to\_mwords()

```
unsigned l4_bytes_to_mwords (
    unsigned size) [inline]
```

Determine how many machine words ([l4\\_umword\\_t](#)) are required to store a buffer of 'size' bytes.

##### Parameters

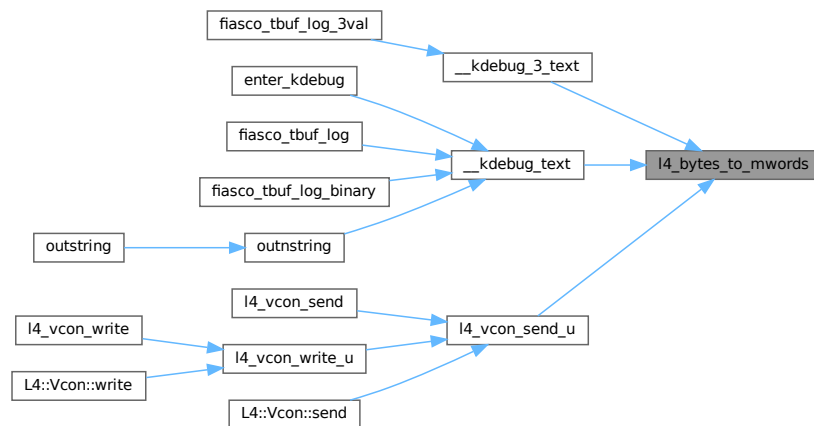
size	The number of bytes to be translated into machine words.
------	----------------------------------------------------------

Definition at line 496 of file [consts.h](#).

References [L4\\_INLINE](#), and [L4\\_NOTHROW](#).

Referenced by [\\_\\_kdebug\\_3\\_text\(\)](#), [\\_\\_kdebug\\_text\(\)](#), and [l4\\_vcon\\_send\\_u\(\)](#).

Here is the caller graph for this function:



##### 14.1.7.4.2 l4\_round\_page()

```
l4_addr_t l4_round_page (
    l4_addr_t address) [inline]
```

Round address up to the next page.

The address is rounded up to the next minimal page boundary. On most architectures this is a 4k page. Check [L4\\_PAGESIZE](#) for the minimal page size.

##### Parameters

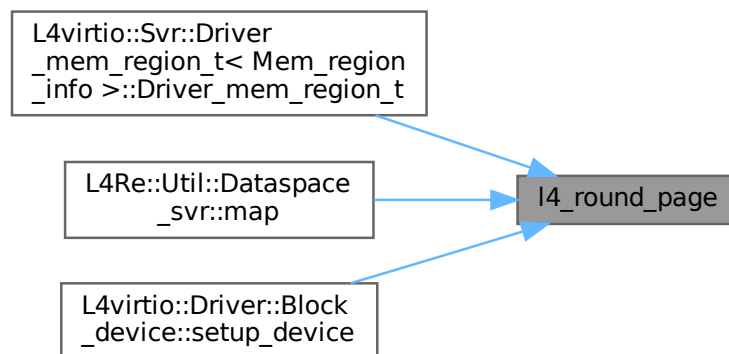
<i>address</i>	The address to round up.
----------------	--------------------------

Definition at line 473 of file [consts.h](#).

References [L4\\_INLINE](#), [L4\\_NOTHROW](#), [L4\\_PAGEMASK](#), and [L4\\_PAGESIZE](#).

Referenced by [L4virtio::Svr::Driver\\_mem\\_region\\_t< Mem\\_region\\_info >::Driver\\_mem\\_region\\_t\(\)](#), [L4Re::Util::Dataspace\\_svr::map\(\)](#), and [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#).

Here is the caller graph for this function:



#### 14.1.7.4.3 l4\_round\_size()

```
l4_addr_t l4_round_size (
    l4_addr_t value,
    unsigned char bits) [inline]
```

Round value up to the next alignment with *bits* size.

##### Parameters

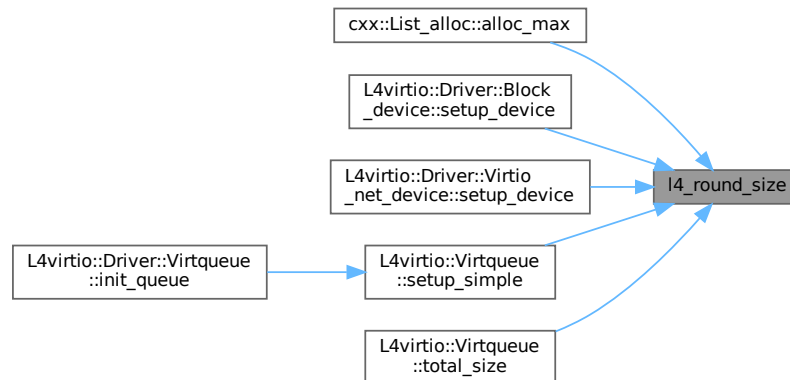
<i>value</i>	The value to round up to the next size-alignment.
<i>bits</i>	The size of the alignment (log2).

Definition at line 484 of file [consts.h](#).

References [L4\\_INLINE](#), and [L4\\_NOTHROW](#).

Referenced by [cxx::List\\_alloc::alloc\\_max\(\)](#), [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#), [L4virtio::Driver::Virtio\\_net\\_device::setup\\_device\(\)](#), [L4virtio::Virtqueue::setup\\_simple\(\)](#), and [L4virtio::Virtqueue::total\\_size\(\)](#).

Here is the caller graph for this function:



#### 14.1.7.4.4 l4\_trunc\_page()

```
l4_addr_t l4_trunc_page (
    l4_addr_t address) [inline]
```

Round an address down to the next lower page boundary.

The address is rounded down to the next lower minimal page boundary. On most architectures this is a 4k page. Check [L4\\_PAGESIZE](#) for the minimal page size.

#### Parameters

<i>address</i>	The address to round.
----------------	-----------------------

#### Examples

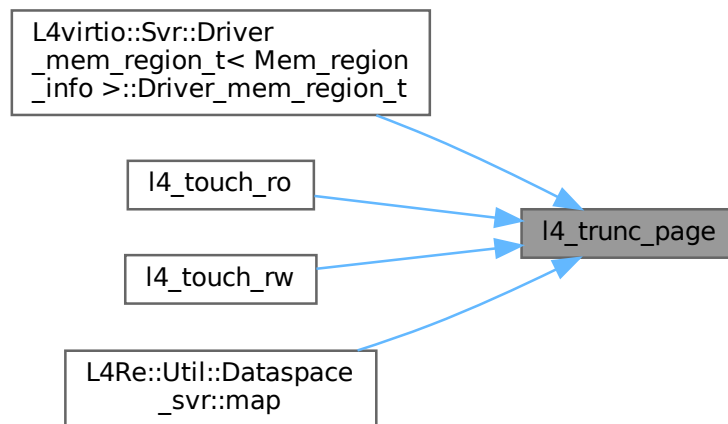
[examples/libs/l4re/c++/mem\\_alloc/ma+rm.cc](#), and [examples/libs/l4re/c/ma+rm.c](#).

Definition at line 448 of file [consts.h](#).

References [L4\\_INLINE](#), [L4\\_NOTHROW](#), and [L4\\_PAGEMASK](#).

Referenced by [L4virtio::Svr::Driver\\_mem\\_region\\_t< Mem\\_region\\_info >::Driver\\_mem\\_region\\_t\(\)](#), [l4\\_touch\\_ro\(\)](#), [l4\\_touch\\_rw\(\)](#), and [L4Re::Util::Dataspace\\_svr::map\(\)](#).

Here is the caller graph for this function:



#### 14.1.7.4.5 l4\_trunc\_size()

```
l4_addr_t l4_trunc_size (
    l4_addr_t address,
    unsigned char bits) [inline]
```

Round an address down to the next lower flexpage with size *bits*.

##### Parameters

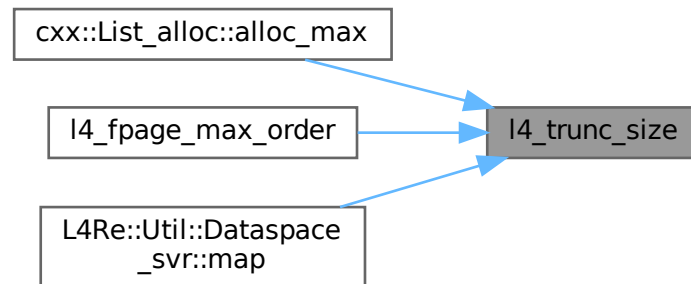
<i>address</i>	The address to round.
<i>bits</i>	The size of the flexpage (log2).

Definition at line 459 of file [consts.h](#).

References [L4\\_INLINE](#), and [L4\\_NOTHROW](#).

Referenced by [cxx::List\\_alloc::alloc\\_max\(\)](#), [l4\\_fpage\\_max\\_order\(\)](#), and [L4Re::Util::Dataspace\\_svr::map\(\)](#).

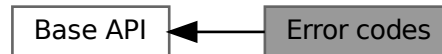
Here is the caller graph for this function:



### 14.1.8 Error codes

Common error codes.

Collaboration diagram for Error codes:



#### Enumerations

- enum `l4_error_code_t` {  
`L4_EOK` = 0 , `L4_EPERM` = 1 , `L4_ENOENT` = 2 , `L4_EIO` = 5 ,  
`L4_ENXIO` = 6 , `L4_E2BIG` = 7 , `L4_EAGAIN` = 11 , `L4_ENOMEM` = 12 ,  
`L4_EACCESS` = 13 , `L4_EFAULT` = 14 , `L4_EBUSY` = 16 , `L4_EEXIST` = 17 ,  
`L4_ENODEV` = 19 , `L4_ENOTDIR` = 20 , `L4_EINVAL` = 22 , `L4_ENOSPC` = 28 ,  
`L4_ERANGE` = 34 , `L4_ENAMETOOLONG` = 36 , `L4_ENOSYS` = 38 , `L4_EBADPROTO` = 39 ,  
`L4_EADDRNOTAVAIL` = 99 , `L4_ERRNOMAX` = 100 , `L4_ENOREPLY` = 1000 , `L4_MSGTOOSHORT` =  
1001 ,  
`L4_MSGTOOLONG` = 1002 , `L4_MSGMISSARG` = 1003 , `L4_EIPC_LO` = 2000 , `L4_EIPC_HI` = 2000 +  
0x1f }

*L4 error codes.*

#### 14.1.8.1 Detailed Description

Common error codes.

##### Include File

```
#include <l4/sys/err.h>
```

### 14.1.8.2 Enumeration Type Documentation

#### 14.1.8.2.1 l4\_error\_code\_t

enum [l4\\_error\\_code\\_t](#)

[L4](#) error codes.

Those error codes are used by both the kernel and the user programs.

#### Enumerator

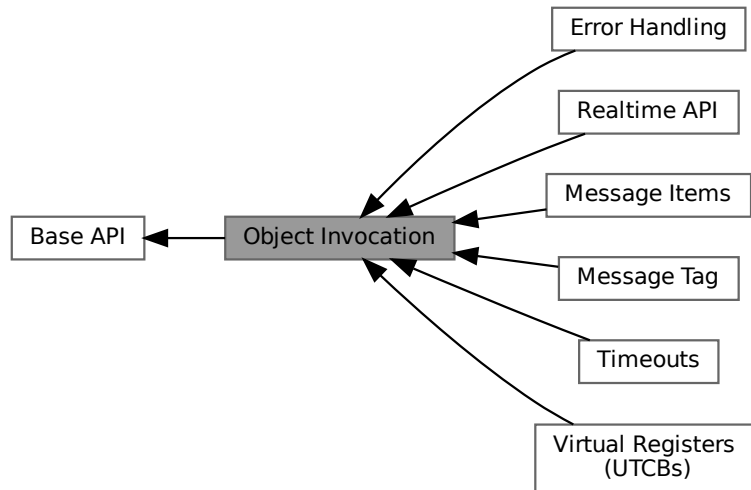
L4_EOK	Ok.
L4_EPERM	No permission.
L4_ENOENT	No such entity.
L4_EIO	I/O error.
L4_ENXIO	No such device or address.
L4_E2BIG	Argument value too big.
L4_EAGAIN	Try again.
L4_ENOMEM	No memory.
L4_EACCESS	Permission denied.
L4_EFAULT	Invalid memory address.
L4_EBUSY	Object currently busy, try later.
L4_EEXIST	Already exists.
L4_ENODEV	No such thing.
L4_ENOTDIR	Not a directory.
L4_EINVAL	Invalid argument.
L4_ENOSPC	No space left on device.
L4_ERANGE	Range error.
L4_ENAMETOOLONG	Name too long.
L4_ENOSYS	No sys.
L4_EBADPROTO	Unsupported protocol.
L4_EADDRNOTAVAIL	Address not available.
L4_ERRNOMAX	Maximum error value.
L4_ENOREPLY	No reply.
L4_MSGTOOSHORT	Message too short.
L4_MSGTOOLONG	Message too long.
L4_MSGMISSARG	Message has invalid capability.
L4_EIPC_LO	Communication error-range low.
L4_EIPC_HI	Communication error-range high.

Definition at line 30 of file [err.h](#).

### 14.1.9 Object Invocation

API for [L4](#) object invocation.

Collaboration diagram for Object Invocation:



#### Topics

- [Message Items](#) . . . . . [236](#)  
*Message-item-related functionality.*
- [Timeouts](#) . . . . . [243](#)  
*All kinds of timeouts and time related functions.*
- [Error Handling](#) . . . . . [251](#)  
*Error handling for [L4](#) object invocation.*
- [Realtime API](#) . . . . . [257](#)
- [Message Tag](#) . . . . . [257](#)  
*API related to the message tag data type.*
- [Virtual Registers \(UTCBS\)](#) . . . . . [270](#)  
[L4](#) Virtual Registers (UTCB).

#### Files

- file [utcb.h](#)  
*UTCB definitions.*

## Enumerations

- enum `l4_syscall_flags_t` {  
`L4_SYSF_NONE`, `L4_SYSF_SEND`, `L4_SYSF_RECV`, `L4_SYSF_OPEN_WAIT`,  
`L4_SYSF_REPLY`, `L4_SYSF_CALL`, `L4_SYSF_WAIT`, `L4_SYSF_SEND_AND_WAIT`,  
`L4_SYSF_REPLY_AND_WAIT` }

*Capability selector flags.*

## Functions

- `l4_msgtag_t l4_ipc_send (l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag, l4_timeout_t timeout) L4_NOTHROW`  
*Send a message to an object (do **not** wait for a reply).*
- `l4_msgtag_t l4_ipc_wait (l4_utcb_t *utcb, l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW`  
*Wait for an incoming message from any possible sender.*
- `l4_msgtag_t l4_ipc_receive (l4_cap_idx_t object, l4_utcb_t *utcb, l4_timeout_t timeout) L4_NOTHROW`  
*Wait for a message from a specific source.*
- `l4_msgtag_t l4_ipc_call (l4_cap_idx_t object, l4_utcb_t *utcb, l4_msgtag_t tag, l4_timeout_t timeout) L4_NOTHROW`  
*Object call (usual invocation).*
- `l4_msgtag_t l4_ipc_reply_and_wait (l4_utcb_t *utcb, l4_msgtag_t tag, l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW`  
*Reply and wait operation (uses the reply capability).*
- `l4_msgtag_t l4_ipc_send_and_wait (l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag, l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW`  
*Send a message and do an open wait.*
- `l4_msgtag_t l4_ipc (l4_cap_idx_t dest, l4_utcb_t *utcb, l4_umword_t flags, l4_umword_t slabel, l4_msgtag_t tag, l4_umword_t *rlabel, l4_timeout_t timeout) L4_NOTHROW`  
*Generic L4 object invocation.*
- `l4_msgtag_t l4_ipc_sleep (l4_timeout_t timeout) L4_NOTHROW`  
*Sleep for an amount of time.*
- `l4_msgtag_t l4_ipc_sleep_ms (l4_uint32_t ms) L4_NOTHROW`  
*Sleep for a certain amount of milliseconds.*
- `l4_msgtag_t l4_ipc_sleep_us (l4_uint64_t us) L4_NOTHROW`  
*Sleep for a certain amount of microseconds.*
- int `l4_sndfpage_add (l4_fpage_t const snd_fpage, unsigned long snd_base, l4_msgtag_t *tag) L4_NOTHROW`  
*Add a flexpage to be sent to the UTCB.*

### 14.1.9.1 Detailed Description

API for L4 object invocation.

#### Include File

```
#include <l4/sys/ipc.h>
```

General abstractions for L4 object invocation. The basic principle is that all objects are denoted by a capability that is accessed via a capability selector (see [Capabilities](#)).

This set of functions is common to all kinds of objects provided by the L4 micro kernel. The concrete semantics of an invocation depends on the object that shall be invoked.

Objects may be invoked in various ways, the most common way is to use a *call* operation (`l4_ipc_call()`). However, there are a lot more flavours available that have a semantics depending on the object.



See also

[IPC-Gate API](#)

[L4 Inter-Process Communication \(IPC\)](#)

### 14.1.9.2 Timeouts during IPC

IPC operation between two communication partners may consist of up to two phases (send phase and receive phase). For both phases, a timeout may be specified (send timeout and receive timeout).

Note

When IPC communication happens across CPU cores and a timeout is specified, then the counting of the timeout only begins after the target thread has been scheduled at least once. In particular, this means that an IPC timeout, including a timeout of zero, may be delayed depending on the scheduling on the target CPU core. If a higher priority thread on the target core is executing a busy loop, that delay may even be indefinitely.

See also

[Timeouts](#)

### 14.1.9.3 Enumeration Type Documentation

#### 14.1.9.3.1 `l4_syscall_flags_t`

```
enum l4_syscall_flags_t
```

Capability selector flags.

These flags determine the concrete operation when a kernel object is invoked.

The following combinations of flags are supported when invoking IPC (see [l4\\_ipc\(\)](#)); with other combinations, behavior is undefined:

- [L4\\_SYSF\\_SEND](#): send to specified partner
- [L4\\_SYSF\\_RECV](#): receive from specified partner
- [L4\\_SYSF\\_RECV](#) | [L4\\_SYSF\\_OPEN\\_WAIT](#): receive from any sending partner; see [L4\\_SYSF\\_WAIT](#)
- [L4\\_SYSF\\_SEND](#) | [L4\\_SYSF\\_RECV](#): call specified partner; see [L4\\_SYSF\\_CALL](#)
- [L4\\_SYSF\\_SEND](#) | [L4\\_SYSF\\_RECV](#) | [L4\\_SYSF\\_OPEN\\_WAIT](#): send to specified partner and receive from any sending partner; see [L4\\_SYSF\\_SEND\\_AND\\_WAIT](#)
- [L4\\_SYSF\\_REPLY](#) | [L4\\_SYSF\\_SEND](#): reply to caller
- [L4\\_SYSF\\_REPLY](#) | [L4\\_SYSF\\_SEND](#) | [L4\\_SYSF\\_RECV](#): call the caller
- [L4\\_SYSF\\_REPLY](#) | [L4\\_SYSF\\_SEND](#) | [L4\\_SYSF\\_RECV](#) | [L4\\_SYSF\\_OPEN\\_WAIT](#): reply to caller and receive from any sending partner; see [L4\\_SYSF\\_REPLY\\_AND\\_WAIT](#)

**Enumerator**

L4_SYSF_NONE	Empty set of flags.
L4_SYSF_SEND	Send-phase flag. Setting this flag in a capability selector induces a send phase, this means a message is sent to the object denoted by the capability. For receive phase see <a href="#">L4_SYSF_RECV</a> . In <a href="#">l4_vcpu_state_t::user_task</a> this flag means that the kernel has cached the user task capability internally, see <a href="#">l4_thread_vcpu_resume_commit()</a> .
L4_SYSF_RECV	Receive-phase flag. Setting this flag in a capability selector induces a receive phase, this means the invoking thread waits for a message from the object denoted by the capability. For a send phase see <a href="#">L4_SYSF_SEND</a> .
L4_SYSF_OPEN_WAIT	Open-wait flag. This flag indicates that the receive operation (see <a href="#">L4_SYSF_RECV</a> ) shall be an <i>open wait</i> . <i>Open wait</i> means that the invoking thread shall wait for a message from any possible sender and <i>not</i> from the sender denoted by the capability.
L4_SYSF_REPLY	Reply flag. This flag indicates that the send phase shall use the in-kernel reply capability instead of the capability denoted by the selector index.
L4_SYSF_CALL	Call flags (combines send and receive). Combines <a href="#">L4_SYSF_SEND</a> and <a href="#">L4_SYSF_RECV</a> .
L4_SYSF_WAIT	Wait flags (combines receive and open wait). Combines <a href="#">L4_SYSF_RECV</a> and <a href="#">L4_SYSF_OPEN_WAIT</a> .
L4_SYSF_SEND_AND_WAIT	Send-and-wait flags. Combines <a href="#">L4_SYSF_SEND</a> and <a href="#">L4_SYSF_WAIT</a> .
L4_SYSF_REPLY_AND_WAIT	Reply-and-wait flags. Combines <a href="#">L4_SYSF_SEND</a> , <a href="#">L4_SYSF_REPLY</a> , and <a href="#">L4_SYSF_WAIT</a> .

Definition at line 50 of file [consts.h](#).

#### 14.1.9.4 Function Documentation

##### 14.1.9.4.1 l4\_ipc()

```
l4_msgtag_t l4_ipc (
    l4_cap_idx_t dest,
    l4_utcb_t * utcb,
    l4_umword_t flags,
    l4_umword_t slabel,
    l4_msgtag_t tag,
    l4_umword_t * rlabel,
    l4_timeout_t timeout) [inline]
```

Generic [L4](#) object invocation.

#### Parameters

<i>dest</i>	Destination object. <a href="#">L4_INVALID_CAP</a> denotes the current thread. An IPC to the current thread will always abort after the specified timeout and can be used for sleeping without busy waiting.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .
<i>flags</i>	Invocation flags (see <a href="#">l4_syscall_flags_t</a> ).
<i>slabel</i>	Send label if applicable (may be seen by the receiver).
<i>tag</i>	Sending message tag.

out	<i>rlabel</i>	Receiving label.
	<i>timeout</i>	Timeout pair (see <a href="#">l4_timeout_t</a> ).

#### Returns

return tag

#### See also

[L4 Inter-Process Communication \(IPC\)](#)

Definition at line 23 of file [ipc.h](#).

References [L4\\_NOTHROW](#).

Referenced by [l4\\_ipc\\_call\(\)](#), [l4\\_ipc\\_receive\(\)](#), [l4\\_ipc\\_reply\\_and\\_wait\(\)](#), [l4\\_ipc\\_send\(\)](#), [l4\\_ipc\\_send\\_and\\_wait\(\)](#), and [l4\\_ipc\\_wait\(\)](#).



```
l4_msgtag_t tag,
l4_timeout_t timeout) [inline]
```

Object call (usual invocation).

### Parameters

<i>object</i>	Capability selector for the object to call. A value of <a href="#">L4_INVALID_CAP</a> denotes the current thread and will abort the IPC after the time specified in the <code>snd</code> part of the <code>timeout</code> parameter has expired.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .
<i>tag</i>	Message tag to describe the message to be sent.
<i>timeout</i>	Timeout pair for send an receive phase (see <a href="#">l4_timeout_t</a> ).

### Returns

result tag

A message is sent to the object and the invoker waits for a reply from the object. Messages from other sources are not accepted.

### Note

The send-to-receive transition needs no time, the object can reply with a send timeout of zero.

If a finite receive timeout is specified, the IPC receive operation could abort before the partner was able to send the reply message. Under certain circumstances the partner may still have the temporary reply capability to the calling thread and may use this capability to reply to the caller at a later, unexpected time specifying an arbitrary IPC label. This case is relevant for servers which call another, possibly untrusted, server while serving a client request.

### See also

[L4 Inter-Process Communication \(IPC\)](#)

### Examples

[examples/sys/aliens/main.c](#), [examples/sys/ipc/ipc\\_example.c](#), and [examples/sys/singlestep/main.c](#).

Definition at line [565](#) of file [ipc.h](#).

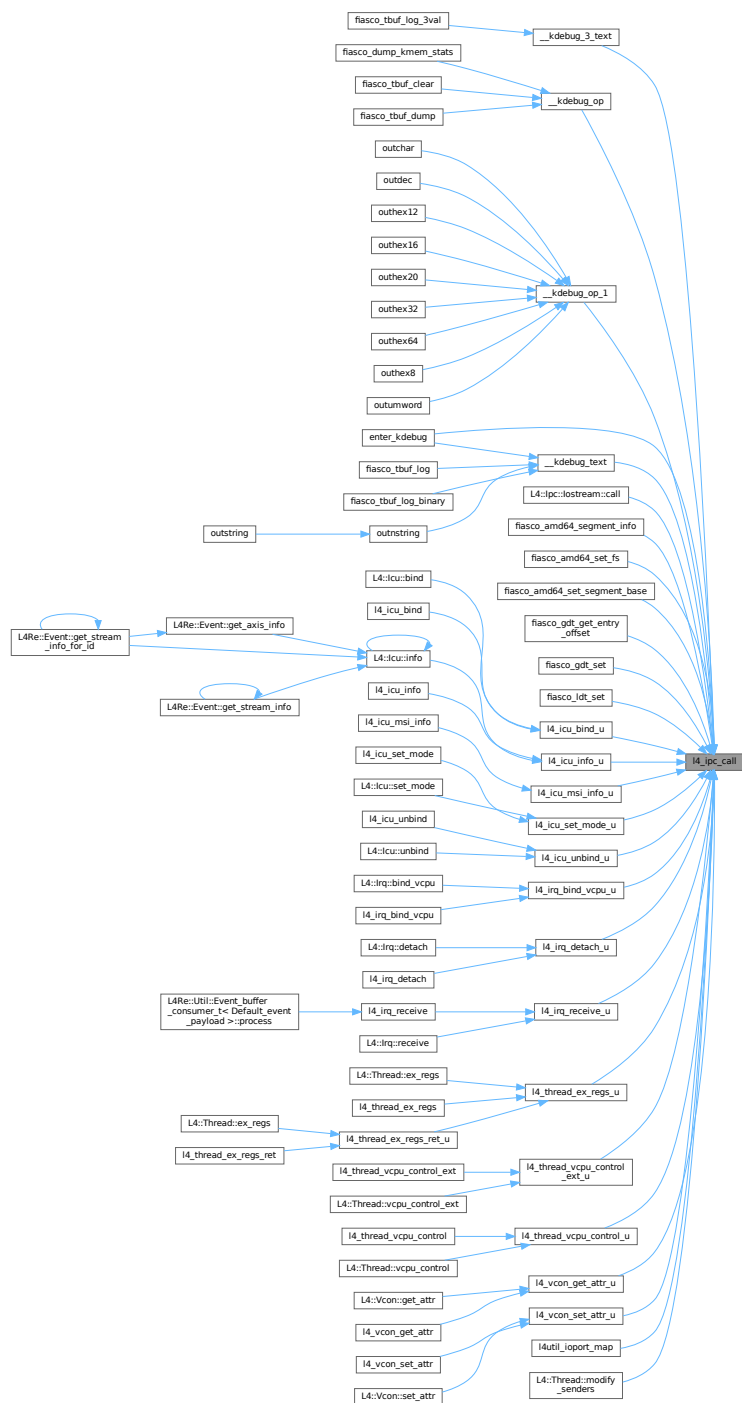
References [l4\\_ipc\(\)](#), [L4\\_NOTHROW](#), and [L4\\_SYSF\\_CALL](#).

Referenced by [\\_\\_kdebug\\_3\\_text\(\)](#), [\\_\\_kdebug\\_op\(\)](#), [\\_\\_kdebug\\_op\\_1\(\)](#), [\\_\\_kdebug\\_text\(\)](#), [L4::lpc::loststream::call\(\)](#), [enter\\_kdebug\(\)](#), [fiasco\\_amd64\\_segment\\_info\(\)](#), [fiasco\\_amd64\\_set\\_fs\(\)](#), [fiasco\\_amd64\\_set\\_segment\\_base\(\)](#), [fiasco\\_gdt\\_get\\_entry\\_offset\(\)](#), [fiasco\\_gdt\\_set\(\)](#), [fiasco\\_ldt\\_set\(\)](#), [l4\\_icu\\_bind\\_u\(\)](#), [l4\\_icu\\_info\\_u\(\)](#), [l4\\_icu\\_msi\\_info\\_u\(\)](#), [l4\\_icu\\_set\\_mode\\_u\(\)](#), [l4\\_icu\\_unbind\\_u\(\)](#), [l4\\_irq\\_bind\\_vcpu\\_u\(\)](#), [l4\\_irq\\_detach\\_u\(\)](#), [l4\\_irq\\_receive\\_u\(\)](#), [l4\\_thread\\_ex\\_regs\\_u\(\)](#), [l4\\_thread\\_vcpu\\_control\\_ext\\_u\(\)](#), [l4\\_thread\\_vcpu\\_control\\_u\(\)](#), [l4\\_vcon\\_get\\_attr\\_u\(\)](#), [l4\\_vcon\\_set\\_attr\\_u\(\)](#), [l4util\\_ioport\\_map\(\)](#), and [L4::Thread::modify\\_senders\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.1.9.4.3 l4\_ipc\_receive()

```
14_msgtag_t 14_ipc_receive (
    14_cap_idx_t object,
    14_utcb_t * utcb,
    14_timeout_t timeout) [inline]
```

Wait for a message from a specific source.

#### Parameters

---



<i>object</i>	Object to receive a message from. A value of <a href="#">L4_INVALID_CAP</a> denotes the current thread. It could be used for sleeping without busy waiting for the time specified in the <code>rcv</code> part of the <code>timeout</code> parameter.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .
<i>timeout</i>	Timeout pair (see <a href="#">l4_timeout_t</a> , only the receive part matters).

**Returns**

result tag.

This operation waits for a message from the specified object. Messages from other sources are not accepted by this operation. The operation does not include a send phase, this means no message is sent to the object.

**Note**

This operation is usually used to receive messages from a specific IRQ or thread. However, it is not common to use this operation for normal applications.

**See also**

[L4 Inter-Process Communication \(IPC\)](#)

**Examples**

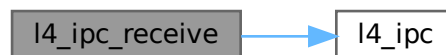
[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 602 of file `ipc.h`.

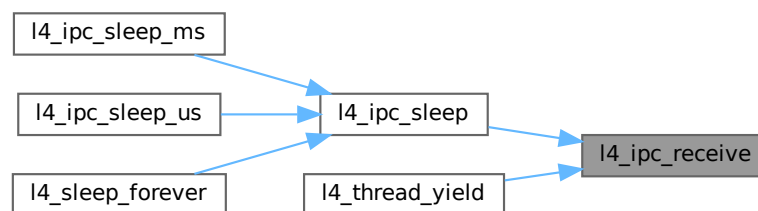
References [l4\\_ipc\(\)](#), [L4\\_NOTHROW](#), [L4\\_SYSF\\_RECV](#), and [l4\\_msgtag\\_t::raw](#).

Referenced by [l4\\_ipc\\_sleep\(\)](#), and [l4\\_thread\\_yield\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.1.9.4.4 l4\_ipc\_reply\_and\_wait()

```
l4_msgtag_t l4_ipc_reply_and_wait (
    l4_utcb_t * utcb,
    l4_msgtag_t tag,
    l4_umword_t * label,
    l4_timeout_t timeout) [inline]
```

Reply and wait operation (uses the *reply* capability).

##### Parameters

	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .
	<i>tag</i>	Describes the message to be sent as reply.
out	<i>label</i>	Label assigned to the source object of the received message.
	<i>timeout</i>	Timeout pair (see <a href="#">l4_timeout_t</a> ).

##### Returns

result tag

A message is sent to the previous caller using the implicit reply capability. Afterwards the invoking thread waits for a message from any source.

##### Note

This is the standard server operation: it sends a reply to the actual client and waits for the next incoming request, which may come from any other client.

In case of multiple senders trying to send to the thread performing this system call, the thread receives from a sender with the highest priority. In this respect, IRQ sources have the highest priority 255.

##### See also

[L4 Inter-Process Communication \(IPC\)](#)

##### Examples

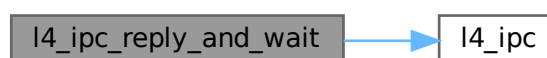
[examples/sys/ipc/ipc\\_example.c](#).

Definition at line 572 of file [ipc.h](#).

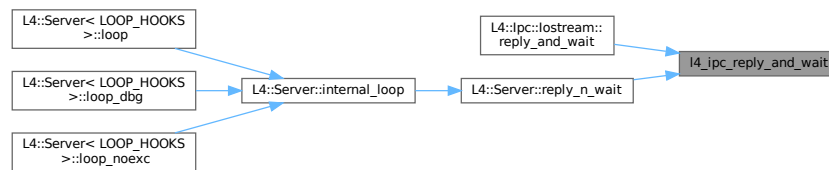
References [L4\\_INVALID\\_CAP](#), [l4\\_ipc\(\)](#), [L4\\_NOTHROW](#), and [L4\\_SYSF\\_REPLY\\_AND\\_WAIT](#).

Referenced by [L4::ipc::loststream::reply\\_and\\_wait\(\)](#), and [L4::Server< LOOP\\_HOOKS >::reply\\_n\\_wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.1.9.4.5 l4\_ipc\_send()

```

l4_msgtag_t l4_ipc_send (
    l4_cap_idx_t dest,
    l4_utcb_t * utcb,
    l4_msgtag_t tag,
    l4_timeout_t timeout) [inline]

```

Send a message to an object (do **not** wait for a reply).

##### Parameters

<i>dest</i>	Capability selector for the destination object. A value of <a href="#">L4_INVALID_CAP</a> denotes the current thread and could be used for sleeping without busy waiting for the time specified in the <code>snd</code> part of the <code>timeout</code> parameter.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .
<i>tag</i>	Descriptor for the message to be sent.
<i>timeout</i>	Timeout pair (see <a href="#">l4_timeout_t</a> ) only send part is relevant.

##### Returns

Syscall return tag for the send-only operation, this means there is no return value except [L4\\_MSGTAG\\_ERROR](#) indicating success or failure of the send operation. Use [l4\\_ipc\\_error\(\)](#) to check for errors and **do not** use [l4\\_error\(\)](#).

A message is sent to the destination object. There is no receive phase included. The invoker continues working after sending the message.

##### Note

This is a special-purpose message transfer. Objects usually support only invocation via [l4\\_ipc\\_call\(\)](#) consisting of a send phase and a receive phase for returning the result of the object invocation. For example, [l4\\_icu\\_unmask\(\)](#), [l4\\_icu\\_mask\(\)](#) and [l4\\_irq\\_trigger\(\)](#) use send-only IPC operations for object invocation.



	<i>dest</i>	Object to send a message to. A value of <a href="#">L4_INVALID_CAP</a> denotes the current thread and will abort the IPC after the time specified in the <code>snd</code> part of the <code>timeout</code> parameter has expired.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .
	<i>tag</i>	Describes the message that shall be sent.
out	<i>label</i>	Label assigned to the source object of the receive phase.
	<i>timeout</i>	Timeout pair (see <a href="#">l4_timeout_t</a> ).

#### Returns

result tag

A message is sent to the destination object and the invoking thread waits for a reply from any source.

#### Note

This is a special-purpose operation and shall not be used in general applications.

In case of multiple senders trying to send to the thread performing this system call, the thread receives from a sender with the highest priority. In this respect, IRQ sources have the highest priority 255.

#### See also

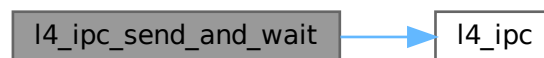
[L4 Inter-Process Communication \(IPC\)](#)

Definition at line 579 of file [ipc.h](#).

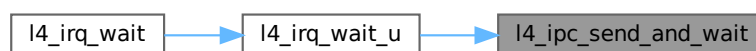
References [l4\\_ipc\(\)](#), [L4\\_NOTHROW](#), and [L4\\_SYSF\\_SEND\\_AND\\_WAIT](#).

Referenced by [l4\\_irq\\_wait\\_u\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.1.9.4.7 l4\_ipc\_sleep()

```
l4_msgtag_t l4_ipc_sleep (  
    l4_timeout_t timeout)  [inline]
```

Sleep for an amount of time.

#### Parameters

---

<i>timeout</i>	Timeout pair (see <a href="#">l4_timeout_t</a> , the receive part matters).
----------------	-----------------------------------------------------------------------------

#### Returns

error code:

- [L4\\_IPC\\_RETIMEOUT](#): success
- [L4\\_IPC\\_RECANCELED](#) woken up by a different thread ([l4\\_thread\\_ex\\_regs\(\)](#)).

The invoking thread waits until the timeout is expired or the wait was aborted by another thread by [l4\\_thread\\_ex\\_regs\(\)](#).

#### See also

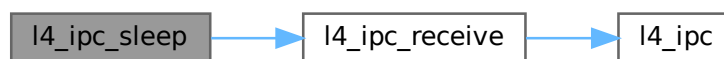
[L4 Inter-Process Communication \(IPC\)](#)

Definition at line 611 of file [ipc.h](#).

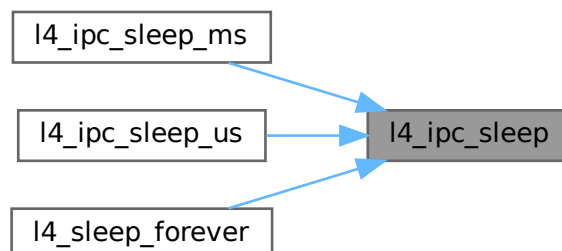
References [L4\\_INVALID\\_CAP](#), [l4\\_ipc\\_receive\(\)](#), and [L4\\_NOTHROW](#).

Referenced by [l4\\_ipc\\_sleep\\_ms\(\)](#), [l4\\_ipc\\_sleep\\_us\(\)](#), and [l4\\_sleep\\_forever\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.1.9.4.8 l4\_ipc\_sleep\_ms()

```
l4_msgtag_t l4_ipc_sleep_ms (  
    l4_uint32_t ms) [inline]
```

Sleep for a certain amount of milliseconds.

#### Parameters

---



<i>ms</i>	Number of milliseconds to wait.
-----------	---------------------------------

#### Returns

error code:

- [L4\\_IPC\\_RETIMEOUT](#): success
- [L4\\_IPC\\_RECANCELED](#) woken up by a different thread ([l4\\_thread\\_ex\\_regs\(\)](#)).

The invoking thread waits until the timeout is expired or the wait was aborted by another thread by [l4\\_thread\\_ex\\_regs\(\)](#).

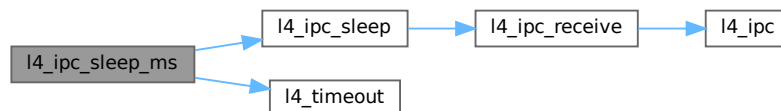
#### See also

[L4 Inter-Process Communication \(IPC\)](#)

Definition at line 615 of file [ipc.h](#).

References [l4\\_ipc\\_sleep\(\)](#), [L4\\_IPC\\_TIMEOUT\\_NEVER](#), [L4\\_NOTHROW](#), and [l4\\_timeout\(\)](#).

Here is the call graph for this function:



#### 14.1.9.4.9 l4\_ipc\_sleep\_us()

```
l4_msgtag_t l4_ipc_sleep_us (
    l4_uint64_t us) [inline]
```

Sleep for a certain amount of microseconds.

#### Parameters

<i>us</i>	Number of microseconds to wait.
-----------	---------------------------------

#### Returns

error code:

- [L4\\_IPC\\_RETIMEOUT](#): success
- [L4\\_IPC\\_RECANCELED](#) woken up by a different thread ([l4\\_thread\\_ex\\_regs\(\)](#)).

The invoking thread waits until the timeout is expired or the wait was aborted by another thread by [l4\\_thread\\_ex\\_regs\(\)](#).

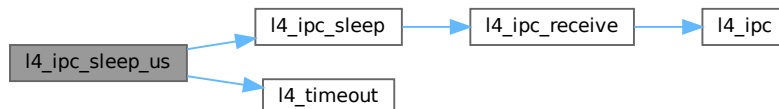
See also

[L4 Inter-Process Communication \(IPC\)](#)

Definition at line 622 of file [ipc.h](#).

References [l4\\_ipc\\_sleep\(\)](#), [L4\\_IPC\\_TIMEOUT\\_NEVER](#), [L4\\_NOTHROW](#), and [l4\\_timeout\(\)](#).

Here is the call graph for this function:



#### 14.1.9.4.10 l4\_ipc\_wait()

```

l4_msgtag_t l4_ipc_wait (
    l4_utcb_t * utcb,
    l4_umword_t * label,
    l4_timeout_t timeout) [inline]
  
```

Wait for an incoming message from any possible sender.

##### Parameters

	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .
out	<i>label</i>	Label assigned to the source object (IPC gate or IRQ).
	<i>timeout</i>	Timeout pair (see <a href="#">l4_timeout_t</a> , only the receive part is used).

##### Returns

return tag

This operation does an open wait, and therefore needs no capability to denote the possible source of a message. This means the calling thread waits for an incoming message from any possible source. There is no send phase included in this operation.

The usual usage of this function is to call that function when entering a server loop in a user-level server that implements user-level objects, see also [l4\\_ipc\\_reply\\_and\\_wait\(\)](#).

##### Note

In case of multiple senders trying to send to the thread performing this system call, the thread receives from a sender with the highest priority. In this respect, IRQ sources have the highest priority 255.

See also

[L4 Inter-Process Communication \(IPC\)](#)

Examples

[examples/sys/ipc/ipc\\_example.c](#).

Definition at line 593 of file [ipc.h](#).

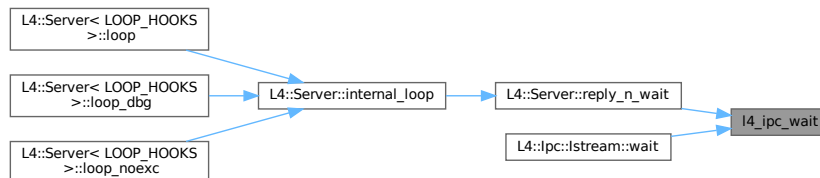
References [L4\\_INVALID\\_CAP](#), [l4\\_ipc\(\)](#), [L4\\_NOTHROW](#), [L4\\_SYSF\\_WAIT](#), and [l4\\_msgtag\\_t::raw](#).

Referenced by [L4::Server< LOOP\\_HOOKS >::reply\\_n\\_wait\(\)](#), and [L4::ipc::lstream::wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.1.9.4.11 l4\_sndfpage\_add()

```

int l4_sndfpage_add (
    l4_fpage_t const snd_fpage,
    unsigned long snd_base,
    l4_msgtag_t * tag) [inline]
  
```

Add a flexpage to be sent to the UTCB.

#### Parameters

<i>snd_fpage</i>	Flexpage.
<i>snd_base</i>	Send base.

<code>in, out</code>	<code>tag</code>	Tag to be updated. Only the number of items is incremented in the updated tag, all other members remain unmodified.
----------------------	------------------	---------------------------------------------------------------------------------------------------------------------

**Returns**

0 on success, negative error code otherwise

**See also**

[L4 Inter-Process Communication \(IPC\)](#)

Definition at line 685 of file [ipc.h](#).

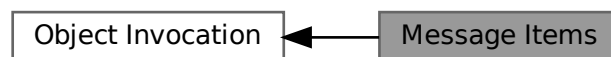
References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**14.1.9.5 Message Items**

Message-item-related functionality.

Collaboration diagram for Message Items:

**Data Structures**

- struct [l4\\_snd\\_fpage\\_t](#)  
*Send-flexpage types.*

## Enumerations

- enum `L4_obj_fpage_ctl` {  
`L4_FPAGE_C_REF_CNT` = 0x00 , `L4_FPAGE_C_NO_REF_CNT` = 0x10 , `L4_FPAGE_C_OBJ_RIGHT1` = 0x20 , `L4_FPAGE_C_OBJ_RIGHT2` = 0x40 ,  
`L4_FPAGE_C_OBJ_RIGHT3` = 0x80 , `L4_FPAGE_C_OBJ_RIGHTS` = 0xe0 , `L4_FPAGE_C_IPCGATE_SVR` = `L4_FPAGE_C_OBJ_RIGHT1` }

*Attributes and additional permissions for object send items.*

- enum `L4_fpage_cacheability_opt_t` { `L4_FPAGE_CACHE_OPT` = 0x1 , `L4_FPAGE_CACHEABLE` = 0x3 ,  
`L4_FPAGE_BUFFERABLE` = 0x5 , `L4_FPAGE_UNCACHEABLE` = 0x1 }

*Cacheability options for memory send items.*

- enum `L4_msg_item_consts_t` {  
`L4_ITEM_MAP` = 8 , `L4_ITEM_CONT` = 1 , `L4_MAP_ITEM_GRANT` = 2 , `L4_MAP_ITEM_MAP` = 0 ,  
`L4_RCV_ITEM_FORWARD_MAPPINGS` = 1 , `L4_RCV_ITEM_SINGLE_CAP` = `L4_ITEM_MAP` | 2 ,  
`L4_RCV_ITEM_LOCAL_ID` = 4 }

*Constants for message items.*

## Functions

- `L4_umword_t` `l4_map_control` (`L4_umword_t` spot, unsigned char cache, unsigned grant) `L4_NOTHROW`

*Create the first word for a map item that is a send item for the memory space.*

- `L4_umword_t` `l4_map_obj_control` (`L4_umword_t` spot, unsigned grant) `L4_NOTHROW`

*Create the first word for a map item that is a send item for the object space.*

### 14.1.9.5.1 Detailed Description

Message-item-related functionality.

Message items are typed items that are used for transferring capabilities during IPC. There are three sub-types of typed message items with variations in the layout:

1. Typed message items set by the sender in its message registers (MRs) of the UTCB for specifying what shall be sent.
2. Typed message items set by the receiver in its buffer registers (BRs) of the UTCB for specifying which types of capabilities may be received at which addresses.
3. Typed message items set by the kernel in the receiver's message registers (MRs) of the UTCB for providing information about the transfer to the receiver.

They are abbreviated by *send item*, *receive item*, and *return item*, respectively.

A typed message item in the message registers (case 1 and case 3) always consists of two words (even if it is a void item). The size of a typed message item in the buffer registers (case 2) is determined by its first word. The size is up to three words (see `L4_RCV_ITEM_SINGLE_CAP` and `L4_RCV_ITEM_FORWARD_MAPPINGS`). A void item in the buffer registers consists of a single word.

## Include File

```
#include <l4/sys/types.h>
```

### 14.1.9.5.2 Enumeration Type Documentation

#### 14.1.9.5.2.1 l4\_fpage\_cacheability\_opt\_t

enum `l4_fpage_cacheability_opt_t`

Cacheability options for memory send items.

Only the IPC sender and the thread performing the map operation can specify the caching mode of the target mapping. By default, the caching mode of the sender is used as caching mode for the target mapping. If `L4_FPAGE_CACHE_OPT` is set in the send item, the caching mode is overridden by the respective mode from below.

#### Enumerator

---

L4_FPAGE_CACHE_OPT	Enable the cacheability option in a memory send item. Without this flag, the options are copied from the sender.
L4_FPAGE_CACHEABLE	Cacheability option to enable caches for the mapping. Implies <a href="#">L4_FPAGE_CACHE_OPT</a> .
L4_FPAGE_BUFFERABLE	Cacheability option to enable buffered writes for the mapping. Implies <a href="#">L4_FPAGE_CACHE_OPT</a> .
L4_FPAGE_UNCACHEABLE	Cacheability option to disable caching for the mapping. Implies <a href="#">L4_FPAGE_CACHE_OPT</a> .

Definition at line 291 of file [\\_\\_l4\\_fpage.h](#).

#### 14.1.9.5.2.2 l4\_msg\_item\_consts\_t

enum [l4\\_msg\\_item\\_consts\\_t](#)

Constants for message items.

##### Enumerator

L4_ITEM_MAP	Identify a message item as <i>map item</i> .
L4_ITEM_CONT	Denote that the following item shall be put into the same receive item as this one.
L4_MAP_ITEM_GRANT	<p>Flag as <i>grant</i> instead of <i>map</i> operation. This means, the sender delegates access to the receiver and the kernel removes the rights from the sender (basically a move operation). The mapping in the receiver gets the new parent of any child mappings of the mapping of the sender. Rights revocation via send item/flexpage is <i>not</i> guaranteed to be applied to descendant mappings in case of grant. See <a href="#">Spaces and Mappings</a> for more details on map/grant.</p> <p><b>Note</b></p> <p>The grant operation is not performed if the resulting rights of the receiver mapping would not contain the <a href="#">L4_CAP_FPAGE_R</a> bit (for object capabilities) or none of the <a href="#">L4_FPAGE_RWX</a> bits (memory and IO ports). In that case, the mapping is not created in the receiver space and not removed from the sender space.</p> <p>If the removal of the whole mapping from the sender is not possible because the size of the mapped frame at the sender exceeds the size defined by the send or receive flexpage, the grant operation is turned into a regular map operation and the mapping is <i>not</i> removed from the sender. This would happen if, for example, a smaller part of an <a href="#">L4</a> superpage mapping shall be granted.</p>
L4_MAP_ITEM_MAP	Flag as usual <i>map</i> operation.

L4_RCV_ITEM_FORWARD_MAPPINGS	<p>This flag specifies if received capabilities shall be mapped to a particular task instead of the invoking task. This flag may be used only if <a href="#">L4_RCV_ITEM_LOCAL_ID</a> is unset.</p> <p>Setting this flag increases the size of the buffer item by one word. This word is used to specify a capability index for the task that shall receive the mappings.</p>
L4_RCV_ITEM_SINGLE_CAP	<p>Mark the receive buffer to be a small receive item that describes a buffer for a single object capability. A receive item needs to specify a <i>receive window</i>. The receive window determines which kind of capabilities (object, memory, I/O ports) may be received where in the respective space. If this flag is unset, the receive window is specified in the second word of the receive item via a <a href="#">flexpage</a>. If this flag is set, the receive window consists of a single capability index in the object space and the capability index is specified in the most significant bits of the first word of the receive item (see <a href="#">L4_CAP_SHIFT</a>).</p>
L4_RCV_ITEM_LOCAL_ID	<p>The receiver requests to receive a local ID instead of a mapping whenever possible. This flag may be used only if <a href="#">L4_RCV_ITEM_SINGLE_CAP</a> is set and <a href="#">L4_RCV_ITEM_FORWARD_MAPPINGS</a> is unset.</p> <p>When this flag is set, then,</p> <ul style="list-style-type: none"> <li>• when sender and receiver are bound to the same task, then no mapping is done for this item and just the raw flexpage (<a href="#">l4_fpage_t</a>) is transferred,</li> <li>• otherwise, when the sender specified an IPC gate for transfer that is bound to a thread that is bound to the same task as the receiving thread, then no mapping is done for this item and just the bitwise OR ( ) of the label and the <a href="#">L4_CAP_FPAGE_W</a> and <a href="#">L4_CAP_FPAGE_S</a> permissions that would have been mapped is transferred,</li> <li>• otherwise a regular mapping is done for this item.</li> </ul>

Definition at line 212 of file [consts.h](#).

#### 14.1.9.5.2.3 L4\_obj\_fpage\_ctl

```
enum L4_obj_fpage_ctl
```

Attributes and additional permissions for object send items.

These rights need to be added to the `snd_base` when mapping and control internal behavior. The exact meaning depends on the type of capability (currently used only with IPC gates).

#### Enumerator

L4_FPAGE_C_REF_CNT	Mapping is reference-counted (default).
L4_FPAGE_C_NO_REF_CNT	Don't increase the reference counter.
L4_FPAGE_C_OBJ_RIGHT1	Object-type specific right.



L4_FPAGE_C_OBJ_RIGHT2	Object-type specific right.
L4_FPAGE_C_OBJ_RIGHT3	Object-type specific right.
L4_FPAGE_C_OBJ_RIGHTS	All Object-type specific right bits.
L4_FPAGE_C_IPCGATE_SVR	The receiver may invoke IPC-gate-specific functions on the capability, e.g. bind a thread to the gate and modify the label. Needed if the receiver implements the server side of an IPC gate.

Definition at line 262 of file [\\_\\_l4\\_fpage.h](#).

### 14.1.9.5.3 Function Documentation

#### 14.1.9.5.3.1 l4\_map\_control()

```
l4_umword_t l4_map_control (
    l4_umword_t spot,
    unsigned char cache,
    unsigned grant) [inline]
```

Create the first word for a map item that is a send item for the memory space.

#### Parameters

<i>spot</i>	Hot spot address, used to determine what is actually mapped when send and receive flexpage have differing sizes.
<i>cache</i>	Cacheability hints for memory flexpages. See <a href="#">Cacheability options</a> .
<i>grant</i>	Indicates if it is a map or a grant item. Allowed values: <a href="#">L4_MAP_ITEM_MAP</a> , <a href="#">L4_MAP_ITEM_GRANT</a> .

#### Returns

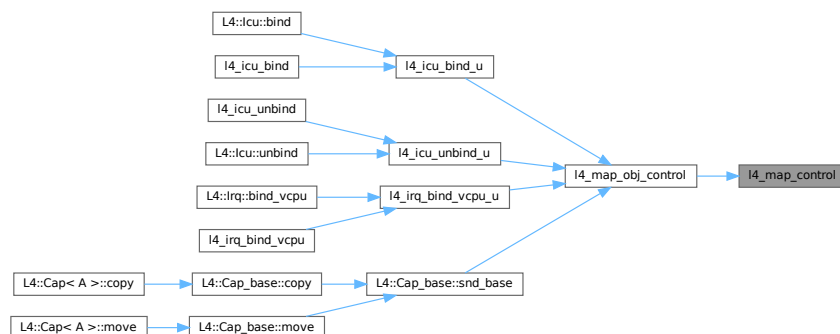
The value to be used as first word in a send item for memory.

Definition at line 742 of file [\\_\\_l4\\_fpage.h](#).

References [L4\\_FPAGE\\_CONTROL\\_MASK](#), [L4\\_ITEM\\_MAP](#), and [L4\\_NOTHROW](#).

Referenced by [l4\\_map\\_obj\\_control\(\)](#).

Here is the caller graph for this function:



#### 14.1.9.5.3.2 l4\_map\_obj\_control()

```
l4_umword_t l4_map_obj_control (
    l4_umword_t spot,
    unsigned grant) [inline]
```

Create the first word for a map item that is a send item for the object space.

#### Parameters

<i>spot</i>	Hot spot address, used to determine what is actually mapped when send and receive flexpages have different size.
<i>grant</i>	Indicates if it is a map item or a grant item. Allowed values: <a href="#">L4_MAP_ITEM_MAP</a> , <a href="#">L4_MAP_ITEM_GRANT</a> .

#### Returns

The value to be used as first word in a send item for kernel objects or IO-ports.

Definition at line 749 of file [\\_\\_l4\\_fpage.h](#).

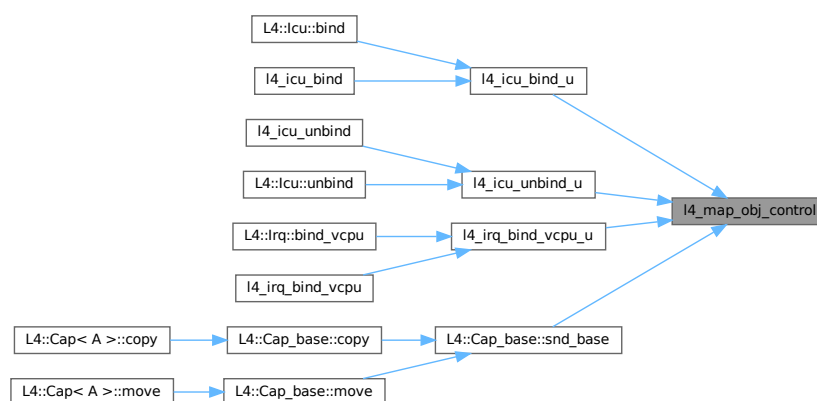
References [l4\\_map\\_control\(\)](#), and [L4\\_NOTHROW](#).

Referenced by [l4\\_icu\\_bind\\_u\(\)](#), [l4\\_icu\\_unbind\\_u\(\)](#), [l4\\_irq\\_bind\\_vcpu\\_u\(\)](#), and [L4::Cap\\_base::snd\\_base\(\)](#).

Here is the call graph for this function:



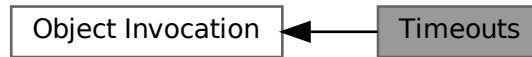
Here is the caller graph for this function:



### 14.1.9.6 Timeouts

All kinds of timeouts and time related functions.

Collaboration diagram for Timeouts:



### Data Structures

- struct [l4\\_timeout\\_s](#)  
*Basic timeout specification.*
- union [l4\\_timeout\\_t](#)  
*Timeout pair.*

### Macros

- #define [L4\\_IPC\\_TIMEOUT\\_0](#) (([l4\\_timeout\\_s](#)){0x0400})  
*Timeout constants.*
- #define [L4\\_IPC\\_TIMEOUT\\_NEVER](#) (([l4\\_timeout\\_s](#)){0})  
*never timeout*
- #define [L4\\_IPC\\_NEVER\\_INITIALIZER](#) {0}  
*never timeout, initializer*
- #define [L4\\_IPC\\_NEVER](#) (([l4\\_timeout\\_t](#)){0})  
*never timeout*
- #define [L4\\_IPC\\_RECV\\_TIMEOUT\\_0](#) (([l4\\_timeout\\_t](#)){0x00000400})  
*0 receive timeout*
- #define [L4\\_IPC\\_SEND\\_TIMEOUT\\_0](#) (([l4\\_timeout\\_t](#)){0x04000000})  
*0 send timeout*
- #define [L4\\_IPC\\_BOTH\\_TIMEOUT\\_0](#) (([l4\\_timeout\\_t](#)){0x04000400})  
*0 receive and send timeout*
- #define [L4\\_TIMEOUT\\_US\\_NEVER](#) (~0ULL)  
*The waiting period in microseconds which is interpreted as "never" by [l4\\_timeout\\_from\\_us\(\)](#).*
- #define [L4\\_TIMEOUT\\_US\\_MAX](#) ((1ULL << 41) - 1)  
*The longest waiting period in microseconds accepted by [l4\\_timeout\\_from\\_us\(\)](#).*

### Typedefs

- typedef struct [l4\\_timeout\\_s](#) [l4\\_timeout\\_s](#)  
*Basic timeout specification.*
- typedef union [l4\\_timeout\\_t](#) [l4\\_timeout\\_t](#)  
*Timeout pair.*

## Functions

- [L4\\_CONSTEXPR l4\\_timeout\\_s l4\\_timeout\\_rel](#) (unsigned man, unsigned exp) [L4\\_NOTHROW](#)  
*Get relative timeout consisting of mantissa and exponent.*
- [L4\\_CONSTEXPR l4\\_timeout\\_t l4\\_ipc\\_timeout](#) (unsigned snd\_man, unsigned snd\_exp, unsigned rcv\_man, unsigned rcv\_exp) [L4\\_NOTHROW](#)  
*Convert explicit timeout values to [l4\\_timeout\\_t](#) type.*
- [L4\\_CONSTEXPR l4\\_timeout\\_t l4\\_timeout](#) (l4\_timeout\_s snd, l4\_timeout\_s rcv) [L4\\_NOTHROW](#)  
*Combine send and receive timeout in a timeout.*
- [L4\\_CONSTEXPR void l4\\_snd\\_timeout](#) (l4\_timeout\_s snd, l4\_timeout\_t \*to) [L4\\_NOTHROW](#)  
*Set send timeout in given to timeout.*
- [L4\\_CONSTEXPR void l4\\_rcv\\_timeout](#) (l4\_timeout\_s rcv, l4\_timeout\_t \*to) [L4\\_NOTHROW](#)  
*Set receive timeout in given to timeout.*
- [L4\\_CONSTEXPR l4\\_kernel\\_clock\\_t l4\\_timeout\\_rel\\_get](#) (l4\_timeout\_s to) [L4\\_NOTHROW](#)  
*Get clock value of out timeout.*
- [L4\\_CONSTEXPR unsigned l4\\_timeout\\_is\\_absolute](#) (l4\_timeout\_s to) [L4\\_NOTHROW](#)  
*Return whether the given timeout is absolute or not.*
- [L4\\_CONSTEXPR l4\\_kernel\\_clock\\_t l4\\_timeout\\_get](#) (l4\_kernel\_clock\_t cur, l4\_timeout\_s to) [L4\\_NOTHROW](#)  
*Get clock value for a clock + a timeout.*
- [l4\\_timeout\\_s l4\\_timeout\\_abs](#) (l4\_kernel\_clock\_t pint, int br) [L4\\_NOTHROW](#)  
*Set an absolute timeout.*
- unsigned [l4\\_utcb\\_mr64\\_idx](#) (unsigned idx) [L4\\_NOTHROW](#)  
*Get index into 64bit message registers alias from native-sized index.*

### 14.1.9.6.1 Detailed Description

All kinds of timeouts and time related functions.

### 14.1.9.6.2 Macro Definition Documentation

#### 14.1.9.6.2.1 L4\_IPC\_TIMEOUT\_0

```
#define L4_IPC_TIMEOUT_0 ((l4_timeout_s){0x0400})
```

Timeout constants.

0 timeout

Definition at line 73 of file [\\_\\_timeout.h](#).

Referenced by [L4::lpc\\_svr::Timeout\\_queue\\_hooks<HOOKS, BR\\_MAN>::timeout\(\)](#).

#### 14.1.9.6.2.2 L4\_TIMEOUT\_US\_MAX

```
#define L4_TIMEOUT_US_MAX ((1ULL << 41) - 1)
```

The longest waiting period in microseconds accepted by [l4\\_timeout\\_from\\_us\(\)](#).

See [l4\\_timeout\\_from\\_us\(\)](#) for an explanation.

Definition at line 91 of file [\\_\\_timeout.h](#).

### 14.1.9.6.3 Typedef Documentation

#### 14.1.9.6.3.1 l4\_timeout\_s

```
typedef struct l4_timeout_s l4_timeout_s
```

Basic timeout specification.

If bit 15 == 0, basically a floating point number with 10 bits mantissa and 5 bits exponent ( $t = m \cdot 2^e$ ).

If the mantissa is zero, the exponent encodes special values, see [L4\\_IPC\\_TIMEOUT\\_0](#) and [L4\\_IPC\\_TIMEOUT\\_NEVER](#).

If bit 15 == 1 the timeout is absolute and the lower 6 bits encode the index of the UTCB buffer register(s) holding the absolute 64-bit timeout value. On 32-bit systems, two consecutive UTCB buffer registers are used.

#### 14.1.9.6.3.2 l4\_timeout\_t

```
typedef union l4_timeout_t l4_timeout_t
```

Timeout pair.

For IPC there are usually a send and a receive timeout. So this structure contains a pair of timeouts.

### 14.1.9.6.4 Function Documentation

#### 14.1.9.6.4.1 l4\_ipc\_timeout()

```
L4_CONSTEXPR l4_timeout_t l4_ipc_timeout (
    unsigned snd_man,
    unsigned snd_exp,
    unsigned rcv_man,
    unsigned rcv_exp) [inline]
```

Convert explicit timeout values to [l4\\_timeout\\_t](#) type.

#### Parameters

<i>snd_man</i>	Mantissa of send timeout.
<i>snd_exp</i>	Exponent of send timeout.
<i>rcv_man</i>	Mantissa of receive timeout.
<i>rcv_exp</i>	Exponent of receive timeout.

Definition at line 203 of file [\\_\\_timeout.h](#).

References [L4\\_NOTHROW](#), and [l4\\_timeout\(\)](#).

Here is the call graph for this function:



#### 14.1.9.6.4.2 l4\_rcv\_timeout()

```
L4_CONSTEXPR void l4_rcv_timeout (
    l4_timeout_s rcv,
    l4_timeout_t * to) [inline]
```

Set receive timeout in given to timeout.

##### Parameters

	<i>rcv</i>	Receive timeout
out	<i>to</i>	L4 timeout

Definition at line 227 of file [\\_\\_timeout.h](#).

References [L4\\_NOTHROW](#).

#### 14.1.9.6.4.3 l4\_snd\_timeout()

```
L4_CONSTEXPR void l4_snd_timeout (
    l4_timeout_s snd,
    l4_timeout_t * to) [inline]
```

Set send timeout in given to timeout.

##### Parameters

	<i>snd</i>	Send timeout
out	<i>to</i>	L4 timeout

Definition at line 220 of file [\\_\\_timeout.h](#).

References [L4\\_NOTHROW](#).

#### 14.1.9.6.4.4 l4\_timeout()

```
L4_CONSTEXPR l4_timeout_t l4_timeout (
    l4_timeout_s snd,
    l4_timeout_s rcv) [inline]
```

Combine send and receive timeout in a timeout.

##### Parameters

<i>snd</i>	Send timeout
<i>rcv</i>	Receive timeout

## Returns

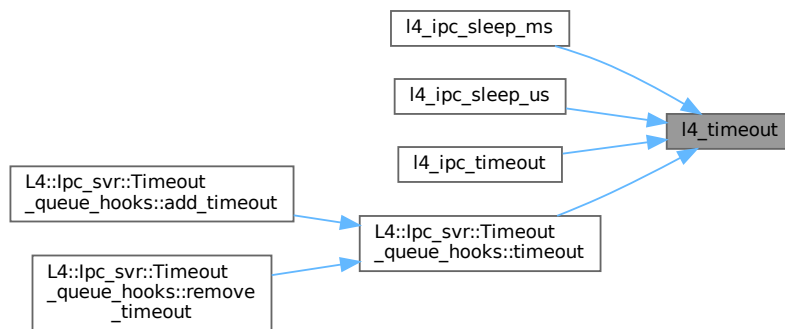
L4 timeout

Definition at line 213 of file `__timeout.h`.

References `L4_NOTHROW`.

Referenced by `l4_ipc_sleep_ms()`, `l4_ipc_sleep_us()`, `l4_ipc_timeout()`, and `L4::lpc_svr::Timeout_queue_hooks<HOOKS, BR_MAN`

Here is the caller graph for this function:



#### 14.1.9.6.4.5 l4\_timeout\_abs()

```
l4_timeout_s l4_timeout_abs (
    l4_kernel_clock_t pint,
    int br) [inline]
```

Set an absolute timeout.

## Parameters

<i>pint</i>	Point in time in clocks
<i>br</i>	The buffer register the timeout shall be placed in. (

## Note

On 32bit architectures the timeout needs two consecutive buffers.)

The absolute timeout value will be placed into the buffer register *br* of the current thread.

**Returns**

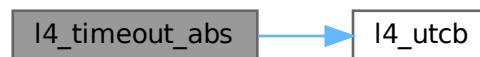
timeout value

Definition at line 389 of file [utcb.h](#).

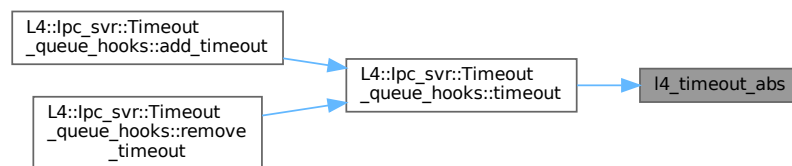
References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Referenced by [L4::lpc\\_svr::Timeout\\_queue\\_hooks<HOOKS, BR\\_MAN>::timeout\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**14.1.9.6.4.6 l4\_timeout\_get()**

```

L4_CONSTEXPR l4_kernel_clock_t l4_timeout_get (
    l4_kernel_clock_t cur,
    l4_timeout_s to) [inline]
  
```

Get clock value for a clock + a timeout.

**Parameters**

<i>cur</i>	Clock value
<i>to</i>	<a href="#">L4</a> timeout



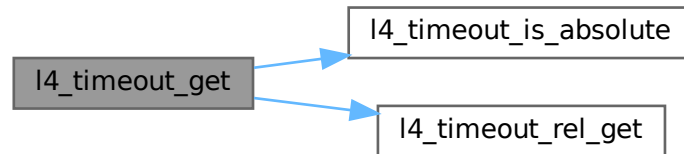
**Returns**

Clock sum

Definition at line 257 of file [\\_\\_timeout.h](#).

References [L4\\_NOTHROW](#), [l4\\_timeout\\_is\\_absolute\(\)](#), and [l4\\_timeout\\_rel\\_get\(\)](#).

Here is the call graph for this function:

**14.1.9.6.4.7 l4\_timeout\_is\_absolute()**

```
L4_CONSTEXPR unsigned l4_timeout_is_absolute (  
    l4_timeout_s to) [inline]
```

Return whether the given timeout is absolute or not.

**Parameters**

<i>to</i>	<a href="#">L4 timeout</a>
-----------	----------------------------

**Returns**

!= 0 if absolute, 0 if relative

Definition at line 250 of file [\\_\\_timeout.h](#).

References [L4\\_NOTHROW](#).

Referenced by [l4\\_timeout\\_get\(\)](#).

Here is the caller graph for this function:



#### 14.1.9.6.4.8 l4\_timeout\_rel()

```
L4_CONSTEXPR l4_timeout_s l4_timeout_rel (  
    unsigned man,  
    unsigned exp) [inline]
```

Get relative timeout consisting of mantissa and exponent.

##### Parameters

<i>man</i>	Mantissa of timeout
<i>exp</i>	Exponent of timeout

##### Returns

timeout value

Definition at line 234 of file [\\_\\_timeout.h](#).

References [L4\\_NOTHROW](#).

#### 14.1.9.6.4.9 l4\_timeout\_rel\_get()

```
L4_CONSTEXPR l4_kernel_clock_t l4_timeout_rel_get (  
    l4_timeout_s to) [inline]
```

Get clock value of out timeout.

##### Parameters

<i>to</i>	<a href="#">L4</a> timeout
-----------	----------------------------

##### Returns

Clock value

Definition at line 241 of file [\\_\\_timeout.h](#).

References [L4\\_NOTHROW](#).

Referenced by [l4\\_timeout\\_get\(\)](#).

Here is the caller graph for this function:



#### 14.1.9.6.4.10 l4\_utcb\_mr64\_idx()

```
unsigned l4_utcb_mr64_idx (  
    unsigned idx) [inline]
```

Get index into 64bit message registers alias from native-sized index.

##### Parameters

<i>idx</i>	Index to native-sized message register
------------	----------------------------------------

##### Returns

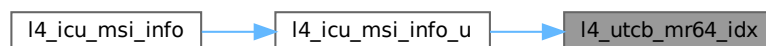
Index to 64bit message register alias

Definition at line 392 of file [utcb.h](#).

References [L4\\_INLINE](#), and [L4\\_NOTHROW](#).

Referenced by [l4\\_icu\\_msi\\_info\\_u\(\)](#).

Here is the caller graph for this function:



#### 14.1.9.7 Error Handling

Error handling for [L4](#) object invocation.

Collaboration diagram for Error Handling:



## Enumerations

- enum [l4\\_ipc\\_tcr\\_error\\_t](#) {  
[L4\\_IPC\\_ERROR\\_MASK](#) = 0x1F , [L4\\_IPC\\_SND\\_ERR\\_MASK](#) = 0x01 , [L4\\_IPC\\_ENOT\\_EXISTENT](#) = 0x04 ,  
[L4\\_IPC\\_RETIMEOUT](#) = 0x03 ,  
[L4\\_IPC\\_SETIMEOUT](#) = 0x02 , [L4\\_IPC\\_RECANCELED](#) = 0x07 , [L4\\_IPC\\_SECANCELED](#) = 0x06 ,  
[L4\\_IPC\\_REMAPFAILED](#) = 0x11 ,  
[L4\\_IPC\\_SEMAPFAILED](#) = 0x10 , [L4\\_IPC\\_RESNDPFTO](#) = 0x0b , [L4\\_IPC\\_SESNDPFTO](#) = 0x0a ,  
[L4\\_IPC\\_RERCVPFTO](#) = 0x0d ,  
[L4\\_IPC\\_SERCVPFTO](#) = 0x0c , [L4\\_IPC\\_REABORTED](#) = 0x0f , [L4\\_IPC\\_SEABORTED](#) = 0x0e ,  
[L4\\_IPC\\_REMSGCUT](#) = 0x09 ,  
[L4\\_IPC\\_SEMSGCUT](#) = 0x08 }

*Error codes in the error TCR.*

## Functions

- [l4\\_umword\\_t l4\\_ipc\\_error](#) ([l4\\_msgtag\\_t](#) tag, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Get the IPC error code for an IPC operation.*
- long [l4\\_error](#) ([l4\\_msgtag\\_t](#) tag) [L4\\_NOTHROW](#)  
*Get IPC error code if any or message tag label otherwise for an IPC call.*
- int [l4\\_ipc\\_is\\_snd\\_error](#) ([l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Returns whether an error occurred in send phase of an invocation.*
- int [l4\\_ipc\\_is\\_rcv\\_error](#) ([l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Returns whether an error occurred in receive phase of an invocation.*
- int [l4\\_ipc\\_error\\_code](#) ([l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Get the error condition of the last invocation from the TCR.*

### 14.1.9.7.1 Detailed Description

Error handling for [L4](#) object invocation.

#### Include File

```
#include <l4/sys/ipc.h>
```

### 14.1.9.7.2 Enumeration Type Documentation

#### 14.1.9.7.2.1 [l4\\_ipc\\_tcr\\_error\\_t](#)

```
enum l4\_ipc\_tcr\_error\_t
```

Error codes in the *error* TCR.

The error codes are accessible via the *error* TCR, see [l4\\_thread\\_regs\\_t.error](#).

#### Enumerator

<a href="#">L4_IPC_ERROR_MASK</a>	Mask for error bits.
-----------------------------------	----------------------

L4_IPC_SND_ERR_MASK	Send error mask.
L4_IPC_ENOT_EXISTENT	Non-existing destination or source.
L4_IPC_RETIMEOUT	Timeout during receive operation.
L4_IPC_SETIMEOUT	Timeout during send operation.
L4_IPC_RECANCELED	Receive operation canceled.
L4_IPC_SECANCELED	Send operation canceled.
L4_IPC_REMAPFAILED	Map flexpage failed in receive operation.
L4_IPC_SEMAPFAILED	Map flexpage failed in send operation.
L4_IPC_RESNDPFTO	Send-pagefault timeout in receive operation.
L4_IPC_SESNDPFTO	Send-pagefault timeout in send operation.
L4_IPC_RERCVPFTO	Receive-pagefault timeout in receive operation.
L4_IPC_SERCVPFTO	Receive-pagefault timeout in send operation.
L4_IPC_REABORTED	Receive operation aborted.
L4_IPC_SEABORTED	Send operation aborted.
L4_IPC_REMSGCUT	Received message truncated. Usually returned when the typed items to be sent by the IPC partner exceed the buffer registers of the respective types.
L4_IPC_SEMSGCUT	Sent message truncated. Usually returned when the typed items to be sent exceed the IPC partner's buffer registers of the respective types.

Definition at line 81 of file [ipc.h](#).

### 14.1.9.7.3 Function Documentation

#### 14.1.9.7.3.1 l4\_error()

```
long l4_error (
    l4_msgtag_t tag) [inline]
```

Get IPC error code if any or message tag label otherwise for an IPC call.

This function shall only be used if the IPC operation includes a receive phase (usually a call operation), otherwise no tag label is received and the return value of this function is undefined.

#### Parameters

<i>tag</i>	Message tag returned by the IPC call.
------------	---------------------------------------

#### Returns

In case of an IPC error, a negative error code in the range of [L4\\_EIPC\\_LO](#) to [L4\\_EIPC\\_HI](#) (see [l4\\_ipc\\_to\\_errno\(\)](#) and [l4\\_ipc\\_tcr\\_error\\_t](#)), otherwise the tag label. By convention, the callee can signal errors via a negative tag label (negated value from [l4\\_error\\_code\\_t](#)) and success via a non-negative value.

#### Examples

[examples/libs/l4re/streammap/client.cc](#), [examples/sys/aliens/main.c](#), [examples/sys/isr/main.c](#), [examples/sys/migrate/thread\\_migration.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 646 of file [ipc.h](#).

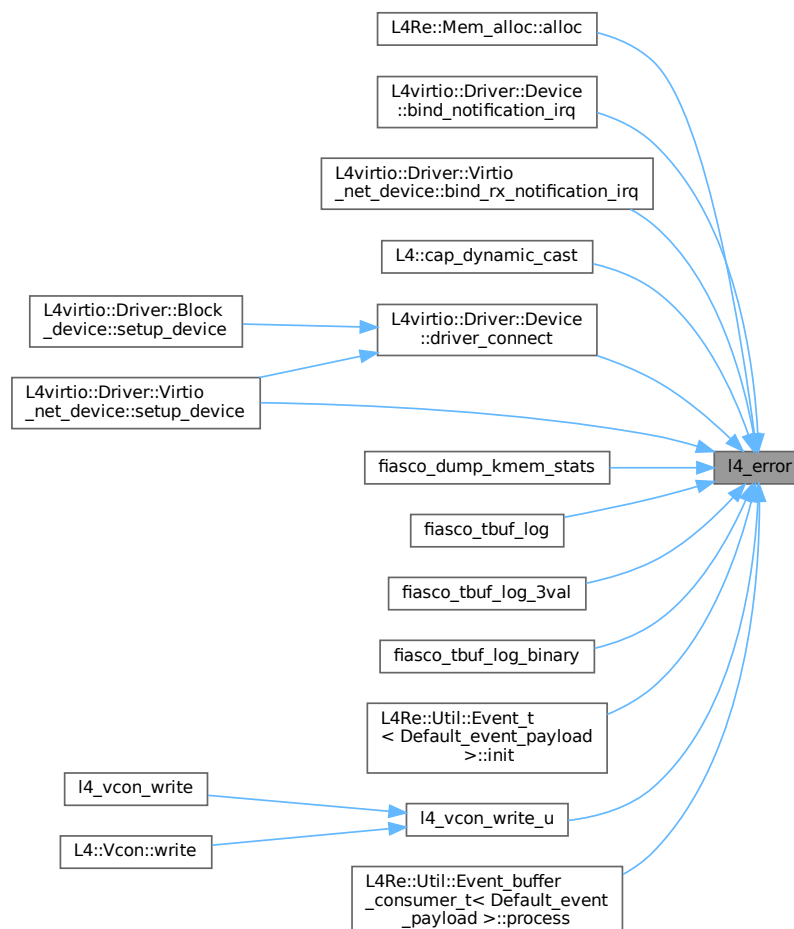
References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Referenced by [L4Re::Mem\\_alloc::alloc\(\)](#), [L4virtio::Driver::Device::bind\\_notification\\_irq\(\)](#), [L4virtio::Driver::Virtio\\_net\\_device::bind\\_rx\\_notification\\_irq\(\)](#), [L4::cap\\_dynamic\\_cast\(\)](#), [L4virtio::Driver::Device::driver\\_connect\(\)](#), [fiasco\\_dump\\_kmem\\_stats\(\)](#), [fiasco\\_tbuf\\_log\(\)](#), [fiasco\\_tbuf\\_log\\_3val\(\)](#), [fiasco\\_tbuf\\_log\\_binary\(\)](#), [L4Re::Util::Event\\_t< Default\\_event\\_payload >::init\(\)](#), [l4\\_vcon\\_write\\_u\(\)](#), [L4Re::Util::Event\\_buffer\\_consumer\\_t< Default\\_event\\_payload >::process\(\)](#), and [L4virtio::Driver::Virtio\\_net\\_device::setup\\_device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 14.1.9.7.3.2 l4\_ipc\_error()

```
l4_umword_t l4_ipc_error (
    l4_msgtag_t tag,
    l4_utcb_t * utcb) [inline]
```

Get the IPC error code for an IPC operation.

#### Parameters

<i>tag</i>	Message tag returned by the IPC operation.
<i>utcb</i>	UTCB that was used for the IPC operation.

#### Returns

0 if no error condition is set, error code otherwise (see [l4\\_ipc\\_tcr\\_error\\_t](#)).

#### Examples

[examples/sys/ipc/ipc\\_example.c](#), and [examples/sys/start-with-exc/main.c](#).

Definition at line 629 of file [ipc.h](#).

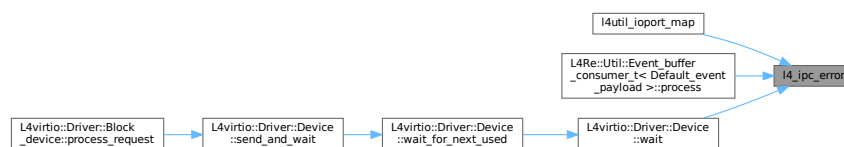
References [L4\\_IPC\\_ERROR\\_MASK](#), [L4\\_LIKELY](#), [l4\\_msgtag\\_has\\_error\(\)](#), and [L4\\_NOTHROW](#).

Referenced by [l4util\\_ioport\\_map\(\)](#), [L4Re::Util::Event\\_buffer\\_consumer\\_t< Default\\_event\\_payload >::process\(\)](#), and [L4virtio::Driver::Device::wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.1.9.7.3.3 `l4_ipc_error_code()`

```
int l4_ipc_error_code (
    l4_utcb_t * utcb) [inline]
```

Get the error condition of the last invocation from the TCR.

##### Precondition

`l4_msgtag_has_error(tag) == true`

##### Parameters

<i>utcb</i>	UTCB to check.
-------------	----------------

##### Returns

Error condition of type [l4\\_ipc\\_tcr\\_error\\_t](#).

Definition at line 658 of file [ipc.h](#).

References [L4\\_INLINE](#), [L4\\_IPC\\_ERROR\\_MASK](#), and [L4\\_NOTHROW](#).

#### 14.1.9.7.3.4 `l4_ipc_is_rcv_error()`

```
int l4_ipc_is_rcv_error (
    l4_utcb_t * utcb) [inline]
```

Returns whether an error occurred in receive phase of an invocation.

##### Precondition

`l4_msgtag_has_error(tag) == true`

##### Parameters

<i>utcb</i>	UTCB to check.
-------------	----------------

##### Returns

Boolean value.

Definition at line 655 of file [ipc.h](#).

References [L4\\_INLINE](#), and [L4\\_NOTHROW](#).

#### 14.1.9.7.3.5 `l4_ipc_is_snd_error()`

```
int l4_ipc_is_snd_error (
    l4_utcb_t * utcb) [inline]
```

Returns whether an error occurred in send phase of an invocation.

##### Precondition

`l4_msgtag_has_error(tag) == true`

##### Parameters

---



<i>utcb</i>	UTCB to check.
-------------	----------------

#### Returns

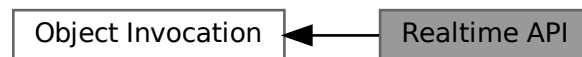
Boolean value.

Definition at line 652 of file [ipc.h](#).

References [L4\\_INLINE](#), and [L4\\_NOTHROW](#).

#### 14.1.9.8 Realtime API

Collaboration diagram for Realtime API:



#### 14.1.9.9 Message Tag

API related to the message tag data type.

Collaboration diagram for Message Tag:



#### Data Structures

- struct [l4\\_msgtag\\_t](#)  
*Message tag data structure.*

#### Typedefs

- typedef struct l4\_msgtag\_t [l4\\_msgtag\\_t](#)  
*Message tag data structure.*

## Enumerations

- enum `L4_platform_ctl_proto` { `L4_PROTO_PLATFORM_CTL` = 0 }  
*Predefined protocol type for messages to platform-control objects.*
- enum `L4_msgtag_protocol` {  
`L4_PROTO_NONE` = 0 , `L4_PROTO_ALLOW_SYSCALL` = 1 , `L4_PROTO_PF_EXCEPTION` = 1 ,  
`L4_PROTO_IRQ` = -1L ,  
`L4_PROTO_PAGE_FAULT` = -2L , `L4_PROTO_EXCEPTION` = -5L , `L4_PROTO_SIGMA0` = -6L ,  
`L4_PROTO_IO_PAGE_FAULT` = -8L ,  
`L4_PROTO_THREAD_GROUP` = -9L , `L4_PROTO_KOBJECT` = -10L , `L4_PROTO_TASK` = -11L ,  
`L4_PROTO_THREAD` = -12L ,  
`L4_PROTO_LOG` = -13L , `L4_PROTO_SCHEDULER` = -14L , `L4_PROTO_FACTORY` = -15L ,  
`L4_PROTO_VM` = -16L ,  
`L4_PROTO_DMA_SPACE` = -17L , `L4_PROTO_IRQ_SENDER` = -18L , `L4_PROTO_SEMAPHORE` = -20L ,  
`L4_PROTO_META` = -21L ,  
`L4_PROTO_IOMMU` = -22L , `L4_PROTO_DEBUGGER` = -23L , `L4_PROTO_SMCCC` = -24L ,  
`L4_PROTO_VCPU_CONTEXT` = -25L }  
*Message tag for IPC operations.*
- enum `L4_msgtag_flags` { `L4_MSGTAG_ERROR` , `L4_MSGTAG_TRANSFER_FPU` , `L4_MSGTAG_SCHEDULE` ,  
`L4_MSGTAG_PROPAGATE` = 0x4000 , `L4_MSGTAG_FLAGS` }  
*Flags for message tags.*

## Functions

- `l4_msgtag_t l4_msgtag` (long label, unsigned words, unsigned items, unsigned flags) `L4_NOTHROW`  
*Create a message tag from the specified values.*
- long `l4_msgtag_label` (`l4_msgtag_t` t) `L4_NOTHROW`  
*Get the protocol of tag.*
- unsigned `l4_msgtag_words` (`l4_msgtag_t` t) `L4_NOTHROW`  
*Get the number of untyped words.*
- unsigned `l4_msgtag_items` (`l4_msgtag_t` t) `L4_NOTHROW`  
*Get the number of typed items.*
- unsigned `l4_msgtag_flags` (`l4_msgtag_t` t) `L4_NOTHROW`  
*Get the flags.*
- unsigned `l4_msgtag_has_error` (`l4_msgtag_t` t) `L4_NOTHROW`  
*Test for error indicator flag.*
- unsigned `l4_msgtag_is_page_fault` (`l4_msgtag_t` t) `L4_NOTHROW`  
*Test for page-fault protocol.*
- unsigned `l4_msgtag_is_exception` (`l4_msgtag_t` t) `L4_NOTHROW`  
*Test for exception protocol.*
- unsigned `l4_msgtag_is_sigma0` (`l4_msgtag_t` t) `L4_NOTHROW`  
*Test for sigma0 protocol.*
- unsigned `l4_msgtag_is_io_page_fault` (`l4_msgtag_t` t) `L4_NOTHROW`  
*Test for IO-page-fault protocol.*

### 14.1.9.9.1 Detailed Description

API related to the message tag data type.

#### Include File

```
#include <l4/sys/types.h>
```

### 14.1.9.9.2 Typedef Documentation

#### 14.1.9.9.2.1 l4\_msgtag\_t

```
typedef struct l4_msgtag_t l4_msgtag_t
```

Message tag data structure.

#### Include File

```
#include <l4/sys/types.h>
```

Describes the details of an IPC operation, in particular which parts of the UTCB have to be transmitted, and also flags to enable real-time and FPU extensions.

The message tag also contains a user-defined label that could be used to specify a protocol ID. Some negative values are reserved for kernel protocols such as page faults and exceptions.

The type must be treated completely opaque.

### 14.1.9.9.3 Enumeration Type Documentation

#### 14.1.9.9.3.1 L4\_msgtag\_flags

```
enum L4_msgtag_flags
```

Flags for message tags.

#### Enumerator

L4_MSGTAG_ERROR	Error indicator flag.
L4_MSGTAG_TRANSFER_FPU	Enable FPU transfer flag for IPC. By enabling this flag when sending IPC, the sender indicates that the contents of the FPU shall be transferred to the receiving thread. However, the receiver has to indicate its willingness to receive FPU context in its buffer descriptor register (BDR).
L4_MSGTAG_SCHEDULE	Enable schedule in IPC flag. Usually IPC operations donate the remaining time slice of a thread to the called thread. Enabling this flag when sending IPC does a real scheduling decision. However, this flag decreases IPC performance.
L4_MSGTAG_FLAGS	Mask for all flags.

Definition at line 86 of file [types.h](#).

#### 14.1.9.9.3.2 L4\_msgtag\_protocol

```
enum L4_msgtag_protocol
```

Message tag for IPC operations.

All predefined protocols used by the kernel.

#### Enumerator

L4_PROTO_NONE	Default protocol tag to reply to kernel.
L4_PROTO_ALLOW_SYSCALL	Allow an alien the system call.
L4_PROTO_PF_EXCEPTION	Make an exception out of a page fault.
L4_PROTO_IRQ	IRQ message.
L4_PROTO_PAGE_FAULT	Page fault message.
L4_PROTO_EXCEPTION	Exception.
L4_PROTO_SIGMA0	Sigma0 protocol.
L4_PROTO_IO_PAGE_FAULT	I/O page fault message.
L4_PROTO_THREAD_GROUP	Protocol for messages to a thread group obj.
L4_PROTO_KOBJECT	Protocol for messages to a generic kobject.
L4_PROTO_TASK	Protocol for messages to a task object.
L4_PROTO_THREAD	Protocol for messages to a thread object.
L4_PROTO_LOG	Protocol for messages to a log object.
L4_PROTO_SCHEDULER	Protocol for messages to a scheduler object.
L4_PROTO_FACTORY	Protocol for messages to a factory object.
L4_PROTO_VM	Protocol for messages to a virtual machine object.
L4_PROTO_DMA_SPACE	Protocol for (creating) kernel DMA space objects.
L4_PROTO_IRQ_SENDER	Protocol for IRQ senders (IRQ -> IPC).
L4_PROTO_SEMAPHORE	Protocol for semaphore objects.
L4_PROTO_META	Meta information protocol.
L4_PROTO_IOMMU	Protocol ID for IO-MMUs.
L4_PROTO_DEBUGGER	Protocol ID for the debugger.
L4_PROTO_SMCCC	Protocol ID for ARM SMCCC calls.
L4_PROTO_VCPU_CONTEXT	Protocol for hardware vCPU contexts.

Definition at line 38 of file [types.h](#).

#### 14.1.9.9.3.3 L4\_platform\_ctl\_proto

```
enum L4_platform_ctl_proto
```

Predefined protocol type for messages to platform-control objects.

##### Enumerator

L4_PROTO_PLATFORM_CTL	Protocol messages to a platform control object. See <a href="#">L4_platform_ctl_ops</a> for allowed operations.
-----------------------	-----------------------------------------------------------------------------------------------------------------

Definition at line 174 of file [platform\\_control.h](#).

#### 14.1.9.9.4 Function Documentation

##### 14.1.9.9.4.1 l4\_msgtag()

```
l4_msgtag_t l4_msgtag (
    long label,
    unsigned words,
    unsigned items,
    unsigned flags) [inline]
```

Create a message tag from the specified values.

Message tag functions.

#### Parameters

<i>label</i>	The user-defined label
<i>words</i>	The number of untyped words within the UTCB
<i>items</i>	The number of typed items (e.g., flexpages) within the UTCB
<i>flags</i>	The IPC flags for realtime and FPU extensions

#### Returns

Message tag

#### Examples

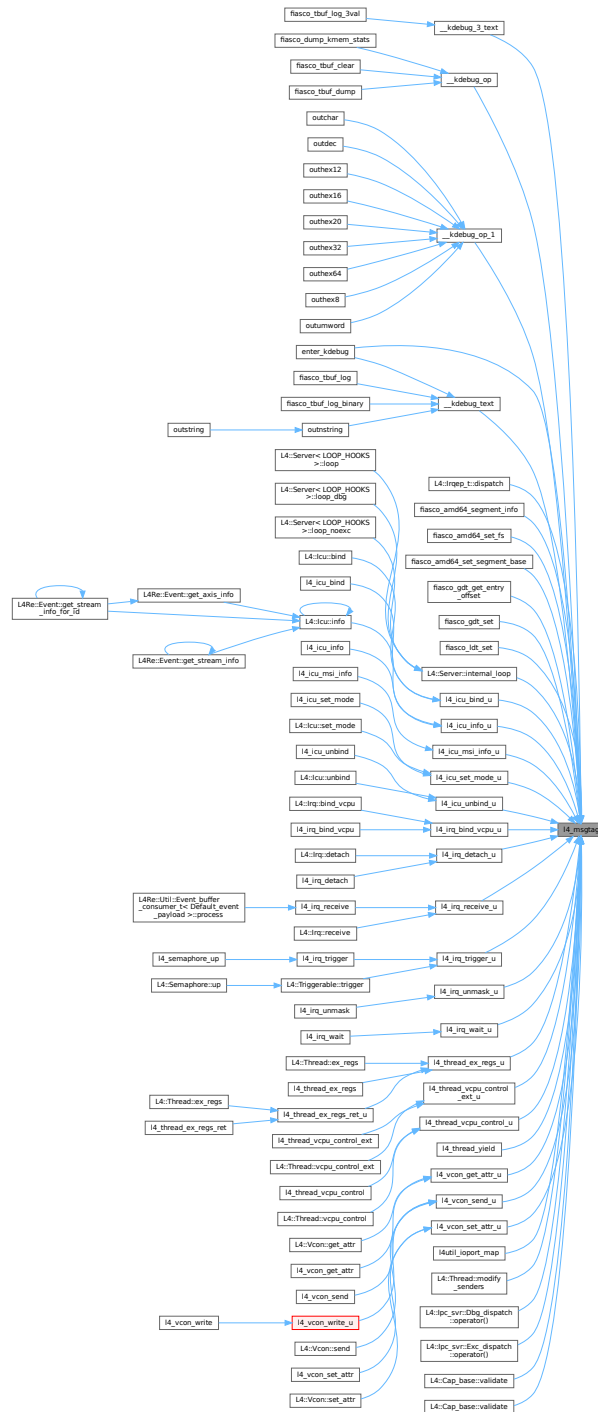
[examples/sys/aliens/main.c](#), [examples/sys/ipc/ipc\\_example.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 405 of file [types.h](#).

References [L4\\_NOTHROW](#).

Referenced by [\\_\\_kdebug\\_3\\_text\(\)](#), [\\_\\_kdebug\\_op\(\)](#), [\\_\\_kdebug\\_op\\_1\(\)](#), [\\_\\_kdebug\\_text\(\)](#), [L4::Irqep\\_t< Derived, BASE, bool >::dispatch\(\)](#), [enter\\_kdebug\(\)](#), [fiasco\\_amd64\\_segment\\_info\(\)](#), [fiasco\\_amd64\\_set\\_fs\(\)](#), [fiasco\\_amd64\\_set\\_segment\\_base\(\)](#), [fiasco\\_gdt\\_get\\_entry\\_offset\(\)](#), [fiasco\\_gdt\\_set\(\)](#), [fiasco\\_ldt\\_set\(\)](#), [L4::Server< LOOP\\_HOOKS >::internal\\_loop\(\)](#), [l4\\_icu\\_bind\\_u\(\)](#), [l4\\_icu\\_info\\_u\(\)](#), [l4\\_icu\\_msi\\_info\\_u\(\)](#), [l4\\_icu\\_set\\_mode\\_u\(\)](#), [l4\\_icu\\_unbind\\_u\(\)](#), [l4\\_irq\\_bind\\_vcpu\\_u\(\)](#), [l4\\_irq\\_detach\\_u\(\)](#), [l4\\_irq\\_receive\\_u\(\)](#), [l4\\_irq\\_trigger\\_u\(\)](#), [l4\\_irq\\_unmask\\_u\(\)](#), [l4\\_irq\\_wait\\_u\(\)](#), [l4\\_thread\\_ex\\_regs\\_u\(\)](#), [l4\\_thread\\_vcpu\\_control\\_ext\\_u\(\)](#), [l4\\_thread\\_vcpu\\_control\\_u\(\)](#), [l4\\_thread\\_yield\(\)](#), [l4\\_vcon\\_get\\_attr\\_u\(\)](#), [l4\\_vcon\\_send\\_u\(\)](#), [l4\\_vcon\\_set\\_attr\\_u\(\)](#), [l4util\\_ioport\\_map\(\)](#), [L4::Thread::modify\\_senders\(\)](#), [L4::lpc\\_svr::Dbg\\_dispatch< R, Exc, Printer >::operator\(\)\(\)](#), [L4::lpc\\_svr::Exc\\_dispatch< R, Exc >::operator\(\)\(\)](#), [L4::Cap\\_base::validate\(\)](#), and [L4::Cap\\_base::validate\(\)](#).

Here is the caller graph for this function:



#### 14.1.9.9.4.2 l4\_msgtag\_flags()

```
unsigned l4_msgtag_flags (
    l4_msgtag_t t) [inline]
```

Get the flags.

The flag are defined by [L4\\_msgtag\\_flags](#).

#### Parameters

---





<i>t</i>	The tag
----------	---------

**Returns**

Boolean value

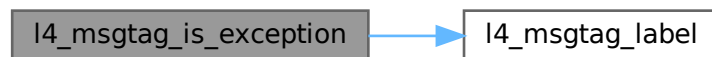
**Examples**

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), and [examples/sys/start-with-exc/main.c](#).

Definition at line 448 of file [types.h](#).

References [L4\\_INLINE](#), [l4\\_msgtag\\_label\(\)](#), [L4\\_NOTHROW](#), and [L4\\_PROTO\\_EXCEPTION](#).

Here is the call graph for this function:

**14.1.9.9.4.5 l4\_msgtag\_is\_io\_page\_fault()**

```
unsigned l4_msgtag_is_io_page_fault (  
    l4_msgtag_t t) [inline]
```

Test for IO-page-fault protocol.

**Parameters**

<i>t</i>	The tag
----------	---------

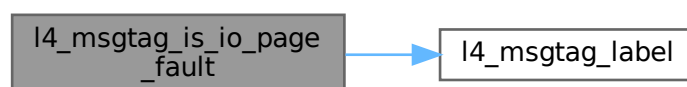
**Returns**

Boolean value

Definition at line 454 of file [types.h](#).

References [L4\\_INLINE](#), [l4\\_msgtag\\_label\(\)](#), [L4\\_NOTHROW](#), and [L4\\_PROTO\\_IO\\_PAGE\\_FAULT](#).

Here is the call graph for this function:



#### 14.1.9.9.4.6 l4\_msgtag\_is\_page\_fault()

```
unsigned l4_msgtag_is_page_fault (  
    l4_msgtag_t t) [inline]
```

Test for page-fault protocol.

##### Parameters

<i>t</i>	The tag
----------	---------

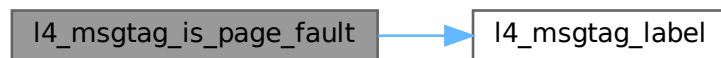
##### Returns

Boolean value

Definition at line 445 of file [types.h](#).

References [L4\\_INLINE](#), [l4\\_msgtag\\_label\(\)](#), [L4\\_NOTHROW](#), and [L4\\_PROTO\\_PAGE\\_FAULT](#).

Here is the call graph for this function:



#### 14.1.9.9.4.7 l4\_msgtag\_is\_sigma0()

```
unsigned l4_msgtag_is_sigma0 (  
    l4_msgtag_t t) [inline]
```

Test for sigma0 protocol.

##### Parameters

<i>t</i>	The tag
----------	---------

**Returns**

Boolean value

Definition at line 451 of file [types.h](#).

References [L4\\_INLINE](#), [l4\\_msgtag\\_label\(\)](#), [L4\\_NOTHROW](#), and [L4\\_PROTO\\_SIGMA0](#).

Here is the call graph for this function:

**14.1.9.9.4.8 l4\_msgtag\_items()**

```
unsigned l4_msgtag_items (  
    l4_msgtag_t t) [inline]
```

Get the number of typed items.

**Parameters**

<i>t</i>	The tag
----------	---------

**Returns**

Number of items.

Definition at line 431 of file [types.h](#).

References [L4\\_NOTHROW](#).

Referenced by [l4util\\_ioport\\_map\(\)](#).

Here is the caller graph for this function:



#### 14.1.9.9.4.9 l4\_msgtag\_label()

```
long l4_msgtag_label (  
    l4_msgtag_t t) [inline]
```

Get the protocol of tag.

#### Parameters

---

<i>t</i>	The tag
----------	---------

**Returns**

Label

**Examples**

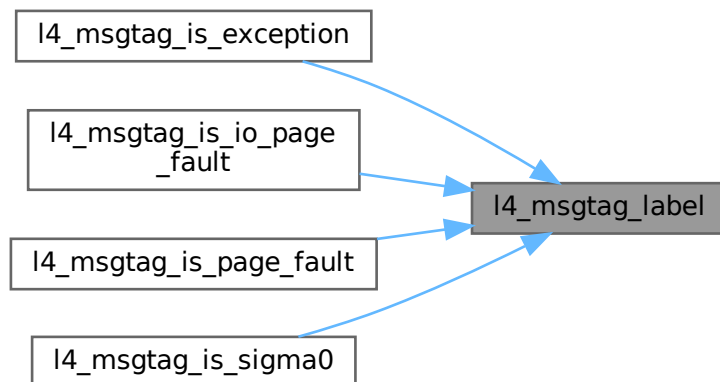
[examples/sys/singlestep/main.c](#), and [examples/sys/start-with-exc/main.c](#).

Definition at line 417 of file [types.h](#).

References [L4\\_NOTHROW](#).

Referenced by [l4\\_msgtag\\_is\\_exception\(\)](#), [l4\\_msgtag\\_is\\_io\\_page\\_fault\(\)](#), [l4\\_msgtag\\_is\\_page\\_fault\(\)](#), and [l4\\_msgtag\\_is\\_sigma0\(\)](#).

Here is the caller graph for this function:

**14.1.9.9.4.10 l4\_msgtag\_words()**

```
unsigned l4_msgtag_words (
    l4_msgtag_t t) [inline]
```

Get the number of untyped words.

**Parameters**

<i>t</i>	The tag
----------	---------

**Returns**

Number of words

**Examples**

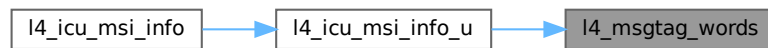
[examples/sys/utcb-ipc/main.c](#).

Definition at line 427 of file [types.h](#).

References [L4\\_NOTHROW](#).

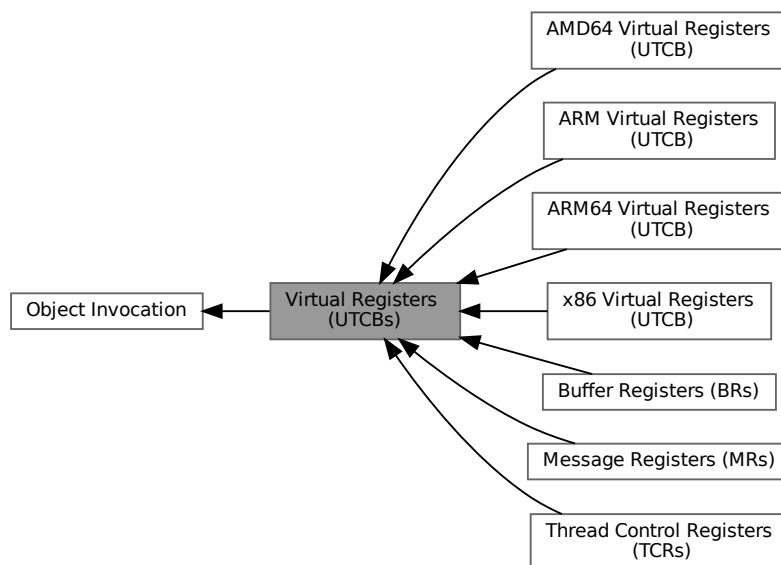
Referenced by [l4\\_icu\\_msi\\_info\\_u\(\)](#).

Here is the caller graph for this function:

**14.1.9.10 Virtual Registers (UTCBS)**

[L4 Virtual Registers \(UTCB\)](#).

Collaboration diagram for Virtual Registers (UTCBS):



## Topics

- [Message Registers \(MRs\)](#) . . . . . 275
- [Buffer Registers \(BRs\)](#) . . . . . 279
- [Thread Control Registers \(TCRs\)](#) . . . . . 280
- [ARM Virtual Registers \(UTCB\)](#) . . . . . 280
- [ARM64 Virtual Registers \(UTCB\)](#) . . . . . 281
- [AMD64 Virtual Registers \(UTCB\)](#) . . . . . 282
- [x86 Virtual Registers \(UTCB\)](#) . . . . . 282

## Files

- file [utcb.h](#)  
*UTCB definitions for ARM.*
- file [utcb.h](#)  
*UTCB definitions for ARM64.*
- file [utcb.h](#)  
*UTCB definitions for AMD64.*
- file [utcb.h](#)  
*UTCB definitions for x86.*

## Typedefs

- typedef struct [l4\\_utcb\\_t](#) [l4\\_utcb\\_t](#)  
*Opaque type for the UTCB.*

## Functions

- [l4\\_utcb\\_t](#) \* [l4\\_utcb](#) (void) [L4\\_NOTHROW](#) [L4\\_PURE](#)  
*Get the UTCB address.*
- [l4\\_msg\\_regs\\_t](#) \* [l4\\_utcb\\_mr](#) (void) [L4\\_NOTHROW](#) [L4\\_PURE](#)  
*Get the message-register block of a UTCB.*
- [l4\\_buf\\_regs\\_t](#) \* [l4\\_utcb\\_br](#) (void) [L4\\_NOTHROW](#) [L4\\_PURE](#)  
*Get the buffer-register block of a UTCB.*
- [l4\\_thread\\_regs\\_t](#) \* [l4\\_utcb\\_tcr](#) (void) [L4\\_NOTHROW](#) [L4\\_PURE](#)  
*Get the thread-control-register block of a UTCB.*

#### 14.1.9.10.1 Detailed Description

[L4](#) Virtual Registers (UTCB).

##### Include File

```
#include <l4/sys/utcb.h>
```

The virtual registers are part of the micro-kernel API and are located in the user-level thread control block (UTCB). The UTCB is a data structure defined by the micro kernel and located on kernel-provided memory. Each [L4](#) thread gets a unique UTCB assigned when it is bound to a task (see [Thread Control](#) , [l4\\_thread\\_control\\_bind\(\)](#) for more information).

The UTCB is arranged in three blocks of virtual registers.

- [Thread Control Registers \(TCRs\)](#)
- [Message Registers \(MRs\)](#)
- [Buffer Registers \(BRs\)](#)

To access the contents of the virtual registers the [l4\\_utcb\\_mr\(\)](#), [l4\\_utcb\\_tcr\(\)](#), and [l4\\_utcb\\_br\(\)](#) functions must be used.

#### 14.1.9.10.2 Typedef Documentation

##### 14.1.9.10.2.1 [l4\\_utcb\\_t](#)

```
typedef struct l4\_utcb\_t l4\_utcb\_t
```

Opaque type for the UTCB.

To access the contents of the virtual registers the [l4\\_utcb\\_mr\(\)](#), [l4\\_utcb\\_tcr\(\)](#), and [l4\\_utcb\\_br\(\)](#) functions must be used.

##### Examples

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line [56](#) of file [utcb.h](#).



### 14.1.9.10.3 Function Documentation

#### 14.1.9.10.3.1 l4\_utcb\_br()

```
l4_buf_regs_t * l4_utcb_br (  
    void )    [inline]
```

Get the buffer-register block of a UTCB.

##### Returns

A pointer to the buffer-register block of `u`.

Definition at line 361 of file `utcb.h`.

References [L4\\_INLINE](#), [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Referenced by [l4util\\_ioport\\_map\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.1.9.10.3.2 l4\_utcb\_mr()

```
l4_msg_regs_t * l4_utcb_mr (  
    void )    [inline]
```

Get the message-register block of a UTCB.

### Returns

A pointer to the message-register block of `u`.

### Examples

[examples/sys/aliens/main.c](#), [examples/sys/ipc/ipc\\_example.c](#), [examples/sys/singlestep/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 358 of file [utcb.h](#).

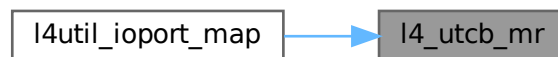
References [L4\\_INLINE](#), [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Referenced by [l4util\\_ioport\\_map\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.1.9.10.3.3 l4\_utcb\_tcr()

```
l4_thread_regs_t * l4_utcb_tcr (  
    void ) [inline]
```

Get the thread-control-register block of a UTCB.

**Returns**

A pointer to the thread-control-register block of `u`.

Definition at line 364 of file `utcb.h`.

References [L4\\_INLINE](#), [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**14.1.9.10.4 Message Registers (MRs)**

Collaboration diagram for Message Registers (MRs):

**Topics**

- [Exception registers](#) . . . . . 276  
*Overly definition of the MRs for exception messages.*

**Data Structures**

- union [l4\\_msg\\_regs\\_t](#)  
*Encapsulation of the message-register block in the UTCB.*

**Typedefs**

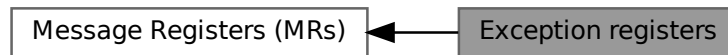
- typedef union `l4_msg_regs_t` **`l4_msg_regs_t`**  
*Encapsulation of the message-register block in the UTCB.*

#### 14.1.9.10.4.1 Detailed Description

#### 14.1.9.10.4.2 Exception registers

Overly definition of the MRs for exception messages.

Collaboration diagram for Exception registers:



### Functions

- `l4_exc_regs_t * l4_utcb_exc (void) L4_NOTHROW L4_PURE`  
Get the message-register block of a UTCB (for an exception IPC).
- `l4_umword_t l4_utcb_exc_pc (l4_exc_regs_t const *u) L4_NOTHROW L4_PURE`  
Access function to get the program counter of the exception state.
- `void l4_utcb_exc_pc_set (l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW`  
Set the program counter register in the exception state.
- `unsigned long l4_utcb_exc_typeval (l4_exc_regs_t const *u) L4_NOTHROW L4_PURE`  
Get the value out of an exception UTCB that describes the type of exception.
- `int l4_utcb_exc_is_pf (l4_exc_regs_t const *u) L4_NOTHROW L4_PURE`  
Check whether an exception IPC is a page fault.
- `l4_addr_t l4_utcb_exc_pfa (l4_exc_regs_t const *u) L4_NOTHROW L4_PURE`  
Function to get the L4 style page fault address out of an exception.
- `int l4_utcb_exc_is_ex_regs_exception (l4_exc_regs_t const *u) L4_NOTHROW L4_PURE`  
Check whether an exception IPC was triggered via `l4_thread_ex_regs()`.

### Detailed Description

Overly definition of the MRs for exception messages.

### Function Documentation

#### `l4_utcb_exc()`

```
l4_exc_regs_t * l4_utcb_exc (
    void ) [inline]
```

Get the message-register block of a UTCB (for an exception IPC).

**Returns**

A pointer to the exception message in `u`.

**Examples**

[examples/sys/aliens/main.c](#), and [examples/sys/singlestep/main.c](#).

Definition at line 367 of file [utcb.h](#).

References [L4\\_INLINE](#), [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**`l4_utcb_exc_is_ex_regs_exception()`**

```
int l4_utcb_exc_is_ex_regs_exception (
    l4_exc_regs_t const * u) [inline]
```

Check whether an exception IPC was triggered via [l4\\_thread\\_ex\\_regs\(\)](#).

**Return values**

<code>0</code>	Exception was not triggered through <code>ex_regs</code> .
<code>!=0</code>	Exception was triggered through <code>ex_regs</code> .

This function checks if the exception was emitted by using the `L4_THREAD_EX_REGS_TRIGGER_EXCEPTION` flag in an [l4\\_thread\\_ex\\_regs\(\)](#) call.

Definition at line 110 of file [utcb.h](#).

References [L4\\_INLINE](#), [L4\\_NOTHROW](#), and [l4\\_utcb\\_exc\\_typeval\(\)](#).

Here is the call graph for this function:



**l4\_utcb\_exc\_is\_pf()**

```
int l4_utcb_exc_is_pf (
    l4_exc_regs_t const * u) [inline]
```

Check whether an exception IPC is a page fault.

**Returns**

0 if not, != 0 if yes

Function to check whether an exception IPC is a page fault, also applies to I/O pagefaults.

Definition at line 100 of file [utcb.h](#).

References [L4\\_INLINE](#), and [L4\\_NOTHROW](#).

**l4\_utcb\_exc\_pc()**

```
l4_umword_t l4_utcb_exc_pc (
    l4_exc_regs_t const * u) [inline]
```

Access function to get the program counter of the exception state.

**Parameters**

<i>u</i>	UTCB
----------	------

**Returns**

The program counter register out of the exception state.

**Examples**

[examples/sys/aliens/main.c](#), and [examples/sys/singlestep/main.c](#).

Definition at line 85 of file [utcb.h](#).

References [L4\\_INLINE](#), and [L4\\_NOTHROW](#).

**l4\_utcb\_exc\_pc\_set()**

```
void l4_utcb_exc_pc_set (
    l4_exc_regs_t * u,
    l4_addr_t pc) [inline]
```

Set the program counter register in the exception state.

**Parameters**

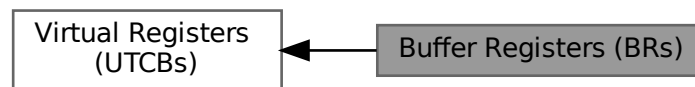
<i>u</i>	UTCB
<i>pc</i>	The program counter to set.

Definition at line 90 of file [utcb.h](#).

References [L4\\_INLINE](#), and [L4\\_NOTHROW](#).

#### 14.1.9.10.5 Buffer Registers (BRs)

Collaboration diagram for Buffer Registers (BRs):



#### Data Structures

- struct [l4\\_buf\\_regs\\_t](#)  
*Encapsulation of the buffer-registers block in the UTCB.*

#### Typedefs

- typedef struct l4\_buf\_regs\_t [l4\\_buf\\_regs\\_t](#)  
*Encapsulation of the buffer-registers block in the UTCB.*

#### Enumerations

- enum [l4\\_buffer\\_desc\\_consts\\_t](#) { [L4\\_BDR\\_MEM\\_SHIFT](#) = 0 , [L4\\_BDR\\_IO\\_SHIFT](#) = 5 , [L4\\_BDR\\_OBJ\\_SHIFT](#) = 10 , [L4\\_BDR\\_OFFSET\\_MASK](#) = (1UL << 20) - 1 }  
*Constants for buffer descriptors.*

#### Functions

- void [l4\\_utcb\\_inherit\\_fpu](#) (int switch\_on) [L4\\_NOTHROW](#)  
*Enable or disable inheritance of FPU state to receiver.*

##### 14.1.9.10.5.1 Detailed Description

##### 14.1.9.10.5.2 Enumeration Type Documentation

#### [l4\\_buffer\\_desc\\_consts\\_t](#)

enum [l4\\_buffer\\_desc\\_consts\\_t](#)

Constants for buffer descriptors.

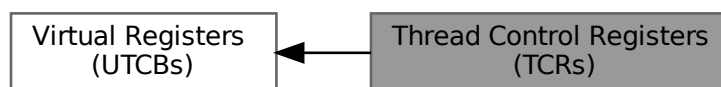
#### Enumerator

L4_BDR_MEM_SHIFT	Bit offset for the memory-buffer index.
L4_BDR_IO_SHIFT	Bit offset for the IO-buffer index.
L4_BDR_OBJ_SHIFT	Bit offset for the capability-buffer index.

Definition at line 303 of file [consts.h](#).

#### 14.1.9.10.6 Thread Control Registers (TCRs)

Collaboration diagram for Thread Control Registers (TCRs):



#### Data Structures

- struct [l4\\_thread\\_regs\\_t](#)  
*Encapsulation of the thread-control-register block of the UTCB.*

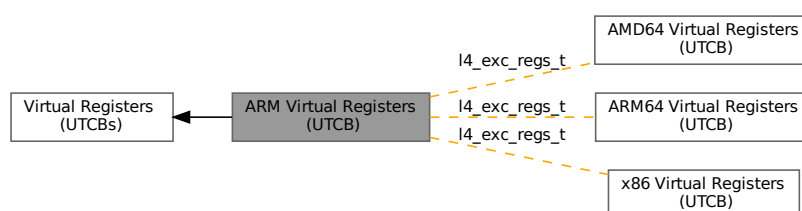
#### Typedefs

- typedef struct l4\_thread\_regs\_t **l4\_thread\_regs\_t**  
*Encapsulation of the thread-control-register block of the UTCB.*

#### 14.1.9.10.6.1 Detailed Description

#### 14.1.9.10.7 ARM Virtual Registers (UTCB)

Collaboration diagram for ARM Virtual Registers (UTCB):





## Data Structures

- struct [l4\\_exc\\_regs\\_t](#)  
*UTCB structure for exceptions.*

## Typedefs

- typedef struct l4\_exc\_regs\_t **l4\_exc\_regs\_t**  
*UTCB structure for exceptions.*

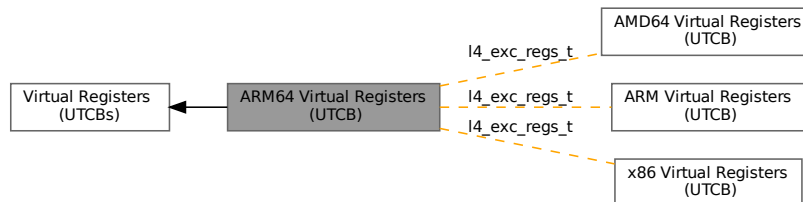
## Enumerations

- enum [L4\\_utcb\\_consts\\_arm](#)  
*UTCB constants for ARM.*

### 14.1.9.10.7.1 Detailed Description

### 14.1.9.10.8 ARM64 Virtual Registers (UTCB)

Collaboration diagram for ARM64 Virtual Registers (UTCB):



## Data Structures

- struct [l4\\_exc\\_regs\\_t](#)  
*UTCB structure for exceptions.*

## Typedefs

- typedef struct l4\_exc\_regs\_t **l4\_exc\_regs\_t**  
*UTCB structure for exceptions.*

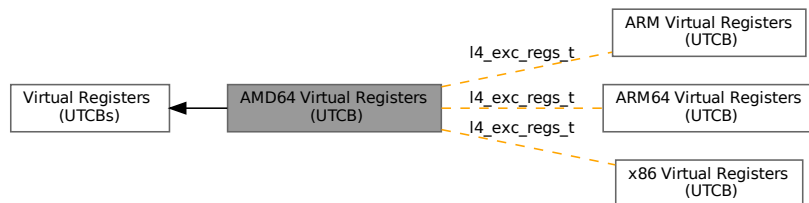
## Enumerations

- enum [L4\\_utcb\\_consts\\_arm64](#)  
*UTCB constants for ARM64.*

#### 14.1.9.10.8.1 Detailed Description

#### 14.1.9.10.9 AMD64 Virtual Registers (UTCB)

Collaboration diagram for AMD64 Virtual Registers (UTCB):



### Data Structures

- struct [l4\\_exc\\_regs\\_t](#)  
*UTCB structure for exceptions.*

### Typedefs

- typedef struct l4\_exc\_regs\_t **l4\_exc\_regs\_t**  
*UTCB structure for exceptions.*

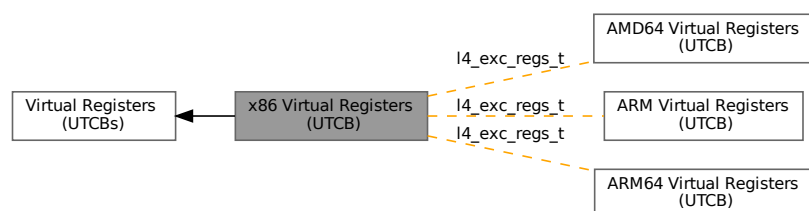
### Enumerations

- enum [L4\\_utcb\\_consts\\_amd64](#)  
*UTCB constants for AMD64.*

#### 14.1.9.10.9.1 Detailed Description

#### 14.1.9.10.10 x86 Virtual Registers (UTCB)

Collaboration diagram for x86 Virtual Registers (UTCB):



## Data Structures

- struct [l4\\_exc\\_regs\\_t](#)  
*UTCB structure for exceptions.*

## Typedefs

- typedef struct l4\_exc\_regs\_t [l4\\_exc\\_regs\\_t](#)  
*UTCB structure for exceptions.*

## Enumerations

- enum [L4\\_utcb\\_consts\\_x86](#) {  
[L4\\_UTCB\\_EXCEPTION\\_REGS\\_SIZE](#) = 19 , [L4\\_UTCB\\_GENERIC\\_DATA\\_SIZE](#) = 63 , [L4\\_UTCB\\_GENERIC\\_BUFFERS\\_SIZE](#) = 58 , [L4\\_UTCB\\_MSG\\_REGS\\_OFFSET](#) = 0 ,  
[L4\\_UTCB\\_BUF\\_REGS\\_OFFSET](#) = 64 \* sizeof(l4\_umword\_t) , [L4\\_UTCB\\_THREAD\\_REGS\\_OFFSET](#) = 123 \* sizeof(l4\_umword\_t) , [L4\\_UTCB\\_INHERIT\\_FPU](#) = 1UL << 24 , [L4\\_UTCB\\_OFFSET](#) = 512 }  
*UTCB constants for x86.*

### 14.1.9.10.10.1 Detailed Description

### 14.1.9.10.10.2 Enumeration Type Documentation

#### L4\_utcb\_consts\_x86

enum [L4\\_utcb\\_consts\\_x86](#)

UTCB constants for x86.

#### Enumerator

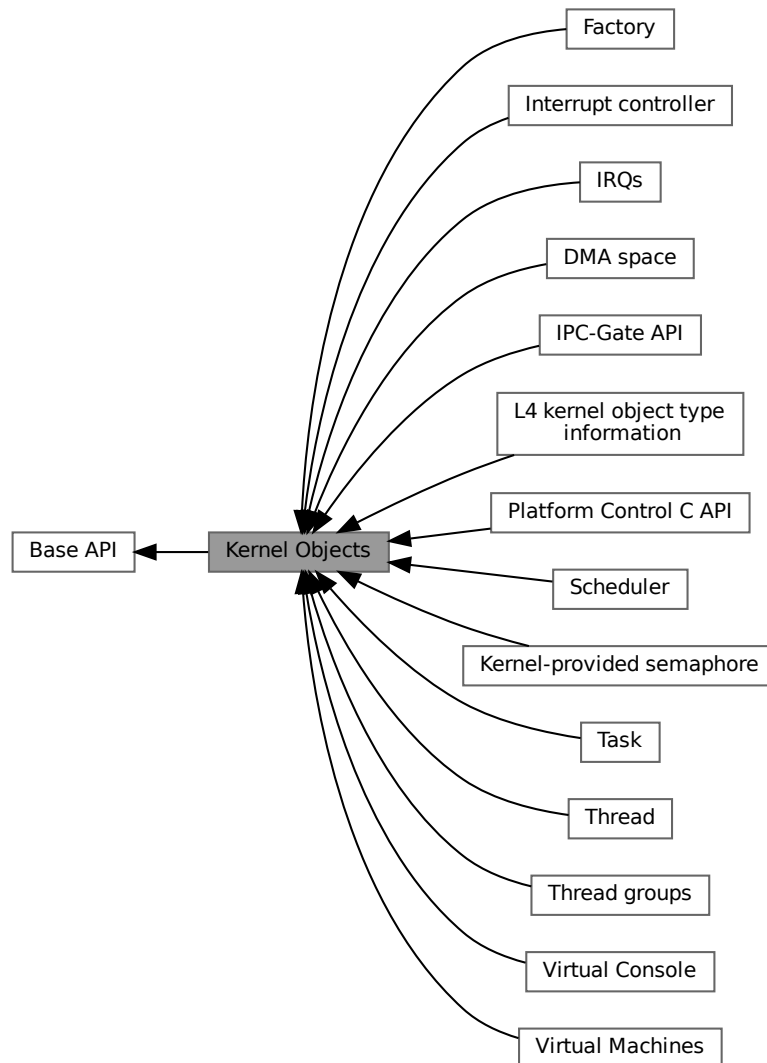
<a href="#">L4_UTCB_EXCEPTION_REGS_SIZE</a>	Number if message registers used for exception IPC.
<a href="#">L4_UTCB_GENERIC_DATA_SIZE</a>	Total number of message register (MRs) available.
<a href="#">L4_UTCB_GENERIC_BUFFERS_SIZE</a>	Total number of buffer registers (BRs) available.
<a href="#">L4_UTCB_MSG_REGS_OFFSET</a>	Offset of MR[0] relative to the UTCB pointer.
<a href="#">L4_UTCB_BUF_REGS_OFFSET</a>	Offset of BR[0] relative to the UTCB pointer.
<a href="#">L4_UTCB_THREAD_REGS_OFFSET</a>	Offset of TCR[0] relative to the UTCB pointer.
<a href="#">L4_UTCB_INHERIT_FPU</a>	BDR flag to accept reception of FPU state.
<a href="#">L4_UTCB_OFFSET</a>	Offset of two consecutive UTCBs.

Definition at line [30](#) of file [utcb.h](#).

### 14.1.10 Kernel Objects

API of kernel objects.

Collaboration diagram for Kernel Objects:



### Topics

- IPC-Gate API . . . . . 286  
*The C IPC gate interface, see [L4::lpc\\_gate](#) for the C++ interface.*
- DMA space . . . . . 291  
*A DMA space represents a device memory address space managed by an IOMMU.*
-

L4 kernel object type information . . . . .	291
<i>Type information for <a href="#">L4</a> server objects that can be called via IPC.</i>	
•	
Factory• . . . . .	293
<i>C factory interface to create objects, see <a href="#">L4::Factory</a> for the C++ interface.</i>	
•	
Virtual Machines . . . . .	303
<i>Virtual Machine API.</i>	
•	
Interrupt controller . . . . .	325
<i>The C Icu interface, see <a href="#">L4::Icu</a> for the C++ interface.</i>	
•	
IRQs • . . . . .	342
<i>C IRQ interface, see <a href="#">L4::Irq</a> for the C++ interface.</i>	
•	
Platform Control C API . . . . .	357
<i>C interface for controlling platform-wide properties, see <a href="#">L4::Platform_control</a> for the C++ interface.</i>	
•	
Scheduler . . . . .	362
<i>C interface of the Scheduler kernel object, see <a href="#">L4::Scheduler</a> for the C++ interface.</i>	
•	
Kernel-provided semaphore . . . . .	370
<i>C semaphore interface, see <a href="#">L4::Semaphore</a> for the C++ interface.</i>	
•	
Task • . . . . .	373
<i>C interface of the Task kernel object, see <a href="#">L4::Task</a> for the C++ interface.</i>	
•	
Thread• . . . . .	386
<i>C Thread object interface, see <a href="#">L4::Thread</a> for the C++ interface.</i>	
•	
Thread groups . . . . .	420
<i>C thread group interface, see <a href="#">L4::Thread_group</a> for the C++ interface.</i>	
•	
Virtual Console . . . . .	422
<i>C Virtual console interface for simple character based input and output, see <a href="#">L4::Vcon</a> for the C++ interface.</i>	

## Data Structures

- class [L4::Kobject](#)  
*Base class for all kinds of kernel objects and remote objects, referenced by capabilities.*
- class [L4::Vm](#)  
*Virtual machine host address space.*

### 14.1.10.1 Detailed Description

API of kernel objects.

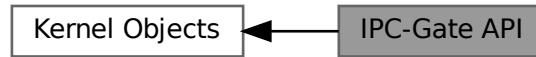
#### Include File

```
#include <l4/sys/kernel_object.h>
```

### 14.1.10.2 IPC-Gate API

The C IPC gate interface, see [L4::ipc\\_gate](#) for the C++ interface.

Collaboration diagram for IPC-Gate API:



### Functions

- [l4\\_msgtag\\_t l4\\_ipc\\_gate\\_get\\_infos](#) ([l4\\_cap\\_idx\\_t](#) gate, [l4\\_umword\\_t](#) \*label)  
*Get information about the IPC-gate.*
- [l4\\_msgtag\\_t l4\\_rcv\\_ep\\_bind\\_thread](#) ([l4\\_cap\\_idx\\_t](#) ep, [l4\\_cap\\_idx\\_t](#) thread, [l4\\_umword\\_t](#) label)  
*Bind the IPC receive endpoint to a thread.*
- [l4\\_msgtag\\_t l4\\_rcv\\_ep\\_bind\\_snd\\_destination](#) ([l4\\_cap\\_idx\\_t](#) ep, [l4\\_cap\\_idx\\_t](#) snd\_dst, [l4\\_umword\\_t](#) label)  
*Bind the IPC receive endpoint to a send destination (a thread).*

#### 14.1.10.2.1 Detailed Description

The C IPC gate interface, see [L4::ipc\\_gate](#) for the C++ interface.

IPC gates are used to create secure communication channels between protection domains. An IPC gate can be created using the [Factory](#) interface.

Depending on the permissions of the capability used, an IPC gate forwards IPC to the [Thread](#) the IPC gate is *bound* to (cf. [l4\\_rcv\\_ep\\_bind\\_thread\(\)](#) and [l4\\_rcv\\_ep\\_bind\\_snd\\_destination\(\)](#)). If the capability has the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission, only IPC using a protocol different from the [L4\\_PROTO\\_KOBJECT](#) protocol is forwarded. Without the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission, all IPC is forwarded. The latter is the usual case for a client in a client/server scenario. When not bound to a thread or thread group yet, the forwarded IPC blocks until the IPC gate is bound to a thread or thread group, or the IPC times out.

Forwarded IPC is always forwarded to the userland of the thread the IPC gate is bound to, either directly or indirectly using a thread group. That means, the [Thread](#) interface of that thread is not accessible via an IPC gate. The [IPC-Gate API](#) of an IPC gate is only accessible if the capability used has the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission (cf. previous paragraph). Conversely that means, if the capability used lacks the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission, [IPC-Gate API](#) calls are forwarded to the thread or thread group the IPC gate is bound to instead of being processed by the IPC gate itself. In a client/server scenario, a client should only get IPC gate capabilities without [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission so the client cannot tamper with the IPC gate.

When binding an IPC gate to a thread or thread group, a user-defined, kernel protected, machine-word sized payload called the IPC gate's *label* is assigned to the IPC gate (note that the two least significant bits of the label must be zero; cf. [l4\\_rcv\\_ep\\_bind\\_thread\(\)](#) and [l4\\_rcv\\_ep\\_bind\\_snd\\_destination\(\)](#)). When a send-only IPC or call IPC is forwarded via an IPC gate, the label provided by the sender is ignored and replaced by the IPC gate's label where the two least significant bits are set to the [L4\\_CAP\\_FPAGE\\_S](#) and [L4\\_CAP\\_FPAGE\\_W](#) permissions of the capability used. The replaced label is only visible to the thread the IPC gate is bound to upon receive (or to the

selected thread from the thread group the IPC gate is bound to). However, the configured label of an IPC gate can also be queried via [l4\\_ipc\\_gate\\_get\\_infos\(\)](#) if the capability used has the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission.

When deleting an IPC gate or when unbinding it from a thread or thread group, the label of IPC already in flight won't be changed. To ensure that no IPC from this IPC gate is received by a thread with an unexpected label, [l4\\_thread\\_modify\\_sender\\_start\(\)](#) shall be used to change the labels of every pending IPC to that gate. This is also required if the label of an already bound IPC gate is changed. It is not necessary after binding the IPC gate to a thread or thread group for the first time.

When binding a currently bound IPC gate to a new thread or thread group, the same label should be used that was used with the old thread. Otherwise the old and the new thread need to synchronize to avoid IPC messages with unexpected labels.

#### Include File

```
#include <l4/sys/ipc_gate.h>
```

For the C++ interface refer to the [L4::ipc\\_gate](#) documentation.

#### See also

[Object Invocation](#)

### 14.1.10.2.2 Function Documentation

#### 14.1.10.2.2.1 l4\_ipc\_gate\_get\_infos()

```
l4_msgtag_t l4_ipc_gate_get_infos (
    l4_cap_idx_t gate,
    l4_umword_t * label) [inline]
```

Get information about the IPC-gate.

#### Parameters

	<i>gate</i>	The IPC gate object to get information about.
out	<i>label</i>	The label of the IPC gate is returned here.

#### Returns

System call return tag.

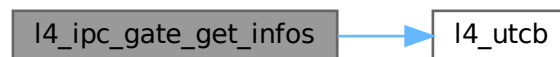
**Precondition**

If `gate` does not possess the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) right, the kernel will not perform this operation. Instead, the underlying IPC message will be forwarded to the thread or thread group the IPC gate is bound to, blocking the caller if the IPC gate is not bound yet.

Definition at line 166 of file [ipc\\_gate.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**14.1.10.2.2.2 l4\_rcv\_ep\_bind\_snd\_destination()**

```

l4_msgtag_t l4_rcv_ep_bind_snd_destination (
    l4_cap_idx_t ep,
    l4_cap_idx_t snd_dst,
    l4_umword_t label) [inline]
  
```

Bind the IPC receive endpoint to a send destination (a thread).

**Parameters**

<i>ep</i>	The IPC receive endpoint object.
<i>snd_dst</i>	The send destination (thread) object <i>ep</i> shall be bound to.
<i>label</i>	Label to assign to <i>ep</i> . For IPC gates, the two least significant bits must be set to zero.

**Returns**

Syscall return tag containing one of the following return codes.

**Return values**

<i>L4_EOK</i>	Operation successful.
<i>-L4_EINVAL</i>	<i>snd_dst</i> is not a thread object or other arguments were malformed.
<i>-L4_EPERM</i>	No <a href="#">L4_CAP_FPAGE_S</a> right on <i>ep</i> or <i>snd_dst</i> .



**Precondition**

If `ep` is an IPC gate capability without the `L4_FPAGE_C_IPCGATE_SVR` right, the kernel will not perform this operation. Instead, the underlying IPC message will be forwarded to the send destination the IPC gate is bound to, blocking the caller if the IPC gate was not bound yet.

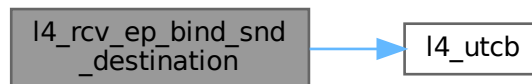
The specified `label` is passed to the receiver of the incoming IPC. It is possible to re-bind a receive endpoint to the same or a different send destination. In this case, IPC already in flight will be delivered with the old label to the previously bound thread unless `l4_thread_modify_sender_start()` is used to change these labels.

Definition at line 138 of file `rcv_endpoint.h`.

References `l4_utcb()`.

Referenced by `L4::Rcv_endpoint::bind_snd_destination()`.

Here is the call graph for this function:



Here is the caller graph for this function:

**14.1.10.2.2.3 l4\_rcv\_ep\_bind\_thread()**

```

l4_msgtag_t l4_rcv_ep_bind_thread (
    l4_cap_idx_t ep,
    l4_cap_idx_t thread,
    l4_umword_t label) [inline]
  
```

Bind the IPC receive endpoint to a thread.

**Parameters**

<i>ep</i>	The IPC receive endpoint object.
<i>thread</i>	The thread object <i>ep</i> shall be bound to.
<i>label</i>	Label to assign to <i>ep</i> . For IPC gates, the two least significant bits must be set to zero.

### Returns

Syscall return tag containing one of the following return codes.

### Return values

<i>L4_EOK</i>	Operation successful.
<i>-L4_EINVAL</i>	<i>thread</i> is not a thread object or other arguments were malformed.
<i>-L4_EPERM</i>	Insufficient permissions; see precondition.

### Precondition

The capabilities *ep* and *thread* both must have the permission [L4\\_CAP\\_FPAGE\\_S](#).

If *ep* is an IPC gate capability without the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) right, the kernel will not perform this operation. Instead, the underlying IPC message will be forwarded to the thread the IPC gate is bound to, blocking the caller if the IPC gate was not bound yet.

The specified *label* is passed to the receiver of the incoming IPC. It is possible to re-bind a receive endpoint to the same or a different thread. In this case, IPC already in flight will be delivered with the old label to the previously bound thread unless [l4\\_thread\\_modify\\_sender\\_start\(\)](#) is used to change these labels.

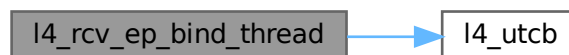
### Examples

[examples/sys/isr/main.c](#).

Definition at line [131](#) of file [rcv\\_endpoint.h](#).

References [l4\\_utcb\(\)](#).

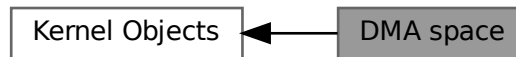
Here is the call graph for this function:



### 14.1.10.3 DMA space

A DMA space represents a device memory address space managed by an IOMMU.

Collaboration diagram for DMA space:



A DMA space represents a device memory address space managed by an IOMMU.

That is, it manages the translation of virtual addresses used by devices to physical addresses. It is accessed via the [L4::Task](#) interface, but with the following caveats:

- No threads can be bound to it.
- No objects (and IO ports on IA32) can be mapped to it.
- No kernel-user memory can be added to it.
- It must be constructed by passing the `L4_PROTO_DMA_SPACE` protocol constant to the kernel factory's [L4::Factory.create\(\)](#) call.

A DMA space must be bound to an [L4::iommu](#) to enable the address translation for specific devices.

The kernel factory allows to create DMA spaces only if the kernel has been configured with IOMMU support and if an IOMMU was detected.

### 14.1.10.4 L4 kernel object type information

Type information for [L4](#) server objects that can be called via IPC.

Collaboration diagram for L4 kernel object type information:



## Data Structures

- struct [L4::Type\\_info](#)  
*Dynamic Type Information for [L4Re](#) Interfaces.*
- struct [L4::Kobject\\_typeid< T >](#)  
*Meta object for handling access to type information of Kobjects.*
- class [L4::Kobject\\_t< Derived, Base, PROTO, S\\_DEMAND >](#)  
*Helper class to create an [L4Re](#) interface class that is derived from a single base class.*
- class [L4::Kobject\\_2t< Derived, Base1, Base2, PROTO, S\\_DEMAND >](#)  
*Helper class to create an [L4Re](#) interface class that is derived from two base classes (see [L4::Kobject\\_t](#)).*
- struct [L4::Kobject\\_3t< Derived, Base1, Base2, Base3, PROTO, S\\_DEMAND >](#)  
*Helper class to create an [L4Re](#) interface class that is derived from three base classes (see [L4::Kobject\\_t](#)).*
- struct [L4::Kobject\\_x< Derived, ARGS >](#)  
*Generic [Kobject](#) inheritance template.*

## Functions

- `template<typename T>`  
`Type\_info const * L4::kobject\_typeid () noexcept`  
*Get the [L4::Type\\_info](#) for the [L4Re](#) interface given in T.*

### 14.1.10.4.1 Detailed Description

Type information for [L4](#) server objects that can be called via IPC.

This type information consists of inheritance information, the protocol number assigned to an interface as well as the demand on server-side resources.

### 14.1.10.4.2 Function Documentation

#### 14.1.10.4.2.1 kobject\_typeid()

```
template<typename T>
Type\_info const * L4::kobject\_typeid () [inline], [noexcept]
```

Get the [L4::Type\\_info](#) for the [L4Re](#) interface given in T.

## Template Parameters

<i>T</i>	The type ( <a href="#">L4Re</a> interface) for which the information shall be returned.
----------	-----------------------------------------------------------------------------------------

**Returns**

A pointer to the [L4::Type\\_info](#) structure for T.

Definition at line 682 of file [\\_\\_typeinfo.h](#).

References [L4::Kobject\\_typeid< T >::id\(\)](#).

Referenced by [cap\\_dynamic\\_cast\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**14.1.10.5 Factory**

C factory interface to create objects, see [L4::Factory](#) for the C++ interface.

Collaboration diagram for Factory:



## Functions

- [l4\\_msgtag\\_t l4\\_factory\\_create\\_task](#) ([l4\\_cap\\_idx\\_t](#) factory, [l4\\_cap\\_idx\\_t](#) target\_cap, [l4\\_fpage\\_t](#) \*utcb\_area) [L4\\_NOTHROW](#)  
*Create a new task.*
- [l4\\_msgtag\\_t l4\\_factory\\_create\\_thread](#) ([l4\\_cap\\_idx\\_t](#) factory, [l4\\_cap\\_idx\\_t](#) target\_cap) [L4\\_NOTHROW](#)  
*Create a new thread.*
- [l4\\_msgtag\\_t l4\\_factory\\_create\\_factory](#) ([l4\\_cap\\_idx\\_t](#) factory, [l4\\_cap\\_idx\\_t](#) target\_cap, unsigned long limit) [L4\\_NOTHROW](#)  
*Create a new factory.*
- [l4\\_msgtag\\_t l4\\_factory\\_create\\_gate](#) ([l4\\_cap\\_idx\\_t](#) factory, [l4\\_cap\\_idx\\_t](#) target\_cap, [l4\\_cap\\_idx\\_t](#) snd\_dst\_↔ cap, [l4\\_umword\\_t](#) label) [L4\\_NOTHROW](#)  
*Create a new IPC gate, optionally bound to a send destination (a thread or thread group).*
- [l4\\_msgtag\\_t l4\\_factory\\_create\\_irq](#) ([l4\\_cap\\_idx\\_t](#) factory, [l4\\_cap\\_idx\\_t](#) target\_cap) [L4\\_NOTHROW](#)  
*Create a new IRQ sender.*
- [l4\\_msgtag\\_t l4\\_factory\\_create\\_vm](#) ([l4\\_cap\\_idx\\_t](#) factory, [l4\\_cap\\_idx\\_t](#) target\_cap) [L4\\_NOTHROW](#)  
*Create a new virtual machine.*
- [l4\\_msgtag\\_t l4\\_factory\\_create\\_vcpu\\_context](#) ([l4\\_cap\\_idx\\_t](#) factory, [l4\\_cap\\_idx\\_t](#) target\_cap) [L4\\_NOTHROW](#)  
*Create a new vCPU context.*
- [l4\\_msgtag\\_t l4\\_factory\\_create\\_thread\\_group](#) ([l4\\_cap\\_idx\\_t](#) factory, [l4\\_cap\\_idx\\_t](#) target\_cap, unsigned policy) [L4\\_NOTHROW](#)  
*Create a new thread group.*
- [l4\\_msgtag\\_t l4\\_factory\\_create](#) ([l4\\_cap\\_idx\\_t](#) factory, long obj, [l4\\_cap\\_idx\\_t](#) target) [L4\\_NOTHROW](#)  
*Create a new object.*

### 14.1.10.5.1 Detailed Description

C factory interface to create objects, see [L4::Factory](#) for the C++ interface.

A factory is used to create all kinds of kernel objects:

- [Task](#)
- [Thread](#)
- [Factory](#)
- [IPC-Gate API](#)
- [IRQs](#)
- [Virtual Machines](#)

To create a new kernel object the caller has to specify the factory to use for creation. The caller has to allocate a capability slot where the kernel stores the new object's capability.

The factory is equipped with a limit that limits the amount of kernel memory available for that factory.

#### Note

The limit does not give any guarantee for the amount of available kernel memory.

#### Include File

```
#include <l4/sys/factory.h>
```

For the C++ interface refer to [L4::Factory](#).

### 14.1.10.5.2 Function Documentation

#### 14.1.10.5.2.1 l4\_factory\_create()

```
l4_msgtag_t l4_factory_create (
    l4_cap_idx_t factory,
    long obj,
    l4_cap_idx_t target) [inline]
```

Create a new object.

##### Parameters

	<i>factory</i>	Factory to use for creation.
	<i>obj</i>	Protocol ID to describe the type of the object to create.
out	<i>target</i>	The kernel stores the new objects's capability into this slot.

##### Return values

<i>L4_EOK</i>	No error occurred.
<i>-L4_EPERM</i>	Insufficient permissions; see precondition.
<i>&lt;0</i>	Error code.

##### Precondition

The capability *factory* must have the permission [L4\\_CAP\\_FPAGE\\_S](#).

Definition at line 707 of file [factory.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.5.2.2 l4\_factory\_create\_factory()

```
l4_msgtag_t l4_factory_create_factory (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    unsigned long limit) [inline]
```

Create a new factory.

##### Parameters

	<i>factory</i>	Capability selector for factory to use for creation.
out	<i>target_cap</i>	The kernel stores the new factory's capability into this slot.
	<i>limit</i>	Limit for the new factory in bytes.

### Returns

Syscall return tag

### Return values

<i>L4_EOK</i>	No error occurred.
<i>-L4_EPERM</i>	Insufficient permissions; see precondition.
<i>&lt;0</i>	Error code.

### Precondition

The capability `factory` must have the permission [L4\\_CAP\\_FPAGE\\_S](#).

### Note

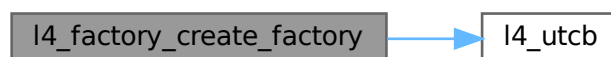
The limit of the new factory is subtracted from the available amount of the factory used for creation.

This method is only guaranteed to work with the [Kernel Factory](#). For other services, use the generic [L4::Factory::create\(\)](#) method and consult the service documentation for information on the arguments that need to be passed to the create stream.

Definition at line 545 of file [factory.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.5.2.3 l4\_factory\_create\_gate()

```

l4_msgtag_t l4_factory_create_gate (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    l4_cap_idx_t snd_dst_cap,
    l4_umword_t label) [inline]
  
```

Create a new IPC gate, optionally bound to a send destination (a thread or thread group).

### Parameters



	<i>factory</i>	Capability selector for factory to use for creation.
out	<i>target_cap</i>	The kernel stores the new IPC gate's capability into this slot.
	<i>snd_dst_cap</i>	Optional capability selector of a thread or thread group to bind the gate to. Use <a href="#">L4_INVALID_CAP</a> to create an unbound IPC gate.
	<i>label</i>	Optional label of the gate (precisely used if <i>snd_dst_cap</i> is valid). If <i>snd_dst_cap</i> is valid, <i>label</i> must be present.

#### Returns

Syscall return tag containing one of the following return codes.

#### Return values

<i>L4_EOK</i>	No error occurred.
<i>-L4_ENOMEM</i>	Out-of-memory during allocation of the <i>lpc_gate</i> object.
<i>-L4_EINVAL</i>	<i>snd_dst_cap</i> is void or points to something that is not a thread or thread group.
<i>-L4_EPERM</i>	Insufficient permissions; see precondition.

#### Precondition

The capability *factory* must have the permission [L4\\_CAP\\_FPAGE\\_S](#). Also *snd\_dst\_cap* (if not [L4\\_INVALID\\_CAP](#)) must have the permission [L4\\_CAP\\_FPAGE\\_S](#).

An unbound IPC gate can be bound to a thread using [l4\\_rcv\\_ep\\_bind\\_thread\(\)](#).

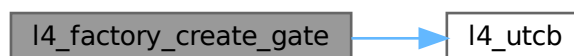
#### See also

[IPC-Gate API](#)

Definition at line 553 of file [factory.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



##### 14.1.10.5.2.4 l4\_factory\_create\_irq()

```

l4_msgtag_t l4_factory_create_irq (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap) [inline]
  
```

Create a new IRQ sender.

#### Parameters

	<i>factory</i>	Factory to use for creation.
out	<i>target_cap</i>	The kernel stores the new IRQ's capability into this slot.

### Return values

<i>L4_EOK</i>	No error occurred.
<i>-L4_EPERM</i>	Insufficient permissions; see precondition.
<i>&lt;0</i>	Error code.

### Precondition

The capability *factory* must have the permission [L4\\_CAP\\_FPAGE\\_S](#).

### See also

[IRQs](#)

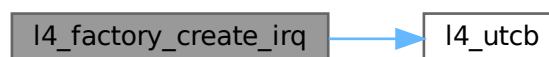
### Examples

[examples/sys/isr/main.c](#).

Definition at line 561 of file [factory.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.5.2.5 l4\_factory\_create\_task()

```

l4_msgtag_t l4_factory_create_task (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    l4_fpage_t * utcb_area) [inline]
  
```

Create a new task.

### Parameters

	<i>factory</i>	Capability selector for factory to use for creation.
out	<i>target_cap</i>	The kernel stores the new task's capability into this slot.
in, out	<i>utcb_area</i>	Pointer to flexpage that describes an area of kernel-user memory that can be used for UTCBs and vCPU state-save-areas of the new task.

On systems without MMU, the flexpage is adjusted to reflect the actually allocated physical address.

#### Returns

Syscall return tag.

#### Return values

<i>L4_EOK</i>	No error occurred.
<i>-L4_EPERM</i>	Insufficient permissions; see precondition.
<i>&lt;0</i>	Error code.

#### Precondition

The capability *factory* must have the permission [L4\\_CAP\\_FPAGE\\_S](#).

#### Note

The size of the UTCB area specifies indirectly the number of UTCBs available for this task. Refer to [l4\\_task\\_add\\_ku\\_mem\(\)](#) for adding more of this type of memory.

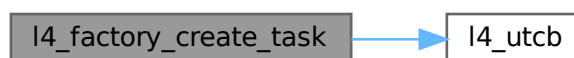
#### See also

[Task](#)

Definition at line 531 of file [factory.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.5.2.6 l4\_factory\_create\_thread()

```

l4_msgtag_t l4_factory_create_thread (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap) [inline]
  
```

Create a new thread.

#### Parameters

	<i>factory</i>	Capability selector for factory to use for creation.
out	<i>target_cap</i>	The kernel stores the new thread's capability into this slot.

### Returns

Syscall return tag

### Return values

<i>L4_EOK</i>	No error occurred.
<i>-L4_EPERM</i>	Insufficient permissions; see precondition.
<i>&lt;0</i>	Error code.

### Precondition

The capability *factory* must have the permission [L4\\_CAP\\_FPAGE\\_S](#).

### See also

[Thread](#)

### Examples

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 538 of file [factory.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.5.2.7 l4\_factory\_create\_thread\_group()

```

l4_msgtag_t l4_factory_create_thread_group (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    unsigned policy) [inline]
  
```

Create a new thread group.

An IPC endpoint can be bound to a thread group. When a message arrives at the IPC endpoint, a specific thread of the thread group is selected to actually receive the message. A thread group is a send destination for an IPC endpoint.

### Parameters

	<i>factory</i>	Capability selector for factory to use for creation.
out	<i>target_cap</i>	The kernel stores the new thread group's capability into this slot.
	<i>policy</i>	Policy parameter for the thread group. See #L4_thread_group_policy for a list of supported values.

**Returns**

Syscall return tag

**Return values**

<i>L4_EOK</i>	No error occurred.
<i>-L4_ENOMEM</i>	Out-of-memory during allocation of the thread group object.
<i>-L4_EINVAL</i>	Invalid policy parameter.
<i>-L4_EPERM</i>	The factory instance requires <a href="#">L4_CAP_FPAGE_S</a> rights on <code>factory</code> and <a href="#">L4_CAP_FPAGE_S</a> is not present.
<i>&lt;0</i>	Error code.

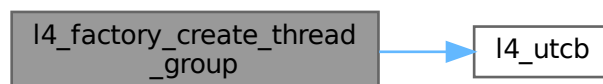
**See also**

[Thread groups](#)

Definition at line 582 of file [factory.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**14.1.10.5.2.8 l4\_factory\_create\_vcpu\_context()**

```

l4_msgtag_t l4_factory_create_vcpu_context (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap) [inline]
  
```

Create a new vCPU context.

A vCPU context typically represents a hardware structure that captures the state of a vCPU on a CPU (e.g. VMX VMCS).

**Parameters**

	<i>factory</i>	Capability selector for factory to use for creation.
out	<i>target_cap</i>	The kernel stores the new vCPU context's capability into this slot.

#### Returns

Syscall return tag

#### Return values

<i>L4_EOK</i>	No error occurred.
<i>-L4_EPERM</i>	Insufficient permissions; see precondition.
<i>&lt;0</i>	Error code.

#### Precondition

The capability *factory* must have the permission [L4\\_CAP\\_FPAGE\\_S](#).

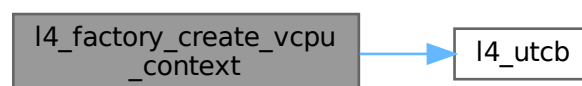
#### See also

[Virtual Machines](#)

Definition at line [575](#) of file [factory.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.5.2.9 l4\_factory\_create\_vm()

```

l4_msgtag_t l4_factory_create_vm (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap) [inline]
  
```

Create a new virtual machine.

#### Parameters

	<i>factory</i>	Capability selector for factory to use for creation.
out	<i>target_cap</i>	The kernel stores the new VM's capability into this slot.

**Returns**

Syscall return tag

**Return values**

<i>L4_EOK</i>	No error occurred.
<i>-L4_EPERM</i>	Insufficient permissions; see precondition.
<i>&lt;0</i>	Error code.

**Precondition**

The capability `factory` must have the permission [L4\\_CAP\\_FPAGE\\_S](#).

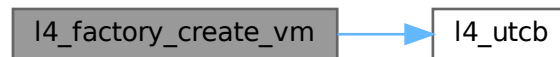
**See also**

[Virtual Machines](#)

Definition at line 568 of file [factory.h](#).

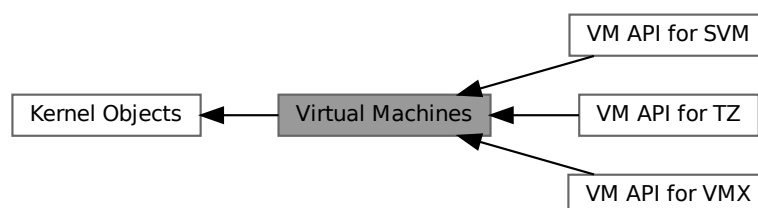
References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**14.1.10.6 Virtual Machines**

Virtual Machine API.

Collaboration diagram for Virtual Machines:



## Topics

- VM API for SVM . . . . . 304  
*Virtual machine API for SVM.*
- VM API for VMX . . . . . 305  
*Virtual machine API for VMX.*
- VM API for TZ . . . . . 325  
*Virtual Machine API for ARM TrustZone.*

### 14.1.10.6.1 Detailed Description

Virtual Machine API.

### 14.1.10.6.2 VM API for SVM

Virtual machine API for SVM.

Collaboration diagram for VM API for SVM:



## Data Structures

- struct [l4\\_vm\\_svm\\_vmcb\\_control\\_area](#)  
*VMCB structure for SVM VMs.*
- struct [l4\\_vm\\_svm\\_vmcb\\_state\\_save\\_area\\_seg](#)  
*State save area segment selector struct.*
- struct [l4\\_vm\\_svm\\_vmcb\\_state\\_save\\_area](#)  
*State save area structure for SVM VMs.*
- struct [l4\\_vm\\_svm\\_vmcb\\_t](#)  
*Control structure for SVM VMs.*

## Typedefs

- typedef struct [l4\\_vm\\_svm\\_vmcb\\_control\\_area](#) [l4\\_vm\\_svm\\_vmcb\\_control\\_area\\_t](#)  
*VMCB structure for SVM VMs.*
- typedef struct [l4\\_vm\\_svm\\_vmcb\\_state\\_save\\_area\\_seg](#) [l4\\_vm\\_svm\\_vmcb\\_state\\_save\\_area\\_seg\\_t](#)  
*State save area segment selector struct.*
- typedef struct [l4\\_vm\\_svm\\_vmcb\\_state\\_save\\_area](#) [l4\\_vm\\_svm\\_vmcb\\_state\\_save\\_area\\_t](#)  
*State save area structure for SVM VMs.*
- typedef struct [l4\\_vm\\_svm\\_vmcb\\_t](#) [l4\\_vm\\_svm\\_vmcb\\_t](#)  
*Control structure for SVM VMs.*



#### 14.1.10.6.2.1 Detailed Description

Virtual machine API for SVM.

#### 14.1.10.6.3 VM API for VMX

Virtual machine API for VMX.

Collaboration diagram for VM API for VMX:



#### Data Structures

- struct [l4\\_vmx\\_offset\\_table\\_t](#)  
*Software VMCS field offset table.*
- struct [l4\\_vm\\_vmx\\_vcpu\\_vmcs\\_t](#)  
*VMX software VMCS.*
- struct [l4\\_vm\\_vmx\\_vcpu\\_infos\\_t](#)  
*VMX information members.*
- struct [l4\\_vm\\_vmx\\_vcpu\\_state\\_t](#)  
*VMX vCPU state.*

#### Typedefs

- typedef struct [l4\\_vmx\\_offset\\_table\\_t](#) [l4\\_vmx\\_offset\\_table\\_t](#)  
*Software VMCS field offset table.*
- typedef struct [l4\\_vm\\_vmx\\_vcpu\\_vmcs\\_t](#) [l4\\_vm\\_vmx\\_vcpu\\_vmcs\\_t](#)  
*VMX software VMCS.*
- typedef struct [l4\\_vm\\_vmx\\_vcpu\\_infos\\_t](#) [l4\\_vm\\_vmx\\_vcpu\\_infos\\_t](#)  
*VMX information members.*
- typedef struct [l4\\_vm\\_vmx\\_vcpu\\_state\\_t](#) [l4\\_vm\\_vmx\\_vcpu\\_state\\_t](#)  
*VMX vCPU state.*

## Enumerations

- enum `L4_vm_vmx_caps_regs` {  
`L4_VM_VMX_BASIC_REG` = 0 , `L4_VM_VMX_TRUE_PINBASED_CTLS_REG` = 1 , `L4_VM_VMX_TRUE_PROCBASED_CTL`  
= 2 , `L4_VM_VMX_TRUE_EXIT_CTLS_REG` = 3 ,  
`L4_VM_VMX_TRUE_ENTRY_CTLS_REG` = 4 , `L4_VM_VMX_MISC_REG` = 5 , `L4_VM_VMX_CR0_FIXED0_REG`  
= 6 , `L4_VM_VMX_CR0_FIXED1_REG` = 7 ,  
`L4_VM_VMX_CR4_FIXED0_REG` = 8 , `L4_VM_VMX_CR4_FIXED1_REG` = 9 , `L4_VM_VMX_VMCS_ENUM_REG`  
= 10 , `L4_VM_VMX_PROCBASED_CTLS2_REG` = 11 ,  
`L4_VM_VMX_EPT_VPID_CAP_REG` = 12 , `L4_VM_VMX_NESTED_REVISION` = 13 , `L4_VM_VMX_NUM_CAPS_REGS`  
}

*Exported VMX capability registers.*

- enum `L4_vm_vmx_dfl1_regs` {  
`L4_VM_VMX_PINBASED_CTLS_DFL1_REG` = 0 , `L4_VM_VMX_PROCBASED_CTLS_DFL1_REG` = 1 ,  
`L4_VM_VMX_EXIT_CTLS_DFL1_REG` = 2 , `L4_VM_VMX_ENTRY_CTLS_DFL1_REG` = 3 ,  
`L4_VM_VMX_NUM_DFL1_REGS` }

*Exported VMX capability registers (default to 1 bits).*

- enum `L4_vm_vmx_sw_fields` {  
`L4_VM_VMX_VMCS_CR2` = 0x6880 , `L4_VM_VMX_VMCS_NAT_ARG0` = 0x6882 , `L4_VM_VMX_VMCS_NAT_ARG1`  
= 0x6884 , `L4_VM_VMX_VMCS_NAT_ARG2` = 0x6886 ,  
`L4_VM_VMX_VMCS_NAT_ARG3` = 0x6888 , `L4_VM_VMX_VMCS_XCR0` = 0x2880 , `L4_VM_VMX_VMCS_MSR_SYSCALL_M`  
= 0x2882 , `L4_VM_VMX_VMCS_MSR_LSTAR` = 0x2884 ,  
`L4_VM_VMX_VMCS_MSR_CSTAR` = 0x2886 , `L4_VM_VMX_VMCS_MSR_TSC_AUX` = 0x2888 ,  
`L4_VM_VMX_VMCS_MSR_STAR` = 0x288a , `L4_VM_VMX_VMCS_MSR_KERNEL_GS_BASE` = 0x288c }

*Additional (software-defined) VMCS fields.*

- enum `L4_vm_vmx_vmcs_sizes` { `L4_VM_VMX_VMCS_SIZE_VALUES` = 2560 , `L4_VM_VMX_VMCS_SIZE_DIRTY_BITMAP`  
= 320 }

*Sizes of software VMCS members.*

## Functions

- `l4_uint64_t l4_vm_vmx_get_caps` (`l4_vm_vmx_vcpu_state_t` const \*vcpu\_state, enum `L4_vm_vmx_caps_regs`  
caps\_reg) `L4_NOTHROW`  
*Get a capability register for VMX.*
- `l4_uint32_t l4_vm_vmx_get_caps_default1` (`l4_vm_vmx_vcpu_state_t` const \*vcpu\_state, enum `L4_vm_vmx_dfl1_regs`  
dfl1\_reg) `L4_NOTHROW`  
*Get a default to one capability register for VMX.*
- unsigned `l4_vm_vmx_field_len` (unsigned field) `L4_NOTHROW`  
*Return length in bytes of a VMCS field.*
- unsigned `l4_vm_vmx_field_order` (unsigned field) `L4_NOTHROW`  
*Return length in power of two (bytes) of a VMCS field.*
- void `l4_vm_vmx_clear` (`l4_vm_vmx_vcpu_vmcs_t` \*vmcs, `l4_vm_vmx_vcpu_vmcs_t` \*dest\_vmcs)  
`L4_NOTHROW`  
*Save the content from the software VMCS to a different software VMCS.*
- void `l4_vm_vmx_ptr_load` (`l4_vm_vmx_vcpu_vmcs_t` \*vmcs, `l4_vm_vmx_vcpu_vmcs_t` \*src\_vmcs)  
`L4_NOTHROW`  
*Load the content from a different software VMCS to the software VMCS.*
- `l4_uint32_t l4_vm_vmx_get_cr2_index` (`l4_vm_vmx_vcpu_vmcs_t` const \*vmcs) `L4_NOTHROW`  
*Get the software VMCS field index of the virtual CR2 register.*
- `l4_umword_t l4_vm_vmx_read_nat` (`l4_vm_vmx_vcpu_vmcs_t` \*vmcs, unsigned field) `L4_NOTHROW`  
*Read a natural-width software VMCS field.*
- `l4_uint16_t l4_vm_vmx_read_16` (`l4_vm_vmx_vcpu_vmcs_t` \*vmcs, unsigned field) `L4_NOTHROW`  
*Read a 16-bit software VMCS field.*

- [l4\\_uint32\\_t l4\\_vm\\_vmx\\_read\\_32](#) ([l4\\_vm\\_vmx\\_vcpu\\_vmcs\\_t](#) \*vmcs, unsigned field) [L4\\_NOTHROW](#)  
*Read a 32-bit software VMCS field.*
- [l4\\_uint64\\_t l4\\_vm\\_vmx\\_read\\_64](#) ([l4\\_vm\\_vmx\\_vcpu\\_vmcs\\_t](#) \*vmcs, unsigned field) [L4\\_NOTHROW](#)  
*Read a 64-bit software VMCS field.*
- [l4\\_uint64\\_t l4\\_vm\\_vmx\\_read](#) ([l4\\_vm\\_vmx\\_vcpu\\_vmcs\\_t](#) \*vmcs, unsigned field) [L4\\_NOTHROW](#)  
*Read any software VMCS field.*
- [void l4\\_vm\\_vmx\\_write\\_nat](#) ([l4\\_vm\\_vmx\\_vcpu\\_vmcs\\_t](#) \*vmcs, unsigned field, [l4\\_umword\\_t](#) val) [L4\\_NOTHROW](#)  
*Write to a natural-width software VMCS field.*
- [void l4\\_vm\\_vmx\\_write\\_16](#) ([l4\\_vm\\_vmx\\_vcpu\\_vmcs\\_t](#) \*vmcs, unsigned field, [l4\\_uint16\\_t](#) val) [L4\\_NOTHROW](#)  
*Write to a 16-bit software VMCS field.*
- [void l4\\_vm\\_vmx\\_write\\_32](#) ([l4\\_vm\\_vmx\\_vcpu\\_vmcs\\_t](#) \*vmcs, unsigned field, [l4\\_uint32\\_t](#) val) [L4\\_NOTHROW](#)  
*Write to a 32-bit software VMCS field.*
- [void l4\\_vm\\_vmx\\_write\\_64](#) ([l4\\_vm\\_vmx\\_vcpu\\_vmcs\\_t](#) \*vmcs, unsigned field, [l4\\_uint64\\_t](#) val) [L4\\_NOTHROW](#)  
*Write to a 64-bit software VMCS field.*
- [void l4\\_vm\\_vmx\\_write](#) ([l4\\_vm\\_vmx\\_vcpu\\_vmcs\\_t](#) \*vmcs, unsigned field, [l4\\_uint64\\_t](#) val) [L4\\_NOTHROW](#)  
*Write to an arbitrary software VMCS field.*
- [void l4\\_vm\\_vmx\\_set\\_hw\\_vmcs](#) ([l4\\_vm\\_vmx\\_vcpu\\_vmcs\\_t](#) \*vmcs, [l4\\_cap\\_idx\\_t](#) vmcs\_cap) [L4\\_NOTHROW](#)  
*Associate the software VMCS with a vCPU context, i.e.*
- [l4\\_cap\\_idx\\_t l4\\_vm\\_vmx\\_get\\_hw\\_vmcs](#) ([l4\\_vm\\_vmx\\_vcpu\\_vmcs\\_t](#) \*vmcs) [L4\\_NOTHROW](#)  
*Get the vCPU context (i.e.*

#### 14.1.10.6.3.1 Detailed Description

Virtual machine API for VMX.

#### 14.1.10.6.3.2 Typedef Documentation

##### [l4\\_vm\\_vmx\\_vcpu\\_state\\_t](#)

```
typedef struct l4_vm_vmx_vcpu_state_t l4_vm_vmx_vcpu_state_t
```

VMX vCPU state.

This is a specialization of the generic vCPU state for VMX. This data structure represents the following memory layout:

- 0x000 - 0x1ff: Standard vCPU state (with padding). See [l4\\_vcpu\\_state\\_t](#).
- 0x200 - 0x3ff: VMX information members (with padding). See [l4\\_vm\\_vmx\\_vcpu\\_infos\\_t](#).
- 0x400 - 0xffff: VMX software VMCS. See [l4\\_vm\\_vmx\\_vcpu\\_vmcs\\_t](#).

#### Note

The memory layout is documented here for reference purposes. However, the users are strongly discouraged from accessing the data structure directly. The API functions defined in this file are the preferred way of achieving the functionality.

## **l4\_vm\_vmx\_vcpu\_vmcs\_t**

```
typedef struct l4_vm_vmx_vcpu_vmcs_t l4_vm_vmx_vcpu_vmcs_t
```

VMX software VMCS.

This data structure represents the following memory layout:

- 0x000 - 0x007: Reserved (ignored by the kernel). In the hardware VMCS, the revision identifier and the abort indicator are stored in this area. Hereby we simply ignore these two entries.
- 0x008 - 0x00f: User space data (ignored by the kernel). This currently stores the pointer to a different software VMCS whose content has been loaded to this software VMCS.
- 0x010 - 0x013: VMCS field index of the software-defined CR2 field in the software VMCS.
- 0x014 - 0x017: Reserved.
- 0x018 - 0x01f: Capability of the vCPU context, i.e. the hardware VMCS object (with padding).
- 0x020 - 0x047: Software VMCS field offset table. See [l4\\_vmx\\_offset\\_table\\_t](#).
- 0x048 - 0x0bf: Reserved.
- 0x0c0 - 0xabf: Software VMCS fields (with padding).
- 0xac0 - 0xbff: Software VMCS fields dirty bitmap (with padding).

### **Note**

The memory layout is documented here for reference purposes. However, the users are strongly discouraged from accessing the data structure directly. The API functions defined in this file are the preferred way of achieving the functionality.

## **l4\_vmx\_offset\_table\_t**

```
typedef struct l4_vmx_offset_table_t l4_vmx_offset_table_t
```

Software VMCS field offset table.

This data structure represents the following memory layout:

- 0x00 - 0x02: 3 offsets for 16-bit fields.
- 0x03: Reserved.
- 0x04 - 0x06: 3 offsets for 64-bit fields.
- 0x07: Reserved.
- 0x08 - 0x0a: 3 offsets for 32-bit fields.
- 0x0b: Reserved.
- 0x0c - 0x0e: 3 offsets for natural-width fields.
- 0x0f: Reserved.
- 0x10 - 0x12: 3 limits for 16-bit fields.

- 0x13: Reserved.
- 0x14 - 0x16: 3 limits for 64-bit fields.
- 0x17: Reserved.
- 0x18 - 0x1a: 3 limits for 32-bit fields.
- 0x1b: Reserved.
- 0x1c - 0x1e: 3 limits for natural-width fields.
- 0x1f: Reserved.
- 0x20 - 0x23: 4 index shifts.
- 0x24: Offset of the first software VMCS field.
- 0x25: Size of the software VMCS fields.
- 0x26 - 0x27: Reserved.

The offsets/limits in each size category are in the following order:

- Control fields.
- Read-only fields.
- Guest fields.

The index shifts are in the following order:

- 16-bit.
- 64-bit.
- 32-bit.
- Natural-width.

All offsets/limits/sizes are represented in a 64-byte granule.

The offsets (after being multiplied by 64) are indexes in the values array in [l4\\_vm\\_vmx\\_vcpu\\_vmcs\\_t](#) and bit indexes in the dirty\_bitmap array in [l4\\_vm\\_vmx\\_vcpu\\_vmcs\\_t](#).

The limits (after being multiplied by 64) represent the range of the available indexes.

#### Note

The memory layout is documented here for reference purposes. However, the users are strongly discouraged from accessing the data structure directly. The API functions defined in this file are the preferred way of achieving the functionality.

### 14.1.10.6.3.3 Enumeration Type Documentation

#### L4\_vm\_vmx\_caps\_regs

```
enum L4_vm_vmx_caps_regs
```

Exported VMX capability registers.

#### Enumerator

---

L4_VM_VMX_BASIC_REG	Basic VMX capabilities.
L4_VM_VMX_TRUE_PINBASED_CTLN_REG	True pin-based control caps.
L4_VM_VMX_TRUE_PROCBASED_CTLN_REG	True processor based control caps.
L4_VM_VMX_TRUE_EXIT_CTLN_REG	True exit control caps.
L4_VM_VMX_TRUE_ENTRY_CTLN_REG	True entry control caps.
L4_VM_VMX_MISC_REG	Misc caps.
L4_VM_VMX_CR0_FIXED0_REG	Fixed to 0 bits of CR0.
L4_VM_VMX_CR0_FIXED1_REG	Fixed to 1 bits of CR0.
L4_VM_VMX_CR4_FIXED0_REG	Fixed to 0 bits of CR4.
L4_VM_VMX_CR4_FIXED1_REG	Fixed to 1 bits of CR4.
L4_VM_VMX_VMCS_ENUM_REG	VMCS enumeration info.
L4_VM_VMX_PROCBASED_CTLN2_REG	Processor based control 2 caps.
L4_VM_VMX_EPT_VPID_CAP_REG	EPT and VPID caps.
L4_VM_VMX_NESTED_REVISION	Nested VMCS revision.
L4_VM_VMX_NUM_CAPS_REGS	Total number of VMX capability registers.

Definition at line 28 of file [\\_\\_vm-vmx.h](#).

### L4\_vm\_vmx\_dfl1\_regs

```
enum L4_vm_vmx_dfl1_regs
```

Exported VMX capability registers (default to 1 bits).

#### Enumerator

L4_VM_VMX_PINBASED_CTLN_DFL1_REG	Default 1 bits in pin-based controls.
L4_VM_VMX_PROCBASED_CTLN_DFL1_REG	Default 1 bits in processor-based controls.
L4_VM_VMX_EXIT_CTLN_DFL1_REG	Default 1 bits in exit controls.
L4_VM_VMX_ENTRY_CTLN_DFL1_REG	Default 1 bits in entry controls.
L4_VM_VMX_NUM_DFL1_REGS	Total number of default on registers.

Definition at line 51 of file [\\_\\_vm-vmx.h](#).

### L4\_vm\_vmx\_sw\_fields

```
enum L4_vm_vmx_sw_fields
```

Additional (software-defined) VMCS fields.

The VMCS offsets defined here are actually not in the hardware VMCS. However our VMMs run in user mode and need to have access to certain registers available in kernel mode only. So we put them into our software VMCS.

#### Enumerator

L4_VM_VMX_VMCS_CR2	Software VMCS offset for CR2.  <b>Note</b>  You usually need to check this value against the value you get from <code>l4_vm_vmx_get_cr2_index()</code> to make sure you are running on a compatible kernel.
L4_VM_VMX_VMCS_NAT_ARG0	Custom argument passed from kernel to user space.
L4_VM_VMX_VMCS_NAT_ARG1	Custom argument passed from kernel to user space.
L4_VM_VMX_VMCS_NAT_ARG2	Custom argument passed from kernel to user space.
L4_VM_VMX_VMCS_NAT_ARG3	Custom argument passed from kernel to user space.
L4_VM_VMX_VMCS_XCR0	VMCS offset of extended control register XCR0.
L4_VM_VMX_VMCS_MSR_SYSCALL_MASK	VMCS offset of system call flag mask MSR.
L4_VM_VMX_VMCS_MSR_LSTAR	VMCS offset of IA32e mode system call target address MSR.
L4_VM_VMX_VMCS_MSR_CSTAR	VMCS offset of IA32 mode system call target address MSR.
L4_VM_VMX_VMCS_MSR_TSC_AUX	VMCS offset of auxiliary TSC signature MSR.
L4_VM_VMX_VMCS_MSR_STAR	VMCS offset of system call target address MSR.
L4_VM_VMX_VMCS_MSR_KERNEL_GS_BASE	VMCS offset of GS base address swap target MSR.

Definition at line 69 of file [\\_\\_vm-vmx.h](#).

### L4\_vm\_vmx\_vmcs\_sizes

```
enum L4_vm_vmx_vmcs_sizes
```

Sizes of software VMCS members.

#### Enumerator

L4_VM_VMX_VMCS_SIZE_VALUES	Size of the software VMCS values member.
L4_VM_VMX_VMCS_SIZE_DIRTY_BITMAP	Size of the software VMCS dirty bitmap member.

Definition at line 170 of file [\\_\\_vm-vmx.h](#).

### 14.1.10.6.3.4 Function Documentation

#### l4\_vm\_vmx\_clear()

```
void l4_vm_vmx_clear (
    l4_vm_vmx_vcpu_vmcs_t * vmcs,
    l4_vm_vmx_vcpu_vmcs_t * dest_vmcs) [inline]
```

Save the content from the software VMCS to a different software VMCS.

#### Parameters

<i>vmcs</i>	Pointer to the source software VMCS.
<i>dest_vmcs</i>	Pointer to the destination software VMCS.

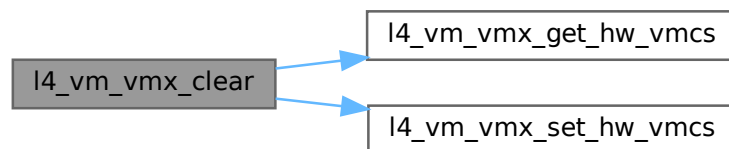
This function is comparable to the VMX VMCLEAR instruction.

Definition at line 698 of file [\\_\\_vm-vmx.h](#).

References [L4\\_NOTHROW](#), [l4\\_vm\\_vmx\\_get\\_hw\\_vmcs\(\)](#), [l4\\_vm\\_vmx\\_set\\_hw\\_vmcs\(\)](#), and [L4\\_VM\\_VMX\\_VMCS\\_SIZE\\_DIRTY\\_BITM](#)

Referenced by [l4\\_vm\\_vmx\\_ptr\\_load\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### **`l4_vm_vmx_field_len()`**

```
unsigned l4_vm_vmx_field_len (  
    unsigned field) [inline]
```

Return length in bytes of a VMCS field.

#### **Parameters**

<i>field</i>	Field number.
--------------	---------------



**Returns**

Width of field in bytes.

Definition at line 593 of file [\\_\\_vm-vmx.h](#).

References [L4\\_NOTHROW](#), and [l4\\_vm\\_vmx\\_field\\_order\(\)](#).

Here is the call graph for this function:

**`l4_vm_vmx_field_order()`**

```
unsigned l4_vm_vmx_field_order (  
    unsigned field) [inline]
```

Return length in power of two (bytes) of a VMCS field.

**Parameters**

<i>field</i>	Field number.
--------------	---------------

**Returns**

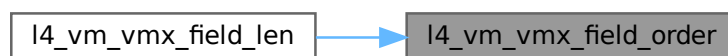
Width of field in power of two (bytes).

Definition at line 600 of file [\\_\\_vm-vmx.h](#).

References [L4\\_NOTHROW](#).

Referenced by [l4\\_vm\\_vmx\\_field\\_len\(\)](#).

Here is the caller graph for this function:



### **l4\_vm\_vmx\_get\_caps()**

```
l4_uint64_t l4_vm_vmx_get_caps (
    l4_vm_vmx_vcpu_state_t const * vcpu_state,
    enum L4_vm_vmx_caps_regs caps_reg) [inline]
```

Get a capability register for VMX.

#### **Parameters**

---

<i>vcpu_state</i>	Pointer to the vCPU state.
<i>caps_reg</i>	Capability register index (see <a href="#">L4_vm_vmx_caps_regs</a> ).

**Returns**

The value of the capability register.

Definition at line 884 of file [\\_\\_vm-vmx.h](#).

References [L4\\_NOTHROW](#).

**l4\_vm\_vmx\_get\_caps\_default1()**

```
l4_uint32_t l4_vm_vmx_get_caps_default1 (
    l4_vm_vmx_vcpu_state_t const * vcpu_state,
    enum L4_vm_vmx_dfl1_regs dfl1_reg) [inline]
```

Get a default to one capability register for VMX.

**Parameters**

<i>vcpu_state</i>	Pointer to the vCPU state.
<i>dfl1_reg</i>	Default to 1 capability register index (see <a href="#">L4_vm_vmx_dfl1_regs</a> ).

**Returns**

The value of the capability register.

Definition at line 892 of file [\\_\\_vm-vmx.h](#).

References [L4\\_NOTHROW](#).

**l4\_vm\_vmx\_get\_cr2\_index()**

```
l4_uint32_t l4_vm_vmx_get_cr2_index (
    l4_vm_vmx_vcpu_vmcs_t const * vmcs) [inline]
```

Get the software VMCS field index of the virtual CR2 register.

**Parameters**

<i>vmcs</i>	Pointer to the software VMCS.
-------------	-------------------------------

**Returns**

The field index used for the virtual CR2 register as used by the current Fiasco.OC interface.

The CR2 register is actually not in the hardware VMCS, however our VMMs run in user mode and need to have access to this register so we put it into our software VMCS.

**See also**

[L4\\_VM\\_VMX\\_VMCS\\_CR2](#)

Definition at line 900 of file [\\_\\_vm-vmx.h](#).

References [L4\\_NOTHROW](#).

## l4\_vm\_vmx\_get\_hw\_vmcs()

```
l4_cap_idx_t l4_vm_vmx_get_hw_vmcs (
    l4_vm_vmx_vcpu_vmcs_t * vmcs) [inline]
```

Get the vCPU context (i.e.

the hardware VMCS object) associated with the software VMCS.

### Parameters

<i>vmcs</i>	Pointer to the software VMCS.
-------------	-------------------------------

### Returns

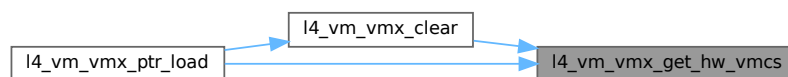
vCPU context (hardware VMCS object) capability.

Definition at line 915 of file [\\_\\_vm-vmx.h](#).

References [L4\\_CAP\\_MASK](#), and [L4\\_NOTHROW](#).

Referenced by [l4\\_vm\\_vmx\\_clear\(\)](#), and [l4\\_vm\\_vmx\\_ptr\\_load\(\)](#).

Here is the caller graph for this function:



## l4\_vm\_vmx\_ptr\_load()

```
void l4_vm_vmx_ptr_load (
    l4_vm_vmx_vcpu_vmcs_t * vmcs,
    l4_vm_vmx_vcpu_vmcs_t * src_vmcs) [inline]
```

Load the content from a different software VMCS to the software VMCS.

### Parameters

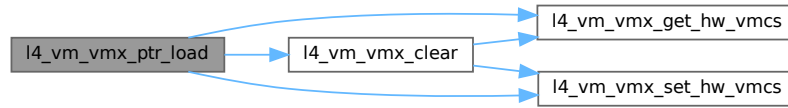
<i>vmcs</i>	Pointer to the destination software VMCS.
<i>src_vmcs</i>	Pointer to the source software VMCS.

This function is comparable to the VMX VMPTRLD instruction.

Definition at line 719 of file [\\_\\_vm-vmx.h](#).

References [L4\\_NOTHROW](#), [l4\\_vm\\_vmx\\_clear\(\)](#), [l4\\_vm\\_vmx\\_get\\_hw\\_vmcs\(\)](#), [l4\\_vm\\_vmx\\_set\\_hw\\_vmcs\(\)](#), and [L4\\_VM\\_VMX\\_VMCS\\_SIZE\\_DIRTY\\_BITMAP](#).

Here is the call graph for this function:



## `l4_vm_vmx_read()`

```

l4_uint64_t l4_vm_vmx_read (
    l4_vm_vmx_vcpu_vmcs_t * vmcs,
    unsigned field) [inline]
  
```

Read any software VMCS field.

### Parameters

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.

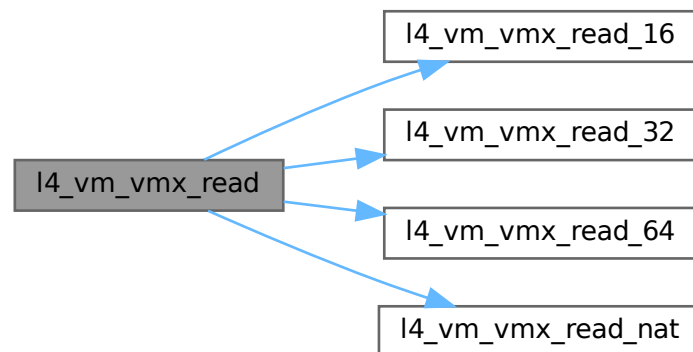
### Returns

The value of the software VMCS field with the given index.

Definition at line 787 of file `__vm-vmx.h`.

References [L4\\_NOTHROW](#), [l4\\_vm\\_vmx\\_read\\_16\(\)](#), [l4\\_vm\\_vmx\\_read\\_32\(\)](#), [l4\\_vm\\_vmx\\_read\\_64\(\)](#), and [l4\\_vm\\_vmx\\_read\\_nat\(\)](#).

Here is the call graph for this function:



## **l4\_vm\_vmx\_read\_16()**

```
l4_uint16_t l4_vm_vmx_read_16 (  
    l4_vm_vmx_vcpu_vmcs_t * vmcs,  
    unsigned field) [inline]
```

Read a 16-bit software VMCS field.

### **Parameters**

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.

### **Returns**

The value of the software VMCS field with the given index.

Definition at line 754 of file [\\_\\_vm-vmx.h](#).

References [L4\\_NOTHROW](#).

Referenced by [l4\\_vm\\_vmx\\_read\(\)](#).

Here is the caller graph for this function:



## **l4\_vm\_vmx\_read\_32()**

```
l4_uint32_t l4_vm_vmx_read_32 (  
    l4_vm_vmx_vcpu_vmcs_t * vmcs,  
    unsigned field) [inline]
```

Read a 32-bit software VMCS field.

### **Parameters**

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.

**Returns**

The value of the software VMCS field with the given index.

Definition at line 765 of file [\\_\\_vm-vmx.h](#).

References [L4\\_NOTHROW](#).

Referenced by [l4\\_vm\\_vmx\\_read\(\)](#).

Here is the caller graph for this function:

**l4\_vm\_vmx\_read\_64()**

```
l4_uint64_t l4_vm_vmx_read_64 (  
    l4_vm_vmx_vcpu_vmcs_t * vmcs,  
    unsigned field) [inline]
```

Read a 64-bit software VMCS field.

**Parameters**

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.

**Returns**

The value of the software VMCS field with the given index.

Definition at line 776 of file [\\_\\_vm-vmx.h](#).

References [L4\\_NOTHROW](#).

Referenced by [l4\\_vm\\_vmx\\_read\(\)](#).

Here is the caller graph for this function:



**l4\_vm\_vmx\_read\_nat()**

```
l4_umword_t l4_vm_vmx_read_nat (
    l4_vm_vmx_vcpu_vmcs_t * vmcs,
    unsigned field) [inline]
```

Read a natural-width software VMCS field.

**Parameters**

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.

**Returns**

The value of the software VMCS field with the given index.

Definition at line 743 of file [\\_\\_vm-vmx.h](#).

References [L4\\_NOTHROW](#).

Referenced by [l4\\_vm\\_vmx\\_read\(\)](#).

Here is the caller graph for this function:

**l4\_vm\_vmx\_set\_hw\_vmcs()**

```
void l4_vm_vmx_set_hw_vmcs (
    l4_vm_vmx_vcpu_vmcs_t * vmcs,
    l4_cap_idx_t vmcs_cap) [inline]
```

Associate the software VMCS with a vCPU context, i.e.

a hardware VMCS object.

The VMX extended vCPU state is unable to be resumed unless it is associated with a vCPU context, i.e. a hardware VMCS object: An `L4::Vcpu_context` from the user space point of view with its kernel counterpart `Vmx_vmcs`.



**Note**

When replacing the vCPU context, the dirty bitmap of the software VMCS is not touched, neither by the kernel nor by the API functions. This is on purpose, to enable efficient switching between separate VMs in the common case. If there is a logical discrepancy between the content of the software VMCS and the replaced vCPU context, the user is responsible for explicitly setting the relevant software VMCS fields and/or the relevant software VMCS dirty bitmap bits to ensure that the discrepancy is rectified on the next vCPU resume. This needs to be done regardless of using the API functions (the preferred way) or accessing the data structures directly (the discouraged way).

Replacing the vCPU context while the vCPU is currently running has no immediate effect until the next vCPU resume. In addition to that, the kernel might cache the vCPU context internally (in other words, the capability is not looked up on every vCPU resume). To remove the association of the current vCPU context, simply replace it by another vCPU context. The reference count of the previous vCPU context will be decremented accordingly on the next vCPU resume.

To remove the association of the current vCPU context without replacing it by another vCPU context, pass an invalid capability with the bit 3 set and trigger a vCPU resume. The vCPU resume will fail in this case (due to the missing vCPU context), but the reference count of the previous vCPU context will be decremented accordingly.

There is no need to explicitly remove the association of the current vCPU context before deleting the software VMCS. Deleting the software VMCS automatically disassociates it from the vCPU context and a vCPU context with a reference count of 0 will be eventually deleted as well.

If the hardware limitations on the usage of the vCPU context are not observed (i.e. no hardware VMCS being active on more than one physical CPU), the vCPU will fail to resume.

**Parameters**

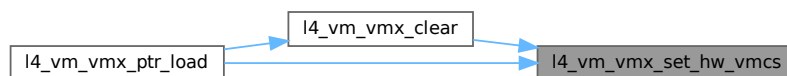
<i>vmcs</i>	Pointer to the software VMCS.
<i>vmcs_cap</i>	vCPU context (hardware VMCS object) capability.

Definition at line 907 of file [\\_\\_vm-vmx.h](#).

References [L4\\_NOTHROW](#).

Referenced by [l4\\_vm\\_vmx\\_clear\(\)](#), and [l4\\_vm\\_vmx\\_ptr\\_load\(\)](#).

Here is the caller graph for this function:

**`l4_vm_vmx_write()`**

```
void l4_vm_vmx_write (
    l4_vm_vmx_vcpu_vmcs_t * vmcs,
    unsigned field,
    l4_uint64_t val) [inline]
```

Write to an arbitrary software VMCS field.

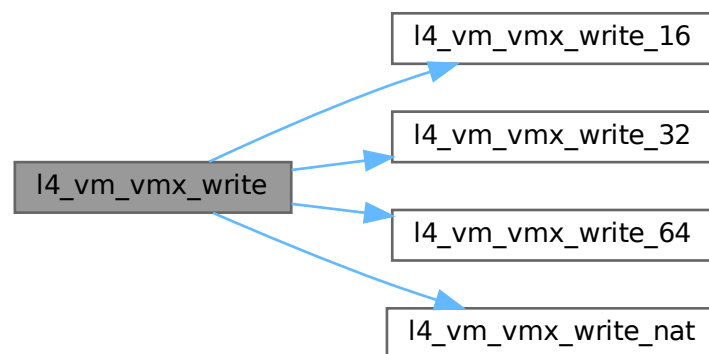
**Parameters**

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.
<i>val</i>	The value that shall be written to the given field.

Definition at line 868 of file [\\_\\_vm-vmx.h](#).

References [L4\\_NOTHROW](#), [l4\\_vm\\_vmx\\_write\\_16\(\)](#), [l4\\_vm\\_vmx\\_write\\_32\(\)](#), [l4\\_vm\\_vmx\\_write\\_64\(\)](#), and [l4\\_vm\\_vmx\\_write\\_nat\(\)](#).

Here is the call graph for this function:



### `l4_vm_vmx_write_16()`

```
void l4_vm_vmx_write_16 (
    l4_vm_vmx_vcpu_vmcs_t * vmcs,
    unsigned field,
    l4_uint16_t val) [inline]
```

Write to a 16-bit software VMCS field.

#### Parameters

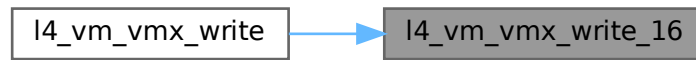
<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.
<i>val</i>	The value that shall be written to the given field.

Definition at line 820 of file [\\_\\_vm-vmx.h](#).

References [L4\\_NOTHROW](#).

Referenced by [l4\\_vm\\_vmx\\_write\(\)](#).

Here is the caller graph for this function:



### **l4\_vm\_vmx\_write\_32()**

```
void l4_vm_vmx_write_32 (
    l4_vm_vmx_vcpu_vmcs_t * vmcs,
    unsigned field,
    l4_uint32_t val) [inline]
```

Write to a 32-bit software VMCS field.

#### **Parameters**

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.
<i>val</i>	The value that shall be written to the given field.

Definition at line 836 of file [\\_\\_vm-vmx.h](#).

References [L4\\_NOTHROW](#).

Referenced by [l4\\_vm\\_vmx\\_write\(\)](#).

Here is the caller graph for this function:



### **l4\_vm\_vmx\_write\_64()**

```
void l4_vm_vmx_write_64 (
    l4_vm_vmx_vcpu_vmcs_t * vmcs,
    unsigned field,
    l4_uint64_t val) [inline]
```

Write to a 64-bit software VMCS field.

#### **Parameters**

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.
<i>val</i>	The value that shall be written to the given field.

Definition at line 852 of file [\\_\\_vm-vmx.h](#).

References [L4\\_NOTHROW](#).

Referenced by [l4\\_vm\\_vmx\\_write\(\)](#).

Here is the caller graph for this function:



### **`l4_vm_vmx_write_nat()`**

```

void l4_vm_vmx_write_nat (
    l4_vm_vmx_vcpu_vmcs_t * vmcs,
    unsigned field,
    l4_umword_t val) [inline]
  
```

Write to a natural-width software VMCS field.

#### **Parameters**

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.
<i>val</i>	The value that shall be written to the given field.

Definition at line 804 of file [\\_\\_vm-vmx.h](#).

References [L4\\_NOTHROW](#).

Referenced by [l4\\_vm\\_vmx\\_write\(\)](#).

Here is the caller graph for this function:



#### 14.1.10.6.4 VM API for TZ

Virtual Machine API for ARM TrustZone.

Collaboration diagram for VM API for TZ:



#### Data Structures

- struct [l4\\_vm\\_tz\\_state](#)  
*state structure for TrustZone VMs*

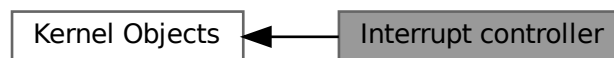
##### 14.1.10.6.4.1 Detailed Description

Virtual Machine API for ARM TrustZone.

#### 14.1.10.7 Interrupt controller

The C Icu interface, see [L4::Icu](#) for the C++ interface.

Collaboration diagram for Interrupt controller:



#### Data Structures

- struct [l4\\_icu\\_info\\_t](#)  
*Info structure for an ICU.*

#### Typedefs

- typedef struct [l4\\_icu\\_info\\_t](#) [l4\\_icu\\_info\\_t](#)  
*Info structure for an ICU.*

## Enumerations

- enum [L4\\_icu\\_flags](#) { [L4\\_ICU\\_FLAG\\_MSI](#) }  
*Flags for IRQ numbers used for the ICU.*

## Functions

- [l4\\_msgtag\\_t l4\\_icu\\_bind](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_cap\\_idx\\_t](#) irq) [L4\\_NOTHROW](#)  
*Bind an interrupt line of an interrupt controller to an interrupt object.*
- [l4\\_msgtag\\_t l4\\_icu\\_bind\\_u](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_cap\\_idx\\_t](#) irq, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Bind an interrupt line of an interrupt controller to an interrupt object.*
- [l4\\_msgtag\\_t l4\\_icu\\_unbind](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_cap\\_idx\\_t](#) irq) [L4\\_NOTHROW](#)  
*Remove binding of an interrupt line from the interrupt controller object.*
- [l4\\_msgtag\\_t l4\\_icu\\_unbind\\_u](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_cap\\_idx\\_t](#) irq, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Remove binding of an interrupt line from the interrupt controller object.*
- [l4\\_msgtag\\_t l4\\_icu\\_set\\_mode](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_umword\\_t](#) mode) [L4\\_NOTHROW](#)  
*Set interrupt mode.*
- [l4\\_msgtag\\_t l4\\_icu\\_set\\_mode\\_u](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_umword\\_t](#) mode, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Set interrupt mode.*
- [l4\\_msgtag\\_t l4\\_icu\\_info](#) ([l4\\_cap\\_idx\\_t](#) icu, [l4\\_icu\\_info\\_t](#) \*info) [L4\\_NOTHROW](#)  
*Get information about the ICU features.*
- [l4\\_msgtag\\_t l4\\_icu\\_info\\_u](#) ([l4\\_cap\\_idx\\_t](#) icu, [l4\\_icu\\_info\\_t](#) \*info, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Get information about the ICU features.*
- [l4\\_msgtag\\_t l4\\_icu\\_msi\\_info](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_uint64\\_t](#) source, [l4\\_icu\\_msi\\_info\\_t](#) \*msi\_info) [L4\\_NOTHROW](#)  
*Get MSI info about IRQ.*
- [l4\\_msgtag\\_t l4\\_icu\\_msi\\_info\\_u](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_uint64\\_t](#) source, [l4\\_icu\\_msi\\_info\\_t](#) \*msi\_info, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Get MSI info about IRQ.*
- [l4\\_msgtag\\_t l4\\_icu\\_unmask](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_umword\\_t](#) \*label, [l4\\_timeout\\_t](#) to) [L4\\_NOTHROW](#)  
*Unmask an IRQ line.*
- [l4\\_msgtag\\_t l4\\_icu\\_unmask\\_u](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_umword\\_t](#) \*label, [l4\\_timeout\\_t](#) to, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Unmask the given interrupt line.*
- [l4\\_msgtag\\_t l4\\_icu\\_mask](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_umword\\_t](#) \*label, [l4\\_timeout\\_t](#) to) [L4\\_NOTHROW](#)  
*Mask an IRQ line.*
- [l4\\_msgtag\\_t l4\\_icu\\_mask\\_u](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_umword\\_t](#) \*label, [l4\\_timeout\\_t](#) to, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Mask an IRQ line.*

#### 14.1.10.7.1 Detailed Description

The C Icu interface, see [L4::Icu](#) for the C++ interface.

##### Note

"ICU" is short for "interrupt control unit".

These functions define the interface for interrupt controllers, for binding IRQ objects to interrupt lines and other interrupt sources, as well as functions for masking and unmasking of interrupts.

To setup an IRQ line the following steps are required:

1. [l4\\_icu\\_set\\_mode\(\)](#) (optional if IRQ has a default mode)
2. [l4\\_rcv\\_ep\\_bind\\_thread\(\)](#) or [l4\\_rcv\\_ep\\_bind\\_snd\\_destination\(\)](#) to attach the IRQ object to a thread object.
3. [l4\\_icu\\_bind\(\)](#)
4. [l4\\_icu\\_unmask\(\)](#) to receive the first IRQ

For certain interrupt sources only some of these steps are necessary and supported, see [Scheduler](#) and [Virtual Console](#).

At most one [IRQs](#) object can be bound to a certain interrupt source and a certain [IRQs](#) object can be bound to at most one interrupt source.

##### Include File

```
#include <l4/sys/icu.h>
```

#### 14.1.10.7.2 Typedef Documentation

##### 14.1.10.7.2.1 l4\_icu\_info\_t

```
typedef struct l4_icu_info_t l4_icu_info_t
```

Info structure for an ICU.

This structure contains information about the features of an ICU.

##### See also

[l4\\_icu\\_info\(\)](#).

#### 14.1.10.7.3 Enumeration Type Documentation

##### 14.1.10.7.3.1 L4\_icu\_flags

```
enum L4_icu_flags
```

Flags for IRQ numbers used for the ICU.

##### Enumerator

<code>L4_ICU_FLAG_MSI</code>	Flag to denote that the IRQ is actually an MSI. This flag may be used for <a href="#">l4_icu_bind()</a> and <a href="#">l4_icu_unbind()</a> functions to denote that the IRQ number is meant to be an MSI.
------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Definition at line 53 of file [icu.h](#).

#### 14.1.10.7.4 Function Documentation

##### 14.1.10.7.4.1 l4\_icu\_bind()

```
l4_msgtag_t l4_icu_bind (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_cap_idx_t irq) [inline]
```

Bind an interrupt line of an interrupt controller to an interrupt object.

#### Parameters

<i>icu</i>	ICU object to bind <i>irq</i> to.
<i>irqnum</i>	IRQ line at the ICU.
<i>irq</i>	IRQ object to bind to this ICU.

#### Returns

Syscall return tag. The caller should check the return value using [l4\\_error\(\)](#) to check for errors and to identify the correct method for unmasking the interrupt. Return values < 0 indicate an error. A return value of 0 means a direct unmask via the IRQ object using [l4\\_irq\\_unmask\(\)](#). A return value of 1 means that the interrupt has to be unmasked via the ICU using [l4\\_icu\\_unmask\(\)](#).

#### Return values

<code>-L4_EINVAL</code>	<i>irq</i> is bound to an interrupt source.
<code>-L4_EPERM</code>	Insufficient permissions; see precondition.

#### Precondition

The capability *irq* must have the permission [L4\\_CAP\\_FPAGE\\_W](#).

In case the *irq* is already bound to an interrupt source, it is unbound first. In case the *irq* is bound and the interrupt source is bound to a different IRQ object, only the unbinding happens. An IRQ object that is bound to an interrupt source will get unbound if the IRQ object is deleted.

#### Examples

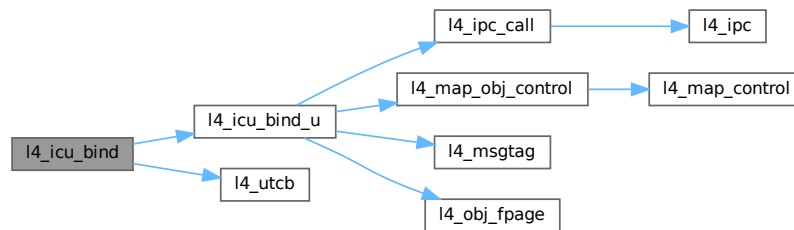
[examples/sys/isr/main.c](#).



Definition at line 496 of file [icu.h](#).

References [l4\\_icu\\_bind\\_u\(\)](#), [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.7.4.2 l4\_icu\_bind\_u()

```

l4_msgtag_t l4_icu_bind_u (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_cap_idx_t irq,
    l4_utcb_t * utcb) [inline]
  
```

Bind an interrupt line of an interrupt controller to an interrupt object.

##### Parameters

<i>icu</i>	The ICU object to bind <i>irq</i> to.
<i>irqnum</i>	IRQ line at the ICU.
<i>irq</i>	IRQ object for the given IRQ line to bind to this ICU.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

##### Returns

Syscall return tag. The caller should check the return value using [l4\\_error\(\)](#) to check for errors and to identify the correct method for unmasking the interrupt. Return values  $< 0$  indicate an error. A return value of 0 means a direct unmask via the IRQ object using [L4::irq::unmask](#). A return value of 1 means that the interrupt has to be unmasked via the ICU using [L4::icu::unmask](#).

##### Return values

<i>-L4_EINVAL</i>	<i>irq</i> is bound to an interrupt source.
<i>-L4_EPERM</i>	Insufficient permissions; see precondition.

**Precondition**

The capability `irq` must have the permission `L4_CAP_FPAGE_W`.

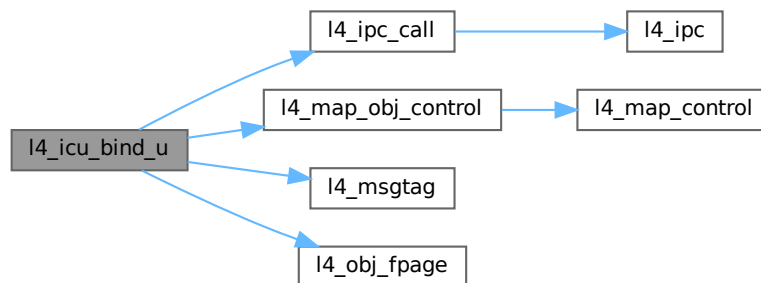
In case the `irq` is already bound to an interrupt source, it is unbound first. In case the `irq` is bound and the interrupt source is bound to a different `L4::lrq` object, only the unbinding happens. An `L4::lrq` object that is bound to an interrupt source will get unbound if the `L4::lrq` object is deleted.

Definition at line 396 of file `icu.h`.

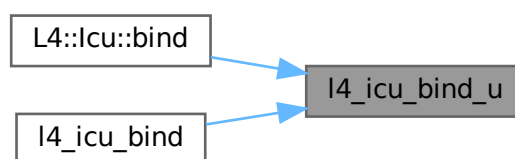
References `L4_CAP_FPAGE_RWS`, `L4_ICU_OP_BIND`, `l4_ipc_call()`, `L4_IPC_NEVER`, `l4_map_obj_control()`, `l4_msgtag()`, `L4_NOTHROW`, `l4_obj_fpage()`, `L4_PROTO_IRQ`, `l4_msg_regs_t::mr`, and `l4_fpage_t::raw`.

Referenced by `L4::lcu::bind()`, and `l4_icu_bind()`.

Here is the call graph for this function:



Here is the caller graph for this function:

**14.1.10.7.4.3 l4\_icu\_info()**

```

l4_msgtag_t l4_icu_info (
    l4_cap_idx_t icu,
    l4_icu_info_t * info) [inline]
  
```

Get information about the ICU features.

**Parameters**

	<i>icu</i>	The ICU object from which information shall be retrieved.
out	<i>info</i>	Info structure to be filled with information.

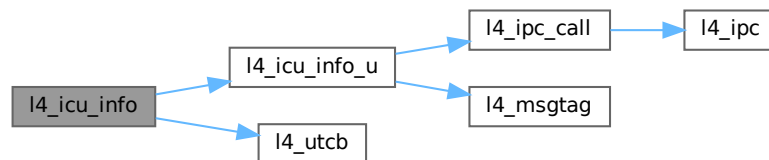
### Returns

Syscall return tag

Definition at line 504 of file [icu.h](#).

References [l4\\_icu\\_info\\_u\(\)](#), [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.7.4.4 l4\_icu\_info\_u()

```

l4_msgtag_t l4_icu_info_u (
    l4_cap_idx_t icu,
    l4_icu_info_t * info,
    l4_utcb_t * utcb) [inline]

```

Get information about the ICU features.

### Parameters

	<i>icu</i>	The ICU object from which MSI information shall be retrieved.
out	<i>info</i>	Info structure to be filled with information.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

**Returns**

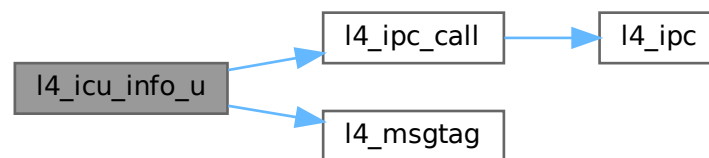
Syscall return tag

Definition at line 420 of file [icu.h](#).

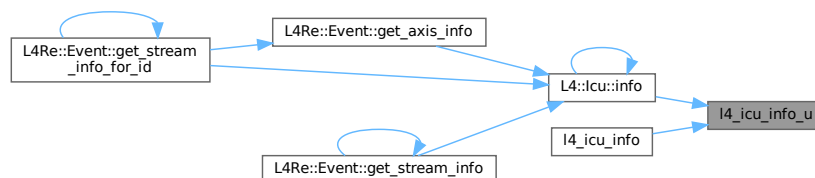
References [L4\\_ICU\\_OP\\_INFO](#), [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_NOTHROW](#), [L4\\_PROTO\\_IRQ](#), and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [L4::l4u::info\(\)](#), and [l4\\_icu\\_info\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**14.1.10.7.4.5 l4\_icu\_mask()**

```

l4_msgtag_t l4_icu_mask (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_umword_t * label,
    l4_timeout_t to) [inline]
  
```

Mask an IRQ line.

**Parameters**

<i>icu</i>	The ICU object where the IRQ line shall be masked.
------------	----------------------------------------------------

<i>irqnum</i>	IRQ line at the ICU.
<i>label</i>	If non-NULL, the function also performs an open wait IPC operation waiting for the next message, and the received label is returned here.
<i>to</i>	IPC timeout, if unsure use L4_IPC_NEVER.

#### Returns

Syscall return tag. If *label* is NULL, this function performs an IPC send-only operation and there is no return value except [L4\\_MSGTAG\\_ERROR](#) indicating success or failure of the send operation. In this case use [l4\\_ipc\\_error\(\)](#) to check for errors and **do not** use [l4\\_error\(\)](#).

Definition at line 518 of file [icu.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.7.4.6 l4\_icu\_mask\_u()

```

l4_msgtag_t l4_icu_mask_u (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_umword_t * label,
    l4_timeout_t to,
    l4_utcb_t * utcb) [inline]
  
```

Mask an IRQ line.

#### Parameters

<i>icu</i>	The ICU object where the IRQ line shall be masked.
<i>irqnum</i>	IRQ line at the ICU.
<i>label</i>	If NULL, this function is a send-only message to the ICU. If not NULL, this function will enter an open wait after sending the mask message and the received label is returned here.
<i>to</i>	The timeout-pair (send and receive) that shall be used for this operation. The receive timeout is used with a non-NULL <i>label</i> only.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

**Returns**

Syscall return tag. If `label` is `NULL`, this function performs an IPC send-only operation and there is no return value except `L4_MSGTAG_ERROR` indicating success or failure of the send operation. In this case use `l4_ipc_error()` to check for errors and **do not** use `l4_error()`.

Definition at line 483 of file `icu.h`.

References `L4_NOTHROW`.

Referenced by `L4::l4cu::mask()`.

Here is the caller graph for this function:

**14.1.10.7.4.7 l4\_icu\_msi\_info()**

```

l4_msgtag_t l4_icu_msi_info (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_uint64_t source,
    l4_icu_msi_info_t * msi_info) [inline]
  
```

Get MSI info about IRQ.

**Parameters**

	<i>icu</i>	The ICU object from which MSI information shall be retrieved.
	<i>irqnum</i>	IRQ line at the ICU.
	<i>source</i>	Platform dependent requester ID for MSIs. On IA32 we use a 20bit source filter value as described in the Intel IRQ remapping specification.
out	<i>msi_info</i>	A <code>l4_icu_msi_info_t</code> structure receiving the address and the data value to trigger this MSI.

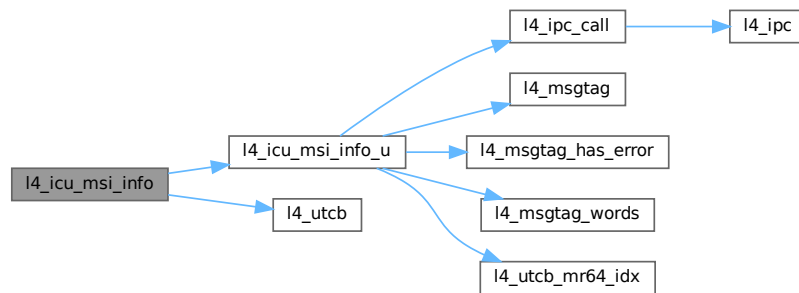
**Returns**

Syscall return tag

Definition at line 508 of file `icu.h`.

References `l4_icu_msi_info_u()`, `L4_NOTHROW`, and `l4_utcb()`.

Here is the call graph for this function:



#### 14.1.10.7.4.8 l4\_icu\_msi\_info\_u()

```

l4_msgtag_t l4_icu_msi_info_u (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_uint64_t source,
    l4_icu_msi_info_t * msi_info,
    l4_utcb_t * utcb) [inline]
  
```

Get MSI info about IRQ.

##### Parameters

	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .
	<i>icu</i>	The ICU object from which MSI information shall be retrieved.
	<i>irqnum</i>	IRQ line at the ICU.
	<i>source</i>	Platform dependent requester ID for MSIs. On IA32 we use a 20bit source filter value as described in the Intel IRQ remapping specification.
out	<i>msi_info</i>	A <a href="#">l4_icu_msi_info_t</a> structure receiving the address and the data value to trigger this MSI.

##### Returns

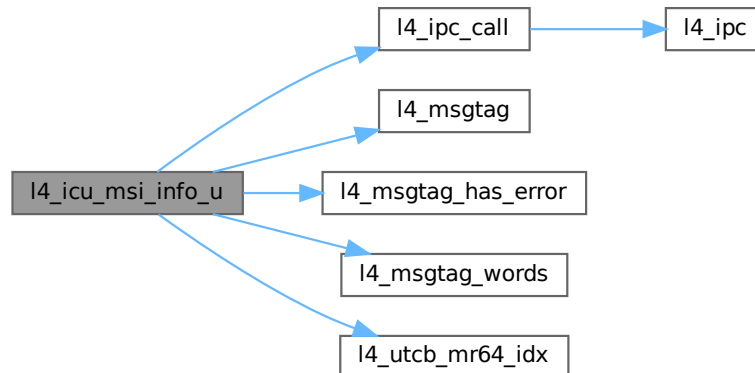
Syscall return tag

Definition at line 434 of file [icu.h](#).

References [L4\\_ICU\\_OP\\_MSI\\_INFO](#), [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [l4\\_msgtag\\_has\\_error\(\)](#), [l4\\_msgtag\\_words\(\)](#), [L4\\_NOTHROW](#), [L4\\_PROTO\\_IRQ](#), [L4\\_UNLIKELY](#), [l4\\_utcb\\_mr64\\_idx\(\)](#), [l4\\_msg\\_regs\\_t::mr](#), and [l4\\_msg\\_regs\\_t::mr64](#).

Referenced by [l4\\_icu\\_msi\\_info\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.1.10.7.4.9 l4\_icu\_set\_mode()

```

l4_msgtag_t l4_icu_set_mode (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_umword_t mode) [inline]
  
```

Set interrupt mode.

#### Parameters

<i>icu</i>	The ICU object.
<i>irqnum</i>	IRQ line at the ICU.
<i>mode</i>	Mode, see <a href="#">L4_irq_mode</a> .



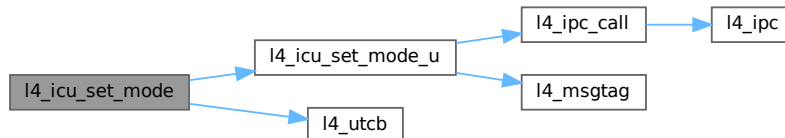
**Returns**

Syscall return tag

Definition at line 523 of file [icu.h](#).

References [l4\\_icu\\_set\\_mode\\_u\(\)](#), [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**14.1.10.7.4.10 l4\_icu\_set\_mode\_u()**

```

l4_msgtag_t l4_icu_set_mode_u (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_umword_t mode,
    l4_utcb_t * utcb) [inline]
  
```

Set interrupt mode.

**Parameters**

<i>icu</i>	The ICU object.
<i>irqnum</i>	IRQ line at the ICU.
<i>mode</i>	Mode, see <a href="#">L4_irq_mode</a> .
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

**Returns**

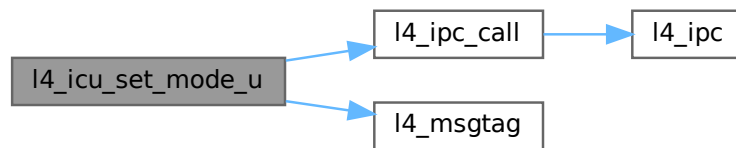
Syscall return tag

Definition at line 457 of file [icu.h](#).

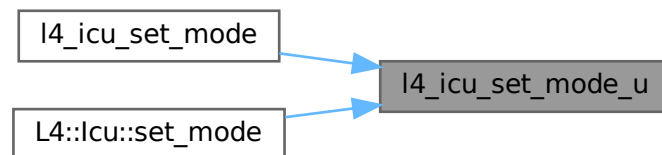
References [L4\\_ICU\\_OP\\_SET\\_MODE](#), [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_NOTHROW](#), [L4\\_PROTO\\_IRQ](#), and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [l4\\_icu\\_set\\_mode\(\)](#), and [L4::l4::set\\_mode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.1.10.7.4.11 l4\_icu\_unbind()

```

l4_msgtag_t l4_icu_unbind (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_cap_idx_t irq) [inline]
  
```

Remove binding of an interrupt line from the interrupt controller object.

##### Parameters

<i>icu</i>	The ICU object from where the binding shall be removed.
<i>irqnum</i>	IRQ line at the ICU.
<i>irq</i>	IRQ object to remove from the ICU.

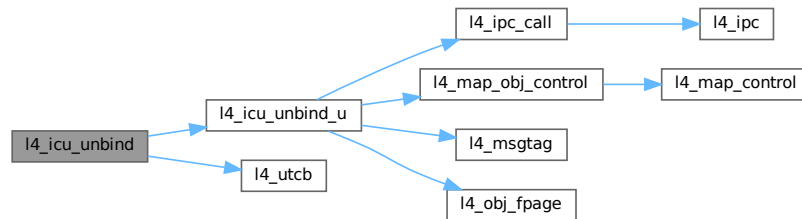
**Returns**

Syscall return tag

Definition at line 500 of file [icu.h](#).

References [l4\\_icu\\_unbind\\_u\(\)](#), [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**14.1.10.7.4.12 l4\_icu\_unbind\_u()**

```

l4_msgtag_t l4_icu_unbind_u (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_cap_idx_t irq,
    l4_utcb_t * utcb) [inline]
  
```

Remove binding of an interrupt line from the interrupt controller object.

**Parameters**

<i>icu</i>	The ICU object from where the binding shall be removed.
<i>irqnum</i>	IRQ line at the ICU.
<i>irq</i>	IRQ object to remove from the ICU.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

**Returns**

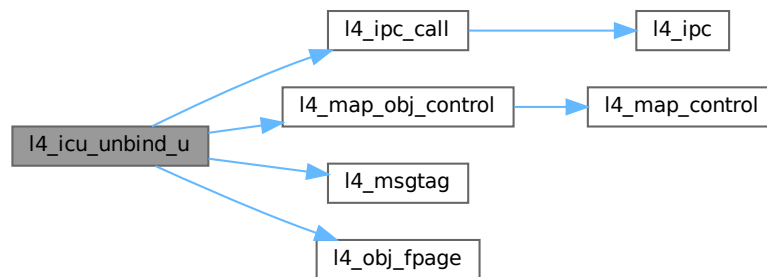
Syscall return tag

Definition at line 408 of file [icu.h](#).

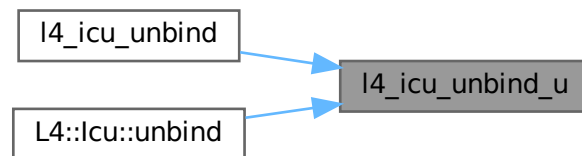
References [L4\\_CAP\\_FPAGE\\_RWS](#), [L4\\_ICU\\_OP\\_UNBIND](#), [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_map\\_obj\\_control\(\)](#), [l4\\_msgtag\(\)](#), [L4\\_NOTHROW](#), [l4\\_obj\\_fpage\(\)](#), [L4\\_PROTO\\_IRQ](#), [l4\\_msg\\_regs\\_t::mr](#), and [l4\\_fpage\\_t::raw](#).

Referenced by [l4\\_icu\\_unbind\(\)](#), and [L4::l4::unbind\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.1.10.7.4.13 l4\_icu\_unmask()

```

l4_msgtag_t l4_icu_unmask (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_umword_t * label,
    l4_timeout_t to) [inline]
  
```

Unmask an IRQ line.

#### Parameters

<i>icu</i>	The ICU object where the IRQ line shall be unmasked.
<i>irqnum</i>	IRQ line at the ICU.
<i>label</i>	If non-NULL, the function also performs an open wait IPC operation waiting for the next message, and the received label is returned here.
<i>to</i>	IPC timeout, if unsure use <code>L4_IPC_NEVER</code> .

## Returns

Syscall return tag. If `label` is `NULL`, this function performs an IPC send-only operation and there is no return value except `L4_MSGTAG_ERROR` indicating success or failure of the send operation. In this case use `l4_ipc_error()` to check for errors and **do not** use `l4_error()`.

Definition at line 513 of file `icu.h`.

References `L4_NOTHROW`, and `l4_utcb()`.

Here is the call graph for this function:

14.1.10.7.4.14 `l4_icu_unmask_u()`

```

l4_msgtag_t l4_icu_unmask_u (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_umword_t * label,
    l4_timeout_t to,
    l4_utcb_t * utcb) [inline]
  
```

Unmask the given interrupt line.

## Parameters

<i>icu</i>	The ICU object where the IRQ line shall be unmasked. When the object is an IRQ, the given interrupt line is ignored and instead the line which the IRQ is bound to (if any) is unmasked.
------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Its counterpart for explicitly masking an interrupt line is `L4::l4u::mask()`.

## Parameters

	<i>irqnum</i>	The interrupt line that shall be unmasked. Ignored if the object is an IRQ.
out	<i>label</i>	If <code>NULL</code> , this is a send-only unmask. If not <code>NULL</code> , this operation enters an open wait and the <i>protected label</i> shall be received here.
	<i>to</i>	The timeout-pair (send and receive) that shall be used for this operation. The receive timeout is used with a non- <code>NULL label</code> only.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <code>l4_utcb</code> .

## Returns

Syscall return tag. If `label` is `NULL`, this function performs an IPC send-only operation and there is no return value except `L4_MSGTAG_ERROR` indicating success or failure of the send operation. In this case use `l4_ipc_error()` to check for errors and **do not** use `l4_error()`.

Definition at line 488 of file `icu.h`.

References `L4_NOTHROW`.

### 14.1.10.8 IRQs

C IRQ interface, see `L4::Irq` for the C++ interface.

Collaboration diagram for IRQs:



## Enumerations

- enum `L4_irq_mode` {  
`L4_IRQ_F_NONE` = 0 , `L4_IRQ_F_SET_MODE` = 0x1 , `L4_IRQ_F_LEVEL` = 0x2 , `L4_IRQ_F_EDGE` = 0x0 ,  
`L4_IRQ_F_POS` = 0x0 , `L4_IRQ_F_NEG` = 0x4 , `L4_IRQ_F_BOTH` = 0x8 , `L4_IRQ_F_LEVEL_HIGH` =  
`L4_IRQ_F_SET_MODE` | `L4_IRQ_F_LEVEL` | `L4_IRQ_F_POS` ,  
`L4_IRQ_F_LEVEL_LOW` = `L4_IRQ_F_SET_MODE` | `L4_IRQ_F_LEVEL` | `L4_IRQ_F_NEG` , `L4_IRQ_F_POS_EDGE`  
= `L4_IRQ_F_SET_MODE` | `L4_IRQ_F_EDGE` | `L4_IRQ_F_POS` , `L4_IRQ_F_NEG_EDGE` = `L4_IRQ_F_`  
`SET_MODE` | `L4_IRQ_F_EDGE` | `L4_IRQ_F_NEG` , `L4_IRQ_F_BOTH_EDGE` = `L4_IRQ_F_SET_MODE` |  
`L4_IRQ_F_EDGE` | `L4_IRQ_F_BOTH` ,  
`L4_IRQ_F_MASK` = 0xf , `L4_IRQ_F_SET_WAKEUP` = 0x10 , `L4_IRQ_F_CLEAR_WAKEUP` = 0x20 }

*Interrupt attributes.*

## Functions

- `l4_msgtag_t l4_irq_detach (l4_cap_idx_t irq) L4_NOTHROW`  
*Detach from an interrupt source.*
- `l4_msgtag_t l4_irq_detach_u (l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW`  
*Detach from this interrupt.*
- `l4_msgtag_t l4_irq_bind_vcpu (l4_cap_idx_t irq, l4_cap_idx_t thread, l4_umword_t cfg) L4_NOTHROW`  
*Bind a thread to this Irq for vCPU interrupt forwarding.*
- `l4_msgtag_t l4_irq_bind_vcpu_u (l4_cap_idx_t irq, l4_cap_idx_t thread, l4_umword_t cfg, l4_utcb_t *utcb) L4_NOTHROW`  
*Bind a thread to this Irq for vCPU interrupt forwarding.*
- `l4_msgtag_t l4_irq_trigger (l4_cap_idx_t irq) L4_NOTHROW`  
*Trigger an IRQ.*

- [l4\\_msgtag\\_t l4\\_irq\\_trigger\\_u](#) ([l4\\_cap\\_idx\\_t](#) irq, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Trigger the object.*
- [l4\\_msgtag\\_t l4\\_irq\\_receive](#) ([l4\\_cap\\_idx\\_t](#) irq, [l4\\_timeout\\_t](#) to) [L4\\_NOTHROW](#)  
*Unmask and wait for specified IRQ.*
- [l4\\_msgtag\\_t l4\\_irq\\_receive\\_u](#) ([l4\\_cap\\_idx\\_t](#) irq, [l4\\_timeout\\_t](#) timeout, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Unmask and wait for this IRQ.*
- [l4\\_msgtag\\_t l4\\_irq\\_wait](#) ([l4\\_cap\\_idx\\_t](#) irq, [l4\\_umword\\_t](#) \*label, [l4\\_timeout\\_t](#) to) [L4\\_NOTHROW](#)  
*Unmask IRQ and wait for any message.*
- [l4\\_msgtag\\_t l4\\_irq\\_wait\\_u](#) ([l4\\_cap\\_idx\\_t](#) irq, [l4\\_umword\\_t](#) \*label, [l4\\_timeout\\_t](#) timeout, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Unmask IRQ and (open) wait for any message.*
- [l4\\_msgtag\\_t l4\\_irq\\_unmask](#) ([l4\\_cap\\_idx\\_t](#) irq) [L4\\_NOTHROW](#)  
*Unmask IRQ.*
- [l4\\_msgtag\\_t l4\\_irq\\_unmask\\_u](#) ([l4\\_cap\\_idx\\_t](#) irq, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Unmask this IRQ.*

#### 14.1.10.8.1 Detailed Description

C IRQ interface, see [L4::lrq](#) for the C++ interface.

The IRQ interface provides access to abstract interrupts provided by the microkernel. Interrupts may be

- hardware interrupts provided by the platform interrupt controller,
- virtual device interrupts provided by the microkernel's virtual devices (virtual serial or trace buffer) or
- virtual interrupts that can be triggered by user programs (IRQs) via [l4\\_irq\\_trigger\(\)](#).

For hardware and virtual device interrupts the `lrq` object must be bound to an interrupt source, see [Interrupt controller](#). To receive interrupts, the `lrq` object must be bound to a thread, see [l4\\_rcv\\_ep\\_bind\\_thread\(\)](#) and [l4\\_rcv\\_ep\\_bind\\_snd\\_destination\(\)](#).

IRQ objects can be created using a factory, see the [Factory](#) API (use [l4\\_factory\\_create\\_irq\(\)](#)).

#### Include File

```
#include <l4/sys/irq.h>
```

For the C++ interface refer to the [L4::lrq](#) API for an overview.

#### 14.1.10.8.2 Enumeration Type Documentation

##### 14.1.10.8.2.1 L4\_irq\_mode

```
enum L4\_irq\_mode
```

Interrupt attributes.

#### Enumerator

L4_IRQ_F_NONE	Flow types. None
L4_IRQ_F_SET_MODE	Valid flag, if not set, the set_mode operation does nothing.
L4_IRQ_F_LEVEL	Level triggered.
L4_IRQ_F_EDGE	Edge triggered.
L4_IRQ_F_POS	Positive trigger.
L4_IRQ_F_NEG	Negative trigger.
L4_IRQ_F_BOTH	Both edges trigger.
L4_IRQ_F_LEVEL_HIGH	Level high trigger.
L4_IRQ_F_LEVEL_LOW	Level low trigger.
L4_IRQ_F_POS_EDGE	Positive edge trigger.
L4_IRQ_F_NEG_EDGE	Negative edge trigger.
L4_IRQ_F_BOTH_EDGE	Both edges trigger.
L4_IRQ_F_MASK	Mask.
L4_IRQ_F_SET_WAKEUP	Wakeup source? Use irq as wakeup source
L4_IRQ_F_CLEAR_WAKEUP	Do not use irq as wakeup source.

Definition at line 70 of file [icu.h](#).

#### 14.1.10.8.3 Function Documentation

##### 14.1.10.8.3.1 l4\_irq\_bind\_vcpu()

```
l4_msgtag_t l4_irq_bind_vcpu (
    l4_cap_idx_t irq,
    l4_cap_idx_t thread,
    l4_umword_t cfg) [inline]
```

Bind a thread to this Irq for vCPU interrupt forwarding.

If the interrupt is triggered, the kernel will directly inject the interrupt into the guest. This requires that the thread is currently in extended vCPU user mode. Otherwise the interrupt will stay pending and gets injected on the next vCPU user mode transition. Optionally a doorbell Irq can be registered on the thread (see `Thread::register_doorbell_irq()`) that is triggered in this case.

If a guest has acknowledged the interrupt but has not yet issued an EOI (i.e. the interrupt is in "active" state), it is not possible to bind the Irq to a new thread object. Either wait for the guest to issue the EOI or detach() from the current thread. In this case the interrupt will stay active in the guest and it is the responsibility of the VMM to handle the eventual EOI of the guest.

#### Parameters

<i>irq</i>	The IRQ object that shall be bound.
<i>thread</i>	Thread object this Irq shall be bound to.
<i>cfg</i>	Architecture specific interrupt configuration.

#### Returns

Syscall return tag

#### Return values



<code>-L4_EPERM</code>	Insufficient permissions; see precondition.
<code>-L4_EBUSY</code>	Cannot bind to new thread because interrupt is active on previous thread and guest has to issue end-of-interrupt first.
<code>-L4_ENOSYS</code>	The kernel does not support direct interrupt forwarding.

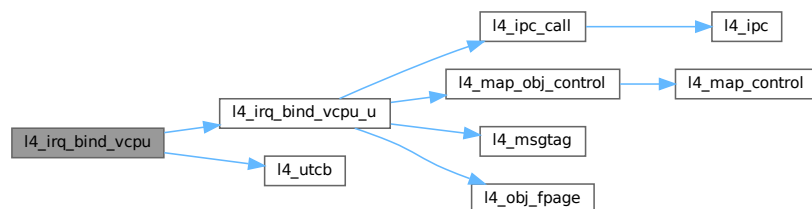
#### Precondition

The capabilities `irq` and `thread` both must have the permission [L4\\_CAP\\_FPAGE\\_S](#).

Definition at line 300 of file [irq.h](#).

References [l4\\_irq\\_bind\\_vcpu\\_u\(\)](#), [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.8.3.2 l4\_irq\_bind\_vcpu\_u()

```

l4_msgtag_t l4_irq_bind_vcpu_u (
    l4_cap_idx_t irq,
    l4_cap_idx_t thread,
    l4_umword_t cfg,
    l4_utcb_t * utcb) [inline]
  
```

Bind a thread to this Irq for vCPU interrupt forwarding.

#### Parameters

<i>irq</i>	The IRQ object that shall be bound. If the interrupt is triggered, the kernel will directly inject the interrupt into the guest. This requires that the thread is currently in extended vCPU user mode. Otherwise the interrupt will stay pending and gets injected on the next vCPU user mode transition. Optionally a doorbell Irq can be registered on the thread (see <code>Thread::register_doorbell_irq()</code> ) that is triggered in this case.
------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

If a guest has acknowledged the interrupt but has not yet issued an EOI (i.e. the interrupt is in "active" state), it is not possible to bind the Irq to a new thread object. Either wait for the guest to issue the EOI or `detach()` from the current thread. In this case the interrupt will stay active in the guest and it is the responsibility of the VMM to handle the eventual EOI of the guest.

#### Parameters

<i>thread</i>	Thread object this Irq shall be bound to.
<i>cfg</i>	Architecture specific interrupt configuration.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

## Returns

Syscall return tag

## Return values

<code>-L4_EPERM</code>	Insufficient permissions; see precondition.
<code>-L4_EBUSY</code>	Cannot bind to the new thread because interrupt is active on previous thread and guest has to issue end-of-interrupt first.
<code>-L4_ENOSYS</code>	The kernel does not support direct interrupt forwarding.

## Precondition

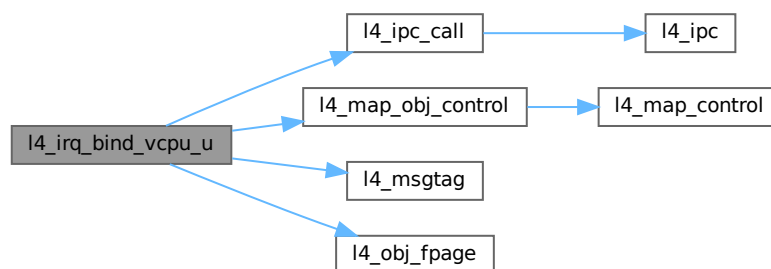
The invoked Irq capability and the capability `thread` both must have the permission [L4\\_CAP\\_FPAGE\\_S](#).

Definition at line 250 of file [irq.h](#).

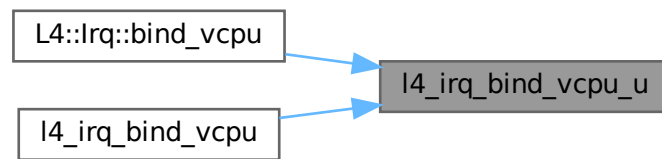
References [L4\\_CAP\\_FPAGE\\_RWS](#), [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_map\\_obj\\_control\(\)](#), [l4\\_msgtag\(\)](#), [L4\\_NOTHROW](#), [l4\\_obj\\_fpage\(\)](#), [L4\\_PROTO\\_IRQ\\_SENDER](#), [l4\\_msg\\_regs\\_t::mr](#), and [l4\\_fpage\\_t::raw](#).

Referenced by [L4::Irq::bind\\_vcpu\(\)](#), and [l4\\_irq\\_bind\\_vcpu\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.1.10.8.3.3 l4\_irq\_detach()

```
l4_msgtag_t l4_irq_detach (
    l4_cap_idx_t irq) [inline]
```

Detach from an interrupt source.

##### Parameters

<i>irq</i>	The IRQ object that shall be detached.
------------	----------------------------------------

##### Returns

Syscall return tag

##### Return values

0	Successfully detached, there was no interrupt pending.
1	Successfully detached, there was an interrupt pending.
2	Successfully detached, an active vIRQ was abandoned.
-L4_EPERM	Insufficient permissions; see precondition.

##### Precondition

The capability `irq` must have the permission `L4_CAP_FPAGE_S`.

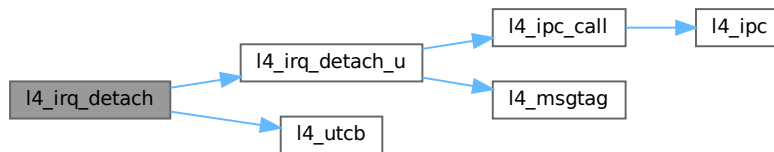
##### Examples

[examples/sys/isr/main.c](#).

Definition at line 294 of file [irq.h](#).

References [l4\\_irq\\_detach\\_u\(\)](#), [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.8.3.4 l4\_irq\_detach\_u()

```

l4_msgtag_t l4_irq_detach_u (
    l4_cap_idx_t irq,
    l4_utcb_t * utcb) [inline]
  
```

Detach from this interrupt.

##### Parameters

<i>irq</i>	The IRQ object that shall be detached.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

##### Returns

Syscall return tag

##### Return values

0	Successfully detached, there was no interrupt pending.
1	Successfully detached, there was an interrupt pending.
2	Successfully detached, an active vIRQ was abandoned.
-L4_EPERM	Insufficient permissions; see precondition.

**Precondition**

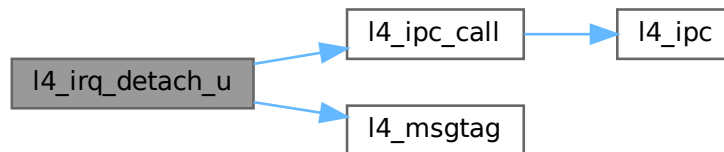
The invoked Irq capability must have the permission [L4\\_CAP\\_FPAGE\\_S](#).

Definition at line 242 of file [irq.h](#).

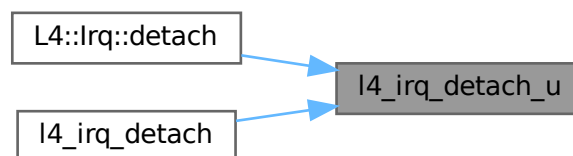
References [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_NOTHROW](#), and [L4\\_PROTO\\_IRQ\\_SENDER](#).

Referenced by [L4::Irq::detach\(\)](#), and [l4\\_irq\\_detach\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**14.1.10.8.3.5 l4\_irq\_receive()**

```

l4_msgtag_t l4_irq_receive (
    l4_cap_idx_t irq,
    l4_timeout_t to) [inline]
  
```

Unmask and wait for specified IRQ.

**Parameters**

<i>irq</i>	The IRQ object that shall be unmasked.
<i>to</i>	Timeout.

**Returns**

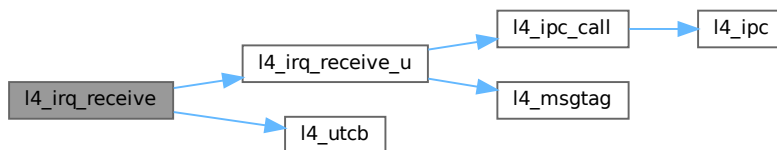
Syscall return tag

Definition at line 313 of file [irq.h](#).

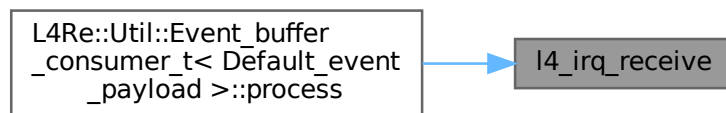
References [l4\\_irq\\_receive\\_u\(\)](#), [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Referenced by [L4Re::Util::Event\\_buffer\\_consumer\\_t< Default\\_event\\_payload >::process\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**14.1.10.8.3.6 l4\_irq\_receive\_u()**

```

l4_msgtag_t l4_irq_receive_u (
    l4_cap_idx_t irq,
    l4_timeout_t timeout,
    l4_utcb_t * utcb) [inline]
  
```

Unmask and wait for this IRQ.

**Parameters**

<i>irq</i>	The IRQ object that shall be unmasked.
<i>timeout</i>	Timeout.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

**Returns**

Syscall return tag

**Note**

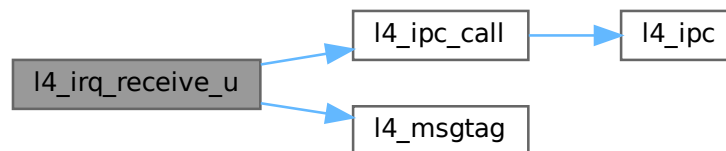
If this is the function normally used for your IRQs consider using [L4::Semaphore](#) instead of [L4::Irq](#).

Definition at line 270 of file [irq.h](#).

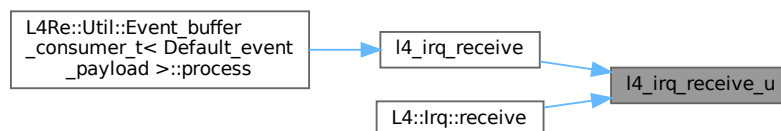
References [l4\\_ipc\\_call\(\)](#), [l4\\_msgtag\(\)](#), [L4\\_NOTHROW](#), and [L4\\_PROTO\\_IRQ](#).

Referenced by [l4\\_irq\\_receive\(\)](#), and [L4::Irq::receive\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**14.1.10.8.3.7 l4\_irq\_trigger()**

```
l4_msgtag_t l4_irq_trigger (
    l4_cap_idx_t irq) [inline]
```

Trigger an IRQ.

**Parameters**

<i>irq</i>	The IRQ object that shall be triggered.
------------	-----------------------------------------

**Returns**

Syscall return tag.

Note that this function is a send only operation, i.e. there is no return value except for a failed send operation. Especially `l4_error()` will return an error value from the message tag which still contains the IRQ protocol used for the send operation.

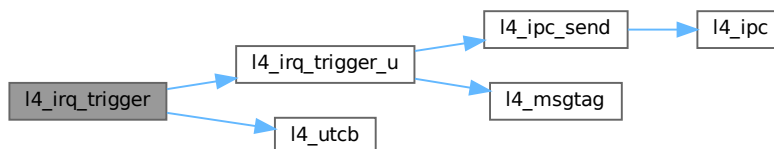
Use `l4_ipc_error()` to check for (send) errors.

Definition at line 307 of file `irq.h`.

References `l4_irq_trigger_u()`, `L4_NOTHROW`, and `l4_utcb()`.

Referenced by `l4_semaphore_up()`.

Here is the call graph for this function:



Here is the caller graph for this function:

**14.1.10.8.3.8 l4\_irq\_trigger\_u()**

```

l4_msgtag_t l4_irq_trigger_u (
    l4_cap_idx_t irq,
    l4_utcb_t * utcb) [inline]
  
```

Trigger the object.

**Parameters**

<i>irq</i>	The IRQ object that shall be triggered.
------------	-----------------------------------------



<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .
-------------	------------------------------------------------------------------------------------------------------------

### Returns

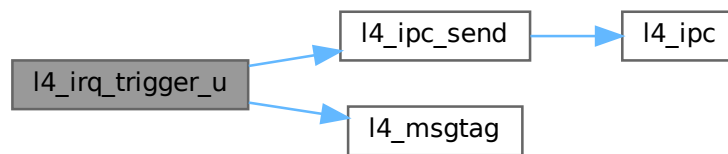
Syscall return tag for a send-only operation, this means there is no return value except [L4\\_MSGTAG\\_ERROR](#) indicating success or failure of the send operation. Use [l4\\_ipc\\_error\(\)](#) to check for errors and **do not** use [l4\\_error\(\)](#).

Definition at line 263 of file [irq.h](#).

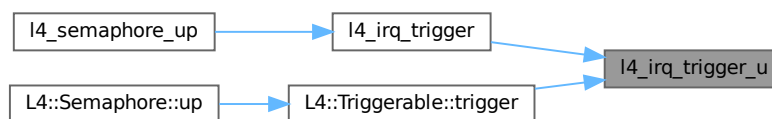
References [L4\\_IPC\\_BOTH\\_TIMEOUT\\_0](#), [l4\\_ipc\\_send\(\)](#), [l4\\_msgtag\(\)](#), [L4\\_NOTHROW](#), and [L4\\_PROTO\\_IRQ](#).

Referenced by [l4\\_irq\\_trigger\(\)](#), and [L4::Triggerable::trigger\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.1.10.8.3.9 l4\_irq\_unmask()

```

l4_msgtag_t l4_irq_unmask (
    l4_cap_idx_t irq) [inline]
  
```

Unmask IRQ.

### Parameters

<i>irq</i>	The IRQ object that shall be unmasked.
------------	----------------------------------------

#### Returns

Syscall return tag

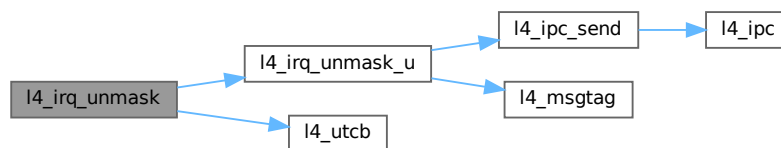
#### Note

[l4\\_irq\\_wait\(\)](#) and [l4\\_irq\\_receive\(\)](#) are doing the unmask themselves.

Definition at line 326 of file [irq.h](#).

References [l4\\_irq\\_unmask\\_u\(\)](#), [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.8.3.10 l4\_irq\_unmask\_u()

```

l4_msgtag_t l4_irq_unmask_u (
    l4_cap_idx_t irq,
    l4_utcb_t * utcb) [inline]
  
```

Unmask this IRQ.

#### Parameters

<i>irq</i>	The IRQ object that shall be unmasked.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

#### Returns

Syscall return tag for a send-only operation, this means there is no return value except [L4\\_MSGTAG\\_ERROR](#) indicating success or failure of the send operation. Use [l4\\_ipc\\_error\(\)](#) to check for errors and **do not** use [l4\\_error\(\)](#).

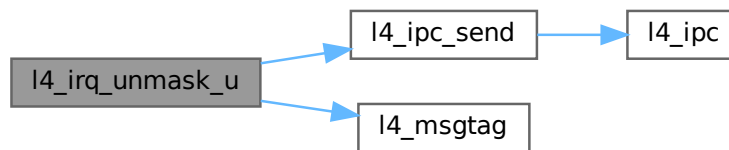
`Irq::wait()` and `Irq::receive()` operations already include an `unmask()`, do not use an extra `unmask()` in these cases.

Definition at line 286 of file `irq.h`.

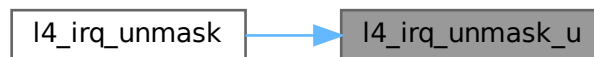
References `L4_IPC_NEVER`, `l4_ipc_send()`, `l4_msgtag()`, `L4_NOTHROW`, and `L4_PROTO_IRQ`.

Referenced by `l4_irq_unmask()`.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.1.10.8.3.11 l4\_irq\_wait()

```

l4_msgtag_t l4_irq_wait (
    l4_cap_idx_t irq,
    l4_unword_t * label,
    l4_timeout_t to) [inline]
  
```

Unmask IRQ and wait for any message.

##### Parameters

<i>irq</i>	The IRQ object that shall be unmasked.
<i>label</i>	Receive label.
<i>to</i>	Timeout.

**Returns**

Syscall return tag

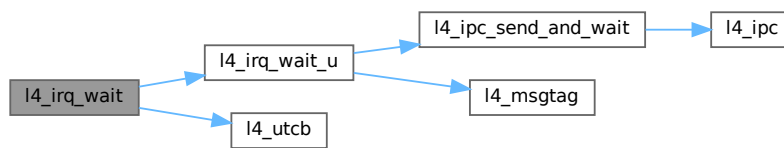
**Examples**

[examples/sys/isr/main.c](#).

Definition at line 319 of file [irq.h](#).

References [l4\\_irq\\_wait\\_u\(\)](#), [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**14.1.10.8.3.12 l4\_irq\_wait\_u()**

```

l4_msgtag_t l4_irq_wait_u (
    l4_cap_idx_t irq,
    l4_umword_t * label,
    l4_timeout_t timeout,
    l4_utcb_t * utcb) [inline]
  
```

Unmask IRQ and (open) wait for any message.

**Parameters**

<i>irq</i>	The IRQ object that shall be unmasked.
<i>label</i>	The <i>protected label</i> shall be received here.
<i>timeout</i>	Timeout.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

**Returns**

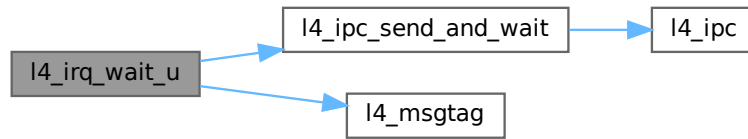
Syscall return tag

Definition at line 277 of file [irq.h](#).

References [l4\\_ipc\\_send\\_and\\_wait\(\)](#), [l4\\_msgtag\(\)](#), [L4\\_NOTHROW](#), and [L4\\_PROTO\\_IRQ](#).

Referenced by [l4\\_irq\\_wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.1.10.9 Platform Control C API

C interface for controlling platform-wide properties, see [L4::Platform\\_control](#) for the C++ interface.

Collaboration diagram for Platform Control C API:



#### Functions

- `l4_msgtag_t l4_platform_ctl_set_task_asid (l4_cap_idx_t pfc, l4_cap_idx_t task, l4_umword_t asid) L4_NOTHROW`  
*Set ASID of task.*
- `l4_msgtag_t l4_platform_ctl_system_suspend (l4_cap_idx_t pfc, l4_umword_t extras) L4_NOTHROW`  
*Enter suspend to RAM.*
- `l4_msgtag_t l4_platform_ctl_system_shutdown (l4_cap_idx_t pfc, l4_umword_t reboot) L4_NOTHROW`  
*Shutdown or reboot the system.*
- `l4_msgtag_t l4_platform_ctl_cpu_allow_shutdown (l4_cap_idx_t pfc, l4_umword_t phys_id, l4_umword_t enable) L4_NOTHROW`

*Allow a CPU to be shut down.*

- [l4\\_msgtag\\_t l4\\_platform\\_ctl\\_cpu\\_enable](#) ([l4\\_cap\\_idx\\_t](#) pfc, [l4\\_umword\\_t](#) phys\_id) [L4\\_NOTHROW](#)

*Enable an offline CPU.*

- [l4\\_msgtag\\_t l4\\_platform\\_ctl\\_cpu\\_disable](#) ([l4\\_cap\\_idx\\_t](#) pfc, [l4\\_umword\\_t](#) phys\_id) [L4\\_NOTHROW](#)

*Disable an online CPU.*

#### 14.1.10.9.1 Detailed Description

C interface for controlling platform-wide properties, see [L4::Platform\\_control](#) for the C++ interface.

##### Include File

```
#include <l4/sys/platform_control.h>
```

The API allows a client to suspend, reboot or shutdown the system.

For the C++ interface refer to [L4::Platform\\_control](#)

#### 14.1.10.9.2 Function Documentation

##### 14.1.10.9.2.1 l4\_platform\_ctl\_cpu\_allow\_shutdown()

```
l4\_msgtag\_t l4_platform_ctl_cpu_allow_shutdown (
    l4\_cap\_idx\_t pfc,
    l4\_umword\_t phys_id,
    l4\_umword\_t enable) [inline]
```

Allow a CPU to be shut down.

##### Parameters

<i>pfc</i>	Capability selector for the platform-control object.
<i>phys_id</i>	Physical CPU id of CPU (e.g. local APIC id) to enable.
<i>enable</i>	Allow shutdown when 1, disallow when 0.

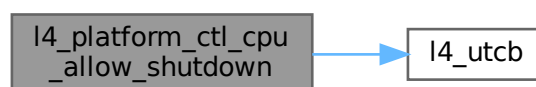
##### Returns

Syscall return tag

Definition at line 242 of file [platform\\_control.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.9.2.2 l4\_platform\_ctl\_cpu\_disable()

```
l4_msgtag_t l4_platform_ctl_cpu_disable (
    l4_cap_idx_t pfc,
    l4_umword_t phys_id) [inline]
```

Disable an online CPU.

##### Parameters

<i>pfc</i>	Capability to the platform control object.
<i>phys_id</i>	Physical CPU id of CPU (e.g. local APIC id) to disable.

##### Returns

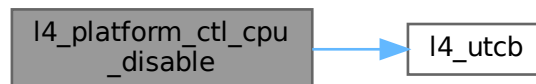
System call message tag

This function is currently only supported on the ARM EXYNOS platform.

Definition at line 281 of file [platform\\_control.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.9.2.3 l4\_platform\_ctl\_cpu\_enable()

```
l4_msgtag_t l4_platform_ctl_cpu_enable (
    l4_cap_idx_t pfc,
    l4_umword_t phys_id) [inline]
```

Enable an offline CPU.

##### Parameters

<i>pfc</i>	Capability to the platform control object.
<i>phys_id</i>	Physical CPU id of CPU (e.g. local APIC id) to enable.

**Returns**

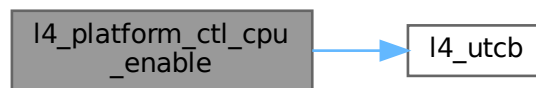
System call message tag

This function is currently only supported on the ARM EXYNOS platform.

Definition at line 274 of file [platform\\_control.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**14.1.10.9.2.4 l4\_platform\_ctl\_set\_task\_asid()**

```

l4_msgtag_t l4_platform_ctl_set_task_asid (
    l4_cap_idx_t pfc,
    l4_cap_idx_t task,
    l4_umword_t asid) [inline]
  
```

Set ASID of task.

On Cortex-R52 platforms, it might be necessary to control the VMID of a task or virtual machine explicitly. The IOMPU on such platforms will use it for further access control of device memory accesses. A privileged component can use this call to control the value.

The caller must have write permissions to the destination task.

**Parameters**

<i>pfc</i>	Capability selector for the platform-control object.
<i>task</i>	Capability selector of destination task
<i>asid</i>	New ASID value



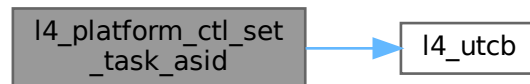
**Returns**

Syscall return tag

Definition at line 62 of file [\\_\\_platform\\_control-arm.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**14.1.10.9.2.5 l4\_platform\_ctl\_system\_shutdown()**

```

l4_msgtag_t l4_platform_ctl_system_shutdown (
    l4_cap_idx_t pfc,
    l4_umword_t reboot) [inline]
  
```

Shutdown or reboot the system.

**Parameters**

<i>pfc</i>	Capability selector for the platform-control object.
<i>reboot</i>	Shutdown when 0, or reboot when 1.

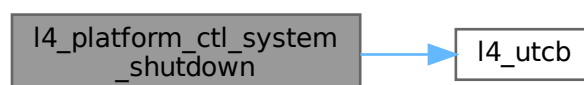
**Returns**

Syscall return tag

Definition at line 221 of file [platform\\_control.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.9.2.6 l4\_platform\_ctl\_system\_suspend()

```
l4_msgtag_t l4_platform_ctl_system_suspend (
    l4_cap_idx_t pfc,
    l4_umword_t extras) [inline]
```

Enter suspend to RAM.

##### Precondition

Must only be invoked on the boot CPU. Furthermore it must be ensured that the invoking thread is not migrated to a different CPU during the suspend.

##### Parameters

<i>pfc</i>	Capability selector for the platform-control object.
<i>extras</i>	Some extra platform-specific information needed to enter suspend to RAM. On x86 platforms and when using the Platform_control object provided by Fiasco, the value defines the sleep state. The sleep states are defined in the ACPI table. Other platforms as well as lo's Platform_control object don't make use of this value at the moment.

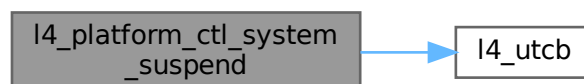
##### Returns

Syscall return tag

Definition at line 214 of file [platform\\_control.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.10 Scheduler

C interface of the Scheduler kernel object, see [L4::Scheduler](#) for the C++ interface.

Collaboration diagram for Scheduler:



## Data Structures

- struct `l4_sched_cpu_set_t`  
*CPU sets.*
- struct `l4_sched_param_t`  
*Scheduler parameter set.*

## Typedefs

- typedef struct `l4_sched_cpu_set_t` **`l4_sched_cpu_set_t`**  
*CPU sets.*
- typedef struct `l4_sched_param_t` **`l4_sched_param_t`**  
*Scheduler parameter set.*

## Enumerations

- enum `l4_scheduler_classes` { `L4_SCHEDULER_CLASS_FIXED_PRIO` = 1UL << 1 , `L4_SCHEDULER_CLASS_WFQ` = 1UL << 2 }  
*Supported scheduler classes.*
- enum `l4_scheduler_ops` { `L4_SCHEDULER_INFO_OP` = 0UL , `L4_SCHEDULER_RUN_THREAD_OP` = 1UL , `L4_SCHEDULER_IDLE_TIME_OP` = 2UL }  
*Operations on the Scheduler object.*

## Functions

- `l4_sched_cpu_set_t l4_sched_cpu_set` (`l4_umword_t` offset, unsigned char granularity, `l4_umword_t` map=1) `L4_NOTHROW`
- `l4_msgtag_t l4_scheduler_info` (`l4_cap_idx_t` scheduler, `l4_umword_t` \*cpu\_max, `l4_sched_cpu_set_t` \*cpus) `L4_NOTHROW`)  
*Get scheduler information.*
- `l4_msgtag_t l4_scheduler_info_with_classes` (`l4_cap_idx_t` scheduler, `l4_umword_t` \*cpu\_max, `l4_sched_cpu_set_t` \*cpus, `l4_umword_t` \*sched\_classes) `L4_NOTHROW`)  
*Get scheduler information.*
- `l4_sched_param_t l4_sched_param` (unsigned prio, `l4_umword_t` quantum=0) `L4_NOTHROW`  
*Construct scheduler parameter.*
- `l4_msgtag_t l4_scheduler_run_thread` (`l4_cap_idx_t` scheduler, `l4_cap_idx_t` thread, `l4_sched_param_t` const \*sp) `L4_NOTHROW`)  
*Run a thread on a Scheduler.*
- `l4_msgtag_t l4_scheduler_idle_time` (`l4_cap_idx_t` scheduler, `l4_sched_cpu_set_t` const \*cpus, `l4_kernel_clock_t` \*us) `L4_NOTHROW`)  
*Query the idle time (in  $\mu$ s) of a CPU.*
- int `l4_scheduler_is_online` (`l4_cap_idx_t` scheduler, `l4_umword_t` cpu) `L4_NOTHROW`  
*Query if a CPU is online.*

#### 14.1.10.10.1 Detailed Description

C interface of the Scheduler kernel object, see [L4::Scheduler](#) for the C++ interface.

The Scheduler interface allows a client to manage CPU resources. The API provides functions to query scheduler information, check the online state of CPUs, query CPU idle time and to start threads on defined CPU sets.

The scheduler offers a virtual device IRQ which triggers when the number of online cores changes, e.g. due to hotplug events. In contrast to hardware IRQs, this IRQ implements a limited functionality:

- Only IRQ line 0 is supported, no MSI vectors.
- The IRQ is edge-triggered and the IRQ mode cannot be changed.
- As the IRQ is edge-triggered, it does not have to be explicitly unmasked.

It depends on the platform, which hotplug events actually trigger the IRQ. Many platforms only support triggering the IRQ when a CPU core different from the boot CPU goes online.

##### Include File

```
#include <l4/sys/scheduler.h>
```

#### 14.1.10.10.2 Enumeration Type Documentation

##### 14.1.10.10.2.1 L4\_scheduler\_classes

```
enum L4_scheduler_classes
```

Supported scheduler classes.

##### Enumerator

L4_SCHEDULER_CLASS_FIXED_PRIO	Fixed-priority scheduler.
L4_SCHEDULER_CLASS_WFQ	Weighted fair queuing scheduler.

Definition at line 46 of file [scheduler.h](#).

##### 14.1.10.10.2.2 L4\_scheduler\_ops

```
enum L4_scheduler_ops
```

Operations on the Scheduler object.

##### Enumerator

L4_SCHEDULER_INFO_OP	Query infos about the scheduler.
----------------------	----------------------------------

<code>L4_SCHEDULER_RUN_THREAD_OP</code>	Run a thread on this scheduler.
<code>L4_SCHEDULER_IDLE_TIME_OP</code>	Query idle time for the scheduler.

Definition at line 269 of file [scheduler.h](#).

### 14.1.10.10.3 Function Documentation

#### 14.1.10.10.3.1 `l4_sched_cpu_set()`

```
l4_sched_cpu_set_t l4_sched_cpu_set (
    l4_umword_t offset,
    unsigned char granularity,
    l4_umword_t map = 1) [inline]
```

##### Parameters

<i>offset</i>	Offset. Must be a multiple of $2^{\text{granularity}}$ .
<i>granularity</i>	Granularity in log2 notation.
<i>map</i>	Bitmap of CPUs, defaults to 1 in C++.

##### Returns

CPU set.

##### Examples

[examples/sys/migrate/thread\\_migrate.cc](#).

Definition at line 279 of file [scheduler.h](#).

References [l4\\_sched\\_cpu\\_set\\_t::gran\\_offset](#), [L4\\_NOTHROW](#), and [l4\\_sched\\_cpu\\_set\\_t::map](#).

Referenced by [l4\\_sched\\_param\(\)](#).

Here is the caller graph for this function:



#### 14.1.10.10.3.2 `l4_sched_param()`

```
l4_sched_param_t l4_sched_param (
    unsigned prio,
    l4_umword_t quantum = 0) [inline]
```

Construct scheduler parameter.

##### Parameters

<i>prio</i>	Thread priority (1-255).
<i>quantum</i>	Timeslice in micro seconds.

The `l4_sched_param_t::affinity` of the returned value contains all CPUs.

#### Examples

`examples/sys/aliens/main.c`, `examples/sys/migrate/thread_migrate.cc`, `examples/sys/singlestep/main.c`, `examples/sys/start-with-exc/main.c`, and `examples/sys/utcb-ipc/main.c`.

Definition at line 289 of file `scheduler.h`.

References `l4_sched_param_t::affinity`, `L4_NOTHROW`, `l4_sched_cpu_set()`, `l4_sched_param_t::prio`, and `l4_sched_param_t::quantum`.

Here is the call graph for this function:



#### 14.1.10.10.3.3 l4\_scheduler\_idle\_time()

```

l4_msgtag_t l4_scheduler_idle_time (
    l4_cap_idx_t scheduler,
    l4_sched_cpu_set_t const * cpus,
    l4_kernel_clock_t * us) [inline]
  
```

Query the idle time (in  $\mu$ s) of a CPU.

#### Parameters

	<i>scheduler</i>	Scheduler object.
	<i>cpus</i>	Set of CPUs to query. Only the idle time of the first selected CPU in <code>cpus.map</code> is queried.
out	<i>us</i>	Idle time of queried CPU in $\mu$ s.

#### Return values

<i>0</i>	Success.
<i>-L4_EINVAL</i>	Invalid CPU requested in cpu set.

This function retrieves the idle time in  $\mu$ s of the first selected CPU in `cpus.map`. The idle time is the accumulated time a CPU has spent in the idle thread since its last reset. To calculate a load estimate  $l$  one has to retrieve the idle time at the beginning ( $i1$ ) and the end ( $i2$ ) of a known time interval  $t$ . The load is then calculated as  $l = 1 - (i2 - i1)/t$ .

The idle time is only defined for online CPUs. Reading the idle time from offline CPUs is undefined and may result in either getting `-L4_EINVAL` or calculating an estimated (incorrect) load of 1.

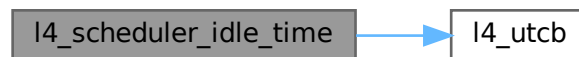
**Note**

The idle time statistics of remote CPUs is updated on context switch events only, hence may not be up-to-date when requested cross-CPU. To get up-to-date idle time you should use a thread running on the same CPU of which the idle time is requested.

Definition at line 403 of file [scheduler.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**14.1.10.10.3.4 l4\_scheduler\_info()**

```

l4_msgtag_t l4_scheduler_info (
    l4_cap_idx_t scheduler,
    l4_umword_t * cpu_max,
    l4_sched_cpu_set_t * cpus) [inline]
  
```

Get scheduler information.

**Parameters**

	<i>scheduler</i>	Scheduler object.
out	<i>cpu_max</i>	Maximum number of CPUs ever available. Optional, can be NULL.
in, out	<i>cpus</i>	<i>cpus.offset</i> is first CPU of interest. <i>cpus.granularity</i> (see <a href="#">l4_sched_cpu_set_t</a> ). <i>cpus.map</i> Bitmap of online CPUs. Must not be NULL.

**Return values**

0	Success.
-L4_ERANGE	The given CPU offset is larger than the maximum number of CPUs.

Definition at line 381 of file [scheduler.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.10.3.5 l4\_scheduler\_info\_with\_classes()

```

l4_msgtag_t l4_scheduler_info_with_classes (
    l4_cap_idx_t scheduler,
    l4_umword_t * cpu_max,
    l4_sched_cpu_set_t * cpus,
    l4_umword_t * sched_classes) [inline]
  
```

Get scheduler information.

#### Parameters

	<i>scheduler</i>	Scheduler object.
out	<i>cpu_max</i>	Maximum number of CPUs ever available. Optional, can be NULL.
in, out	<i>cpus</i>	<i>cpus.offset</i> is first CPU of interest. <i>cpus.granularity</i> (see <a href="#">l4_sched_cpu_set_t</a> ). <i>cpus.map</i> Bitmap of online CPUs. Must not be NULL.
out	<i>sched_classes</i>	A bitmap of available scheduling classes (see <a href="#">L4_scheduler_classes</a> ). Optional, can be NULL.

#### Return values

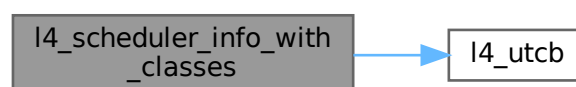
0	Success.
-L4_ERANGE	The given CPU offset is larger than the maximum number of CPUs.

This function delivers the same information as [l4\\_scheduler\\_info](#) plus the available scheduler classes (see [L4\\_scheduler\\_classes](#)).

Definition at line 388 of file [scheduler.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:





#### 14.1.10.10.3.6 l4\_scheduler\_is\_online()

```
int l4_scheduler_is_online (
    l4_cap_idx_t scheduler,
    l4_umword_t cpu) [inline]
```

Query if a CPU is online.

##### Parameters

<i>scheduler</i>	Scheduler object.
<i>cpu</i>	CPU number whose online status should be queried.

##### Return values

<i>true</i>	The CPU is online.
<i>false</i>	The CPU is offline

Definition at line 410 of file [scheduler.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.10.3.7 l4\_scheduler\_run\_thread()

```
l4_msgtag_t l4_scheduler_run_thread (
    l4_cap_idx_t scheduler,
    l4_cap_idx_t thread,
    l4_sched_param_t const * sp) [inline]
```

Run a thread on a Scheduler.

##### Parameters

<i>scheduler</i>	Scheduler object.
<i>thread</i>	Capability of the thread to run.
<i>sp</i>	Scheduling parameters.

##### Return values

0	Success.
-L4_EINVAL	Invalid size of the scheduling parameter.

This function launches a thread on a CPU determined by the scheduling parameter `sp.affinity`. A thread can be intentionally stopped by migrating it on an offline or an invalid CPU. The thread is only guaranteed to run if the CPU it is migrated to is currently online.

#### Note

If the target CPU is currently not online, there is no guarantee that the thread will ever run, even if the CPU comes online later on.

A scheduler may impose a policy with regard to selecting CPUs. However the scheduler is required to ensure the following two properties:

- Two threads with disjoint CPU sets must be scheduled to different CPUs.
- Two threads with identical CPU sets selecting only a single CPU must be scheduled to the same CPU.

#### Examples

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 396 of file [scheduler.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.11 Kernel-provided semaphore

C semaphore interface, see [L4::Semaphore](#) for the C++ interface.

Collaboration diagram for Kernel-provided semaphore:



## Functions

- [l4\\_msgtag\\_t l4\\_semaphore\\_up](#) ([l4\\_cap\\_idx\\_t](#) sem) [L4\\_NOTHROW](#)  
*Semaphore up operation (wrapper for trigger()).*
- [l4\\_msgtag\\_t l4\\_semaphore\\_down](#) ([l4\\_cap\\_idx\\_t](#) sem, [l4\\_timeout\\_t](#) timeout) [L4\\_NOTHROW](#)  
*Semaphore down operation.*

### 14.1.10.11.1 Detailed Description

C semaphore interface, see [L4::Semaphore](#) for the C++ interface.

#### Include File

```
#include <l4/sys/semaphore.h>
```

### 14.1.10.11.2 Function Documentation

#### 14.1.10.11.2.1 l4\_semaphore\_down()

```
l4\_msgtag\_t l4_semaphore_down (
    l4\_cap\_idx\_t sem,
    l4\_timeout\_t timeout) [inline]
```

Semaphore down operation.

#### Parameters

<i>sem</i>	Semaphore object.
<i>timeout</i>	Timeout for blocking the semaphore down operation. Note: The receive timeout of this timeout-pair is significant for blocking, the send part is usually non-blocking.

#### Returns

Syscall return tag. Use [l4\\_error\(\)](#) to check for errors.

#### Return values

<a href="#">-L4_EPERM</a>	Insufficient permissions; see precondition.
---------------------------	---------------------------------------------

**Precondition**

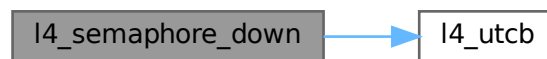
The capability `sem` must have the permission `L4_CAP_FPAGE_S`.

This method decrements the semaphore counter by one, or blocks if the counter is already zero, until either a timeout or cancel condition hits or the counter is increased by an `up()` operation.

Definition at line 100 of file `semaphore.h`.

References `L4_NOTHROW`, and `l4_utcb()`.

Here is the call graph for this function:

**14.1.10.11.2.2 l4\_semaphore\_up()**

```
l4_msgtag_t l4_semaphore_up (
    l4_cap_idx_t sem) [inline]
```

Semaphore up operation (wrapper for `trigger()`).

**Parameters**

<code>sem</code>	Semaphore object.
------------------	-------------------

**Returns**

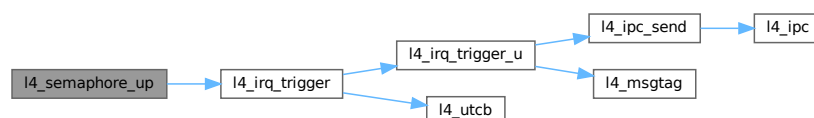
Send-only IPC message return tag. Use `l4_ipc_error()` to check for errors, do **not** use `l4_error()`.

Increases the semaphore counter by one if it is smaller than an unspecified limit. The unspecified limit is guaranteed to be at least  $2^{31}-1$ .

Definition at line 45 of file `semaphore.h`.

References `l4_irq_trigger()`, and `L4_NOTHROW`.

Here is the call graph for this function:



### 14.1.10.12 Task

C interface of the Task kernel object, see [L4::Task](#) for the C++ interface.

Collaboration diagram for Task:



### Enumerations

- enum [l4\\_unmap\\_flags\\_t](#) { [L4\\_FP\\_ALL\\_SPACES](#) , [L4\\_FP\\_DELETE\\_OBJ](#) , [L4\\_FP\\_OTHER\\_SPACES](#) }  
Flags for the unmap operation.

### Functions

- [l4\\_msgtag\\_t l4\\_task\\_vgicc\\_map](#) ([l4\\_cap\\_idx\\_t](#) task, [l4\\_fpage\\_t](#) vgicc\_fpage) [L4\\_NOTHROW](#)  
Map the GIC virtual CPU interface page to the task in case Fiasco detected a GIC version 2.
- [l4\\_msgtag\\_t l4\\_task\\_map](#) ([l4\\_cap\\_idx\\_t](#) dst\_task, [l4\\_cap\\_idx\\_t](#) src\_task, [l4\\_fpage\\_t](#) snd\_fpage, [l4\\_umword\\_t](#) snd\_base) [L4\\_NOTHROW](#)  
Map resources available in the source task to a destination task.
- [l4\\_msgtag\\_t l4\\_task\\_unmap](#) ([l4\\_cap\\_idx\\_t](#) task, [l4\\_fpage\\_t](#) fpage, [l4\\_umword\\_t](#) map\_mask) [L4\\_NOTHROW](#)  
Revoke rights from the task.
- [l4\\_msgtag\\_t l4\\_task\\_unmap\\_batch](#) ([l4\\_cap\\_idx\\_t](#) task, [l4\\_fpage\\_t](#) const \*fpages, unsigned num\_fpages, [l4\\_umword\\_t](#) map\_mask) [L4\\_NOTHROW](#)  
Revoke rights from a task.
- [l4\\_msgtag\\_t l4\\_task\\_delete\\_obj](#) ([l4\\_cap\\_idx\\_t](#) task, [l4\\_cap\\_idx\\_t](#) obj) [L4\\_NOTHROW](#)  
Release capability and delete object.
- [l4\\_msgtag\\_t l4\\_task\\_release\\_cap](#) ([l4\\_cap\\_idx\\_t](#) task, [l4\\_cap\\_idx\\_t](#) cap) [L4\\_NOTHROW](#)  
Release object capability.
- [l4\\_msgtag\\_t l4\\_task\\_cap\\_valid](#) ([l4\\_cap\\_idx\\_t](#) task, [l4\\_cap\\_idx\\_t](#) cap) [L4\\_NOTHROW](#)  
Check whether a capability is present (refers to an object).
- [l4\\_msgtag\\_t l4\\_task\\_cap\\_equal](#) ([l4\\_cap\\_idx\\_t](#) task, [l4\\_cap\\_idx\\_t](#) cap\_a, [l4\\_cap\\_idx\\_t](#) cap\_b) [L4\\_NOTHROW](#)  
Test whether two capabilities point to the same object with the same permissions (only considering selected permissions).
- [l4\\_msgtag\\_t l4\\_task\\_add\\_ku\\_mem](#) ([l4\\_cap\\_idx\\_t](#) task, [l4\\_fpage\\_t](#) \*ku\_mem) [L4\\_NOTHROW](#)  
Add kernel-user memory.

### 14.1.10.12.1 Detailed Description

C interface of the Task kernel object, see [L4::Task](#) for the C++ interface.

A task represents a combination of the address spaces provided by the [L4Re](#) micro kernel. A task consists of at least a memory address space and an object address space. On IA32 there is also an IO-port address space.

Task objects are created using the [Factory](#) interface.

### Include File

```
#include <l4/sys/task.h>
```

### 14.1.10.12.2 Enumeration Type Documentation

#### 14.1.10.12.2.1 l4\_unmap\_flags\_t

enum [l4\\_unmap\\_flags\\_t](#)

Flags for the unmap operation.

See also

[L4::Task::unmap\(\)](#) and [l4\\_task\\_unmap\(\)](#)

#### Enumerator

L4_FP_ALL_SPACES	<p>Flag to tell the unmap operation to revoke permissions from all child mappings including the mapping in the invoked task.</p> <p><b>Note</b></p> <p>Object capabilities are not hierarchical – they have no children. The result of the map operation on an object capability is a copy of that capability in the object space of the destination task. An unmap operation on object capabilities is a no-op if this flag is not specified.</p> <p><b>See also</b></p> <p><a href="#">L4::Task::unmap()</a> <a href="#">l4_task_unmap()</a></p>
L4_FP_DELETE_OBJ	<p>Flag that indicates that an unmap operation on object capabilities shall try to delete the corresponding objects immediately. This flag implies the <a href="#">L4_FP_ALL_SPACES</a> flag. The concept of deletion is only applicable to kernel objects. Therefore, for memory and I/O port capabilities, this flag has the same effect as <a href="#">L4_FP_ALL_SPACES</a> alone.</p> <p><b>See also</b></p> <p><a href="#">L4::Task::unmap()</a> <a href="#">l4_task_unmap()</a></p> <p><b>Note</b></p> <p>Specifying <a href="#">L4_FP_DELETE_OBJ</a> ^ <a href="#">L4_FP_ALL_SPACES</a> is treated as <a href="#">L4_FP_OTHER_SPACES</a>.</p>
L4_FP_OTHER_SPACES	<p>Counterpart to <a href="#">L4_FP_ALL_SPACES</a>; revoke permissions from child mappings only.</p> <p><b>See also</b></p> <p><a href="#">L4::Task::unmap()</a> <a href="#">l4_task_unmap()</a></p>

Definition at line 169 of file [consts.h](#).

### 14.1.10.12.3 Function Documentation

#### 14.1.10.12.3.1 l4\_task\_add\_ku\_mem()

```
l4_msgtag_t l4_task_add_ku_mem (
    l4_cap_idx_t task,
    l4_fpage_t * ku_mem) [inline]
```

Add kernel-user memory.

#### Parameters

	<i>task</i>	Capability selector of the task to add the memory to.
<i>in, out</i>	<i>ku_mem</i>	Flexpage describing the virtual area the memory goes to. On systems without MMU, the flexpage is adjusted to reflect the actually allocated physical address.

#### Returns

Syscall return tag

Kernel-user memory (*ku\_mem*) is memory that is shared between the kernel and user-space. It is needed for the UTCB area of threads (see [l4\\_thread\\_control\\_bind\(\)](#)) and for (extended) vCPU state. Note that existing kernel-user memory cannot be unmapped or mapped somewhere else.

#### Note

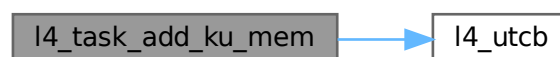
The amount of kernel-user memory that can be allocated at once is limited by the used kernel implementation. The minimum allocatable amount is one page ([L4\\_PAGESIZE](#)). A portable implementation should not depend on allocations greater than 16KiB to succeed.

This function is only guaranteed to work on [L4::Task](#) objects. It might or might not work on [L4::Vm](#) objects or on [L4Re::Dma\\_space](#) objects but there is no practical use for adding kernel-user memory to [L4::Vm](#) objects or to [L4Re::Dma\\_space](#) objects.

Definition at line 497 of file [task.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.12.3.2 l4\_task\_cap\_equal()

```
l4_msgtag_t l4_task_cap_equal (
    l4_cap_idx_t task,
    l4_cap_idx_t cap_a,
    l4_cap_idx_t cap_b) [inline]
```

Test whether two capabilities point to the same object with the same permissions (only considering selected permissions).

#### Parameters

---



<i>task</i>	Capability selector for the destination task to do the lookup in.
<i>cap↔ _a</i>	Capability selector for the first capability to compare.
<i>cap↔ _b</i>	Capability selector for the second capability to compare.

### Return values

<i>l4_msgtag_t::label()</i> = 1	The compared capabilities point to the same object with same considered permission.
<i>l4_msgtag_t::label()</i> = 0	The compared capabilities do <b>not</b> point to the same object or differ in the considered permission.

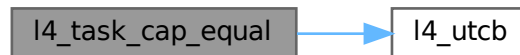
- For [L4::lpc\\_gate](#) objects, only the permissions [L4\\_CAP\\_FPAGE\\_W](#), [L4\\_CAP\\_FPAGE\\_S](#), and [L4\\_FPAGE\\_C\\_OBJ\\_RIGHT1](#) are considered for the comparison. Differences in other permissions are ignored.
- For other objects, only the permissions [L4\\_CAP\\_FPAGE\\_W](#) and [L4\\_CAP\\_FPAGE\\_S](#) are considered for the comparison. Differences in other permissions are ignored.

Note that having the [L4\\_CAP\\_FPAGE\\_R](#) permission is implicit in possessing the capability.

Definition at line 490 of file [task.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.12.3.3 l4\_task\_cap\_valid()

```

l4_msgtag_t l4_task_cap_valid (
    l4_cap_idx_t task,
    l4_cap_idx_t cap) [inline]
  
```

Check whether a capability is present (refers to an object).

### Parameters

<i>task</i>	Task to check the capability in.
<i>cap</i>	Valid capability to check for presence.

### Return values

<code>l4_msgtag_t::label() &gt; 0</code>	Capability is present (refers to an object).
<code>l4_msgtag_t::label() == 0</code>	No capability present (void object).

A capability is considered present when it refers to an existing kernel object.

#### Precondition

`cap` must be a valid capability index (i.e. not `L4_INVALID_CAP` or the like).

Definition at line 484 of file [task.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.12.3.4 l4\_task\_delete\_obj()

```

l4_msgtag_t l4_task_delete_obj (
    l4_cap_idx_t task,
    l4_cap_idx_t obj) [inline]
  
```

Release capability and delete object.

#### Parameters

<i>task</i>	Capability selector of destination task.
<i>obj</i>	Capability index of the object to delete.

#### Returns

Syscall return tag

If `obj` has the delete permission, initiates the deletion of the object. This implies that all capabilities for that object are gone afterwards. However, kernel-internally, objects are not destroyed until all other kernel objects holding a reference to it drop the reference. Hence, quota used by that object might not be freed immediately.

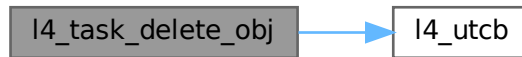
If `obj` does not have the delete permission, no error will be reported and only the capability `obj` is removed. (Note that, depending on the object's reference counter, this might still imply initiation of deletion.)

This operation is equivalent to [l4\\_task\\_unmap\(\)](#) with `L4_FP_DELETE_OBJ` flag.

Definition at line 463 of file [task.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.12.3.5 l4\_task\_map()

```

l4_msgtag_t l4_task_map (
    l4_cap_idx_t dst_task,
    l4_cap_idx_t src_task,
    l4_fpage_t snd_fpage,
    l4_umword_t snd_base) [inline]
  
```

Map resources available in the source task to a destination task.

##### Parameters

<i>dst_task</i>	Capability selector of the destination task.
<i>src_task</i>	Capability selector of the source task.
<i>snd_fpage</i>	Send flexpage that describes an area in the address space or object space of the source task.
<i>snd_base</i>	Send base that describes an offset in the receive window of the destination task. The lower bits contain additional map control flags (see <a href="#">l4_fpage_cacheability_opt_t</a> for memory mappings, <a href="#">L4_obj_fpage_ctl</a> for object mappings, and <a href="#">L4_MAP_ITEM_GRANT</a> ; also see <a href="#">l4_map_control()</a> and <a href="#">l4_map_obj_control()</a> ).

##### Returns

Syscall return tag. The function [l4\\_error\(\)](#) shall be used to test if the map operation was successful.

##### Return values

<i>L4_EOK</i>	Operation successful (but see notes below).
<i>-L4_EPERM</i>	Insufficient permissions; see precondition.
<i>-L4_EINVAL</i>	Invalid source task capability.
<i>-L4_IPC_SEMAPFAILED</i>	The map operation failed due to limited quota.

**Precondition**

The capability `dst_task` must have the permission [L4\\_CAP\\_FPAGE\\_W](#).

This method allows for asynchronous transfer of capabilities, memory mappings, and IO-port mappings (on IA32) from one task to another. The receive window is the whole address space of `dst_task`. By specifying proper rights in `snd_fpage` and `snd_base`, it is possible to remove rights during transfer.

**Note**

If the send flexpage is of type [L4\\_FPAGE\\_OBJ](#), the [L4\\_CAP\\_FPAGE\\_S](#) right is removed from the transferred capability unless both the source and destination task capabilities possess the [L4\\_CAP\\_FPAGE\\_S](#) right themselves.

Even with [l4\\_error\(\)](#) returning `L4_EOK` there might be cases where not all pages of the send flexpage were mapped respectively granted to the destination task, for instance, if the corresponding mapping in the destination task does already exist.

For more information on spaces and mappings, see [Spaces and Mappings](#). The flexpage API is described in more detail at [Flexpages](#).

**Note**

For peculiarities when using `grant`, see [L4\\_MAP\\_ITEM\\_GRANT](#).

Definition at line 433 of file [task.h](#).

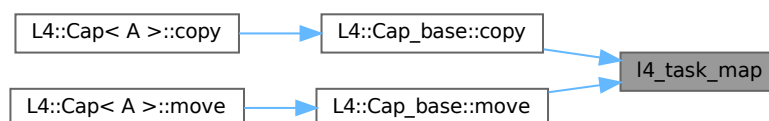
References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Referenced by [L4::Cap\\_base::copy\(\)](#), and [L4::Cap\\_base::move\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.1.10.12.3.6 l4\_task\_release\_cap()

```
l4_msgtag_t l4_task_release_cap (  
    l4_cap_idx_t task,  
    l4_cap_idx_t cap) [inline]
```

Release object capability.

#### Parameters

---

<i>task</i>	Capability selector of destination task
<i>cap</i>	Capability selector of object to release

#### Returns

Syscall return tag

This operation unmaps the capability from the specified task. This operation is equivalent to unmapping a single object capability by specifying all object rights as unmap mask.

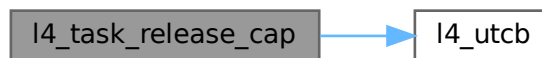
#### Note

If the reference counter of the kernel object referenced by `cap` goes down to zero, deletion of the object is initiated. Objects are not destroyed until all other kernel objects holding a reference to it drop the reference.

Definition at line 478 of file [task.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.12.3.7 l4\_task\_unmap()

```

l4_msgtag_t l4_task_unmap (
    l4_cap_idx_t task,
    l4_fpage_t fpage,
    l4_umword_t map_mask) [inline]
  
```

Revoke rights from the task.

#### Parameters

<i>task</i>	Capability selector of destination task
<i>fpage</i>	Flexpage that describes an area in one capability space of the destination task and the rights to revoke.
<i>map_mask</i>	Unmap mask, see <a href="#">l4_unmap_flags_t</a>

**Returns**

Syscall return tag

This method allows to revoke rights from the destination task. The rights to revoke are specified in the flexpage, see [l4\\_fpage\\_rights\(\)](#). For a flexpage describing IO ports or memory, it also revokes rights from all the tasks that got the rights delegated from the destination task (i.e., this operation does a recursive rights revocation). The capability is unmapped if certain rights are specified, see below for details. It is guaranteed that the rights revocation is completed before this function returns.

Note that this function cannot be used to revoke the reference counting permission (see [L4\\_FPAGE\\_C\\_REF\\_CNT](#)) or the IPC-gate server permission (see [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#)) from object capabilities.

It depends on the platform and the object type which rights need to be specified in the `rights` field of `fpage` to unmap a capability:

- An object capability is unmapped if and only if the [L4\\_CAP\\_FPAGE\\_R](#) right bit is set.
- An IO port is unmapped if and only if any right bit is set.
- Memory is unmapped if and only if the [L4\\_FPAGE\\_RO](#) right bit is set.

**Note**

Depending on the page-table features supported by the hardware, revocation of certain rights from a memory capability can be a no-op (i.e., the rights are not revoked). Further, revocation of certain rights may grant other rights which were not present before. For instance, on an architecture without support for NX, revoking X does nothing. For another example, revoking only X from an execute-only page grants read permission (because the mapping remains present in the page table).

If the reference counter of a kernel object referenced in `fpage` goes down to zero (as a result of deleting capabilities), the deletion of the object is initiated. Objects are not destroyed until all other kernel objects holding a reference to it drop the reference.

**Examples**

[examples/sys/utcb-ipc/main.c](#).

Definition at line [440](#) of file [task.h](#).

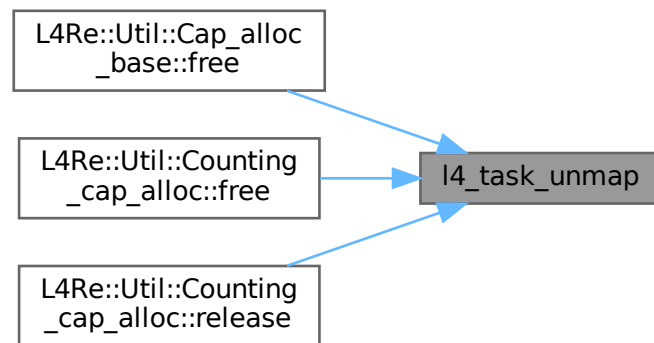
References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Referenced by [L4Re::Util::Cap\\_alloc\\_base::free\(\)](#), [L4Re::Util::Counting\\_cap\\_alloc< COUNTERTYPE, Dbg >::free\(\)](#), and [L4Re::Util::Counting\\_cap\\_alloc< COUNTERTYPE, Dbg >::release\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.1.10.12.3.8 l4\_task\_unmap\_batch()

```

l4_msgtag_t l4_task_unmap_batch (
    l4_cap_idx_t task,
    l4_fpage_t const * fpages,
    unsigned num_fpages,
    l4_umword_t map_mask) [inline]
  
```

Revoke rights from a task.

#### Parameters

<i>task</i>	Capability selector of destination task
<i>fpages</i>	An array of flexpages. Each item describes an area in one capability space of the destination task.
<i>num_fpages</i>	The size of the fpages array in elements (number of fpages sent).
<i>map_mask</i>	Unmap mask, see <a href="#">l4_unmap_flags_t</a>

#### Returns

Syscall return tag

Revoke rights specified in an array of flexpages, see [l4\\_task\\_unmap](#) for details.



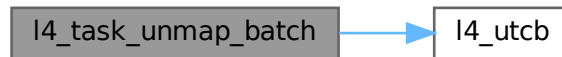
**Precondition**

The caller needs to take care that `num_fpages` is not bigger than `L4_UTCB_GENERIC_DATA_SIZE - 2`.

Definition at line 447 of file [task.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**14.1.10.12.3.9 l4\_task\_vgicc\_map()**

```

l4_msgtag_t l4_task_vgicc_map (
    l4_cap_idx_t task,
    l4_fpage_t vgicc_fpage) [inline]
  
```

Map the GIC virtual CPU interface page to the task in case Fiasco detected a GIC version 2.

**Parameters**

<i>task</i>	Capability selector of destination task
<i>vgicc_fpage</i>	Flexpage that describes an area in the address space of the destination task to map the vGICC page to

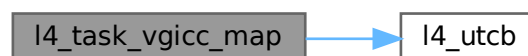
**Returns**

Syscall return tag

Definition at line 46 of file [\\_\\_task-arm.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

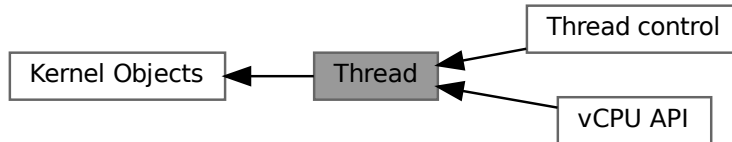
Here is the call graph for this function:



### 14.1.10.13 Thread

C Thread object interface, see [L4::Thread](#) for the C++ interface.

Collaboration diagram for Thread:



### Topics

- [Thread control](#) . . . . . 409  
API for Thread Control method.
- [vCPU API](#) . . . . . 415  
vCPU API.

### Enumerations

- enum [L4\\_thread\\_control\\_flags](#) { [L4\\_THREAD\\_CONTROL\\_SET\\_PAGER](#) = 0x0010000 , [L4\\_THREAD\\_CONTROL\\_BIND\\_TASK](#) = 0x0200000 , [L4\\_THREAD\\_CONTROL\\_ALIEN](#) = 0x0400000 , [L4\\_THREAD\\_CONTROL\\_SET\\_EXC\\_HANDLER](#) = 0x1000000 }
- Flags for the thread control operation.
- enum [L4\\_thread\\_control\\_mr\\_indices](#) { [L4\\_THREAD\\_CONTROL\\_MR\\_IDX\\_FLAGS](#) = 0 , [L4\\_THREAD\\_CONTROL\\_MR\\_IDX\\_PAGER](#) = 1 , [L4\\_THREAD\\_CONTROL\\_MR\\_IDX\\_EXC\\_HANDLER](#) = 2 , [L4\\_THREAD\\_CONTROL\\_MR\\_IDX\\_FLAG\\_VALS](#) = 4 , [L4\\_THREAD\\_CONTROL\\_MR\\_IDX\\_BIND\\_UTCB](#) = 5 , [L4\\_THREAD\\_CONTROL\\_MR\\_IDX\\_BIND\\_TASK](#) = 6 }
- Indices for the values in the message register for thread control.
- enum [L4\\_thread\\_ex\\_regs\\_flags](#) { [L4\\_THREAD\\_EX\\_REGS\\_CANCEL](#) = 0x10000UL , [L4\\_THREAD\\_EX\\_REGS\\_TRIGGER\\_EXC](#) = 0x20000UL , [L4\\_THREAD\\_EX\\_REGS\\_ARCH\\_MASK](#) = 0xff000000UL }
- Flags for the thread ex-regs operation.
- enum [L4\\_thread\\_ex\\_regs\\_flags\\_arm](#) { [L4\\_THREAD\\_EX\\_REGS\\_ARM\\_SET\\_EL\\_MASK](#) = 0x3 << 24 , [L4\\_THREAD\\_EX\\_REGS\\_ARM\\_SET\\_EL\\_KEEP](#) = 0x0 << 24 , [L4\\_THREAD\\_EX\\_REGS\\_ARM\\_SET\\_EL\\_EL0](#) = 0x1 << 24 , [L4\\_THREAD\\_EX\\_REGS\\_ARM\\_SET\\_EL\\_EL1](#) = 0x2 << 24 }
- Arm specific [L4::Thread::ex\\_regs\(\)](#) flags.
- enum [L4\\_thread\\_ex\\_regs\\_flags\\_arm64](#) { [L4\\_THREAD\\_EX\\_REGS\\_ARM64\\_SET\\_EL\\_MASK](#) = 0x3 << 24 , [L4\\_THREAD\\_EX\\_REGS\\_ARM64\\_SET\\_EL\\_KEEP](#) = 0x0 << 24 , [L4\\_THREAD\\_EX\\_REGS\\_ARM64\\_SET\\_EL\\_EL0](#) = 0x1 << 24 , [L4\\_THREAD\\_EX\\_REGS\\_ARM64\\_SET\\_EL\\_EL1](#) = 0x2 << 24 }
- Arm64 specific [L4::Thread::ex\\_regs\(\)](#) flags.

## Functions

- `l4_msgtag_t l4_thread_ex_regs (l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp, l4_umword_t flags) L4_NOTHROW`  
*Exchange basic thread registers.*
- `l4_msgtag_t l4_thread_ex_regs_u (l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp, l4_umword_t flags, l4_utcb_t *utcb) L4_NOTHROW`  
*Exchange basic thread registers.*
- `l4_msgtag_t l4_thread_ex_regs_ret (l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp, l4_umword_t *flags) L4_NOTHROW`  
*Exchange basic thread registers and return previous values.*
- `l4_msgtag_t l4_thread_ex_regs_ret_u (l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp, l4_umword_t *flags, l4_utcb_t *utcb) L4_NOTHROW`  
*Exchange basic thread registers and return previous values.*
- `l4_msgtag_t l4_thread_yield (void) L4_NOTHROW`  
*Yield current time slice.*
- `l4_msgtag_t l4_thread_switch (l4_cap_idx_t to_thread) L4_NOTHROW`  
*Switch to another thread (and donate the remaining time slice).*
- `l4_msgtag_t l4_thread_stats_time (l4_cap_idx_t thread, l4_kernel_clock_t *us) L4_NOTHROW`  
*Get consumed time of thread in  $\mu$ s.*
- `l4_msgtag_t l4_thread_vcpu_resume_start (void) L4_NOTHROW`  
*vCPU return from event handler.*
- `l4_msgtag_t l4_thread_vcpu_resume_commit (l4_cap_idx_t thread, l4_msgtag_t tag) L4_NOTHROW`  
*Commit vCPU resume.*
- `l4_msgtag_t l4_thread_vcpu_control (l4_cap_idx_t thread, l4_addr_t vcpu_state) L4_NOTHROW`  
*Enable the vCPU feature for the thread.*
- `l4_msgtag_t l4_thread_vcpu_control_u (l4_cap_idx_t thread, l4_addr_t vcpu_state, l4_utcb_t *utcb) L4_NOTHROW`  
*Enable the vCPU feature for the thread.*
- `l4_msgtag_t l4_thread_vcpu_control_ext (l4_cap_idx_t thread, l4_addr_t ext_vcpu_state) L4_NOTHROW`  
*Enable the extended vCPU feature for the thread.*
- `l4_msgtag_t l4_thread_vcpu_control_ext_u (l4_cap_idx_t thread, l4_addr_t ext_vcpu_state, l4_utcb_t *utcb) L4_NOTHROW`  
*Enable the extended vCPU feature for the thread.*
- `l4_msgtag_t l4_thread_register_del_irq (l4_cap_idx_t thread, l4_cap_idx_t irq) L4_NOTHROW`  
*Register an IRQ that will trigger upon deletion events.*
- `l4_msgtag_t l4_thread_modify_sender_start (void) L4_NOTHROW`  
*Start a thread sender modification sequence.*
- `int l4_thread_modify_sender_add (l4_umword_t match_mask, l4_umword_t match, l4_umword_t del_bits, l4_umword_t add_bits, l4_msgtag_t *tag) L4_NOTHROW`  
*Add a modification pattern to a sender modification sequence.*
- `l4_msgtag_t l4_thread_modify_sender_commit (l4_cap_idx_t thread, l4_msgtag_t tag) L4_NOTHROW`  
*Apply (commit) a sender modification sequence.*
- `l4_msgtag_t l4_thread_register_doorbell_irq (l4_cap_idx_t thread, l4_cap_idx_t irq) L4_NOTHROW`  
*Register an IRQ that will trigger when a forwarded virtual interrupt is pending.*
- `l4_msgtag_t l4_thread_arm_set_tpidruro (l4_cap_idx_t thread, l4_addr_t tpidruro) L4_NOTHROW`  
*Set the TPIDRURO thread specific register.*

#### 14.1.10.13.1 Detailed Description

C Thread object interface, see [L4::Thread](#) for the C++ interface.

An [L4](#) thread is a thread of execution in the [L4](#) context. Usually user-level and kernel threads are mapped 1:1 to each other. Thread kernel objects are created using a factory, see [Factory](#) ([l4\\_factory\\_create\\_thread\(\)](#)).

Amongst other things an [L4](#) thread encapsulates:

- CPU state
  - General-purpose registers
  - Program counter
  - Stack pointer
- FPU state
- Scheduling parameters, see the [Scheduler](#) API
- Execution state
  - Blocked, Runnable, Running

Thread objects provide an API for

- Thread configuration and manipulation
- Thread switching.

The thread control functions are used to control various aspects of a thread. See [l4\\_thread\\_control\\_start\(\)](#) for more information.

On ARM newly created threads run in EL0 by default and the exception level can be changed there with [ex\\_regs\(\)](#).

#### Include File

```
#include <l4/sys/thread.h>
```

For the C++ interface refer to [L4::Thread](#).

#### 14.1.10.13.2 Enumeration Type Documentation

##### 14.1.10.13.2.1 L4\_thread\_control\_flags

```
enum L4_thread_control_flags
```

Flags for the thread control operation.

#### Enumerator

L4_THREAD_CONTROL_SET_PAGER	The pager will be given.
-----------------------------	--------------------------

L4_THREAD_CONTROL_BIND_TASK	The task to bind the thread to will be given.
L4_THREAD_CONTROL_ALIEN	Alien state of the thread is set.
L4_THREAD_CONTROL_SET_EXC_HANDLER	The exception handler of the thread will be given.

Definition at line 761 of file [thread.h](#).

#### 14.1.10.13.2.2 L4\_thread\_control\_mr\_indices

enum [L4\\_thread\\_control\\_mr\\_indices](#)

Indices for the values in the message register for thread control.

##### Enumerator

L4_THREAD_CONTROL_MR_IDX_FLAGS	See also <a href="#">L4_thread_control_flags</a> .
L4_THREAD_CONTROL_MR_IDX_PAGER	Index for pager cap.
L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER	Index for exception handler.
L4_THREAD_CONTROL_MR_IDX_FLAG_VALS	Index for feature values.
L4_THREAD_CONTROL_MR_IDX_BIND_UTCB	Index for UTCB address for bind.
L4_THREAD_CONTROL_MR_IDX_BIND_TASK	Index for task flexpage for bind.

Definition at line 782 of file [thread.h](#).

#### 14.1.10.13.2.3 L4\_thread\_ex\_regs\_flags

enum [L4\\_thread\\_ex\\_regs\\_flags](#)

Flags for the thread ex-regs operation.

##### Enumerator

L4_THREAD_EX_REGS_CANCEL	Cancel ongoing IPC in the thread.
L4_THREAD_EX_REGS_TRIGGER_EXCEPTION	Trigger artificial exception in thread.
L4_THREAD_EX_REGS_ARCH_MASK	Arch specific flags.

Definition at line 797 of file [thread.h](#).

#### 14.1.10.13.2.4 L4\_thread\_ex\_regs\_flags\_arm

enum [L4\\_thread\\_ex\\_regs\\_flags\\_arm](#)

Arm specific [L4::Thread::ex\\_regs\(\)](#) flags.

Only one option must be used in calls to [L4::Thread::ex\\_regs\(\)](#). Using more than one option results in undefined behaviour.

##### Enumerator

L4_THREAD_EX_REGS_ARM_SET_EL_MASK	Exception level set mask.
L4_THREAD_EX_REGS_ARM_SET_EL_KEEP	Keep current exception level of thread (default).
L4_THREAD_EX_REGS_ARM_SET_EL_EL0	Set exception level of thread to EL0 (usr mode).
L4_THREAD_EX_REGS_ARM_SET_EL_EL1	Set exception level of thread to EL1 (sys mode).

Definition at line 39 of file [thread.h](#).

#### 14.1.10.13.2.5 L4\_thread\_ex\_regs\_flags\_arm64

```
enum L4_thread_ex_regs_flags_arm64
```

Arm64 specific [L4::Thread::ex\\_regs\(\)](#) flags.

Only one option must be used in calls to [L4::Thread::ex\\_regs\(\)](#). Using more than one option results in undefined behaviour.

##### Enumerator

L4_THREAD_EX_REGS_ARM64_SET_EL_MASK	Exception level set mask.
L4_THREAD_EX_REGS_ARM64_SET_EL_KEEP	Keep current exception level of thread (default).
L4_THREAD_EX_REGS_ARM64_SET_EL_EL0	Set exception level of thread to EL0.
L4_THREAD_EX_REGS_ARM64_SET_EL_EL1	Set exception level of thread to EL1t.

Definition at line 46 of file [thread.h](#).

#### 14.1.10.13.3 Function Documentation

##### 14.1.10.13.3.1 l4\_thread\_arm\_set\_tpidruro()

```
l4_msgtag_t l4_thread_arm_set_tpidruro (
    l4_cap_idx_t thread,
    l4_addr_t tpidruro) [inline]
```

Set the TPIDRURO thread specific register.

##### Parameters

<i>thread</i>	Thread to manipulate
<i>tpidruro</i>	The value to be set

##### Returns

System call return tag

**Note**

When this function is invoked for a thread currently executing on a different core, then the changed register content will not be visible to that thread until a thread switch happens on that core.

Definition at line 72 of file [thread.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**14.1.10.13.3.2 l4\_thread\_ex\_regs()**

```

l4_msgtag_t l4_thread_ex_regs (
    l4_cap_idx_t thread,
    l4_addr_t ip,
    l4_addr_t sp,
    l4_umword_t flags) [inline]
  
```

Exchange basic thread registers.

**Parameters**

<i>thread</i>	Capability selector of the thread to manipulate.
<i>ip</i>	New instruction pointer, use ~0UL to leave the instruction pointer unchanged.
<i>sp</i>	New stack pointer, use ~0UL to leave the stack pointer unchanged.
<i>flags</i>	Ex-regs flags, see <a href="#">L4_thread_ex_regs_flags</a> .

**Returns**

System call return tag

This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see [flags](#)). If the thread is in an IPC operation or if [L4\\_THREAD\\_EX\\_REGS\\_TRIGGER\\_EXCEPTION](#) forces an IPC then changes in IP and SP take effect directly after returning from this IPC. On ARM this method allows to change the exception level, see [L4\\_thread\\_ex\\_regs\\_flags\\_arm](#) and [L4\\_thread\\_ex\\_regs\\_flags\\_arm64](#).

The thread is started using [l4\\_scheduler\\_run\\_thread\(\)](#). However, if at the time [l4\\_scheduler\\_run\\_thread\(\)](#) is called, the instruction pointer of the thread is invalid, a later call to [l4\\_thread\\_ex\\_regs\(\)](#) with a valid instruction pointer might start the thread.

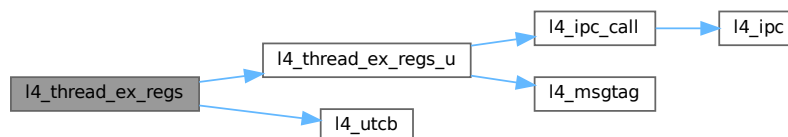
## Examples

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 942 of file [thread.h](#).

References [L4\\_NOTHROW](#), [l4\\_thread\\_ex\\_regs\\_u\(\)](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 14.1.10.13.3.3 l4\_thread\_ex\_regs\_ret()

```

l4_msgtag_t l4_thread_ex_regs_ret (
    l4_cap_idx_t thread,
    l4_addr_t * ip,
    l4_addr_t * sp,
    l4_umword_t * flags) [inline]
  
```

Exchange basic thread registers and return previous values.

## Parameters

	<i>thread</i>	Capability selector of the thread to manipulate.
in, out	<i>ip</i>	New instruction pointer, use ~0UL to leave the instruction pointer unchanged, return previous instruction pointer.
in, out	<i>sp</i>	New stack pointer, use ~0UL to leave the stack pointer unchanged, returns previous stack pointer.
in, out	<i>flags</i>	Ex-regs flags, see <a href="#">L4_thread_ex_regs_flags</a> , return previous CPU flags of the thread.

## Returns

System call return tag

This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see `flags`). If the thread is in an IPC operation or if [L4\\_THREAD\\_EX\\_REGS\\_TRIGGER\\_EXCEPTION](#) forces an IPC then changes in IP and SP take effect directly after returning from this IPC. On ARM this method allows to change the exception level, see [L4\\_thread\\_ex\\_regs\\_flags\\_arm](#) and [L4\\_thread\\_ex\\_regs\\_flags\\_arm64](#).



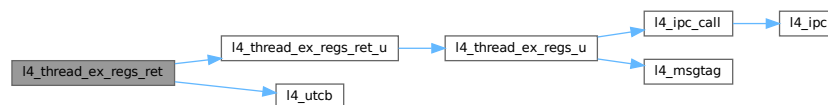
The thread is started using `l4_scheduler_run_thread()`. However, if at the time `l4_scheduler_run_thread()` is called, the instruction pointer of the thread is invalid, a later call to `l4_thread_ex_regs()` with a valid instruction pointer might start the thread.

Returned values are valid only if function returns successfully.

Definition at line 949 of file `thread.h`.

References `L4_NOTHROW`, `l4_thread_ex_regs_ret_u()`, and `l4_utcb()`.

Here is the call graph for this function:



#### 14.1.10.13.3.4 l4\_thread\_ex\_regs\_ret\_u()

```

l4_msgtag_t l4_thread_ex_regs_ret_u (
    l4_cap_idx_t thread,
    l4_addr_t * ip,
    l4_addr_t * sp,
    l4_umword_t * flags,
    l4_utcb_t * utcb) [inline]
  
```

Exchange basic thread registers and return previous values.

#### Parameters

	<i>thread</i>	Capability selector of the thread to manipulate.
in, out	<i>ip</i>	New instruction pointer, use <code>~0UL</code> to leave the instruction pointer unchanged, return previous instruction pointer.
in, out	<i>sp</i>	New stack pointer, use <code>~0UL</code> to leave the stack pointer unchanged, returns previous stack pointer.
in, out	<i>flags</i>	Ex-regs flags, see <code>L4_thread_ex_regs_flags</code> , return previous CPU flags of the thread.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <code>l4_utcb</code> .

#### Returns

System call return tag. [out] parameters are only valid if the function returns successfully. Use `l4_error()` to check.

This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see `flags`). If the thread is in an IPC operation or if `L4_THREAD_EX_REGS_TRIGGER_EXCEPTION` forces an IPC then changes in IP and SP take effect directly after returning from this IPC. On ARM this method allows to change the exception level, see `L4_thread_ex_regs_flags_arm` and `L4_thread_ex_regs_flags_arm64`.

The thread is started using `L4::Scheduler::run_thread()`. However, if at the time `L4::Scheduler::run_thread()` is called, the instruction pointer of the thread is invalid, a later call to `ex_regs()` with a valid instruction pointer might start the thread.

Definition at line 823 of file `thread.h`.

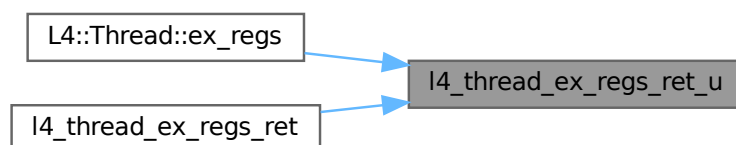
References `L4_NOTHROW`, `l4_thread_ex_regs_u()`, and `l4_msg_regs_t::mr`.

Referenced by `L4::Thread::ex_regs()`, and `l4_thread_ex_regs_ret()`.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.1.10.13.3.5 l4\_thread\_ex\_regs\_u()

```

l4_msgtag_t l4_thread_ex_regs_u (
    l4_cap_idx_t thread,
    l4_addr_t ip,
    l4_addr_t sp,
    l4_umword_t flags,
    l4_utcb_t * utcb) [inline]
  
```

Exchange basic thread registers.

#### Parameters

<i>thread</i>	Capability selector of the thread to manipulate.
<i>ip</i>	New instruction pointer, use <code>~0UL</code> to leave the instruction pointer unchanged.
<i>sp</i>	New stack pointer, use <code>~0UL</code> to leave the stack pointer unchanged.
<i>flags</i>	Ex-regs flags, see <a href="#">L4_thread_ex_regs_flags</a> .
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

### Returns

System call return tag.

This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see `flags`). If the thread is in an IPC operation or if [L4\\_THREAD\\_EX\\_REGS\\_TRIGGER\\_EXCEPTION](#) forces an IPC then changes in IP and SP take effect directly after returning from this IPC. On ARM this method allows to change the exception level, see [L4\\_thread\\_ex\\_regs\\_flags\\_arm](#) and [L4\\_thread\\_ex\\_regs\\_flags\\_arm64](#).

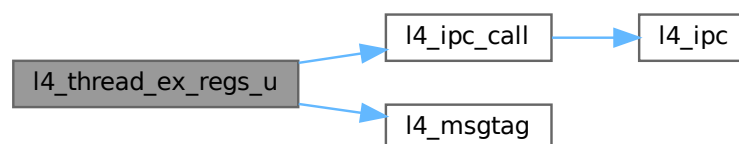
The thread is started using [L4::Scheduler::run\\_thread\(\)](#). However, if at the time [L4::Scheduler::run\\_thread\(\)](#) is called, the instruction pointer of the thread is invalid, a later call to `ex_regs()` with a valid instruction pointer might start the thread.

Definition at line 812 of file [thread.h](#).

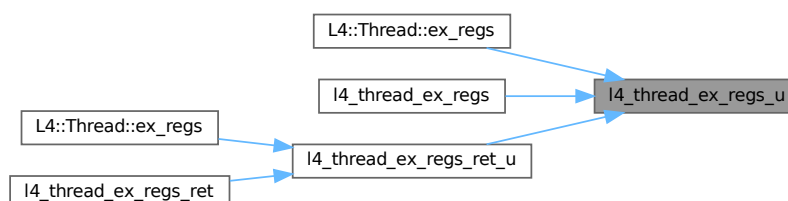
References [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_NOTHROW](#), [L4\\_PROTO\\_THREAD](#), [L4\\_THREAD\\_EX\\_REGS\\_OP](#), and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [L4::Thread::ex\\_regs\(\)](#), [l4\\_thread\\_ex\\_regs\(\)](#), and [l4\\_thread\\_ex\\_regs\\_ret\\_u\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.1.10.13.3.6 `l4_thread_modify_sender_add()`

```
int l4_thread_modify_sender_add (
    l4_umword_t match_mask,
    l4_umword_t match,
    l4_umword_t del_bits,
    l4_umword_t add_bits,
    l4_msgtag_t * tag) [inline]
```

Add a modification pattern to a sender modification sequence.

##### Parameters

<i>tag</i>	Tag received from <a href="#">l4_thread_modify_sender_start()</a> or previous <a href="#">l4_thread_modify_sender_add()</a> calls from the same sequence.
<i>match_mask</i>	Bitmask of bits to match the label.
<i>match</i>	Bitmask that must be equal to the label after applying <i>match_mask</i> .
<i>del_bits</i>	Bits to be deleted from the label.
<i>add_bits</i>	Bits to be added to the label.

##### Returns

0 on success, <0 on error

In pseudo code: if ((sender\_label & match\_mask) == match) { sender\_label = (sender\_label & ~del\_bits) | add\_bits; }

Only the first match is applied.

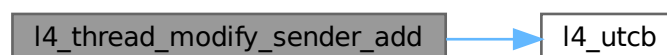
##### See also

[l4\\_thread\\_modify\\_sender\\_start](#)  
[l4\\_thread\\_modify\\_sender\\_commit](#)

Definition at line 1116 of file [thread.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.13.3.7 `l4_thread_modify_sender_commit()`

```
l4_msgtag_t l4_thread_modify_sender_commit (
    l4_cap_idx_t thread,
    l4_msgtag_t tag) [inline]
```

Apply (commit) a sender modification sequence.

The modification rules are applied to all IPCs to the thread (whether directly or by IPC gate) that are already in flight, that is that the sender is already blocking on.

##### Note

Modifying the senders of a thread running on a different CPU core is not supported.

To ensure that no in-flight senders are missed, either the thread itself must execute `modify_senders`, or the thread executing the `modify_senders` must synchronize with the target thread. This synchronization must ensure the following:

1. Before `modify_senders` is executed the target thread must execute at least shortly (so that pending DRQs are handled).
2. The target thread must pause its IPC dispatch, until `modify_senders` is completed. In other words, the target thread must not be receive ready, because otherwise an IPC message with an unmodified label can be transferred to its UTCB or vCPU state.

##### See also

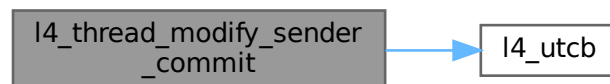
[l4\\_thread\\_modify\\_sender\\_start](#)

[l4\\_thread\\_modify\\_sender\\_add](#)

Definition at line 1127 of file [thread.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.13.3.8 l4\_thread\_modify\_sender\_start()

```
l4_msgtag_t l4_thread_modify_sender_start (
    void ) [inline]
```

Start a thread sender modification sequence.

Add modification rules with [l4\\_thread\\_modify\\_sender\\_add\(\)](#) and commit with [l4\\_thread\\_modify\\_sender\\_commit\(\)](#). Do not touch the UTCB between [l4\\_thread\\_modify\\_sender\\_start\(\)](#) and [l4\\_thread\\_modify\\_sender\\_commit\(\)](#).

This mechanism shall be used to change the source object labels of every pending IPC of an IPC gate or an IRQ if the labels in such pending IPC become invalid for the receiving thread, potentially because:

- an IPC gate / IRQ was unbound from a thread, or
- an IPC gate / IRQ was removed, or
- the label of an IPC gate /IRQ bound to a thread was changed.

It is not required to perform the modify\_sender mechanism after an IPC gate or an IRQ was bound to a thread for the first time.

See also

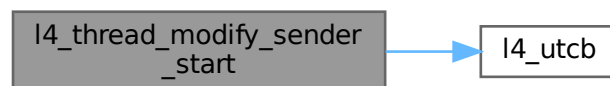
[l4\\_thread\\_modify\\_sender\\_add](#)

[l4\\_thread\\_modify\\_sender\\_commit](#)

Definition at line 1110 of file [thread.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.13.3.9 l4\_thread\_register\_del\_irq()

```
l4_msgtag_t l4_thread_register_del_irq (
    l4_cap_idx_t thread,
    l4_cap_idx_t irq) [inline]
```

Register an IRQ that will trigger upon deletion events.

##### Parameters

---

<i>thread</i>	Thread to register IRQ for.
<i>irq</i>	Capability selector for the IRQ object to be triggered.

### Returns

System call return tag containing the return code.

### Return values

<code>-L4_EPERM</code>	Insufficient permissions; see precondition.
------------------------	---------------------------------------------

### Precondition

The capability `irq` must have the permission `L4_CAP_FPAGE_W`.

In case the `irq` is already bound to an interrupt source, it is unbound first. When `irq` is deleted, it will be deregistered first. A registered deletion `Irq` can only be deregistered by deleting the `Irq` or the thread.

List of deletion events:

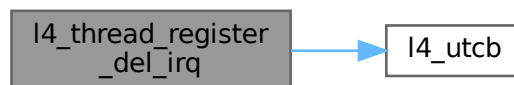
- deletion of one or several IPC gates bound to this thread.

When the deletion event is delivered, there is no indication about which IPC gate was deleted.

Definition at line 1037 of file `thread.h`.

References `L4_NOTHROW`, and `l4_utcb()`.

Here is the call graph for this function:



#### 14.1.10.13.3.10 l4\_thread\_register\_doorbell\_irq()

```

l4_msgtag_t l4_thread_register_doorbell_irq (
    l4_cap_idx_t thread,
    l4_cap_idx_t irq) [inline]
  
```

Register an IRQ that will trigger when a forwarded virtual interrupt is pending.

### Parameters

<i>thread</i>	Thread to register IRQ for.
<i>irq</i>	Capability selector for the IRQ object to be triggered.

### Returns

System call return tag containing the return code.

### Return values

<code>-L4_EPERM</code>	Insufficient permissions; see precondition.
------------------------	---------------------------------------------

### Precondition

The capability `irq` must have the permission [L4\\_CAP\\_FPAGE\\_W](#).

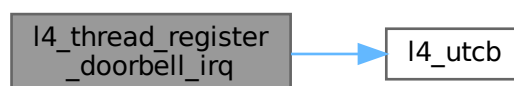
See [l4\\_irq\\_bind\\_vcpu\(\)](#) for more details about how interrupts can be forwarded directly by the kernel to extended vCPU user mode.

In case the `irq` is already bound to an interrupt source, it is unbound first. When `irq` is deleted, it will be deregistered first. A registered deletion `l4_irq` can only be deregistered by deleting the `l4_irq` or the thread.

Definition at line 1146 of file [thread.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.13.3.11 l4\_thread\_stats\_time()

```

l4_msgtag_t l4_thread_stats_time (
    l4_cap_idx_t thread,
    l4_kernel_clock_t * us) [inline]
  
```

Get consumed time of thread in  $\mu$ s.

### Parameters



	<i>thread</i>	Thread to get the consumed time from.
out	<i>us</i>	Consumed time in $\mu$ s.

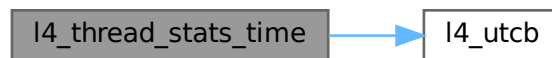
#### Returns

system call return tag

Definition at line 1005 of file [thread.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.13.3.12 l4\_thread\_switch()

```
l4_msgtag_t l4_thread_switch (  
    l4_cap_idx_t to_thread) [inline]
```

Switch to another thread (and donate the remaining time slice).

#### Parameters

<i>to_thread</i>	The thread to switch to.
------------------	--------------------------

#### Returns

system call return tag

Definition at line 996 of file [thread.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.13.3.13 `l4_thread_vcpu_control()`

```
l4_msgtag_t l4_thread_vcpu_control (
    l4_cap_idx_t thread,
    l4_addr_t vcpu_state) [inline]
```

Enable the vCPU feature for the thread.

##### Parameters

<i>thread</i>	Capability selector of the thread for which the vCPU feature shall be enabled.
<i>vcpu_state</i>	The virtual address where the kernel shall store the vCPU state in case of vCPU exits. The address must be a valid kernel-user-memory address (see <a href="#">l4_task_add_ku_mem()</a> ).

##### Returns

Syscall return tag.

This function enables the vCPU feature of the `thread`.

The kernel-user memory area starting at `vcpu_state` must be at least 128-byte aligned and must cover the size of [l4\\_vcpu\\_state\\_t](#).

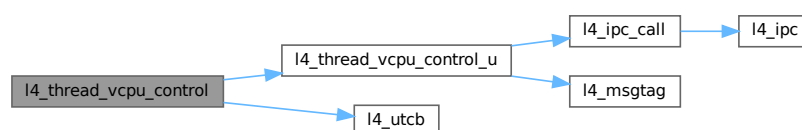
##### Note

Disabling of the vCPU feature is optional and currently not supported.

Definition at line 1054 of file [thread.h](#).

References [L4\\_NOTHROW](#), [l4\\_thread\\_vcpu\\_control\\_u\(\)](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.13.3.14 `l4_thread_vcpu_control_ext()`

```
l4_msgtag_t l4_thread_vcpu_control_ext (
    l4_cap_idx_t thread,
    l4_addr_t ext_vcpu_state) [inline]
```

Enable the extended vCPU feature for the thread.

##### Parameters

<i>thread</i>	Capability selector of the thread for which the extended vCPU feature shall be enabled.
<i>ext_vcpu_state</i>	The virtual address where the kernel shall store the vCPU state in case of vCPU exits. The address must be a valid kernel-user-memory address (see <a href="#">l4_task_add_ku_mem()</a> ).

### Returns

Syscall return tag.

The extended vCPU feature allows the use of hardware-virtualization features such as Intel's VT or AMD's SVM.

This function enables the extended vCPU feature of the `thread`. Enabling the extended vCPU feature also enables the vCPU feature.

The kernel-user memory area starting at `ext_vcpu_state` must be at least 4 KiB aligned and must cover a size of `L4_PAGESIZE`. It includes the data of [l4\\_vcpu\\_state\\_t](#) at offset 0, the extended vCPU state at offset `L4_VCPU_OFFSET_EXT_STATE`, and, on some platforms, the extended vCPU information at offset `L4_VCPU_OFFSET_EXT_INFOS`.

### Note

Enabling the extended vCPU feature for a thread running on a different CPU core is currently not supported.

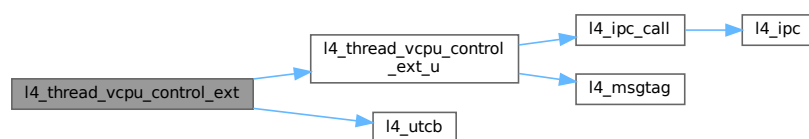
Disabling of the extended vCPU feature is currently not supported.

Upgrading from non-extended vCPU feature to extended vCPU feature is currently not supported.

Definition at line 1069 of file [thread.h](#).

References [L4\\_NOTHROW](#), [l4\\_thread\\_vcpu\\_control\\_ext\\_u\(\)](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.13.3.15 l4\_thread\_vcpu\_control\_ext\_u()

```

l4_msgtag_t l4_thread_vcpu_control_ext_u (
    l4_cap_idx_t thread,
    l4_addr_t ext_vcpu_state,
    l4_utcb_t * utcb) [inline]

```

Enable the extended vCPU feature for the thread.

### Parameters

<i>thread</i>	Capability selector of the thread for which the extended vCPU feature shall be enabled.
<i>ext_vcpu_state</i>	The virtual address where the kernel shall store the vCPU state in case of vCPU exits. The address must be a valid kernel-user-memory address (see <a href="#">L4::Task::add_ku_mem()</a> ).
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

#### Returns

Syscall return tag.

The extended vCPU feature allows the use of hardware-virtualization features such as Intel's VT-x (VMX) or AMD's AMD-V (SVM).

This function enables the extended vCPU feature of `this` thread. Enabling the extended vCPU feature also enables the vCPU feature.

The kernel-user memory area starting at `ext_vcpu_state` must be at least 4 KiB aligned and must cover a size of `L4_PAGESIZE`. It includes the data of [l4\\_vcpu\\_state\\_t](#) at offset 0, the extended vCPU state at offset `L4_VCPU_OFFSET_EXT_STATE`, and, on some platforms, the extended vCPU information at offset `L4_VCPU_OFFSET_EXT_INFOS`.

On Intel's VT-x (VMX), the extended vCPU state is [l4\\_vm\\_vmx\\_vcpu\\_vmcs\\_t](#) and the extended vCPU information is [l4\\_vm\\_vmx\\_vcpu\\_infos\\_t](#). Furthermore, the extended vCPU state needs to be associated with a vCPU context (see [l4\\_vm\\_vmx\\_set\\_hw\\_vmcs\(\)](#)).

On AMD's AMD-V (SVM), the extended vCPU state is [l4\\_vm\\_svm\\_vmcb\\_t](#).

#### Note

Enabling the extended vCPU feature for a thread running on a different CPU core is currently not supported.

Disabling of the extended vCPU feature is currently not supported.

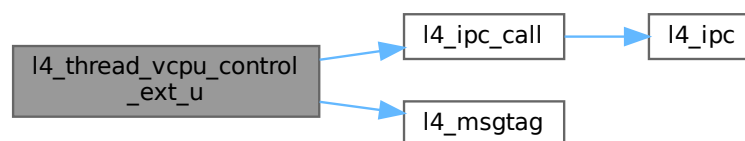
Upgrading from non-extended vCPU feature to extended vCPU feature is currently not supported.

Definition at line 1059 of file [thread.h](#).

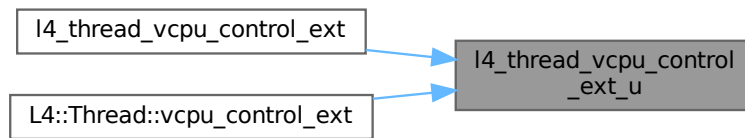
References [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_NOTHROW](#), [L4\\_PROTO\\_THREAD](#), and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [l4\\_thread\\_vcpu\\_control\\_ext\(\)](#), and [L4::Thread::vcpu\\_control\\_ext\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.1.10.13.3.16 l4\_thread\_vcpu\_control\_u()

```
l4_msgtag_t l4_thread_vcpu_control_u (
    l4_cap_idx_t thread,
    l4_addr_t vcpu_state,
    l4_utcb_t * utcb) [inline]
```

Enable the vCPU feature for the thread.

##### Parameters

<i>thread</i>	Capability selector of the thread for which the vCPU feature shall be enabled.
<i>vcpu_state</i>	A virtual address pointing to a <a href="#">l4_vcpu_state_t</a> . It must be a valid kernel-user-memory address (see <a href="#">L4::Task::add_ku_mem()</a> ).
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

##### Returns

Syscall return tag.

This function enables the vCPU feature of `this` thread

The kernel-user memory starting at `vcpu_state` must be at least 128-byte aligned and must cover the size of [l4\\_vcpu\\_state\\_t](#).

The asynchronous IPC handling is described at [vCPU API](#).

##### Note

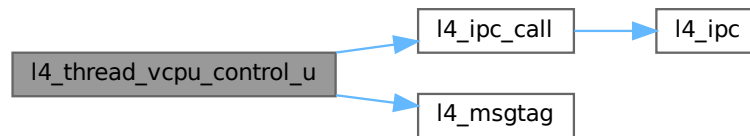
Disabling of the vCPU feature is optional and currently not supported.

Definition at line 1044 of file [thread.h](#).

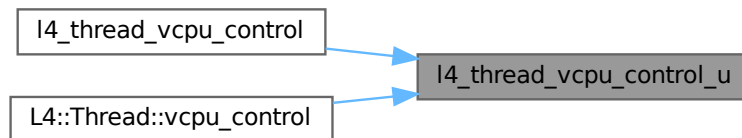
References [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_NOTHROW](#), [L4\\_PROTO\\_THREAD](#), [L4\\_THREAD\\_VCPU\\_CONTROL\\_ON](#) and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [l4\\_thread\\_vcpu\\_control\(\)](#), and [L4::Thread::vcpu\\_control\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.1.10.13.3.17 l4\_thread\_vcpu\_resume\_commit()

```

l4_msgtag_t l4_thread_vcpu_resume_commit (
    l4_cap_idx_t thread,
    l4_msgtag_t tag) [inline]
  
```

Commit vCPU resume.

##### Parameters

<i>thread</i>	Thread to be resumed, the invalid cap can be used for the current thread.
<i>tag</i>	Tag to use, returned by <a href="#">l4_thread_vcpu_resume_start()</a>

##### Returns

Syscall return tag containing one of the following return codes.

##### Return values

0	Indicates a VM exit, provided that <code>thread</code> is in extended vCPU mode with virtual interrupts cleared.
---	------------------------------------------------------------------------------------------------------------------

1	Indicates an incoming IPC message, provided that the <code>thread</code> is in extended vCPU mode with virtual interrupts cleared.
-L4_EPERM	The user task capability set in the vCPU state is missing the <a href="#">L4_CAP_FPAGE_S</a> right. On Intel's VT-x (VMX): The vCPU context capability set in the extended vCPU state is missing the <a href="#">L4_CAP_FPAGE_S</a> right.
-L4_ENOENT	The user task capability set in the vCPU state is invalid.
-L4_EINVAL	<code>thread</code> is not the current running thread, or does not have the vCPU feature enabled. On Intel's VT-x (VMX): No vCPU context associated with the extended vCPU state.
-L4_EBUSY	On Intel's VT-x (VMX): The vCPU context associated with the extended vCPU state is already active on a different CPU.
-L4_ENODEV	On Intel's VT-x (VMX): The vCPU context associated with the extended vCPU state cannot be initialized or activated.
<0	A supplied mapping failed.

All flexpages in the UTCB (added with [l4\\_sndfpage\\_add\(\)](#) after [l4\\_thread\\_vcpu\\_resume\\_start\(\)](#)) are unconditionally mapped into the user task configured in the vCPU state.

To resume into another address space, the capability to the target [Task](#) (or [L4::Vm](#)) must be set in [l4\\_vcpu\\_state\\_t::user\\_task](#) together with [L4\\_VCPU\\_F\\_USER\\_MODE](#). The capability selector must have all lower bits clear (see [L4\\_CAP\\_MASK](#)). The kernel adds the [L4\\_SYSF\\_SEND](#) flag there to indicate that the capability has been referenced in the kernel. Consecutive resumes will not reference the task capability again until all lower bits are cleared again. To release a task use a different task capability or use an invalid capability with the [L4\\_SYSF\\_REPLY](#) flag set.

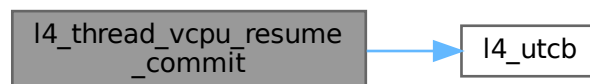
See also

[l4\\_vcpu\\_state\\_t](#)

Definition at line 1017 of file [thread.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.13.3.18 [l4\\_thread\\_vcpu\\_resume\\_start\(\)](#)

```

l4_msgtag_t l4_thread_vcpu_resume_start (
    void ) [inline]
  
```

vCPU return from event handler.

**Returns**

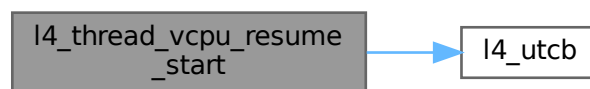
Message tag to be used for [l4\\_sndfpage\\_add\(\)](#) and [l4\\_thread\\_vcpu\\_resume\\_commit\(\)](#)

The vCPU resume functionality is split in multiple functions to allow the specification of additional send-flexpages using [l4\\_sndfpage\\_add\(\)](#).

Definition at line [1011](#) of file [thread.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**14.1.10.13.3.19 l4\_thread\_yield()**

```
l4_msgtag_t l4_thread_yield (
    void ) [inline]
```

Yield current time slice.

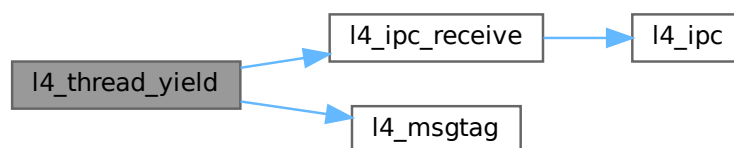
**Returns**

system call return tag

Definition at line [891](#) of file [thread.h](#).

References [L4\\_INVALID\\_CAP](#), [L4\\_IPC\\_BOTH\\_TIMEOUT\\_0](#), [l4\\_ipc\\_receive\(\)](#), [l4\\_msgtag\(\)](#), and [L4\\_NOTHROW](#).

Here is the call graph for this function:





#### 14.1.10.13.4 Thread control

API for Thread Control method.

Collaboration diagram for Thread control:



#### Functions

- void [l4\\_thread\\_control\\_start](#) (void) [L4\\_NOTHROW](#)  
*Start a thread control API sequence.*
- void [l4\\_thread\\_control\\_pager](#) ([l4\\_cap\\_idx\\_t](#) pager) [L4\\_NOTHROW](#)  
*Set the pager.*
- void [l4\\_thread\\_control\\_exc\\_handler](#) ([l4\\_cap\\_idx\\_t](#) exc\_handler) [L4\\_NOTHROW](#)  
*Set the exception handler.*
- void [l4\\_thread\\_control\\_bind](#) ([l4\\_utcb\\_t](#) \*thread\_utcb, [l4\\_cap\\_idx\\_t](#) task) [L4\\_NOTHROW](#)  
*Bind the thread to a task.*
- void [l4\\_thread\\_control\\_alien](#) (int on) [L4\\_NOTHROW](#)  
*Enable alien mode.*
- [l4\\_msgtag\\_t](#) [l4\\_thread\\_control\\_commit](#) ([l4\\_cap\\_idx\\_t](#) thread) [L4\\_NOTHROW](#)  
*Commit the thread control parameters.*

#### 14.1.10.13.4.1 Detailed Description

API for Thread Control method.

The thread control API provides access to almost any parameter of a thread object. The API is based on a single invocation of the thread object. However, because of the huge amount of parameters, the API provides a set of functions to set specific parameters of a thread and a commit function to commit the thread control call (see [l4\\_thread\\_control\\_commit\(\)](#)).

A thread control operation must always start with [l4\\_thread\\_control\\_start\(\)](#) and be committed with [l4\\_thread\\_control\\_commit\(\)](#). All other thread control parameter setter functions must be called between these two functions.

An example for a sequence of thread control API calls can be found below.

```

l4_thread_control_start();
l4_thread_control_pager(pager_cap);
l4_thread_control_bind (thread_utcb, task);
l4_thread_control_commit(thread_cap);
  
```

#### 14.1.10.13.4.2 Function Documentation

##### **l4\_thread\_control\_alien()**

```
void l4_thread_control_alien (  
    int on) [inline]
```

Enable alien mode.

##### **Parameters**

---

<i>on</i>	Boolean value defining the state of the feature.
-----------	--------------------------------------------------

For a thread in alien mode the kernel produces just an exception IPC for each IPC and exception caused by the alien thread instead of handling these events regularly. (Page faults of alien threads and interrupts occurring while the alien thread is running are always handled regularly.) While the alien thread is blocking, the exception handler can inspect and modify the state of the alien thread and potentially also the system call arguments. If the exception handler replies with [L4\\_PROTO\\_ALLOW\\_SYSCALL](#) as message tag, the kernel handles the next IPC or exception of the alien thread in a regular way. If the exception handler leaves certain thread state unchanged (in particular the instruction pointer), this will be the IPC or exception that caused the call of the exception handler. For a regularly processed IPC or exception of the alien thread the kernel also performs an exception IPC on kernel exit.

This feature can be used to attach a debugger to a thread and trace all object invocations and their results. It could also be used to handle other systems that use the same syscall instruction as [L4Re](#).

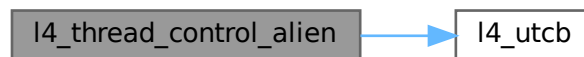
#### Examples

[examples/sys/aliens/main.c](#), and [examples/sys/singlestep/main.c](#).

Definition at line 981 of file [thread.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### **`l4_thread_control_bind()`**

```

void l4_thread_control_bind (
    l4_utcb_t * thread_utcb,
    l4_cap_idx_t task) [inline]
  
```

Bind the thread to a task.

#### Parameters

<i>thread_utcb</i>	The thread's UTCB address within the task it shall be bound to. The address must be aligned (architecture dependent; at least word aligned) and it must point to at least <a href="#">L4_UTCB_OFFSET</a> bytes of kernel-user memory.
<i>task</i>	The task the thread shall be bound to.

**Precondition**

The thread must not be bound to a task yet.

The capability `task` must have the permission [L4\\_CAP\\_FPAGE\\_S](#), otherwise the later call to [l4\\_thread\\_control\\_commit\(\)](#) will fail with [L4\\_EPERM](#).

A thread may execute code in the context of a task if and only if the thread is bound to the task. To actually start execution, [l4\\_thread\\_ex\\_regs\(\)](#) needs to be used. Execution in the context of the task means that the code has access to all the task's resources (and only those). The executed code itself must be one of those resources. A thread can be bound at most once to a task.

**Note**

The UTCBs of different threads in the same task should not overlap in order to prevent data corruption.

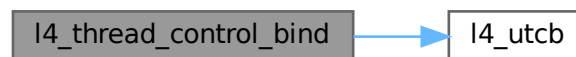
**Examples**

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 975 of file [thread.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**l4\_thread\_control\_commit()**

```
l4_msgtag_t l4_thread_control_commit (
    l4_cap_idx_t thread) [inline]
```

Commit the thread control parameters.

**Parameters**

<i>thread</i>	Capability selector of target thread to commit to.
---------------	----------------------------------------------------

**Returns**

Syscall return tag containing one of the following return codes.

**Return values**

<code>L4_EOK</code>	Operation successful.
<code>-L4_EPERM</code>	Insufficient permissions; see precondition.
<code>-L4_EINVAL</code>	Malformed thread control parameters.

#### Precondition

The capability `thread` must have the permission `L4_CAP_FPAGE_S`. When using `l4_thread_control_bind()`, also the respective task capability must have the permission `L4_CAP_FPAGE_S`.

#### Examples

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 987 of file [thread.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### `l4_thread_control_exc_handler()`

```
void l4_thread_control_exc_handler (  
    l4_cap_idx_t exc_handler) [inline]
```

Set the exception handler.

#### Parameters

<code>exc_handler</code>	Capability selector invoked to send an exception IPC.
--------------------------	-------------------------------------------------------

**Note**

The exception-handler capability selector is interpreted in the task the thread is bound to (executes in).

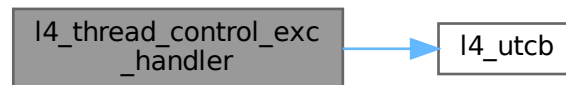
**Examples**

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 968 of file [thread.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**l4\_thread\_control\_pager()**

```
void l4_thread_control_pager (
    l4_cap_idx_t pager) [inline]
```

Set the pager.

**Parameters**

<i>pager</i>	Capability selector invoked to send a page-fault IPC.
--------------	-------------------------------------------------------

**Note**

The pager capability selector is interpreted in the task the thread is bound to (executes in).

**Examples**

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 962 of file [thread.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



**l4\_thread\_control\_start()**

```
void l4_thread_control_start (
    void ) [inline]
```

Start a thread control API sequence.

This function starts a sequence of thread control API functions. After this functions any of following functions may be called in any order.

- [l4\\_thread\\_control\\_pager\(\)](#)
- [l4\\_thread\\_control\\_exc\\_handler\(\)](#)
- [l4\\_thread\\_control\\_bind\(\)](#)
- [l4\\_thread\\_control\\_alien\(\)](#)

To commit the changes to the thread [l4\\_thread\\_control\\_commit\(\)](#) must be called in the end.

**Note**

The thread control API calls store the parameters for the thread in the UTCB of the caller (see [l4\\_utcb\(\)](#)), this means between [l4\\_thread\\_control\\_start\(\)](#) and [l4\\_thread\\_control\\_commit\(\)](#) no functions that modify the UTCB contents must be called.

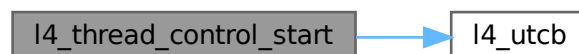
**Examples**

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 956 of file [thread.h](#).

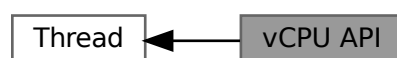
References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**14.1.10.13.5 vCPU API**

vCPU API.

Collaboration diagram for vCPU API:



## Data Structures

- struct [l4\\_vcpu\\_state\\_t](#)  
*State of a vCPU.*
- struct [l4\\_vcpu\\_regs\\_t](#)  
*vCPU registers.*
- struct [l4\\_vcpu\\_ipc\\_regs\\_t](#)  
*vCPU message registers.*

## Typedefs

- typedef struct [l4\\_vcpu\\_state\\_t](#) [l4\\_vcpu\\_state\\_t](#)  
*State of a vCPU.*
- typedef struct [l4\\_vcpu\\_regs\\_t](#) [l4\\_vcpu\\_regs\\_t](#)  
*vCPU registers.*
- typedef struct [l4\\_vcpu\\_ipc\\_regs\\_t](#) [l4\\_vcpu\\_ipc\\_regs\\_t](#)  
*vCPU message registers.*
- typedef [l4\\_exc\\_regs\\_t](#) [l4\\_vcpu\\_regs\\_t](#)  
*vCPU registers.*
- typedef struct [l4\\_vcpu\\_ipc\\_regs\\_t](#) [l4\\_vcpu\\_ipc\\_regs\\_t](#)  
*vCPU message registers.*
- typedef struct [l4\\_vcpu\\_regs\\_t](#) [l4\\_vcpu\\_regs\\_t](#)  
*vCPU registers.*
- typedef struct [l4\\_vcpu\\_ipc\\_regs\\_t](#) [l4\\_vcpu\\_ipc\\_regs\\_t](#)  
*vCPU message registers.*
- typedef struct [l4\\_vcpu\\_regs\\_t](#) [l4\\_vcpu\\_regs\\_t](#)  
*vCPU registers.*
- typedef struct [l4\\_vcpu\\_ipc\\_regs\\_t](#) [l4\\_vcpu\\_ipc\\_regs\\_t](#)  
*vCPU message registers.*

## Enumerations

- enum [l4\\_vcpu\\_state\\_flags](#) {  
  [L4\\_VCPU\\_F\\_IRQ](#) = 0x01 , [L4\\_VCPU\\_F\\_PAGE\\_FAULTS](#) = 0x02 , [L4\\_VCPU\\_F\\_EXCEPTIONS](#) = 0x04 ,  
  [L4\\_VCPU\\_F\\_USER\\_MODE](#) = 0x20 ,  
  [L4\\_VCPU\\_F\\_FPU\\_ENABLED](#) = 0x80 }  
*State flags of a vCPU.*
- enum [l4\\_vcpu\\_sticky\\_flags](#) { [L4\\_VCPU\\_SF\\_IRQ\\_PENDING](#) = 0x01 }  
*Sticky flags of a vCPU.*
- enum [l4\\_vcpu\\_state\\_offset](#) { [L4\\_VCPU\\_OFFSET\\_EXT\\_STATE](#) = 0x180 , [L4\\_VCPU\\_OFFSET\\_EXT\\_INFOS](#) = 0x100 }  
*Offsets for vCPU state layouts.*
- enum [l4\\_vcpu\\_state\\_offset](#) { [L4\\_VCPU\\_OFFSET\\_EXT\\_STATE](#) = 0x280 , [L4\\_VCPU\\_OFFSET\\_EXT\\_INFOS](#) = 0x200 }  
*Offsets for vCPU state layouts.*
- enum [l4\\_vcpu\\_state\\_offset](#) { [L4\\_VCPU\\_OFFSET\\_EXT\\_STATE](#) = 0x400 , [L4\\_VCPU\\_OFFSET\\_EXT\\_INFOS](#) = 0x200 }  
*Offsets for vCPU state layouts.*
- enum [l4\\_vcpu\\_state\\_offset](#) { [L4\\_VCPU\\_OFFSET\\_EXT\\_STATE](#) = 0x400 , [L4\\_VCPU\\_OFFSET\\_EXT\\_INFOS](#) = 0x200 }  
*Offsets for vCPU state layouts.*



#### 14.1.10.13.5.1 Detailed Description

vCPU API.

The vCPU API in [L4Re](#) implements virtual processors (vCPUs) on top of [L4::Thread](#). This API can be used for user level threading, operating system rehosting (see L4Linux) and virtualization.

You switch a thread into vCPU operation with [L4::Thread::vcpu\\_control](#).

In vCPU mode, incoming IPC can be redirected to a handler function. If an IPC is sent to the vCPU, the thread's normal execution is interrupted and the handler called. Which kind of IPC is redirected is specified by the [L4\\_vcpu\\_state\\_flags](#) set in the [l4\\_vcpu\\_state\\_t::state](#) field of [vcpu\\_state](#). All events enabled in the [vcpu\\_state](#) field are redirected to the handler. The handler is set via [l4\\_vcpu\\_state\\_t::entry\\_ip](#) and [l4\\_vcpu\\_state\\_t::entry\\_sp](#). IPC redirection works independent of "kernel" and "user" mode, but see [l4\\_vcpu\\_state\\_t::entry\\_sp](#). When the entry handler is called, the UTCB contains the result of the IPC and content normally found in CPU register is in [l4\\_vcpu\\_state\\_t::i](#).

Furthermore, the thread can execute in the context of different tasks, called the "kernel" and the "user" mode. The kernel task is the one to which the thread was originally bound via [L4::Thread::control\(\)](#). Execution starts in the kernel task and it is always switched to when the asynchronous IPC handler is invoked. When returning from the handler via [l4\\_thread\\_vcpu\\_resume\\_start\(\)](#) and [l4\\_thread\\_vcpu\\_resume\\_commit\(\)](#), a different user task can be specified by setting [l4\\_vcpu\\_state\\_t::user\\_task](#) and enabling the [L4\\_VCPU\\_F\\_USER\\_MODE](#) flag in [l4\\_vcpu\\_state\\_t::state](#). Note that the kernel may cache the user task internally, see [l4\\_thread\\_vcpu\\_resume\\_commit\(\)](#).

If the [L4\\_VCPU\\_F\\_USER\\_MODE](#) flag is enabled, the following flags will be automatically enabled in [l4\\_vcpu\\_state\\_t::state](#) on [L4::Thread::vcpu\\_resume\\_commit\(\)](#):

- [L4\\_VCPU\\_F\\_IRQ](#)
- [L4\\_VCPU\\_F\\_PAGE\\_FAULTS](#)
- [L4\\_VCPU\\_F\\_EXCEPTIONS](#)

When the kernel mode is entered, the following flags will be automatically disabled in [l4\\_vcpu\\_state\\_t::state](#):

- [L4\\_VCPU\\_F\\_IRQ](#)
- [L4\\_VCPU\\_F\\_PAGE\\_FAULTS](#)
- [L4\\_VCPU\\_F\\_USER\\_MODE](#)

Extended vCPU operation is used for hardware CPU virtualization. It can be enabled with [L4::Thread::vcpu\\_control\\_ext\(\)](#).

[vCPU Support Library](#) defines a convenience API for working with vCPUs.

See also

[vCPU Support Library](#)

#### 14.1.10.13.5.2 Enumeration Type Documentation

##### **L4\_vcpu\_state\_flags**

```
enum L4_vcpu_state_flags
```

State flags of a vCPU.

##### **Enumerator**

L4_VCPU_F_IRQ	<p>Receiving of IRQs and IPC enabled. While this flag is not set, the corresponding vCPU thread will not receive any IPC and threads attempting to send an IPC to this thread will block (according to the selected send timeout).</p> <p><b>Note</b></p> <p>On <a href="#">L4::Thread::vcpu_resume_commit()</a> this flag is automatically enabled in <a href="#">l4_vcpu_state_t::state</a> if <a href="#">L4_VCPU_F_USER_MODE</a> is enabled.</p> <p>When the kernel mode is entered, this flags is automatically disabled in <a href="#">l4_vcpu_state_t::state</a>.</p>
L4_VCPU_F_PAGE_FAULTS	<p>Page faults enabled. If this flag is set, a page fault switches to kernel mode (potentially causing a VM exit) and calls the entry handler. If this flag is not set, a page fault generates a page fault IPC to the pager of the vCPU thread.</p> <p><b>Note</b></p> <p>IPC redirection for page faults controlled by this flag works independent of "kernel" and "user" mode.</p> <p>On <a href="#">L4::Thread::vcpu_resume_commit()</a> this flag is automatically enabled in <a href="#">l4_vcpu_state_t::state</a> if <a href="#">L4_VCPU_F_USER_MODE</a> is enabled.</p> <p>When the kernel mode is entered, this flags is automatically disabled in <a href="#">l4_vcpu_state_t::state</a>.</p>
L4_VCPU_F_EXCEPTIONS	<p>Exceptions enabled. If this flag is set, then, on the event of an exception, the vCPU switches to kernel mode (potentially causing a VM exit) and calls the entry handler. If this flag is not set, an exception generates an exception IPC to the exception handler of the vCPU thread.</p> <p><b>Note</b></p> <p>IPC redirection for exceptions controlled by this flag works independent of "kernel" and "user" mode.</p> <p>On <a href="#">L4::Thread::vcpu_resume_commit()</a> this flag is automatically enabled in <a href="#">l4_vcpu_state_t::state</a> if <a href="#">L4_VCPU_F_USER_MODE</a> is enabled.</p>
L4_VCPU_F_USER_MODE	<p>User task will be used. If set, the vCPU switches to user mode on next <a href="#">L4::Thread::vcpu_resume_commit()</a>. If clear, the vCPU stays in "kernel" mode.</p> <p><b>Note</b></p> <p>When the kernel mode is entered, this flags is automatically disabled in <a href="#">l4_vcpu_state_t::state</a>.</p>
L4_VCPU_F_FPU_ENABLED	<p>FPU enabled. This flag is only relevant if <a href="#">L4_VCPU_F_USER_MODE</a> is set. Setting this flag allows code in vCPU mode to use the FPU. IF this flag is not set, any FPU operation will trigger a corresponding exception (FPU fault).</p>

Definition at line 101 of file [vcpu.h](#).

#### L4\_vcpu\_state\_offset [1/4]

enum [L4\\_vcpu\\_state\\_offset](#)

Offsets for vCPU state layouts.

#### Enumerator

L4_VCPU_OFFSET_EXT_STATE	Offset where extended state begins.
L4_VCPU_OFFSET_EXT_INFOS	Offset where extended infos begin.

Definition at line 34 of file [\\_\\_vcpu-arch.h](#).

#### L4\_vcpu\_state\_offset [2/4]

enum [L4\\_vcpu\\_state\\_offset](#)

Offsets for vCPU state layouts.

#### Enumerator

L4_VCPU_OFFSET_EXT_STATE	Offset where extended state begins.
L4_VCPU_OFFSET_EXT_INFOS	Offset where extended infos begin.

Definition at line 35 of file [\\_\\_vcpu-arch.h](#).

#### L4\_vcpu\_state\_offset [3/4]

enum [L4\\_vcpu\\_state\\_offset](#)

Offsets for vCPU state layouts.

#### Enumerator

L4_VCPU_OFFSET_EXT_STATE	Offset where extended state begins.
L4_VCPU_OFFSET_EXT_INFOS	Offset where extended infos begin.

Definition at line 36 of file [\\_\\_vcpu-arch.h](#).

#### L4\_vcpu\_state\_offset [4/4]

enum [L4\\_vcpu\\_state\\_offset](#)

Offsets for vCPU state layouts.

#### Enumerator

L4_VCPU_OFFSET_EXT_STATE	Offset where extended state begins.
--------------------------	-------------------------------------

L4_VCPU_OFFSET_EXT_INFOS	Offset where extended infos begin.
--------------------------	------------------------------------

Definition at line 34 of file [\\_\\_vcpu-arch.h](#).

## L4\_vcpu\_sticky\_flags

enum [L4\\_vcpu\\_sticky\\_flags](#)

Sticky flags of a vCPU.

### Enumerator

L4_VCPU_SF_IRQ_PENDING	An event is pending: Either an IRQ or another thread attempts to send an IPC to this vCPU thread.
------------------------	---------------------------------------------------------------------------------------------------

Definition at line 167 of file [vcpu.h](#).

## 14.1.10.14 Thread groups

C thread group interface, see [L4::Thread\\_group](#) for the C++ interface.

Collaboration diagram for Thread groups:



### Functions

- [l4\\_msgtag\\_t l4\\_thread\\_group\\_add](#) ([l4\\_cap\\_idx\\_t](#) tg, [l4\\_cap\\_idx\\_t](#) thread) [L4\\_NOTHROW](#)  
*Add thread to a thread group.*
- [l4\\_msgtag\\_t l4\\_thread\\_group\\_remove](#) ([l4\\_cap\\_idx\\_t](#) tg, [l4\\_cap\\_idx\\_t](#) thread) [L4\\_NOTHROW](#)  
*Remove thread from a thread group.*

### 14.1.10.14.1 Detailed Description

C thread group interface, see [L4::Thread\\_group](#) for the C++ interface.

An [L4](#) thread group is a collection of threads used as indirection for IPC gate and IRQ objects such that these objects can have multiple receivers, from which the kernel selects one according to a policy.

The primary use case for thread groups are multi-threaded servers and CPU core local IRQ / IPC delivery.

A thread can be bound to at most one thread group. Before binding a thread to a thread group, the thread must be bound to a task. All threads bound to the same thread group must belong to the same task.

### 14.1.10.14.2 Function Documentation

#### 14.1.10.14.2.1 l4\_thread\_group\_add()

```
l4_msgtag_t l4_thread_group_add (  
    l4_cap_idx_t tg,  
    l4_cap_idx_t thread) [inline]
```

Add thread to a thread group.

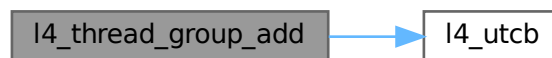
##### Parameters

<i>tg</i>	Thread group object.
<i>thread</i>	Thread to add to the thread group.

Definition at line 114 of file [thread\\_group.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.14.2.2 l4\_thread\_group\_remove()

```
l4_msgtag_t l4_thread_group_remove (  
    l4_cap_idx_t tg,  
    l4_cap_idx_t thread) [inline]
```

Remove thread from a thread group.

##### Parameters

<i>tg</i>	Thread group object.
<i>thread</i>	Thread to remove from the thread group.

Definition at line 121 of file [thread\\_group.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 14.1.10.15 Virtual Console

C Virtual console interface for simple character based input and output, see [L4::Vcon](#) for the C++ interface.

Collaboration diagram for Virtual Console:



#### Data Structures

- struct [l4\\_vcon\\_attr\\_t](#)  
*Vcon attribute structure.*

#### Typedefs

- typedef struct l4\_vcon\_attr\_t [l4\\_vcon\\_attr\\_t](#)  
*Vcon attribute structure.*

#### Enumerations

- enum [L4\\_vcon\\_size\\_consts](#) { [L4\\_VCON\\_WRITE\\_SIZE](#) = (L4\_UTCB\_GENERIC\_DATA\_SIZE - 2) \* sizeof(l4\_umword\_t) , [L4\\_VCON\\_READ\\_SIZE](#) = (L4\_UTCB\_GENERIC\_DATA\_SIZE - 1) \* sizeof(l4\_umword\_t) }  
*Size constants.*
- enum [L4\\_vcon\\_i\\_flags](#) { [L4\\_VCON\\_INLCR](#) = 000100 , [L4\\_VCON\\_IGNCR](#) = 000200 , [L4\\_VCON\\_ICRNL](#) = 000400 }  
*Input flags.*
- enum [L4\\_vcon\\_o\\_flags](#) { [L4\\_VCON\\_ONLCR](#) = 000004 , [L4\\_VCON\\_OCRNL](#) = 000010 , [L4\\_VCON\\_ONLRET](#) = 000040 }  
*Output flags.*
- enum [L4\\_vcon\\_l\\_flags](#) { [L4\\_VCON\\_ICANON](#) = 000002 , [L4\\_VCON\\_ECHO](#) = 000010 }  
*Local flags.*

## Functions

- [l4\\_msgtag\\_t l4\\_vcon\\_send](#) ([l4\\_cap\\_idx\\_t](#) vcon, char const \*buf, unsigned size) [L4\\_NOTHROW](#)  
*Send data to virtual console.*
- [l4\\_msgtag\\_t l4\\_vcon\\_send\\_u](#) ([l4\\_cap\\_idx\\_t](#) vcon, char const \*buf, unsigned size, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Send data to *this* virtual console.*
- long [l4\\_vcon\\_write](#) ([l4\\_cap\\_idx\\_t](#) vcon, char const \*buf, unsigned size) [L4\\_NOTHROW](#)  
*Write data to virtual console.*
- long [l4\\_vcon\\_write\\_u](#) ([l4\\_cap\\_idx\\_t](#) vcon, char const \*buf, unsigned size, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Write data to *this* virtual console.*
- int [l4\\_vcon\\_read](#) ([l4\\_cap\\_idx\\_t](#) vcon, char \*buf, unsigned size) [L4\\_NOTHROW](#)  
*Read data from virtual console.*
- int [l4\\_vcon\\_read\\_u](#) ([l4\\_cap\\_idx\\_t](#) vcon, char \*buf, unsigned size, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Read data from *this* virtual console.*
- int [l4\\_vcon\\_read\\_with\\_flags](#) ([l4\\_cap\\_idx\\_t](#) vcon, char \*buf, unsigned size) [L4\\_NOTHROW](#)  
*Read data from virtual console, extended version including flags.*
- [l4\\_msgtag\\_t l4\\_vcon\\_set\\_attr](#) ([l4\\_cap\\_idx\\_t](#) vcon, [l4\\_vcon\\_attr\\_t](#) const \*attr) [L4\\_NOTHROW](#)  
*Set attributes of a Vcon.*
- [l4\\_msgtag\\_t l4\\_vcon\\_set\\_attr\\_u](#) ([l4\\_cap\\_idx\\_t](#) vcon, [l4\\_vcon\\_attr\\_t](#) const \*attr, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Set the attributes of *this* virtual console.*
- [l4\\_msgtag\\_t l4\\_vcon\\_get\\_attr](#) ([l4\\_cap\\_idx\\_t](#) vcon, [l4\\_vcon\\_attr\\_t](#) \*attr) [L4\\_NOTHROW](#)  
*Get attributes of a Vcon.*
- [l4\\_msgtag\\_t l4\\_vcon\\_get\\_attr\\_u](#) ([l4\\_cap\\_idx\\_t](#) vcon, [l4\\_vcon\\_attr\\_t](#) \*attr, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Get attributes of *this* virtual console.*
- void [l4\\_vcon\\_set\\_attr\\_raw](#) ([l4\\_vcon\\_attr\\_t](#) \*attr) [L4\\_NOTHROW](#)  
*Set terminal attributes to disable all special processing.*

### 14.1.10.15.1 Detailed Description

C Virtual console interface for simple character based input and output, see [L4::Vcon](#) for the C++ interface.

The interrupt for read events is provided by the virtual key interrupt which, in contrast to hardware IRQs, implements a limited functionality:

- Only IRQ line 0 is supported, no MSI vectors.
- The IRQ is edge-triggered and the IRQ mode cannot be changed.
- As the IRQ is edge-triggered, it does not have to be explicitly unmasked.

A server implementing the virtual console protocol has a queue for input events. When the first input event is added to the empty queue, the virtual key interrupt is triggered. Further events are added to the queue without generating further interrupts. The queue is emptied when a client reads all queued input events.

#### Include File

```
#include <l4/sys/vcon.h>
```

See [L4::Vcon](#) for the C++ interface.

### 14.1.10.15.2 Typedef Documentation

#### 14.1.10.15.2.1 l4\_vcon\_attr\_t

```
typedef struct l4_vcon_attr_t l4_vcon_attr_t
```

Vcon attribute structure.

The flags members can be a combination of their respective enums.

See also

[L4\\_vcon\\_i\\_flags](#)

[L4\\_vcon\\_o\\_flags](#)

[L4\\_vcon\\_l\\_flags](#)

### 14.1.10.15.3 Enumeration Type Documentation

#### 14.1.10.15.3.1 L4\_vcon\_i\_flags

```
enum L4_vcon_i_flags
```

Input flags.

##### Enumerator

L4_VCON_INLCR	Translate NL to CR.
L4_VCON_IGNCR	Ignore CR.
L4_VCON_ICRNL	Translate CR to NL if L4_VCON_IGNCR is not set.

Definition at line 208 of file [vcon.h](#).

#### 14.1.10.15.3.2 L4\_vcon\_l\_flags

```
enum L4_vcon_l_flags
```

Local flags.

##### Enumerator

L4_VCON_ICANON	Canonical mode.
L4_VCON_ECHO	Echo input.

Definition at line 230 of file [vcon.h](#).

#### 14.1.10.15.3.3 L4\_vcon\_o\_flags

```
enum L4_vcon_o_flags
```

Output flags.

##### Enumerator



L4_VCON_ONLCR	Translate NL to CR-NL.
L4_VCON_OCRNL	Translate CR to NL.
L4_VCON_ONLRET	Do not output CR.

Definition at line 219 of file [vcon.h](#).

#### 14.1.10.15.3.4 L4\_vcon\_size\_consts

```
enum L4_vcon_size_consts
```

Size constants.

##### Enumerator

L4_VCON_WRITE_SIZE	Maximum size that can be written with one l4_vcon_write call.
L4_VCON_READ_SIZE	Maximum size that can be read with one l4_vcon_read* call.

Definition at line 95 of file [vcon.h](#).

#### 14.1.10.15.4 Function Documentation

##### 14.1.10.15.4.1 l4\_vcon\_get\_attr()

```
l4_msgtag_t l4_vcon_get_attr (
    l4_cap_idx_t vcon,
    l4_vcon_attr_t * attr) [inline]
```

Get attributes of a Vcon.

##### Parameters

	<i>vcon</i>	Vcon object.
out	<i>attr</i>	Attribute structure.

##### Returns

Syscall return tag

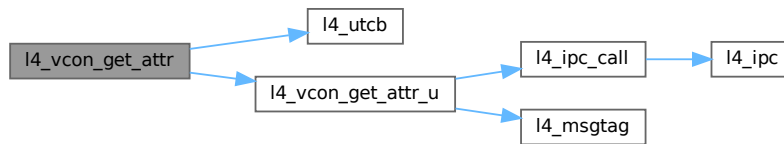
##### Examples

[examples/sys/isr/main.c](#).

Definition at line 435 of file [vcon.h](#).

References [L4\\_NOTHROW](#), [l4\\_utcb\(\)](#), and [l4\\_vcon\\_get\\_attr\\_u\(\)](#).

Here is the call graph for this function:



#### 14.1.10.15.4.2 l4\_vcon\_get\_attr\_u()

```

l4_msgtag_t l4_vcon_get_attr_u (
    l4_cap_idx_t vcon,
    l4_vcon_attr_t * attr,
    l4_utcb_t * utcb) [inline]
  
```

Get attributes of this virtual console.

#### Parameters

	<i>vcon</i>	Capability index of the vcon object.
out	<i>attr</i>	Attribute structure. Contains the attributes after a successful call of this function.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

#### Returns

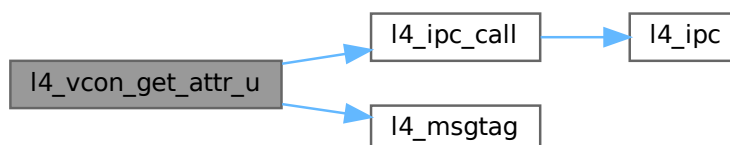
Syscall return tag.

Definition at line 417 of file [vcon.h](#).

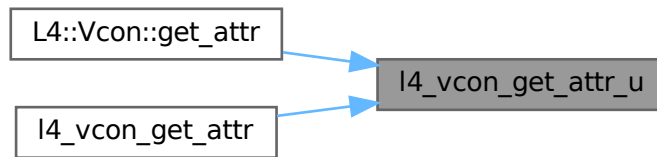
References [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_NOTHROW](#), [L4\\_PROTO\\_LOG](#), [L4\\_VCON\\_GET\\_ATTR\\_OP](#), and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [L4::Vcon::get\\_attr\(\)](#), and [l4\\_vcon\\_get\\_attr\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.1.10.15.4.3 l4\_vcon\_read()

```
int l4_vcon_read (
    l4_cap_idx_t vcon,
    char * buf,
    unsigned size) [inline]
```

Read data from virtual console.

##### Parameters

	<i>vcon</i>	Vcon object.
out	<i>buf</i>	Pointer to data buffer.
	<i>size</i>	Size of buffer in bytes.

##### Return values

<i>-L4_EPERM</i>	Insufficient permissions; see precondition.
<i>&gt;size</i>	More bytes to read, <i>size</i> bytes are in the buffer <i>buf</i> .
<i>&lt;=size</i>	Number of bytes read.

##### Precondition

The capability *vcon* must have the permission [L4\\_CAP\\_FPAGE\\_W](#).

##### Note

Size must not exceed [L4\\_VCON\\_READ\\_SIZE](#).

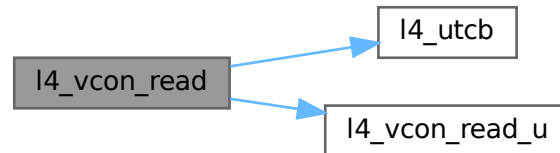
##### Examples

[examples/sys/isr/main.c](#).

Definition at line 391 of file [vcon.h](#).

References [L4\\_NOTHROW](#), [l4\\_utcb\(\)](#), and [l4\\_vcon\\_read\\_u\(\)](#).

Here is the call graph for this function:



#### 14.1.10.15.4.4 l4\_vcon\_read\_u()

```
int l4_vcon_read_u (
    l4_cap_idx_t vcon,
    char * buf,
    unsigned size,
    l4_utcb_t * utcb) [inline]
```

Read data from this virtual console.

##### Parameters

	<i>vcon</i>	Capability index of the vcon object.
out	<i>buf</i>	Pointer to data buffer.
	<i>size</i>	Size of the data buffer in bytes.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

##### Return values

<i>-L4_EPERM</i>	Insufficient permissions; see precondition.
<i>&gt;size</i>	More bytes to read, <i>size</i> bytes are in the buffer <i>buf</i> .
<i>&lt;=size</i>	Number of bytes read.

##### Precondition

The invoked Vcon capability must have the permission [L4\\_CAP\\_FPAGE\\_W](#).

**Note**

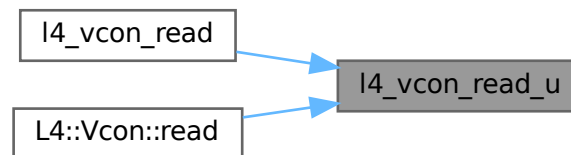
Size must not exceed [L4\\_VCON\\_READ\\_SIZE](#).

Definition at line 381 of file [vcon.h](#).

References [L4\\_NOTHROW](#), and [L4\\_VCON\\_READ\\_SIZE\\_MASK](#).

Referenced by [l4\\_vcon\\_read\(\)](#), and [L4::Vcon::read\(\)](#).

Here is the caller graph for this function:

**14.1.10.15.4.5 l4\_vcon\_read\_with\_flags()**

```
int l4_vcon_read_with_flags (
    l4_cap_idx_t vcon,
    char * buf,
    unsigned size) [inline]
```

Read data from virtual console, extended version including flags.

**Parameters**

	<i>vcon</i>	Vcon object.
out	<i>buf</i>	Pointer to data buffer.
	<i>size</i>	Size of buffer in bytes.

If this function returns a positive value the caller can check the [L4\\_VCON\\_READ\\_STAT\\_BREAK](#) flag bit for a break condition. The bytes read can be obtained by masking the return value with [L4\\_VCON\\_READ\\_SIZE\\_MASK](#).

If a break condition is signaled, it is always the first event in the transmitted content, i.e. all characters supplied by this read call follow the break condition.

*buf* might be a `NULL`, in this case the input data will be dropped.

**Note**

Size must not exceed [L4\\_VCON\\_READ\\_SIZE](#).

**Return values**

<code>-L4_EPERM</code>	Insufficient permissions; see precondition.
<code>&gt;size</code>	More bytes to read, <code>size</code> bytes are in the buffer <code>buf</code> .
<code>&lt;=size</code>	Number of bytes read.

#### Precondition

The capability `vcon` must have the permission `L4_CAP_FPAGE_W`.

Definition at line 375 of file `vcon.h`.

References `L4_NOTHROW`, and `l4_utcb()`.

Here is the call graph for this function:



#### 14.1.10.15.4.6 l4\_vcon\_send()

```

l4_msgtag_t l4_vcon_send (
    l4_cap_idx_t vcon,
    char const * buf,
    unsigned size) [inline]
  
```

Send data to virtual console.

#### Parameters

<code>vcon</code>	Vcon object.
<code>buf</code>	Pointer to data buffer.
<code>size</code>	Size of buffer in bytes.

#### Returns

Syscall return tag

**Note**

Size must not exceed [L4\\_VCON\\_WRITE\\_SIZE](#), a proper value of the `size` parameter is NOT checked. Also, this function is a send only operation, this means there is no return value except for a failed send operation. Use [l4\\_ipc\\_error\(\)](#) to check for send errors, and **do not** use [l4\\_error\(\)](#).

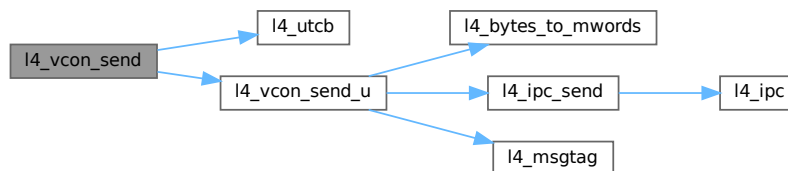
**Examples**

[examples/sys/utcb-ipc/main.c](#).

Definition at line 315 of file [vcon.h](#).

References [L4\\_NOTHROW](#), [l4\\_utcb\(\)](#), and [l4\\_vcon\\_send\\_u\(\)](#).

Here is the call graph for this function:

**14.1.10.15.4.7 l4\_vcon\_send\_u()**

```

l4_msgtag_t l4_vcon_send_u (
    l4_cap_idx_t vcon,
    char const * buf,
    unsigned size,
    l4_utcb_t * utcb) [inline]
  
```

Send data to this virtual console.

**Parameters**

<i>vcon</i>	Capability index of the Vcon object.
<i>buf</i>	Pointer to the data buffer.
<i>size</i>	Size of the data buffer in bytes.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

**Returns**

Syscall return tag

**Note**

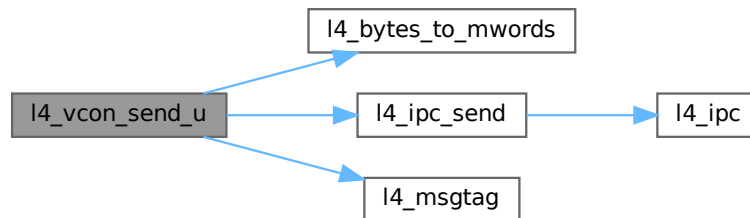
Size must not exceed [L4\\_VCON\\_WRITE\\_SIZE](#), a proper value of the `size` parameter is NOT checked. Also, this function is a send only operation, this means there is no return value except for a failed send operation. Use [l4\\_ipc\\_error\(\)](#) to check for send errors, do not use [l4\\_error\(\)](#), as [l4\\_error\(\)](#) will always return an error.

Definition at line 302 of file [vcon.h](#).

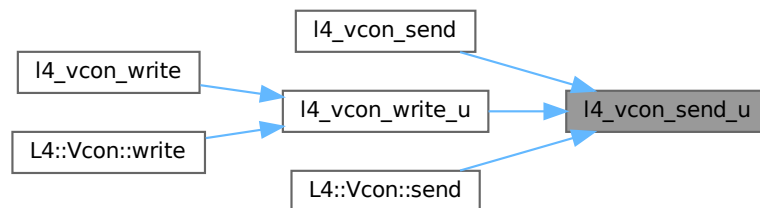
References [l4\\_bytes\\_to\\_mwords\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_ipc\\_send\(\)](#), [l4\\_msgtag\(\)](#), [L4\\_MSGTAG\\_SCHEDULE](#), [L4\\_NOTHROW](#), [L4\\_PROTO\\_LOG](#), [L4\\_VCON\\_WRITE\\_OP](#), and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [l4\\_vcon\\_send\(\)](#), [l4\\_vcon\\_write\\_u\(\)](#), and [L4::Vcon::send\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.1.10.15.4.8 l4\_vcon\_set\_attr()

```

l4_msgtag_t l4_vcon_set_attr (
    l4_cap_idx_t vcon,
    l4_vcon_attr_t const * attr) [inline]
  
```

Set attributes of a Vcon.

#### Parameters



<i>vcon</i>	Vcon object.
<i>attr</i>	Attribute structure.

**Returns**

Syscall return tag

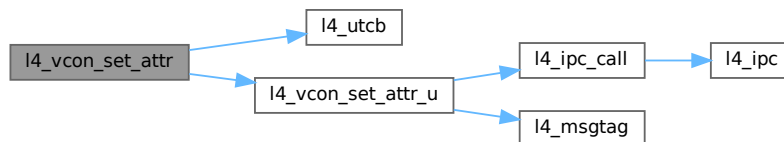
**Examples**

[examples/sys/isr/main.c](#).

Definition at line 411 of file [vcon.h](#).

References [L4\\_NOTHROW](#), [l4\\_utcb\(\)](#), and [l4\\_vcon\\_set\\_attr\\_u\(\)](#).

Here is the call graph for this function:

**14.1.10.15.4.9 l4\_vcon\_set\_attr\_raw()**

```
void l4_vcon_set_attr_raw (
    l4_vcon_attr_t * attr) [inline]
```

Set terminal attributes to disable all special processing.

Removes all flags that would mangle the read or written characters. Also disables echoing and any special processing of characters.

**Parameters**

<i>in, out</i>	<i>attr</i>	Attribute structure to update.
----------------	-------------	--------------------------------

Definition at line 441 of file [vcon.h](#).

References [L4\\_NOTHROW](#).

Referenced by [l4\\_vcon\\_attr\\_t::set\\_raw\(\)](#).

Here is the caller graph for this function:



#### 14.1.10.15.4.10 l4\_vcon\_set\_attr\_u()

```
l4_msgtag_t l4_vcon_set_attr_u (
    l4_cap_idx_t vcon,
    l4_vcon_attr_t const * attr,
    l4_utcb_t * utcb) [inline]
```

Set the attributes of this virtual console.

##### Parameters

<i>vcon</i>	Capability index of the vcon object.
<i>attr</i>	Attribute structure with the attributes for the virtual console.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

##### Returns

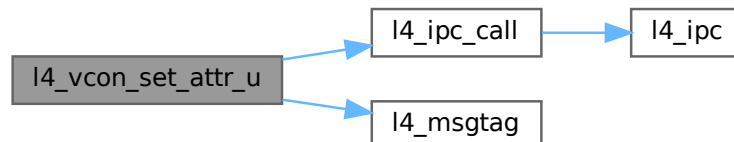
Syscall return tag.

Definition at line 397 of file [vcon.h](#).

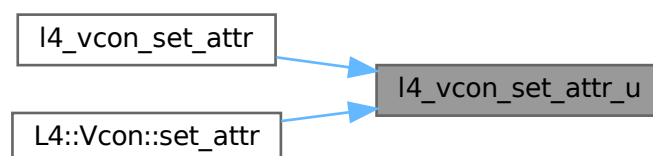
References [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_NOTHROW](#), [L4\\_PROTO\\_LOG](#), [L4\\_VCON\\_SET\\_ATTR\\_OP](#), and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [l4\\_vcon\\_set\\_attr\(\)](#), and [L4::Vcon::set\\_attr\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



**14.1.10.15.4.11 l4\_vcon\_write()**

```
long l4_vcon_write (
    l4_cap_idx_t vcon,
    char const * buf,
    unsigned size) [inline]
```

Write data to virtual console.

**Parameters**

<i>vcon</i>	Vcon object.
<i>buf</i>	Pointer to data buffer.
<i>size</i>	Size of buffer in bytes.

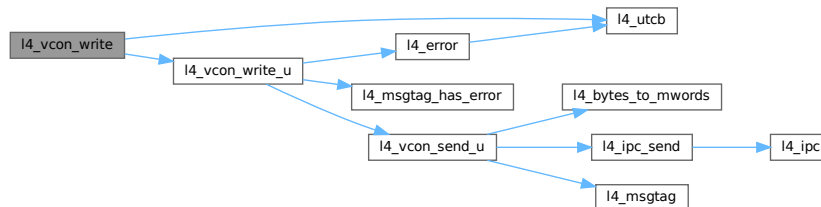
**Return values**

<0	Error.
>=0	Number of bytes written to the virtual console

Definition at line 336 of file [vcon.h](#).

References [L4\\_NOTHROW](#), [l4\\_utcb\(\)](#), and [l4\\_vcon\\_write\\_u\(\)](#).

Here is the call graph for this function:

**14.1.10.15.4.12 l4\_vcon\_write\_u()**

```
long l4_vcon_write_u (
    l4_cap_idx_t vcon,
    char const * buf,
    unsigned size,
    l4_utcb_t * utcb) [inline]
```

Write data to this virtual console.

**Parameters**

<i>vcon</i>	Capability index of the vcon object.
<i>buf</i>	Pointer to the data buffer.
<i>size</i>	Size of the data buffer in bytes.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

### Return values

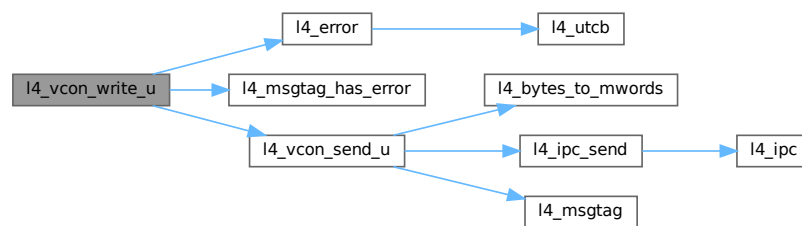
$< 0$	Error.
$\geq 0$	Number of bytes written to the virtual console.

Definition at line 321 of file [vcon.h](#).

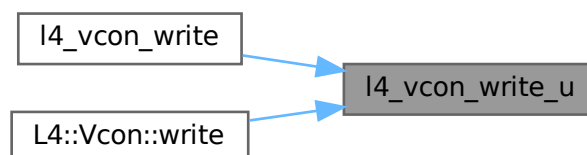
References [l4\\_error\(\)](#), [l4\\_msgtag\\_has\\_error\(\)](#), [L4\\_NOTHROW](#), [l4\\_vcon\\_send\\_u\(\)](#), and [L4\\_VCON\\_WRITE\\_SIZE](#).

Referenced by [l4\\_vcon\\_write\(\)](#), and [L4::Vcon::write\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



## 14.1.11 Kernel Interface Page

Kernel Interface Page.

Collaboration diagram for Kernel Interface Page:



## Topics

- [Memory descriptors \(C version\)](#) . . . . . 445  
*C Interface for KIP memory descriptors.*

## Data Structures

- class [L4::Kip::Mem\\_desc](#)  
*Memory descriptors stored in the kernel interface page.*
- struct [l4\\_kernel\\_info\\_t](#)  
*L4 Kernel Interface Page.*

## Macros

- `#define L4_KERNEL_INFO_MAGIC (0x4BE6344CL) /* "L4μK" */`  
*Kernel Info Page identifier ("L4μK").*

## Typedefs

- typedef struct [l4\\_kernel\\_info\\_t](#) [l4\\_kernel\\_info\\_t](#)  
*L4 Kernel Interface Page.*

## Enumerations

- enum { [L4\\_KIP\\_OFFS\\_READ\\_US](#) = 0x900 , [L4\\_KIP\\_OFFS\\_READ\\_NS](#) = 0x980 }

## Functions

- [l4\\_kernel\\_info\\_t](#) const \* [l4\\_kip](#) (void) [L4\\_NOTHROW](#)  
*Get Kernel Info Page.*
- [l4\\_umword\\_t](#) [l4\\_kip\\_version](#) ([l4\\_kernel\\_info\\_t](#) const \*kip) [L4\\_NOTHROW](#)  
*Get the kernel version.*
- const char \* [l4\\_kip\\_version\\_string](#) ([l4\\_kernel\\_info\\_t](#) const \*kip) [L4\\_NOTHROW](#)  
*Get the kernel version string.*
- int [l4\\_kernel\\_info\\_version\\_offset](#) ([l4\\_kernel\\_info\\_t](#) const \*kip) [L4\\_NOTHROW](#)  
*Return offset in bytes of version\_strings relative to the KIP base.*
- [l4\\_cpu\\_time\\_t](#) [l4\\_kip\\_clock](#) ([l4\\_kernel\\_info\\_t](#) const \*kip) [L4\\_NOTHROW](#)  
*Return clock value from the KIP.*
- [l4\\_umword\\_t](#) [l4\\_kip\\_clock\\_lw](#) ([l4\\_kernel\\_info\\_t](#) const \*kip) [L4\\_NOTHROW](#)  
*Return least significant machine word of clock value from the KIP.*
- [l4\\_uint64\\_t](#) [l4\\_kip\\_clock\\_ns](#) ([l4\\_kernel\\_info\\_t](#) const \*kip) [L4\\_NOTHROW](#)  
*Return current clock using the KIP in nanoseconds.*

#### 14.1.11.1 Detailed Description

Kernel Interface Page.

C interface for the Kernel Interface Page:

C++ interface for the Kernel Interface Page:

##### Include File

```
#include <l4/sys/kip>
```

##### Include File

```
#include <l4/sys/kip.h>
```

#### 14.1.11.2 Typedef Documentation

##### 14.1.11.2.1 l4\_kernel\_info\_t

```
typedef struct l4_kernel_info_t l4_kernel_info_t
```

[L4](#) Kernel Interface Page.

32-bit architecture may assume that the upper 32 bits of addresses is 0

#### 14.1.11.3 Enumeration Type Documentation

##### 14.1.11.3.1 anonymous enum

```
anonymous enum
```

##### Enumerator

L4_KIP_OFFS_READ_US	Offset of KIP code (provided by the kernel) for reading the KIP clock in microseconds. If the kernel is configured for a fine-grained KIP clock (CONFIG_SYNC↔TSC enabled for IA32, ARM_SYNC_CLOCK for ARM), this code provides the KIP clock with microseconds granularity and accuracy by reading the hardware clock used by the kernel and transforming this value into microseconds. Otherwise this code just reads the KIP clock value.
L4_KIP_OFFS_READ_NS	Offset of KIP code (provided by the kernel) for reading the time stamp counter and transforming this value into nanoseconds. If the kernel is configured for fine-grained KIP clock (CONFIG_SYNC enabled for IA32, ARM_SYNC_CLOCK for ARM), this code provides the KIP clock with nanoseconds granularity and accuracy by reading the hardware clock used by the kernel and transforming this value into nanoseconds. Otherwise this code just reads the KIP clock value and multiplies it by 1000.

Definition at line 95 of file [kip.h](#).

#### 14.1.11.4 Function Documentation

##### 14.1.11.4.1 l4\_kernel\_info\_version\_offset()

```
int l4_kernel_info_version_offset (  
    l4_kernel_info_t const * kip)  [inline]
```

Return offset in bytes of version\_strings relative to the KIP base.

#### Parameters

---

<i>kip</i>	Pointer to the kernel info page (KIP).
------------	----------------------------------------

#### Returns

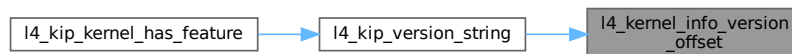
offset of version\_strings relative to the KIP base address, in bytes.

Definition at line 238 of file [kip.h](#).

References [L4\\_NOTHROW](#).

Referenced by [l4\\_kip\\_version\\_string\(\)](#).

Here is the caller graph for this function:



#### 14.1.11.4.2 l4\_kip()

```
l4_kernel_info_t const * l4_kip (
    void ) [inline]
```

Get Kernel Info Page.

#### Returns

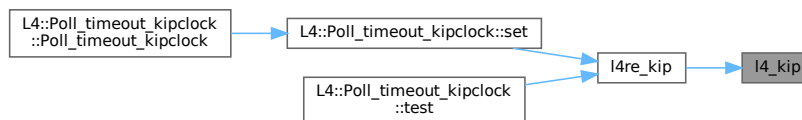
Pointer to Kernel Info Page (KIP) structure.

Definition at line 226 of file [kip.h](#).

References [L4\\_NOTHROW](#).

Referenced by [l4re\\_kip\(\)](#).

Here is the caller graph for this function:



#### 14.1.11.4.3 l4\_kip\_clock()

```
l4_cpu_time_t l4_kip_clock (
    l4_kernel_info_t const * kip) [inline]
```

Return clock value from the KIP.

#### Parameters



<i>kip</i>	Pointer to the kernel info page (KIP).
------------	----------------------------------------

### Returns

Value of the clock field in the KIP.

The KIP clock always contains the current (relative) time in micro seconds independently of the CPU frequency. The clock is only guaranteed to be accurate within the scheduling granularity announced in the KIP.

This function basically calls the KIP code for reading the KIP clock with microseconds resolution. The accuracy depends on the platform and the kernel configuration.

### See also

[L4\\_KIP\\_OFFS\\_READ\\_US](#).

### Examples

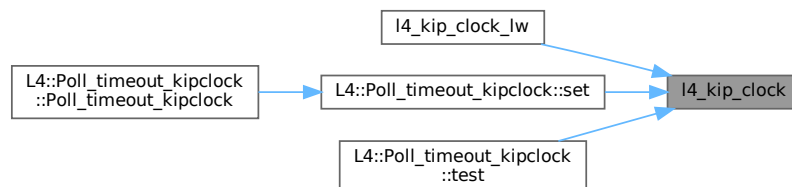
[examples/libs/shmc/prodcons.c](#).

Definition at line 242 of file [kip.h](#).

References [L4\\_KIP\\_OFFS\\_READ\\_US](#), and [L4\\_NOTHROW](#).

Referenced by [l4\\_kip\\_clock\\_lw\(\)](#), [L4::Poll\\_timeout\\_kipclock::set\(\)](#), and [L4::Poll\\_timeout\\_kipclock::test\(\)](#).

Here is the caller graph for this function:



#### 14.1.11.4.4 l4\_kip\_clock\_lw()

```
l4_umword_t l4_kip_clock_lw (
    l4_kernel_info_t const * kip) [inline]
```

Return least significant machine word of clock value from the KIP.

### Parameters

<i>kip</i>	Pointer to the kernel info page (KIP).
------------	----------------------------------------

#### Returns

Lower machine word of clock value from the KIP.

**Deprecated** Use [l4\\_kip\\_clock\(\)](#) instead.

This function will always provide the least significant machine word of the clock value from the KIP, regardless of the kernel configuration.

Definition at line 261 of file [kip.h](#).

References [l4\\_kip\\_clock\(\)](#), and [L4\\_NOTHROW](#).

Here is the call graph for this function:



#### 14.1.11.4.5 l4\_kip\_clock\_ns()

```
l4_cpu_time_t l4_kip_clock_ns (
    l4_kernel_info_t const * kip) [inline]
```

Return current clock using the KIP in nanoseconds.

#### Parameters

<i>kip</i>	Pointer to the kernel info page (KIP).
------------	----------------------------------------

#### Returns

Value of the current clock in nanoseconds.

This function basically calls the KIP code for reading the KIP clock with nanoseconds resolution. The accuracy depends on the platform and the kernel configuration.

#### See also

[L4\\_KIP\\_OFFS\\_READ\\_NS](#).

Definition at line 252 of file [kip.h](#).

References [L4\\_KIP\\_OFFS\\_READ\\_NS](#), and [L4\\_NOTHROW](#).

#### 14.1.11.4.6 l4\_kip\_version()

```
l4_umword_t l4_kip_version (  
    l4_kernel_info_t const * kip)  [inline]
```

Get the kernel version.

#### Parameters

---

<i>kip</i>	Kernel Info Page.
------------	-------------------

**Returns**

Kernel version string. 0 if KIP could not be mapped.

Definition at line 230 of file [kip.h](#).

References [L4\\_NOTHROW](#).

**14.1.11.4.7 l4\_kip\_version\_string()**

```
const char * l4_kip_version_string (
    l4_kernel_info_t const * kip) [inline]
```

Get the kernel version string.

**Parameters**

<i>kip</i>	Kernel Info Page.
------------	-------------------

**Returns**

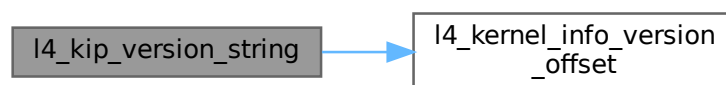
Kernel version string.

Definition at line 234 of file [kip.h](#).

References [l4\\_kernel\\_info\\_version\\_offset\(\)](#), and [L4\\_NOTHROW](#).

Referenced by [l4\\_kip\\_kernel\\_has\\_feature\(\)](#).

Here is the call graph for this function:



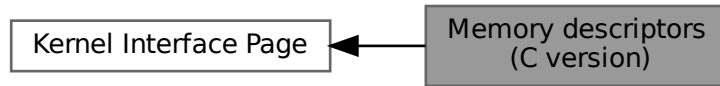
Here is the caller graph for this function:



#### 14.1.11.5 Memory descriptors (C version)

C Interface for KIP memory descriptors.

Collaboration diagram for Memory descriptors (C version):



#### Data Structures

- struct `l4_kernel_info_mem_desc_t`  
*Memory descriptor data structure.*

#### Typedefs

- typedef struct `l4_kernel_info_mem_desc_t` `l4_kernel_info_mem_desc_t`  
*Memory descriptor data structure.*

#### Enumerations

- enum `l4_mem_type_t` {  
`l4_mem_type_undefined` = 0x0 , `l4_mem_type_conventional` = 0x1 , `l4_mem_type_reserved` = 0x2 ,  
`l4_mem_type_dedicated` = 0x3 ,  
`l4_mem_type_shared` = 0x4 , `l4_mem_type_info` = 0xd , `l4_mem_type_bootloader` = 0xe , `l4_mem_type_archspecific`  
= 0xf }  
*Type of a memory descriptor.*
- enum `l4_mem_info_sub_type_t` { `l4_mem_info_acpi_rsdp` = 0 , `l4_mem_reserved_kernel` = 0 ,  
`l4_mem_reserved_heap` = 1 , `l4_mem_reserved_mmio` = 2 }  
*Memory sub types for l4\_mem\_type\_info descriptors.*
- enum `l4_mem_archspecific_sub_type_common_t` { `l4_mem_archspecific_acpi_tables` = 3 , `l4_mem_archspecific_acpi_nvs`  
= 4 }  
*Memory sub types for l4\_mem\_type\_archspecific descriptors.*

## Functions

- `l4_kernel_info_mem_desc_t * l4_kernel_info_get_mem_descs (l4_kernel_info_t *kip) L4_NOTHROW`  
*Get pointer to memory descriptors from KIP.*
- `unsigned l4_kernel_info_get_num_mem_descs (l4_kernel_info_t *kip) L4_NOTHROW`  
*Get number of memory descriptors in KIP.*
- `void l4_kernel_info_set_mem_desc (l4_kernel_info_mem_desc_t *md, l4_addr_t start, l4_addr_t end, unsigned type, unsigned virt, unsigned sub_type) L4_NOTHROW`  
*Populate a memory descriptor.*
- `l4_umword_t l4_kernel_info_get_mem_desc_start (l4_kernel_info_mem_desc_t *md) L4_NOTHROW`  
*Get start address of the region described by the memory descriptor.*
- `l4_umword_t l4_kernel_info_get_mem_desc_end (l4_kernel_info_mem_desc_t *md) L4_NOTHROW`  
*Get end address of the region described by the memory descriptor.*
- `l4_umword_t l4_kernel_info_get_mem_desc_type (l4_kernel_info_mem_desc_t *md) L4_NOTHROW`  
*Get type of the memory region.*
- `l4_umword_t l4_kernel_info_get_mem_desc_subtype (l4_kernel_info_mem_desc_t *md) L4_NOTHROW`  
*Get sub-type of memory region.*
- `l4_umword_t l4_kernel_info_get_mem_desc_is_virtual (l4_kernel_info_mem_desc_t *md) L4_NOTHROW`  
*Get virtual flag of the memory descriptor.*

### 14.1.11.5.1 Detailed Description

C Interface for KIP memory descriptors.

#### Include File

```
#include <l4/sys/memdesc.h>
```

This module contains the C functions to access the memory descriptor in the kernel interface page (KIP).

### 14.1.11.5.2 Typedef Documentation

#### 14.1.11.5.2.1 l4\_kernel\_info\_mem\_desc\_t

```
typedef struct l4_kernel_info_mem_desc_t l4_kernel_info_mem_desc_t
```

Memory descriptor data structure.

#### Note

This data type is opaque, and must be accessed by the accessor functions defined in this module.

### 14.1.11.5.3 Enumeration Type Documentation

#### 14.1.11.5.3.1 l4\_mem\_archspecific\_sub\_type\_common\_t

```
enum l4_mem_archspecific_sub_type_common_t
```

Memory sub types for l4\_mem\_type\_archspecific descriptors.

#### Enumerator

<code>l4_mem_archspecific_acpi_tables</code>	Firmware ACPI tables.
<code>l4_mem_archspecific_acpi_nvs</code>	Firmware reserved address space.

Definition at line 63 of file [memdesc.h](#).

#### 14.1.11.5.3.2 `l4_mem_info_sub_type_t`

```
enum l4_mem_info_sub_type_t
```

Memory sub types for `l4_mem_type_info` descriptors.

##### Enumerator

<code>l4_mem_info_acpi_rsdp</code>	Physical address of the ACPI root pointer.
<code>l4_mem_reserved_kernel</code>	Kernel image.
<code>l4_mem_reserved_heap</code>	Kernel heap.
<code>l4_mem_reserved_mmio</code>	MMIO range reserved by kernel.

Definition at line 50 of file [memdesc.h](#).

#### 14.1.11.5.3.3 `l4_mem_type_t`

```
enum l4_mem_type_t
```

Type of a memory descriptor.

##### Enumerator

<code>l4_mem_type_undefined</code>	Undefined, unused descriptor.
<code>l4_mem_type_conventional</code>	Conventional memory.
<code>l4_mem_type_reserved</code>	Reserved memory for kernel etc.
<code>l4_mem_type_dedicated</code>	Dedicated memory (some device memory).
<code>l4_mem_type_shared</code>	Shared memory (not implemented).
<code>l4_mem_type_info</code>	Info from the boot loader.
<code>l4_mem_type_bootloader</code>	Memory owned by the boot loader.
<code>l4_mem_type_archspecific</code>	Architecture specific memory (e.g., ACPI memory).

Definition at line 33 of file [memdesc.h](#).

#### 14.1.11.5.4 Function Documentation

##### 14.1.11.5.4.1 l4\_kernel\_info\_get\_mem\_desc\_end()

```
l4_umword_t l4_kernel_info_get_mem_desc_end (  
    l4_kernel_info_mem_desc_t * md) [inline]
```

Get end address of the region described by the memory descriptor.

#### Returns

End address.

Definition at line 219 of file [memdesc.h](#).

References [L4\\_NOTHROW](#).

##### 14.1.11.5.4.2 l4\_kernel\_info\_get\_mem\_desc\_is\_virtual()

```
l4_umword_t l4_kernel_info_get_mem_desc_is_virtual (  
    l4_kernel_info_mem_desc_t * md) [inline]
```

Get virtual flag of the memory descriptor.

#### Returns

1 if region is virtual memory, 0 if region is physical memory

Definition at line 240 of file [memdesc.h](#).

References [L4\\_NOTHROW](#).

##### 14.1.11.5.4.3 l4\_kernel\_info\_get\_mem\_desc\_start()

```
l4_umword_t l4_kernel_info_get_mem_desc_start (  
    l4_kernel_info_mem_desc_t * md) [inline]
```

Get start address of the region described by the memory descriptor.

#### Returns

Start address.

Definition at line 212 of file [memdesc.h](#).

References [L4\\_NOTHROW](#).



#### 14.1.11.5.4.4 l4\_kernel\_info\_get\_mem\_desc\_subtype()

```
l4_umword_t l4_kernel_info_get_mem_desc_subtype (  
    l4_kernel_info_mem_desc_t * md) [inline]
```

Get sub-type of memory region.

##### Returns

Sub-type.

The sub type is defined for architecture specific memory descriptors (see [l4\\_mem\\_type\\_archspecific](#)) and has architecture specific meaning.

Definition at line 233 of file [memdesc.h](#).

References [L4\\_NOTHROW](#).

#### 14.1.11.5.4.5 l4\_kernel\_info\_get\_mem\_desc\_type()

```
l4_umword_t l4_kernel_info_get_mem_desc_type (  
    l4_kernel_info_mem_desc_t * md) [inline]
```

Get type of the memory region.

##### Returns

Type of the region (see [l4\\_mem\\_type\\_t](#)).

Definition at line 226 of file [memdesc.h](#).

References [L4\\_NOTHROW](#).

#### 14.1.11.5.4.6 l4\_kernel\_info\_get\_num\_mem\_descs()

```
unsigned l4_kernel_info_get_num_mem_descs (  
    l4_kernel_info_t * kip) [inline]
```

Get number of memory descriptors in KIP.

##### Returns

Number of memory descriptors.

Definition at line 190 of file [memdesc.h](#).

References [L4\\_NOTHROW](#).

#### 14.1.11.5.4.7 l4\_kernel\_info\_set\_mem\_desc()

```
void l4_kernel_info_set_mem_desc (  
    l4_kernel_info_mem_desc_t * md,  
    l4_addr_t start,  
    l4_addr_t end,  
    unsigned type,  
    unsigned virt,  
    unsigned sub_type) [inline]
```

Populate a memory descriptor.

##### Parameters

---

<i>md</i>	Pointer to memory descriptor
<i>start</i>	Start of region
<i>end</i>	End of region
<i>type</i>	Type of region
<i>virt</i>	1 if virtual region, 0 if physical region
<i>sub_type</i>	Sub type.

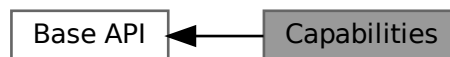
Definition at line 197 of file [memdesc.h](#).

References [L4\\_NOTHROW](#).

### 14.1.12 Capabilities

C interface for capabilities.

Collaboration diagram for Capabilities:



#### Typedefs

- typedef unsigned long [l4\\_cap\\_idx\\_t](#)  
*Capability selector type.*

#### Enumerations

- enum [l4\\_cap\\_consts\\_t](#) {  
[L4\\_CAP\\_SHIFT](#) , [L4\\_CAP\\_SIZE](#) = 1UL << [L4\\_CAP\\_SHIFT](#) , [L4\\_CAP\\_OFFSET](#) , [L4\\_CAP\\_MASK](#) ,  
[L4\\_INVALID\\_CAP](#) , [L4\\_INVALID\\_CAP\\_BIT](#) = 1UL << ([L4\\_CAP\\_SHIFT](#) - 1) }  
*Constants related to capability selectors.*
- enum [l4\\_default\\_caps\\_t](#) {  
[L4\\_BASE\\_TASK\\_CAP](#) , [L4\\_BASE\\_FACTORY\\_CAP](#) , [L4\\_BASE\\_THREAD\\_CAP](#) , [L4\\_BASE\\_PAGER\\_CAP](#) ,  
[L4\\_BASE\\_LOG\\_CAP](#) , [L4\\_BASE\\_ICU\\_CAP](#) , [L4\\_BASE\\_SCHEDULER\\_CAP](#) , [L4\\_BASE\\_IOMMU\\_CAP](#) ,  
[L4\\_BASE\\_DEBUGGER\\_CAP](#) , [L4\\_BASE\\_ARM\\_SMCCC\\_CAP](#) , [L4\\_BASE\\_CAPS\\_LAST\\_P1](#) , [L4\\_BASE\\_CAPS\\_LAST](#)  
= [L4\\_BASE\\_CAPS\\_LAST\\_P1](#) - 1 }  
*Default capabilities setup for the initial tasks.*

## Functions

- unsigned [l4\\_is\\_invalid\\_cap](#) ([l4\\_cap\\_idx\\_t](#) c) [L4\\_NOTHROW](#)  
*Test if a capability selector is the invalid capability.*
- unsigned [l4\\_is\\_valid\\_cap](#) ([l4\\_cap\\_idx\\_t](#) c) [L4\\_NOTHROW](#)  
*Test if a capability selector is a valid selector.*
- unsigned [l4\\_capability\\_equal](#) ([l4\\_cap\\_idx\\_t](#) c1, [l4\\_cap\\_idx\\_t](#) c2) [L4\\_NOTHROW](#)  
*Test if the capability indices of two capability selectors are equal.*

### 14.1.12.1 Detailed Description

C interface for capabilities.

Add

```
#include <l4/sys/types.h>
#include <l4/sys/consts.h>
```

to your code to use the functions and definitions explained here.

### 14.1.12.2 Typedef Documentation

#### 14.1.12.2.1 [l4\\_cap\\_idx\\_t](#)

```
typedef unsigned long l4\_cap\_idx\_t
```

Capability selector type.

A capability selector is either a (shifted) capability index or the invalid capability selector [L4\\_INVALID\\_CAP](#).

Usage of the invalid capability selector is defined only for invoking IPC (see [Object Invocation](#)): When IPC is invoked on [L4\\_INVALID\\_CAP](#), then it is resolved to a capability for the current thread with full permissions.

Otherwise, the API assumes that each argument of type [l4\\_cap\\_idx\\_t](#) is a capability index, i.e., `idx << L4\_CAP\_SHIFT` for arbitrary `idx`. The behavior for other arguments is then undefined.

#### Examples

[examples/libs/shmc/prodcons.c](#), [examples/sys/aliens/main.c](#), [examples/sys/isr/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line [336](#) of file [types.h](#).

### 14.1.12.3 Enumeration Type Documentation

#### 14.1.12.3.1 [l4\\_cap\\_consts\\_t](#)

```
enum l4\_cap\_consts\_t
```

Constants related to capability selectors.

#### Enumerator

L4_CAP_SHIFT	Capability index shift.
L4_CAP_SIZE	<b>Deprecated</b> Superseded by <a href="#">L4_CAP_OFFSET</a> .
L4_CAP_OFFSET	Offset of two consecutive capability selectors.
L4_CAP_MASK	Mask to get only the relevant bits of an <a href="#">l4_cap_idx_t</a> .
L4_INVALID_CAP	Invalid capability selector.

Definition at line 139 of file [consts.h](#).

#### 14.1.12.3.2 l4\_default\_caps\_t

```
enum l4_default_caps_t
```

Default capabilities setup for the initial tasks.

These capability selectors are setup per default by the micro kernel for the two initial tasks, the Root-Pager (Sigma0) and the Root-Task (Moe).

##### Attention

These constants do not have any particular meaning for applications started by Moe, see [Initial Environment](#) for this kind of information.

##### See also

[Initial Environment](#) for information useful for normal user applications.

#### Enumerator

L4_BASE_TASK_CAP	Capability selector for the current task.
L4_BASE_FACTORY_CAP	Capability selector for the factory.
L4_BASE_THREAD_CAP	Capability selector for the first thread.
L4_BASE_PAGER_CAP	Capability selector for the pager gate. For Sigma0, the pager is not present since it never raises page faults. For Moe, the pager is set to Sigma0.
L4_BASE_LOG_CAP	Capability selector for the log object. Present if the corresponding feature is turned on in the microkernel configuration.
L4_BASE_ICU_CAP	Capability selector for the base icu object.
L4_BASE_SCHEDULER_CAP	Capability selector for the scheduler cap.
L4_BASE_IOMMU_CAP	Capability selector for the IO-MMU cap. Present if the microkernel detected an IO-MMU.
L4_BASE_DEBUGGER_CAP	Capability selector for the debugger cap. Present if the corresponding feature is turned on in the microkernel configuration.
L4_BASE_ARM_SMCCC_CAP	Capability selector for the ARM SMCCC cap. Present if the microkernel detected an ARM SMC capable trusted execution environment.
L4_BASE_CAPS_LAST	Last capability index used for base capabilities.

Definition at line 324 of file [consts.h](#).

#### 14.1.12.4 Function Documentation

##### 14.1.12.4.1 `l4_capability_equal()`

```
unsigned l4_capability_equal (  
    l4_cap_idx_t c1,  
    l4_cap_idx_t c2) [inline]
```

Test if the capability indices of two capability selectors are equal.

##### Parameters

<i>c1</i>	Capability selector.
<i>c2</i>	Capability selector.

##### Return values

0	The index parts of the capability selectors differ.
1	The index parts of the capability selectors are equal.

##### Precondition

Both capability selectors must be valid (cf. [l4\\_is\\_valid\\_cap\(\)](#)) otherwise the return value is undefined.

Definition at line 397 of file [types.h](#).

References [L4\\_CAP\\_SHIFT](#), and [L4\\_NOTHROW](#).

##### 14.1.12.4.2 `l4_is_invalid_cap()`

```
unsigned l4_is_invalid_cap (  
    l4_cap_idx_t c) [inline]
```

Test if a capability selector is the invalid capability.

##### Parameters

<i>c</i>	Capability selector
----------	---------------------

##### Return values

0	The capability selector is not the invalid capability.
>0	The capability selector is the invalid capability.

##### Examples

[examples/libs/l4re/c/ma+rm.c](#), [examples/sys/aliens/main.c](#), [examples/sys/isr/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 389 of file [types.h](#).

References [L4\\_NOTHROW](#).

#### 14.1.12.4.3 l4\_is\_valid\_cap()

```
unsigned l4_is_valid_cap (
    l4_cap_idx_t c) [inline]
```

Test if a capability selector is a valid selector.

##### Parameters

<i>c</i>	Capability selector
----------	---------------------

##### Return values

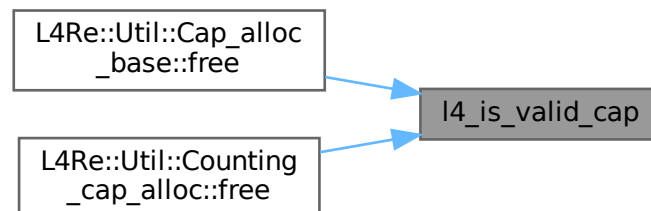
0	The capability selector is not valid.
>0	The capability selector is valid.

Definition at line 393 of file [types.h](#).

References [L4\\_NOTHROW](#).

Referenced by [L4Re::Util::Cap\\_alloc\\_base::free\(\)](#), and [L4Re::Util::Counting\\_cap\\_alloc< COUNTERTYPE, Dbg >::free\(\)](#).

Here is the caller graph for this function:



#### 14.1.13 Memory operations.

Operations for memory access.

Collaboration diagram for Memory operations.:



## Enumerations

- enum `L4_mem_op_widths` { `L4_MEM_WIDTH_1BYTE` = 0 , `L4_MEM_WIDTH_2BYTE` = 1 , `L4_MEM_WIDTH_4BYTE` = 2 }
- Memory access width definitions.*

## Functions

- unsigned long `l4_mem_read` (unsigned long *virtaddress*, unsigned *width*)  
*Read user task memory from kernel privilege level.*
- void `l4_mem_write` (unsigned long *virtaddress*, unsigned *width*, unsigned long *value*)  
*Write user task memory from kernel privilege level.*

### 14.1.13.1 Detailed Description

Operations for memory access.

This module provides functionality to access user task memory from the kernel. This is needed for some devices that are only accessible from privileged processor mode. Only use this when absolutely required. This functionality is only available on the ARM architecture.

```
#include <l4/sys/mem_op.h>
```

### 14.1.13.2 Enumeration Type Documentation

#### 14.1.13.2.1 L4\_mem\_op\_widths

```
enum L4_mem_op_widths
```

Memory access width definitions.

#### Enumerator

<code>L4_MEM_WIDTH_1BYTE</code>	Access one byte (8-bit width).
<code>L4_MEM_WIDTH_2BYTE</code>	Access two bytes (16-bit width).
<code>L4_MEM_WIDTH_4BYTE</code>	Access four bytes (32-bit width).

Definition at line 40 of file `mem_op.h`.

### 14.1.13.3 Function Documentation

#### 14.1.13.3.1 l4\_mem\_read()

```
unsigned long l4_mem_read (
    unsigned long virtaddress,
    unsigned width) [inline]
```

Read user task memory from kernel privilege level.

#### Parameters

---

<i>virtaddress</i>	Virtual address in the calling task.
<i>width</i>	Width of access in bytes in log2,

See also

[L4\\_mem\\_op\\_widths](#)

Returns

Read value.

Upon an given invalid address or invalid width value the function does nothing.

Definition at line 130 of file [mem\\_op.h](#).

References [l4\\_mem\\_arm\\_op\\_call\(\)](#).

Here is the call graph for this function:



#### 14.1.13.3.2 l4\_mem\_write()

```
void l4_mem_write (  
    unsigned long virtaddress,  
    unsigned width,  
    unsigned long value) [inline]
```

Write user task memory from kernel privilege level.

Parameters

<i>virtaddress</i>	Virtual address in the calling task.
<i>width</i>	Width of access in bytes in log2 (i.e. allowed values: 0, 1, 2)
<i>value</i>	Value to write.

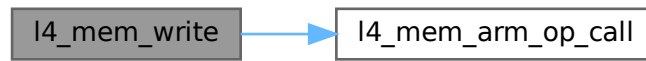
Upon an given invalid address or invalid width value the function does nothing.

Definition at line 136 of file [mem\\_op.h](#).

References [l4\\_mem\\_arm\\_op\\_call\(\)](#).

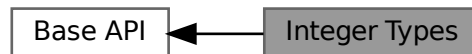


Here is the call graph for this function:



### 14.1.14 Integer Types

Collaboration diagram for Integer Types:



#### Files

- file [l4int.h](#)  
*Fixed sized integer types, generic version.*
- file [l4int.h](#)  
*Fixed sized integer types, arm version.*
- file [l4int.h](#)  
*Fixed sized integer types, arm version.*
- file [l4int.h](#)  
*Fixed sized integer types, AMD64 version.*
- file [l4int.h](#)  
*Fixed sized integer types, x86 version.*

#### Macros

- `#define L4_MWORD_BITS 32`  
*Size of machine words in bits.*
- `#define L4_MWORD_BITS 64`  
*Size of machine words in bits.*
- `#define L4_MWORD_BITS 64`  
*Size of machine words in bits.*
- `#define L4_MWORD_BITS 32`  
*Size of machine words in bits.*

## Typedefs

- typedef signed char **l4\_int8\_t**  
*Signed 8bit value.*
- typedef unsigned char **l4\_uint8\_t**  
*Unsigned 8bit value.*
- typedef signed short int **l4\_int16\_t**  
*Signed 16bit value.*
- typedef unsigned short int **l4\_uint16\_t**  
*Unsigned 16bit value.*
- typedef signed int **l4\_int32\_t**  
*Signed 32bit value.*
- typedef unsigned int **l4\_uint32\_t**  
*Unsigned 32bit value.*
- typedef signed long long **l4\_int64\_t**  
*Signed 64bit value.*
- typedef unsigned long long **l4\_uint64\_t**  
*Unsigned 64bit value.*
- typedef unsigned long **l4\_addr\_t**  
*Address type.*
- typedef signed long **l4\_mword\_t**  
*Signed machine word.*
- typedef unsigned long **l4\_umword\_t**  
*Unsigned machine word.*
- typedef [l4\\_uint64\\_t](#) **l4\_cpu\_time\_t**  
*CPU clock type.*
- typedef [l4\\_uint64\\_t](#) **l4\_kernel\_clock\_t**  
*Kernel clock type.*
- typedef unsigned int **l4\_size\_t**  
*Unsigned size type.*
- typedef signed int **l4\_ssize\_t**  
*Signed size type.*
- typedef unsigned long **l4\_size\_t**  
*Unsigned size type.*
- typedef signed long **l4\_ssize\_t**  
*Signed size type.*
- typedef unsigned long **l4\_size\_t**  
*Unsigned size type.*
- typedef signed long **l4\_ssize\_t**  
*Signed size type.*
- typedef unsigned int **l4\_size\_t**  
*Unsigned size type.*
- typedef signed int **l4\_ssize\_t**  
*Signed size type.*

### 14.1.14.1 Detailed Description

#### Include File

```
#include <l4/sys/l4int.h>
```

## 14.2 EDID parsing functionality

### Enumerations

- enum `Libedid_consts` { `Libedid_block_size` = 128 }  
*EDID constants.*

### Functions

- int `libedid_check_header` (const unsigned char \*edid)  
*Check for valid EDID header.*
- int `libedid_checksum` (const unsigned char \*edid)  
*Calculates the EDID checksum.*
- unsigned `libedid_version` (const unsigned char \*edid)  
*Returns the EDID version number.*
- unsigned `libedid_revision` (const unsigned char \*edid)  
*Returns the EDID revision number.*
- void `libedid_pnp_id` (const unsigned char \*edid, unsigned char \*id)  
*Extracts the display's PnP ID.*
- void `libedid_prefered_resolution` (const unsigned char \*edid, unsigned \*w, unsigned \*h)  
*Extract the display's prefered mode.*
- unsigned `libedid_num_ext_blocks` (const unsigned char \*edid)  
*Get the number of EDID extension blocks.*
- unsigned `libedid_dump_standard_timings` (const unsigned char \*edid)  
*Dump the standard timings to stdout.*
- void `libedid_dump` (const unsigned char \*edid)  
*Dump raw EDID data to stdout.*

### 14.2.1 Detailed Description

### 14.2.2 Enumeration Type Documentation

#### 14.2.2.1 Libedid\_consts

enum `Libedid_consts`

EDID constants.

#### Enumerator

<code>Libedid_block_size</code>	Size of one EDID block in bytes.
---------------------------------	----------------------------------

Definition at line 23 of file `edid.h`.

### 14.2.3 Function Documentation

#### 14.2.3.1 libedid\_check\_header()

```
int libedid_check_header (
    const unsigned char * edid)
```

Check for valid EDID header.

#### Parameters

---

<i>edid</i>	Pointer to a 128byte EDID block
-------------	---------------------------------

**Returns**

0 if the header is correct, -EINVAL otherwise

**14.2.3.2 libedid\_checksum()**

```
int libedid_checksum (  
    const unsigned char * edid)
```

Calculates the EDID checksum.

**Parameters**

<i>edid</i>	Pointer to a 128byte EDID block
-------------	---------------------------------

**Returns**

0 if checksum is correct, -EINVAL otherwise

**14.2.3.3 libedid\_dump()**

```
void libedid_dump (  
    const unsigned char * edid)
```

Dump raw EDID data to stdout.

**Parameters**

<i>edid</i>	Pointer to a 128byte EDID block
-------------	---------------------------------

**14.2.3.4 libedid\_dump\_standard\_timings()**

```
unsigned libedid_dump_standard_timings (  
    const unsigned char * edid)
```

Dump the standard timings to stdout.

**Parameters**

<i>edid</i>	Pointer to a 128byte EDID block
-------------	---------------------------------

**Returns**

Number of standard timings stored in EDID

#### 14.2.3.5 libedid\_num\_ext\_blocks()

```
unsigned libedid_num_ext_blocks (  
    const unsigned char * edid)
```

Get the number of EDID extension blocks.

##### Parameters

---

<i>edid</i>	Pointer to a 128byte EDID block
-------------	---------------------------------

**Returns**

Number of EDID extension blocks

**14.2.3.6 libedid\_pnp\_id()**

```
void libedid_pnp_id (  
    const unsigned char * edid,  
    unsigned char * id)
```

Extracts the display's PnP ID.

**Parameters**

	<i>edid</i>	Pointer to a 128byte EDID block
out	<i>id</i>	Return the PnP id. Must point to 4 bytes.

**14.2.3.7 libedid\_preferred\_resolution()**

```
void libedid_preferred_resolution (  
    const unsigned char * edid,  
    unsigned * w,  
    unsigned * h)
```

Extract the display's preferred mode.

**Parameters**

	<i>edid</i>	Pointer to a 128byte EDID block
out	<i>w</i>	X resolution of preferred video mode in pixels.
out	<i>h</i>	Y resolution of preferred video mode in pixels.

**14.2.3.8 libedid\_revision()**

```
unsigned libedid_revision (  
    const unsigned char * edid)
```

Returns the EDID revision number.

**Parameters**

<i>edid</i>	Pointer to a 128 EDID block
-------------	-----------------------------

**Returns**

Revision number

### 14.2.3.9 libedid\_version()

```
unsigned libedid_version (
    const unsigned char * edid)
```

Returns the EDID version number.

#### Parameters

<i>edid</i>	Pointer to a 128byte EDID block
-------------	---------------------------------

#### Returns

Version number

## 14.3 IO interface

#### Typedefs

- typedef [l4vbus\\_resource\\_t](#) [l4io\\_resource\\_t](#)  
*Resource descriptor.*
- typedef [l4vbus\\_device\\_t](#) [l4io\\_device\\_t](#)  
*Device descriptor.*

#### Enumerations

- enum [l4io\\_iomem\\_flags\\_t](#) {  
[L4IO\\_MEM\\_NONCACHED](#) = 0 , [L4IO\\_MEM\\_CACHED](#) = 1 , [L4IO\\_MEM\\_USE\\_MTRR](#) = 2 , [L4IO\\_MEM\\_ATTR\\_MASK](#) = 0xf ,  
[L4IO\\_MEM\\_WRITE\\_COMBINED](#) = [L4IO\\_MEM\\_USE\\_MTRR](#) | [L4IO\\_MEM\\_CACHED](#) , [L4IO\\_MEM\\_USE\\_RESERVED\\_AREA](#) = 0x40 << 8 , [L4IO\\_MEM\\_EAGER\\_MAP](#) = 0x80 << 8 }  
*Flags for IO memory.*
- enum [l4io\\_device\\_types\\_t](#) {  
[L4IO\\_DEVICE\\_INVALID](#) = 0 , [L4IO\\_DEVICE\\_PCI](#) , [L4IO\\_DEVICE\\_USB](#) , [L4IO\\_DEVICE\\_OTHER](#) ,  
[L4IO\\_DEVICE\\_ANY](#) = ~0 }  
*Device types.*
- enum [l4io\\_resource\\_types\\_t](#) {  
[L4IO\\_RESOURCE\\_INVALID](#) = [L4VBUS\\_RESOURCE\\_INVALID](#) , [L4IO\\_RESOURCE\\_IRQ](#) = [L4VBUS\\_RESOURCE\\_IRQ](#) , [L4IO\\_RESOURCE\\_MEM](#) = [L4VBUS\\_RESOURCE\\_MEM](#) , [L4IO\\_RESOURCE\\_PORT](#) = [L4VBUS\\_RESOURCE\\_PORT](#) ,  
[L4IO\\_RESOURCE\\_ANY](#) = ~0 }  
*Resource types.*

## Functions

- long [l4io\\_request\\_iomem](#) ([l4\\_addr\\_t](#) phys, unsigned long size, int flags, [l4\\_addr\\_t](#) \*virt)  
*Request an IO memory region.*
- long [l4io\\_request\\_iomem\\_region](#) ([l4\\_addr\\_t](#) phys, [l4\\_addr\\_t](#) virt, unsigned long size, int flags)  
*Request an IO memory region and map it to a specified region.*
- long [l4io\\_release\\_iomem](#) ([l4\\_addr\\_t](#) virt, unsigned long size)  
*Release an IO memory region.*
- long [l4io\\_request\\_ioport](#) (unsigned portnum, unsigned len)  
*Request an IO port region.*
- long [l4io\\_release\\_ioport](#) (unsigned portnum, unsigned len)  
*Release an IO port region.*
- int [l4io\\_lookup\\_device](#) (const char \*devname, [l4io\\_device\\_handle\\_t](#) \*dev\_handle, [l4io\\_device\\_t](#) \*dev, [l4io\\_resource\\_handle\\_t](#) \*res\_handle)  
*Find a device by name.*
- int [l4io\\_lookup\\_resource](#) ([l4io\\_device\\_handle\\_t](#) devhandle, enum [l4io\\_resource\\_types\\_t](#) type, [l4io\\_resource\\_handle\\_t](#) \*reshandle, [l4io\\_resource\\_t](#) \*res)  
*Request a specific resource from a device description.*
- [l4\\_addr\\_t](#) [l4io\\_request\\_resource\\_iomem](#) ([l4io\\_device\\_handle\\_t](#) devhandle, [l4io\\_resource\\_handle\\_t](#) \*reshandle)  
*Request IO memory.*
- int [l4io\\_has\\_resource](#) (enum [l4io\\_resource\\_types\\_t](#) type, [l4vbus\\_paddr\\_t](#) start, [l4vbus\\_paddr\\_t](#) end)  
*Check if a resource is available.*

### 14.3.1 Detailed Description

### 14.3.2 Typedef Documentation

#### 14.3.2.1 [l4io\\_resource\\_t](#)

```
typedef l4vbus\_resource\_t l4io\_resource\_t
```

Resource descriptor.

For IRQ types, the end field is not used, i.e. only a single interrupt can be described with a [l4io\\_resource\\_t](#)

Definition at line 67 of file [types.h](#).

### 14.3.3 Enumeration Type Documentation

#### 14.3.3.1 [l4io\\_device\\_types\\_t](#)

```
enum l4io\_device\_types\_t
```

Device types.

#### Enumerator

---



L4IO_DEVICE_INVALID	Invalid type.
L4IO_DEVICE_PCI	PCI device.
L4IO_DEVICE_USB	USB device.
L4IO_DEVICE_OTHER	Any other device without unique IDs.
L4IO_DEVICE_ANY	any type

Definition at line 36 of file [types.h](#).

#### 14.3.3.2 l4io\_iomem\_flags\_t

enum [l4io\\_iomem\\_flags\\_t](#)

Flags for IO memory.

##### Enumerator

L4IO_MEM_NONCACHED	Non-cache memory.
L4IO_MEM_CACHED	Cache memory.
L4IO_MEM_USE_MTRR	Use MTRR.
L4IO_MEM_USE_RESERVED_AREA	Use reserved area for mapping I/O memory. Flag only valid for <a href="#">l4io_request_iomem_region()</a>
L4IO_MEM_EAGER_MAP	Eagerly map the I/O memory. Passthrough to the l4re-rm.

Definition at line 14 of file [types.h](#).

#### 14.3.3.3 l4io\_resource\_types\_t

enum [l4io\\_resource\\_types\\_t](#)

Resource types.

##### Enumerator

L4IO_RESOURCE_INVALID	Invalid type.
L4IO_RESOURCE_IRQ	Interrupt resource.
L4IO_RESOURCE_MEM	I/O memory resource.
L4IO_RESOURCE_PORT	I/O port resource (x86 only).
L4IO_RESOURCE_ANY	any type

Definition at line 48 of file [types.h](#).

### 14.3.4 Function Documentation

#### 14.3.4.1 l4io\_has\_resource()

```
int l4io_has_resource (
    enum l4io_resource_types_t type,
    l4vbus_paddr_t start,
    l4vbus_paddr_t end)
```

Check if a resource is available.

#### Parameters

---

<i>type</i>	Type of resource
<i>start</i>	Minimal value.
<i>end</i>	Maximum value.

References [L4\\_INLINE](#).

#### 14.3.4.2 l4io\_lookup\_device()

```
int l4io_lookup_device (
    const char * devname,
    l4io_device_handle_t * dev_handle,
    l4io_device_t * dev,
    l4io_resource_handle_t * res_handle)
```

Find a device by name.

##### Parameters

	<i>devname</i>	Name of device.
out	<i>dev_handle</i>	Device handle for found device, can be NULL.
out	<i>dev</i>	Device information, filled by the function, can be NULL.
out	<i>res_handle</i>	Resource handle, can be NULL.

##### Returns

0 on success, error code otherwise

References [L4\\_CV](#), and [L4\\_EXPORT](#).

#### 14.3.4.3 l4io\_lookup\_resource()

```
int l4io_lookup_resource (
    l4io_device_handle_t devhandle,
    enum l4io_resource_types_t type,
    l4io_resource_handle_t * reshandle,
    l4io_resource_t * res)
```

Request a specific resource from a device description.

##### Parameters

	<i>devhandle</i>	Device handle.
	<i>type</i>	Type of resource to request (see <a href="#">l4io_resource_types_t</a> ).
in, out	<i>reshandle</i>	Resource handle, start with handle returned by device functions. The next resource handle is returned here.

<i>out</i>	<i>res</i>	Device descriptor.
------------	------------	--------------------

**Returns**

0 on success, error code otherwise, esp. -L4\_ENOENT if no more resources found

References [L4\\_CV](#), and [L4\\_EXPORT](#).

**14.3.4.4 l4io\_release\_iomem()**

```
long l4io_release_iomem (
    l4_addr_t virt,
    unsigned long size)
```

Release an IO memory region.

**Parameters**

<i>virt</i>	Virtual address of region to free, see <a href="#">l4io_request_iomem</a>
<i>size</i>	Size of the region to release.

**Returns**

0 on success, <0 on error

References [L4\\_CV](#), and [L4\\_EXPORT](#).

**14.3.4.5 l4io\_release\_ioport()**

```
long l4io_release_ioport (
    unsigned portnum,
    unsigned len)
```

Release an IO port region.

**Parameters**

<i>portnum</i>	Start of port range to release
<i>len</i>	Length of range to request

**Returns**

0 on success, <0 on error

**Note**

X86 architecture only

References [L4\\_CV](#), [L4\\_EXPORT](#), and [L4\\_INLINE](#).

#### 14.3.4.6 `l4io_request_iomem()`

```
long l4io_request_iomem (
    l4_addr_t phys,
    unsigned long size,
    int flags,
    l4_addr_t * virt)
```

Request an IO memory region.

##### Parameters

	<i>phys</i>	Physical address of the I/O memory region
	<i>size</i>	Size of the region in Bytes, granularity pages.
	<i>flags</i>	See <a href="#">l4io_iomem_flags_t</a>
<i>in, out</i>	<i>virt</i>	Virtual address where the IO memory region should be mapped to. If the caller passes '0' a region in the caller's address space is searched and the virtual address is returned.

##### Return values

0	Success.
-L4_ENOENT	No area in the caller's address space could be found to map the IO memory region.
-L4_EPERM	Operation not allowed.
-L4_EINVAL	Invalid value.
-L4_EADDRNOTAVAIL	The requested virtual address is not available.
-L4_ENOMEM	The requested IO memory region could not be allocated.
<0	IPC errors.

##### Note

This function uses [L4Re](#) functionality to reserve a part of the virtual address space of the caller.

References [L4\\_CV](#), and [L4\\_EXPORT](#).

#### 14.3.4.7 `l4io_request_iomem_region()`

```
long l4io_request_iomem_region (
    l4_addr_t phys,
    l4_addr_t virt,
    unsigned long size,
    int flags)
```

Request an IO memory region and map it to a specified region.

##### Parameters

<i>phys</i>	Physical address of the I/O memory region
<i>virt</i>	Virtual address.
<i>size</i>	Size of the region in Bytes, granularity pages.
<i>flags</i>	See <a href="#">l4io_iomem_flags_t</a>

### Return values

<i>0</i>	Success.
<i>-L4_ENOENT</i>	No area could be found to map the IO memory region.
<i>-L4_EPERM</i>	Operation not allowed.
<i>-L4_EINVAL</i>	Invalid value.
<i>-L4_EADDRNOTAVAIL</i>	The requested virtual address is not available.
<i>-L4_ENOMEM</i>	The requested IO memory region could not be allocated.
<i>&lt;0</i>	IPC errors.

### Note

This function uses [L4Re](#) functionality to reserve a part of the virtual address space of the caller.

References [L4\\_CV](#), and [L4\\_EXPORT](#).

#### 14.3.4.8 l4io\_request\_ioport()

```
long l4io_request_ioport (
    unsigned portnum,
    unsigned len)
```

Request an IO port region.

### Parameters

<i>portnum</i>	Start of port range to request
<i>len</i>	Length of range to request

### Returns

0 on success, <0 on error

### Note

X86 architecture only

References [L4\\_CV](#), and [L4\\_EXPORT](#).

#### 14.3.4.9 l4io\_request\_resource\_iomem()

```
l4_addr_t l4io_request_resource_iomem (  
    l4io_device_handle_t devhandle,  
    l4io_resource_handle_t * reshandle)
```

Request IO memory.

#### Parameters

---

	<i>devhandle</i>	Device handle.
<i>in, out</i>	<i>reshandle</i>	Resource handle from which IO memory should be requested. Upon successful completion 'reshandle' points to the device's next resource.

### Return values

0	An error occurred. The value of 'reshandle' is undefined.
>0	The virtual address of the IO memory mapping.

References [L4\\_CV](#), and [L4\\_EXPORT](#).

## 14.4 IPC Helpers

### Functions

- void [L4::throw\\_ipc\\_exception](#) ([L4::Cap](#)< void > const &o, [l4\\_msgtag\\_t](#) const &err, [l4\\_utcb\\_t](#) \*utcb)  
*Throw an [L4](#) IPC error as exception.*
- void [L4::throw\\_ipc\\_exception](#) (void const \*o, [l4\\_msgtag\\_t](#) const &err, [l4\\_utcb\\_t](#) \*utcb)  
*Throw an [L4](#) IPC error as exception.*

### 14.4.1 Detailed Description

### 14.4.2 Function Documentation

#### 14.4.2.1 [throw\\_ipc\\_exception\(\)](#) [1/2]

```
void L4::throw_ipc_exception (
    L4::Cap< void > const & o,
    l4\_msgtag\_t const & err,
    l4\_utcb\_t * utcb) [inline]
```

Throw an [L4](#) IPC error as exception.

### Parameters

<i>o</i>	The client side object, for which the IPC was invoked.
<i>err</i>	The IPC result code (error code).
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

Definition at line 34 of file [ipc\\_helper](#).

References [l4\\_msgtag\\_t::has\\_error\(\)](#).

Referenced by [throw\\_ipc\\_exception\(\)](#).



Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.4.2.2 throw\_ipc\_exception() [2/2]

```

void L4::throw_ipc_exception (
    void const * o,
    l4_msgtag_t const & err,
    l4_utcb_t * utcb) [inline]
  
```

Throw an [L4](#) IPC error as exception.

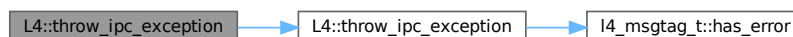
##### Parameters

<i>o</i>	The client side object, for which the IPC was invoked.
<i>err</i>	The IPC result code (error code).
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

Definition at line [50](#) of file [ipc\\_helper](#).

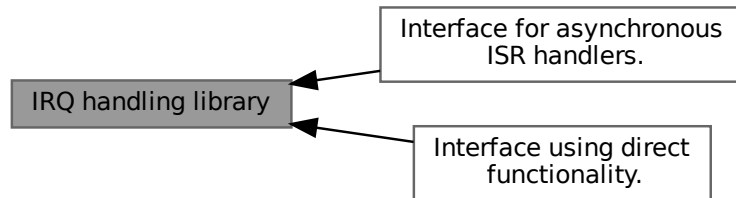
References [throw\\_ipc\\_exception\(\)](#).

Here is the call graph for this function:



## 14.5 IRQ handling library

Collaboration diagram for IRQ handling library:



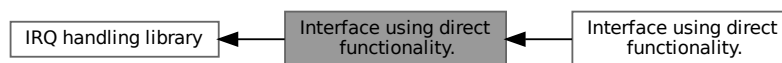
### Topics

- [Interface using direct functionality. . . . . 474](#)
- [Interface for asynchronous ISR handlers. . . . . 481](#)  
*This interface has just two (main) functions.*

### 14.5.1 Detailed Description

### 14.5.2 Interface using direct functionality.

Collaboration diagram for Interface using direct functionality.:



### Topics

- [Interface using direct functionality. . . . . 479](#)

## Functions

- `l4irq_t * l4irq_attach` (int irqnum)  
*Attach/connect to IRQ.*
- `l4irq_t * l4irq_attach_ft` (int irqnum, unsigned mode)  
*Attach/connect to IRQ using given type.*
- `l4irq_t * l4irq_attach_thread` (int irqnum, `l4_cap_idx_t` to\_thread)  
*Attach/connect to IRQ.*
- `l4irq_t * l4irq_attach_thread_ft` (int irqnum, `l4_cap_idx_t` to\_thread, unsigned mode)  
*Attach/connect to IRQ using given type.*
- `long l4irq_wait` (`l4irq_t *irq`)  
*Wait for specified IRQ.*
- `long l4irq_unmask_and_wait_any` (`l4irq_t *unmask_irq`, `l4irq_t **ret_irq`)  
*Unmask a specific IRQ and wait for any attached IRQ.*
- `long l4irq_wait_any` (`l4irq_t **irq`)  
*Wait for any attached IRQ.*
- `long l4irq_unmask` (`l4irq_t *irq`)  
*Unmask a specific IRQ.*
- `long l4irq_detach` (`l4irq_t *irq`)  
*Detach from IRQ.*

### 14.5.2.1 Detailed Description

### 14.5.2.2 Function Documentation

#### 14.5.2.2.1 `l4irq_attach()`

```
l4irq_t * l4irq_attach (
    int irqnum)
```

Attach/connect to IRQ.

#### Parameters

<code>irqnum</code>	IRQ number to request
---------------------	-----------------------

#### Returns

Pointer to `l4irq_t` structure, 0 on error

This `l4irq_attach` has to be called in the same thread as `l4irq_wait` and caller has to be a pthread thread.

#### Examples

[examples/libs/libirq/loop.c](#).

References [L4\\_CV](#).

#### 14.5.2.2.2 `l4irq_attach_ft()`

```
l4irq_t * l4irq_attach_ft (
    int irqnum,
    unsigned mode)
```

Attach/connect to IRQ using given type.

#### Parameters

<i>irqnum</i>	IRQ number to request
<i>mode</i>	Interrupt type,

See also

[L4\\_irq\\_mode](#)

Returns

Pointer to `l4irq_t` structure, 0 on error

This `l4irq_attach` has to be called in the same thread as `l4irq_wait` and caller has to be a pthread thread.

References [L4\\_CV](#).

#### 14.5.2.2.3 l4irq\_attach\_thread()

```
l4irq_t * l4irq_attach_thread (
    int irqnum,
    l4_cap_idx_t to_thread)
```

Attach/connect to IRQ.

Parameters

<i>irqnum</i>	IRQ number to request
<i>to_thread</i>	Attach IRQ to this specified thread.

Returns

Pointer to `l4irq_t` structure, 0 on error

The pointer to the IRQ structure is used as a label in the IRQ object.

References [L4\\_CV](#).

#### 14.5.2.2.4 l4irq\_attach\_thread\_ft()

```
l4irq_t * l4irq_attach_thread_ft (
    int irqnum,
    l4_cap_idx_t to_thread,
    unsigned mode)
```

Attach/connect to IRQ using given type.

Parameters

---

<i>irqnum</i>	IRQ number to request
<i>to_thread</i>	Attach IRQ to this specified thread.
<i>mode</i>	Interrupt type,

See also

[L4\\_irq\\_mode](#)

Returns

Pointer to `l4irq_t` structure, 0 on error

The pointer to the IRQ structure is used as a label in the IRQ object.

References [L4\\_CV](#).

#### 14.5.2.2.5 `l4irq_detach()`

```
long l4irq_detach (  
    l4irq_t * irq)
```

Detach from IRQ.

Parameters

<i>irq</i>	IRQ data structure
------------	--------------------

Returns

0 on success, != 0 on error

References [L4\\_CV](#).

#### 14.5.2.2.6 `l4irq_unmask()`

```
long l4irq_unmask (  
    l4irq_t * irq)
```

Unmask a specific IRQ.

Parameters

<i>irq</i>	IRQ data structure
------------	--------------------

Returns

0 on success, != 0 on error

This function is useful if a thread wants to wait for multiple IRQs using `l4_ipc_wait`.

References [L4\\_CV](#).

#### 14.5.2.2.7 l4irq\_unmask\_and\_wait\_any()

```
long l4irq_unmask_and_wait_any (
    l4irq_t * unmask_irq,
    l4irq_t ** ret_irq)
```

Unmask a specific IRQ and wait for any attached IRQ.

##### Parameters

	<i>unmask_irq</i>	IRQ data structure for unmask.
out	<i>ret_irq</i>	Received interrupt.

##### Returns

0 on success, != 0 on error

References [L4\\_CV](#).

#### 14.5.2.2.8 l4irq\_wait()

```
long l4irq_wait (
    l4irq_t * irq)
```

Wait for specified IRQ.

##### Parameters

<i>irq</i>	IRQ data structure
------------	--------------------

##### Returns

0 on success, != 0 on error

##### Examples

[examples/libs/libirq/loop.c](#).

References [L4\\_CV](#).

#### 14.5.2.2.9 l4irq\_wait\_any()

```
long l4irq_wait_any (
    l4irq_t ** irq)
```

Wait for any attached IRQ.

##### Return values

---

<i>irq</i>	Received interrupt.
------------	---------------------

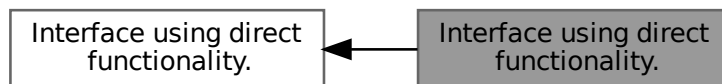
**Returns**

0 on success, != 0 on error

References [L4\\_CV](#).

**14.5.2.3 Interface using direct functionality.**

Collaboration diagram for Interface using direct functionality.:

**Functions**

- `l4irq_t * l4irq_attach_cap (l4_cap_idx_t irqcap)`  
*Attach/connect to IRQ.*
- `l4irq_t * l4irq_attach_cap_ft (l4_cap_idx_t irqcap, unsigned mode)`  
*Attach/connect to IRQ using given type.*
- `l4irq_t * l4irq_attach_thread_cap (l4_cap_idx_t irqcap, l4_cap_idx_t to_thread)`  
*Attach/connect to IRQ.*
- `l4irq_t * l4irq_attach_thread_cap_ft (l4_cap_idx_t irqcap, l4_cap_idx_t to_thread, unsigned mode)`  
*Attach/connect to IRQ using given type.*

**14.5.2.3.1 Detailed Description****14.5.2.3.2 Function Documentation****14.5.2.3.2.1 l4irq\_attach\_cap()**

```
l4irq_t * l4irq_attach_cap (
    l4_cap_idx_t irqcap)
```

Attach/connect to IRQ.

**Parameters**

<i>irqcap</i>	IRQ capability
---------------	----------------

**Returns**

Pointer to `l4irq_t` structure, 0 on error

This `l4irq_attach` has to be called in the same thread as `l4irq_wait` and caller has to be a pthread thread.

References [L4\\_CV](#).

**14.5.2.3.2.2 l4irq\_attach\_cap\_ft()**

```
l4irq_t * l4irq_attach_cap_ft (
    l4_cap_idx_t irqcap,
    unsigned mode)
```

Attach/connect to IRQ using given type.

**Parameters**

<i>irqcap</i>	IRQ capability
<i>mode</i>	Interrupt type,

**See also**

[L4\\_irq\\_mode](#)

**Returns**

Pointer to `l4irq_t` structure, 0 on error

This `l4irq_attach` has to be called in the same thread as `l4irq_wait` and caller has to be a pthread thread.

References [L4\\_CV](#).

**14.5.2.3.2.3 l4irq\_attach\_thread\_cap()**

```
l4irq_t * l4irq_attach_thread_cap (
    l4_cap_idx_t irqcap,
    l4_cap_idx_t to_thread)
```

Attach/connect to IRQ.

**Parameters**

<i>irqcap</i>	IRQ capability
---------------	----------------



<i>to_thread</i>	Attach IRQ to this thread.
------------------	----------------------------

**Returns**

Pointer to `l4irq_t` structure, 0 on error

The pointer to the IRQ structure is used as a label in the IRQ object.

References [L4\\_CV](#).

**14.5.2.3.2.4 l4irq\_attach\_thread\_cap\_ft()**

```
l4irq_t * l4irq_attach_thread_cap_ft (
    l4_cap_idx_t irqcap,
    l4_cap_idx_t to_thread,
    unsigned mode)
```

Attach/connect to IRQ using given type.

**Parameters**

<i>irqcap</i>	IRQ capability
<i>to_thread</i>	Attach IRQ to this thread.
<i>mode</i>	Interrupt type,

**See also**

[L4\\_irq\\_mode](#)

**Returns**

Pointer to `l4irq_t` structure, 0 on error

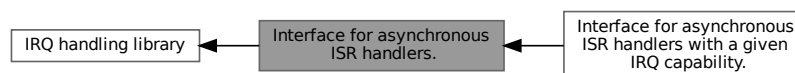
The pointer to the IRQ structure is used as a label in the IRQ object.

References [L4\\_CV](#).

**14.5.3 Interface for asynchronous ISR handlers.**

This interface has just two (main) functions.

Collaboration diagram for Interface for asynchronous ISR handlers.:



## Topics

- Interface for asynchronous ISR handlers with a given IRQ capability. . . . . 483  
*This group is just an enhanced version to [l4irq\\_request\(\)](#) which takes a capability object instead of a plain number.*

## Functions

- `l4irq_t * l4irq_request` (int irqnum, void(\*isr\_handler)(void \*), void \*isr\_data, int irq\_thread\_prio, unsigned mode)  
*Attach asynchronous ISR handler to IRQ.*
- `long l4irq_release` (l4irq\_t \*irq)  
*Release asynchronous ISR handler and free resources.*

### 14.5.3.1 Detailed Description

This interface has just two (main) functions.

`l4irq_request` to install a handler for an interrupt and `l4irq_release` to uninstall the handler again and release all resources associated with it.

### 14.5.3.2 Function Documentation

#### 14.5.3.2.1 l4irq\_release()

```
long l4irq_release (
    l4irq_t * irq)
```

Release asynchronous ISR handler and free resources.

#### Parameters

<i>irq</i>	IRQ data structure
------------	--------------------

#### Returns

0 success, != 0 failure

#### Examples

[examples/libs/libirq/async\\_isr.c](#).

References [L4\\_CV](#).

#### 14.5.3.2.2 l4irq\_request()

```
l4irq_t * l4irq_request (
    int irqnum,
    void(* isr_handler ) (void *),
    void * isr_data,
    int irq_thread_prio,
    unsigned mode)
```

Attach asynchronous ISR handler to IRQ.

#### Parameters

<i>irqnum</i>	IRQ number to request
<i>isr_handler</i>	Handler routine that is called when an interrupt triggers
<i>isr_data</i>	Pointer given as argument to <i>isr_handler</i>
<i>irq_thread_prio</i>	<a href="#">L4</a> thread priority of the ISR handler. Give -1 for same priority as creator.
<i>mode</i>	Interrupt type,

**See also**

[L4\\_irq\\_mode](#)

**Returns**

Pointer to `l4irq_t` structure, 0 on error

**Examples**

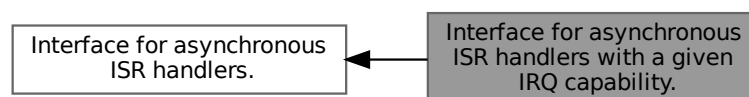
[examples/libs/libirq/async\\_isr.c](#).

References [L4\\_CV](#).

**14.5.3.3 Interface for asynchronous ISR handlers with a given IRQ capability.**

This group is just an enhanced version to [l4irq\\_request\(\)](#) which takes a capability object instead of a plain number.

Collaboration diagram for Interface for asynchronous ISR handlers with a given IRQ capability.:

**Functions**

- `l4irq_t * l4irq\_request\_cap (l4\_cap\_idx\_t irqcap, void(*isr_handler)(void *), void *isr_data, int irq_thread_↔prio, unsigned mode)`

*Attach asynchronous ISR handler to IRQ.*

**14.5.3.3.1 Detailed Description**

This group is just an enhanced version to [l4irq\\_request\(\)](#) which takes a capability object instead of a plain number.

### 14.5.3.3.2 Function Documentation

#### 14.5.3.3.2.1 l4irq\_request\_cap()

```
l4irq_t * l4irq_request_cap (
    l4_cap_idx_t irqcap,
    void(* isr_handler )(void *),
    void * isr_data,
    int irq_thread_prio,
    unsigned mode)
```

Attach asynchronous ISR handler to IRQ.

#### Parameters

<i>irqcap</i>	IRQ capability
<i>isr_handler</i>	Handler routine that is called when an interrupt triggers
<i>isr_data</i>	Pointer given as argument to isr_handler
<i>irq_thread_prio</i>	<a href="#">L4</a> thread priority of the ISR handler. Give -1 for same priority as creator.
<i>mode</i>	Interrupt type,

#### See also

[L4\\_irq\\_mode](#)

#### Returns

Pointer to l4irq\_t structure, 0 on error

## 14.6 L4 IPC Opcodes

List of protocol specific opcodes used for communication with [L4Re](#) and Kernel objects.

#### Enumerations

- enum [L4\\_icu\\_opcode](#) {  
[L4\\_ICU\\_OP\\_BIND](#) , [L4\\_ICU\\_OP\\_UNBIND](#) , [L4\\_ICU\\_OP\\_INFO](#) , [L4\\_ICU\\_OP\\_MSI\\_INFO](#) ,  
[L4\\_ICU\\_OP\\_UNMASK](#) , [L4\\_ICU\\_OP\\_MASK](#) , [L4\\_ICU\\_OP\\_SET\\_MODE](#) }  
*Opcodes to the ICU interface.*
- enum [L4\\_ipc\\_gate\\_ops](#) { [L4\\_IPC\\_GATE\\_BIND\\_OP](#) = 0x10 , [L4\\_IPC\\_GATE\\_GET\\_INFO\\_OP](#) = 0x11 }  
*Operations on the IPC-gate.*
- enum [L4\\_platform\\_ctl\\_ops](#) {  
[L4\\_PLATFORM\\_CTL\\_SYS\\_SUSPEND\\_OP](#) = 0UL , [L4\\_PLATFORM\\_CTL\\_SYS\\_SHUTDOWN\\_OP](#) = 1UL ,  
[L4\\_PLATFORM\\_CTL\\_CPU\\_ALLOW\\_SHUTDOWN\\_OP](#) = 2UL , [L4\\_PLATFORM\\_CTL\\_CPU\\_ENABLE\\_OP](#) =  
3UL ,  
[L4\\_PLATFORM\\_CTL\\_CPU\\_DISABLE\\_OP](#) = 4UL , [L4\\_PLATFORM\\_CTL\\_SET\\_TASK\\_ASID\\_OP](#) = 0x10UL }  
*Operations on platform-control objects.*

- enum `L4_task_ops` {  
`L4_TASK_MAP_OP` = 0UL , `L4_TASK_UNMAP_OP` = 1UL , `L4_TASK_CAP_INFO_OP` = 2UL ,  
`L4_TASK_ADD_KU_MEM_OP` = 3UL ,  
`L4_TASK_LDT_SET_X86_OP` = 0x11UL , `L4_TASK_MAP_VGICC_ARM_OP` = 0x12UL }  
*Operations on task objects.*
- enum `L4_thread_ops` {  
`L4_THREAD_CONTROL_OP` = 0UL , `L4_THREAD_EX_REGS_OP` = 1UL , `L4_THREAD_SWITCH_OP` =  
2UL , `L4_THREAD_STATS_OP` = 3UL ,  
`L4_THREAD_VCPU_RESUME_OP` = 4UL , `L4_THREAD_REGISTER_DELETE_IRQ_OP` = 5UL ,  
`L4_THREAD_MODIFY_SENDER_OP` = 6UL , `L4_THREAD_VCPU_CONTROL_OP` = 7UL ,  
`L4_THREAD_VCPU_CONTROL_EXT_OP` = `L4_THREAD_VCPU_CONTROL_OP` | 0x10000 , `L4_THREAD_REGISTER_DO`  
= 8UL , `L4_THREAD_X86_GDT_OP` = 0x10UL , `L4_THREAD_ARM_TPIDRURO_OP` = 0x10UL ,  
`L4_THREAD_AMD64_SET_SEGMENT_BASE_OP` = 0x12UL , `L4_THREAD_AMD64_GET_SEGMENT_INFO_OP`  
= 0x13UL , `L4_THREAD_OPCODE_MASK` = 0xffff }  
*Operations on thread objects.*
- enum `L4_vcon_ops` { `L4_VCON_WRITE_OP` = 0UL , `L4_VCON_READ_OP` = 1UL , `L4_VCON_SET_ATTR_OP`  
= 2UL , `L4_VCON_GET_ATTR_OP` = 3UL }  
*Operations on vcon objects.*

### 14.6.1 Detailed Description

List of protocol specific opcodes used for communication with [L4Re](#) and Kernel objects.

### 14.6.2 Enumeration Type Documentation

#### 14.6.2.1 L4\_icu\_opcode

enum `L4_icu_opcode`

Opcodes to the ICU interface.

##### Enumerator

<code>L4_ICU_OP_BIND</code>	Bind opcode.  See also <a href="#">l4_icu_bind()</a>
<code>L4_ICU_OP_UNBIND</code>	Unbind opcode.  See also <a href="#">l4_icu_unbind()</a>
<code>L4_ICU_OP_INFO</code>	Info opcode.  See also <a href="#">l4_icu_info()</a>

L4_ICU_OP_MSI_INFO	Msi-info opcode.  See also <a href="#">l4_icu_msi_info()</a>
L4_ICU_OP_UNMASK	Unmask opcode.  See also <a href="#">l4_icu_unmask()</a>
L4_ICU_OP_MASK	Mask opcode.  See also <a href="#">l4_icu_mask()</a>
L4_ICU_OP_SET_MODE	Set-mode opcode.  See also <a href="#">l4_icu_set_mode()</a>

Definition at line 97 of file [icu.h](#).

#### 14.6.2.2 L4\_ipc\_gate\_ops

```
enum L4_ipc_gate_ops
```

Operations on the IPC-gate.

##### Enumerator

L4_IPC_GATE_BIND_OP	Bind operation.
L4_IPC_GATE_GET_INFO_OP	Info operation.

Definition at line 108 of file [ipc\\_gate.h](#).

#### 14.6.2.3 L4\_platform\_ctl\_ops

```
enum L4_platform_ctl_ops
```

Operations on platform-control objects.

See [L4\\_PROTO\\_PLATFORM\\_CTL](#) for the protocol type to use for messages to platform-control objects.

##### Enumerator

L4_PLATFORM_CTL_SYS_SUSPEND_OP	Suspend.
--------------------------------	----------

L4_PLATFORM_CTL_SYS_SHUTDOWN_OP	shutdown/reboot
L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP	allow CPU shutdown
L4_PLATFORM_CTL_CPU_ENABLE_OP	enable an offline CPU
L4_PLATFORM_CTL_CPU_DISABLE_OP	disable an online CPU
L4_PLATFORM_CTL_SET_TASK_ASID_OP	Arm: set task ASID.

Definition at line 159 of file [platform\\_control.h](#).

#### 14.6.2.4 L4\_task\_ops

enum [L4\\_task\\_ops](#)

Operations on task objects.

##### Enumerator

L4_TASK_MAP_OP	Map.
L4_TASK_UNMAP_OP	Unmap.
L4_TASK_CAP_INFO_OP	Cap info.
L4_TASK_ADD_KU_MEM_OP	Add kernel-user memory.
L4_TASK_LDT_SET_X86_OP	x86: LDT set
L4_TASK_MAP_VGICC_ARM_OP	Arm: Map virtual GICC area.

Definition at line 341 of file [task.h](#).

#### 14.6.2.5 L4\_thread\_ops

enum [L4\\_thread\\_ops](#)

Operations on thread objects.

##### Enumerator

L4_THREAD_CONTROL_OP	Control operation.
L4_THREAD_EX_REGS_OP	Exchange registers operation.
L4_THREAD_SWITCH_OP	Do a thread switch.
L4_THREAD_STATS_OP	Thread statistics.
L4_THREAD_VCPU_RESUME_OP	VCPU resume.
L4_THREAD_REGISTER_DELETE_IRQ_OP	Register an IPC-gate deletion IRQ.
L4_THREAD_MODIFY_SENDER_OP	Modify all senders IDs that match the given pattern.
L4_THREAD_VCPU_CONTROL_OP	Enable / disable VCPU feature.
L4_THREAD_REGISTER_DOORBELL_IRQ_OP	Register direct IRQ injection doorbell IRQ.
L4_THREAD_X86_GDT_OP	Gdt.

L4_THREAD_ARM_TPIDRURO_OP	Set TPIDRURO register.
L4_THREAD_AMD64_SET_SEGMENT_BASE_OP	Set segment base.
L4_THREAD_AMD64_GET_SEGMENT_INFO_OP	Get segment information.
L4_THREAD_OPCODE_MASK	Mask for opcodes.

Definition at line 732 of file [thread.h](#).

#### 14.6.2.6 L4\_vcon\_ops

enum [L4\\_vcon\\_ops](#)

Operations on vcon objects.

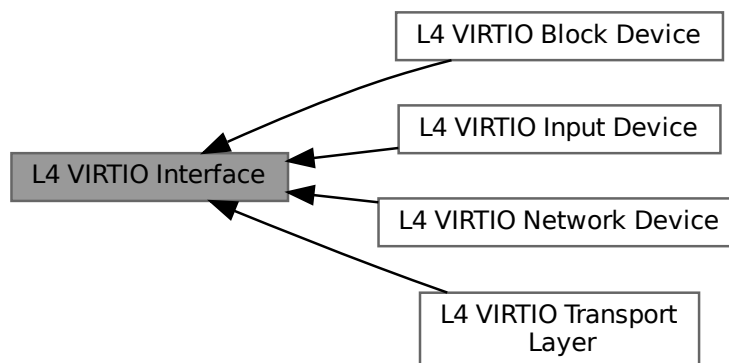
##### Enumerator

L4_VCON_WRITE_OP	Write.
L4_VCON_READ_OP	Read.
L4_VCON_SET_ATTR_OP	Get console attributes.
L4_VCON_GET_ATTR_OP	Set console attributes.

Definition at line 291 of file [vcon.h](#).

## 14.7 L4 VIRTIO Interface

Collaboration diagram for L4 VIRTIO Interface:





## Topics

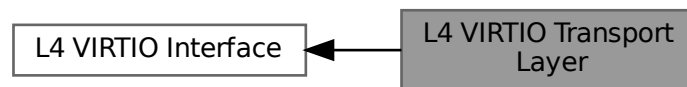
- L4 VIRTIO Transport Layer . . . . . 489  
*L4 specific VIRTIO Transport layer.*
- L4 VIRTIO Block Device . . . . . 499
- L4 VIRTIO Input Device . . . . . 500
- L4 VIRTIO Network Device . . . . . 501

### 14.7.1 Detailed Description

#### 14.7.2 L4 VIRTIO Transport Layer

*L4 specific VIRTIO Transport layer.*

Collaboration diagram for L4 VIRTIO Transport Layer:



## Namespaces

- namespace [L4virtio](#)  
*L4-VIRTIO Transport C++ API.*

## Data Structures

- struct [l4virtio\\_config\\_hdr\\_t](#)  
*L4-VIRTIO config header, provided in shared data space.*
- struct [l4virtio\\_config\\_queue\\_t](#)  
*Queue configuration entry.*

## Typedefs

- typedef struct l4virtio\_config\_hdr\_t [l4virtio\\_config\\_hdr\\_t](#)  
*L4-VIRTIO config header, provided in shared data space.*
- typedef struct l4virtio\_config\_queue\_t [l4virtio\\_config\\_queue\\_t](#)  
*Queue configuration entry.*

## Enumerations

- enum [L4\\_virtio\\_protocol](#)  
*L4-VIRTIO protocol number.*
- enum [L4\\_virtio\\_opcodes](#) {  
L4VIRTIO\_OP\_SET\_STATUS = 0 , L4VIRTIO\_OP\_CONFIG\_QUEUE = 1 , L4VIRTIO\_OP\_REGISTER\_DS = 3 , L4VIRTIO\_OP\_DEVICE\_CONFIG = 4 ,  
L4VIRTIO\_OP\_GET\_DEVICE\_IRQ = 5 }  
*Opcodes to setup and configure a device.*
- enum [L4virtio\\_device\\_ids](#) {  
L4VIRTIO\_ID\_NET = 1 , L4VIRTIO\_ID\_BLOCK = 2 , L4VIRTIO\_ID\_CONSOLE = 3 , L4VIRTIO\_ID\_RNG = 4 ,  
,  
L4VIRTIO\_ID\_BALLOON = 5 , L4VIRTIO\_ID\_RPMMSG = 7 , L4VIRTIO\_ID\_SCSI = 8 , L4VIRTIO\_ID\_9P = 9 ,  
L4VIRTIO\_ID\_RPROC\_SERIAL = 11 , L4VIRTIO\_ID\_CAIF = 12 , L4VIRTIO\_ID\_GPU = 16 , L4VIRTIO\_ID\_INPUT = 18 ,  
L4VIRTIO\_ID\_VSOCK = 19 , L4VIRTIO\_ID\_CRYPTIO = 20 , L4VIRTIO\_ID\_FS = 26 , L4VIRTIO\_ID\_SCMI = 32 ,  
L4VIRTIO\_ID\_I2C = 34 , L4VIRTIO\_ID\_GPIO = 41 , L4VIRTIO\_ID\_SOCKET = 0x9999 }  
*Virtio device IDs as reported in the driver's config space.*
- enum [L4virtio\\_device\\_status](#) {  
L4VIRTIO\_STATUS\_ACKNOWLEDGE = 1 , L4VIRTIO\_STATUS\_DRIVER = 2 , L4VIRTIO\_STATUS\_DRIVER\_OK = 4 , L4VIRTIO\_STATUS\_FEATURES\_OK = 8 ,  
L4VIRTIO\_STATUS\_DEVICE\_NEEDS\_RESET = 0x40 , L4VIRTIO\_STATUS\_FAILED = 0x80 }  
*Virtio device status bits.*
- enum [L4virtio\\_feature\\_bits](#) { L4VIRTIO\_FEATURE\_VERSION\_1 = 32 , L4VIRTIO\_FEATURE\_CMD\_CONFIG = 160 }  
*L4virtio-specific feature bits.*
- enum [L4\\_virtio\\_irq\\_status](#) { L4VIRTIO\_IRQ\_STATUS\_VRING = 1 , L4VIRTIO\_IRQ\_STATUS\_CONFIG = 2 }  
*VIRTIO IRQ status codes (l4virtio\_config\_hdr\_t::irq\_status).*
- enum [L4\\_virtio\\_cmd](#) {  
L4VIRTIO\_CMD\_NONE = 0x00000000 , L4VIRTIO\_CMD\_SET\_STATUS = 0x01000000 , L4VIRTIO\_CMD\_CFG\_QUEUE = 0x02000000 , L4VIRTIO\_CMD\_CFG\_CHANGED = 0x04000000 ,  
L4VIRTIO\_CMD\_NOTIFY\_QUEUE = 0x08000000 , L4VIRTIO\_CMD\_MASK = 0xff000000 }  
*Virtio commands for device configuration.*

## Functions

- [L4\\_BEGIN\\_DECLS](#) [l4virtio\\_config\\_queue\\_t](#) \* [l4virtio\\_config\\_queues](#) ([l4virtio\\_config\\_hdr\\_t](#) const \*cfg)  
*Get the pointer to the first queue config.*
- void \* [l4virtio\\_device\\_config](#) ([l4virtio\\_config\\_hdr\\_t](#) const \*cfg)  
*Get the pointer to the device configuration.*
- void [l4virtio\\_set\\_feature](#) ([l4\\_uint32\\_t](#) \*feature\_map, unsigned feat)  
*Set the given feature bit in a feature map.*
- void [l4virtio\\_clear\\_feature](#) ([l4\\_uint32\\_t](#) \*feature\_map, unsigned feat)  
*Clear the given feature bit in a feature map.*
- unsigned [l4virtio\\_get\\_feature](#) ([l4\\_uint32\\_t](#) \*feature\_map, unsigned feat)  
*Check if the given bit in a feature map is set.*
- int [l4virtio\\_set\\_status](#) ([l4\\_cap\\_idx\\_t](#) cap, unsigned status) [L4\\_NOTHROW](#)
- int [l4virtio\\_config\\_queue](#) ([l4\\_cap\\_idx\\_t](#) cap, unsigned queue) [L4\\_NOTHROW](#)
- int [l4virtio\\_register\\_ds](#) ([l4\\_cap\\_idx\\_t](#) cap, [l4\\_cap\\_idx\\_t](#) ds\_cap, [l4\\_uint64\\_t](#) base, [l4\\_umword\\_t](#) offset, [l4\\_umword\\_t](#) size) [L4\\_NOTHROW](#)
- int [l4virtio\\_device\\_config\\_ds](#) ([l4\\_cap\\_idx\\_t](#) cap, [l4\\_cap\\_idx\\_t](#) config\_ds, [l4\\_addr\\_t](#) \*ds\_offset) [L4\\_NOTHROW](#)
- int [l4virtio\\_device\\_notification\\_irq](#) ([l4\\_cap\\_idx\\_t](#) cap, unsigned index, [l4\\_cap\\_idx\\_t](#) irq) [L4\\_NOTHROW](#)

### 14.7.2.1 Detailed Description

[L4](#) specific VIRTIO Transport layer.

The [L4](#) specific VIRTIO Transport layer is based on [L4Re::Dataspace](#) as shared memory and [L4::lrq](#) for signaling. The VIRTIO configuration space is mostly based on a shared memory implementation too and accompanied by two IPC functions to synchronize the configuration between device and driver.

### 14.7.2.2 Typedef Documentation

#### 14.7.2.2.1 l4virtio\_config\_queue\_t

```
typedef struct l4virtio_config_queue_t l4virtio_config_queue_t
```

Queue configuration entry.

An array of such entries is available at the [l4virtio\\_config\\_hdr\\_t::queues\\_offset](#) in the config data space.

Consistency rules for the queue config are:

- A driver might read `num_max` at any time.
- A driver must write to `num`, `desc_addr`, `avail_addr`, and `used_addr` only when `ready` is zero (0). Values in these fields are validated and used by the device only after successfully setting `ready` to one (1), either by the IPC or by `L4VIRTIO_CMD_CFG_QUEUE`.
- The value of `device_notify_index` is valid only when `ready` is one.
- The driver might write to `device_notify_index` at any time, however the change is guaranteed to take effect after a successful `L4VIRTIO_CMD_CFG_QUEUE` or after a `config_queue` IPC. Note, the change might also have immediate effect, depending on the device implementation.

### 14.7.2.3 Enumeration Type Documentation

#### 14.7.2.3.1 L4\_virtio\_cmd

```
enum L4_virtio_cmd
```

Virtio commands for device configuration.

#### Enumerator

L4VIRTIO_CMD_NONE	No command pending.
L4VIRTIO_CMD_SET_STATUS	Set the status register.
L4VIRTIO_CMD_CFG_QUEUE	Configure a queue.
L4VIRTIO_CMD_CFG_CHANGED	Device config changed.
L4VIRTIO_CMD_NOTIFY_QUEUE	Configure a queue.
L4VIRTIO_CMD_MASK	Mask to get command bits.

Definition at line 119 of file [virtio.h](#).

#### 14.7.2.3.2 L4\_virtio\_irq\_status

enum [L4\\_virtio\\_irq\\_status](#)

VIRTIO IRQ status codes (`l4virtio_config_hdr_t::irq_status`).

##### Note

`l4virtio_config_hdr_t::irq_status` is currently unused.

##### Enumerator

L4VIRTIO_IRQ_STATUS_VRING	VRING IRQ pending flag.
L4VIRTIO_IRQ_STATUS_CONFIG	CONFIG IRQ pending flag.

Definition at line 110 of file [virtio.h](#).

#### 14.7.2.3.3 L4\_virtio\_opcodes

enum [L4\\_virtio\\_opcodes](#)

Opcodes to setup and configure a device.

##### Enumerator

L4VIRTIO_OP_SET_STATUS	Write device status register.
L4VIRTIO_OP_CONFIG_QUEUE	Configure queue.
L4VIRTIO_OP_REGISTER_DS	Register shared memory with device.
L4VIRTIO_OP_DEVICE_CONFIG	Get device config page.
L4VIRTIO_OP_GET_DEVICE_IRQ	Retrieve device notification IRQ.

Definition at line 52 of file [virtio.h](#).

#### 14.7.2.3.4 L4virtio\_device\_ids

enum [L4virtio\\_device\\_ids](#)

Virtio device IDs as reported in the driver's config space.

##### Enumerator

L4VIRTIO_ID_NET	Virtual ethernet card.
L4VIRTIO_ID_BLOCK	General block device.
L4VIRTIO_ID_CONSOLE	Simple device for data IO via ports.
L4VIRTIO_ID_RNG	Entropy source.

L4VIRTIO_ID_BALLOON	Memory ballooning device.
L4VIRTIO_ID_RPMMSG	Device using rpmsg protocol.
L4VIRTIO_ID_SCSI	SCSI host device.
L4VIRTIO_ID_9P	Device using 9P transport protocol.
L4VIRTIO_ID_RPROC_SERIAL	Rproc serial device.
L4VIRTIO_ID_CAIF	Device using CAIF network protocol.
L4VIRTIO_ID_GPU	GPU.
L4VIRTIO_ID_INPUT	Input.
L4VIRTIO_ID_VSOCK	Vsock transport.
L4VIRTIO_ID_CRYPTIO	Crypto.
L4VIRTIO_ID_FS	FS.
L4VIRTIO_ID_SCSI	Scmi device.
L4VIRTIO_ID_I2C	I2C device.
L4VIRTIO_ID_GPIO	Gpio device.
L4VIRTIO_ID_SOCKET	Unofficial socket device.

Definition at line 62 of file [virtio.h](#).

#### 14.7.2.3.5 L4virtio\_device\_status

enum [L4virtio\\_device\\_status](#)

Virtio device status bits.

##### Enumerator

L4VIRTIO_STATUS_ACKNOWLEDGE	Guest OS has found device.
L4VIRTIO_STATUS_DRIVER	Guest OS knows how to drive device.
L4VIRTIO_STATUS_DRIVER_OK	Driver is set up.
L4VIRTIO_STATUS_FEATURES_OK	Driver has acknowledged feature set.
L4VIRTIO_STATUS_DEVICE_NEEDS_RESET	Device detected fatal error.
L4VIRTIO_STATUS_FAILED	Driver detected fatal error.

Definition at line 87 of file [virtio.h](#).

#### 14.7.2.3.6 L4virtio\_feature\_bits

enum [L4virtio\\_feature\\_bits](#)

L4virtio-specific feature bits.

##### Enumerator

L4VIRTIO_FEATURE_VERSION_1	Virtio protocol version 1 supported. Must be 1 for <a href="#">L4virtio</a> .
L4VIRTIO_FEATURE_CMD_CONFIG	Status and queue config are set via cmd field instead of via IPC.

Definition at line 98 of file [virtio.h](#).

## 14.7.2.4 Function Documentation

### 14.7.2.4.1 l4virtio\_config\_queue()

```
int l4virtio_config_queue (
    l4_cap_idx_t cap,
    unsigned queue)
```

#### Parameters

<i>cap</i>	Capability to the VIRTIO host.
------------	--------------------------------

Trigger queue configuration of the given queue.

Usually all queues are configured when the status is written to running. However, in some cases queues shall be disabled or enabled dynamically, in this case this function triggers a reconfiguration from the shared memory register of the queue config.

#### Parameters

<i>queue</i>	Queue index for the queue to be configured.
--------------	---------------------------------------------

#### Return values

0	on success.
-L4_EIO	The queue's status is invalid.
-L4_ERANGE	The queue index exceeds the number of queues.
-L4_EINVAL	Otherwise.

References [L4\\_CV](#), and [L4\\_NOTHROW](#).

### 14.7.2.4.2 l4virtio\_config\_queues()

```
L4_BEGIN_DECLS l4virtio_config_queue_t * l4virtio_config_queues (
    l4virtio_config_hdr_t const * cfg) [inline]
```

Get the pointer to the first queue config.

#### Parameters

<i>cfg</i>	Pointer to the config header.
------------	-------------------------------

**Returns**

pointer to queue config of queue 0.

Definition at line 252 of file [virtio.h](#).

References [l4virtio\\_config\\_hdr\\_t::queues\\_offset](#).

**14.7.2.4.3 l4virtio\_device\_config()**

```
void * l4virtio_device_config (
    l4virtio_config_hdr_t const * cfg) [inline]
```

Get the pointer to the device configuration.

**Parameters**

<i>cfg</i>	Pointer to the config header.
------------	-------------------------------

**Returns**

pointer to device configuration structure.

Definition at line 263 of file [virtio.h](#).

**14.7.2.4.4 l4virtio\_device\_config\_ds()**

```
int l4virtio_device_config_ds (
    l4_cap_idx_t cap,
    l4_cap_idx_t config_ds,
    l4_addr_t * ds_offset)
```

**Parameters**

<i>cap</i>	Capability to the L4-VIRTIO host
------------	----------------------------------

Get the dataspace with the [L4virtio](#) configuration page.

**Parameters**

<i>config_ds</i>	Capability for receiving the dataspace capability for the shared L4-VIRTIO config data space.
<i>ds_offset</i>	Offset into the dataspace where the device configuration structure starts.

References [L4\\_CV](#), and [L4\\_NOTHROW](#).

#### 14.7.2.4.5 l4virtio\_device\_notification\_irq()

```
int l4virtio_device_notification_irq (  
    l4_cap_idx_t cap,  
    unsigned index,  
    l4_cap_idx_t irq)
```

#### Parameters

---



<i>cap</i>	Capability to the L4-VIRTIO host
------------	----------------------------------

Get the notification interrupt corresponding to the given index.

#### Parameters

	<i>index</i>	Index of the interrupt.
out	<i>irq</i>	Triggerable for the given index.

#### Return values

<i>L4_EOK</i>	Success.
<i>L4_ENOSYS</i>	IRQ notification not supported by device.
<i>&lt;0</i>	Other error.

An index is only guaranteed to return an IRQ object when the index is set in one of the device notify index fields. The device must return the same interrupt for a given index as long as the index is in use. If an index disappears as a result of a configuration change and then is reused later, the interrupt is not guaranteed to be the same.

Interrupts must always be rerequested after a device reset.

References [L4\\_END\\_DECLS](#), and [L4\\_NOTHROW](#).

#### 14.7.2.4.6 l4virtio\_register\_ds()

```
int l4virtio_register_ds (
    l4_cap_idx_t cap,
    l4_cap_idx_t ds_cap,
    l4_uint64_t base,
    l4_umword_t offset,
    l4_umword_t size)
```

#### Parameters

<i>cap</i>	Capability to the VIRTIO host
------------	-------------------------------

Register a shared data space with VIRTIO host.

#### Parameters

<i>ds_cap</i>	Dataspace capability to register. The lower 8 bits determine the rights mask with which the guest's rights are masked during the registration of the dataspace at the VIRTIO host.
<i>base</i>	VIRTIO guest physical start address of shared memory region
<i>offset</i>	Offset within the data space that is attached to the given <i>base</i> in the guest physical memory.
<i>size</i>	Size of the memory region in the guest

#### Return values

<i>L4_EOK</i>	Operation successful.
<i>-L4_EINVAL</i>	The <code>ds_cap</code> capability is invalid, does not refer to a valid dataspace, is not a trusted dataspace if trusted dataspace validation is enabled, or <code>size</code> and <code>offset</code> specify an invalid region.
<i>-L4_ENOMEM</i>	The limit of dataspaces that can be registered has been reached or no capability slot could be allocated.
<i>-L4_ERANGE</i>	<code>offset</code> is larger than the size of the dataspace.
<i>&lt;0</i>	Any error returned by the dataspace when queried for information during setup or any error returned by the region manager from attaching the dataspace.

References [L4\\_CV](#), and [L4\\_NOTHROW](#).

#### 14.7.2.4.7 l4virtio\_set\_status()

```
int l4virtio_set_status (
    l4_cap_idx_t cap,
    unsigned status)
```

##### Parameters

<i>cap</i>	Capability to the VIRTIO host
------------	-------------------------------

Write the VIRTIO status register.

##### Parameters

<i>status</i>	Status word to write to the VIRTIO status.
---------------	--------------------------------------------

##### Return values

<i>0</i>	on success.
----------	-------------

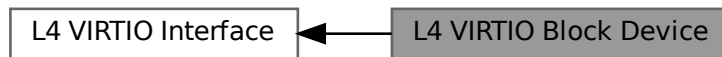
##### Note

All other registers are accessed via shared memory.

References [L4\\_CV](#), and [L4\\_NOTHROW](#).

### 14.7.3 L4 VIRTIO Block Device

Collaboration diagram for L4 VIRTIO Block Device:



#### Data Structures

- struct `l4virtio_block_header_t`  
*Header structure of a request for a block device.*
- struct `l4virtio_block_discard_t`  
*Structure used for the write zeroes and discard commands.*
- struct `l4virtio_block_config_t`  
*Device configuration for block devices.*

#### Typedefs

- typedef struct `l4virtio_block_header_t` **`l4virtio_block_header_t`**  
*Header structure of a request for a block device.*
- typedef struct `l4virtio_block_discard_t` **`l4virtio_block_discard_t`**  
*Structure used for the write zeroes and discard commands.*
- typedef struct `l4virtio_block_config_t` **`l4virtio_block_config_t`**  
*Device configuration for block devices.*

#### Enumerations

- enum `L4virtio_block_operations` {  
`L4VIRTIO_BLOCK_T_IN` = 0 , `L4VIRTIO_BLOCK_T_OUT` = 1 , `L4VIRTIO_BLOCK_T_FLUSH` = 4 ,  
`L4VIRTIO_BLOCK_T_GET_ID` = 8 ,  
`L4VIRTIO_BLOCK_T_DISCARD` = 11 , `L4VIRTIO_BLOCK_T_WRITE_ZEROES` = 13 }  
*Kinds of operation over a block device.*
- enum `L4virtio_block_status` { `L4VIRTIO_BLOCK_S_OK` = 0 , `L4VIRTIO_BLOCK_S_IOERR` = 1 ,  
`L4VIRTIO_BLOCK_S_UNSUPP` = 2 }  
*Status of a finished block request.*

#### 14.7.3.1 Detailed Description

#### 14.7.3.2 Enumeration Type Documentation

##### 14.7.3.2.1 L4virtio\_block\_operations

enum `L4virtio_block_operations`

Kinds of operation over a block device.

#### Enumerator

L4VIRTIO_BLOCK_T_IN	Read from device.
L4VIRTIO_BLOCK_T_OUT	Write to device.
L4VIRTIO_BLOCK_T_FLUSH	Flush data to disk.
L4VIRTIO_BLOCK_T_GET_ID	Get device ID.
L4VIRTIO_BLOCK_T_DISCARD	Discard a range of sectors.
L4VIRTIO_BLOCK_T_WRITE_ZEROES	Write zeroes to a range of sectors.

Definition at line 19 of file [virtio\\_block.h](#).

#### 14.7.3.2.2 L4virtio\_block\_status

enum [L4virtio\\_block\\_status](#)

Status of a finished block request.

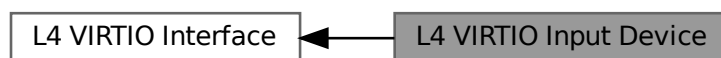
##### Enumerator

L4VIRTIO_BLOCK_S_OK	Request finished successfully.
L4VIRTIO_BLOCK_S_IOERR	IO error on device.
L4VIRTIO_BLOCK_S_UNSUPP	Operation is not supported.

Definition at line 32 of file [virtio\\_block.h](#).

### 14.7.4 L4 VIRTIO Input Device

Collaboration diagram for L4 VIRTIO Input Device:



##### Data Structures

- struct [l4virtio\\_input\\_absinfo\\_t](#)  
*Information about the absolute axis in the underlying evdev implementation.*
- struct [l4virtio\\_input\\_devids\\_t](#)  
*Device ID information for the device.*
- struct [l4virtio\\_input\\_config\\_t](#)  
*Device configuration for input devices.*
- struct [l4virtio\\_input\\_event\\_t](#)  
*Single event in event or status queue.*

### Typedefs

- typedef struct [l4virtio\\_input\\_absinfo\\_t](#) **l4virtio\_absinfo\_t**  
*Information about the absolute axis in the underlying evdev implementation.*
- typedef struct [l4virtio\\_input\\_devids\\_t](#) **l4virtio\_input\_devids\_t**  
*Device ID information for the device.*
- typedef struct [l4virtio\\_input\\_config\\_t](#) **l4virtio\_input\_config\_t**  
*Device configuration for input devices.*
- typedef struct [l4virtio\\_input\\_event\\_t](#) **l4virtio\_input\_event\_t**  
*Single event in event or status queue.*

### Enumerations

- enum [L4virtio\\_input\\_config\\_select](#)  
*Device information selectors.*

#### 14.7.4.1 Detailed Description

### 14.7.5 L4 VIRTIO Network Device

Collaboration diagram for L4 VIRTIO Network Device:



### Data Structures

- struct [l4virtio\\_net\\_header\\_t](#)  
*Header structure of a request for a network device.*
- struct [l4virtio\\_net\\_config\\_t](#)  
*Device configuration for network devices.*

### Typedefs

- typedef struct [l4virtio\\_net\\_header\\_t](#) **l4virtio\_net\_header\_t**  
*Header structure of a request for a network device.*
- typedef struct [l4virtio\\_net\\_config\\_t](#) **l4virtio\_net\_config\_t**  
*Device configuration for network devices.*

### Enumerations

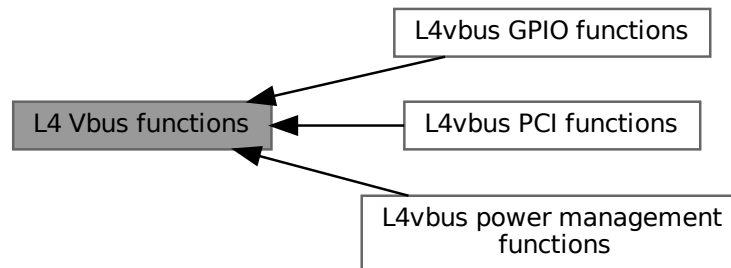
- enum [L4virtio\\_net\\_feature\\_bits](#)  
*Network device feature bits.*

### 14.7.5.1 Detailed Description

## 14.8 L4 Vbus functions

C interface of the Vbus API.

Collaboration diagram for L4 Vbus functions:



### Topics

- [L4vbus GPIO functions](#) . . . . . 513
- [L4vbus PCI functions](#) . . . . . 523
- [L4vbus power management functions](#) . . . . . 529

### Enumerations

- enum [L4vbus\\_dma\\_domain\\_assign\\_flags](#) { [L4VBUS\\_DMAD\\_UNBIND](#) = 0 , [L4VBUS\\_DMAD\\_BIND](#) = 1 , [L4VBUS\\_DMAD\\_L4RE\\_DMA\\_SPACE](#) = 0 , [L4VBUS\\_DMAD\\_KERNEL\\_DMA\\_SPACE](#) = 2 }
- Flags for [l4vbus\\_assign\\_dma\\_domain\(\)](#).*

### Functions

- [L4\\_BEGIN\\_DECLS](#) int [l4vbus\\_get\\_device\\_by\\_hid](#) ([l4\\_cap\\_idx\\_t](#) vbus, [l4vbus\\_device\\_handle\\_t](#) parent, [l4vbus\\_device\\_handle\\_t](#) \*child, char const \*hid, int depth, [l4vbus\\_device\\_t](#) \*devinfo)  
*Find a device by the hardware interface identifier (HID).*
- int [l4vbus\\_get\\_next\\_device](#) ([l4\\_cap\\_idx\\_t](#) vbus, [l4vbus\\_device\\_handle\\_t](#) parent, [l4vbus\\_device\\_handle\\_t](#) \*child, int depth, [l4vbus\\_device\\_t](#) \*devinfo)  
*Find next child following *child*.*
- int [l4vbus\\_get\\_device](#) ([l4\\_cap\\_idx\\_t](#) vbus, [l4vbus\\_device\\_handle\\_t](#) dev, [l4vbus\\_device\\_t](#) \*devinfo)  
*Obtain detailed information about a Vbus device.*
- int [l4vbus\\_get\\_resource](#) ([l4\\_cap\\_idx\\_t](#) vbus, [l4vbus\\_device\\_handle\\_t](#) dev, unsigned res\_idx, [l4vbus\\_resource\\_t](#) \*res)

*Obtain the resource description of an individual device resource.*

- int [l4vbus\\_is\\_compatible](#) ([l4\\_cap\\_idx\\_t](#) vbus, [l4vbus\\_device\\_handle\\_t](#) dev, char const \*cid)

*Check if the given device has a compatibility ID (CID) or HID that matches cid.*

- int [l4vbus\\_get\\_hid](#) ([l4\\_cap\\_idx\\_t](#) vbus, [l4vbus\\_device\\_handle\\_t](#) dev, char \*hid, unsigned long max\_len)

*Get the HID (hardware identifier) of a device.*

- int [l4vbus\\_get\\_adr](#) ([l4\\_cap\\_idx\\_t](#) vbus, [l4vbus\\_device\\_handle\\_t](#) dev, [l4\\_uint32\\_t](#) \*adr)

*Get the bus-specific address of a device.*

- int [l4vbus\\_request\\_ioport](#) ([l4\\_cap\\_idx\\_t](#) vbus, [l4vbus\\_resource\\_t](#) const \*res)

*Request an IO port resource.*

- int [l4vbus\\_assign\\_dma\\_domain](#) ([l4\\_cap\\_idx\\_t](#) vbus, unsigned domain\_id, unsigned flags, [l4\\_cap\\_idx\\_t](#) dma\_space)

*Bind or unbind a kernel [DMA space](#) or a [L4Re::Dma\\_space](#) to a DMA domain.*

- int [l4vbus\\_release\\_ioport](#) ([l4\\_cap\\_idx\\_t](#) vbus, [l4vbus\\_resource\\_t](#) const \*res)

*Release a previously requested IO port resource.*

- int [l4vbus\\_vicu\\_get\\_cap](#) ([l4\\_cap\\_idx\\_t](#) vbus, [l4vbus\\_device\\_handle\\_t](#) icu, [l4\\_cap\\_idx\\_t](#) cap)

*Get capability of ICU.*

## 14.8.1 Detailed Description

C interface of the Vbus API.

The virtual bus (Vbus) is a hierarchical (tree) structure of device nodes where each device has a set of resources attached to it. Each virtual bus provides an Icu ([Interrupt controller](#)) for interrupt handling.

The Vbus interface allows a client to find and query devices present on his virtual bus. After obtaining a device handle for a specific device the client can enumerate its resources.

### Include File

```
#include <l4/vbus/vbus.h>
```

Refer to [L4vbus](#) for the C++ API.

## 14.8.2 Enumeration Type Documentation

### 14.8.2.1 L4vbus\_dma\_domain\_assign\_flags

```
enum L4vbus\_dma\_domain\_assign\_flags
```

Flags for [l4vbus\\_assign\\_dma\\_domain\(\)](#).

### Enumerator

L4VBUS_DMAD_UNBIND	Unbind the given DMA space from the DMA domain.
L4VBUS_DMAD_BIND	Bind the given DMA space to the DMA domain.
L4VBUS_DMAD_L4RE_DMA_SPACE	The given DMA space is an <a href="#">L4Re::Dma_space</a> .
L4VBUS_DMAD_KERNEL_DMA_SPACE	The given DMA space is a kernel DMA space ( <a href="#">L4::Task</a> ).

Definition at line 174 of file [vbus.h](#).

### 14.8.3 Function Documentation

#### 14.8.3.1 l4vbus\_assign\_dma\_domain()

```
int l4vbus_assign_dma_domain (
    l4_cap_idx_t vbus,
    unsigned domain_id,
    unsigned flags,
    l4_cap_idx_t dma_space)
```

Bind or unbind a kernel [DMA space](#) or a [L4Re::Dma\\_space](#) to a DMA domain.

#### Parameters

<i>vbus</i>	Capability of the system bus
<i>domain_id</i>	DMA domain ID (resource address of DMA domain found on the vBUS). If the value is ~0U the DMA space of the whole vBUS is used.
<i>flags</i>	A combination of <a href="#">L4vbus_dma_domain_assign_flags</a> .
<i>dma_space</i>	The DMA space capability to bind or unbind, this must either be an <a href="#">L4Re::Dma_space</a> or a kernel <a href="#">DMA space</a> ( <a href="#">L4::Task</a> created with L4_PROTO_DMA_SPACE) and the type must be reflected in the <i>flags</i> .

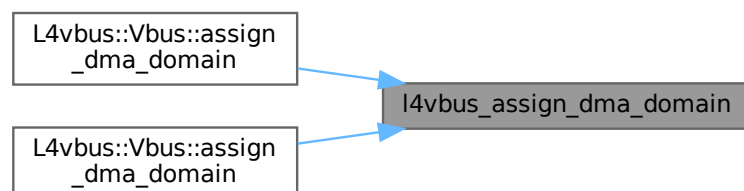
#### Return values

<i>0</i>	Operation completed successfully.
<i>-L4_ENOENT</i>	The vbus does not support a global DMA domain or no DMA domain could be found.
<i>-L4_EINVAL</i>	Invalid argument used.
<i>-L4_EBUSY</i>	DMA domain is already active, this means another DMA space is already assigned.

References [L4\\_CV](#).

Referenced by [L4vbus::Vbus::assign\\_dma\\_domain\(\)](#), and [L4vbus::Vbus::assign\\_dma\\_domain\(\)](#).

Here is the caller graph for this function:





**14.8.3.2 l4vbus\_get\_adr()**

```
int l4vbus_get_adr (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t dev,
    l4_uint32_t * adr)
```

Get the bus-specific address of a device.

**Parameters**

	<i>vbus</i>	Capability of the system bus
	<i>dev</i>	Handle of the device
out	<i>adr</i>	Address

**Return values**

<i>L4_EOK</i>	Success.
<i>-L4_ENOSYS</i>	Device has no valid address.

References [L4\\_CV](#).

**14.8.3.3 l4vbus\_get\_device()**

```
int l4vbus_get_device (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t dev,
    l4vbus_device_t * devinfo)
```

Obtain detailed information about a Vbus device.

**Parameters**

	<i>vbus</i>	Capability of the vbus to which the device is connected.
	<i>dev</i>	Device handle of the device from which to retrieve the details.
out	<i>devinfo</i>	Information structure which contains details about the device. The pointer might be NULL.

**Return values**

<i>0</i>	Success.
<i>-L4_ENODEV</i>	No device with the given device handle <i>dev</i> could be found.

References [L4\\_CV](#).

Referenced by [L4vbus::Device::device\(\)](#).

Here is the caller graph for this function:



#### 14.8.3.4 l4vbus\_get\_device\_by\_hid()

```

L4_BEGIN_DECLS int l4vbus_get_device_by_hid (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t parent,
    l4vbus_device_handle_t * child,
    char const * hid,
    int depth,
    l4vbus_device_t * devinfo)
  
```

Find a device by the hardware interface identifier (HID).

##### Parameters

<i>vbus</i>	Capability of the system bus
<i>parent</i>	Handle to the parent to start the search

This function searches the vbus for a device with the given HID and returns a handle to the first matching device. The HID usually conforms to an ACPI HID or a Linux device tree compatible identifier.

It is possible to have multiple devices with the same HID on a vbus. In order to find all matching devices this function has to be called repeatedly with *child* pointing to the device found in the previous iteration. The iteration starts at *child* that might be any device node in the tree.

##### Parameters

in, out	<i>child</i>	Handle of the device from where in the device tree the search should start. To start searching from the beginning <i>child</i> must be initialized using the default ( <a href="#">L4VBUS_NULL</a> ). If a matching device is found, its handle is returned through this parameter.
	<i>hid</i>	HID of the device
	<i>depth</i>	Maximum depth for the recursive lookup
out	<i>devinfo</i>	Device information structure (might be NULL)

##### Return values

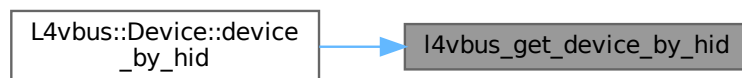
<i>&gt;=0</i>	A device with the given HID was found.
---------------	----------------------------------------

-L4_ENOENT	No device with the given HID could be found on the vbus.
-L4_EINVAL	Invalid or no HID provided.
-L4_ENODEV	Function called on a non-existing device.

References [L4\\_CV](#).

Referenced by [L4vbus::Device::device\\_by\\_hid\(\)](#).

Here is the caller graph for this function:



### 14.8.3.5 l4vbus\_get\_hid()

```
int l4vbus_get_hid (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t dev,
    char * hid,
    unsigned long max_len)
```

Get the HID (hardware identifier) of a device.

#### Parameters

<i>vbus</i>	Capability of the system bus
<i>dev</i>	Handle of the device
<i>hid</i>	Pointer to a buffer for the HID string
<i>max_len</i>	The size of the buffer ( <i>hid</i> )

#### Returns

the length of the HID string on success, else failure

References [L4\\_CV](#).

#### 14.8.3.6 l4vbus\_get\_next\_device()

```
int l4vbus_get_next_device (  
    l4_cap_idx_t vbus,  
    l4vbus_device_handle_t parent,  
    l4vbus_device_handle_t * child,  
    int depth,  
    l4vbus_device_t * devinfo)
```

Find next child following `child`.

#### Parameters

---

	<i>vbus</i>	Capability of the system bus
	<i>parent</i>	Handle to the parent device (use <a href="#">L4VBUS_ROOT_BUS</a> for the system bus)
<i>in, out</i>	<i>child</i>	Handle of the device that precedes the device that shall be returned. To start from the beginning, <i>child</i> must be initialized with <a href="#">L4VBUS_NULL</a> . If a device is found, its handle is returned through this parameter.
	<i>depth</i>	Depth to look for
<i>out</i>	<i>devinfo</i>	Device information (might be NULL)

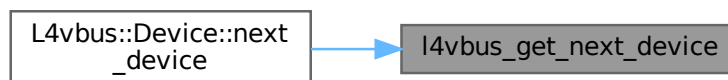
**Returns**

0 on success, else failure

References [L4\\_CV](#).

Referenced by [L4vbus::Device::next\\_device\(\)](#).

Here is the caller graph for this function:

**14.8.3.7 l4vbus\_get\_resource()**

```

int l4vbus_get_resource (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t dev,
    unsigned res_idx,
    l4vbus_resource_t * res)
  
```

Obtain the resource description of an individual device resource.

**Parameters**

	<i>vbus</i>	Capability of the vbus to which the device is connected.
	<i>dev</i>	Device handle of the device on the vbus. The device handle can be obtained by using the <a href="#">l4vbus_get_device_by_hid()</a> and <a href="#">l4vbus_get_next_device()</a> functions.
	<i>res_idx</i>	Index of the resource for which the resource description should be returned. The total number of resources for a device is available in the <a href="#">l4vbus_device_t</a> structure that is returned by <a href="#">L4vbus::Device::device_by_hid()</a> and <a href="#">L4vbus::Device::next_device()</a> .
<i>out</i>	<i>res</i>	Descriptor of the resource.

This function returns the resource descriptor of an individual device resource selected by the *res\_idx* parameter.

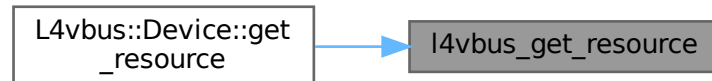
**Return values**

0	Success.
-L4_ENOENT	Invalid resource index <code>res_idx</code> .

References [L4\\_CV](#).

Referenced by [L4vbus::Device::get\\_resource\(\)](#).

Here is the caller graph for this function:



#### 14.8.3.8 l4vbus\_is\_compatible()

```

int l4vbus_is_compatible (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t dev,
    char const * cid)
  
```

Check if the given device has a compatibility ID (CID) or HID that matches *cid*.

##### Parameters

<i>vbus</i>	Capability of the system bus
<i>dev</i>	device handle for which the CID shall be tested
<i>cid</i>	the compatibility ID to test

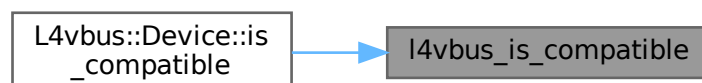
##### Returns

1 when the given ID (*cid*) matches this device, 0 when the given ID does not match, <0 on error.

References [L4\\_CV](#).

Referenced by [L4vbus::Device::is\\_compatible\(\)](#).

Here is the caller graph for this function:



### 14.8.3.9 l4vbus\_release\_ioport()

```
int l4vbus_release_ioport (
    l4_cap_idx_t vbus,
    l4vbus_resource_t const * res)
```

Release a previously requested IO port resource.

#### Parameters

	<i>vbus</i>	Capability of the system bus.
in	<i>res</i>	The IO port resource to be released from the bus.

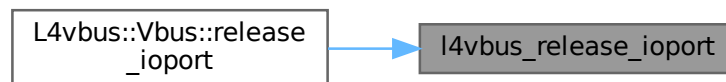
#### Returns

>=0 on success, <0 on error.

References [L4\\_CV](#).

Referenced by [L4vbus::Vbus::release\\_ioport\(\)](#).

Here is the caller graph for this function:



### 14.8.3.10 l4vbus\_request\_ioport()

```
int l4vbus_request_ioport (
    l4_cap_idx_t vbus,
    l4vbus_resource_t const * res)
```

Request an IO port resource.

#### Parameters

	<i>vbus</i>	Capability of the system bus.
in	<i>res</i>	The IO port resource to be requested from the bus.

#### Return values

---

0	Success.
-L4_EINVAL	Resource is not an IO port resource.
-L4_ENOENT	No matching IO port resource found.

If any IO port resource is found that contains the requested IO port range the IO ports are obtained.

Referenced by [L4vbus::Vbus::request\\_ioport\(\)](#).

Here is the caller graph for this function:



#### 14.8.3.11 l4vbus\_vicu\_get\_cap()

```
int l4vbus_vicu_get_cap (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t icu,
    l4_cap_idx_t cap)
```

Get capability of ICU.

##### Parameters

<i>vbus</i>	Capability of the system bus.
<i>icu</i>	ICU device handle.
<i>cap</i>	Capability slot for the capability.

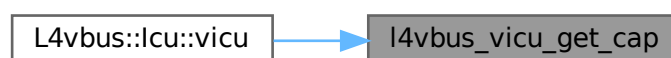
##### Returns

0 on success, else failure

References [L4\\_END\\_DECLS](#).

Referenced by [L4vbus::Icu::vicu\(\)](#).

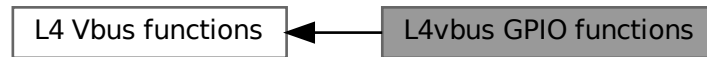
Here is the caller graph for this function:





### 14.8.4 L4vbus GPIO functions

Collaboration diagram for L4vbus GPIO functions:



#### Enumerations

- enum `L4vbus_gpio_generic_func` { `L4VBUS_GPIO_SETUP_INPUT` = 0x100 , `L4VBUS_GPIO_SETUP_OUTPUT` = 0x200 , `L4VBUS_GPIO_SETUP_IRQ` = 0x300 }
- Constants for generic GPIO functions.*
- enum `L4vbus_gpio_pull_modes` { `L4VBUS_GPIO_PIN_PULL_NONE` = 0x100 , `L4VBUS_GPIO_PIN_PULL_UP` = 0x200 , `L4VBUS_GPIO_PIN_PULL_DOWN` = 0x300 }
- Constants for generic GPIO pull up/down resistor configuration.*

#### Functions

- int `l4vbus_gpio_setup` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned pin, unsigned mode, int value)  
*Configure the function of a GPIO pin.*
- int `l4vbus_gpio_config_pull` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned pin, unsigned mode)  
*Generic function to set pull up/down mode.*
- int `l4vbus_gpio_config_pad` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned pin, unsigned func, unsigned value)  
*Hardware specific configuration function.*
- int `l4vbus_gpio_config_get` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned pin, unsigned func, unsigned \*value)  
*Read hardware specific configuration.*
- int `l4vbus_gpio_get` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned pin)  
*Read value of GPIO input pin.*
- int `l4vbus_gpio_set` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned pin, int value)  
*Set GPIO output pin.*
- int `l4vbus_gpio_multi_setup` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned offset, unsigned mask, unsigned mode, unsigned value)  
*Configure function of multiple GPIO pins at once.*
- int `l4vbus_gpio_multi_config_pad` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned offset, unsigned mask, unsigned func, unsigned value)  
*Hardware specific configuration function for multiple GPIO pins.*
- int `l4vbus_gpio_multi_get` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned offset, unsigned \*data)  
*Read values of multiple GPIO pins at once.*
- int `l4vbus_gpio_multi_set` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned offset, unsigned mask, unsigned data)  
*Set multiple GPIO output pins at once.*
- int `l4vbus_gpio_to_irq` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned pin)  
*Create IRQ for GPIO pin.*

#### 14.8.4.1 Detailed Description

#### 14.8.4.2 Enumeration Type Documentation

##### 14.8.4.2.1 L4vbus\_gpio\_generic\_func

enum [L4vbus\\_gpio\\_generic\\_func](#)

Constants for generic GPIO functions.

##### Enumerator

L4VBUS_GPIO_SETUP_INPUT	Set GPIO pin to input.
L4VBUS_GPIO_SETUP_OUTPUT	Set GPIO pin to output.
L4VBUS_GPIO_SETUP_IRQ	Set GPIO pin to IRQ.

Definition at line 24 of file [vbus\\_gpio.h](#).

##### 14.8.4.2.2 L4vbus\_gpio\_pull\_modes

enum [L4vbus\\_gpio\\_pull\\_modes](#)

Constants for generic GPIO pull up/down resistor configuration.

##### Enumerator

L4VBUS_GPIO_PIN_PULL_NONE	No pull up or pull down resistors.
L4VBUS_GPIO_PIN_PULL_UP	enable pull up resistor
L4VBUS_GPIO_PIN_PULL_DOWN	enable pull down resistor

Definition at line 34 of file [vbus\\_gpio.h](#).

#### 14.8.4.3 Function Documentation

##### 14.8.4.3.1 l4vbus\_gpio\_config\_get()

```
int l4vbus_gpio_config_get (  
    l4\_cap\_idx\_t vbus,  
    l4vbus\_device\_handle\_t handle,  
    unsigned pin,  
    unsigned func,  
    unsigned * value)
```

Read hardware specific configuration.

##### Parameters

	<i>vbus</i>	V-BUS capability
	<i>handle</i>	Device handle for the GPIO chip
	<i>pin</i>	GPIO pin number
	<i>func</i>	Hardware specific configuration register to read from. Usually this is an offset to the GPIO chip's base address.
out	<i>value</i>	The configuration value.

**Returns**

0 if OK, error code otherwise

References [L4\\_CV](#).

Referenced by [L4vbus::Gpio\\_pin::config\\_get\(\)](#).

Here is the caller graph for this function:

**14.8.4.3.2 l4vbus\_gpio\_config\_pad()**

```

int l4vbus_gpio_config_pad (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned pin,
    unsigned func,
    unsigned value)
  
```

Hardware specific configuration function.

**Parameters**

<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>pin</i>	GPIO pin number
<i>func</i>	Hardware specific configuration register, usually offset to the GPIO chip's base address
<i>value</i>	Value which is written into the hardware specific configuration register for the specified pin

**Returns**

0 if OK, error code otherwise

References [L4\\_CV](#).

Referenced by [L4vbus::Gpio\\_pin::config\\_pad\(\)](#).

Here is the caller graph for this function:

**14.8.4.3.3 l4vbus\_gpio\_config\_pull()**

```

int l4vbus_gpio_config_pull (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned pin,
    unsigned mode)
  
```

Generic function to set pull up/down mode.

**Parameters**

<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>pin</i>	GPIO pin number
<i>mode</i>	mode for pull up/down resistors, see <a href="#">L4vbus_gpio_pull_modes</a>

**Returns**

0 if OK, error code otherwise

References [L4\\_CV](#).

Referenced by [L4vbus::Gpio\\_pin::config\\_pull\(\)](#).

Here is the caller graph for this function:



#### 14.8.4.3.4 l4vbus\_gpio\_get()

```
int l4vbus_gpio_get (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned pin)
```

Read value of GPIO input pin.

##### Parameters

<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>pin</i>	GPIO pin number to read from

##### Returns

Value of GPIO pin (usually 0 or 1), negative error code otherwise.

References [L4\\_CV](#).

Referenced by [L4vbus::Gpio\\_pin::get\(\)](#).

Here is the caller graph for this function:



#### 14.8.4.3.5 l4vbus\_gpio\_multi\_config\_pad()

```
int l4vbus_gpio_multi_config_pad (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned offset,
    unsigned mask,
    unsigned func,
    unsigned value)
```

Hardware specific configuration function for multiple GPIO pins.

##### Parameters

---

<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>offset</i>	Pin corresponding to the LSB in <i>mask</i> . Note: allowed may be hardware specific.
<i>mask</i>	Mask of GPIO pins to configure. A bit set to 1 configures this pin. A maximum of 32 pins can be configured at once. The real number depends on the hardware and the driver implementation.
<i>func</i>	Hardware specific configuration register, usually offset to the GPIO chip's base address.
<i>value</i>	Value which is written into the hardware specific configuration register for the specified pins

#### Returns

0 if OK, error code otherwise

References [L4\\_CV](#).

Referenced by [L4vbus::Gpio\\_module::config\\_pad\(\)](#).

Here is the caller graph for this function:



#### 14.8.4.3.6 l4vbus\_gpio\_multi\_get()

```
int l4vbus_gpio_multi_get (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned offset,
    unsigned * data)
```

Read values of multiple GPIO pins at once.

#### Parameters

	<i>vbus</i>	V-BUS capability
	<i>handle</i>	Device handle for the GPIO chip
	<i>offset</i>	Pin corresponding to the LSB in <i>data</i> . Note: allowed may be hardware specific.
out	<i>data</i>	Each bit returns the value (0 or 1) for the corresponding GPIO pin. The value of pins that are not accessible is undefined.

**Returns**

0 if OK, error code otherwise

References [L4\\_CV](#).

Referenced by [L4vbus::Gpio\\_module::get\(\)](#).

Here is the caller graph for this function:

**14.8.4.3.7 l4vbus\_gpio\_multi\_set()**

```
int l4vbus_gpio_multi_set (  
    l4_cap_idx_t vbus,  
    l4vbus_device_handle_t handle,  
    unsigned offset,  
    unsigned mask,  
    unsigned data)
```

Set multiple GPIO output pins at once.

**Parameters**

<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>offset</i>	Pin corresponding to the LSB in <i>data</i> . Note: allowed may be hardware specific.
<i>mask</i>	Mask of GPIO pins to set. A bit set to 1 selects this pin. A maximum of 32 pins can be set at once. The real number depends on the hardware and the driver implementation.
<i>data</i>	Each bit corresponds to the GPIO pin in <i>mask</i> . The value of each bit is written to the GPIO pin if its bit in <i>mask</i> is set.

**Returns**

0 if OK, error code otherwise

References [L4\\_CV](#).

Referenced by [L4vbus::Gpio\\_module::set\(\)](#).

Here is the caller graph for this function:



#### 14.8.4.3.8 l4vbus\_gpio\_multi\_setup()

```

int l4vbus_gpio_multi_setup (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned offset,
    unsigned mask,
    unsigned mode,
    unsigned value)
  
```

Configure function of multiple GPIO pins at once.

##### Parameters

<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>offset</i>	Pin corresponding to the LSB in <i>mask</i> . Note: allowed may be hardware specific.
<i>mask</i>	Mask of GPIO pins to configure. A bit set to 1 configures this pin. A maximum of 32 pins can be configured at once. The real number depends on the hardware and the driver implementation.
<i>mode</i>	GPIO function, see <a href="#">L4vbus_gpio_generic_func</a> for generic functions. Hardware specific functions must be provided in the lower 8 bits.
<i>value</i>	Optional value to set the GPIO pins to if they are configured as output pins

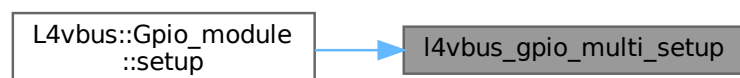
##### Returns

0 if OK, error code otherwise

References [L4\\_CV](#).

Referenced by [L4vbus::Gpio\\_module::setup\(\)](#).

Here is the caller graph for this function:





#### 14.8.4.3.9 l4vbus\_gpio\_set()

```
int l4vbus_gpio_set (  
    l4_cap_idx_t vbus,  
    l4vbus_device_handle_t handle,  
    unsigned pin,  
    int value)
```

Set GPIO output pin.

##### Parameters

<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>pin</i>	GPIO pin number to write to
<i>value</i>	Value to write to the GPIO pin (usually 0 or 1)

##### Returns

0 if OK, error code otherwise

References [L4\\_CV](#).

Referenced by [L4vbus::Gpio\\_pin::set\(\)](#).

Here is the caller graph for this function:



#### 14.8.4.3.10 l4vbus\_gpio\_setup()

```
int l4vbus_gpio_setup (  
    l4_cap_idx_t vbus,  
    l4vbus_device_handle_t handle,  
    unsigned pin,  
    unsigned mode,  
    int value)
```

Configure the function of a GPIO pin.

##### Parameters

<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>pin</i>	GPIO pin number
<i>mode</i>	GPIO function, see <a href="#">L4vbus_gpio_generic_func</a> for generic functions. Hardware specific functions must be provided in the lower 8 bits.
<i>value</i>	Optional value to set the GPIO pin to if it is configured as an output pin

#### Returns

0 if OK, error code otherwise

References [L4\\_CV](#).

Referenced by [L4vbus::Gpio\\_pin::setup\(\)](#).

Here is the caller graph for this function:



#### 14.8.4.3.11 l4vbus\_gpio\_to\_irq()

```
int l4vbus_gpio_to_irq (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned pin)
```

Create IRQ for GPIO pin.

#### Parameters

<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>pin</i>	GPIO pin to create an IRQ for.

**Returns**

IRQ number if OK, negative error code otherwise

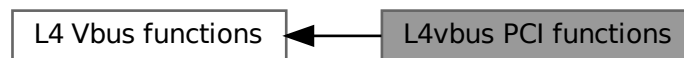
References [L4\\_END\\_DECLS](#).

Referenced by [L4vbus::Gpio\\_pin::to\\_irq\(\)](#).

Here is the caller graph for this function:

**14.8.5 L4vbus PCI functions**

Collaboration diagram for L4vbus PCI functions:

**Functions**

- [L4\\_BEGIN\\_DECLS](#) `int l4vbus_pci_cfg_read (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, l4_uint32_t bus, l4_uint32_t devfn, l4_uint32_t reg, l4_uint32_t *value, l4_uint32_t width)`  
*Read from the vPCI configuration space using the PCI root bridge.*
- `int l4vbus_pci_cfg_write (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, l4_uint32_t bus, l4_uint32_t devfn, l4_uint32_t reg, l4_uint32_t value, l4_uint32_t width)`  
*Write to the vPCI configuration space using the PCI root bridge.*
- `int l4vbus_pci_irq_enable (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, l4_uint32_t bus, l4_uint32_t devfn, int pin, unsigned char *trigger, unsigned char *polarity)`  
*Enable PCI interrupt for a specific device using the PCI root bridge.*
- `int l4vbus_pcidv_cfg_read (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, l4_uint32_t reg, l4_uint32_t *value, l4_uint32_t width)`  
*Read from the device's vPCI configuration space.*
- `int l4vbus_pcidv_cfg_write (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, l4_uint32_t reg, l4_uint32_t value, l4_uint32_t width)`  
*Write to the device's vPCI configuration space.*
- `int l4vbus_pcidv_irq_enable (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, unsigned char *trigger, unsigned char *polarity)`  
*Enable the device's PCI interrupt.*

### 14.8.5.1 Detailed Description

### 14.8.5.2 Function Documentation

#### 14.8.5.2.1 l4vbus\_pci\_cfg\_read()

```
L4_BEGIN_DECLS int l4vbus_pci_cfg_read (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    l4_uint32_t bus,
    l4_uint32_t devfn,
    l4_uint32_t reg,
    l4_uint32_t * value,
    l4_uint32_t width)
```

Read from the vPCI configuration space using the PCI root bridge.

#### Parameters

	<i>vbus</i>	Capability of the system bus
	<i>handle</i>	Device handle of the PCI root bridge
	<i>bus</i>	Bus number
	<i>devfn</i>	Device id (upper 16bit) and function (lower 16bit)
	<i>reg</i>	Register in configuration space to read
out	<i>value</i>	Value that has been read
	<i>width</i>	Width to read in bits (e.g. 8, 16, 32)

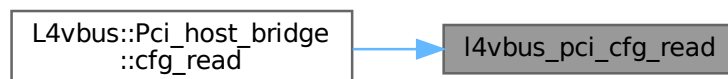
#### Returns

0 on success, else failure

References [L4\\_CV](#).

Referenced by [L4vbus::Pci\\_host\\_bridge::cfg\\_read\(\)](#).

Here is the caller graph for this function:



### 14.8.5.2.2 l4vbus\_pci\_cfg\_write()

```
int l4vbus_pci_cfg_write (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    l4_uint32_t bus,
    l4_uint32_t devfn,
    l4_uint32_t reg,
    l4_uint32_t value,
    l4_uint32_t width)
```

Write to the vPCI configuration space using the PCI root bridge.

#### Parameters

<i>vbus</i>	Capability of the system bus
<i>handle</i>	Device handle of the PCI root bridge
<i>bus</i>	Bus number
<i>devfn</i>	Device id (upper 16bit) and function (lower 16bit)
<i>reg</i>	Register in configuration space to write
<i>value</i>	Value to write
<i>width</i>	Width to write in bits (e.g. 8, 16, 32)

#### Returns

0 on success, else failure

References [L4\\_CV](#).

Referenced by [L4vbus::Pci\\_host\\_bridge::cfg\\_write\(\)](#).

Here is the caller graph for this function:



### 14.8.5.2.3 l4vbus\_pci\_irq\_enable()

```
int l4vbus_pci_irq_enable (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    l4_uint32_t bus,
```

```
l4_uint32_t devfn,  
int pin,  
unsigned char * trigger,  
unsigned char * polarity)
```

Enable PCI interrupt for a specific device using the PCI root bridge.

#### Parameters

---

	<i>vbus</i>	Capability of the system bus
	<i>handle</i>	Device handle of the PCI root bridge
	<i>bus</i>	Bus number
	<i>devfn</i>	Device id (upper 16bit) and function (lower 16bit)
	<i>pin</i>	Interrupt pin (normally as reported in configuration register INTR)
out	<i>trigger</i>	False if interrupt is level-triggered
out	<i>polarity</i>	True if interrupt is of low polarity

#### Returns

On success: Interrupt line to be used, else failure

References [L4\\_CV](#).

Referenced by [L4vbus::Pci\\_host\\_bridge::irq\\_enable\(\)](#).

Here is the caller graph for this function:



#### 14.8.5.2.4 l4vbus\_pciddev\_cfg\_read()

```

int l4vbus_pciddev_cfg_read (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    l4_uint32_t reg,
    l4_uint32_t * value,
    l4_uint32_t width)
  
```

Read from the device's vPCI configuration space.

#### Parameters

	<i>vbus</i>	Capability of the system bus
	<i>handle</i>	Device handle of the PCI device
	<i>reg</i>	Register in configuration space to read
out	<i>value</i>	Value that has been read
	<i>width</i>	Width to read in bits (e.g. 8, 16, 32)

**Returns**

0 on success, else failure

References [L4\\_CV](#).

Referenced by [L4vbus::Pci\\_dev::cfg\\_read\(\)](#).

Here is the caller graph for this function:

**14.8.5.2.5 l4vbus\_pciddev\_cfg\_write()**

```

int l4vbus_pciddev_cfg_write (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    l4_uint32_t reg,
    l4_uint32_t value,
    l4_uint32_t width)
  
```

Write to the device's vPCI configuration space.

**Parameters**

<i>vbus</i>	Capability of the system bus
<i>handle</i>	Device handle of the PCI device
<i>reg</i>	Register in configuration space to write
<i>value</i>	Value to write
<i>width</i>	Width to write in bits (e.g. 8, 16, 32)

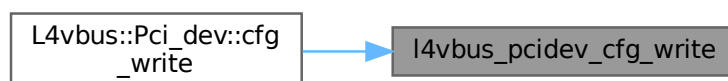
**Returns**

0 on success, else failure

References [L4\\_CV](#).

Referenced by [L4vbus::Pci\\_dev::cfg\\_write\(\)](#).

Here is the caller graph for this function:





#### 14.8.5.2.6 l4vbus\_pcidev\_irq\_enable()

```
int l4vbus_pcidev_irq_enable (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned char * trigger,
    unsigned char * polarity)
```

Enable the device's PCI interrupt.

##### Parameters

	<i>vbus</i>	Capability of the system bus
	<i>handle</i>	Device handle of the PCI device
out	<i>trigger</i>	False if interrupt is level-triggered
out	<i>polarity</i>	True if interrupt is of low polarity

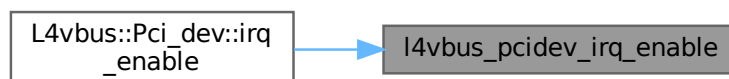
##### Returns

On success: Interrupt line to be used, else failure

References [L4\\_END\\_DECLS](#).

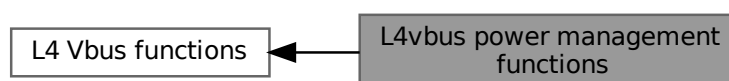
Referenced by [L4vbus::Pci\\_dev::irq\\_enable\(\)](#).

Here is the caller graph for this function:



### 14.8.6 L4vbus power management functions

Collaboration diagram for L4vbus power management functions:



## Functions

- [L4\\_BEGIN\\_DECLS](#) `int l4vbus_pm_suspend (l4_cap_idx_t vbus, l4vbus_device_handle_t handle)`  
*Suspend the device.*
- `int l4vbus_pm_resume (l4_cap_idx_t vbus, l4vbus_device_handle_t handle)`  
*Resume the device.*

### 14.8.6.1 Detailed Description

### 14.8.6.2 Function Documentation

#### 14.8.6.2.1 l4vbus\_pm\_resume()

```
int l4vbus_pm_resume (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle)
```

Resume the device.

#### Parameters

<i>vbus</i>	V-BUS capability
<i>handle</i>	Handle for the device to be resumed Switches the device from low-power mode to normal operation and restores the saved state.

#### Return values

0	Success.
---	----------

References [L4\\_END\\_DECLS](#).

Referenced by [L4vbus::Pm< DEC >::pm\\_resume\(\)](#).

Here is the caller graph for this function:



#### 14.8.6.2.2 l4vbus\_pm\_suspend()

```
L4_BEGIN_DECLS int l4vbus_pm_suspend (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle)
```

Suspend the device.

#### Parameters

<i>vbus</i>	V-BUS capability
<i>handle</i>	Handle for the device to be suspended Saves the state of the device and puts it into a low-power mode.

### Return values

0	Success.
---	----------

References [L4\\_CV](#).

Referenced by [L4vbus::Pm< DEC >::pm\\_suspend\(\)](#).

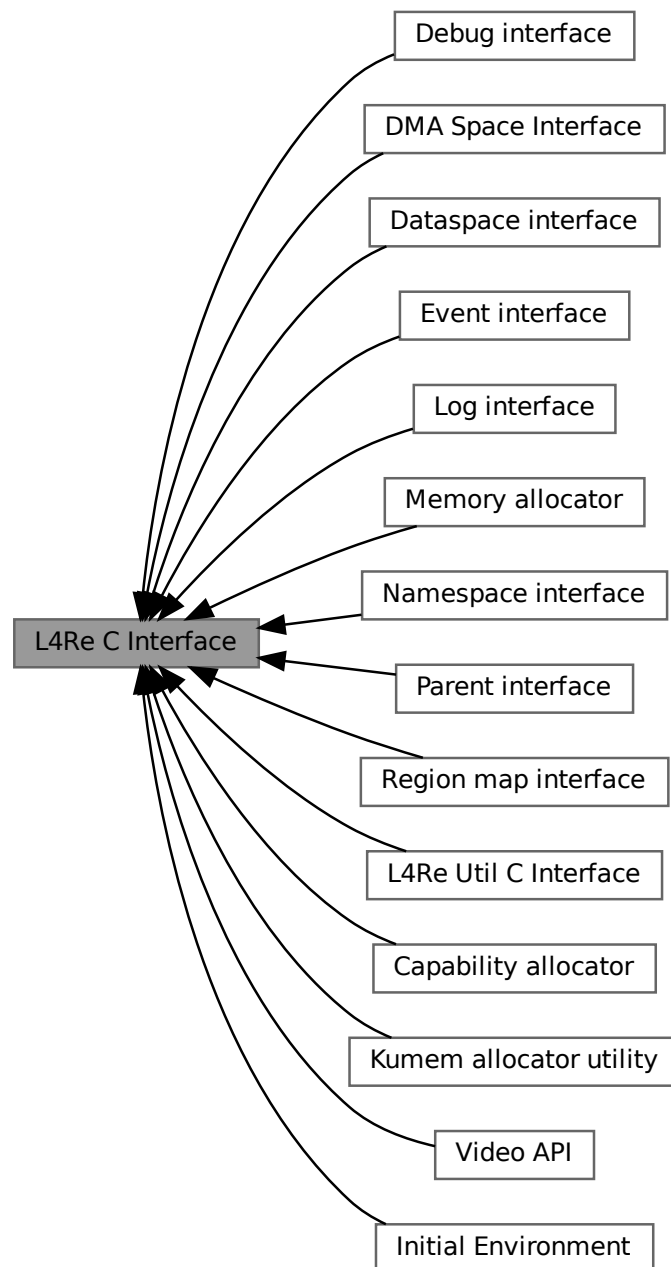
Here is the caller graph for this function:



## 14.9 L4Re C Interface

Documentation for the [L4Re C Interface](#).

Collaboration diagram for L4Re C Interface:



## Topics

- L4Re Util C Interface . . . . . [534](#)  
*Documentation of the [L4](#) Runtime Environment utility functionality in C.*
- Dataspace interface . . . . . [534](#)

<i>Dataspace C interface.</i>	
•	
Debug interface . . . . .	539
•	
DMA Space Interface . . . . .	540
<i>DMA Space C interface.</i>	
•	
Event interface . . . . .	544
<i>Event C interface.</i>	
•	
Log interface . . . . .	547
<i>Log C interface.</i>	
•	
Memory allocator . . . . .	550
<i>Memory allocator C interface.</i>	
•	
Namespace interface . . . . .	556
<i>Namespace C interface.</i>	
•	
Parent interface . . . . .	560
•	
Region map interface . . . . .	560
<i>Region map C interface.</i>	
•	
Capability allocator . . . . .	575
<i>Capability allocator C interface.</i>	
•	
Kumem allocator utility . . . . .	576
<i>Kumem allocator utility C interface.</i>	
•	
Video API . . . . .	577
•	
Initial Environment . . . . .	584
<i>C interface of the initial environment that is provided to an L4 task.</i>	

## Files

- file [inhibitor.h](#)  
*Inhibitor C interface.*

### 14.9.1 Detailed Description

Documentation for the [L4Re C Interface](#).

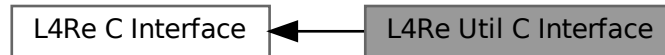
The interface functions closely align with the C++ functions and add no further functionalities.

For new programs it is advised to use the C++ interface.

### 14.9.2 L4Re Util C Interface

Documentation of the [L4](#) Runtime Environment utility functionality in C.

Collaboration diagram for L4Re Util C Interface:



Documentation of the [L4](#) Runtime Environment utility functionality in C.

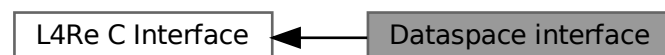
The interface functions closely align with the C++ functions and add no further functionalities.

For new programs it is advised to use the C++ interface.

### 14.9.3 Dataspace interface

Dataspace C interface.

Collaboration diagram for Dataspace interface:



#### Data Structures

- struct [l4re\\_ds\\_stats\\_t](#)  
*Information about the data space.*

#### Enumerations

- enum [l4re\\_ds\\_map\\_flags](#) { }  
*Flags to specify the memory mapping type of a request.*

## Functions

- `long l4re_ds_clear (l4re_ds_t ds, l4re_ds_offset_t offset, l4re_ds_size_t size) L4_NOTHROW`  
*Clear parts of a dataspace.*
- `long l4re_ds_allocate (l4re_ds_t ds, l4re_ds_offset_t offset, l4re_ds_size_t size) L4_NOTHROW`  
*Allocate a range in the dataspace.*
- `int l4re_ds_copy_in (l4re_ds_t ds, l4re_ds_offset_t dst_offs, l4re_ds_t src, l4re_ds_offset_t src_offs, l4re_ds_size_t size) L4_NOTHROW`  
*Copy contents from another dataspace.*
- `l4re_ds_size_t l4re_ds_size (l4re_ds_t ds) L4_NOTHROW`  
*Get size of a dataspace.*
- `l4re_ds_flags_t l4re_ds_flags (l4re_ds_t ds) L4_NOTHROW`  
*Get flags of the dataspace.*
- `int l4re_ds_info (l4re_ds_t ds, l4re_ds_stats_t *stats) L4_NOTHROW`  
*Get information on the dataspace.*
- `int l4re_ds_map_info (l4re_ds_t ds, l4_addr_t *start_addr, l4_addr_t *end_addr) L4_NOTHROW`  
*Get mapping range of dataspace.*

## Variables

- `L4_BEGIN_DECLS typedef l4_cap_idx_t l4re_ds_t`  
*Dataspace type.*

### 14.9.3.1 Detailed Description

Dataspace C interface.

### 14.9.3.2 Enumeration Type Documentation

#### 14.9.3.2.1 l4re\_ds\_map\_flags

```
enum l4re_ds_map_flags
```

Flags to specify the memory mapping type of a request.

#### Enumerator

L4RE_DS_F_NORMAL	request normal memory mapping
L4RE_DS_F_CACHEABLE	request normal memory mapping
L4RE_DS_F_BUFFERABLE	request bufferable (write buffered) mappings
L4RE_DS_F_UNCACHEABLE	request uncacheable memory mappings
L4RE_DS_F_CACHING_MASK	mask for caching flags
L4RE_DS_F_CACHING_SHIFT	shift value for caching flags

Definition at line 48 of file [dataspace.h](#).

### 14.9.3.3 Function Documentation

#### 14.9.3.3.1 l4re\_ds\_allocate()

```
long l4re_ds_allocate (
    l4re_ds_t ds,
    l4re_ds_offset_t offset,
    l4re_ds_size_t size)
```

Allocate a range in the dataspace.

##### Parameters

<i>ds</i>	Dataspace capability.
<i>offset</i>	Offset in the dataspace, in bytes.
<i>size</i>	Size of the range, in bytes.

##### Return values

<i>L4_EOK</i>	Success
<i>-L4_ERANGE</i>	Given range is outside the dataspace. (A dataspace provider may also silently ignore areas outside the dataspace.)
<i>-L4_ENOMEM</i>	Not enough memory available.
<i>&lt;0</i>	IPC errors

On success, at least the given range is guaranteed to be allocated. The dataspace manager may also allocate more memory due to page granularity.

The memory is allocated with the same rights as the dataspace capability.

References [L4\\_CV](#), [L4\\_NOTHROW](#), and [l4re\\_ds\\_t](#).

#### 14.9.3.3.2 l4re\_ds\_clear()

```
long l4re_ds_clear (
    l4re_ds_t ds,
    l4re_ds_offset_t offset,
    l4re_ds_size_t size)
```

Clear parts of a dataspace.

##### Parameters

<i>ds</i>	Dataspace capability.
<i>offset</i>	Offset within dataspace (in bytes).
<i>size</i>	Size of region to clear (in bytes).

##### Return values



$\geq 0$	Success.
<code>-L4_ERANGE</code>	Given range is outside the dataspace. (A dataspace provider may also silently ignore areas outside the dataspace.)
<code>-L4_EACCESS</code>	No <a href="#">L4_CAP_FPAGE_W</a> right on dataspace capability.
$< 0$	IPC errors

Zeroes out the memory. Depending on the type of memory the memory could also be deallocated and replaced by a shared zero-page.

References [L4\\_CV](#), [L4\\_NOTHROW](#), and [l4re\\_ds\\_t](#).

#### 14.9.3.3.3 l4re\_ds\_copy\_in()

```
int l4re_ds_copy_in (
    l4re_ds_t ds,
    l4re_ds_offset_t dst_offs,
    l4re_ds_t src,
    l4re_ds_offset_t src_offs,
    l4re_ds_size_t size)
```

Copy contents from another dataspace.

##### Parameters

<i>ds</i>	Destination dataspace.
<i>dst_offs</i>	Offset in destination dataspace.
<i>src</i>	Source dataspace to copy from.
<i>src_offs</i>	Offset in the source dataspace.
<i>size</i>	Size to copy (in bytes).

##### Return values

<code>L4_EOK</code>	Success
<code>-L4_EACCESS</code>	No <a href="#">L4_CAP_FPAGE_W</a> right on the destination dataspace.
<code>-L4_EINVAL</code>	Invalid parameter supplied.
$< 0$	IPC errors

The copy operation may use copy-on-write mechanisms. The operation may also fail if both dataspaces are not from the same dataspace manager or the dataspace managers do not cooperate.

References [L4\\_CV](#), [L4\\_NOTHROW](#), and [l4re\\_ds\\_t](#).

#### 14.9.3.3.4 l4re\_ds\_flags()

```
l4re_ds_flags_t l4re_ds_flags (
    l4re_ds_t ds)
```

Get flags of the dataspace.

##### Parameters

<i>ds</i>	Dataspace capability.
-----------	-----------------------

### Return values

$\geq 0$	Flags of the dataspace
$< 0$	IPC errors

### See also

[L4Re::Dataspace::F::Flags](#)

References [L4\\_CV](#), [L4\\_NOTHROW](#), and [l4re\\_ds\\_t](#).

#### 14.9.3.3.5 l4re\_ds\_info()

```
int l4re_ds_info (
    l4re_ds_t ds,
    l4re_ds_stats_t * stats)
```

Get information on the dataspace.

### Parameters

	<i>ds</i>	Dataspace capability.
<i>out</i>	<i>stats</i>	Dataspace information

### Return values

$0$	Success
$< 0$	IPC errors

References [L4\\_CV](#), [L4\\_NOTHROW](#), and [l4re\\_ds\\_t](#).

#### 14.9.3.3.6 l4re\_ds\_map\_info()

```
int l4re_ds_map_info (
    l4re_ds_t ds,
    l4_addr_t * start_addr,
    l4_addr_t * end_addr)
```

Get mapping range of dataspace.

### Parameters

<i>ds</i>	Dataspace capability. In case of a MMU-less system, the dataspace must be mapped at the correct address in the task because virtual and physical address must match. This method returns the start and end address of the physically contiguous buffer backing the dataspace.
-----------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

On MMU-enabled system any page aligned address is permissible. On such systems the method is just a stub.

#### Parameters

out	<i>start_addr</i>	Start address of dataspace.
out	<i>end_addr</i>	End address (inclusive) of dataspace.

#### Return values

$>0$	Start/end address have been set and need to be obeyed.
$0$	No constraint of mapping address.
$-L4\_EPMR$	Cannot infer mapping address. Dataspace not mappable.
$<0$	IPC errors.

References [L4\\_END\\_DECLS](#), [L4\\_NOTHROW](#), and [l4re\\_ds\\_t](#).

#### 14.9.3.3.7 l4re\_ds\_size()

```
l4re_ds_size_t l4re_ds_size (
    l4re_ds_t ds)
```

Get size of a dataspace.

#### Parameters

<i>ds</i>	Dataspace capability.
-----------	-----------------------

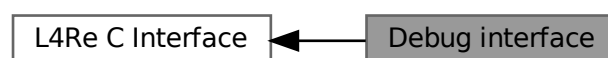
#### Returns

Size of the dataspace in bytes.

References [L4\\_CV](#), [L4\\_NOTHROW](#), and [l4re\\_ds\\_t](#).

### 14.9.4 Debug interface

Collaboration diagram for Debug interface:



## Functions

- [L4\\_BEGIN\\_DECLS](#) long [l4re\\_debug\\_obj\\_debug](#) ([l4\\_cap\\_idx\\_t](#) *srv*, unsigned long *function*) [L4\\_NOTHROW](#)  
Call debug function of [L4Re](#) service.

### 14.9.4.1 Detailed Description

### 14.9.4.2 Function Documentation

#### 14.9.4.2.1 [l4re\\_debug\\_obj\\_debug\(\)](#)

```
L4\_BEGIN\_DECLS long l4re\_debug\_obj\_debug (  
    l4\_cap\_idx\_t srv,  
    unsigned long function)
```

Call debug function of [L4Re](#) service.

#### Parameters

<i>srv</i>	Object to call.
<i>function</i>	Function to call.

See also

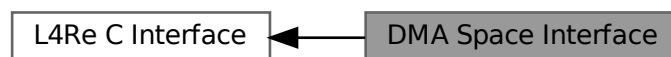
[L4Re::Debug\\_obj::debug](#)

References [L4\\_END\\_DECLS](#), and [L4\\_NOTHROW](#).

## 14.9.5 DMA Space Interface

DMA Space C interface.

Collaboration diagram for DMA Space Interface:



## Typedefs

- typedef [l4\\_cap\\_idx\\_t](#) [l4re\\_dma\\_space\\_t](#)  
DMA space capability type.

## Functions

- long [l4re\\_dma\\_space\\_map](#) ([l4re\\_dma\\_space\\_t](#) dma, [l4re\\_ds\\_t](#) src, [l4re\\_ds\\_offset\\_t](#) offset, [l4\\_size\\_t](#) \*size, unsigned long attrs, enum [l4re\\_dma\\_space\\_direction](#) dir, [l4re\\_dma\\_space\\_dma\\_addr\\_t](#) \*dma\_addr) [L4\\_NOTHROW](#)  
*Map the given part of this data space into the DMA address space.*
- long [l4re\\_dma\\_space\\_unmap](#) ([l4re\\_dma\\_space\\_t](#) dma, [l4re\\_dma\\_space\\_dma\\_addr\\_t](#) dma\_addr, [l4\\_size\\_t](#) size, unsigned long attrs, enum [l4re\\_dma\\_space\\_direction](#) dir) [L4\\_NOTHROW](#)  
*Unmap the given part of this data space from the DMA address space.*
- long [l4re\\_dma\\_space\\_associate](#) ([l4re\\_dma\\_space\\_t](#) dma, [l4\\_cap\\_idx\\_t](#) dma\_task, unsigned long attr) [L4\\_NOTHROW](#)  
*Associate a (kernel) DMA space for a device to this Dma\_space.*
- long [l4re\\_dma\\_space\\_disassociate](#) ([l4re\\_dma\\_space\\_t](#) dma)  
*Disassociate the (kernel) DMA space from this Dma\_space.*

### 14.9.5.1 Detailed Description

DMA Space C interface.

### 14.9.5.2 Typedef Documentation

#### 14.9.5.2.1 l4re\_dma\_space\_t

```
typedef l4_cap_idx_t l4re_dma_space_t
```

DMA space capability type.

Managed DMA Address Space.

A managed Dma\_space represents the [L4Re](#) abstraction of an DMA address space of one or several devices. Devices are assigned to a managed Dma\_space by binding the Dma\_space to the respective DMA domain (see [L4vbus::Vbus::assign\\_dma\\_domain\(\)](#)), which might link the Dma\_space with a kernel [DMA space](#). Note that several DMA domains can be bound to the same Dma\_space. Whenever a device needs direct access to parts of an [L4Re::Dataspace](#), that part of the data space must be mapped to the managed Dma\_space that is assigned to that device. Binding to DMA domains must happen before mapping. After the DMA accesses to the memory are finished the memory must be unmapped from the device's DMA address space.

Mapping to a managed DMA address space, using [map\(\)](#), makes the given parts of the data space visible to the associated device at the returned DMA address. As long as the memory is mapped into a DMA space it is 'pinned' and cannot be subject to dynamic memory management such as swapping. Additionally, [map\(\)](#) is responsible for the necessary syncing operations before the DMA.

[unmap\(\)](#) is the reverse operation to [map\(\)](#) and unmaps the given data-space part for the DMA address space. [unmap\(\)](#) is responsible for the necessary sync operations after the DMA.

Definition at line 49 of file [dma\\_space.h](#).

### 14.9.5.3 Function Documentation

#### 14.9.5.3.1 l4re\_dma\_space\_associate()

```
long l4re_dma_space_associate (
    l4re_dma_space_t dma,
    l4_cap_idx_t dma_task,
    unsigned long attr)
```

Associate a (kernel) [DMA space](#) for a device to this Dma\_space.

## Parameters

	<i>dma</i>	DMA space capability
in	<i>dma_task</i>	The (kernel) <a href="#">DMA space</a> used for the device that shall be associated with this DMA space. In case no IOMMU is present or configured, the <i>dma_task</i> might be an invalid capability when <a href="#">L4Re::Dma_space::Phys_space</a> is set in <i>attr</i> , in this case the CPUs physical memory is used as DMA address space.
in	<i>attr</i>	Attributes for this DMA space. See <a href="#">L4Re::Dma_space::Space_attr</a> .

### Return values

<i>L4_EOK</i>	Operation successful.
<i>-L4_EPERM</i>	Insufficient permissions; see precondition.
<i>-L4_EINVAL</i>	
<i>-L4_ENOENT</i>	

### Precondition

The invoked *Dma\_space* capability must have the permission [L4\\_CAP\\_FPAGE\\_W](#).

References [L4\\_CV](#), and [L4\\_NOTHROW](#).

#### 14.9.5.3.2 *l4re\_dma\_space\_disassociate()*

```
long l4re_dma_space_disassociate (
    l4re_dma_space_t dma)
```

Disassociate the (kernel) [DMA space](#) from this *Dma\_space*.

### Parameters

<i>dma</i>	DMA space capability
------------	----------------------

### Return values

<i>L4_EOK</i>	Operation successful.
<i>-L4_EPERM</i>	Insufficient permissions; see precondition.
<i>-L4_ENOENT</i>	

### Precondition

The invoked *Dma\_space* capability must have the permission [L4\\_CAP\\_FPAGE\\_W](#).

References [L4\\_END\\_DECLS](#).

### 14.9.5.3.3 l4re\_dma\_space\_map()

```
long l4re_dma_space_map (
    l4re_dma_space_t dma,
    l4re_ds_t src,
    l4re_ds_offset_t offset,
    l4_size_t * size,
    unsigned long attrs,
    enum l4re_dma_space_direction dir,
    l4re_dma_space_dma_addr_t * dma_addr)
```

Map the given part of this data space into the DMA address space.

#### Parameters

	<i>dma</i>	DMA space capability
in	<i>src</i>	Source data space (that describes the memory). Caller needs write right to the data space.
in	<i>offset</i>	The offset (bytes) within <i>src</i> .
in, out	<i>size</i>	The size (bytes) of the region to be mapped for DMA, after successful mapping the size returned is the size mapped for DMA as a single block. This size might be smaller than the original input size, in this case the caller might call <code>map()</code> again with a new offset and the remaining size.
in	<i>attrs</i>	The attributes used for this DMA mapping (a combination of <code>Dma_space::Attribute</code> values).
in	<i>dir</i>	The direction of the DMA transfer issued with this mapping. The same value must later be passed to <code>unmap()</code> .
out	<i>dma_addr</i>	The DMA address to use for DMA with the associated device.

#### Return values

<i>L4_EOK</i>	Operation successful.
<i>-L4_EPERM</i>	Insufficient permissions; see precondition.
<i>-L4_EINVAL</i>	The capability <i>src</i> is invalid or does not refer to a valid dataspace.
<i>-L4_EEXIST</i>	The specified region overlaps an existing mapping.
<i>-L4_ENOMEM</i>	Not enough memory to allocate internal datastructures.
<i>-L4_ERANGE</i>	<i>offset</i> is larger than the size of the dataspace.

#### Precondition

The capability *src* must have the permission [L4\\_CAP\\_FPAGE\\_W](#).

#### Note

`associate()` must be called prior to mapping memory. Usually this is done implicitly when binding the managed `Dma_space` to a DMA domain (see [L4vbus::Vbus::assign\\_dma\\_domain\(\)](#)).

References [L4\\_CV](#), [L4\\_NOTHROW](#), and [l4re\\_ds\\_t](#).

#### 14.9.5.3.4 l4re\_dma\_space\_unmap()

```
long l4re_dma_space_unmap (
    l4re_dma_space_t dma,
    l4re_dma_space_dma_addr_t dma_addr,
    l4_size_t size,
    unsigned long attrs,
    enum l4re_dma_space_direction dir)
```

Unmap the given part of this data space from the DMA address space.

#### Parameters

<i>dma</i>	DMA space capability
<i>dma_addr</i>	The DMA address (returned by <code>Dma_space::map()</code> ).
<i>size</i>	The size (bytes) of the memory region to unmap.
<i>attrs</i>	The attributes for the unmap (currently none).
<i>dir</i>	The direction of the finished DMA operation.

#### Returns

0 in the case of success, a negative error code otherwise.

References [L4\\_CV](#), and [L4\\_NOTHROW](#).

### 14.9.6 Event interface

Event C interface.

Collaboration diagram for Event interface:



#### Functions

- long [l4re\\_event\\_get\\_buffer](#) (const [l4\\_cap\\_idx\\_t](#) server, const [l4re\\_ds\\_t](#) ds) [L4\\_NOTHROW](#)  
*Get an event signal buffer.*
- long [l4re\\_event\\_get\\_num\\_streams](#) (const [l4\\_cap\\_idx\\_t](#) server) [L4\\_NOTHROW](#)  
*Get number of streams.*
- long [l4re\\_event\\_get\\_stream\\_info](#) (const [l4\\_cap\\_idx\\_t](#) server, int idx, [l4re\\_event\\_stream\\_info\\_t](#) \*info) [L4\\_NOTHROW](#)  
*Get information on a stream.*
- long [l4re\\_event\\_get\\_stream\\_info\\_for\\_id](#) (const [l4\\_cap\\_idx\\_t](#) server, [l4\\_umword\\_t](#) stream\_id, [l4re\\_event\\_stream\\_info\\_t](#) \*info) [L4\\_NOTHROW](#)  
*Get info for a stream given a stream id.*
- long [l4re\\_event\\_get\\_axis\\_info](#) (const [l4\\_cap\\_idx\\_t](#) server, [l4\\_umword\\_t](#) id, unsigned naxes, unsigned const \*axis, [l4re\\_event\\_absinfo\\_t](#) \*info) [L4\\_NOTHROW](#)  
*Get Axis information for a stream.*



### 14.9.6.1 Detailed Description

Event C interface.

### 14.9.6.2 Function Documentation

#### 14.9.6.2.1 l4re\_event\_get\_axis\_info()

```
long l4re_event_get_axis_info (
    const l4_cap_idx_t server,
    l4_umword_t id,
    unsigned naxes,
    unsigned const * axis,
    l4re_event_absinfo_t * info)
```

Get Axis information for a stream.

#### Parameters

	<i>server</i>	Server to talk to.
	<i>id</i>	Id of the stream to get information from.
	<i>naxes</i>	Number of axes in <i>axis</i> array.
in	<i>axis</i>	Array of axis IDs whose information should be retrieved.
out	<i>info</i>	Information buffer to store the retrieved axis infos.

#### Return values

0	Success
<0	Error

See also

[L4Re::Event::get\\_axis\\_info](#)

References [L4\\_END\\_DECLS](#), and [L4\\_NOTHROW](#).

#### 14.9.6.2.2 l4re\_event\_get\_buffer()

```
long l4re_event_get_buffer (
    const l4_cap_idx_t server,
    const l4re_ds_t ds)
```

Get an event signal buffer.

#### Parameters

<i>server</i>	Server to talk to.
<i>ds</i>	<a href="#">Buffer</a> to event data.

#### Returns

0 for success, <0 on error

#### See also

[L4Re::Event::get\\_buffer](#)

References [L4\\_CV](#), [L4\\_NOTHROW](#), and [l4re\\_ds\\_t](#).

#### 14.9.6.2.3 l4re\_event\_get\_num\_streams()

```
long l4re_event_get_num_streams (  
    const l4\_cap\_idx\_t server)
```

Get number of streams.

#### Parameters

<i>server</i>	Server to talk to.
---------------	--------------------

#### Returns

0 for success, <0 on error

#### See also

[L4Re::Event::get\\_num\\_streams](#)

References [L4\\_CV](#), and [L4\\_NOTHROW](#).

#### 14.9.6.2.4 l4re\_event\_get\_stream\_info()

```
long l4re_event_get_stream_info (  
    const l4\_cap\_idx\_t server,  
    int idx,  
    l4re\_event\_stream\_info\_t * info)
```

Get information on a stream.

#### Parameters

---

	<i>server</i>	Server to talk to.
	<i>idx</i>	Index value.
out	<i>info</i>	Information buffer.

**Returns**

0 for success, <0 on error

**See also**

[L4Re::Event::get\\_stream\\_info](#)

References [L4\\_CV](#), and [L4\\_NOTHROW](#).

**14.9.6.2.5 l4re\_event\_get\_stream\_info\_for\_id()**

```
long l4re_event_get_stream_info_for_id (
    const l4_cap_idx_t server,
    l4_umword_t stream_id,
    l4re_event_stream_info_t * info)
```

Get info for a stream given a stream id.

**Parameters**

	<i>server</i>	Server to talk to.
	<i>stream↔ _id</i>	Stream ID.
out	<i>info</i>	Information buffer.

**Returns**

0 for success, <0 on error

**See also**

[L4Re::Event::get\\_stream\\_info\\_for\\_id](#)

References [L4\\_CV](#), and [L4\\_NOTHROW](#).

**14.9.7 Log interface**

Log C interface.

Collaboration diagram for Log interface:



## Functions

- [L4\\_BEGIN\\_DECLS](#) void [l4re\\_log\\_print](#) (char const \*string) [L4\\_NOTHROW](#)  
Write a null terminated string to the default log.
- void [l4re\\_log\\_printn](#) (char const \*string, int len) [L4\\_NOTHROW](#)  
Write a string of a given length to the default log.
- void [l4re\\_log\\_print\\_srv](#) (const [l4\\_cap\\_idx\\_t](#) logcap, char const \*string) [L4\\_NOTHROW](#)  
Write a null terminated string to a log.
- void [l4re\\_log\\_printn\\_srv](#) (const [l4\\_cap\\_idx\\_t](#) logcap, char const \*string, int len) [L4\\_NOTHROW](#)  
Write a string of a given length to a log.

### 14.9.7.1 Detailed Description

Log C interface.

### 14.9.7.2 Function Documentation

#### 14.9.7.2.1 l4re\_log\_print()

```
void l4re_log_print (
    char const * string) [inline]
```

Write a null terminated string to the default log.

#### Parameters

<i>string</i>	Text to print, null terminated.
---------------	---------------------------------

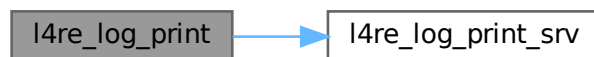
See also

[L4Re::Log::print](#)

Definition at line 81 of file [log.h](#).

References [L4\\_NOTHROW](#), and [l4re\\_log\\_print\\_srv\(\)](#).

Here is the call graph for this function:



#### 14.9.7.2.2 l4re\_log\_print\_srv()

```
void l4re_log_print_srv (
    const l4\_cap\_idx\_t logcap,
    char const * string)
```

Write a null terminated string to a log.

#### Parameters

<i>logcap</i>	Log capability (service).
<i>string</i>	Text to print, null terminated.

See also

[L4Re::Log::print](#)

References [L4\\_CV](#), and [L4\\_NOTHROW](#).

Referenced by [l4re\\_log\\_print\(\)](#).

Here is the caller graph for this function:



#### 14.9.7.2.3 l4re\_log\_printn()

```
void l4re_log_printn (  
    char const * string,  
    int len) [inline]
```

Write a string of a given length to the default log.

##### Parameters

<i>string</i>	Text to print, null terminated.
<i>len</i>	Length of string in bytes.

See also

[L4Re::Log::printn](#)

Definition at line 87 of file [log.h](#).

References [L4\\_NOTHROW](#), and [l4re\\_log\\_printn\\_srv\(\)](#).

Here is the call graph for this function:



#### 14.9.7.2.4 l4re\_log\_printn\_srv()

```
void l4re_log_printn_srv (
    const l4_cap_idx_t logcap,
    char const * string,
    int len)
```

Write a string of a given length to a log.

##### Parameters

<i>logcap</i>	Log capability (service).
<i>string</i>	Text to print, null terminated.
<i>len</i>	Length of string in bytes.

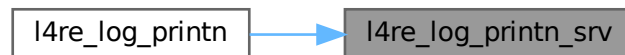
##### See also

[L4Re::Log::printn](#)

References [L4\\_CV](#), [L4\\_INLINE](#), and [L4\\_NOTHROW](#).

Referenced by [l4re\\_log\\_printn\(\)](#).

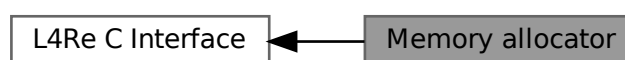
Here is the caller graph for this function:



### 14.9.8 Memory allocator

Memory allocator C interface.

Collaboration diagram for Memory allocator:



## Enumerations

- enum [l4re\\_ma\\_flags](#)  
*Flags for requesting memory at the memory allocator.*

## Functions

- long [l4re\\_ma\\_alloc](#) (long size, [l4re\\_ds\\_t](#) const mem, unsigned long flags) [L4\\_NOTHROW](#)  
*Allocate memory.*
- long [l4re\\_ma\\_alloc\\_align](#) (long size, [l4re\\_ds\\_t](#) const mem, unsigned long flags, unsigned long align) [L4\\_NOTHROW](#)  
*Allocate memory.*
- long [l4re\\_ma\\_alloc\\_align\\_srv](#) ([l4\\_cap\\_idx\\_t](#) srv, long size, [l4re\\_ds\\_t](#) const mem, unsigned long flags, unsigned long align) [L4\\_NOTHROW](#)  
*Allocate memory.*

### 14.9.8.1 Detailed Description

Memory allocator C interface.

### 14.9.8.2 Enumeration Type Documentation

#### 14.9.8.2.1 l4re\_ma\_flags

```
enum l4re_ma_flags
```

Flags for requesting memory at the memory allocator.

See also

[L4Re::Mem\\_alloc::Mem\\_alloc\\_flags](#)

Definition at line 32 of file [mem\\_alloc.h](#).

### 14.9.8.3 Function Documentation

#### 14.9.8.3.1 l4re\_ma\_alloc()

```
long l4re_ma_alloc (
    long size,
    l4re_ds_t const mem,
    unsigned long flags) [inline]
```

Allocate memory.

## Parameters

<i>size</i>	Size in bytes to be requested. Allocation granularity is (super)pages, however, the allocator will store the byte-granular given size as the size of the dataspace and consecutively will use this byte-granular size for servicing the dataspace. Allocators may optionally also implement a maximum allocation strategy: if <i>size</i> is a negative value and <i>flags</i> set the <code>Mem_alloc_flags::Continuous</code> bit, the allocator tries to allocate as much memory as possible leaving an amount of at least <code>-size</code> bytes within the associated quota.
<i>mem</i>	Capability slot where the capability to the dataspace is received.
<i>flags</i>	Special dataspace properties, see <a href="#">l4re_ma_flags</a>

### Return values

<i>0</i>	Success
<i>-L4_ERANGE</i>	Given size not supported.
<i>-L4_ENOMEM</i>	Not enough memory available.
<i>&lt;0</i>	IPC error

### See also

[L4Re::Mem\\_alloc::alloc](#)

The memory allocator returns a dataspace.

### Note

This function is using the [L4Re::Env::env\(\)](#)->`mem_alloc()` service.

### Examples

[examples/libs/l4re/c/ma+rm.c](#).

Definition at line 136 of file [mem\\_alloc.h](#).

References [L4\\_NOTHROW](#), [l4re\\_ds\\_t](#), and [l4re\\_ma\\_alloc\\_align\\_srv\(\)](#).

Here is the call graph for this function:





#### 14.9.8.3.2 l4re\_ma\_alloc\_align()

```
long l4re_ma_alloc_align (  
    long size,  
    l4re_ds_t const mem,  
    unsigned long flags,  
    unsigned long align) [inline]
```

Allocate memory.

#### Parameters

---

<i>size</i>	Size in bytes to be requested. Allocation granularity is (super)pages, however, the allocator will store the byte-granular given size as the size of the dataspace and consecutively will use this byte-granular size for servicing the dataspace. Allocators may optionally also implement a maximum allocation strategy: if <i>size</i> is a negative value and <i>flags</i> set the <code>Mem_alloc_flags::Continuous</code> bit, the allocator tries to allocate as much memory as possible leaving an amount of at least <code>-size</code> bytes within the associated quota.
<i>mem</i>	Capability slot where the capability to the dataspace is received.
<i>flags</i>	Special dataspace properties, see <a href="#">l4re_ma_flags</a>
<i>align</i>	Log2 alignment of dataspace if supported by allocator, will be at least <code>L4_PAGESHIFT</code> , with <code>Super_↔</code> pages flag set at least <code>L4_SUPERPAGESHIFT</code>

### Return values

<code>0</code>	Success
<code>-L4_ERANGE</code>	Given size not supported.
<code>-L4_ENOMEM</code>	Not enough memory available.
<code>&lt;0</code>	IPC error

### See also

[L4Re::Mem\\_alloc::alloc](#) and  
[l4re\\_ma\\_alloc](#)

The memory allocator returns a dataspace.

### Note

This function is using the [L4Re::Env::env\(\)](#)->`mem_alloc()` service.

Definition at line [144](#) of file [mem\\_alloc.h](#).

References [L4\\_NOTHROW](#), [l4re\\_ds\\_t](#), and [l4re\\_ma\\_alloc\\_align\\_srv\(\)](#).

Here is the call graph for this function:



#### 14.9.8.3.3 l4re\_ma\_alloc\_align\_srv()

```
long l4re_ma_alloc_align_srv (  
    l4_cap_idx_t srv,  
    long size,  
    l4re_ds_t const mem,  
    unsigned long flags,  
    unsigned long align)
```

Allocate memory.

#### Parameters

---

<i>srv</i>	Memory allocator service.
<i>size</i>	Size to be requested.
<i>mem</i>	Capability slot to put the requested dataspace in
<i>flags</i>	Flags, see <a href="#">l4re_ma_flags</a>
<i>align</i>	Log2 alignment of dataspace if supported by allocator, will be at least L4_PAGESHIFT, with Super_↔ pages flag set at least L4_SUPERPAGESHIFT, default 0

#### Returns

0 on success, <0 on error

#### See also

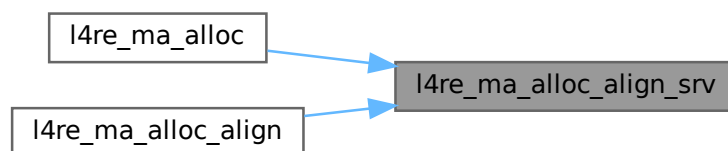
[L4Re::Mem\\_alloc::alloc](#)

The memory allocator returns a dataspace.

References [L4\\_CV](#), [L4\\_INLINE](#), [L4\\_NOTHROW](#), and [l4re\\_ds\\_t](#).

Referenced by [l4re\\_ma\\_alloc\(\)](#), and [l4re\\_ma\\_alloc\\_align\(\)](#).

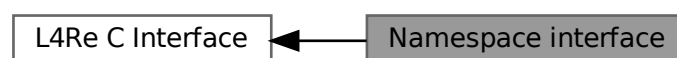
Here is the caller graph for this function:



### 14.9.9 Namespace interface

Namespace C interface.

Collaboration diagram for Namespace interface:



## Enumerations

- enum [l4re\\_ns\\_register\\_flags](#)  
*Namespace register flags.*

## Functions

- long [l4re\\_ns\\_query\\_to\\_srv](#) ([l4re\\_namespace\\_t](#) srv, char const \*name, [l4\\_cap\\_idx\\_t](#) const cap, int timeout) [L4\\_NOTHROW](#)  
*Query the name space for the object named by `name`.*
- long [l4re\\_ns\\_query\\_srv](#) ([l4re\\_namespace\\_t](#) srv, char const \*name, [l4\\_cap\\_idx\\_t](#) const cap) [L4\\_NOTHROW](#)  
*Query the name space for the object named by `name`.*
- long [l4re\\_ns\\_register\\_obj\\_srv](#) ([l4re\\_namespace\\_t](#) srv, char const \*name, [l4\\_cap\\_idx\\_t](#) const obj, unsigned flags) [L4\\_NOTHROW](#)  
*Register an object with a name.*

## Variables

- [L4\\_BEGIN\\_DECLS](#) typedef [l4\\_cap\\_idx\\_t](#) [l4re\\_namespace\\_t](#)  
*Namespace type.*

### 14.9.9.1 Detailed Description

Namespace C interface.

### 14.9.9.2 Enumeration Type Documentation

#### 14.9.9.2.1 l4re\_ns\_register\_flags

```
enum l4re_ns_register_flags
```

Namespace register flags.

See also

[L4Re::Namespace::Register\\_flags](#)

Definition at line 29 of file [namespace.h](#).

### 14.9.9.3 Function Documentation

#### 14.9.9.3.1 l4re\_ns\_query\_srv()

```
long l4re_ns_query_srv (  
    l4re\_namespace\_t srv,  
    char const * name,  
    l4\_cap\_idx\_t const cap) [inline]
```

Query the name space for the object named by `name`.

## Parameters

---

<i>srv</i>	Name space server to use for the query.
<i>name</i>	String to query.
<i>cap</i>	Capability slot where the received capability will be stored.

### Return values

<i>0</i>	Name could be fully resolved.
<i>&gt;0</i>	Name could only be partly resolved. The number of remaining characters is returned.
<i>-L4_ENOENT</i>	Entry could not be found.
<i>-L4_EAGAIN</i>	Entry exists but no object is yet attached. Try again later.
<i>&lt;0</i>	IPC errors, see <a href="#">l4_error_code_t</a> .

Definition at line 95 of file [namespace.h](#).

References [L4\\_NOTHROW](#), [l4re\\_namespace\\_t](#), and [l4re\\_ns\\_query\\_to\\_srv\(\)](#).

Here is the call graph for this function:



#### 14.9.9.3.2 l4re\_ns\_query\_to\_srv()

```

long l4re_ns_query_to_srv (
    l4re_namespace_t srv,
    char const * name,
    l4_cap_idx_t const cap,
    int timeout)
  
```

Query the name space for the object named by *name*.

### Parameters

<i>timeout</i>	Timeout of query in milliseconds. The client will only wait if a name already has been registered with the server but no object has been attached yet.
<i>srv</i>	Name space server to use for the query.
<i>name</i>	String to query.
<i>cap</i>	Capability slot where the received capability will be stored.

### Return values

<i>0</i>	Name could be fully resolved.
<i>&gt;0</i>	Name could only be partly resolved. The number of remaining characters is returned.
<i>-L4_ENOENT</i>	Entry could not be found.
<i>-L4_EAGAIN</i>	Entry exists but no object is yet attached. Try again later.
<i>&lt;0</i>	IPC errors, see <a href="#">l4_error_code_t</a> .

References [L4\\_CV](#), [L4\\_INLINE](#), [L4\\_NOTHROW](#), and [l4re\\_namespace\\_t](#).

Referenced by [l4re\\_ns\\_query\\_srv\(\)](#).

Here is the caller graph for this function:



#### 14.9.9.3.3 l4re\_ns\_register\_obj\_srv()

```

long l4re_ns_register_obj_srv (
    l4re_namespace_t srv,
    char const * name,
    l4_cap_idx_t const obj,
    unsigned flags)
  
```

Register an object with a name.

##### Parameters

<i>srv</i>	Name space server to use for the query.
<i>name</i>	Name under which the object should be registered.
<i>obj</i>	Capability to object to register. An invalid capability may be given to only reserve the name for later use.
<i>flags</i>	Flags to assign to the entry, see <a href="#">L4Re::Namespace::Register_flags</a> . Note that the rights that are assigned to a capability are not only determined by the rights given in these flags but also by the rights with which the <code>obj</code> capability was mapped to the name space.

##### Return values

<i>0</i>	Object was successfully registered with <i>name</i> .
<i>-L4_EEXIST</i>	Name already registered.
<i>-L4_EPERM</i>	Insufficient permissions; see precondition.

<code>-L4_ENOMEM</code>	Server has insufficient resources.
<code>-L4_EINVAL</code>	Invalid parameter.
<code>&lt; 0</code>	IPC errors, see <a href="#">l4_error_code_t</a> .

#### Precondition

The invoked Namespace capability must have the permission [L4\\_CAP\\_FPAGE\\_W](#).

References [L4\\_CV](#), [L4\\_INLINE](#), [L4\\_NOTHROW](#), and [l4re\\_namespace\\_t](#).

### 14.9.10 Parent interface

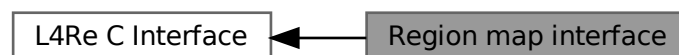
Collaboration diagram for Parent interface:



### 14.9.11 Region map interface

Region map C interface.

Collaboration diagram for Region map interface:



#### Enumerations

- enum [l4re\\_rm\\_flags\\_values](#) {  
[L4RE\\_RM\\_F\\_R](#) = [L4RE\\_DS\\_F\\_R](#) , [L4RE\\_RM\\_F\\_W](#) = [L4RE\\_DS\\_F\\_W](#) , [L4RE\\_RM\\_F\\_X](#) = [L4RE\\_DS\\_F\\_X](#)  
, [L4RE\\_RM\\_F\\_RX](#) = [L4RE\\_DS\\_F\\_RX](#) ,  
[L4RE\\_RM\\_F\\_RW](#) = [L4RE\\_DS\\_F\\_RW](#) , [L4RE\\_RM\\_F\\_RWX](#) = [L4RE\\_DS\\_F\\_RWX](#) , [L4RE\\_RM\\_F\\_KERNEL](#)  
= 0x100 , [L4RE\\_RM\\_F\\_DETACH\\_FREE](#) = 0x200 ,  
[L4RE\\_RM\\_F\\_PAGER](#) = 0x400 , [L4RE\\_RM\\_F\\_RESERVED](#) = 0x800 , [L4RE\\_RM\\_CACHING\\_SHIFT](#) = 4 ,  
[L4RE\\_RM\\_F\\_CACHING](#) = [L4RE\\_DS\\_F\\_CACHING\\_MASK](#) ,  
[L4RE\\_RM\\_REGION\\_FLAGS](#) = 0xffff , [L4RE\\_RM\\_F\\_CACHE\\_NORMAL](#) = [L4RE\\_DS\\_F\\_NORMAL](#) ,  
[L4RE\\_RM\\_F\\_CACHE\\_BUFFERED](#) = [L4RE\\_DS\\_F\\_BUFFERABLE](#) , [L4RE\\_RM\\_F\\_CACHE\\_UNCACHED](#)  
= [L4RE\\_DS\\_F\\_UNCACHEABLE](#) ,  
[L4RE\\_RM\\_F\\_SEARCH\\_ADDR](#) = 0x020000 , [L4RE\\_RM\\_F\\_IN\\_AREA](#) = 0x040000 , [L4RE\\_RM\\_F\\_EAGER\\_MAP](#)  
= 0x080000 , [L4RE\\_RM\\_F\\_NO\\_EAGER\\_MAP](#) = 0x100000 ,  
[L4RE\\_RM\\_F\\_ATTACH\\_FLAGS](#) = 0x1f0000 }

*Flags for region operations.*



## Functions

- int [l4re\\_rm\\_reserve\\_area](#) ([l4\\_addr\\_t](#) \*start, unsigned long size, [l4re\\_rm\\_flags\\_t](#) flags, unsigned char align) [L4\\_NOTHROW](#)  
*Reserve the given area in the region map.*
- int [l4re\\_rm\\_free\\_area](#) ([l4\\_addr\\_t](#) addr) [L4\\_NOTHROW](#)  
*Free an area from the region map.*
- int [l4re\\_rm\\_attach](#) (void \*\*start, unsigned long size, [l4re\\_rm\\_flags\\_t](#) flags, [l4re\\_ds\\_t](#) mem, [l4re\\_rm\\_offset\\_t](#) offs, unsigned char align) [L4\\_NOTHROW](#)  
*Attach a data space to a region.*
- int [l4re\\_rm\\_detach](#) (void \*addr) [L4\\_NOTHROW](#)  
*Detach and unmap a region from the address space in the current task.*
- int [l4re\\_rm\\_detach\\_ds](#) (void \*addr, [l4re\\_ds\\_t](#) \*ds) [L4\\_NOTHROW](#)  
*Detach and unmap a region and return affected dataspace in the current task.*
- int [l4re\\_rm\\_detach\\_unmap](#) ([l4\\_addr\\_t](#) addr, [l4\\_cap\\_idx\\_t](#) task) [L4\\_NOTHROW](#)  
*Detach and unmap in specified task.*
- int [l4re\\_rm\\_detach\\_ds\\_unmap](#) (void \*addr, [l4re\\_ds\\_t](#) \*ds, [l4\\_cap\\_idx\\_t](#) task) [L4\\_NOTHROW](#)  
*Detach and unmap in specified task.*
- int [l4re\\_rm\\_find](#) ([l4\\_addr\\_t](#) \*addr, unsigned long \*size, [l4re\\_rm\\_offset\\_t](#) \*offset, [l4re\\_rm\\_flags\\_t](#) \*flags, [l4re\\_ds\\_t](#) \*m) [L4\\_NOTHROW](#)  
*Find a region given an address and size.*
- int [l4re\\_rm\\_get\\_info](#) ([l4\\_addr\\_t](#) addr, char \*name, unsigned int len, [l4re\\_rm\\_offset\\_t](#) \*backing\_offset) [L4\\_NOTHROW](#)  
*Return auxiliary information of a region.*
- void [l4re\\_rm\\_show\\_lists](#) (void) [L4\\_NOTHROW](#)  
*Dump region map internal data structures.*
- int [l4re\\_rm\\_reserve\\_area\\_srv](#) ([l4\\_cap\\_idx\\_t](#) rm, [l4\\_addr\\_t](#) \*start, unsigned long size, [l4re\\_rm\\_flags\\_t](#) flags, unsigned char align) [L4\\_NOTHROW](#)
- int [l4re\\_rm\\_free\\_area\\_srv](#) ([l4\\_cap\\_idx\\_t](#) rm, [l4\\_addr\\_t](#) addr) [L4\\_NOTHROW](#)
- int [l4re\\_rm\\_attach\\_srv](#) ([l4\\_cap\\_idx\\_t](#) rm, void \*\*start, unsigned long size, [l4re\\_rm\\_flags\\_t](#) flags, [l4re\\_ds\\_t](#) mem, [l4re\\_rm\\_offset\\_t](#) offs, unsigned char align) [L4\\_NOTHROW](#)
- int [l4re\\_rm\\_detach\\_srv](#) ([l4\\_cap\\_idx\\_t](#) rm, [l4\\_addr\\_t](#) addr, [l4re\\_ds\\_t](#) \*ds, [l4\\_cap\\_idx\\_t](#) task) [L4\\_NOTHROW](#)
- int [l4re\\_rm\\_find\\_srv](#) ([l4\\_cap\\_idx\\_t](#) rm, [l4\\_addr\\_t](#) \*addr, unsigned long \*size, [l4re\\_rm\\_offset\\_t](#) \*offset, [l4re\\_rm\\_flags\\_t](#) \*flags, [l4re\\_ds\\_t](#) \*m) [L4\\_NOTHROW](#)
- int [l4re\\_rm\\_get\\_info\\_srv](#) ([l4\\_cap\\_idx\\_t](#) rm, [l4\\_addr\\_t](#) addr, char \*name, unsigned int len, [l4re\\_rm\\_offset\\_t](#) \*backing\_offset) [L4\\_NOTHROW](#)
- void [l4re\\_rm\\_show\\_lists\\_srv](#) ([l4\\_cap\\_idx\\_t](#) rm) [L4\\_NOTHROW](#)  
*Dump region map internal data structures.*

### 14.9.11.1 Detailed Description

Region map C interface.

### 14.9.11.2 Enumeration Type Documentation

#### 14.9.11.2.1 l4re\_rm\_flags\_values

```
enum l4re\_rm\_flags\_values
```

Flags for region operations.

#### Enumerator

L4RE_RM_F_R	Region is read-only.
L4RE_RM_F_KERNEL	Kernel-provided memory (KUMEM).
L4RE_RM_F_DETACH_FREE	Free the portion of the data space after detach.
L4RE_RM_F_PAGER	Region has a pager.
L4RE_RM_F_RESERVED	Region is reserved (blocked).
L4RE_RM_CACHING_SHIFT	Start of region mapper cache bits.
L4RE_RM_F_CACHING	Mask of all region manager cache bits.
L4RE_RM_REGION_FLAGS	Mask of all region flags.
L4RE_RM_F_CACHE_NORMAL	Cache bits for normal cacheable memory.
L4RE_RM_F_CACHE_BUFFERED	Cache bits for buffered (write combining) memory.
L4RE_RM_F_CACHE_UNCACHED	Cache bits for uncached memory.
L4RE_RM_F_SEARCH_ADDR	Search for a suitable address range.
L4RE_RM_F_IN_AREA	Search only in area, or map into area.
L4RE_RM_F_EAGER_MAP	Eagerly map the attached data space in.
L4RE_RM_F_NO_EAGER_MAP	Prevent eager mapping of the attached data space.
L4RE_RM_F_ATTACH_FLAGS	Mask of all attach flags.

Definition at line 30 of file [rm.h](#).

### 14.9.11.3 Function Documentation

#### 14.9.11.3.1 l4re\_rm\_attach()

```
int l4re_rm_attach (
    void ** start,
    unsigned long size,
    l4re_rm_flags_t flags,
    l4re_ds_t mem,
    l4re_rm_offset_t offs,
    unsigned char align) [inline]
```

Attach a data space to a region.

#### Parameters

<code>in, out</code>	<code>start</code>	Virtual start address where the region manager shall attach the data space. Will be rounded down to the nearest start of a page. If <a href="#">L4Re::Rm::F::Search_addr</a> is given this value is used as the start address to search for a free virtual memory region and the resulting address is returned here. If <a href="#">L4Re::Rm::F::In_area</a> is given the value is used as a selector for the area (see <a href="#">L4Re::Rm::reserve_area</a> ) to attach the data space to.
	<code>size</code>	Size of the data space to attach (in bytes). Will be rounded up to the nearest multiple of the page size.

	<i>flags</i>	The flags control how and with which rights the dataspace is attached to the region. See <a href="#">L4Re::Rm::F::Attach_flags</a> and <a href="#">L4Re::Rm::F::Region_flags</a> . The caller must specify the desired rights of the attached region explicitly. The default set of rights is empty. If the <code>F : ↔ Eager_map</code> flag is set this function may also return <a href="#">L4Re::Dataspace::map</a> error codes if the mapping fails.
	<i>mem</i>	Data space.
	<i>offs</i>	Offset into the data space to use.
	<i>align</i>	Alignment of the virtual region, log2-size, default: a page ( <a href="#">L4_PAGESHIFT</a> ). This is only meaningful if the <a href="#">L4Re::Rm::F::Search_addr</a> flag is used.

### Return values

0	Success
-L4_ENOENT	No area could be found (see <a href="#">L4Re::Rm::F::In_area</a> )
-L4_EPERM	Operation not allowed.
-L4_EINVAL	
-L4_EADDRNOTAVAIL	The given address is not available.
<0	IPC errors

Makes the whole or parts of a data space visible in the virtual memory of the corresponding task. The corresponding region in the virtual address space is backed with the contents of the dataspace.

### Note

When searching for a free place in the virtual address space, the space between *start* and the end of the virtual address space is

### See also

[L4Re::Rm::attach](#)

This function is using the `L4::Env::env()->rm()` service.

### Examples

[examples/libs/l4re/c/ma+rm.c](#).

Definition at line 342 of file [rm.h](#).

References [L4\\_NOTHROW](#), [l4re\\_ds\\_t](#), and [l4re\\_rm\\_attach\\_srv\(\)](#).

Here is the call graph for this function:



### 14.9.11.3.2 l4re\_rm\_attach\_srv()

```
int l4re_rm_attach_srv (
    l4_cap_idx_t rm,
    void ** start,
    unsigned long size,
    l4re_rm_flags_t flags,
    l4re_ds_t mem,
    l4re_rm_offset_t offs,
    unsigned char align)
```

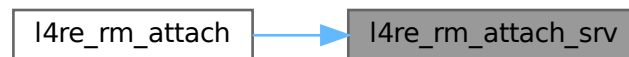
See also

[L4Re::Rm::attach](#)

References [L4\\_CV](#), [L4\\_NOTHROW](#), and [l4re\\_ds\\_t](#).

Referenced by [l4re\\_rm\\_attach\(\)](#).

Here is the caller graph for this function:



### 14.9.11.3.3 l4re\_rm\_detach()

```
int l4re_rm_detach (
    void * addr) [inline]
```

Detach and unmap a region from the address space in the current task.

#### Parameters

<i>addr</i>	Address of the region to detach.
-------------	----------------------------------

#### Return values

<a href="#">L4Re::Rm::Detach_result</a>	On success.
<code>-L4_ENOENT</code>	No region found.
<code>&lt;0</code>	IPC errors

Frees a region in the virtual address space given by *addr*. The corresponding part of the address space is now available again.

Also

See also

[L4Re::Rm::detach](#)

This function is using the `L4::Env::env()->rm()` service.

Definition at line 352 of file [rm.h](#).

References [L4\\_BASE\\_TASK\\_CAP](#), [L4\\_NOTHROW](#), and [l4re\\_rm\\_detach\\_srv\(\)](#).

Here is the call graph for this function:



#### 14.9.11.3.4 l4re\_rm\_detach\_ds()

```
int l4re_rm_detach_ds (
    void * addr,
    l4re_ds_t * ds) [inline]
```

Detach and unmap a region and return affected dataspace in the current task.

##### Parameters

	<i>addr</i>	Address of the region to detach.
out	<i>ds</i>	Returns dataspace that is affected.

##### Return values

<a href="#">L4Re::Rm::Detach_result</a>	On success.
<code>-L4_ENOENT</code>	No region found.
<code>&lt;0</code>	IPC errors

Frees a region in the virtual address space given by *addr*. The corresponding part of the address space is now available again.

Also

See also

[L4Re::Rm::detach](#)

This function is using the `L4::Env::env()->rm()` service.

Examples

[examples/libs/l4re/c/ma+rm.c](#).

Definition at line 365 of file [rm.h](#).

References [L4\\_BASE\\_TASK\\_CAP](#), [L4\\_NOTHROW](#), [l4re\\_ds\\_t](#), and [l4re\\_rm\\_detach\\_srv\(\)](#).

Here is the call graph for this function:



#### 14.9.11.3.5 l4re\_rm\_detach\_ds\_unmap()

```
int l4re_rm_detach_ds_unmap (
    void * addr,
    l4re_ds_t * ds,
    l4_cap_idx_t task) [inline]
```

Detach and unmap in specified task.

##### Parameters

	<i>addr</i>	Address of the region to detach.
out	<i>ds</i>	Returns dataspace that is affected.
	<i>task</i>	Task to unmap pages from, specify <code>L4_INVALID_CAP</code> to not unmap

##### Returns

0 on success, <0 on error

Also

See also

[L4Re::Rm::detach](#)

This function is using the `L4::Env::env()->rm()` service.

Definition at line 372 of file [rm.h](#).

References [L4\\_NOTHROW](#), [l4re\\_ds\\_t](#), and [l4re\\_rm\\_detach\\_srv\(\)](#).

Here is the call graph for this function:



#### 14.9.11.3.6 l4re\_rm\_detach\_srv()

```

int l4re_rm_detach_srv (
    l4_cap_idx_t rm,
    l4_addr_t addr,
    l4re_ds_t * ds,
    l4_cap_idx_t task)

```

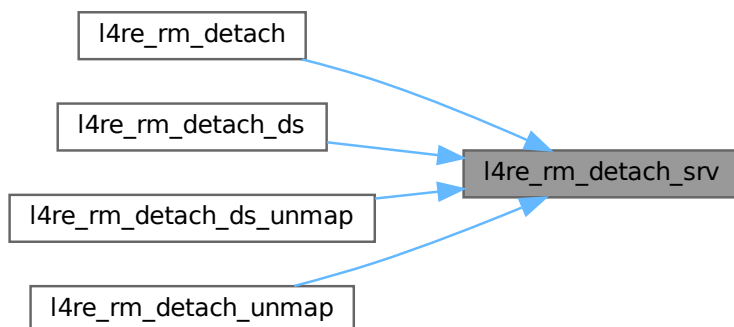
See also

[L4Re::Rm::detach](#)

References [L4\\_CV](#), [L4\\_NOTHROW](#), and [l4re\\_ds\\_t](#).

Referenced by [l4re\\_rm\\_detach\(\)](#), [l4re\\_rm\\_detach\\_ds\(\)](#), [l4re\\_rm\\_detach\\_ds\\_unmap\(\)](#), and [l4re\\_rm\\_detach\\_unmap\(\)](#).

Here is the caller graph for this function:



#### 14.9.11.3.7 l4re\_rm\_detach\_unmap()

```
int l4re_rm_detach_unmap (
    l4_addr_t addr,
    l4_cap_idx_t task) [inline]
```

Detach and unmap in specified task.

##### Parameters

<i>addr</i>	Address of the region to detach.
<i>task</i>	Task to unmap pages from, specify L4_INVALID_CAP to not unmap

##### Returns

0 on success, <0 on error

Also

See also

[L4Re::Rm::detach](#)

This function is using the `L4::Env::env()->rm()` service.

Definition at line 359 of file [rm.h](#).

References [L4\\_NOTHROW](#), and [l4re\\_rm\\_detach\\_srv\(\)](#).

Here is the call graph for this function:



#### 14.9.11.3.8 l4re\_rm\_find()

```
int l4re_rm_find (
    l4_addr_t * addr,
    unsigned long * size,
    l4re_rm_offset_t * offset,
    l4re_rm_flags_t * flags,
    l4re_ds_t * m) [inline]
```

Find a region given an address and size.

##### Parameters

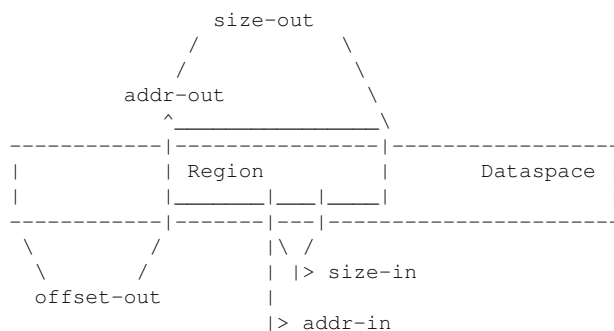


<i>in, out</i>	<i>addr</i>	Address to look for. Returns the start address of the found region.
<i>in, out</i>	<i>size</i>	Size of the area to look for (in bytes). Returns the size of the found region (in bytes).
<i>out</i>	<i>offset</i>	Offset at the beginning of the region within the associated dataspace.
<i>out</i>	<i>flags</i>	Region flags, see <code>F::Region_flags</code> (and <code>F::In_area</code> ).
<i>out</i>	<i>m</i>	Associated dataspace or paging service.

### Return values

<i>0</i>	Success
<i>-L4_EPERM</i>	Operation not allowed.
<i>-L4_ENOENT</i>	No region found.
<i>&lt;0</i>	IPC errors

This function returns the properties of the region that contains the area described by the `addr` and `size` parameter. If no such region is found but a reserved area, the area is returned and `F::In_area` is set in `flags`. Note, in the case of an area the `offset` and `m` return values are invalid.



### Note

The value of the `size` input parameter should be 1 to assure that a region can be determined unambiguously.

### See also

[L4Re::Rm::find](#)

Definition at line 379 of file [rm.h](#).

References [L4\\_NOTHROW](#), [l4re\\_ds\\_t](#), and [l4re\\_rm\\_find\\_srv\(\)](#).

Here is the call graph for this function:



#### 14.9.11.3.9 l4re\_rm\_find\_srv()

```
int l4re_rm_find_srv (
    l4_cap_idx_t rm,
    l4_addr_t * addr,
    unsigned long * size,
    l4re_rm_offset_t * offset,
    l4re_rm_flags_t * flags,
    l4re_ds_t * m)
```

See also

[L4Re::Rm::find](#)

References [L4\\_CV](#), [L4\\_NOTHROW](#), and [l4re\\_ds\\_t](#).

Referenced by [l4re\\_rm\\_find\(\)](#).

Here is the caller graph for this function:



#### 14.9.11.3.10 l4re\_rm\_free\_area()

```
int l4re_rm_free_area (
    l4_addr_t addr) [inline]
```

Free an area from the region map.

##### Parameters

<i>addr</i>	An address within the area to free.
-------------	-------------------------------------

##### Return values

0	Success
-L4_ENOENT	No area found.
<0	IPC errors

**Note**

The data spaces that are attached to that area are not detached by this operation.

**See also**

`reserve_area()` for more information about areas.

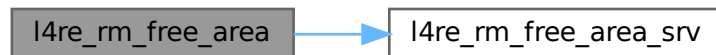
[L4Re::Rm::free\\_area](#)

This function is using the `L4::Env::env()->rm()` service.

Definition at line 336 of file `rm.h`.

References [L4\\_NOTHROW](#), and [l4re\\_rm\\_free\\_area\\_srv\(\)](#).

Here is the call graph for this function:

**14.9.11.3.11 l4re\_rm\_free\_area\_srv()**

```
int l4re_rm_free_area_srv (  
    l4_cap_idx_t rm,  
    l4_addr_t addr)
```

**See also**

[L4Re::Rm::free\\_area](#)

References [L4\\_CV](#), and [L4\\_NOTHROW](#).

Referenced by [l4re\\_rm\\_free\\_area\(\)](#).

Here is the caller graph for this function:



### 14.9.11.3.12 l4re\_rm\_get\_info()

```
int l4re_rm_get_info (
    l4_addr_t addr,
    char * name,
    unsigned int len,
    l4re_rm_offset_t * backing_offset) [inline]
```

Return auxiliary information of a region.

This is a debugging feature and might not be available.

#### Parameters

	<i>addr</i>	Virtual address of the region.
out	<i>name</i>	Name of the region.
out	<i>backing_offset</i>	Backing offset information.

#### Return values

0	Success
-L4_ENOENT	Region not found.
-L4_ENOSYS	Function not available.
<0	IPC errors

#### Parameters

<i>len</i>	Length of the name given in name argument, in bytes.
------------	------------------------------------------------------

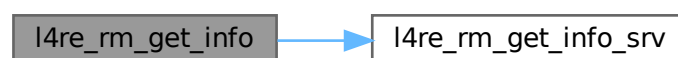
#### See also

[L4Re::Rm::get\\_info](#)

Definition at line 395 of file [rm.h](#).

References [L4\\_NOTHROW](#), and [l4re\\_rm\\_get\\_info\\_srv\(\)](#).

Here is the call graph for this function:



**14.9.11.3.13 l4re\_rm\_get\_info\_srv()**

```
int l4re_rm_get_info_srv (
    l4_cap_idx_t rm,
    l4_addr_t addr,
    char * name,
    unsigned int len,
    l4re_rm_offset_t * backing_offset)
```

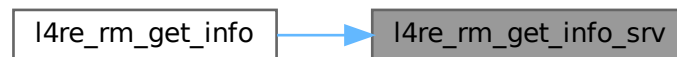
See also

[L4Re::Rm::get\\_info](#)

References [L4\\_CV](#), and [L4\\_NOTHROW](#).

Referenced by [l4re\\_rm\\_get\\_info\(\)](#).

Here is the caller graph for this function:

**14.9.11.3.14 l4re\_rm\_reserve\_area()**

```
int l4re_rm_reserve_area (
    l4_addr_t * start,
    unsigned long size,
    l4re_rm_flags_t flags,
    unsigned char align) [inline]
```

Reserve the given area in the region map.

**Parameters**

in, out	<i>start</i>	The virtual start address of the area to reserve. Returns the start address of the area.
	<i>size</i>	The size of the area to reserve (in bytes).
	<i>flags</i>	Flags for the reserved area (see <a href="#">L4Re::Rm::F::Region_flags</a> and <a href="#">L4Re::Rm::F::Attach_flags</a> ).
	<i>align</i>	Alignment of area if searched as bits (log2 value).

**Return values**

0	Success
-L4_EADDRNOTAVAIL	The given area cannot be reserved.
<0	IPC errors

This function reserves an area within the virtual address space managed by the region map. There are two kinds of areas available:

- Reserved areas (*flags* = [L4Re::Rm::F::Reserved](#)), where no data spaces can be attached
- Special purpose areas (*flags* = 0), where data spaces can be attached to the area via the [L4Re::Rm::F::In\\_area](#) flag and a start address within the area itself.

#### Note

When searching for a free place in the virtual address space (with *flags* = [L4Re::Rm::F::Search\\_addr](#)), the space between *start* and the end of the virtual address space is searched.

#### See also

[L4Re::Rm::reserve\\_area](#)

This function is using the `L4::Env::env()->rm()` service.

Definition at line 328 of file [rm.h](#).

References [L4\\_NOTHROW](#), and [l4re\\_rm\\_reserve\\_area\\_srv\(\)](#).

Here is the call graph for this function:



#### 14.9.11.3.15 l4re\_rm\_reserve\_area\_srv()

```

int l4re_rm_reserve_area_srv (
    l4_cap_idx_t rm,
    l4_addr_t * start,
    unsigned long size,
    l4re_rm_flags_t flags,
    unsigned char align)
  
```

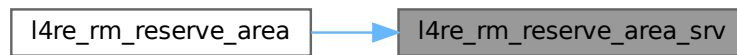
See also

[L4Re::Rm::reserve\\_area](#)

References [L4\\_CV](#), and [L4\\_NOTHROW](#).

Referenced by [l4re\\_rm\\_reserve\\_area\(\)](#).

Here is the caller graph for this function:



#### 14.9.11.3.16 l4re\_rm\_show\_lists()

```
void l4re_rm_show_lists (
    void ) [inline]
```

Dump region map internal data structures.

This function is using the `L4::Env::env()->rm()` service.

Definition at line 387 of file [rm.h](#).

References [L4\\_NOTHROW](#), and [l4re\\_rm\\_show\\_lists\\_srv\(\)](#).

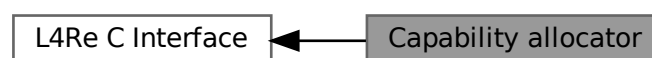
Here is the call graph for this function:



### 14.9.12 Capability allocator

Capability allocator C interface.

Collaboration diagram for Capability allocator:



## Functions

- [L4\\_BEGIN\\_DECLS](#) [l4\\_cap\\_idx\\_t](#) **l4re\_util\_cap\_alloc** (void) [L4\\_NOTHROW](#)  
*Get free capability index at capability allocator.*
- void **l4re\_util\_cap\_free** ([l4\\_cap\\_idx\\_t](#) cap) [L4\\_NOTHROW](#)  
*Return capability index to capability allocator.*
- void **l4re\_util\_cap\_free\_um** ([l4\\_cap\\_idx\\_t](#) cap) [L4\\_NOTHROW](#)  
*Return capability index to capability allocator, and unmaps the object.*
- long **l4re\_util\_cap\_last** (void) [L4\\_NOTHROW](#)  
*Return last capability index the allocator can return.*

### 14.9.12.1 Detailed Description

Capability allocator C interface.

### 14.9.12.2 Function Documentation

#### 14.9.12.2.1 l4re\_util\_cap\_last()

```
long l4re_util_cap_last (
    void )
```

Return last capability index the allocator can return.

#### Returns

last/biggest capability index the allocator can return

References [L4\\_END\\_DECLS](#), and [L4\\_NOTHROW](#).

### 14.9.13 Kumem allocator utility

Kumem allocator utility C interface.

Collaboration diagram for Kumem allocator utility:



Kumem allocator utility C interface.



### 14.9.14 Video API

Collaboration diagram for Video API:



#### Data Structures

- struct [l4re\\_video\\_color\\_component\\_t](#)  
*Color component structure.*
- struct [l4re\\_video\\_pixel\\_info\\_t](#)  
*Pixel\_info structure.*
- struct [l4re\\_video\\_goos\\_info\\_t](#)  
*Goos information structure.*
- struct [l4re\\_video\\_view\\_info\\_t](#)  
*View information structure.*
- struct [l4re\\_video\\_view\\_t](#)  
*C representation of a goos view.*

#### Typedefs

- typedef struct [l4re\\_video\\_color\\_component\\_t](#) [l4re\\_video\\_color\\_component\\_t](#)  
*Color component structure.*
- typedef struct [l4re\\_video\\_pixel\\_info\\_t](#) [l4re\\_video\\_pixel\\_info\\_t](#)  
*Pixel\_info structure.*
- typedef [l4\\_cap\\_idx\\_t](#) [l4re\\_video\\_goos\\_t](#)  
*Goos object type.*
- typedef struct [l4re\\_video\\_view\\_info\\_t](#) [l4re\\_video\\_view\\_info\\_t](#)  
*View information structure.*
- typedef struct [l4re\\_video\\_view\\_t](#) [l4re\\_video\\_view\\_t](#)  
*C representation of a goos view.*

#### Enumerations

- enum [l4re\\_video\\_goos\\_info\\_flags\\_t](#) { [F\\_l4re\\_video\\_goos\\_auto\\_refresh](#) = 0x01 , [F\\_l4re\\_video\\_goos\\_pointer](#) = 0x02 , [F\\_l4re\\_video\\_goos\\_dynamic\\_views](#) = 0x04 , [F\\_l4re\\_video\\_goos\\_dynamic\\_buffers](#) = 0x08 }  
*Flags of information on the goos.*
- enum [l4re\\_video\\_view\\_info\\_flags\\_t](#) {  
[F\\_l4re\\_video\\_view\\_none](#) = 0x00 , [F\\_l4re\\_video\\_view\\_set\\_buffer](#) = 0x01 , [F\\_l4re\\_video\\_view\\_set\\_buffer\\_offset](#) = 0x02 , [F\\_l4re\\_video\\_view\\_set\\_bytes\\_per\\_line](#) = 0x04 ,  
[F\\_l4re\\_video\\_view\\_set\\_pixel](#) = 0x08 , [F\\_l4re\\_video\\_view\\_set\\_position](#) = 0x10 , [F\\_l4re\\_video\\_view\\_dyn\\_allocated](#) = 0x20 , [F\\_l4re\\_video\\_view\\_set\\_background](#) = 0x40 ,  
[F\\_l4re\\_video\\_view\\_set\\_flags](#) = 0x80 , [F\\_l4re\\_video\\_view\\_fully\\_dynamic](#) , [F\\_l4re\\_video\\_view\\_above](#) = 0x01000 , [F\\_l4re\\_video\\_view\\_flags\\_mask](#) = 0xff000 }  
*Flags of information on a view.*

## Functions

- [L4\\_BEGIN\\_DECLS](#) `int l4re_video_goos_info (l4re_video_goos_t goos, l4re_video_goos_info_t *ginfo) L4_NOTHROW`  
*Get information on a goos.*
- `int l4re_video_goos_refresh (l4re_video_goos_t goos, int x, int y, int w, int h) L4_NOTHROW`  
*Flush a rectangle of pixels of the goos screen.*
- `int l4re_video_goos_create_buffer (l4re_video_goos_t goos, unsigned long size, l4_cap_idx_t buffer) L4_NOTHROW`  
*Create a new buffer (memory buffer) for pixel data.*
- `int l4re_video_goos_delete_buffer (l4re_video_goos_t goos, unsigned idx) L4_NOTHROW`  
*Delete a pixel buffer.*
- `int l4re_video_goos_get_static_buffer (l4re_video_goos_t goos, unsigned idx, l4_cap_idx_t buffer) L4_NOTHROW`  
*Get the data-space capability of the static pixel buffer.*
- `int l4re_video_goos_create_view (l4re_video_goos_t goos, l4re_video_view_t *view) L4_NOTHROW`  
*Create a new view (.*
- `int l4re_video_goos_delete_view (l4re_video_goos_t goos, l4re_video_view_t *view) L4_NOTHROW`  
*Delete a view.*
- `int l4re_video_goos_get_view (l4re_video_goos_t goos, unsigned idx, l4re_video_view_t *view) L4_NOTHROW`  
*Get a view for the given index.*
- [L4\\_BEGIN\\_DECLS](#) `int l4re_video_view_refresh (l4re_video_view_t *view, int x, int y, int w, int h) L4_NOTHROW`  
*Flush the given rectangle of pixels of the given view.*
- `int l4re_video_view_get_info (l4re_video_view_t *view, l4re_video_view_info_t *info) L4_NOTHROW`  
*Retrieve information about the given view.*
- `int l4re_video_view_set_info (l4re_video_view_t *view, l4re_video_view_info_t *info) L4_NOTHROW`  
*Set properties of the view.*
- `int l4re_video_view_set_viewport (l4re_video_view_t *view, int x, int y, int w, int h, unsigned long bofs) L4_NOTHROW`  
*Set the viewport parameters of a view.*
- `int l4re_video_view_stack (l4re_video_view_t *view, l4re_video_view_t *pivot, int behind) L4_NOTHROW`  
*Change the stacking order in the stack of visible views.*

### 14.9.14.1 Detailed Description

### 14.9.14.2 Typedef Documentation

#### 14.9.14.2.1 l4re\_video\_view\_t

```
typedef struct l4re_video_view_t l4re_video_view_t
```

C representation of a goos view.

A view is a visible rectangle that provides a view to the contents of a buffer (frame buffer) memory object and is placed on a real screen.

### 14.9.14.3 Enumeration Type Documentation

#### 14.9.14.3.1 l4re\_video\_goos\_info\_flags\_t

```
enum l4re_video_goos_info_flags_t
```

Flags of information on the goos.

#### Enumerator

F_l4re_video_goos_auto_refresh	The graphics display is automatically refreshed.
F_l4re_video_goos_pointer	We have a mouse pointer.
F_l4re_video_goos_dynamic_views	Supports dynamically allocated views.
F_l4re_video_goos_dynamic_buffers	Supports dynamically allocated buffers.

Definition at line 29 of file [goos.h](#).

#### 14.9.14.3.2 l4re\_video\_view\_info\_flags\_t

enum [l4re\\_video\\_view\\_info\\_flags\\_t](#)

Flags of information on a view.

##### Enumerator

F_l4re_video_view_none	everything for this view is static (the VESA-FB case)
F_l4re_video_view_set_buffer	buffer object for this view can be changed
F_l4re_video_view_set_buffer_offset	buffer offset can be set
F_l4re_video_view_set_bytes_per_line	bytes per line can be set
F_l4re_video_view_set_pixel	pixel type can be set
F_l4re_video_view_set_position	position on screen can be set
F_l4re_video_view_dyn_allocated	View is dynamically allocated.
F_l4re_video_view_set_background	Set view as background for session.
F_l4re_video_view_set_flags	Set view property flags.
F_l4re_video_view_above	Flag the view as stay on top.
F_l4re_video_view_flags_mask	Mask containing all possible property flags.

Definition at line 23 of file [view.h](#).

### 14.9.14.4 Function Documentation

#### 14.9.14.4.1 l4re\_video\_goos\_create\_buffer()

```
int l4re_video_goos_create_buffer (
    l4re\_video\_goos\_t goos,
    unsigned long size,
    l4\_cap\_idx\_t buffer)
```

Create a new buffer (memory buffer) for pixel data.

##### Parameters

<i>goos</i>	the target object for the operation.
<i>size</i>	the size in bytes for the pixel buffer.

<i>buffer</i>	a capability index to receive the data-space capability for the buffer.
---------------	-------------------------------------------------------------------------

#### Returns

>=0: The index of the created buffer (used to assign views and for deletion). < 0: on error

References [L4\\_CV](#), and [L4\\_NOTHROW](#).

#### 14.9.14.4.2 l4re\_video\_goos\_create\_view()

```
int l4re_video_goos_create_view (
    l4re_video_goos_t goos,
    l4re_video_view_t * view)
```

Create a new view (.

#### See also

[l4re\\_video\\_view\\_t](#)

#### Parameters

	<i>goos</i>	the goos session to use.
out	<i>view</i>	structure initialized to the new view.

References [L4\\_CV](#), and [L4\\_NOTHROW](#).

#### 14.9.14.4.3 l4re\_video\_goos\_delete\_buffer()

```
int l4re_video_goos_delete_buffer (
    l4re_video_goos_t goos,
    unsigned idx)
```

Delete a pixel buffer.

#### Parameters

<i>goos</i>	the target goos object.
<i>idx</i>	the buffer index of the buffer to delete (the return value of <a href="#">l4re_video_goos_create_buffer()</a> )

References [L4\\_CV](#), and [L4\\_NOTHROW](#).

#### 14.9.14.4.4 l4re\_video\_goos\_delete\_view()

```
int l4re_video_goos_delete_view (
    l4re_video_goos_t goos,
    l4re_video_view_t * view)
```

Delete a view.

#### Parameters

<i>goos</i>	the goos session to use.
<i>view</i>	the view to delete, the given data-structure is invalid afterwards.

References [L4\\_CV](#), and [L4\\_NOTHROW](#).

#### 14.9.14.4.5 l4re\_video\_goos\_get\_static\_buffer()

```
int l4re_video_goos_get_static_buffer (
    l4re_video_goos_t goos,
    unsigned idx,
    l4_cap_idx_t buffer)
```

Get the data-space capability of the static pixel buffer.

##### Parameters

<i>goos</i>	The target goos object.
<i>idx</i>	Index of the static buffer.
<i>buffer</i>	A capability index to receive the data-space capability.

This function allows access to static, preexisting pixel buffers. Such static buffers exist for static configurations, such as the VESA framebuffer.

References [L4\\_CV](#), and [L4\\_NOTHROW](#).

#### 14.9.14.4.6 l4re\_video\_goos\_get\_view()

```
int l4re_video_goos_get_view (
    l4re_video_goos_t goos,
    unsigned idx,
    l4re_video_view_t * view)
```

Get a view for the given index.

##### Parameters

	<i>goos</i>	the target goos session.
	<i>idx</i>	the index of the view to retrieve.
out	<i>view</i>	structure initialized to the view with the given index.

This function allows to access static views as provided by the VESA framebuffer (the monitor). However, it also allows to access dynamic views created with [l4re\\_video\\_goos\\_create\\_view\(\)](#).

References [L4\\_END\\_DECLS](#), and [L4\\_NOTHROW](#).

#### 14.9.14.4.7 l4re\_video\_goos\_info()

```
L4_BEGIN_DECLS int l4re_video_goos_info (
    l4re_video_goos_t goos,
    l4re_video_goos_info_t * ginfo)
```

Get information on a goos.

##### Parameters

	<i>goos</i>	Goos object
out	<i>ginfo</i>	Pointer to goos information structure.

#### Returns

0 for success, <0 on error

- [-L4\\_ENODEV](#)
- IPC errors

References [L4\\_CV](#), and [L4\\_NOTHROW](#).

#### 14.9.14.4.8 l4re\_video\_goos\_refresh()

```
int l4re_video_goos_refresh (
    l4re_video_goos_t goos,
    int x,
    int y,
    int w,
    int h)
```

Flush a rectangle of pixels of the goos screen.

#### Parameters

<i>goos</i>	the target object of the operation.
<i>x</i>	the x-coordinate of the upper left corner of the rectangle
<i>y</i>	the y-coordinate of the upper left corner of the rectangle
<i>w</i>	the width of the rectangle to be flushed
<i>h</i>	the height of the rectangle

References [L4\\_CV](#), and [L4\\_NOTHROW](#).

#### 14.9.14.4.9 l4re\_video\_view\_get\_info()

```
int l4re_video_view_get_info (
    l4re_video_view_t * view,
    l4re_video_view_info_t * info)
```

Retrieve information about the given *view*.

#### Parameters

	<i>view</i>	the target view for the operation.
out	<i>info</i>	a buffer receiving the information about the view.

References [L4\\_CV](#), and [L4\\_NOTHROW](#).

**14.9.14.4.10 l4re\_video\_view\_refresh()**

```
L4_BEGIN_DECLS int l4re_video_view_refresh (
    l4re_video_view_t * view,
    int x,
    int y,
    int w,
    int h)
```

Flush the given rectangle of pixels of the given *view*.

**Parameters**

<i>view</i>	the target view of the operation.
<i>x</i>	x-coordinate of the upper left corner
<i>y</i>	y-coordinate of the upper left corner
<i>w</i>	the width of the rectangle
<i>h</i>	the height of the rectangle

References [L4\\_CV](#), and [L4\\_NOTHROW](#).

**14.9.14.4.11 l4re\_video\_view\_set\_info()**

```
int l4re_video_view_set_info (
    l4re_video_view_t * view,
    l4re_video_view_info_t * info)
```

Set properties of the view.

**Parameters**

<i>view</i>	the target view of the operation.
<i>info</i>	the parameters to be set on the view.

Which parameters can be manipulated on a given view can be figured out with [l4re\\_video\\_view\\_get\\_info\(\)](#) and this depends on the concrete instance the view object.

References [L4\\_CV](#), and [L4\\_NOTHROW](#).

**14.9.14.4.12 l4re\_video\_view\_set\_viewport()**

```
int l4re_video_view_set_viewport (
    l4re_video_view_t * view,
    int x,
    int y,
    int w,
    int h,
    unsigned long bofs)
```

Set the viewport parameters of a view.

**Parameters**

<i>view</i>	the target view of the operation.
<i>x</i>	the x-coordinate of the upper left corner on the screen.
<i>y</i>	the y-coordinate of the upper left corner on the screen.
<i>w</i>	the width of the view.
<i>h</i>	the height of the view.
<i>bofs</i>	the offset (in bytes) of the upper left pixel in the memory buffer

This function is a convenience wrapper for [l4re\\_video\\_view\\_set\\_info\(\)](#), just setting the often changed parameters of a dynamic view. With this function a view can be placed on the real screen and at the same time on its backing buffer.

References [L4\\_CV](#), and [L4\\_NOTHROW](#).

#### 14.9.14.4.13 l4re\_video\_view\_stack()

```
int l4re_video_view_stack (
    l4re_video_view_t * view,
    l4re_video_view_t * pivot,
    int behind)
```

Change the stacking order in the stack of visible views.

##### Parameters

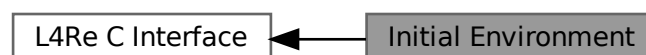
<i>view</i>	the target view for the operation.
<i>pivot</i>	the neighbor view, relative to which <i>view</i> shall be stacked. a NULL value allows top ( <i>behind</i> = 1) and bottom ( <i>behind</i> = 0) placement of the view.
<i>behind</i>	describes the placement of the view relative to the <i>pivot</i> view.

References [L4\\_END\\_DECLS](#), and [L4\\_NOTHROW](#).

#### 14.9.15 Initial Environment

C interface of the initial environment that is provided to an [L4](#) task.

Collaboration diagram for Initial Environment:





## Data Structures

- struct [l4re\\_env\\_cap\\_entry\\_t](#)

*Entry in the [L4Re](#) environment array for the named initial objects.*

## Typedefs

- typedef struct [l4re\\_env\\_cap\\_entry\\_t](#) [l4re\\_env\\_cap\\_entry\\_t](#)

*Entry in the [L4Re](#) environment array for the named initial objects.*

## Functions

- [l4re\\_env\\_t](#) \* [l4re\\_env](#) (void) [L4\\_NOTHROW](#)  
*Get [L4Re](#) initial environment.*
- [l4\\_kernel\\_info\\_t](#) const \* [l4re\\_kip](#) (void) [L4\\_NOTHROW](#)  
*Get Kernel Info Page.*
- [l4\\_cap\\_idx\\_t](#) [l4re\\_env\\_get\\_cap](#) (char const \*name) [L4\\_NOTHROW](#)  
*Get the capability selector for the object named name.*
- [l4\\_cap\\_idx\\_t](#) [l4re\\_env\\_get\\_cap\\_e](#) (char const \*name, [l4re\\_env\\_t](#) const \*e) [L4\\_NOTHROW](#)  
*Get the capability selector for the object named name.*
- [l4re\\_env\\_cap\\_entry\\_t](#) const \* [l4re\\_env\\_get\\_cap\\_l](#) (char const \*name, unsigned l, [l4re\\_env\\_t](#) const \*e) [L4\\_NOTHROW](#)  
*Get the full [l4re\\_env\\_cap\\_entry\\_t](#) for the object named name.*

### 14.9.15.1 Detailed Description

C interface of the initial environment that is provided to an [L4](#) task.

#### Include File

```
#include <l4/re/env.h>
```

For an explanation of the default task capabilities see [l4\\_default\\_caps\\_t](#).

For the C++ interface refer to [L4Re::Env](#).

### 14.9.15.2 Function Documentation

#### 14.9.15.2.1 [l4re\\_env\(\)](#)

```
l4re\_env\_t * l4re\_env (  
    void ) [inline]
```

Get [L4Re](#) initial environment.

**Returns**

Pointer to [L4Re](#) initial environment.

**Examples**

[examples/sys/aliens/main.c](#), [examples/sys/isr/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 181 of file [env.h](#).

References [L4\\_NOTHROW](#).

Referenced by [l4re\\_env\\_get\\_cap\(\)](#).

Here is the caller graph for this function:

**14.9.15.2.2 l4re\_env\_get\_cap()**

```
l4_cap_idx_t l4re_env_get_cap (
    char const * name) [inline]
```

Get the capability selector for the object named *name*.

**Parameters**

<i>name</i>	is the name of the object to lookup in the initial objects.
-------------	-------------------------------------------------------------

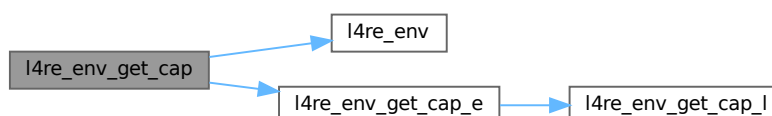
**Returns**

A valid capability selector if the object exists or an invalid capability selector if not ([l4\\_is\\_invalid\\_cap\(\)](#)).

Definition at line 220 of file [env.h](#).

References [L4\\_NOTHROW](#), [l4re\\_env\(\)](#), and [l4re\\_env\\_get\\_cap\\_e\(\)](#).

Here is the call graph for this function:



### 14.9.15.2.3 l4re\_env\_get\_cap\_e()

```
l4_cap_idx_t l4re_env_get_cap_e (
    char const * name,
    l4re_env_t const * e) [inline]
```

Get the capability selector for the object named *name*.

#### Parameters

<i>name</i>	is the name of the object to lookup in the initial objects.
<i>e</i>	is the environment structure to use for the operation.

#### Returns

A valid capability selector if the object exists or an invalid capability selector if not ([l4\\_is\\_invalid\\_cap\(\)](#)).

Definition at line 207 of file [env.h](#).

References [l4re\\_env\\_cap\\_entry\\_t::cap](#), [L4\\_INVALID\\_CAP](#), [L4\\_NOTHROW](#), and [l4re\\_env\\_get\\_cap\\_l\(\)](#).

Referenced by [l4re\\_env\\_get\\_cap\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 14.9.15.2.4 l4re\_env\_get\_cap\_l()

```
l4re_env_cap_entry_t const * l4re_env_get_cap_l (
    char const * name,
    unsigned l,
    l4re_env_t const * e) [inline]
```

Get the full [l4re\\_env\\_cap\\_entry\\_t](#) for the object named *name*.

#### Parameters

<i>name</i>	is the name of the object to lookup in the initial objects.
<i>l</i>	is the length of the name string, thus <i>name</i> might not be zero terminated.
<i>e</i>	is the environment structure to use for the operation.

#### Returns

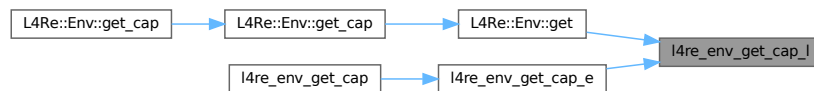
A pointer to an [l4re\\_env\\_cap\\_entry\\_t](#) if the object exists or NULL if not.

Definition at line 189 of file [env.h](#).

References [l4re\\_env\\_cap\\_entry\\_t::flags](#), [L4\\_NOTHROW](#), and [l4re\\_env\\_cap\\_entry\\_t::name](#).

Referenced by [L4Re::Env::get\(\)](#), and [l4re\\_env\\_get\\_cap\\_e\(\)](#).

Here is the caller graph for this function:



#### 14.9.15.2.5 l4re\_kip()

```
l4_kernel_info_t const * l4re_kip (
    void ) [inline]
```

Get Kernel Info Page.

#### Returns

Pointer to Kernel Info Page (KIP) structure.

#### Examples

[examples/libs/shmc/prodcons.c](#), and [examples/sys/aliens/main.c](#).

Definition at line 185 of file [env.h](#).

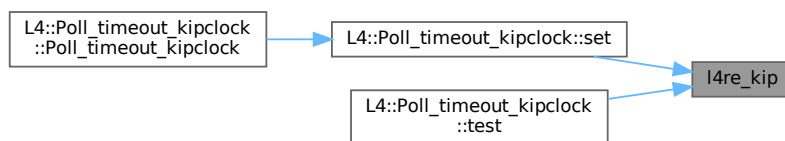
References [l4\\_kip\(\)](#), and [L4\\_NOTHROW](#).

Referenced by [L4::Poll\\_timeout\\_kipclock::set\(\)](#), and [L4::Poll\\_timeout\\_kipclock::test\(\)](#).

Here is the call graph for this function:



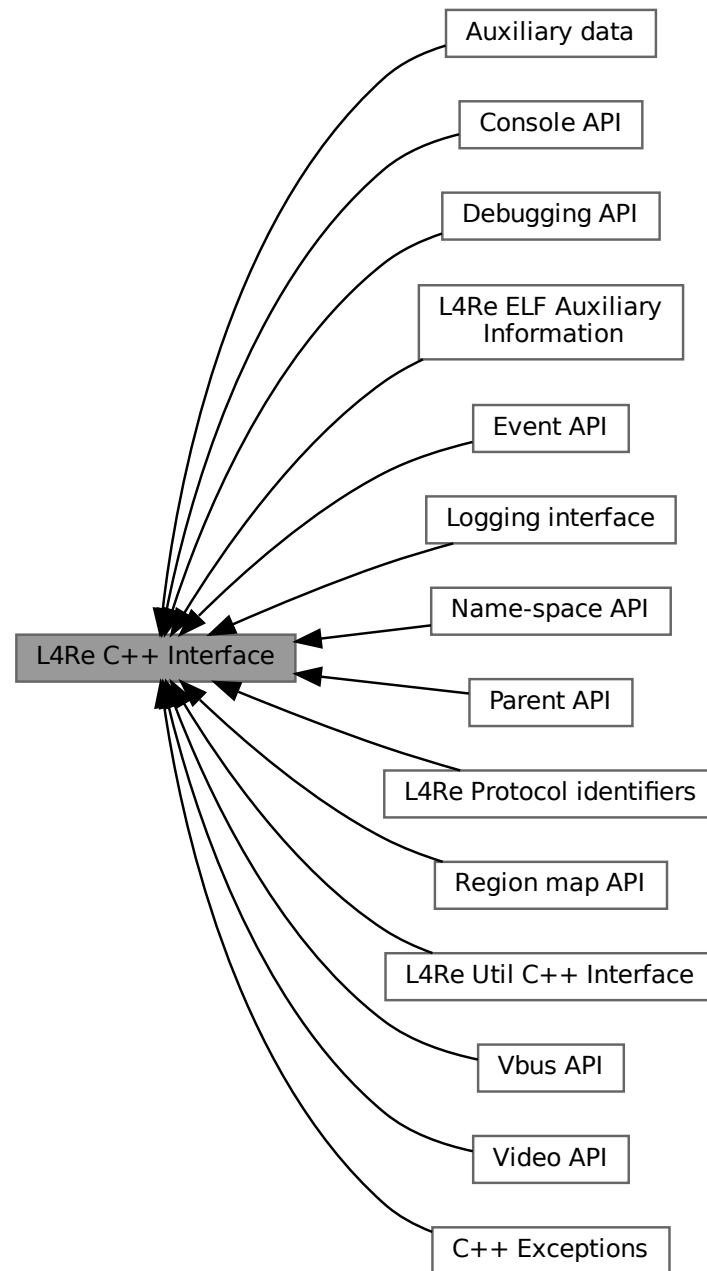
Here is the caller graph for this function:



## 14.10 L4Re C++ Interface

Documentation of the [L4](#) Runtime Environment C++ API.

Collaboration diagram for L4Re C++ Interface:



## Topics

- L4Re Util C++ Interface . . . . . [591](#)  
*Documentation of the [L4](#) Runtime Environment utility functionality in C++.*
- Console API . . . . . [596](#)

•	<i>Console interface.</i>	
•	Debugging API . . . . .	596
	<i>Debugging Interface.</i>	
•	L4Re ELF Auxiliary Information . . . . .	597
	<i>API for embedding auxiliary information into binary programs.</i>	
•	Event API . . . . .	599
	<i>Event API.</i>	
•	Auxiliary data . . . . .	600
•	Logging interface . . . . .	601
	<i>Interface for log output.</i>	
•	Name-space API . . . . .	601
	<i>API for name spaces that store capabilities.</i>	
•	Parent API . . . . .	602
	<i>Parent interface.</i>	
•	L4Re Protocol identifiers . . . . .	603
	<i>Fix L4Re Protocol Constants.</i>	
•	Region map API . . . . .	605
	<i>Virtual address-space management.</i>	
•	Video API . . . . .	606
	<i>API for framebuffer based graphics.</i>	
•	C++ Exceptions . . . . .	607
•	Vbus API . . . . .	608
	<i>C++ interface of the Vbus API.</i>	

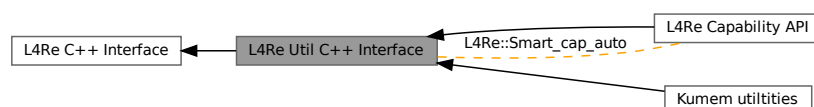
### 14.10.1 Detailed Description

Documentation of the [L4](#) Runtime Environment C++ API.

### 14.10.2 L4Re Util C++ Interface

Documentation of the [L4](#) Runtime Environment utility functionality in C++.

Collaboration diagram for L4Re Util C++ Interface:



## Topics

- L4Re Capability API . . . . . 592
- Kumem•utilities . . . . . 594

## Data Structures

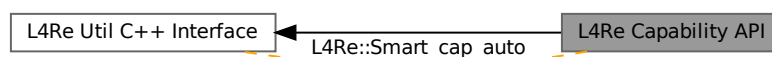
- class [L4Re::Smart\\_cap\\_auto< Unmap\\_flags >](#)  
*Helper for [Unique\\_cap](#) and [Unique\\_del\\_cap](#).*
- class [L4Re::Util::Cap\\_alloc\\_base](#)  
*Capability allocator.*
- class [L4Re::Util::Br\\_manager](#)  
*Buffer-register (BR) manager for [L4::Server](#).*
- class [L4Re::Util::Counting\\_cap\\_alloc< COUNTERTYPE, Dbg >](#)  
*Internal reference-counting cap allocator.*
- class [L4Re::Util::Event\\_buffer\\_t< PAYLOAD >](#)  
*Event\_buffer utility class.*
- class [L4Re::Util::Event\\_buffer\\_consumer\\_t< PAYLOAD >](#)  
*An event buffer consumer.*
- class [L4Re::Util::Vcon\\_svr< SVR >](#)  
*[Console](#) server template class.*
- class [L4Re::Util::Video::Goos\\_svr](#)  
*Goos server class.*

### 14.10.2.1 Detailed Description

Documentation of the [L4](#) Runtime Environment utility functionality in C++.

### 14.10.2.2 L4Re Capability API

Collaboration diagram for L4Re Capability API:





## Data Structures

- class [L4Re::Cap\\_alloc](#)  
*Capability allocator interface.*
- class [L4Re::Smart\\_cap\\_auto< Unmap\\_flags >](#)  
*Helper for [Unique\\_cap](#) and [Unique\\_del\\_cap](#).*
- class [L4Re::Smart\\_count\\_cap< Unmap\\_flags >](#)  
*Helper for [Ref\\_cap](#) and [Ref\\_del\\_cap](#).*
- class [L4Re::Util::Smart\\_cap\\_auto< Unmap\\_flags >](#)  
*Helper for [Unique\\_cap](#) and [Unique\\_del\\_cap](#).*
- class [L4Re::Util::Smart\\_count\\_cap< Unmap\\_flags >](#)  
*Helper for [Ref\\_cap](#) and [Ref\\_del\\_cap](#).*
- struct [L4Re::Util::Ref\\_cap< T >](#)  
*Automatic capability that implements automatic free and unmap of the capability selector.*
- struct [L4Re::Util::Ref\\_del\\_cap< T >](#)  
*Automatic capability that implements automatic free and unmap+delete of the capability selector.*

## Functions

- `template<typename T>  
Ref\_cap< T >::Cap L4Re::Util::make\_ref\_cap ()`  
*Allocate a capability slot and wrap it in a [Ref\\_cap](#).*
- `template<typename T>  
Ref\_del\_cap< T >::Cap L4Re::Util::make\_ref\_del\_cap ()`  
*Allocate a capability slot and wrap it in a [Ref\\_del\\_cap](#).*
- `virtual L4Re::Cap\_alloc::~Cap\_alloc ()=0`  
*Destructor.*

## Variables

- [\\_Cap\\_alloc](#) & [L4Re::Util::cap\\_alloc](#)  
*Capability allocator.*

### 14.10.2.2.1 Detailed Description

### 14.10.2.2.2 Function Documentation

#### 14.10.2.2.2.1 [make\\_ref\\_cap\(\)](#)

```
template<typename T>
Ref\_cap< T >::Cap L4Re::Util::make\_ref\_cap ()
```

Allocate a capability slot and wrap it in a [Ref\\_cap](#).

### Template Parameters

<i>T</i>	Type of capability the slot is used for.
----------	------------------------------------------

Definition at line 195 of file [cap\\_alloc](#).

References [cap\\_alloc](#).

#### 14.10.2.2.2.2 make\_ref\_del\_cap()

```
template<typename T>
Ref_del_cap< T >::Cap L4Re::Util::make_ref_del_cap ()
```

Allocate a capability slot and wrap it in a [Ref\\_del\\_cap](#).

#### Template Parameters

<i>T</i>	Type of capability the slot is used for.
----------	------------------------------------------

Definition at line 204 of file [cap\\_alloc](#).

References [cap\\_alloc](#).

#### 14.10.2.2.3 Variable Documentation

##### 14.10.2.2.3.1 cap\_alloc

```
_Cap_alloc& L4Re::Util::cap_alloc [extern]
```

Capability allocator.

This is the instance of the capability allocator that is used by usual applications.

The capability allocator uses the [Counting\\_cap\\_alloc](#), a reference-counting thread-safe capability allocator, that keeps a reference counter for each managed capability selector.

#### Examples

[examples/libs/l4re/c++/mem\\_alloc/ma+rm.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#), [examples/libs/l4re/c++/shared\\_ds/d](#)  
and [examples/libs/l4re/streammap/client.cc](#).

Referenced by [L4Re::Util::Br\\_manager::alloc\\_buffer\\_demand\(\)](#), [L4Re::Util::Smart\\_count\\_cap< Unmap\\_flags >::copy\(\)](#), [L4Re::Util::Smart\\_cap\\_auto< Unmap\\_flags >::free\(\)](#), [L4Re::Util::Smart\\_count\\_cap< Unmap\\_flags >::free\(\)](#), [L4Re::Util::Event\\_t< Default\\_event\\_payload >::init\(\)](#), [L4Re::Util::Event\\_t< Default\\_event\\_payload >::init\\_poll\(\)](#), [make\\_ref\\_cap\(\)](#), [make\\_ref\\_del\\_cap\(\)](#), [make\\_shared\\_cap\(\)](#), [make\\_shared\\_del\\_cap\(\)](#), [make\\_unique\\_cap\(\)](#), [make\\_unique\\_del\\_cap\(\)](#), [L4Re::Util::Br\\_manager::realloc\\_rcv\\_cap\(\)](#), and [L4Re::Util::Object\\_registry::unregister\\_obj\(\)](#).

#### 14.10.2.3 Kumem utilities

Collaboration diagram for Kumem utilities:



## Functions

- `int L4Re::Util::kumem_alloc (l4_addr_t *mem, unsigned pages_order, L4::Cap< L4::Task > task=L4Re::Env::env() ->task(), L4::Cap< L4Re::Rm > rm=L4Re::Env::env() ->rm()) noexcept`  
*Allocate state area.*

### 14.10.2.3.1 Detailed Description

### 14.10.2.3.2 Function Documentation

#### 14.10.2.3.2.1 kumem\_alloc()

```
int L4Re::Util::kumem_alloc (
    l4_addr_t * mem,
    unsigned pages_order,
    L4::Cap< L4::Task > task = L4Re::Env::env() ->task(),
    L4::Cap< L4Re::Rm > rm = L4Re::Env::env() ->rm())    [noexcept]
```

Allocate state area.

#### Parameters

out	<i>mem</i>	Pointer to memory that has been allocated.
	<i>pages_order</i>	Size to allocate, in log2 pages.
	<i>task</i>	Task to use for allocation.
	<i>rm</i>	Region manager to use for allocation.

#### Return values

0	for success
<0	error code on failure

#### Note

The amount of kernel-user memory that can be allocated at once is limited by the used kernel implementation. The minimum allocatable amount is one page. A portable implementation should not depend on allocations greater than 16KiB to succeed.

References [L4Re::Env::env\(\)](#).

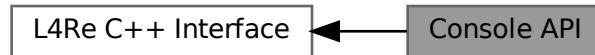
Here is the call graph for this function:



### 14.10.3 Console API

[Console](#) interface.

Collaboration diagram for Console API:



#### Data Structures

- class [L4Re::Console](#)  
*Console class.*

#### 14.10.3.1 Detailed Description

[Console](#) interface.

### 14.10.4 Debugging API

Debugging Interface.

Collaboration diagram for Debugging API:



#### Data Structures

- class [L4Re::Debug\\_obj](#)  
*Debug interface.*

#### 14.10.4.1 Detailed Description

Debugging Interface.

The debugging interface can be provided to retrieve, or log debugging information for an object. Each class may realize the debug interface to provide debugging functionality. For example, the region map objects provide a facility to dump the currently established memory regions.

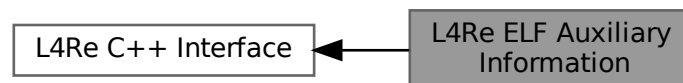
See also

L4::Debug\_obj for more information.

#### 14.10.5 L4Re ELF Auxiliary Information

API for embedding auxiliary information into binary programs.

Collaboration diagram for L4Re ELF Auxiliary Information:



#### Data Structures

- struct [l4re\\_elf\\_aux\\_t](#)  
*Generic header for each auxiliary vector element.*
- struct [l4re\\_elf\\_aux\\_vma\\_t](#)  
*Auxiliary vector element for a reserved virtual memory area.*
- struct [l4re\\_elf\\_aux\\_mword\\_t](#)  
*Auxiliary vector element for a single unsigned data word.*

#### Macros

- #define [L4RE\\_ELF\\_AUX\\_ELEM](#) const \_\_attribute\_\_((used, section(".rol4re\_elf\_aux"), aligned(sizeof(l4\_umword\_t))))  
*Define an auxiliary vector element.*
- #define [L4RE\\_ELF\\_AUX\\_ELEM\\_T](#)(type, id, tag, val...)  
*Define an auxiliary vector element.*

#### Typedefs

- typedef struct l4re\_elf\_aux\_t **l4re\_elf\_aux\_t**  
*Generic header for each auxiliary vector element.*
- typedef struct l4re\_elf\_aux\_vma\_t **l4re\_elf\_aux\_vma\_t**  
*Auxiliary vector element for a reserved virtual memory area.*
- typedef struct l4re\_elf\_aux\_mword\_t **l4re\_elf\_aux\_mword\_t**  
*Auxiliary vector element for a single unsigned data word.*

## Enumerations

- enum {  
[L4RE\\_ELF\\_AUX\\_T\\_NONE](#) = 0 , [L4RE\\_ELF\\_AUX\\_T\\_VMA](#) , [L4RE\\_ELF\\_AUX\\_T\\_STACK\\_SIZE](#) ,  
[L4RE\\_ELF\\_AUX\\_T\\_STACK\\_ADDR](#) ,  
[L4RE\\_ELF\\_AUX\\_T\\_KIP\\_ADDR](#) , [L4RE\\_ELF\\_AUX\\_T\\_EX\\_REGS\\_FLAGS](#) }

### 14.10.5.1 Detailed Description

API for embedding auxiliary information into binary programs.

This API allows information for the binary loader to be embedded into a binary application. This information can be reserved areas in the virtual memory of an application and things such as the stack size to be allocated for the first application thread.

### 14.10.5.2 Macro Definition Documentation

#### 14.10.5.2.1 L4RE\_ELF\_AUX\_ELEM

```
#define L4RE_ELF_AUX_ELEM const __attribute__((used, section(".ro14re_elf_aux"), aligned(sizeof(l4\_umword\_t))))
```

Define an auxiliary vector element.

This is the generic method for defining auxiliary vector elements. A more convenient way is to use [L4RE\\_ELF\\_AUX\\_ELEM\\_T](#).

Usage:

```
L4RE\_ELF\_AUX\_ELEM l4re\_elf\_aux\_vma\_t decl_name =  
{ L4RE\_ELF\_AUX\_T\_VMA, sizeof(l4re\_elf\_aux\_vma\_t), 0x2000, 0x4000 };
```

Definition at line 41 of file [elf\\_aux.h](#).

#### 14.10.5.2.2 L4RE\_ELF\_AUX\_ELEM\_T

```
#define L4RE_ELF_AUX_ELEM_T(  
    type,  
    id,  
    tag,  
    val...)
```

Value:

```
static L4RE\_ELF\_AUX\_ELEM type id = {tag, sizeof(type), val}
```

Define an auxiliary vector element.

## Parameters

<i>type</i>	is the data type for the element (e.g., <a href="#">l4re_elf_aux_vma_t</a> )
<i>id</i>	is the identifier (variable name) for the declaration (the variable is defined with <code>static</code> storage class)
<i>tag</i>	is the tag value for the element e.g., <a href="#">L4RE_ELF_AUX_T_VMA</a>

<i>val</i>	are the values to be set in the descriptor
------------	--------------------------------------------

Usage:

```
L4RE_ELF_AUX_ELEM_T(l4re_elf_aux_vma_t, decl_name, L4RE_ELF_AUX_T_VMA, 0x2000, 0x4000 );
```

Definition at line 56 of file [elf\\_aux.h](#).

### 14.10.5.3 Enumeration Type Documentation

#### 14.10.5.3.1 anonymous enum

anonymous enum

##### Enumerator

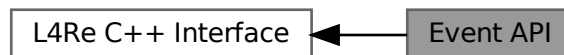
L4RE_ELF_AUX_T_NONE	Tag for an invalid element in the auxiliary vector.
L4RE_ELF_AUX_T_VMA	Tag for descriptor for a reserved virtual memory area.
L4RE_ELF_AUX_T_STACK_SIZE	Tag for descriptor that defines the stack size for the first application thread.
L4RE_ELF_AUX_T_STACK_ADDR	Tag for descriptor that defines the stack address for the first application thread.
L4RE_ELF_AUX_T_KIP_ADDR	Tag for descriptor that defines the KIP address for the binaries address space.
L4RE_ELF_AUX_T_EX_REGS_FLAGS	Tag for descriptor to override <code>ex_regs()</code> flags.

Definition at line 59 of file [elf\\_aux.h](#).

## 14.10.6 Event API

[Event API](#).

Collaboration diagram for Event API:



### Data Structures

- class [L4Re::Event](#)  
*Event class.*
- struct [L4Re::Default\\_event\\_payload](#)  
*Default event stream payload.*
- class [L4Re::Event\\_buffer\\_t< PAYLOAD >](#)  
*Event buffer class.*

### 14.10.6.1 Detailed Description

Event API.

On top of a shared [L4Re::Dataspace](#) (and optionally using [L4::Triggerable](#)), the event API implements asynchronous event transmission from an event provider (server) to an event receiver (client). Events are put into an [Event\\_buffer\\_t](#) residing on the shared [L4Re::Dataspace](#).

This interface is usually not used directly. Instead use [L4Re::Util::Event\\_t](#) for clients. An example server portion is implemented in [L4Re::Util::Event\\_svr](#).

This interface is usually used with [L4Re::Default\\_event\\_payload](#) which delivers HID events modeled on the Linux evdev API, and the interface's methods allow further querying of information about the HID event streams.

### 14.10.7 Auxiliary data

Collaboration diagram for Auxiliary data:



### Data Structures

- struct [l4re\\_aux\\_t](#)  
*Auxiliary descriptor.*

### Typedefs

- typedef struct l4re\_aux\_t [l4re\\_aux\\_t](#)  
*Auxiliary descriptor.*

### Enumerations

- enum [l4re\\_aux\\_ldr\\_flags\\_t](#)  
*Flags for program loading.*

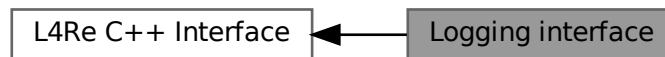


### 14.10.7.1 Detailed Description

## 14.10.8 Logging interface

Interface for log output.

Collaboration diagram for Logging interface:



### Data Structures

- class [L4Re::Log](#)  
*Log interface class.*

### 14.10.8.1 Detailed Description

Interface for log output.

The logging interface provides a facility sending log output. One purpose of the interface is to serialize the output and provide the possibility to tag output sent to a specific log object.

## 14.10.9 Name-space API

API for name spaces that store capabilities.

Collaboration diagram for Name-space API:



### Data Structures

- class [L4Re::Namespace](#)  
*Name-space interface.*

### 14.10.9.1 Detailed Description

API for name spaces that store capabilities.

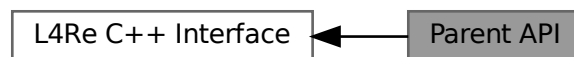
This is a basic abstraction for managing a mapping from human-readable names to capabilities. In particular, a name can also be mapped to a capability that refers to another name space object. By this means name spaces can be constructed hierarchically.

Name spaces play a central role in [L4Re](#), because the implementation of the name space objects determines the policy which capabilities (which objects) are accessible to a client of a name space.

### 14.10.10 Parent API

[Parent](#) interface.

Collaboration diagram for Parent API:



### Data Structures

- class [L4Re::Parent](#)  
*[Parent](#) interface.*

### 14.10.10.1 Detailed Description

[Parent](#) interface.

The parent interface provides means for an [L4](#) task to signal changes in its execution state. The main purpose is to signal program termination to the program that started it, so that its resources can be reclaimed. In a typical [L4Re](#) system, this program will be Moe or Ned.

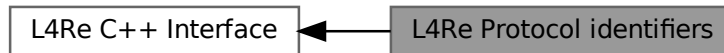
See also

[L4Re::Parent](#) for information about the concrete interface.

### 14.10.11 L4Re Protocol identifiers

Fix [L4Re](#) Protocol Constants.

Collaboration diagram for L4Re Protocol identifiers:



#### Enumerations

- enum [L4Re::Dataspace\\_::Opcodes](#)  
*Data-space communication-protocol opcodes.*
- enum [L4Re::Event\\_::Opcodes](#)  
*Event communication-protocol opcodes.*
- enum [L4Re::Inhibitor\\_::Opcodes](#)  
*Inhibitor communication-protocol opcodes.*
- enum [L4Re::Log\\_::Opcodes](#)  
*Logging-service communication-protocol opcodes.*
- enum [L4Re::Mem\\_alloc\\_::Opcodes](#)  
*Memory-allocator communication-protocol opcodes.*
- enum [L4Re::Namespace\\_::Opcodes](#)  
*Name-space communication-protocol opcodes.*
- enum [L4Re::Parent\\_::Opcodes](#)  
*Parent communication-protocol opcodes.*
- enum [L4re\\_protocols](#) {  
[L4RE\\_PROTO\\_DATASPACE](#) = 0x4000 , [L4RE\\_PROTO\\_NAMESPACE](#) , [L4RE\\_PROTO\\_PARENT](#) ,  
[L4RE\\_PROTO\\_GOOS](#) ,  
[L4RE\\_PROTO\\_RSVD\\_1](#) , [L4RE\\_PROTO\\_RM](#) , [L4RE\\_PROTO\\_EVENT](#) , [L4RE\\_PROTO\\_INHIBITOR](#) ,  
[L4RE\\_PROTO\\_DMA\\_SPACE](#) , [L4RE\\_PROTO\\_MMIO\\_SPACE](#) , [L4RE\\_PROTO\\_ITAS](#) , [L4RE\\_PROTO\\_MEM\\_ALLOC](#)  
 ,  
[L4RE\\_PROTO\\_REMOTE\\_ACCESS](#) , [L4RE\\_PROTO\\_DEBUG](#) = ~0x7fffL }  
*Common [L4Re](#) Protocol Constants.*
- enum [L4Re::Rm\\_::Opcodes](#)  
*Region-map communication-protocol opcodes.*
- enum [L4Re::Video::Goos\\_::Opcodes](#)  
*Frame buffer communication-protocol opcodes.*

#### 14.10.11.1 Detailed Description

Fix [L4Re](#) Protocol Constants.

### 14.10.11.2 Enumeration Type Documentation

#### 14.10.11.2.1 L4re\_protocols

enum [L4re\\_protocols](#)

Common [L4Re](#) Protocol Constants.

#### Enumerator

---

L4RE_PROTO_DATASPACE	ID for <a href="#">L4Re::Dataspace</a> RPCs.
L4RE_PROTO_NAMESPACE	ID for <a href="#">L4Re::Namespace</a> RPCs.
L4RE_PROTO_PARENT	ID for <a href="#">L4Re::Parent</a> RPCs.
L4RE_PROTO_GOOS	ID for <a href="#">L4Re::Video::Goos</a> RPCs.
L4RE_PROTO_RSVD_1	Reserved ID.
L4RE_PROTO_RM	ID for <a href="#">L4Re::Rm</a> RPCs.
L4RE_PROTO_EVENT	ID for <a href="#">L4Re::Event</a> RPCs.
L4RE_PROTO_INHIBITOR	ID for <a href="#">L4Re::Inhibitor</a> RPCs.
L4RE_PROTO_DMA_SPACE	ID for <a href="#">L4Re::Dma_space</a> RPCs.
L4RE_PROTO_MMIO_SPACE	ID for <a href="#">L4Re::Mmio_space</a> .
L4RE_PROTO_ITAS	ID for <a href="#">L4Re::Itas</a> .
L4RE_PROTO_MEM_ALLOC	ID for <a href="#">L4Re::Mem_alloc</a> .
L4RE_PROTO_REMOTE_ACCESS	ID for <a href="#">L4Re::Remote_access</a> .
L4RE_PROTO_DEBUG	ID for debugging RPCs.

Definition at line 24 of file [protocols.h](#).

### 14.10.12 Region map API

Virtual address-space management.

Collaboration diagram for Region map API:



#### Data Structures

- class [L4Re::Rm](#)  
*Region map.*

#### 14.10.12.1 Detailed Description

Virtual address-space management.

A region map object implements two protocols. The first protocol is the kernel page-fault protocol, to resolve page faults for threads running in an [L4](#) task. The second protocol is the region map protocol itself, which allows managing the virtual memory address space of an [L4](#) task.

There are two basic concepts provided by the region map abstraction:

- **Areas** are reserved ranges in the virtual memory address space.
- **Regions** are ranges that are backed by (part of) a dataspace, i.e. accessing them results in access to the physical memory the dataspace manages.

Note that regions may live outside of areas and that an area does not necessarily contain any region.

Areas can be reserved for special use or for attaching a dataspace at a later time. When attaching a dataspace, the user can instruct the region map to search for an appropriate range to attach to. Regions are skipped in this search since they already have dataspace attached to them, and, depending on [L4Re::Rm::F::In\\_area](#), areas are skipped because they are reserved. Amongst others, areas can be used to attach several dataspace inside a certain range of addresses without interference from other threads.

When a region map receives a page fault IPC, the region map will check if the faulting virtual address lies in a region. If yes, it will answer the page fault IPC with a mapping from the backing dataspace. If not, an error is returned.

Depending on the system type, an attached dataspace might or might not be mapped eagerly. MMU-based systems resort to lazy mapping while systems without MMU do eager mappings by default. The [L4Re::Rm::F::Eager\\_map](#) and [L4Re::Rm::F::No\\_eager\\_map](#) flags can be used to force the respective behaviour, independent of the underlying system. In case both flags are given, the [L4Re::Rm::F::No\\_eager\\_map](#) flag wins.

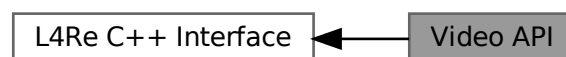
See also

[L4Re::Dataspace](#), [L4Re::Rm](#),  
[Memory management - Data Spaces and the Region Map](#)

### 14.10.13 Video API

API for framebuffer based graphics.

Collaboration diagram for Video API:



#### Data Structures

- class [L4Re::Video::Color\\_component](#)  
*A color component.*
- class [L4Re::Video::Pixel\\_info](#)  
*Pixel information.*

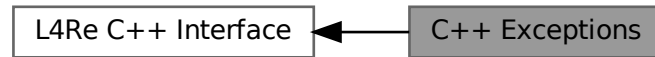
#### 14.10.13.1 Detailed Description

API for framebuffer based graphics.

Contains the basic APIs that abstract framebuffers and views into them for [L4Re](#) applications.

### 14.10.14 C++ Exceptions

Collaboration diagram for C++ Exceptions:



#### Files

- file [exceptions](#)  
*Base exceptions.*

#### Data Structures

- class [L4::Exception\\_tracer](#)  
*Back-trace support for exceptions.*
- class [L4::Base\\_exception](#)  
*Base class for all exceptions, thrown by the [L4Re](#) framework.*
- class [L4::Runtime\\_error](#)  
*Exception for an abstract runtime error.*
- class [L4::Out\\_of\\_memory](#)  
*Exception signalling insufficient memory.*
- class [L4::Element\\_already\\_exists](#)  
*Exception for duplicate element insertions.*
- class [L4::Unknown\\_error](#)  
*Exception for an unknown condition.*
- class [L4::Element\\_not\\_found](#)  
*Exception for a failed lookup (element not found).*
- class [L4::Invalid\\_capability](#)  
*Indicates that an invalid object was invoked.*
- class [L4::Com\\_error](#)  
*Error conditions during IPC.*
- class [L4::Bounds\\_error](#)  
*Access out of bounds.*

#### Macros

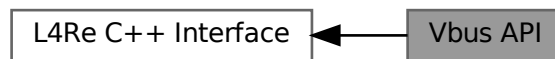
- `#define L4_CXX_EXCEPTION_BACKTRACE 20`  
*Number of instruction pointers in backtrace.*

#### 14.10.14.1 Detailed Description

### 14.10.15 Vbus API

C++ interface of the Vbus API.

Collaboration diagram for Vbus API:



#### Data Structures

- class [L4vbus::Pm< DEC >](#)  
*Power-management API mixin.*
- class [L4vbus::Device](#)  
*Device on a L4vbus::Vbus.*
- class [L4vbus::Icu](#)  
*Vbus Interrupt controller API.*
- class [L4vbus::Vbus](#)  
*The virtual bus (Vbus) interface.*
- class [L4vbus::Gpio\\_pin](#)  
*A GPIO pin.*
- class [L4vbus::Gpio\\_module](#)  
*A Gpio\_module groups multiple GPIO pins together.*
- class [L4vbus::Pci\\_host\\_bridge](#)  
*A Pci host bridge.*
- class [L4vbus::Pci\\_dev](#)  
*A PCI device.*

#### 14.10.15.1 Detailed Description

C++ interface of the Vbus API.

The virtual bus (Vbus) is a hierarchical (tree) structure of device nodes where each device has a set of resources attached to it. Each virtual bus provides an Icu ([Interrupt controller](#)) for interrupt handling.

The Vbus interface allows a client to find and query devices present on his virtual bus. After obtaining a device handle for a specific device the client can enumerate its resources.

Refer to [L4 Vbus functions](#) for the C API.

##### Include File

```
#include <l4/vbus/vbus>
```

##### Include File

```
#include <l4/vbus/vbus_gpio>
```

##### Include File

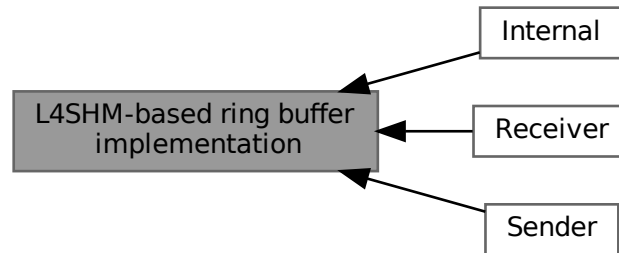
```
#include <l4/vbus/vbus_pci>
```



## 14.11 L4SHM-based ring buffer implementation

The library provides a non-locking (strictly 1:1) shared-memory-based ring buffer implementation based on the L4SHM library.

Collaboration diagram for L4SHM-based ring buffer implementation:



### Topics

- Sender • . . . . . [609](#)
- Receiver . . . . . [610](#)
- Internal • . . . . . [610](#)

### 14.11.1 Detailed Description

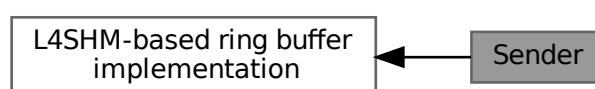
The library provides a non-locking (strictly 1:1) shared-memory-based ring buffer implementation based on the L4SHM library.

It requires an already allocated L4SHM area to be attached to sender and receiver. It will allocate an SHM chunk within this area and provides functions to produce data and consume data in FIFO order from the ring buffer.

The sender side of the buffer needs to be initialized *before* the receiver side, because allocation of the SHM chunk and the necessary signals is done on the sender side and the receiver initialization tries to attach to these objects.

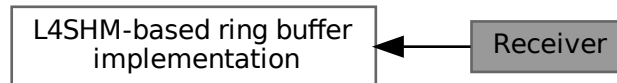
### 14.11.2 Sender

Collaboration diagram for Sender:



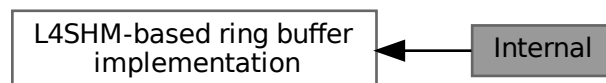
### 14.11.3 Receiver

Collaboration diagram for Receiver:



### 14.11.4 Internal

Collaboration diagram for Internal:



### Data Structures

- struct [l4shmc\\_ringbuf\\_head\\_t](#)  
*Head field of a ring buffer.*
- struct [l4shmc\\_ringbuf\\_t](#)  
*Ring buffer.*

### Macros

- #define [L4SHMC\\_RINGBUF\\_HEAD](#)(ringbuf)  
*Get ring buffer head pointer.*
- #define [L4SHMC\\_RINGBUF\\_DATA](#)(ringbuf)  
*Get ring buffer data pointer.*
- #define [L4SHMC\\_RINGBUF\\_DATA\\_SIZE](#)(ringbuf)  
*Get size of data area.*

#### 14.11.4.1 Detailed Description

#### 14.11.4.2 Macro Definition Documentation

##### 14.11.4.2.1 L4SHMC\_RINGBUF\_DATA

```
#define L4SHMC_RINGBUF_DATA(  
    ringbuf)
```

**Value:**

```
(L4SHMC_RINGBUF_HEAD(ringbuf)->data)
```

Get ring buffer data pointer.

**Parameters**

<i>ringbuf</i>	<a href="#">l4shmc_ringbuf_t</a> struct
----------------	-----------------------------------------

Definition at line 114 of file [ringbuf.h](#).

##### 14.11.4.2.2 L4SHMC\_RINGBUF\_DATA\_SIZE

```
#define L4SHMC_RINGBUF_DATA_SIZE(  
    ringbuf)
```

**Value:**

```
((ringbuf)->_size - sizeof(l4shmc_ringbuf_head_t))
```

Get size of data area.

**Parameters**

<i>ringbuf</i>	<a href="#">l4shmc_ringbuf_t</a> struct
----------------	-----------------------------------------

Definition at line 123 of file [ringbuf.h](#).

##### 14.11.4.2.3 L4SHMC\_RINGBUF\_HEAD

```
#define L4SHMC_RINGBUF_HEAD(  
    ringbuf)
```

**Value:**

```
((l4shmc_ringbuf_head_t*) ((ringbuf)->_addr))
```

Get ring buffer head pointer.

**Parameters**

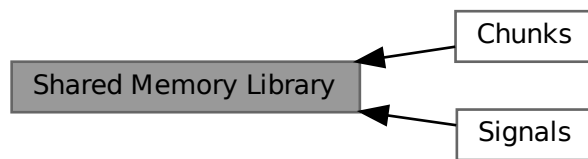
<code>ringbuf</code>	<code>l4shmc_ringbuf_t</code> struct
----------------------	--------------------------------------

Definition at line 105 of file `ringbuf.h`.

## 14.12 Shared Memory Library

L4SHM provides a shared memory infrastructure that establishes a shared memory area between multiple parties and uses a fast notification mechanism.

Collaboration diagram for Shared Memory Library:



### Topics

- [Chunks](#) . . . . . 617
- [Signals](#) . . . . . 630

### Functions

- [L4\\_BEGIN\\_DECLS](#) long [l4shmc\\_create](#) (char const \*shmc\_name)  
*Create a shared memory area.*
- long [l4shmc\\_attach](#) (char const \*shmc\_name, l4shmc\_area\_t \*shmarea)  
*Attach to a shared memory area.*
- long [l4shmc\\_get\\_client\\_nr](#) (l4shmc\_area\_t const \*shmarea)  
*Determine the client number of the shared memory region.*
- long [l4shmc\\_mark\\_client\\_initialized](#) (l4shmc\_area\_t \*shmarea)  
*Mark this shared memory client as 'initialized'.*
- long [l4shmc\\_get\\_initialized\\_clients](#) (l4shmc\_area\_t \*shmarea, l4\_umword\_t \*bitmask)  
*Fetch the `_clients_init_done` bitmask of the shared memory area.*
- long [l4shmc\\_connect\\_chunk\\_signal](#) (l4shmc\_chunk\_t \*chunk, l4shmc\_signal\_t \*signal)  
*Connect a signal with a chunk.*
- long [l4shmc\\_area\\_size](#) (l4shmc\_area\_t const \*shmarea)  
*Get size of shared memory area.*
- long [l4shmc\\_area\\_size\\_free](#) (l4shmc\_area\_t const \*shmarea)  
*Get free size of shared memory area.*
- long [l4shmc\\_area\\_overhead](#) (void)  
*Get memory overhead per area that is not available for chunks.*
- long [l4shmc\\_chunk\\_overhead](#) (void)  
*Get memory overhead required in addition to the chunk capacity for adding one chunk.*

### 14.12.1 Detailed Description

L4SHM provides a shared memory infrastructure that establishes a shared memory area between multiple parties and uses a fast notification mechanism.

A shared memory area consists of chunks and signals. A chunk is a defined chunk of memory within the memory area with a maximum size. A chunk is filled (written) by a producer and read by a consumer. When a producer has finished writing to the chunk it signals a data ready notification to the consumer.

A consumer attaches to a chunk and waits for the producer to fill the chunk. After reading out the chunk it marks the chunk free again.

A shared memory area can have multiple chunks.

The interface is divided in three roles.

- The master role, responsible for setting up a shared memory area.
- A producer, generating data into a chunk
- A consumer, receiving data.

A signal can be connected with a chunk or can be used independently (e.g. for multiple chunks).

### 14.12.2 Function Documentation

#### 14.12.2.1 `l4shmc_area_overhead()`

```
long l4shmc_area_overhead (
    void )
```

Get memory overhead per area that is not available for chunks.

##### Returns

Size of the overhead in bytes.

References [L4\\_CV](#).

#### 14.12.2.2 `l4shmc_area_size()`

```
long l4shmc_area_size (
    l4shmc_area_t const * shmarea)
```

Get size of shared memory area.

##### Parameters

<i>shmarea</i>	Shared memory area.
----------------	---------------------

##### Return values

---

$>0$	Size of the shared memory area.
$<0$	Error.

References [L4\\_CV](#).

#### 14.12.2.3 l4shmc\_area\_size\_free()

```
long l4shmc_area_size_free (
    l4shmc_area_t const * shmarea)
```

Get free size of shared memory area.

To get the max size to pass to `l4shmc_add_chunk`, subtract [l4shmc\\_chunk\\_overhead\(\)](#).

##### Parameters

<i>shmarea</i>	Shared memory area.
----------------	---------------------

##### Returns

Size of the shared memory area.

References [L4\\_CV](#).

#### 14.12.2.4 l4shmc\_attach()

```
long l4shmc_attach (
    char const * shm_name,
    l4shmc_area_t * shmarea)
```

Attach to a shared memory area.

##### Parameters

	<i>shm_name</i>	Name of the shared memory area.
out	<i>shmarea</i>	Pointer to shared memory area descriptor to be filled with information for the shared memory area.

On success, the data in 'shmarea' contains a client number which can be used to mutual agree about client initialization:

- [l4shmc\\_get\\_client\\_nr\(\)](#) returns the client number stored in 'shmarea'. The first attached client will get 0 and this number is increased for each attached client.
- [l4shmc\\_mark\\_client\\_initialized\(\)](#) tells other clients that this client has finished its initialization.
- [l4shmc\\_get\\_initialized\\_clients\(\)](#) returns the bitmap of initialized clients attached to this shared memory.

##### Return values

---

0	Success.
<0	Error.

**Examples**

[examples/libs/shmc/prodcons.c](#).

References [L4\\_CV](#).

**14.12.2.5 l4shmc\_chunk\_overhead()**

```
long l4shmc_chunk_overhead (
    void )
```

Get memory overhead required in addition to the chunk capacity for adding one chunk.

**Returns**

Size of the overhead in bytes.

References [L4\\_END\\_DECLS](#).

**14.12.2.6 l4shmc\_connect\_chunk\_signal()**

```
long l4shmc_connect_chunk_signal (
    l4shmc_chunk_t * chunk,
    l4shmc_signal_t * signal)
```

Connect a signal with a chunk.

**Parameters**

<i>chunk</i>	Chunk to attach the signal to.
<i>signal</i>	Signal to attach.

**Return values**

0	Success.
<0	Error.

**Examples**

[examples/libs/shmc/prodcons.c](#).

References [L4\\_CV](#), and [L4\\_INLINE](#).

**14.12.2.7 l4shmc\_create()**

```
L4\_BEGIN\_DECLS long l4shmc_create (
    char const * shmc_name)
```

Create a shared memory area.

**Parameters**

<i>shmc_name</i>	Name of the shared memory area.
------------------	---------------------------------

### Return values

0	Success.
-L4_ENOMEM	The requested size is too big.
-L4_ENOENT	No valid capability with the name of the shared memory area found.
<0	Errors from l4re_rm_attach or l4re_ns_register_obj_srv.

### Examples

[examples/libs/shmc/prodcons.c](#).

References [L4\\_CV](#).

#### 14.12.2.8 l4shmc\_get\_client\_nr()

```
long l4shmc_get_client_nr (
    l4shmc_area_t const * shmarea)
```

Determine the client number of the shared memory region.

### Parameters

<i>shmarea</i>	The shared memory area.
----------------	-------------------------

### Returns

client number.

References [L4\\_CV](#).

#### 14.12.2.9 l4shmc\_get\_initialized\_clients()

```
long l4shmc_get_initialized_clients (
    l4shmc_area_t * shmarea,
    l4_umword_t * bitmask)
```

Fetch the `_clients_init_done` bitmask of the shared memory area.

### Parameters

	<i>shmarea</i>	The shared memory area.
out	<i>bitmask</i>	The bitmask describing which clients are initialized.

### Return values



0	Success.
<0	Error.

See also

[l4shmc\\_mark\\_client\\_initialized\(\)](#), [l4shmc\\_get\\_client\\_nr\(\)](#)

Examples

[examples/libs/shmc/prodcons.c](#).

References [L4\\_CV](#).

#### 14.12.2.10 l4shmc\_mark\_client\_initialized()

```
long l4shmc_mark_client_initialized (
    l4shmc_area_t * shmarea)
```

Mark this shared memory client as 'initialized'.

The corresponding bit is set in the `_clients_init_done` bitmask. The bitmask can be fetched with [l4shmc\\_get\\_initialized\\_clients\(\)](#).

Parameters

<i>shmarea</i>	The shared memory area.
----------------	-------------------------

Return values

0	Success.
<0	Error.

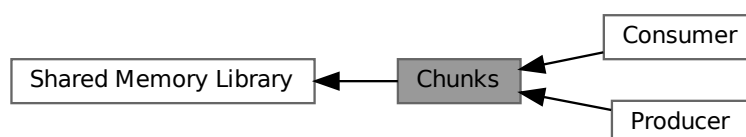
Examples

[examples/libs/shmc/prodcons.c](#).

References [L4\\_CV](#).

### 14.12.3 Chunks

Collaboration diagram for Chunks:



## Topics

- [Producer](#) . . . . . 621
- [Consumer](#) . . . . . 625

## Functions

- long [l4shmc\\_add\\_chunk](#) (l4shmc\_area\_t \*shmarea, char const \*chunk\_name, [l4\\_umword\\_t](#) chunk\_capacity, l4shmc\_chunk\_t \*chunk)  
*Add a chunk in the shared memory area.*
- long [l4shmc\\_get\\_chunk](#) (l4shmc\_area\_t \*shmarea, char const \*chunk\_name, l4shmc\_chunk\_t \*chunk)  
*Get chunk out of shared memory area.*
- long [l4shmc\\_get\\_chunk\\_to](#) (l4shmc\_area\_t \*shmarea, char const \*chunk\_name, [l4\\_umword\\_t](#) timeout\_ms, l4shmc\_chunk\_t \*chunk)  
*Get chunk out of shared memory area, with timeout.*
- long [l4shmc\\_iterate\\_chunk](#) (l4shmc\_area\_t const \*shmarea, char const \*\*chunk\_name, long offs)  
*Iterate over names of all existing chunks.*
- void \* [l4shmc\\_chunk\\_ptr](#) (l4shmc\_chunk\_t const \*chunk)  
*Get data pointer to chunk.*
- long [l4shmc\\_chunk\\_capacity](#) (l4shmc\_chunk\_t const \*chunk)  
*Get capacity of a chunk.*
- l4shmc\_signal\_t \* [l4shmc\\_chunk\\_signal](#) (l4shmc\_chunk\_t const \*chunk)  
*Get the registered signal of a chunk.*

### 14.12.3.1 Detailed Description

### 14.12.3.2 Function Documentation

#### 14.12.3.2.1 l4shmc\_add\_chunk()

```
long l4shmc_add_chunk (
    l4shmc_area_t * shmarea,
    char const * chunk_name,
    l4\_umword\_t chunk_capacity,
    l4shmc_chunk_t * chunk)
```

Add a chunk in the shared memory area.

#### Parameters

	<i>shmarea</i>	The shared memory area to put the chunk in.
	<i>chunk_name</i>	Name of the chunk.
	<i>chunk_capacity</i>	Capacity for payload of the chunk in bytes.
out	<i>chunk</i>	Chunk structure to fill in.

#### Return values

0	Success.
<0	Error.

#### Examples

[examples/libs/shmc/prodcons.c](#).

References [L4\\_CV](#).

#### 14.12.3.2.2 l4shmc\_chunk\_capacity()

```
long l4shmc_chunk_capacity (  
    l4shmc_chunk_t const * chunk) [inline]
```

Get capacity of a chunk.

#### Parameters

<i>chunk</i>	Chunk.
--------------	--------

#### Returns

Capacity of the chunk in bytes.

References [L4\\_CV](#), and [L4\\_INLINE](#).

#### 14.12.3.2.3 l4shmc\_chunk\_ptr()

```
void * l4shmc_chunk_ptr (  
    l4shmc_chunk_t const * chunk) [inline]
```

Get data pointer to chunk.

#### Parameters

<i>chunk</i>	Chunk.
--------------	--------

#### Returns

Chunk pointer.

#### Examples

[examples/libs/shmc/prodcons.c](#).

References [L4\\_CV](#), and [L4\\_INLINE](#).

#### 14.12.3.2.4 l4shmc\_chunk\_signal()

```
l4shmc_signal_t * l4shmc_chunk_signal (  
    l4shmc_chunk_t const * chunk) [inline]
```

Get the registered signal of a chunk.

#### Parameters

<i>chunk</i>	Chunk.
--------------	--------

### Return values

0	No signal has been registered with this chunk.
!=0	Pointer to signal otherwise.

References [L4\\_CV](#), and [L4\\_INLINE](#).

#### 14.12.3.2.5 l4shmc\_get\_chunk()

```
long l4shmc_get_chunk (
    l4shmc_area_t * shmarea,
    char const * chunk_name,
    l4shmc_chunk_t * chunk) [inline]
```

Get chunk out of shared memory area.

### Parameters

	<i>shmarea</i>	Shared memory area.
	<i>chunk_name</i>	Name of the chunk.
out	<i>chunk</i>	Chunk data structure to fill.

### Return values

0	Success.
<0	Error.

### Examples

[examples/libs/shmc/prodcons.c](#).

References [L4\\_CV](#).

#### 14.12.3.2.6 l4shmc\_get\_chunk\_to()

```
long l4shmc_get_chunk_to (
    l4shmc_area_t * shmarea,
    char const * chunk_name,
    l4_umword_t timeout_ms,
    l4shmc_chunk_t * chunk)
```

Get chunk out of shared memory area, with timeout.

### Parameters

	<i>shmarea</i>	Shared memory area.
	<i>chunk_name</i>	Name of the chunk.
	<i>timeout_ms</i>	Timeout in milliseconds to wait for the chunk to appear in the shared memory area.
out	<i>chunk</i>	Chunk data structure to fill.

**Return values**

0	Success.
<0	Error.

References [L4\\_CV](#).

**14.12.3.2.7 l4shmc\_iterate\_chunk()**

```
long l4shmc_iterate_chunk (
    l4shmc_area_t const * shmarea,
    char const ** chunk_name,
    long offs)
```

Iterate over names of all existing chunks.

**Parameters**

<i>shmarea</i>	Shared memory area.
<i>chunk_name</i>	Where the name of the current chunk will be stored
<i>offs</i>	0 to start iteration, return value of previous call to <a href="#">l4shmc_iterate_chunk()</a> to get next chunk

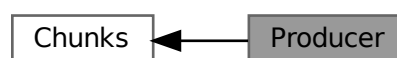
**Return values**

0	No more chunks available.
<0	Error.
>0	Iterator value for the next call.

References [L4\\_CV](#).

**14.12.3.3 Producer**

Collaboration diagram for Producer:



## Functions

- long [l4shmc\\_chunk\\_try\\_to\\_take](#) (l4shmc\_chunk\_t \*chunk)  
*Try to mark chunk busy.*
- long [l4shmc\\_chunk\\_try\\_to\\_take\\_for\\_writing](#) (l4shmc\_chunk\_t \*chunk)  
*Try to mark chunk busy writing.*
- long [l4shmc\\_chunk\\_try\\_to\\_take\\_for\\_overwriting](#) (l4shmc\_chunk\_t \*chunk)  
*Try to mark the chunk busy writing after it was ready for reading.*
- long [l4shmc\\_chunk\\_ready](#) (l4shmc\_chunk\_t \*chunk, [l4\\_umword\\_t](#) size)  
*Mark chunk as filled (ready).*
- long [l4shmc\\_chunk\\_ready\\_sig](#) (l4shmc\_chunk\_t \*chunk, [l4\\_umword\\_t](#) size)  
*Mark chunk as filled (ready) and signal consumer.*
- long [l4shmc\\_is\\_chunk\\_clear](#) (l4shmc\_chunk\_t const \*chunk)  
*Check whether chunk is free.*

### 14.12.3.3.1 Detailed Description

### 14.12.3.3.2 Function Documentation

#### 14.12.3.3.2.1 l4shmc\_chunk\_ready()

```
long l4shmc_chunk_ready (
    l4shmc_chunk_t * chunk,
    l4_umword_t size) [inline]
```

Mark chunk as filled (ready).

#### Parameters

<i>chunk</i>	chunk.
<i>size</i>	Size of data in the chunk, in bytes.

#### Return values

0	Success.
<0	Error.

References [L4\\_CV](#), and [L4\\_INLINE](#).

#### 14.12.3.3.2.2 l4shmc\_chunk\_ready\_sig()

```
long l4shmc_chunk_ready_sig (
    l4shmc_chunk_t * chunk,
    l4_umword_t size) [inline]
```

Mark chunk as filled (ready) and signal consumer.

#### Parameters

<i>chunk</i>	chunk.
<i>size</i>	Size of data in the chunk, in bytes.

**Return values**

0	Success.
<0	Error.

**Examples**

[examples/libs/shmc/prodcons.c](#).

References [L4\\_CV](#), and [L4\\_INLINE](#).

**14.12.3.3.2.3 l4shmc\_chunk\_try\_to\_take()**

```
long l4shmc_chunk_try_to_take (  
    l4shmc_chunk_t * chunk) [inline]
```

Try to mark chunk busy.

**Parameters**

<i>chunk</i>	chunk to mark.
--------------	----------------

**Return values**

0	Chunk could be taken.
<0	Chunk could not be taken, try again.

**Examples**

[examples/libs/shmc/prodcons.c](#).

References [L4\\_CV](#), and [L4\\_INLINE](#).

**14.12.3.3.2.4 l4shmc\_chunk\_try\_to\_take\_for\_overwriting()**

```
long l4shmc_chunk_try_to_take_for_overwriting (  
    l4shmc_chunk_t * chunk) [inline]
```

Try to mark the chunk busy writing after it was ready for reading.

**Parameters**

<i>chunk</i>	chunk to mark busy writing.
--------------	-----------------------------

This function is used by the producer to overwrite a message if the consumer did not read the message within an expected time. This function can only be used if the consumer uses [l4shmc\\_chunk\\_try\\_to\\_take\\_for\\_reading\(\)](#) before reading the chunk.

#### Return values

0	Chunk could be taken and can be written.
<0	Chunk could not be taken, try again.

References [L4\\_CV](#), and [L4\\_INLINE](#).

#### 14.12.3.3.2.5 l4shmc\_chunk\_try\_to\_take\_for\_writing()

```
long l4shmc_chunk_try_to_take_for_writing (
    l4shmc_chunk_t * chunk) [inline]
```

Try to mark chunk busy writing.

This function is actually an alias for [l4shmc\\_chunk\\_try\\_to\\_take\(\)](#).

#### Parameters

<i>chunk</i>	chunk to mark busy writing.
--------------	-----------------------------

#### Return values

0	Chunk could be taken and can be written.
<0	Chunk could not be taken, try again.

References [L4\\_CV](#), and [L4\\_INLINE](#).

#### 14.12.3.3.2.6 l4shmc\_is\_chunk\_clear()

```
long l4shmc_is_chunk_clear (
    l4shmc_chunk_t const * chunk) [inline]
```

Check whether chunk is free.

#### Parameters

<i>chunk</i>	Chunk to check.
--------------	-----------------

#### Return values

---

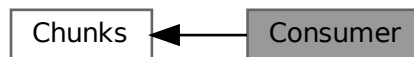


<code>!=0</code>	Chunk is clear.
<code>0</code>	Chunk is not clear.

References [L4\\_CV](#), and [L4\\_INLINE](#).

#### 14.12.3.4 Consumer

Collaboration diagram for Consumer:



#### Functions

- long [l4shmc\\_chunk\\_try\\_to\\_take\\_for\\_reading](#) (l4shmc\_chunk\_t \*chunk)  
*Try to mark chunk busy reading.*
- long [l4shmc\\_enable\\_chunk](#) (l4shmc\_chunk\_t \*chunk)  
*Enable a signal connected with a chunk.*
- long [l4shmc\\_wait\\_chunk](#) (l4shmc\_chunk\_t \*chunk)  
*Wait on a specific chunk.*
- long [l4shmc\\_wait\\_chunk\\_to](#) (l4shmc\_chunk\_t \*chunk, [l4\\_timeout\\_t](#) timeout)  
*Check whether a specific chunk has an event pending, with timeout.*
- long [l4shmc\\_wait\\_chunk\\_try](#) (l4shmc\_chunk\_t \*chunk)  
*Check whether a specific chunk has an event pending.*
- long [l4shmc\\_chunk\\_consumed](#) (l4shmc\_chunk\_t \*chunk)  
*Mark a chunk as free.*
- long [l4shmc\\_is\\_chunk\\_ready](#) (l4shmc\_chunk\_t const \*chunk)  
*Check whether data is available.*
- long [l4shmc\\_chunk\\_size](#) (l4shmc\_chunk\_t const \*chunk)  
*Get current size of a chunk.*

##### 14.12.3.4.1 Detailed Description

##### 14.12.3.4.2 Function Documentation

###### 14.12.3.4.2.1 l4shmc\_chunk\_consumed()

```
long l4shmc_chunk_consumed (
    l4shmc_chunk_t * chunk) [inline]
```

Mark a chunk as free.

#### Parameters

---

<i>chunk</i>	Chunk to mark as free.
--------------	------------------------

**Return values**

0	Success.
<0	Error.

**Examples**

[examples/libs/shmc/prodcons.c](#).

References [L4\\_CV](#).

**14.12.3.4.2.2 l4shmc\_chunk\_size()**

```
long l4shmc_chunk_size (  
    l4shmc_chunk_t const * chunk) [inline]
```

Get current size of a chunk.

**Parameters**

<i>chunk</i>	Chunk.
--------------	--------

**Returns**

Current size of the chunk in bytes.

**Examples**

[examples/libs/shmc/prodcons.c](#).

References [L4\\_CV](#), and [L4\\_INLINE](#).

**14.12.3.4.2.3 l4shmc\_chunk\_try\_to\_take\_for\_reading()**

```
long l4shmc_chunk_try_to_take_for_reading (  
    l4shmc_chunk_t * chunk) [inline]
```

Try to mark chunk busy reading.

**Parameters**

<i>chunk</i>	chunk to mark busy reading.
--------------	-----------------------------

**Return values**

---

0	Chunk could be taken and can be read.
<0	Chunk could not be taken, try again.

References [L4\\_CV](#), and [L4\\_INLINE](#).

#### 14.12.3.4.2.4 l4shmc\_enable\_chunk()

```
long l4shmc_enable_chunk (  
    l4shmc_chunk_t * chunk)
```

Enable a signal connected with a chunk.

##### Parameters

<i>chunk</i>	Chunk to enable.
--------------	------------------

##### Return values

0	Success.
<0	Error.

A signal must be enabled before waiting when the consumer waits on any signal. Enabling is not needed if the consumer waits for a specific signal or chunk.

References [L4\\_CV](#), and [L4\\_INLINE](#).

#### 14.12.3.4.2.5 l4shmc\_is\_chunk\_ready()

```
long l4shmc_is_chunk_ready (  
    l4shmc_chunk_t const * chunk) [inline]
```

Check whether data is available.

##### Parameters

<i>chunk</i>	Chunk to check.
--------------	-----------------

##### Return values

<i>!=0</i>	Data is available.
0	No data available.

References [L4\\_CV](#), and [L4\\_INLINE](#).

#### 14.12.3.4.2.6 l4shmc\_wait\_chunk()

```
long l4shmc_wait_chunk (  
    l4shmc_chunk_t * chunk)    [inline]
```

Wait on a specific chunk.

#### Parameters

---

<i>chunk</i>	Chunk to wait for.
--------------	--------------------

**Return values**

0	Success.
<0	Error.

**Examples**

[examples/libs/shmc/prodcons.c](#).

References [L4\\_CV](#).

**14.12.3.4.2.7 l4shmc\_wait\_chunk\_to()**

```
long l4shmc_wait_chunk_to (  
    l4shmc_chunk_t * chunk,  
    l4_timeout_t timeout)
```

Check whether a specific chunk has an event pending, with timeout.

**Parameters**

<i>chunk</i>	Chunk to check.
<i>timeout</i>	Timeout.

**Return values**

0	Success.
<0	Error.

The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

References [L4\\_CV](#), and [L4\\_INLINE](#).

**14.12.3.4.2.8 l4shmc\_wait\_chunk\_try()**

```
long l4shmc_wait_chunk_try (  
    l4shmc_chunk_t * chunk) [inline]
```

Check whether a specific chunk has an event pending.

**Parameters**

<i>chunk</i>	Chunk to check.
--------------	-----------------

### Return values

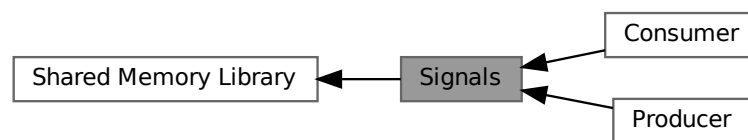
0	Success.
<0	Error.

The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

References [L4\\_CV](#), and [L4\\_INLINE](#).

## 14.12.4 Signals

Collaboration diagram for Signals:



### Topics

- [Producer](#) ..... 633
- [Consumer](#) ..... 634

### Functions

- long [l4shmc\\_add\\_signal](#) (l4shmc\_area\_t \*shmarea, char const \*signal\_name, l4shmc\_signal\_t \*signal)  
Add a signal for the shared memory area.
- long [l4shmc\\_attach\\_signal](#) (l4shmc\_area\_t \*shmarea, char const \*signal\_name, [l4\\_cap\\_idx\\_t](#) thread, l4shmc\_signal\_t \*signal)  
Attach to signal.
- long [l4shmc\\_get\\_signal](#) (l4shmc\_area\_t \*shmarea, char const \*signal\_name, l4shmc\_signal\_t \*signal)  
Get signal object from the shared memory area.
- [l4\\_cap\\_idx\\_t](#) [l4shmc\\_signal\\_cap](#) (l4shmc\_signal\_t const \*signal)  
Get the signal capability of a signal.
- long [l4shmc\\_check\\_magic](#) (l4shmc\_chunk\_t const \*chunk)  
Check magic value of a chunk.

**14.12.4.1 Detailed Description****14.12.4.2 Function Documentation****14.12.4.2.1 l4shmc\_add\_signal()**

```
long l4shmc_add_signal (
    l4shmc_area_t * shmarea,
    char const * signal_name,
    l4shmc_signal_t * signal)
```

Add a signal for the shared memory area.

**Parameters**

	<i>shmarea</i>	The shared memory area.
	<i>signal_name</i>	Name of the signal.
out	<i>signal</i>	Signal structure to fill in.

**Return values**

0	Success.
<0	Error.

**Examples**

[examples/libs/shmc/prodcons.c](#).

References [L4\\_CV](#), and [L4\\_INLINE](#).

**14.12.4.2.2 l4shmc\_attach\_signal()**

```
long l4shmc_attach_signal (
    l4shmc_area_t * shmarea,
    char const * signal_name,
    l4_cap_idx_t thread,
    l4shmc_signal_t * signal)
```

Attach to signal.

**Parameters**

	<i>shmarea</i>	Shared memory area.
	<i>signal_name</i>	Name of the signal.
	<i>thread</i>	Thread capability index to attach the signal to.
out	<i>signal</i>	Signal data structure to fill.

**Return values**

<i>0</i>	Success.
<i>&lt;0</i>	Error.

#### Examples

[examples/libs/shmc/prodcons.c](#).

References [L4\\_CV](#).

#### 14.12.4.2.3 l4shmc\_check\_magic()

```
long l4shmc_check_magic (
    l4shmc_chunk_t const * chunk) [inline]
```

Check magic value of a chunk.

#### Parameters

<i>chunk</i>	Chunk.
--------------	--------

#### Return values

<i>0</i>	Magic value is not valid.
<i>&gt;0</i>	Chunk is OK, the magic value is valid.

References [L4\\_CV](#).

#### 14.12.4.2.4 l4shmc\_get\_signal()

```
long l4shmc_get_signal (
    l4shmc_area_t * shmarea,
    char const * signal_name,
    l4shmc_signal_t * signal)
```

Get signal object from the shared memory area.

#### Parameters

	<i>shmarea</i>	Shared memory area.
	<i>signal_name</i>	Name of the signal.
out	<i>signal</i>	Signal data structure to fill.

#### Return values

---



0	Success.
<0	Error.

References [L4\\_CV](#).

#### 14.12.4.2.5 l4shmc\_signal\_cap()

```
l4_cap_idx_t l4shmc_signal_cap (
    l4shmc_signal_t const * signal) [inline]
```

Get the signal capability of a signal.

##### Parameters

<i>signal</i>	Signal.
---------------	---------

##### Returns

Capability of the signal object.

References [L4\\_CV](#), and [L4\\_INLINE](#).

#### 14.12.4.3 Producer

Collaboration diagram for Producer:



##### Functions

- long [l4shmc\\_trigger](#) (l4shmc\_signal\_t \*signal)  
*Trigger a signal.*

#### 14.12.4.3.1 Detailed Description

#### 14.12.4.3.2 Function Documentation

##### 14.12.4.3.2.1 l4shmc\_trigger()

```
long l4shmc_trigger (
    l4shmc_signal_t * signal) [inline]
```

Trigger a signal.

##### Parameters

<i>signal</i>	Signal to trigger.
---------------	--------------------

### Return values

0	Success.
<0	Error.

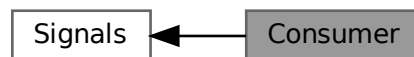
### Examples

[examples/libs/shmc/prodcons.c](#).

References [L4\\_CV](#), and [L4\\_INLINE](#).

#### 14.12.4.4 Consumer

Collaboration diagram for Consumer:



### Functions

- long [l4shmc\\_enable\\_signal](#) (l4shmc\_signal\_t \*signal)  
*Enable a signal.*
- long [l4shmc\\_wait\\_any](#) (l4shmc\_signal\_t \*\*retsignal)  
*Wait on any signal.*
- long [l4shmc\\_wait\\_any\\_try](#) (l4shmc\_signal\_t \*\*retsignal)  
*Check whether any waited signal has an event pending.*
- long [l4shmc\\_wait\\_any\\_to](#) (l4\_timeout\_t timeout, l4shmc\_signal\_t \*\*retsignal)  
*Wait for any signal with timeout.*
- long [l4shmc\\_wait\\_signal](#) (l4shmc\_signal\_t \*signal)  
*Wait on a specific signal.*
- long [l4shmc\\_wait\\_signal\\_to](#) (l4shmc\_signal\_t \*signal, l4\_timeout\_t timeout)  
*Wait on a specific signal, with timeout.*
- long [l4shmc\\_wait\\_signal\\_try](#) (l4shmc\_signal\_t \*signal)  
*Check whether a specific signal has an event pending.*

#### 14.12.4.4.1 Detailed Description

#### 14.12.4.4.2 Function Documentation

##### 14.12.4.4.2.1 l4shmc\_enable\_signal()

```
long l4shmc_enable_signal (  
    l4shmc_signal_t * signal)
```

Enable a signal.

#### Parameters

---

<i>signal</i>	Signal to enable.
---------------	-------------------

#### Return values

0	Success.
<0	Error.

A signal must be enabled before waiting when the consumer waits on any signal. Enabling is not needed if the consumer waits for a specific signal or chunk.

References [L4\\_CV](#).

#### 14.12.4.4.2.2 l4shmc\_wait\_any()

```
long l4shmc_wait_any (
    l4shmc_signal_t ** retsignal) [inline]
```

Wait on any signal.

#### Parameters

out	<i>retsignal</i>	Signal received.
-----	------------------	------------------

#### Return values

0	Success.
<0	Error.

References [L4\\_CV](#), and [L4\\_INLINE](#).

#### 14.12.4.4.2.3 l4shmc\_wait\_any\_to()

```
long l4shmc_wait_any_to (
    l4_timeout_t timeout,
    l4shmc_signal_t ** retsignal)
```

Wait for any signal with timeout.

#### Parameters

	<i>timeout</i>	Timeout.
out	<i>retsignal</i>	Signal that has the event pending if any.

#### Return values

0	Success.
<0	Error.

The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

References [L4\\_CV](#), and [L4\\_INLINE](#).

#### 14.12.4.4.2.4 l4shmc\_wait\_any\_try()

```
long l4shmc_wait_any_try (  
    l4shmc_signal_t ** retsignal) [inline]
```

Check whether any waited signal has an event pending.

##### Parameters

out	<i>retsignal</i>	Signal that has the event pending if any.
-----	------------------	-------------------------------------------

##### Return values

0	Success.
<0	Error.

The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

References [L4\\_CV](#).

#### 14.12.4.4.2.5 l4shmc\_wait\_signal()

```
long l4shmc_wait_signal (  
    l4shmc_signal_t * signal) [inline]
```

Wait on a specific signal.

##### Parameters

<i>signal</i>	Signal to wait for.
---------------	---------------------

##### Return values

0	Success.
<0	Error.

##### Examples

[examples/libs/shmc/prodcons.c](#).

References [L4\\_CV](#).

#### 14.12.4.4.2.6 l4shmc\_wait\_signal\_to()

```
long l4shmc_wait_signal_to (  
    l4shmc_signal_t * signal,  
    l4_timeout_t timeout)
```

Wait on a specific signal, with timeout.

##### Parameters

<i>signal</i>	Signal to wait for.
<i>timeout</i>	Timeout.

##### Return values

0	Success.
<0	Error.

References [L4\\_CV](#), and [L4\\_INLINE](#).

#### 14.12.4.4.2.7 l4shmc\_wait\_signal\_try()

```
long l4shmc_wait_signal_try (  
    l4shmc_signal_t * signal) [inline]
```

Check whether a specific signal has an event pending.

##### Parameters

<i>signal</i>	Signal to check.
---------------	------------------

##### Return values

0	Success.
<0	Error.

The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

References [L4\\_CV](#), and [L4\\_INLINE](#).

## 14.13 Sigma0 API

Sigma0 API bindings.

Collaboration diagram for Sigma0 API:



### Topics

- [Internal constants](#) . . . . . 643  
*Internal sigma0 definitions.*

### Files

- file [sigma0.h](#)  
*Sigma0 interface.*

### Enumerations

- enum [l4sigma0\\_return\\_flags\\_t](#) {  
  [L4SIGMA0\\_OK](#) , [L4SIGMA0\\_NOTALIGNED](#) , [L4SIGMA0\\_IPCERROR](#) , [L4SIGMA0\\_NOFPAGE](#) ,  
  [L4SIGMA0\\_4](#) , [L4SIGMA0\\_5](#) , [L4SIGMA0\\_SMALLERFPAGE](#) }  
*Return flags of libsigma0 functions.*

### Functions

- [L4\\_BEGIN\\_DECLS](#) [l4\\_kernel\\_info\\_t](#) \* [l4sigma0\\_map\\_kip](#) ([l4\\_cap\\_idx\\_t](#) sigma0, void \*addr, unsigned log2\_↔\_size)  
*Map the kernel info page from sigma0 to addr.*
- int [l4sigma0\\_map\\_mem](#) ([l4\\_cap\\_idx\\_t](#) sigma0, [l4\\_addr\\_t](#) phys, [l4\\_addr\\_t](#) virt, [l4\\_addr\\_t](#) size)  
*Request a memory mapping from sigma0.*
- int [l4sigma0\\_map\\_iomem](#) ([l4\\_cap\\_idx\\_t](#) sigma0, [l4\\_addr\\_t](#) phys, [l4\\_addr\\_t](#) virt, [l4\\_addr\\_t](#) size, int cached)  
*Request IO memory from sigma0.*
- int [l4sigma0\\_map\\_anypage](#) ([l4\\_cap\\_idx\\_t](#) sigma0, [l4\\_addr\\_t](#) map\_area, unsigned log2\_map\_size, [l4\\_addr\\_t](#) \*base, unsigned sz)  
*Request an arbitrary free page of RAM.*
- void [l4sigma0\\_debug\\_dump](#) ([l4\\_cap\\_idx\\_t](#) sigma0)  
*Request sigma0 to dump internal debug information.*
- char const \* [l4sigma0\\_map\\_errstr](#) (int err)  
*Get user readable error messages for the return codes.*

### 14.13.1 Detailed Description

Sigma0 API bindings.

Convenience bindings for the Sigma0 protocol.

### 14.13.2 Enumeration Type Documentation

#### 14.13.2.1 l4sigma0\_return\_flags\_t

```
enum l4sigma0_return_flags_t
```

Return flags of libsigma0 functions.

##### Enumerator

L4SIGMA0_OK	Ok.
L4SIGMA0_NOTALIGNED	Phys, virt or size not aligned.
L4SIGMA0_IPCERROR	IPC error.
L4SIGMA0_NOFPAGE	No fpage received.
L4SIGMA0_SMALLERFPAGE	Superpage requested but smaller flexpage received.

Definition at line 74 of file [sigma0.h](#).

### 14.13.3 Function Documentation

#### 14.13.3.1 l4sigma0\_debug\_dump()

```
void l4sigma0_debug_dump (
    l4_cap_idx_t sigma0)
```

Request sigma0 to dump internal debug information.

##### Parameters

<i>sigma0</i>	Capability selector for the sigma0 gate.
---------------	------------------------------------------

The debug information, such as internal memory maps, as well as statistics about the internal allocators is dumped to the kernel debugger.

References [L4\\_CV](#), and [L4\\_INLINE](#).

#### 14.13.3.2 l4sigma0\_map\_anypage()

```
int l4sigma0_map_anypage (
    l4_cap_idx_t sigma0,
    l4_addr_t map_area,
    unsigned log2_map_size,
    l4_addr_t * base,
    unsigned sz)
```

Request an arbitrary free page of RAM.

##### Parameters



	<i>sigma0</i>	Capability selector for the sigma0 gate.
	<i>map_area</i>	The base address of the local virtual memory area where the page should be mapped.
	<i>log2_map_size</i>	The size of the requested page log 2 (the size in bytes is $2^{\log2\_map\_size}$ ). This must be at least the minimal page size. By specifying larger sizes the largest possible hardware page size will be used.
out	<i>base</i>	Physical address of the page received (i.e. the send base of the received mapping if any).
	<i>sz</i>	Size to map by the server in $2^{sz}$ bytes.

### Return values

0	Success.
-L4SIGMA0_IPCERROR	IPC error.
-L4SIGMA0_NOFPAGE	No fpage received.

This function requests arbitrary free memory from sigma0. It should be used whenever spare memory is needed, instead of requesting specific physical memory with [l4sigma0\\_map\\_mem\(\)](#).

See [l4sigma0\\_map\\_errstr\(\)](#) to get a description of the return value.

References [L4\\_CV](#).

#### 14.13.3.3 l4sigma0\_map\_errstr()

```
char const * l4sigma0_map_errstr (
    int err) [inline]
```

Get user readable error messages for the return codes.

### Parameters

<i>err</i>	The error code reported by the <i>map</i> functions.
------------	------------------------------------------------------

### Returns

A string containing the error message.

Definition at line 206 of file [sigma0.h](#).

References [L4\\_INLINE](#).

#### 14.13.3.4 l4sigma0\_map\_iomem()

```
int l4sigma0_map_iomem (
    l4_cap_idx_t sigma0,
    l4_addr_t phys,
    l4_addr_t virt,
    l4_addr_t size,
    int cached)
```

Request IO memory from sigma0.

### Parameters

<i>sigma0</i>	Capability selector for the sigma0 gate.
<i>phys</i>	The physical address to be requested (page aligned).
<i>virt</i>	The virtual address where the memory should be mapped to (page aligned).
<i>size</i>	The size of the IO memory area to be mapped (multiple of page size)
<i>cached</i>	Requests cacheable IO memory if 1 and uncached if 0.

### Return values

0	Success.
-L4SIGMA0_NOTALIGNED	<i>phys</i> , <i>virt</i> , or <i>size</i> are not aligned.
-L4SIGMA0_IPCERROR	IPC error.
-L4SIGMA0_NOFPAGE	No fpage received.

This function is similar to [l4sigma0\\_map\\_mem\(\)](#), the difference is that it requests IO memory. IO memory is everything that is not known to be normal RAM. Also ACPI tables or the BIOS memory is treated as IO memory.

See [l4sigma0\\_map\\_errstr\(\)](#) to get a description of the return value.

References [L4\\_CV](#).

#### 14.13.3.5 l4sigma0\_map\_kip()

```
L4_BEGIN_DECLS l4_kernel_info_t * l4sigma0_map_kip (
    l4_cap_idx_t sigma0,
    void * addr,
    unsigned log2_size)
```

Map the kernel info page from sigma0 to addr.

### Parameters

<i>sigma0</i>	Capability selector for the sigma0 gate.
<i>addr</i>	Start of the receive window to receive KIP in.
<i>log2_size</i>	Size of the receive window to receive KIP in.

### Returns

Address KIP was mapped to, 0 indicates an error.

#### 14.13.3.6 l4sigma0\_map\_mem()

```
int l4sigma0_map_mem (
    l4_cap_idx_t sigma0,
    l4_addr_t phys,
    l4_addr_t virt,
    l4_addr_t size)
```

Request a memory mapping from sigma0.

### Parameters

<i>sigma0</i>	Capability selector for the sigma0 gate.
<i>phys</i>	The physical address of the requested page (must be at least aligned to the minimum page size).
<i>virt</i>	The virtual address where the paged should be mapped in the local address space (must be at least aligned to the minimum page size).
<i>size</i>	The size of the requested page, this must be a multiple of the minimum page size.

### Return values

<i>0</i>	Success.
<i>-L4SIGMA0_NOTALIGNED</i>	<i>phys</i> , <i>virt</i> , or <i>size</i> are not aligned.
<i>-L4SIGMA0_IPCERROR</i>	IPC error.
<i>-L4SIGMA0_NOFPAGE</i>	No fpage received.

This function only maps normal RAM. To map other memory, use [l4sigma0\\_map\\_iomem\(\)](#). See also there for the distinction between both memory types.

This is the direct method to request memory from sigma0. There is also the indirect method where sigma0 will answer page faults with a mapping that is one-to-one between the faulting virtual page and the backing physical page. See [L4::Pager::page\\_fault\(\)](#). For an overview of the memory hierarchy, see [Memory management - Data Spaces and the Region Map](#).

See [l4sigma0\\_map\\_errstr\(\)](#) to get a description of the return value.

References [L4\\_CV](#).

### 14.13.4 Internal constants

Internal sigma0 definitions.

Collaboration diagram for Internal constants:



### Macros

- `#define SIGMA0_REQ_MAGIC ~0xFFUL`  
*Request magic.*
- `#define SIGMA0_REQ_MASK ~0xFFUL`  
*Request mask.*
- `#define SIGMA0_REQ_ID_MASK 0xF0`

- ID mask.*
- `#define SIGMA0_REQ_ID_FPAGE_RAM 0x60`
- RAM.*
- `#define SIGMA0_REQ_ID_FPAGE_IOMEM 0x70`
- I/O memory.*
- `#define SIGMA0_REQ_ID_FPAGE_IOMEM_CACHED 0x80`
- Cached I/O memory.*
- `#define SIGMA0_REQ_ID_FPAGE_ANY 0x90`
- Any.*
- `#define SIGMA0_REQ_ID_KIP 0xA0`
- KIP.*
- `#define SIGMA0_REQ_ID_DEBUG_DUMP 0xC0`
- Debug dump.*
- `#define SIGMA0_IS_MAGIC_REQ(d1)`
- Check if magic.*
- `#define SIGMA0_REQ(x)`
- Construct.*
- `#define SIGMA0_REQ_FPAGE_RAM (SIGMA0_REQ(FPAGE_RAM))`
- RAM.*
- `#define SIGMA0_REQ_FPAGE_IOMEM (SIGMA0_REQ(FPAGE_IOMEM))`
- I/O memory.*
- `#define SIGMA0_REQ_FPAGE_IOMEM_CACHED (SIGMA0_REQ(FPAGE_IOMEM_CACHED))`
- Cache I/O memory.*
- `#define SIGMA0_REQ_FPAGE_ANY (SIGMA0_REQ(FPAGE_ANY))`
- Any.*
- `#define SIGMA0_REQ_KIP (SIGMA0_REQ(KIP))`
- KIP.*
- `#define SIGMA0_REQ_DEBUG_DUMP (SIGMA0_REQ(DEBUG_DUMP))`
- Debug dump.*

#### 14.13.4.1 Detailed Description

Internal sigma0 definitions.

## 14.14 Small C++ Template Library

### Namespaces

- namespace `cxx`
- Our C++ library.*

## Data Structures

- class [L4::Alloc\\_list](#)  
*A simple list-based allocator.*
- class [cxx::List\\_item](#)  
*Basic list item.*
- struct [cxx::Pair< First, Second >](#)  
*Pair of two values.*
- class [cxx::Base\\_slab< Obj\\_size, Slab\\_size, Max\\_free, Alloc >](#)  
*Basic slab allocator.*
- class [cxx::Slab< Type, Slab\\_size, Max\\_free, Alloc >](#)  
*Slab allocator for object of type `Type`.*
- class [cxx::Base\\_slab\\_static< Obj\\_size, Slab\\_size, Max\\_free, Alloc >](#)  
*Merged slab allocator (allocators for objects of the same size are merged together).*
- class [cxx::Slab\\_static< Type, Slab\\_size, Max\\_free, Alloc >](#)  
*Merged slab allocator (allocators for objects of the same size are merged together).*
- class [cxx::Nothrow](#)  
*Helper type to distinguish the `operator new` version that does not throw exceptions.*
- class [cxx::New\\_allocator< \\_Type >](#)  
*Standard allocator based on `operator new ()`.*
- class [L4::String](#)  
*A null-terminated string container class.*

## Functions

- `template<typename A, typename ... ARGS>`  
`constexpr A const & cxx::min (A const &a1, A const &a2, ARGS const &...a)`  
*Get the minimum of `a1` and `a2` up to `aN`.*
- `template<typename A, typename ... ARGS>`  
`constexpr A const & cxx::min (cxx::identity_t< A > const &a1, cxx::identity_t< A > const &a2, ARGS const &...a)`  
*Get the minimum of `a1` and `a2` up to `aN`.*
- `template<typename A, typename ... ARGS>`  
`constexpr A const & cxx::max (A const &a1, A const &a2, ARGS const &...a)`  
*Get the maximum of `a1` and `a2` up to `aN`.*
- `template<typename A, typename ... ARGS>`  
`constexpr A const & cxx::max (cxx::identity_t< A > const &a1, cxx::identity_t< A > const &a2, ARGS const &...a)`  
*Get the maximum of `a1` and `a2` up to `aN`.*
- `template<typename T1>`  
`T1 cxx::clamp (T1 v, T1 lo, T1 hi)`  
*Limit `v` to the range given by `lo` and `hi`.*
- `void * operator new (size_t, void *mem, cxx::Nothrow const &) noexcept`  
*Simple placement new operator.*
- `void * operator new (size_t, cxx::Nothrow const &) noexcept`  
*New operator that does not throw exceptions.*
- `void operator delete (void *, cxx::Nothrow const &) noexcept`  
*Delete operator complementing the new operator not throwing exceptions.*

### 14.14.1 Detailed Description

### 14.14.2 Function Documentation

#### 14.14.2.1 clamp()

```
template<typename T1>
T1 cxx::clamp (
    T1 v,
    T1 lo,
    T1 hi) [inline]
```

Limit *v* to the range given by *lo* and *hi*.

#### Parameters

<i>v</i>	The value to clamp.
<i>lo</i>	The lower boundary to clamp <i>v</i> to.
<i>hi</i>	The upper boundary to clamp <i>v</i> to.

Definition at line 109 of file [minmax](#).

#### 14.14.2.2 max() [1/2]

```
template<typename A, typename ... ARGS>
A const & cxx::max (
    A const & a1,
    A const & a2,
    ARGS const &... a) [constexpr]
```

Get the maximum of *a1* and *a2* up to *aN*.

#### Parameters

<i>a1</i>	The first value.
<i>a2</i>	The second value.
<i>...↔ a</i>	Arbitrary number of additional parameters.

Matches with automatic argument type deduction.

Definition at line 78 of file [minmax](#).

#### 14.14.2.3 max() [2/2]

```
template<typename A, typename ... ARGS>
A const & cxx::max (
    cxx::identity_t< A > const & a1,
    cxx::identity_t< A > const & a2,
    ARGS const &... a) [constexpr]
```

Get the maximum of *a1* and *a2* up to *aN*.

#### Parameters

<i>a1</i>	The first value.
<i>a2</i>	The second value.
<i>...↔ a</i>	Arbitrary number of additional parameters.

Matches with explicit template type A.

Definition at line 93 of file [minmax](#).

#### 14.14.2.4 min() [1/2]

```
template<typename A, typename ...  ARGS>
A const & cxx::min (
    A const & a1,
    A const & a2,
    ARGS const &...  a) [constexpr]
```

Get the minimum of a1 and a2 upt to aN.

##### Parameters

<i>a1</i>	The first value.
<i>a2</i>	The second value.
<i>...↔ a</i>	Arbitrary number of additional parameters.

Matches with automatic argument type deduction.

Definition at line 36 of file [minmax](#).

#### 14.14.2.5 min() [2/2]

```
template<typename A, typename ...  ARGS>
A const & cxx::min (
    cxx::identity_t< A > const & a1,
    cxx::identity_t< A > const & a2,
    ARGS const &...  a) [constexpr]
```

Get the minimum of a1 and a2 upt to aN.

##### Parameters

<i>a1</i>	The first value.
<i>a2</i>	The second value.
<i>...↔ a</i>	Arbitrary number of additional parameters.

Matches with explicit template type A.

Definition at line 53 of file [minmax](#).

### 14.14.2.6 operator new()

```
void * operator new (
    size_t ,
    void * mem,
    cxx::Nothrow const & ) [inline], [noexcept]
```

Simple placement new operator.

#### Parameters

<i>mem</i>	the address of the memory block to place the new object.
------------	----------------------------------------------------------

#### Returns

the address given by *mem*.

Definition at line 28 of file [std\\_alloc](#).

## 14.15 The L4Re IPC Framework

The mechanisms for IPC communication between [L4Re](#) applications.

Collaboration diagram for The L4Re IPC Framework:



#### Topics

- [Server-Side IPC framework](#) . . . . . [649](#)  
*Server-Side framework for implementing object-oriented servers.*

### 14.15.1 Detailed Description

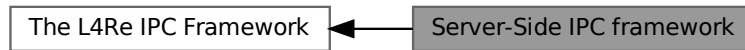
The mechanisms for IPC communication between [L4Re](#) applications.



## 14.15.2 Server-Side IPC framework

Server-Side framework for implementing object-oriented servers.

Collaboration diagram for Server-Side IPC framework:



### Namespaces

- namespace `L4::lpc_svr`  
*Helper classes for `L4::Server` instantiation.*

### Data Structures

- class `L4::lpc_svr::Server_iface`  
*Interface for server-loop related functions.*
- class `L4::Basic_registry`  
*This registry returns the corresponding server object based on the label of an `lpc_gate`.*
- struct `L4::lpc_svr::Ignore_errors`  
*Mix in for `LOOP_HOOKS` to ignore IPC errors.*
- struct `L4::lpc_svr::Default_timeout`  
*Mix in for `LOOP_HOOKS` to use a 0 send and a infinite receive timeout.*
- struct `L4::lpc_svr::Compound_reply`  
*Mix in for `LOOP_HOOKS` to always use compound reply and wait.*
- struct `L4::lpc_svr::Default_setup_wait`  
*Mix in for `LOOP_HOOKS` for setup\_wait no op.*
- class `L4::lpc_svr::Br_manager_no_buffers`  
*Empty implementation of `Server_iface`.*
- struct `L4::lpc_svr::Default_loop_hooks`  
*Default `LOOP_HOOKS`.*
- class `L4::Server< LOOP_HOOKS >`  
*Basic server loop for handling client requests.*
- class `L4::Server_object`  
*Abstract server object to be used with `L4::Server` and `L4::Basic_registry`.*
- struct `L4::Server_object_t< IFACE, BASE >`  
*Base class (template) for server implementing server objects.*
- struct `L4::Server_object_x< Derived, IFACE, BASE >`  
*Helper class to implement p\_dispatch based server objects.*
- struct `L4::lrq_handler_object`  
*Server object base class for handling IRQ messages.*
- class `L4::lpc_svr::Timeout`  
*Callback interface for `Timeout_queue`.*
- class `L4::lpc_svr::Timeout_queue`  
*Timeout queue to be used in l4re server loop.*
- class `L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >`  
*Loop hooks mixin for integrating a timeout queue into the server loop.*

## Enumerations

- enum [L4::lpc\\_svr::Reply\\_mode](#) { [L4::lpc\\_svr::Reply\\_compound](#) , [L4::lpc\\_svr::Reply\\_separate](#) }  
*Reply mode for server loop.*

### 14.15.2.1 Detailed Description

Server-Side framework for implementing object-oriented servers.

,

### 14.15.2.2 Enumeration Type Documentation

#### 14.15.2.2.1 Reply\_mode

enum [L4::Ipc\\_svr::Reply\\_mode](#)

Reply mode for server loop.

The reply mode specifies if the server loop shall do a compound reply and wait operation ([Reply\\_compound](#)), which is the most performant method. Note, `setup_wait()` is called before the reply. The other way is to call reply and wait separately and call `setup_wait` in between.

The actual mode is determined by the return value of the `before_reply()` hook in the `LOOP_HOOKS` of [L4::Server](#).

#### Enumerator

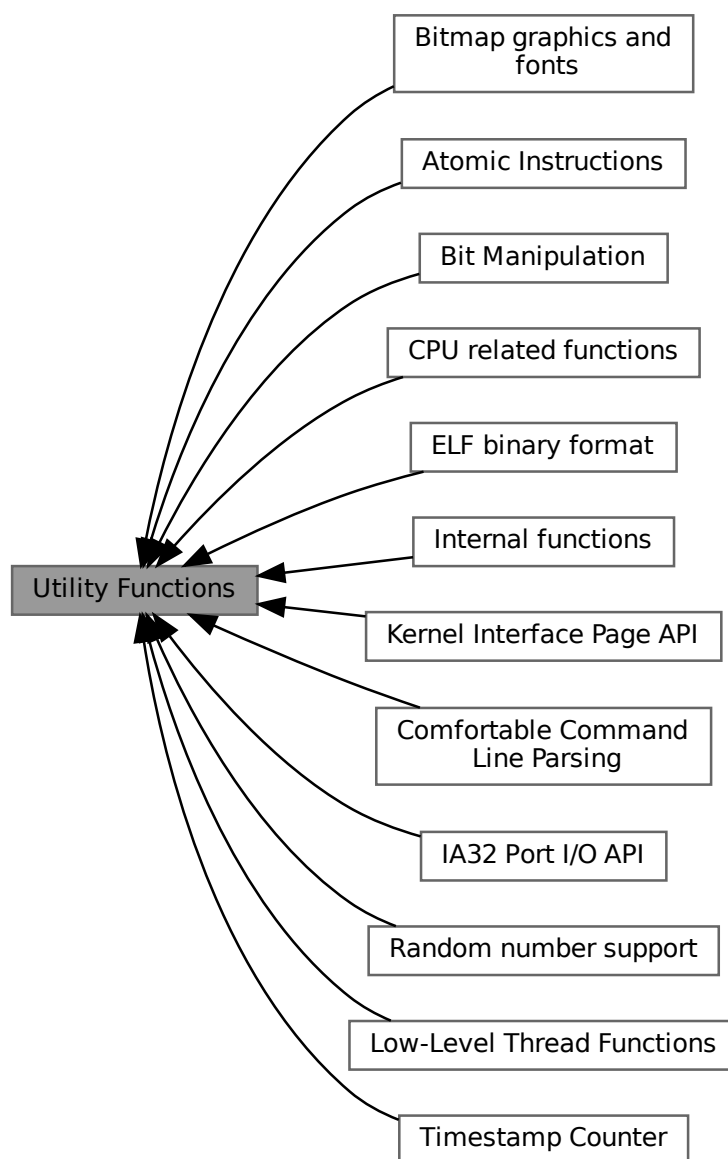
<code>Reply_compound</code>	<a href="#">Server</a> shall use a compound reply and wait (fast).
<code>Reply_separate</code>	<a href="#">Server</a> shall call reply and wait separately.

Definition at line 46 of file [ipc\\_server\\_loop](#).

## 14.16 Utility Functions

Utilities, generic file.

Collaboration diagram for Utility Functions:



## Topics

- [Bitmap graphics and fonts](#) . . . . . [657](#)  
*This library provides some functions for bitmap handling in frame buffers.*
- [CPU related functions](#) . . . . . [658](#)
- [Timestamp Counter](#) . . . . . [660](#)

•	Atomic Instructions . . . . .	667
•	Internal functions . . . . .	685
•	Bit Manipulation . . . . .	685
•	ELF binary format . . . . .	693
	<i>Functions and types related to ELF binaries.</i>	
•	Kernel Interface Page API . . . . .	719
•	Comfortable Command Line Parsing . . . . .	721
•	Random number support . . . . .	724
•	Low-Level Thread Functions . . . . .	725
•	IA32 Port I/O API . . . . .	725

## Files

- file [rand.h](#)  
*Simple Pseudo-Random Number Generator.*

## Functions

- [L4\\_BEGIN\\_DECLS](#) long [l4util\\_splitlog2\\_hdl](#) ([l4\\_addr\\_t](#) start, [l4\\_addr\\_t](#) end, long(\*handler)([l4\\_addr\\_t](#) s, [l4\\_addr\\_t](#) e, int log2size))  
*Split a range into log2 base and size aligned chunks.*
- [l4\\_addr\\_t](#) [l4util\\_splitlog2\\_size](#) ([l4\\_addr\\_t](#) start, [l4\\_addr\\_t](#) end)  
*Return log2 base and size aligned length of a range.*
- [L4\\_BEGIN\\_DECLS](#) [l4\\_timeout\\_s](#) [l4util\\_micros2l4to](#) ([l4\\_uint64\\_t](#) us) [L4\\_NOTHROW](#)  
*Calculate l4 timeouts.*
- void [l4\\_sleep](#) ([l4\\_uint32\\_t](#) ms) [L4\\_NOTHROW](#)  
*Suspend thread for a period of ms milliseconds.*
- void [l4\\_usleep](#) ([l4\\_uint64\\_t](#) us) [L4\\_NOTHROW](#)  
*Suspend thread for a period of us microseconds.*
- void [l4\\_sleep\\_forever](#) (void) [L4\\_NOTHROW](#) [L4\\_NORETURN](#)  
*Go sleep and never wake up.*
- void [l4\\_touch\\_ro](#) (const void \*addr, unsigned size) [L4\\_NOTHROW](#)  
*Touch data area to force mapping (read-only).*
- void [l4\\_touch\\_rw](#) (const void \*addr, unsigned size) [L4\\_NOTHROW](#)  
*Touch data areas to force mapping (read-write).*

### 14.16.1 Detailed Description

Utilities, generic file.

## 14.16.2 Function Documentation

### 14.16.2.1 l4\_sleep()

```
void l4_sleep (
    l4_uint32_t ms)
```

Suspend thread for a period of *ms* milliseconds.

#### Parameters

<i>ms</i>	Time in milliseconds
-----------	----------------------

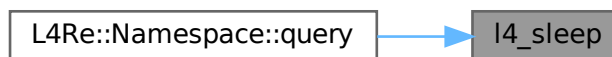
#### Examples

[examples/libs/libirq/async\\_isr.c](#), [examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), and [examples/sys/start-with-exc/r](#)

References [L4\\_CV](#), and [L4\\_NOTHROW](#).

Referenced by [L4Re::Namespace::query\(\)](#).

Here is the caller graph for this function:



### 14.16.2.2 l4\_touch\_ro()

```
void l4_touch_ro (
    const void * addr,
    unsigned size) [inline]
```

Touch data area to force mapping (read-only).

#### Parameters

<i>addr</i>	Start of memory area to touch.
<i>size</i>	Size of area to touch.

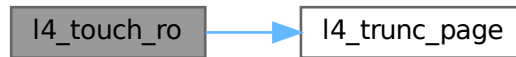
#### Examples

[examples/sys/singlestep/main.c](#).

Definition at line 92 of file [util.h](#).

References [L4\\_NOTHROW](#), [L4\\_PAGESIZE](#), and [l4\\_trunc\\_page\(\)](#).

Here is the call graph for this function:



### 14.16.2.3 l4\_touch\_rw()

```
void l4_touch_rw (
    const void * addr,
    unsigned size) [inline]
```

Touch data areas to force mapping (read-write).

#### Parameters

<i>addr</i>	Start of memory area to touch.
<i>size</i>	Size of area to touch.

#### Examples

[examples/sys/aliens/main.c](#), and [examples/sys/singlestep/main.c](#).

Definition at line 105 of file [util.h](#).

References [L4\\_NOTHROW](#), [L4\\_PAGESIZE](#), and [l4\\_trunc\\_page\(\)](#).

Here is the call graph for this function:



### 14.16.2.4 l4\_usleep()

```
void l4_usleep (
    l4_uint64_t us)
```

Suspend thread for a period of *us* microseconds.

#### Parameters

<i>us</i>	Time in microseconds
-----------	----------------------

**Note**

The timer resolution of [L4](#) kernels is usually 1ms.

References [L4\\_CV](#), [L4\\_INLINE](#), [L4\\_NORETURN](#), and [L4\\_NOTHROW](#).

**14.16.2.5 l4util\_micros2l4to()**

```
L4_BEGIN_DECLS l4_timeout_s l4util_micros2l4to (  
    l4_uint64_t us)
```

Calculate l4 timeouts.

**Parameters**

<i>us</i>	time in microseconds. Special cases: <ul style="list-style-type: none"><li>• 0 -&gt; timeout 0</li><li>• ~0U -&gt; timeout NEVER</li></ul>
-----------	--------------------------------------------------------------------------------------------------------------------------------------------

**Returns**

the corresponding l4\_timeout value

**Deprecated** Use [l4\\_timeout\\_from\\_us\(\)](#).

References [L4\\_CV](#), and [L4\\_NOTHROW](#).

**14.16.2.6 l4util\_splitlog2\_hdl()**

```
L4_END_DECLS long l4util_splitlog2_hdl (  
    l4_addr_t start,  
    l4_addr_t end,  
    long(* handler )(l4_addr_t s, l4_addr_t e, int log2size)) [inline]
```

Split a range into log2 base and size aligned chunks.

**Parameters**

<i>start</i>	Start of range
<i>end</i>	End of range (inclusive) (e.g. 2-4 is len 3)

<i>handler</i>	Handler function that is called with start and end (both inclusive) of the chunk. On success, the handler must return 0, if it returns !=0 the function will immediately return with the return code of the handler.
----------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

#### Returns

0 on success, != 0 otherwise

Definition at line 51 of file [splitlog2.h](#).

References [L4\\_EINVAL](#), and [l4util\\_splitlog2\\_size\(\)](#).

Here is the call graph for this function:



#### 14.16.2.7 l4util\_splitlog2\_size()

```

l4_addr_t l4util_splitlog2_size (
    l4_addr_t start,
    l4_addr_t end) [inline]
  
```

Return log2 base and size aligned length of a range.

#### Parameters

<i>start</i>	Start of range
<i>end</i>	End of range (inclusive) (e.g. 2-4 is len 3)

#### Returns

length of elements in log2size (length is  $1 \ll \log_2 \text{size}$ )

Definition at line 70 of file [splitlog2.h](#).

Referenced by [l4util\\_splitlog2\\_hdl\(\)](#).

Here is the caller graph for this function:

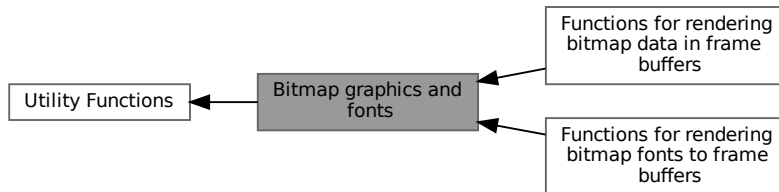




### 14.16.3 Bitmap graphics and fonts

This library provides some functions for bitmap handling in frame buffers.

Collaboration diagram for Bitmap graphics and fonts:



#### Topics

- Functions for rendering bitmap data in frame buffers . . . . . [657](#)
- Functions for rendering bitmap fonts to frame buffers . . . . . [658](#)

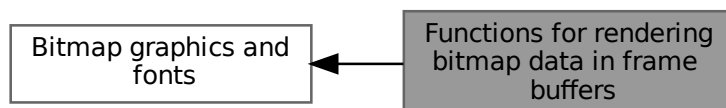
#### 14.16.3.1 Detailed Description

This library provides some functions for bitmap handling in frame buffers.

Includes simple functions like filling or copying an area of the frame buffer going up to rendering text into the frame buffer using bitmap fonts.

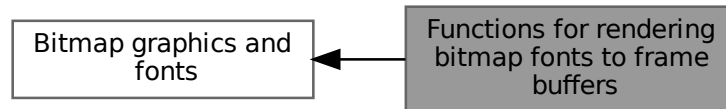
#### 14.16.3.2 Functions for rendering bitmap data in frame buffers

Collaboration diagram for Functions for rendering bitmap data in frame buffers:



### 14.16.3.3 Functions for rendering bitmap fonts to frame buffers

Collaboration diagram for Functions for rendering bitmap fonts to frame buffers:



### 14.16.4 CPU related functions

Collaboration diagram for CPU related functions:



#### Functions

- int [l4util\\_cpu\\_has\\_cpuid](#) (void)  
*Check whether the CPU supports the "cpuid" instruction.*
- unsigned int [l4util\\_cpu\\_capabilities](#) (void)  
*Returns the CPU capabilities if the "cpuid" instruction is available.*
- unsigned int [l4util\\_cpu\\_capabilities\\_nocheck](#) (void)  
*Returns the CPU capabilities.*
- void [l4util\\_cpu\\_cpuid](#) (unsigned long mode, unsigned long \*eax, unsigned long \*ebx, unsigned long \*ecx, unsigned long \*edx)  
*Generic CPUID access function.*

#### 14.16.4.1 Detailed Description

#### 14.16.4.2 Function Documentation

##### 14.16.4.2.1 l4util\_cpu\_capabilities()

```

unsigned int l4util_cpu_capabilities (
    void ) [inline]
  
```

Returns the CPU capabilities if the "cpuid" instruction is available.

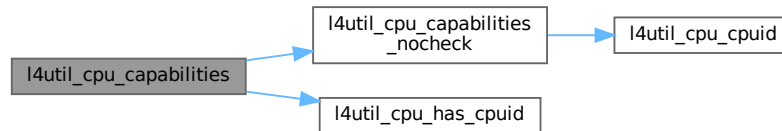
**Returns**

CPU capabilities if the "cpuid" instruction is available, 0 if the "cpuid" instruction is not supported.

Definition at line 95 of file [cpu.h](#).

References [l4util\\_cpu\\_capabilities\\_nocheck\(\)](#), and [l4util\\_cpu\\_has\\_cpuid\(\)](#).

Here is the call graph for this function:

**14.16.4.2.2 l4util\_cpu\_capabilities\_nocheck()**

```
unsigned int l4util_cpu_capabilities_nocheck (
    void ) [inline]
```

Returns the CPU capabilities.

**Returns**

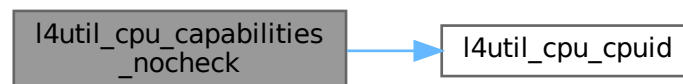
CPU capabilities.

Definition at line 84 of file [cpu.h](#).

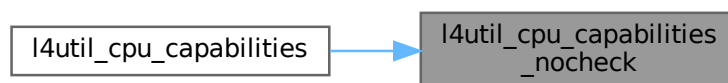
References [l4util\\_cpu\\_cpuid\(\)](#).

Referenced by [l4util\\_cpu\\_capabilities\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.16.4.2.3 l4util\_cpu\_has\_cpuid()

```
int l4util_cpu_has_cpuid (  
    void ) [inline]
```

Check whether the CPU supports the "cpuid" instruction.

##### Returns

1 if it has, 0 if it has not

Definition at line 64 of file [cpu.h](#).

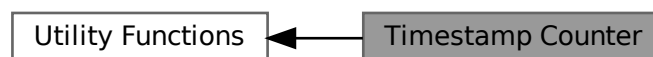
Referenced by [l4util\\_cpu\\_capabilities\(\)](#).

Here is the caller graph for this function:



### 14.16.5 Timestamp Counter

Collaboration diagram for Timestamp Counter:



##### Files

- file [rdtsc.h](#)  
*Timestamp counter related functions.*
- file [rdtsc.h](#)  
*Timestamp counter related functions.*

## Functions

- [l4\\_cpu\\_time\\_t l4\\_rdtsc](#) (void)  
*Read current value of CPU-internal timestamp counter.*
- [l4\\_uint32\\_t l4\\_rdtsc\\_32](#) (void)  
*Read the least significant 32 bit of the TSC.*
- [l4\\_uint64\\_t l4\\_rdpmc](#) (int ecx)  
*Return current value of CPU-internal performance measurement counter.*
- [l4\\_uint32\\_t l4\\_rdpmc\\_32](#) (int ecx)  
*Return the least significant 32 bit of a performance counter.*
- [l4\\_uint64\\_t l4\\_tsc\\_to\\_ns](#) ([l4\\_cpu\\_time\\_t](#) tsc)  
*Convert timestamp to ns value.*
- [l4\\_uint64\\_t l4\\_tsc\\_to\\_us](#) ([l4\\_cpu\\_time\\_t](#) tsc)  
*Convert timestamp into micro seconds value.*
- void [l4\\_tsc\\_to\\_s\\_and\\_ns](#) ([l4\\_cpu\\_time\\_t](#) tsc, [l4\\_uint32\\_t](#) \*s, [l4\\_uint32\\_t](#) \*ns)  
*Convert timestamp to s.ns value.*
- [l4\\_cpu\\_time\\_t l4\\_ns\\_to\\_tsc](#) ([l4\\_uint64\\_t](#) ns)  
*Convert nano seconds into CPU ticks.*
- void [l4\\_busy\\_wait\\_ns](#) ([l4\\_uint64\\_t](#) ns)  
*Wait busy for a small amount of time.*
- void [l4\\_busy\\_wait\\_us](#) ([l4\\_uint64\\_t](#) us)  
*Wait busy for a small amount of time.*
- [l4\\_uint32\\_t l4\\_calibrate\\_tsc](#) ([l4\\_kernel\\_info\\_t](#) const \*kip)  
*Determine scalers for timestamp calculations.*
- [l4\\_uint32\\_t l4\\_tsc\\_init](#) ([l4\\_kernel\\_info\\_t](#) const \*kip)  
*Initialize scaler for TSC calibrations from the kernel.*
- [l4\\_uint32\\_t l4\\_get\\_hz](#) (void)  
*Get CPU frequency in Hz.*

### 14.16.5.1 Detailed Description

### 14.16.5.2 Function Documentation

#### 14.16.5.2.1 l4\_busy\_wait\_ns()

```
void l4_busy_wait_ns (
    l4\_uint64\_t ns) [inline]
```

Wait busy for a small amount of time.

#### Parameters

<i>ns</i>	nano seconds to wait
-----------	----------------------

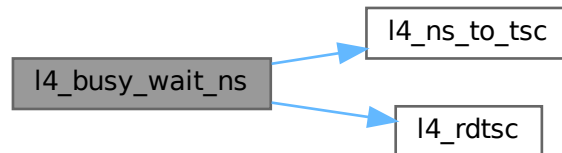
**Attention**

Not intended for any use!

Definition at line 262 of file [rdtsc.h](#).

References [l4\\_ns\\_to\\_tsc\(\)](#), and [l4\\_rdtsc\(\)](#).

Here is the call graph for this function:

**14.16.5.2.2 l4\_busy\_wait\_us()**

```
void l4_busy_wait_us (
    l4_uint64_t us) [inline]
```

Wait busy for a small amount of time.

**Parameters**

<i>us</i>	micro seconds to wait
-----------	-----------------------

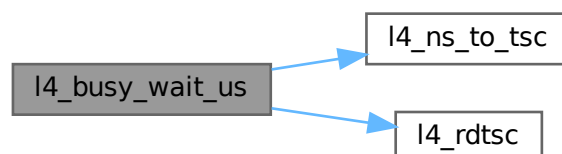
**Attention**

Not intended for any use!

Definition at line 272 of file [rdtsc.h](#).

References [l4\\_ns\\_to\\_tsc\(\)](#), and [l4\\_rdtsc\(\)](#).

Here is the call graph for this function:



#### 14.16.5.2.3 l4\_calibrate\_tsc()

```
l4_uint32_t l4_calibrate_tsc (
    l4_kernel_info_t const * kip) [inline]
```

Determine scalars for timestamp calculations.

Determine some scalars to be able to convert between real time and CPU ticks. Just calls [l4\\_tsc\\_init\(\)](#).

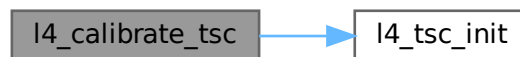
##### Examples

[examples/sys/aliens/main.c](#).

Definition at line 159 of file [rdtsc.h](#).

References [l4\\_tsc\\_init\(\)](#).

Here is the call graph for this function:



#### 14.16.5.2.4 l4\_get\_hz()

```
l4_uint32_t l4_get_hz (
    void )
```

Get CPU frequency in Hz.

##### Returns

frequency in Hz

References [L4\\_END\\_DECLS](#), and [L4\\_INLINE](#).

#### 14.16.5.2.5 l4\_ns\_to\_tsc()

```
l4_cpu_time_t l4_ns_to_tsc (
    l4_uint64_t ns) [inline]
```

Convert nano seconds into CPU ticks.

##### Parameters

<i>ns</i>	nano seconds
-----------	--------------

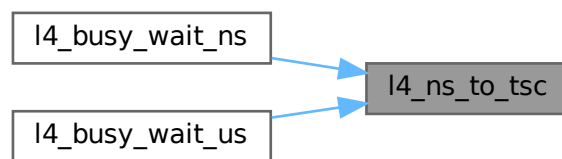
**Returns**

CPU ticks

Definition at line 248 of file [rdtsc.h](#).

Referenced by [l4\\_busy\\_wait\\_ns\(\)](#), and [l4\\_busy\\_wait\\_us\(\)](#).

Here is the caller graph for this function:

**14.16.5.2.6 l4\_rdpmc()**

```
l4_uint64_t l4_rdpmc (
    int ecx) [inline]
```

Return current value of CPU-internal performance measurement counter.

**Parameters**

<i>ecx</i>	ECX value for the rdpmc instruction. For details see the Intel IA-32 Architectures Software Developer's Manual.
------------	-----------------------------------------------------------------------------------------------------------------

**Returns**

64-bit PMC

Definition at line 175 of file [rdtsc.h](#).

**14.16.5.2.7 l4\_rdpmc\_32()**

```
l4_uint32_t l4_rdpmc_32 (
    int ecx) [inline]
```

Return the least significant 32 bit of a performance counter.

Useful for smaller differences, needs less cycles.

Definition at line 195 of file [rdtsc.h](#).



#### 14.16.5.2.8 l4\_rdtsc()

```
l4_cpu_time_t l4_rdtsc (  
    void ) [inline]
```

Read current value of CPU-internal timestamp counter.

##### Returns

64-bit timestamp

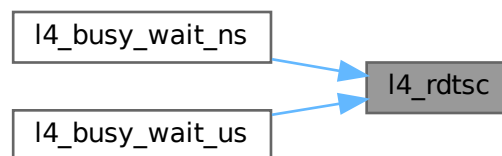
##### Examples

[examples/sys/aliens/main.c](#).

Definition at line 165 of file [rdtsc.h](#).

Referenced by [l4\\_busy\\_wait\\_ns\(\)](#), and [l4\\_busy\\_wait\\_us\(\)](#).

Here is the caller graph for this function:



#### 14.16.5.2.9 l4\_rdtsc\_32()

```
l4_uint32_t l4_rdtsc_32 (  
    void ) [inline]
```

Read the least significant 32 bit of the TSC.

Useful for smaller differences, needs less cycles.

Definition at line 185 of file [rdtsc.h](#).

#### 14.16.5.2.10 l4\_tsc\_init()

```
l4_uint32_t l4_tsc_init (  
    l4_kernel_info_t const * kip)
```

Initialize scaler for TSC calibrations from the kernel.

Initialize the scalers needed by [l4\\_tsc\\_to\\_ns\(\)](#)/[l4\\_ns\\_to\\_tsc\(\)](#) and so on. Use the kernel-provided frequency.

##### Parameters

<i>kip</i>	KIP pointer
------------	-------------

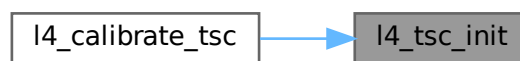
#### Returns

0 on error (no scalars exported by kernel) otherwise returns ( $2^{32} / (\text{tsc per } \mu\text{sec})$ ). This value has the same semantics as the value returned by the `calibrate_delay_loop()` function of the Linux kernel.

References [L4\\_CV](#).

Referenced by [l4\\_calibrate\\_tsc\(\)](#).

Here is the caller graph for this function:



#### 14.16.5.2.11 l4\_tsc\_to\_ns()

```
l4_uint64_t l4_tsc_to_ns (
    l4_cpu_time_t tsc) [inline]
```

Convert timestamp to ns value.

#### Parameters

<i>tsc</i>	time value in CPU ticks
------------	-------------------------

#### Returns

time value in ns

#### Examples

[examples/sys/aliens/main.c](#).

Definition at line 205 of file [rdtsc.h](#).

#### 14.16.5.2.12 l4\_tsc\_to\_s\_and\_ns()

```
void l4_tsc_to_s_and_ns (
    l4_cpu_time_t tsc,
    l4_uint32_t * s,
    l4_uint32_t * ns) [inline]
```

Convert timestamp to s.ns value.

#### Parameters

	<i>tsc</i>	time value in CPU ticks
out	<i>s</i>	seconds
out	<i>ns</i>	nano seconds

Definition at line 233 of file [rdtsc.h](#).

#### 14.16.5.2.13 l4\_tsc\_to\_us()

```
l4_uint64_t l4_tsc_to_us (  
    l4_cpu_time_t tsc) [inline]
```

Convert timestamp into micro seconds value.

##### Parameters

<i>tsc</i>	time value in CPU ticks
------------	-------------------------

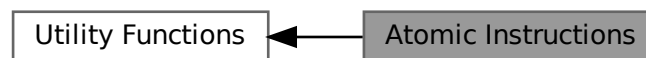
##### Returns

time value in micro seconds

Definition at line 219 of file [rdtsc.h](#).

## 14.16.6 Atomic Instructions

Collaboration diagram for Atomic Instructions:



##### Files

- file [atomic.h](#)  
*atomic operations header and generic implementations*

## Functions

- `int l4util_cmpxchg32` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` cmp\_val, `l4_uint32_t` new\_val)  
*Atomic compare and exchange (32 bit version).*
- `int l4util_cmpxchg16` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` cmp\_val, `l4_uint16_t` new\_val)  
*Atomic compare and exchange (16 bit version).*
- `int l4util_cmpxchg8` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` cmp\_val, `l4_uint8_t` new\_val)  
*Atomic compare and exchange (8 bit version).*
- `int l4util_cmpxchg` (volatile `l4_umword_t` \*dest, `l4_umword_t` cmp\_val, `l4_umword_t` new\_val)  
*Atomic compare and exchange (machine wide fields).*
- `l4_uint32_t l4util_xchg32` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` val)  
*Atomic exchange (32 bit version).*
- `l4_uint16_t l4util_xchg16` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)  
*Atomic exchange (16 bit version).*
- `l4_uint8_t l4util_xchg8` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)  
*Atomic exchange (8 bit version).*
- `l4_umword_t l4util_xchg` (volatile `l4_umword_t` \*dest, `l4_umword_t` val)  
*Atomic exchange (machine wide fields).*
- `void l4util_atomic_add` (volatile long \*dest, long val)  
*Atomic add.*
- `void l4util_atomic_inc` (volatile long \*dest)  
*Atomic increment.*

### Atomic add/sub/and/or (8,16,32 bit version) without result

- `void l4util_add8` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)
- `void l4util_add16` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)
- `void l4util_add32` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` val)
- `void l4util_sub8` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)
- `void l4util_sub16` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)
- `void l4util_sub32` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` val)
- `void l4util_and8` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)
- `void l4util_and16` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)
- `void l4util_and32` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` val)
- `void l4util_or8` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)
- `void l4util_or16` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)
- `void l4util_or32` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` val)

### Atomic add/sub/and/or operations (8,16,32 bit) with result

- `l4_uint8_t l4util_add8_res` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)
- `l4_uint16_t l4util_add16_res` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)
- `l4_uint32_t l4util_add32_res` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` val)
- `l4_uint8_t l4util_sub8_res` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)
- `l4_uint16_t l4util_sub16_res` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)
- `l4_uint32_t l4util_sub32_res` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` val)
- `l4_uint8_t l4util_and8_res` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)
- `l4_uint16_t l4util_and16_res` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)
- `l4_uint32_t l4util_and32_res` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` val)
- `l4_uint8_t l4util_or8_res` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)
- `l4_uint16_t l4util_or16_res` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)
- `l4_uint32_t l4util_or32_res` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` val)

**Atomic inc/dec (8,16,32 bit) without result**

- void [l4util\\_inc8](#) (volatile [l4\\_uint8\\_t](#) \*dest)
- void [l4util\\_inc16](#) (volatile [l4\\_uint16\\_t](#) \*dest)
- void [l4util\\_inc32](#) (volatile [l4\\_uint32\\_t](#) \*dest)
- void [l4util\\_dec8](#) (volatile [l4\\_uint8\\_t](#) \*dest)
- void [l4util\\_dec16](#) (volatile [l4\\_uint16\\_t](#) \*dest)
- void [l4util\\_dec32](#) (volatile [l4\\_uint32\\_t](#) \*dest)

**Atomic inc/dec (8,16,32 bit) with result**

- [l4\\_uint8\\_t](#) [l4util\\_inc8\\_res](#) (volatile [l4\\_uint8\\_t](#) \*dest)
- [l4\\_uint16\\_t](#) [l4util\\_inc16\\_res](#) (volatile [l4\\_uint16\\_t](#) \*dest)
- [l4\\_uint32\\_t](#) [l4util\\_inc32\\_res](#) (volatile [l4\\_uint32\\_t](#) \*dest)
- [l4\\_uint8\\_t](#) [l4util\\_dec8\\_res](#) (volatile [l4\\_uint8\\_t](#) \*dest)
- [l4\\_uint16\\_t](#) [l4util\\_dec16\\_res](#) (volatile [l4\\_uint16\\_t](#) \*dest)
- [l4\\_uint32\\_t](#) [l4util\\_dec32\\_res](#) (volatile [l4\\_uint32\\_t](#) \*dest)

**14.16.6.1 Detailed Description****14.16.6.2 Function Documentation****14.16.6.2.1 l4util\_add16()**

```
void l4util_add16 (
    volatile l4\_uint16\_t * dest,
    l4\_uint16\_t val) [inline]
```

**Parameters**

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

Definition at line [472](#) of file [atomic.h](#).

**14.16.6.2.2 l4util\_add16\_res()**

```
l4\_uint16\_t l4util_add16_res (
    volatile l4\_uint16\_t * dest,
    l4\_uint16\_t val) [inline]
```

**Parameters**

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

**Returns**

res

Definition at line [524](#) of file [atomic.h](#).

#### 14.16.6.2.3 l4util\_add32()

```
void l4util_add32 (
    volatile l4_uint32_t * dest,
    l4_uint32_t val) [inline]
```

##### Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

Definition at line 476 of file [atomic.h](#).

#### 14.16.6.2.4 l4util\_add32\_res()

```
l4_uint32_t l4util_add32_res (
    volatile l4_uint32_t * dest,
    l4_uint32_t val) [inline]
```

##### Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

##### Returns

res

Definition at line 528 of file [atomic.h](#).

#### 14.16.6.2.5 l4util\_add8()

```
void l4util_add8 (
    volatile l4_uint8_t * dest,
    l4_uint8_t val) [inline]
```

##### Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

Definition at line 468 of file [atomic.h](#).

#### 14.16.6.2.6 l4util\_add8\_res()

```
l4_uint8_t l4util_add8_res (
    volatile l4_uint8_t * dest,
    l4_uint8_t val) [inline]
```

##### Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

**Returns**

res

Definition at line 520 of file [atomic.h](#).

**14.16.6.2.7 l4util\_and16()**

```
void l4util_and16 (  
    volatile l4_uint16_t * dest,  
    l4_uint16_t val) [inline]
```

**Parameters**

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

Definition at line 500 of file [atomic.h](#).

**14.16.6.2.8 l4util\_and16\_res()**

```
l4_uint16_t l4util_and16_res (  
    volatile l4_uint16_t * dest,  
    l4_uint16_t val) [inline]
```

**Parameters**

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

**Returns**

res

Definition at line 548 of file [atomic.h](#).

**14.16.6.2.9 l4util\_and32()**

```
void l4util_and32 (  
    volatile l4_uint32_t * dest,  
    l4_uint32_t val) [inline]
```

**Parameters**

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

Definition at line 504 of file [atomic.h](#).

#### 14.16.6.2.10 l4util\_and32\_res()

```
l4_uint32_t l4util_and32_res (  
    volatile l4_uint32_t * dest,  
    l4_uint32_t val) [inline]
```

##### Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

##### Returns

res

Definition at line 552 of file [atomic.h](#).

#### 14.16.6.2.11 l4util\_and8()

```
void l4util_and8 (  
    volatile l4_uint8_t * dest,  
    l4_uint8_t val) [inline]
```

##### Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

Definition at line 496 of file [atomic.h](#).

#### 14.16.6.2.12 l4util\_and8\_res()

```
l4_uint8_t l4util_and8_res (  
    volatile l4_uint8_t * dest,  
    l4_uint8_t val) [inline]
```

##### Parameters

<i>dest</i>	destination operand
-------------	---------------------



<i>val</i>	value to add/sub/and/or
------------	-------------------------

**Returns**

res

Definition at line 544 of file [atomic.h](#).

**14.16.6.2.13 l4util\_atomic\_add()**

```
void l4util_atomic_add (  
    volatile long * dest,  
    long val) [inline]
```

Atomic add.

**Parameters**

<i>dest</i>	destination operand
<i>val</i>	value to add

Definition at line 480 of file [atomic.h](#).

**14.16.6.2.14 l4util\_atomic\_inc()**

```
void l4util_atomic_inc (  
    volatile long * dest) [inline]
```

Atomic increment.

**Parameters**

<i>dest</i>	destination operand
-------------	---------------------

Definition at line 423 of file [atomic.h](#).

**14.16.6.2.15 l4util\_cmpxchg()**

```
int l4util_cmpxchg (  
    volatile l4_umword_t * dest,  
    l4_umword_t cmp_val,  
    l4_umword_t new_val) [inline]
```

Atomic compare and exchange (machine wide fields).

**Parameters**

<i>dest</i>	destination operand
<i>cmp_val</i>	compare value
<i>new_val</i>	new value for dest

#### Returns

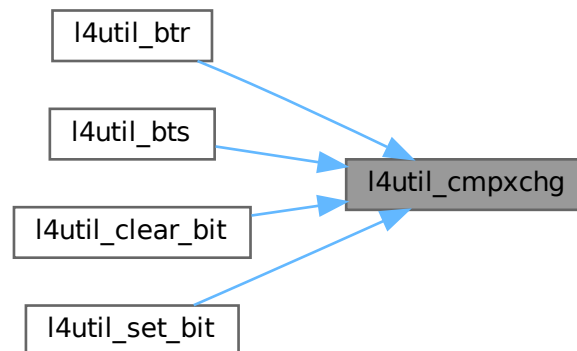
0 if comparison failed, 1 otherwise

Compare the value in *dest* with *cmp\_val*, if equal set *dest* to *new\_val*

Definition at line 379 of file [atomic.h](#).

Referenced by [l4util\\_btr\(\)](#), [l4util\\_bts\(\)](#), [l4util\\_clear\\_bit\(\)](#), and [l4util\\_set\\_bit\(\)](#).

Here is the caller graph for this function:



#### 14.16.6.2.16 l4util\_cmpxchg16()

```

int l4util_cmpxchg16 (
    volatile l4_uint16_t * dest,
    l4_uint16_t cmp_val,
    l4_uint16_t new_val) [inline]

```

Atomic compare and exchange (16 bit version).

#### Parameters

<i>dest</i>	destination operand
<i>cmp_val</i>	compare value
<i>new_val</i>	new value for dest

#### Returns

0 if comparison failed, !=0 otherwise

Compare the value in *dest* with *cmp\_val*, if equal set *dest* to *new\_val*

Definition at line 363 of file [atomic.h](#).

#### 14.16.6.2.17 l4util\_cmpxchg32()

```
L4_END_DECLS int l4util_cmpxchg32 (  
    volatile l4_uint32_t * dest,  
    l4_uint32_t cmp_val,  
    l4_uint32_t new_val) [inline]
```

Atomic compare and exchange (32 bit version).

##### Parameters

<i>dest</i>	destination operand
<i>cmp_val</i>	compare value
<i>new_val</i>	new value for dest

##### Returns

0 if comparison failed, !=0 otherwise

Compare the value in *dest* with *cmp\_val*, if equal set *dest* to *new\_val*

Definition at line 355 of file [atomic.h](#).

#### 14.16.6.2.18 l4util\_cmpxchg8()

```
int l4util_cmpxchg8 (  
    volatile l4_uint8_t * dest,  
    l4_uint8_t cmp_val,  
    l4_uint8_t new_val) [inline]
```

Atomic compare and exchange (8 bit version).

##### Parameters

<i>dest</i>	destination operand
<i>cmp_val</i>	compare value
<i>new_val</i>	new value for dest

##### Returns

0 if comparison failed, !=0 otherwise

Compare the value in *dest* with *cmp\_val*, if equal set *dest* to *new\_val*

Definition at line 371 of file [atomic.h](#).

#### 14.16.6.2.19 l4util\_dec16()

```
void l4util_dec16 (  
    volatile l4_uint16_t * dest) [inline]
```

##### Parameters

<i>dest</i>	destination operand
-------------	---------------------

Definition at line 431 of file [atomic.h](#).

#### 14.16.6.2.20 l4util\_dec16\_res()

```
l4_uint16_t l4util_dec16_res (
    volatile l4_uint16_t * dest) [inline]
```

##### Parameters

<i>dest</i>	destination operand
-------------	---------------------

##### Returns

res

Definition at line 456 of file [atomic.h](#).

#### 14.16.6.2.21 l4util\_dec32()

```
void l4util_dec32 (
    volatile l4_uint32_t * dest) [inline]
```

##### Parameters

<i>dest</i>	destination operand
-------------	---------------------

Definition at line 435 of file [atomic.h](#).

#### 14.16.6.2.22 l4util\_dec32\_res()

```
l4_uint32_t l4util_dec32_res (
    volatile l4_uint32_t * dest) [inline]
```

##### Parameters

<i>dest</i>	destination operand
-------------	---------------------

##### Returns

res

Definition at line 460 of file [atomic.h](#).

#### 14.16.6.2.23 l4util\_dec8()

```
void l4util_dec8 (
    volatile l4_uint8_t * dest) [inline]
```

##### Parameters

<i>dest</i>	destination operand
-------------	---------------------

Definition at line 427 of file [atomic.h](#).

#### 14.16.6.2.24 l4util\_dec8\_res()

```
l4_uint8_t l4util_dec8_res (  
    volatile l4_uint8_t * dest) [inline]
```

##### Parameters

<i>dest</i>	destination operand
-------------	---------------------

##### Returns

res

Definition at line 452 of file [atomic.h](#).

#### 14.16.6.2.25 l4util\_inc16()

```
void l4util_inc16 (  
    volatile l4_uint16_t * dest) [inline]
```

##### Parameters

<i>dest</i>	destination operand
-------------	---------------------

Definition at line 415 of file [atomic.h](#).

#### 14.16.6.2.26 l4util\_inc16\_res()

```
l4_uint16_t l4util_inc16_res (  
    volatile l4_uint16_t * dest) [inline]
```

##### Parameters

<i>dest</i>	destination operand
-------------	---------------------

##### Returns

res

Definition at line 444 of file [atomic.h](#).

#### 14.16.6.2.27 l4util\_inc32()

```
void l4util_inc32 (  
    volatile l4_uint32_t * dest) [inline]
```

##### Parameters

<i>dest</i>	destination operand
-------------	---------------------

Definition at line 419 of file [atomic.h](#).

#### 14.16.6.2.28 l4util\_inc32\_res()

```
l4_uint32_t l4util_inc32_res (
    volatile l4_uint32_t * dest) [inline]
```

##### Parameters

<i>dest</i>	destination operand
-------------	---------------------

##### Returns

res

Definition at line 448 of file [atomic.h](#).

#### 14.16.6.2.29 l4util\_inc8()

```
void l4util_inc8 (
    volatile l4_uint8_t * dest) [inline]
```

##### Parameters

<i>dest</i>	destination operand
-------------	---------------------

Definition at line 411 of file [atomic.h](#).

#### 14.16.6.2.30 l4util\_inc8\_res()

```
l4_uint8_t l4util_inc8_res (
    volatile l4_uint8_t * dest) [inline]
```

##### Parameters

<i>dest</i>	destination operand
-------------	---------------------

##### Returns

res

Definition at line 440 of file [atomic.h](#).

#### 14.16.6.2.31 l4util\_or16()

```
void l4util_or16 (
    volatile l4_uint16_t * dest,
    l4_uint16_t val) [inline]
```

##### Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

Definition at line 512 of file [atomic.h](#).

#### 14.16.6.2.32 l4util\_or16\_res()

```
l4_uint16_t l4util_or16_res (  
    volatile l4_uint16_t * dest,  
    l4_uint16_t val) [inline]
```

##### Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

##### Returns

res

Definition at line 560 of file [atomic.h](#).

#### 14.16.6.2.33 l4util\_or32()

```
void l4util_or32 (  
    volatile l4_uint32_t * dest,  
    l4_uint32_t val) [inline]
```

##### Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

Definition at line 516 of file [atomic.h](#).

#### 14.16.6.2.34 l4util\_or32\_res()

```
l4_uint32_t l4util_or32_res (  
    volatile l4_uint32_t * dest,  
    l4_uint32_t val) [inline]
```

##### Parameters

<i>dest</i>	destination operand
-------------	---------------------

<i>val</i>	value to add/sub/and/or
------------	-------------------------

**Returns**

res

Definition at line 564 of file [atomic.h](#).

**14.16.6.2.35 l4util\_or8()**

```
void l4util_or8 (  
    volatile l4_uint8_t * dest,  
    l4_uint8_t val) [inline]
```

**Parameters**

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

Definition at line 508 of file [atomic.h](#).

**14.16.6.2.36 l4util\_or8\_res()**

```
l4_uint8_t l4util_or8_res (  
    volatile l4_uint8_t * dest,  
    l4_uint8_t val) [inline]
```

**Parameters**

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

**Returns**

res

Definition at line 556 of file [atomic.h](#).

**14.16.6.2.37 l4util\_sub16()**

```
void l4util_sub16 (  
    volatile l4_uint16_t * dest,  
    l4_uint16_t val) [inline]
```

**Parameters**

--



<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

Definition at line 488 of file [atomic.h](#).

#### 14.16.6.2.38 l4util\_sub16\_res()

```
l4_uint16_t l4util_sub16_res (  
    volatile l4_uint16_t * dest,  
    l4_uint16_t val) [inline]
```

##### Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

##### Returns

res

Definition at line 536 of file [atomic.h](#).

#### 14.16.6.2.39 l4util\_sub32()

```
void l4util_sub32 (  
    volatile l4_uint32_t * dest,  
    l4_uint32_t val) [inline]
```

##### Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

Definition at line 492 of file [atomic.h](#).

#### 14.16.6.2.40 l4util\_sub32\_res()

```
l4_uint32_t l4util_sub32_res (  
    volatile l4_uint32_t * dest,  
    l4_uint32_t val) [inline]
```

##### Parameters

<i>dest</i>	destination operand
-------------	---------------------

<i>val</i>	value to add/sub/and/or
------------	-------------------------

#### Returns

res

Definition at line 540 of file [atomic.h](#).

#### 14.16.6.2.41 l4util\_sub8()

```
void l4util_sub8 (  
    volatile l4_uint8_t * dest,  
    l4_uint8_t val) [inline]
```

#### Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

Definition at line 484 of file [atomic.h](#).

#### 14.16.6.2.42 l4util\_sub8\_res()

```
l4_uint8_t l4util_sub8_res (  
    volatile l4_uint8_t * dest,  
    l4_uint8_t val) [inline]
```

#### Parameters

<i>dest</i>	destination operand
<i>val</i>	value to add/sub/and/or

#### Returns

res

Definition at line 532 of file [atomic.h](#).

#### 14.16.6.2.43 l4util\_xchg()

```
l4_umword_t l4util_xchg (  
    volatile l4_umword_t * dest,  
    l4_umword_t val) [inline]
```

Atomic exchange (machine wide fields).

#### Parameters

---

<i>dest</i>	destination operand
<i>val</i>	new value for dest

**Returns**

old value at destination

Definition at line 405 of file [atomic.h](#).

**14.16.6.2.44 l4util\_xchg16()**

```
l4_uint16_t l4util_xchg16 (  
    volatile l4_uint16_t * dest,  
    l4_uint16_t val) [inline]
```

Atomic exchange (16 bit version).

**Parameters**

<i>dest</i>	destination operand
<i>val</i>	new value for dest

**Returns**

old value at destination

Definition at line 393 of file [atomic.h](#).

**14.16.6.2.45 l4util\_xchg32()**

```
l4_uint32_t l4util_xchg32 (  
    volatile l4_uint32_t * dest,  
    l4_uint32_t val) [inline]
```

Atomic exchange (32 bit version).

**Parameters**

<i>dest</i>	destination operand
<i>val</i>	new value for dest

**Returns**

old value at destination

Definition at line 387 of file [atomic.h](#).

#### 14.16.6.2.46 l4util\_xchg8()

```
l4_uint8_t l4util_xchg8 (  
    volatile l4_uint8_t * dest,  
    l4_uint8_t val) [inline]
```

Atomic exchange (8 bit version).

#### Parameters

---

<i>dest</i>	destination operand
<i>val</i>	new value for dest

#### Returns

old value at destination

Definition at line 399 of file [atomic.h](#).

### 14.16.7 Internal functions

Collaboration diagram for Internal functions:



#### Functions

- void **base64\_encode** (const char \*infile, unsigned int in\_size, char \*\*outfile)  
*base-64-encode string infile*
- void **base64\_decode** (const char \*infile, unsigned int in\_size, char \*\*outfile)  
*decode base-64-encoded string infile*

#### 14.16.7.1 Detailed Description

### 14.16.8 Bit Manipulation

Collaboration diagram for Bit Manipulation:



## Files

- file [bitops\\_arch.h](#)  
*amd64 bit manipulation functions*
- file [bitops.h](#)  
*bit manipulation functions*
- file [bitops\\_arch.h](#)  
*x86 bit manipulation functions*

## Functions

- void [l4util\\_set\\_bit](#) (int b, volatile [l4\\_umword\\_t](#) \*dest)  
*Set bit in memory.*
- void [l4util\\_clear\\_bit](#) (int b, volatile [l4\\_umword\\_t](#) \*dest)  
*Clear bit in memory.*
- void [l4util\\_complement\\_bit](#) (int b, volatile [l4\\_umword\\_t](#) \*dest)  
*Complement bit in memory.*
- int [l4util\\_test\\_bit](#) (int b, const volatile [l4\\_umword\\_t](#) \*dest)  
*Test bit (return value of bit).*
- int [l4util\\_bts](#) (int b, volatile [l4\\_umword\\_t](#) \*dest)  
*Bit test and set.*
- int [l4util\\_btr](#) (int b, volatile [l4\\_umword\\_t](#) \*dest)  
*Bit test and reset.*
- int [l4util\\_btc](#) (int b, volatile [l4\\_umword\\_t](#) \*dest)  
*Bit test and complement.*
- int [l4util\\_bsr](#) ([l4\\_umword\\_t](#) word)  
*Bit scan reverse.*
- int [l4util\\_bsf](#) ([l4\\_umword\\_t](#) word)  
*Bit scan forward.*
- int [l4util\\_find\\_first\\_set\\_bit](#) (const void \*dest, [l4\\_size\\_t](#) size)  
*Find the first set bit in a memory region.*
- int [l4util\\_find\\_first\\_zero\\_bit](#) (const void \*dest, [l4\\_size\\_t](#) size)  
*Find the first zero bit in a memory region.*
- int [l4util\\_next\\_power2](#) (unsigned long val)  
*Find the next power of 2 for a given number.*

### 14.16.8.1 Detailed Description

### 14.16.8.2 Function Documentation

#### 14.16.8.2.1 l4util\_bsf()

```
int l4util_bsf (
    l4\_umword\_t word) [inline]
```

Bit scan forward.

#### Parameters

---

<i>word</i>	value (machine size)
-------------	----------------------

**Returns**

index of least significant bit set in word, -1 if no bit is set (word == 0)

"bit scan forward", find least significant bit set in word.

Definition at line 316 of file [bitops.h](#).

**14.16.8.2.2 l4util\_bsr()**

```
int l4util_bsr (  
    l4_umword_t word)    [inline]
```

Bit scan reverse.

**Parameters**

<i>word</i>	value (machine size)
-------------	----------------------

**Returns**

index of most significant set bit in word, -1 if no bit is set (word == 0)

"bit scan reverse", find most significant set bit in word (-> LOG2(word))

Definition at line 299 of file [bitops.h](#).

**14.16.8.2.3 l4util\_btc()**

```
int l4util_btc (  
    int b,  
    volatile l4_umword_t * dest)    [inline]
```

Bit test and complement.

**Parameters**

<i>b</i>	bit position
<i>dest</i>	destination operand

**Returns**

Old value of bit *b*.

Complement bit *b* and return old value.

Definition at line 394 of file [bitops.h](#).

#### 14.16.8.2.4 l4util\_btr()

```
int l4util_btr (  
    int b,  
    volatile l4_umword_t * dest) [inline]
```

Bit test and reset.

#### Parameters

---



<i>b</i>	bit position
<i>dest</i>	destination operand

#### Returns

Old value of bit *b*.

Reset bit *b* and return old value.

Definition at line 278 of file [bitops.h](#).

References [l4util\\_cmpxchg\(\)](#).

Here is the call graph for this function:



#### 14.16.8.2.5 l4util\_bts()

```
int l4util_bts (  
    int b,  
    volatile l4_umword_t * dest) [inline]
```

Bit test and set.

#### Parameters

<i>b</i>	bit position
<i>dest</i>	destination operand

#### Returns

Old value of bit *b*.

Set the *b* bit of *dest* to 1 and return the old value.

Definition at line 256 of file [bitops.h](#).

References [l4util\\_cmpxchg\(\)](#).

Here is the call graph for this function:



#### 14.16.8.2.6 l4util\_clear\_bit()

```

void l4util_clear_bit (
    int b,
    volatile l4_umword_t * dest) [inline]
  
```

Clear bit in memory.

##### Parameters

<i>b</i>	bit position
<i>dest</i>	destination operand

Definition at line 226 of file [bitops.h](#).

References [l4util\\_cmpxchg\(\)](#).

Here is the call graph for this function:



#### 14.16.8.2.7 l4util\_complement\_bit()

```

void l4util_complement_bit (
    int b,
    volatile l4_umword_t * dest) [inline]
  
```

Complement bit in memory.

##### Parameters

<i>b</i>	bit position
<i>dest</i>	destination operand

Definition at line 359 of file [bitops.h](#).

#### 14.16.8.2.8 l4util\_find\_first\_set\_bit()

```
int l4util_find_first_set_bit (
    const void * dest,
    l4_size_t size) [inline]
```

Find the first set bit in a memory region.

##### Parameters

<i>dest</i>	bit string
<i>size</i>	size of string in bits (must be a multiple of L4_MWORD_BITS!)

##### Returns

number of the first set bit, >= size if no bit is set

Definition at line 400 of file [bitops.h](#).

#### 14.16.8.2.9 l4util\_find\_first\_zero\_bit()

```
int l4util_find_first_zero_bit (
    const void * dest,
    l4_size_t size) [inline]
```

Find the first zero bit in a memory region.

##### Parameters

<i>dest</i>	bit string
<i>size</i>	size of string in bits (must be a multiple of L4_MWORD_BITS!)

##### Returns

number of the first zero bit, >= size if no bit is set

Definition at line 333 of file [bitops.h](#).

#### 14.16.8.2.10 l4util\_next\_power2()

```
int l4util_next_power2 (
    unsigned long val) [inline]
```

Find the next power of 2 for a given number.

##### Parameters

<i>val</i>	initial value
------------	---------------

**Returns**

next-highest power of 2

Definition at line 373 of file [bitops.h](#).

**14.16.8.2.11 l4util\_set\_bit()**

```
void l4util_set_bit (
    int b,
    volatile l4_umword_t * dest) [inline]
```

Set bit in memory.

**Parameters**

<i>b</i>	bit position
<i>dest</i>	destination operand

Definition at line 207 of file [bitops.h](#).

References [l4util\\_cmpxchg\(\)](#).

Here is the call graph for this function:

**14.16.8.2.12 l4util\_test\_bit()**

```
int l4util_test_bit (
    int b,
    const volatile l4_umword_t * dest) [inline]
```

Test bit (return value of bit).

**Parameters**

<i>b</i>	bit position
<i>dest</i>	destination operand

#### Returns

Value of bit *b*.

Definition at line 244 of file [bitops.h](#).

### 14.16.9 ELF binary format

Functions and types related to ELF binaries.

Collaboration diagram for ELF binary format:



#### Files

- file [elf.h](#)  
*ELF definition.*

#### Data Structures

- struct [Elf32\\_Ehdr](#)  
*ELF32 header.*
- struct [Elf64\\_Ehdr](#)  
*ELF64 header.*
- struct [Elf32\\_Shdr](#)  
*ELF32 section header.*
- struct [Elf64\\_Shdr](#)  
*ELF64 section header.*
- struct [Elf32\\_Phdr](#)  
*ELF32 program header.*
- struct [Elf64\\_Phdr](#)  
*ELF64 program header.*
- struct [Elf32\\_Dyn](#)  
*ELF32 dynamic entry.*
- struct [Elf64\\_Dyn](#)  
*ELF64 dynamic entry.*

- struct [Elf32\\_Rel](#)  
*ELF32 relocation entry w/o addend.*
- struct [Elf32\\_Rela](#)  
*ELF32 relocation entry w/ addend.*
- struct [Elf64\\_Rel](#)  
*ELF64 relocation entry w/o addend.*
- struct [Elf64\\_Rela](#)  
*ELF64 relocation entry w/ addend.*
- struct [Elf32\\_Sym](#)  
*ELF32 symbol table entry.*
- struct [Elf64\\_Sym](#)  
*ELF64 symbol table entry.*
- struct [Elf32\\_Auxv](#)  
*Auxiliary vector (32-bit).*
- struct [Elf64\\_Auxv](#)  
*Auxiliary vector (64-bit).*

## Macros

- #define [ElfW](#)(type)  
*Use 64 or 32 bits types depending on the target architecture.*
- #define [ELF32\\_R\\_SYM](#)(i)  
*Symbol table index.*
- #define [ELF32\\_R\\_TYPE](#)(i)
- #define [ELF32\\_R\\_INFO](#)(s, t)  
*Create info from symbol table index + type.*
- #define [ELF64\\_R\\_SYM](#)(i)  
*Symbol table index.*
- #define [ELF64\\_R\\_TYPE](#)(i)
- #define [ELF64\\_R\\_INFO](#)(s, t)  
*Create info from symbol table index + type.*
- #define [ELF32\\_ST\\_BIND](#)(i)
- #define [ELF32\\_ST\\_TYPE](#)(i)
- #define [ELF32\\_ST\\_INFO](#)(b, t)  
*Make info from bind + type.*
- #define [ELF64\\_ST\\_BIND](#)(i)
- #define [ELF64\\_ST\\_TYPE](#)(i)
- #define [ELF64\\_ST\\_INFO](#)(b, t)  
*Make info from bind + type.*

## Typedefs

- typedef struct [Elf32\\_Auxv](#) [Elf32\\_Auxv](#)  
*Auxiliary vector (32-bit).*
- typedef struct [Elf64\\_Auxv](#) [Elf64\\_Auxv](#)  
*Auxiliary vector (64-bit).*

## Enumerations

- enum { EI\_NIDENT = 16 }
- enum Elf\_ETs {  
 ET\_NONE = 0 , ET\_REL = 1 , ET\_EXEC = 2 , ET\_DYN = 3 ,  
 ET\_CORE = 4 , ET\_LOPROC = 0xff00 , ET\_HIPROC = 0xffff }  
*Object file type.*
- enum Elf\_EMs {  
 EM\_NONE = 0 , EM\_M32 = 1 , EM\_SPARC = 2 , EM\_386 = 3 ,  
 EM\_68K = 4 , EM\_88K = 5 , EM\_860 = 7 , EM\_MIPS = 8 ,  
 EM\_MIPS\_RS4\_BE = 10 , EM\_SPARC64 = 11 , EM\_PARISC = 15 , EM\_VPP500 = 17 ,  
 EM\_SPARC32PLUS = 18 , EM\_960 = 19 , EM\_PPC = 20 , EM\_V800 = 36 ,  
 EM\_FR20 = 37 , EM\_RH32 = 38 , EM\_RCE = 39 , EM\_ARM = 40 ,  
 EM\_ALPHA = 41 , EM\_SH = 42 , EM\_SPARCV9 = 43 , EM\_TRICORE = 44 ,  
 EM\_ARC = 45 , EM\_H8\_300 = 46 , EM\_H8\_300H = 47 , EM\_H8S = 48 ,  
 EM\_H8\_500 = 49 , EM\_IA\_64 = 50 , EM\_MIPS\_X = 51 , EM\_COLDFIRE = 52 ,  
 EM\_68HC12 = 53 , EM\_X86\_64 = 62 , EM\_PDSP = 63 , EM\_FX66 = 66 ,  
 EM\_ST9PLUS = 67 , EM\_ST7 = 68 , EM\_68HC16 = 69 , EM\_68HC11 = 70 ,  
 EM\_68HC08 = 71 , EM\_68HC05 = 72 , EM\_SVX = 73 , EM\_ST19 = 74 ,  
 EM\_VAX = 75 , EM\_CRIS = 76 , EM\_JAVELIN = 77 , EM\_FIREPATH = 78 ,  
 EM\_ZSP = 79 , EM\_MMIX = 80 , EM\_HUANY = 81 , EM\_PRISM = 82 ,  
 EM\_AVR = 83 , EM\_FR30 = 84 , EM\_D10V = 85 , EM\_D30V = 86 ,  
 EM\_V850 = 87 , EM\_M32R = 88 , EM\_MN10300 = 89 , EM\_MN10200 = 90 ,  
 EM\_PJ = 91 , EM\_OPENRISC = 92 , EM\_ARC\_A5 = 93 , EM\_XTENSA = 94 ,  
 EM\_ALTERA\_NIOS2 = 113 , EM\_AARCH64 = 183 , EM\_TILEPRO = 188 , EM\_MICROBLAZE = 189 ,  
 EM\_TILEGX = 191 , EM\_RISCV = 243 , EM\_NUM = 244 }  
*Required architecture.*
- enum Elf\_EVs { EV\_NONE = 0 , EV\_CURRENT = 1 }
- enum Elf\_EIs {  
 EI\_MAG0 = 0 , EI\_MAG1 = 1 , EI\_MAG2 = 2 , EI\_MAG3 = 3 ,  
 EI\_CLASS = 4 , EI\_DATA = 5 , EI\_VERSION = 6 , EI\_OSABI = 7 ,  
 EI\_ABIVERSION = 8 , EI\_PAD = 9 }  
*Identification Indices.*
- enum Elf\_MAGs { ELF\_MAG0 = 0x7f , ELF\_MAG1 = 'E' , ELF\_MAG2 = 'L' , ELF\_MAG3 = 'F' }  
*Magic number.*
- enum Elf\_CLASSs { ELF\_CLASSNONE = 0 , ELF\_CLASS32 = 1 , ELF\_CLASS64 = 2 , ELF\_CLASSNUM = 3 }  
*File class or capacity.*
- enum Elf\_DATAs { ELFDATANONE = 0 , ELFDATA2LSB = 1 , ELFDATA2MSB = 2 , ELFDATANUM = 3 }  
*Data encoding.*
- enum Elf\_OSABIs {  
 ELFOSABI\_NONE = 0 , ELFOSABI\_SYSV = 0 , ELFOSABI\_HPUX = 1 , ELFOSABI\_NETBSD = 2 ,  
 ELFOSABI\_LINUX = 3 , ELFOSABI\_SOLARIS = 6 , ELFOSABI\_AIX = 7 , ELFOSABI\_IRIX = 8 ,  
 ELFOSABI\_FREEBSD = 9 , ELFOSABI\_TRU64 = 10 , ELFOSABI\_MODESTO = 11 , ELFOSABI\_OPENBSD  
 = 12 ,  
 ELFOSABI\_ARM = 97 , ELFOSABI\_STANDALONE = 255 }  
*Identify operating system and ABI to which the object is targeted.*
- enum Elf\_SHNs {  
 SHN\_UNDEF = 0 , SHN\_LORESERVE = 0xff00 , SHN\_LOPROC = 0xff00 , SHN\_HIPROC = 0xff1f ,  
 SHN\_ABS = 0xffff , SHN\_COMMON = 0xffff2 , SHN\_HIRESERVE = 0xffff }  
*Special section indexes.*
- enum Elf\_SHTs {  
 SHT\_NULL = 0 , SHT\_PROGBITS = 1 , SHT\_SYMTAB = 2 , SHT\_STRTAB = 3 ,  
 SHT\_RELA = 4 , SHT\_HASH = 5 , SHT\_DYNAMIC = 6 , SHT\_NOTE = 7 ,  
 SHT\_NOBITS = 8 , SHT\_REL = 9 , SHT\_SHLIB = 10 , SHT\_DYNSYM = 11 ,

SHT\_INIT\_ARRAY = 14 , SHT\_FINI\_ARRAY = 15 , SHT\_PREINIT\_ARRAY = 16 , SHT\_GROUP = 17 ,  
 SHT\_SYMTAB\_SHNDX = 18 , SHT\_NUM = 19 , SHT\_LOOS = 0x60000000 , SHT\_HIOS = 0x6fffffff ,  
 SHT\_LOPROC = 0x70000000 , SHT\_HIPROC = 0x7fffffff , SHT\_LOUSER = 0x80000000 , SHT\_HIUSER =  
 0xffffffff }

*Section type.*

- enum **Elf\_SHFs** {  
 SHF\_WRITE = 0x1 , SHF\_ALLOC = 0x2 , SHF\_EXECINSTR = 0x4 , SHF\_MERGE = 0x10 ,  
 SHF\_STRINGS = 0x20 , SHF\_INFO\_LINK = 0x40 , SHF\_OS\_NONCONFORMING = 0x100 , SHF\_GROUP  
 = 0x200 ,  
 SHF\_TLS = 0x400 , SHF\_MASKOS = 0x0ff00000 , SHF\_MASKPROC = 0xf0000000 }

*Section attribute flags.*

- enum **Elf\_PT**s {  
 PT\_NULL = 0 , PT\_LOAD = 1 , PT\_DYNAMIC = 2 , PT\_INTERP = 3 ,  
 PT\_NOTE = 4 , PT\_SHLIB = 5 , PT\_PHDR = 6 , PT\_TLS = 7 ,  
 PT\_NUM = 8 , PT\_LOOS = 0x60000000 , PT\_HIOS = 0x6fffffff , PT\_LOPROC = 0x70000000 ,  
 PT\_HIPROC = 0x7fffffff , PT\_GNU\_EH\_FRAME = PT\_LOOS + 0x474e550 , PT\_GNU\_STACK = PT\_LOOS  
 + 0x474e551 , PT\_GNU\_RELRO = PT\_LOOS + 0x474e552 ,  
 PT\_L4\_STACK = PT\_LOOS + 0x12 , PT\_L4\_AUX = PT\_LOOS + 0x14 }

*Segment types.*

- enum **Elf\_PF**s {  
 PF\_X = 0x1 , PF\_W = 0x2 , PF\_R = 0x4 , PF\_MASKOS = 0x0ff00000 ,  
 PF\_MASKPROC = 0x7fffffff }

*Segment permissions.*

- enum **Elf\_NT**s\_core {  
 NT\_PRSTATUS = 1 , NT\_FPREGSET = 2 , NT\_PRPSINFO = 3 , NT\_PRXREG = 4 ,  
 NT\_TASKSTRUCT = 4 , NT\_PLATFORM = 5 , NT\_AUXV = 6 , NT\_GWINDOWS = 7 ,  
 NT\_ASRS = 8 , NT\_PSTATUS = 10 , NT\_PSINFO = 13 , NT\_PRCRED = 14 ,  
 NT\_UTSNAME = 15 , NT\_LWPSTATUS = 16 , NT\_LWPSINFO = 17 , NT\_PRFPXREG = 20 }

*Legal values for note segment descriptor types for core files.*

- enum **Elf\_NT**s\_obj { NT\_VERSION = 1 }

*Legal values for the note segment descriptor types for object files.*

- enum **Elf\_DT**s {  
 DT\_NULL = 0 , DT\_NEEDED = 1 , DT\_PLTRELSZ = 2 , DT\_PLTGOT = 3 ,  
 DT\_HASH = 4 , DT\_STRTAB = 5 , DT\_SYMTAB = 6 , DT\_RELA = 7 ,  
 DT\_RELASZ = 8 , DT\_RELAENT = 9 , DT\_STRSZ = 10 , DT\_SYMENT = 11 ,  
 DT\_INIT = 12 , DT\_FINI = 13 , DT\_SONAME = 14 , DT\_RPATH = 15 ,  
 DT\_SYMBOLIC = 16 , DT\_REL = 17 , DT\_RELSZ = 18 , DT\_RELENT = 19 ,  
 DT\_PTRREL = 20 , DT\_DEBUG = 21 , DT\_TEXTREL = 22 , DT\_JMPREL = 23 ,  
 DT\_BIND\_NOW = 24 , DT\_INIT\_ARRAY = 25 , DT\_FINI\_ARRAY = 26 , DT\_INIT\_ARRAYSZ = 27 ,  
 DT\_FINI\_ARRAYSZ = 28 , DT\_RUNPATH = 29 , DT\_FLAGS = 30 , DT\_ENCODING = 32 ,  
 DT\_PREINIT\_ARRAY = 32 , DT\_PREINIT\_ARRAYSZ = 33 , DT\_NUM = 34 , DT\_LOOS = 0x6000000d ,  
 DT\_HIOS = 0x6ffff000 , DT\_LOPROC = 0x70000000 , DT\_HIPROC = 0x7fffffff }

*Dynamic Array Tags.*

- enum **Elf\_DF**s {  
 DF\_ORIGIN = 0x00000001 , DF\_SYMBOLIC = 0x00000002 , DF\_TEXTREL = 0x00000004 ,  
 DF\_BIND\_NOW = 0x00000008 ,  
 DF\_STATIC\_TLS = 0x00000010 }

*Values of Elf32\_Dyn.d\_un.d\_val, Elf64\_Dyn.d\_un.d\_val in the DT\_FLAGS entry.*

- enum **Elf\_DF\_1**s {  
 DF\_1\_NOW = 0x00000001 , DF\_1\_GLOBAL = 0x00000002 , DF\_1\_GROUP = 0x00000004 ,  
 DF\_1\_NODELETE = 0x00000008 ,  
 DF\_1\_LOADFLTR = 0x00000010 , DF\_1\_INITFIRST = 0x00000020 , DF\_1\_NOOPEN = 0x00000040 ,  
 DF\_1\_ORIGIN = 0x00000080 ,  
 DF\_1\_DIRECT = 0x00000100 , DF\_1\_TRANS = 0x00000200 , DF\_1\_INTERPOSE = 0x00000400 ,  
 DF\_1\_NODEFLIB = 0x00000800 ,  
 DF\_1\_NODUMP = 0x00001000 , DF\_1\_CONFALT = 0x00002000 , DF\_1\_ENDFILTEE = 0x00004000 ,



```
DF_1_DISPRELDNE = 0x00008000 ,
DF_1_DISPRELPND = 0x00010000 }
```

State flags selectable in the *Elf32\_Dyn.d\_un.d\_val* / *Elf64\_Dyn.d\_un.d\_val* element of the *DT\_FLAGS\_1* entry in the dynamic section.

- enum *Elf\_DTF\_1s*

Flags for the feature selection in *DT\_FEATURE\_1*.

- enum *Elf\_DF\_P1s* { *DF\_P1\_LAZYLOAD* = 0x00000001 , *DF\_P1\_GROUPPERM* = 0x00000002 }

Flags in the *DT\_POSFLAG\_1* entry effecting only the next *DT\_\** entry.

- enum *Elf\_R\_386\_s* {  
*R\_386\_NONE* = 0 , *R\_386\_32* = 1 , *R\_386\_PC32* = 2 , *R\_386\_GOT32* = 3 ,  
*R\_386\_PLT32* = 4 , *R\_386\_COPY* = 5 , *R\_386\_GLOB\_DAT* = 6 , *R\_386\_JMP\_SLOT* = 7 ,  
*R\_386\_RELATIVE* = 8 , *R\_386\_GOTOFF* = 9 , *R\_386\_GOTPC* = 10 , *R\_386\_32PLT* = 11 ,  
*R\_386\_TLS\_TPOFF* = 14 , *R\_386\_TLS\_IE* = 15 , *R\_386\_TLS\_GOTIE* = 16 , *R\_386\_TLS\_LE* = 17 ,  
*R\_386\_TLS\_GD* = 18 , *R\_386\_TLS\_LDM* = 19 , *R\_386\_16* = 20 , *R\_386\_PC16* = 21 ,  
*R\_386\_8* = 22 , *R\_386\_PC8* = 23 , *R\_386\_TLS\_GD\_32* = 24 , *R\_386\_TLS\_GD\_PUSH* = 25 ,  
*R\_386\_TLS\_GD\_CALL* = 26 , *R\_386\_TLS\_GD\_POP* = 27 , *R\_386\_TLS\_LDM\_32* = 28 , *R\_386\_TLS\_LDM\_PUSH*  
= 29 ,  
*R\_386\_TLS\_LDM\_CALL* = 30 , *R\_386\_TLS\_LDM\_POP* = 31 , *R\_386\_TLS\_LDO\_32* = 32 , *R\_386\_TLS\_IE\_32*  
= 33 ,  
*R\_386\_TLS\_LE\_32* = 34 , *R\_386\_TLS\_DTPMOD32* = 35 , *R\_386\_TLS\_DTPOFF32* = 36 , *R\_386\_TLS\_TPOFF32*  
= 37 ,  
*R\_386\_NUM* = 38 }

Relocation types (processor specific).

- enum *Elf\_EF\_ARM\_s* { }

ARM specific declarations.

- enum *Elf\_STT\_ARM\_s*

Additional symbol types for Thumb.

- enum *Elf\_SHF\_s\_ARM* { *SHF\_ARM\_ENTRYSECT* = 0x10000000 , *SHF\_ARM\_COMDEF* = 0x80000000 }

ARM-specific values for *Elf32\_Shdr.sh\_flags* / *Elf64\_Shdr.sh\_flags*.

- enum *Elf\_ARM\_SBs* { *PF\_ARM\_SB* = 0x10000000 }

ARM-specific program header flags.

- enum *Elf\_R\_ARM\_s* {  
*R\_ARM\_NONE* = 0 , *R\_ARM\_PC24* = 1 , *R\_ARM\_ABS32* = 2 , *R\_ARM\_REL32* = 3 ,  
*R\_ARM\_PC13* = 4 , *R\_ARM\_ABS16* = 5 , *R\_ARM\_ABS12* = 6 , *R\_ARM\_THM\_ABS5* = 7 ,  
*R\_ARM\_ABS8* = 8 , *R\_ARM\_SBREL32* = 9 , *R\_ARM\_THM\_PC22* = 10 , *R\_ARM\_THM\_PC8* = 11 ,  
*R\_ARM\_AMP\_VCALL9* = 12 , *R\_ARM\_SWI24* = 13 , *R\_ARM\_THM\_SWI8* = 14 , *R\_ARM\_XPC25* = 15 ,  
*R\_ARM\_THM\_XPC22* = 16 , *R\_ARM\_COPY* = 20 , *R\_ARM\_GLOB\_DAT* = 21 , *R\_ARM\_JUMP\_SLOT* = 22 ,  
*R\_ARM\_RELATIVE* = 23 , *R\_ARM\_GOTOFF* = 24 , *R\_ARM\_GOTPC* = 25 , *R\_ARM\_GOT32* = 26 ,  
*R\_ARM\_PLT32* = 27 , *R\_ARM\_ALU\_PCREL\_7\_0* = 32 , *R\_ARM\_ALU\_PCREL\_15\_8* = 33 , *R\_ARM\_↵*  
*ALU\_PCREL\_23\_15* = 34 ,  
*R\_ARM\_LDR\_SBREL\_11\_0* = 35 , *R\_ARM\_ALU\_SBREL\_19\_12* = 36 , *R\_ARM\_ALU\_SBREL\_27\_20* =  
37 , *R\_ARM\_GNU\_VTENTRY* = 100 ,  
*R\_ARM\_GNU\_VTINHERIT* = 101 , *R\_ARM\_THM\_PC11* = 102 , *R\_ARM\_THM\_PC9* = 103 , *R\_ARM\_↵*  
*RXPC25* = 249 ,  
*R\_ARM\_RSBREL32* = 250 , *R\_ARM\_THM\_RPC22* = 251 , *R\_ARM\_RREL32* = 252 , *R\_ARM\_RABS22* =  
253 ,  
*R\_ARM\_RPC24* = 254 , *R\_ARM\_RBASE* = 255 , *R\_ARM\_NUM* = 256 }

ARM relocations.

- enum *Elf\_R\_AARCH64\_s* { *R\_AARCH64\_NONE* = 0 , *R\_AARCH64\_RELATIVE* = 1027 }

AARCH64 relocations.

- enum *Elf\_R\_X86\_64\_s* {  
*R\_X86\_64\_NONE* = 0 , *R\_X86\_64\_64* = 1 , *R\_X86\_64\_PC32* = 2 , *R\_X86\_64\_GOT32* = 3 ,  
*R\_X86\_64\_PLT32* = 4 , *R\_X86\_64\_COPY* = 5 , *R\_X86\_64\_GLOB\_DAT* = 6 , *R\_X86\_64\_JUMP\_SLOT* = 7 ,  
*R\_X86\_64\_RELATIVE* = 8 , *R\_X86\_64\_GOTPCREL* = 9 , *R\_X86\_64\_32* = 10 , *R\_X86\_64\_32S* = 11 ,  
*R\_X86\_64\_16* = 12 , *R\_X86\_64\_PC16* = 13 , *R\_X86\_64\_8* = 14 , *R\_X86\_64\_PC8* = 15 ,

```

R_X86_64_DTPMOD64 = 16 , R_X86_64_DTPOFF64 = 17 , R_X86_64_TPOFF64 = 18 , R_X86_64_TLSD
= 19 ,
R_X86_64_TLSD = 20 , R_X86_64_DTPOFF32 = 21 , R_X86_64_GOTTPOFF = 22 , R_X86_64_TPOFF32
= 23 ,
R_X86_64_NUM = 24 }

```

*AMD x86-64 relocations.*

- enum **Elf\_STNs**

*Symbol Table Entry.*

- enum **Elf\_STBs** {  
**STB\_LOCAL** = 0 , **STB\_GLOBAL** = 1 , **STB\_WEAK** = 2 , **STB\_LOOS** = 10 ,  
**STB\_HIOS** = 12 , **STB\_LOPROC** = 13 , **STB\_HIPROC** = 15 }

*Symbol Binding.*

- enum **Elf\_STTs** {  
**STT\_NOTYPE** = 0 , **STT\_OBJECT** = 1 , **STT\_FUNC** = 2 , **STT\_SECTION** = 3 ,  
**STT\_FILE** = 4 , **STT\_LOOS** = 10 , **STT\_HIOS** = 12 , **STT\_LOPROC** = 13 ,  
**STT\_HIPROC** = 15 }

*Symbol Types.*

- enum **Elf\_ATs** {  
**AT\_NULL** = 0 , **AT\_IGNORE** = 1 , **AT\_EXECD** = 2 , **AT\_PHDR** = 3 ,  
**AT\_PHENT** = 4 , **AT\_PHNUM** = 5 , **AT\_PAGESZ** = 6 , **AT\_BASE** = 7 ,  
**AT\_FLAGS** = 8 , **AT\_ENTRY** = 9 , **AT\_NOTELF** = 10 , **AT\_UID** = 11 ,  
**AT\_EUID** = 12 , **AT\_GID** = 13 , **AT\_EGID** = 14 , **AT\_L4\_AUX** = 0xf0 ,  
**AT\_L4\_ENV** = 0xf1 }

*Legal values for [Elf32\\_Auxv.atype](#) / [Elf64\\_Auxv.atype](#).*

## ELF types

- typedef **l4\_uint32\_t** **Elf32\_Addr**  
*size 4 align 4*
- typedef **l4\_uint32\_t** **Elf32\_Off**  
*size 4 align 4*
- typedef **l4\_uint16\_t** **Elf32\_Half**  
*size 2 align 2*
- typedef **l4\_uint32\_t** **Elf32\_Word**  
*size 4 align 4*
- typedef **l4\_int32\_t** **Elf32\_Sword**  
*size 4 align 4*
- typedef **l4\_uint64\_t** **Elf64\_Addr**  
*size 8 align 8*
- typedef **l4\_uint64\_t** **Elf64\_Off**  
*size 8 align 8*
- typedef **l4\_uint16\_t** **Elf64\_Half**  
*size 2 align 2*
- typedef **l4\_uint32\_t** **Elf64\_Word**  
*size 4 align 4*
- typedef **l4\_int32\_t** **Elf64\_Sword**  
*size 4 align 4*
- typedef **l4\_uint64\_t** **Elf64\_Xword**  
*size 8 align 8*
- typedef **l4\_int64\_t** **Elf64\_Sxword**  
*size 8 align 8*

### 14.16.9.1 Detailed Description

Functions and types related to ELF binaries.

### 14.16.9.2 Macro Definition Documentation

#### 14.16.9.2.1 ELF32\_R\_TYPE

```
#define ELF32_R_TYPE(  
    i)
```

**Value:**

```
((unsigned char)(i))
```

**See also**

[Elf\\_R\\_386s](#).

Definition at line [663](#) of file [elf.h](#).

#### 14.16.9.2.2 ELF32\_ST\_BIND

```
#define ELF32_ST_BIND(  
    i)
```

**Value:**

```
((i) >> 4)
```

**See also**

[Elf\\_STBs](#).

Definition at line [893](#) of file [elf.h](#).

#### 14.16.9.2.3 ELF32\_ST\_TYPE

```
#define ELF32_ST_TYPE(  
    i)
```

**Value:**

```
((i) & 0xf)
```

**See also**

[Elf\\_STTs](#).

Definition at line [896](#) of file [elf.h](#).

#### 14.16.9.2.4 ELF64\_R\_TYPE

```
#define ELF64_R_TYPE(  
    i)
```

**Value:**

```
((i) & 0xffffffffL)
```

**See also**

[Elf\\_R\\_386s](#).

Definition at line [671](#) of file [elf.h](#).

#### 14.16.9.2.5 ELF64\_ST\_BIND

```
#define ELF64_ST_BIND(  
    i)
```

**Value:**

```
((i) >> 4)
```

**See also**

[Elf\\_STBs](#)

Definition at line [902](#) of file [elf.h](#).

#### 14.16.9.2.6 ELF64\_ST\_TYPE

```
#define ELF64_ST_TYPE(  
    i)
```

**Value:**

```
((i) & 0xf)
```

**See also**

[Elf\\_STTs](#)

Definition at line [905](#) of file [elf.h](#).

### 14.16.9.3 Enumeration Type Documentation

#### 14.16.9.3.1 anonymous enum

```
anonymous enum
```

**Enumerator**

---

EL_NIDENT	Number of characters.
-----------	-----------------------

Definition at line 117 of file [elf.h](#).

#### 14.16.9.3.2 Elf\_ARM\_SBs

enum [Elf\\_ARM\\_SBs](#)

ARM-specific program header flags.

##### Enumerator

PF_ARM_SB	Segment contains the location addressed by the static base.
-----------	-------------------------------------------------------------

Definition at line 770 of file [elf.h](#).

#### 14.16.9.3.3 Elf\_ATs

enum [Elf\\_ATs](#)

Legal values for [Elf32\\_Auxv.atype](#) / [Elf64\\_Auxv.atype](#).

##### Enumerator

AT_NULL	End of vector.
AT_IGNORE	Entry should be ignored.
AT_EXECFD	File descriptor of program.
AT_PHDR	Program headers for program.
AT_PHEMT	Size of program header entry.
AT_PHNUM	Number of program headers.
AT_PAGESZ	System page size.
AT_BASE	Base address of interpreter.
AT_FLAGS	Flags.
AT_ENTRY	Entry point of program.
AT_NOTELF	Program is not ELF.
AT_UID	Real UID.
AT_EUID	Effective UID.
AT_GID	Real GID.
AT_EGID	Effective GID.
AT_L4_AUX	<a href="#">L4Re</a> AUX section.
AT_L4_ENV	<a href="#">L4Re</a> ENV section.

Definition at line 939 of file [elf.h](#).

#### 14.16.9.3.4 Elf\_CLASSs

enum `Elf_CLASSs`

File class or capacity.

##### Enumerator

---

ELFCLASSNONE	Invalid class.
ELFCLASS32	32-bit object
ELFCLASS64	64-bit object
ELFCLASSNUM	Mask for 32-bit or 64-bit class.

Definition at line 298 of file [elf.h](#).

#### 14.16.9.3.5 Elf\_DATAs

enum [Elf\\_DATAs](#)

Data encoding.

##### Enumerator

ELFDATANONE	invalid data encoding
ELFDATA2LSB	0x01020304 => [ 0x04 0x03 0x02 0x01 ]
ELFDATA2MSB	0x01020304 => [ 0x01 0x02 0x03 0x04 ]
ELFDATANUM	Mask for valid data encoding.

Definition at line 307 of file [elf.h](#).

#### 14.16.9.3.6 Elf\_DF\_1s

enum [Elf\\_DF\\_1s](#)

State flags selectable in the Elf32\_Dyn.d\_un.d\_val / Elf64\_Dyn.d\_un.d\_val element of the DT\_FLAGS\_1 entry in the dynamic section.

##### Enumerator

DF_1_NOW	Set RTLD_NOW for this object.
DF_1_GLOBAL	Set RTLD_GLOBAL for this object.
DF_1_GROUP	Set RTLD_GROUP for this object.
DF_1_NODELETE	Set RTLD_NODELETE for this object.
DF_1_LOADFLTR	Trigger filtee loading at runtime.
DF_1_INITFIRST	Set RTLD_INITFIRST for this object.
DF_1_NOOPEN	Set RTLD_NOOPEN for this object.
DF_1_ORIGIN	\$ORIGIN must be handled.
DF_1_DIRECT	Direct binding enabled.
DF_1_INTERPOSE	Object is used to interpose.
DF_1_NODEFLIB	Ignore default lib search path.
DF_1_NODUMP	Object can't be dldump'ed.

DF_1_CONFALT	Configuration alternative created.
DF_1_ENDFILTEE	Filtee terminates filters search.
DF_1_DISPRELDNE	Disp reloc applied at build time.
DF_1_DISPRELPND	Disp reloc applied at run-time.

Definition at line 594 of file [elf.h](#).

#### 14.16.9.3.7 Elf\_DF\_P1s

enum [Elf\\_DF\\_P1s](#)

Flags in the DT\_POSFLAG\_1 entry effecting only the next DT\_\* entry.

##### Enumerator

DF_P1_LAZYLOAD	Lazyload following object.
DF_P1_GROUPPERM	Symbols from next object are not generally available.

Definition at line 623 of file [elf.h](#).

#### 14.16.9.3.8 Elf\_DFs

enum [Elf\\_DFs](#)

Values of Elf32\_Dyn.d\_un.d\_val, Elf64\_Dyn.d\_un.d\_val in the DT\_FLAGS entry.

##### Enumerator

DF_ORIGIN	Object may use DF_ORIGIN.
DF_SYMBOLIC	Symbol resolutions starts here.
DF_TEXTREL	Object contains text relocations.
DF_BIND_NOW	No lazy binding for this object.
DF_STATIC_TLS	Module uses the static TLS model.

Definition at line 581 of file [elf.h](#).

#### 14.16.9.3.9 Elf\_DTs

enum [Elf\\_DTs](#)

Dynamic Array Tags.

See also

[Elf32\\_Dyn.d\\_tag](#), [Elf64\\_Dyn.d\\_tag](#).

##### Enumerator

---



DT_NULL	end of _DYNAMIC array
DT_NEEDED	name of a needed library
DT_PLTRELSZ	total size of relocation entry
DT_PLTGOT	address assoc with prog link table
DT_HASH	address of symbol hash table
DT_STRTAB	address of string table
DT_SYMTAB	address of symbol table
DT_RELA	address of relocation table
DT_RELASZ	total size of relocation table
DT_RELAENT	size of DT_RELA relocation entry
DT_STRSZ	size of the string table
DT_SYMENT	size of a symbol table entry
DT_INIT	address of initialization function
DT_FINI	address of termination function
DT_SONAME	name of the shared object
DT_RPATH	search library path
DT_SYMBOLIC	alter symbol resolution algorithm
DT_REL	address of relocation table
DT_RELSZ	total size of DT_REL relocation table
DT_RELENT	size of the DT_REL relocation entry
DT_PTRREL	type of relocation entry
DT_DEBUG	for debugging purposes
DT_TEXTREL	at least on entry changes r/o section
DT_JMPREL	address of relocation entries
DT_BIND_NOW	Process relocations of object.
DT_INIT_ARRAY	Array with addresses of init fct.
DT_FINI_ARRAY	Array with addresses of fini fct.
DT_INIT_ARRAYSZ	Size in bytes of DT_INIT_ARRAY.
DT_FINI_ARRAYSZ	Size in bytes of DT_FINI_ARRAY.
DT_RUNPATH	Library search path.
DT_FLAGS	Flags for the object being loaded.
DT_ENCODING	Start of encoded range.
DT_PREINIT_ARRAY	Array with addresses of preinit fct.
DT_PREINIT_ARRAYSZ	size in bytes of DT_PREINIT_ARRAY
DT_NUM	Number used.
DT_LOOS	Start of OS-specific.
DT_HIOS	End of OS-specific.
DT_LOPROC	processor-specific
DT_HIPROC	processor-specific

Definition at line 535 of file [elf.h](#).

#### 14.16.9.3.10 Elf\_EF\_ARM\_s

enum [Elf\\_EF\\_ARM\\_s](#)

ARM specific declarations.

Processor specific flags for the ELF header e\_flags field.

##### Enumerator

EF_ARM_ALIGN8	8-bit structure alignment is in use
---------------	-------------------------------------

Definition at line [730](#) of file [elf.h](#).

#### 14.16.9.3.11 Elf\_EIs

enum [Elf\\_EIs](#)

Identification Indices.

See also

[Elf32\\_Ehdr.e\\_ident](#), [Elf64\\_Ehdr.e\\_ident](#)

##### Enumerator

EI_MAG0	file id 0
EI_MAG1	file id 1
EI_MAG2	file id 2
EI_MAG3	file id 3
EI_CLASS	file class
EI_DATA	data encoding
EI_VERSION	file version
EI_OSABI	Operating system / ABI identification.
EI_ABIVERSION	ABI version.
EI_PAD	start of padding bytes

Definition at line [274](#) of file [elf.h](#).

#### 14.16.9.3.12 Elf\_EMs

enum [Elf\\_EMs](#)

Required architecture.

See also

[Elf32\\_Ehdr.e\\_machine](#), [Elf64\\_Ehdr.e\\_machine](#)

##### Enumerator

---

EM_NONE	no machine
EM_M32	AT&T WE 32100.
EM_SPARC	SPARC.
EM_386	Intel 80386.
EM_68K	Motorola 68000.
EM_88K	Motorola 88000.
EM_860	Intel 80860.
EM_MIPS	MIPS RS3000 big-endian.
EM_MIPS_RS4_BE	MIPS RS4000 big-endian.
EM_SPARC64	SPARC 64-bit.
EM_PARISC	HP PA-RISC.
EM_VPP500	Fujitsu VPP500.
EM_SPARC32PLUS	Sun's V8plus.
EM_960	Intel 80960.
EM_PPC	PowerPC.
EM_V800	NEC V800.
EM_FR20	Fujitsu FR20.
EM_RH32	TRW RH-32.
EM_RCE	Motorola RCE.
EM_ARM	Advanced RISC Machines ARM.
EM_ALPHA	Digital Alpha.
EM_SH	Hitachi SuperH.
EM_SPARCV9	SPARC v9 64-bit.
EM_TRICORE	Siemens Tricore embedded processor.
EM_ARC	Argonaut RISC Core, Argonaut Techn Inc.
EM_H8_300	Hitachi H8/300.
EM_H8_300H	Hitachi H8/300H.
EM_H8S	Hitachi H8/S.
EM_H8_500	Hitachi H8/500.
EM_IA_64	HP/Intel IA-64.
EM_MIPS_X	Stanford MIPS-X.
EM_COLDFIRE	Motorola Coldfire.
EM_68HC12	Motorola M68HC12.
EM_X86_64	Advanced Micro Devices x86-64.
EM_PDSP	Sony DSP Processor.
EM_FX66	Siemens FX66 microcontroller.
EM_ST9PLUS	STMicroelectronics ST9+ 8/16 mc.
EM_ST7	STmicroelectronics ST7 8 bit mc.
EM_68HC16	Motorola MC68HC16 microcontroller.
EM_68HC11	Motorola MC68HC11 microcontroller.
EM_68HC08	Motorola MC68HC08 microcontroller.
EM_68HC05	Motorola MC68HC05 microcontroller.

EM_SVX	Silicon Graphics SVx.
EM_ST19	STMicroelectronics ST19 8 bit mc.
EM_VAX	Digital VAX.
EM_CRIS	Axis Communications 32-bit embedded processor.
EM_JAVELIN	Infineon Technologies 32-bit embedded processor.
EM_FIREPATH	Element 14 64-bit DSP Processor.
EM_ZSP	LSI Logic 16-bit DSP Processor.
EM_MMIX	Donald Knuth's educational 64-bit processor.
EM_HUANY	Harvard University machine-independent object files.
EM_PRISM	SiTera Prism.
EM_AVR	Atmel AVR 8-bit microcontroller.
EM_FR30	Fujitsu FR30.
EM_D10V	Mitsubishi D10V.
EM_D30V	Mitsubishi D30V.
EM_V850	NEC v850.
EM_M32R	Mitsubishi M32R.
EM_MN10300	Matsushita MN10300.
EM_MN10200	Matsushita MN10200.
EM_PJ	picoJava
EM_OPENRISC	OpenRISC 32-bit embedded processor.
EM_ARC_A5	ARC Cores Tangent-A5.
EM_XTENSA	Tensilica Xtensa Architecture.
EM_ALTERA_NIOS2	Altera Nios II.
EM_AARCH64	ARM AARCH64.
EM_TILEPRO	Tilera TILEPro.
EM_MICROBLAZE	Xilinx MicroBlaze.
EM_TILEGX	Tilera TILE-Gx.
EM_RISCV	RISC-V.

Definition at line 183 of file [elf.h](#).

#### 14.16.9.3.13 Elf\_ETs

```
enum Elf_ETs
```

Object file type.

See also

[Elf32\\_Ehdr.e\\_type](#), [Elf64\\_Ehdr.e\\_type](#)

#### Enumerator

---

ET_NONE	no file type
ET_REL	relocatable file
ET_EXEC	executable file
ET_DYN	shared object file
ET_CORE	core file
ET_LOPROC	processor-specific
ET_HIPROC	processor-specific

Definition at line 168 of file [elf.h](#).

#### 14.16.9.3.14 Elf\_EVs

enum [Elf\\_EVs](#)

Object file version.

See also

[Elf32\\_Ehdr.e\\_version](#), [Elf64\\_Ehdr.e\\_version](#)

##### Enumerator

EV_NONE	Invalid version.
EV_CURRENT	Current version.

Definition at line 266 of file [elf.h](#).

#### 14.16.9.3.15 Elf\_MAGs

enum [Elf\\_MAGs](#)

Magic number.

##### Enumerator

ELFMAG0	e_ident[EI_MAG0]
ELFMAG1	e_ident[EI_MAG1]
ELFMAG2	e_ident[EI_MAG2]
ELFMAG3	e_ident[EI_MAG3]

Definition at line 289 of file [elf.h](#).

#### 14.16.9.3.16 Elf\_NTscore

enum [Elf\\_NTscore](#)

Legal values for note segment descriptor types for core files.

##### Enumerator

---

NT_PRSTATUS	Contains copy of prstatus struct.
NT_FPREGSET	Contains copy of fpregset struct.
NT_PRPSINFO	Contains copy of prpsinfo struct.
NT_PRXREG	Contains copy of prxregset struct.
NT_TASKSTRUCT	Contains copy of task structure.
NT_PLATFORM	String from sysinfo(SI_PLATFORM).
NT_AUXV	Contains copy of auxv array.
NT_GWINDOWS	Contains copy of gwindows struct.
NT_ASRS	Contains copy of asrset struct.
NT_PSTATUS	Contains copy of pstatus struct.
NT_PSINFO	Contains copy of psinfo struct.
NT_PRCRED	Contains copy of prcred struct.
NT_UTSNAME	Contains copy of utsname struct.
NT_LWPSTATUS	Contains copy of lwpstatus struct.
NT_LWPSINFO	Contains copy of lwpinfo struct.
NT_PRFPXREG	Contains copy of fprxregset struct.

Definition at line 486 of file [elf.h](#).

#### 14.16.9.3.17 Elf\_NTs\_obj

enum [Elf\\_NTs\\_obj](#)

Legal values for the note segment descriptor types for object files.

##### Enumerator

NT_VERSION	Contains a version string.
------------	----------------------------

Definition at line 507 of file [elf.h](#).

#### 14.16.9.3.18 Elf\_OSABIs

enum [Elf\\_OSABIs](#)

Identify operating system and ABI to which the object is targeted.

##### Enumerator

ELFOSABI_NONE	UNIX System V ABI.
ELFOSABI_SYSV	Alias.
ELFOSABI_HPUX	HP-UX.
ELFOSABI_NETBSD	NetBSD.

ELFOSABI_LINUX	Linux.
ELFOSABI_SOLARIS	Sun Solaris.
ELFOSABI_AIX	IBM AIX.
ELFOSABI_IRIX	SGI Irix.
ELFOSABI_FREEBSD	FreeBSD.
ELFOSABI_TRU64	Compaq TRU64 UNIX.
ELFOSABI_MODESTO	Novell Modesto.
ELFOSABI_OPENBSD	OpenBSD.
ELFOSABI_ARM	ARM.
ELFOSABI_STANDALONE	Standalone (embedded) application.

Definition at line 316 of file [elf.h](#).

#### 14.16.9.3.19 ELF\_PFs

enum [ELF\\_PFs](#)

Segment permissions.

##### Enumerator

PF_X	Executable.
PF_W	Write.
PF_R	Read.
PF_MASKOS	OS-specific.
PF_MASKPROC	Processor-specific.

Definition at line 476 of file [elf.h](#).

#### 14.16.9.3.20 Elf\_PTs

enum [Elf\\_PTs](#)

Segment types.

##### Enumerator

PT_NULL	array is unused
PT_LOAD	loadable
PT_DYNAMIC	dynamic linking information
PT_INTERP	path to interpreter

PT_NOTE	auxiliary information
PT_SHLIB	reserved
PT_PHDR	location of the pht itself
PT_TLS	Thread-local storage segment.
PT_NUM	Number of defined types.
PT_LOOS	OS-specific.
PT_HIOS	OS-specific.
PT_LOPROC	processor-specific
PT_HIPROC	processor-specific
PT_GNU_EH_FRAME	EH frame information.
PT_GNU_STACK	Flags for stack.
PT_GNU_RELRO	Read only after reloc.
PT_L4_STACK	Address of the stack.
PT_L4_AUX	Address of the AUX structures.

Definition at line 451 of file [elf.h](#).

#### 14.16.9.3.21 Elf\_R\_386\_s

enum [Elf\\_R\\_386\\_s](#)

Relocation types (processor specific).

##### Enumerator

R_386_NONE	none
R_386_32	$S + A$ .
R_386_PC32	$S + A - P$ .
R_386_GOT32	$G + A - P$ .
R_386_PLT32	$L + A - P$ .
R_386_COPY	none
R_386_GLOB_DAT	$S$ .
R_386_JMP_SLOT	$S$ .
R_386_RELATIVE	$B + A$ .
R_386_GOTOFF	$S + A - GOT$ .
R_386_GOTPC	$GOT + A - P$ .
R_386_TLS_TPOFF	Offset in static TLS block.
R_386_TLS_IE	Address of GOT entry for static TLS block offset.
R_386_TLS_GOTIE	GOT entry for static TLS block offset.
R_386_TLS_LE	Offset relative to static TLS block.
R_386_TLS_GD	Direct 32 bit for GNU version of general dynamic thread local data.
R_386_TLS_LDM	Direct 32 bit for GNU version of local dynamic thread local data in LE code.
R_386_TLS_GD_32	Direct 32 bit for general dynamic thread local data.



R_386_TLS_GD_PUSH	Tag for pushl in GD TLS code.
R_386_TLS_GD_CALL	Relocation for call to __tls_get_addr().
R_386_TLS_GD_POP	Tag for popl in GD TLS code.
R_386_TLS_LDM_32	Direct 32 bit for local dynamic thread local data in LE code.
R_386_TLS_LDM_PUSH	Tag for pushl in LDM TLS code.
R_386_TLS_LDM_CALL	Relocation for call to __tls_get_addr() in LDM code.
R_386_TLS_LDM_POP	Tag for popl in LDM TLS code.
R_386_TLS_LDO_32	Offset relative to TLS block.
R_386_TLS_IE_32	GOT entry for negated static TLS block offset.
R_386_TLS_LE_32	Negated offset relative to static TLS block.
R_386_TLS_DTPMOD32	ID of module containing symbol.
R_386_TLS_DTPOFF32	Offset in TLS block.
R_386_TLS_TPOFF32	Negated offset in static TLS block.
R_386_NUM	Keep this the last entry.

Definition at line 677 of file [elf.h](#).

#### 14.16.9.3.22 Elf\_R\_AARCH64\_s

enum [Elf\\_R\\_AARCH64\\_s](#)

AARCH64 relocations.

##### Enumerator

R_AARCH64_NONE	No reloc.
----------------	-----------

Definition at line 825 of file [elf.h](#).

#### 14.16.9.3.23 Elf\_R\_ARM\_s

enum [Elf\\_R\\_ARM\\_s](#)

ARM relocations.

##### Enumerator

R_ARM_NONE	No reloc.
R_ARM_PC24	PC relative 26 bit branch.
R_ARM_ABS32	Direct 32 bit.
R_ARM_REL32	PC relative 32 bit.
R_ARM_ABS16	Direct 16 bit.
R_ARM_ABS12	Direct 12 bit.

R_ARM_ABS8	Direct 8 bit.
R_ARM_COPY	Copy symbol at runtime.
R_ARM_GLOB_DAT	Create GOT entry.
R_ARM_JUMP_SLOT	Create PLT entry.
R_ARM_RELATIVE	Adjust by program base.
R_ARM_GOTOFF	32 bit offset to GOT
R_ARM_GOTPC	32 bit PC relative offset to GOT
R_ARM_GOT32	32 bit GOT entry
R_ARM_PLT32	32 bit PLT address
R_ARM_THM_PC11	thumb unconditional branch
R_ARM_THM_PC9	thumb conditional branch
R_ARM_NUM	Keep this the last entry.

Definition at line 777 of file [elf.h](#).

#### 14.16.9.3.24 Elf\_R\_X86\_64\_s

enum [Elf\\_R\\_X86\\_64\\_s](#)

AMD x86-64 relocations.

##### Enumerator

R_X86_64_NONE	No reloc.
R_X86_64_64	Direct 64 bit.
R_X86_64_PC32	PC relative 32 bit signed.
R_X86_64_GOT32	32 bit GOT entry
R_X86_64_PLT32	32 bit PLT address
R_X86_64_COPY	Copy symbol at runtime.
R_X86_64_GLOB_DAT	Create GOT entry.
R_X86_64_JUMP_SLOT	Create PLT entry.
R_X86_64_RELATIVE	Adjust by program base.
R_X86_64_GOTPCREL	32 bit signed PC relative offset to GOT
R_X86_64_32	Direct 32 bit zero extended.
R_X86_64_32S	Direct 32 bit sign extended.
R_X86_64_16	Direct 16 bit zero extended.
R_X86_64_PC16	16 bit sign extended pc relative
R_X86_64_8	Direct 8 bit sign extended.
R_X86_64_PC8	8 bit sign extended pc relative
R_X86_64_DTPOFF64	ID of module containing symbol.
R_X86_64_DTPOFF64	Offset in module's TLS block.
R_X86_64_TPOFF64	Offset in initial TLS block.

R_X86_64_TLSD	32 bit signed PC relative offset to two GOT entries for GD symbol
R_X86_64_TLSD	32 bit signed PC relative offset to two GOT entries for LD symbol
R_X86_64_DTPOFF32	Offset in TLS block.
R_X86_64_GOTPOFF	32 bit signed PC relative offset to GOT entry for IE symbol
R_X86_64_TPOFF32	Offset in initial TLS block.

Definition at line 832 of file [elf.h](#).

#### 14.16.9.3.25 Elf\_SHF\_s\_ARM

enum [Elf\\_SHF\\_s\\_ARM](#)

ARM-specific values for [Elf32\\_Shdr.sh\\_flags](#) / [Elf64\\_Shdr.sh\\_flags](#).

##### Enumerator

SHF_ARM_ENTRYSECT	Section contains an entry point.
SHF_ARM_COMDEF	Section may be multiply defined in the input to a link step.

Definition at line 762 of file [elf.h](#).

#### 14.16.9.3.26 Elf\_SHFs

enum [Elf\\_SHFs](#)

Section attribute flags.

##### Enumerator

SHF_WRITE	writable during execution
SHF_ALLOC	section occupies virt memory
SHF_EXECINSTR	code section
SHF_MERGE	Might be merged.
SHF_STRINGS	Contains nul-terminated strings.
SHF_INFO_LINK	`sh_info' contains SHT index */ SHF_LINK_ORDER = 0x80, /**< Preserve order after combining
SHF_OS_NONCONFORMING	Non-standard OS-specific handling required.
SHF_GROUP	Section is member of a group.
SHF_TLS	Section hold thread-local data.
SHF_MASKOS	OS-specific.
SHF_MASKPROC	processor-specific mask

Definition at line 406 of file [elf.h](#).

#### 14.16.9.3.27 Elf\_SHNs

enum [Elf\\_SHNs](#)

Special section indexes.

##### Enumerator

---

SHN_UNDEF	undefined section header entry
SHN_LORESERVE	lower bound of reserved indexes
SHN_LOPROC	lower bound of proc spec entr
SHN_HIPROC	upper bound of proc spec entr
SHN_ABS	absolute values for ref
SHN_COMMON	common symbols
SHN_HIRESERVE	upper bound of reserved indexes

Definition at line 335 of file [elf.h](#).

#### 14.16.9.3.28 Elf\_SHTs

enum [Elf\\_SHTs](#)

Section type.

##### Enumerator

SHT_NULL	inactive section header
SHT_PROGBITS	information defined by program
SHT_SYMTAB	symbol table
SHT_STRTAB	string table
SHT_RELA	reloc entries w/ explicit addends
SHT_HASH	symbol hash table
SHT_DYNAMIC	information for dynamic linking
SHT_NOTE	information that marks the file
SHT_NOBITS	occupies no space in the file
SHT_REL	reloc entries w/o explicit addends
SHT_SHLIB	reserved + unspecified semantics
SHT_DYNSYM	symbol table (dynamic)
SHT_INIT_ARRAY	Array of constructors.
SHT_FINI_ARRAY	Array of destructors.
SHT_PREINIT_ARRAY	Array of pre-constructors.
SHT_GROUP	Section group.
SHT_SYMTAB_SHNDX	Extended section indices.
SHT_NUM	Number of defined types.
SHT_LOOS	Start OS-specific.
SHT_HIOS	End OS-specific.
SHT_LOPROC	Start processor-specific.
SHT_HIPROC	End processor-specific.
SHT_LOUSER	Start application-specific.
SHT_HIUSER	End application-specific.

Definition at line 377 of file [elf.h](#).

#### 14.16.9.3.29 Elf\_STBs

enum [Elf\\_STBs](#)

Symbol Binding.

See also

[ELF32\\_ST\\_BIND](#), [ELF64\\_ST\\_BIND](#)

##### Enumerator

STB_LOCAL	not visible outside object file
STB_GLOBAL	visible to all objects being combined
STB_WEAK	resemble global symbols
STB_LOOS	OS-specific.
STB_HIOS	OS-specific.
STB_LOPROC	Processor-specific.
STB_HIPROC	Processor-specific.

Definition at line [912](#) of file [elf.h](#).

#### 14.16.9.3.30 Elf\_STTs

enum [Elf\\_STTs](#)

Symbol Types.

See also

[ELF32\\_ST\\_TYPE](#), [ELF64\\_ST\\_TYPE](#)

##### Enumerator

STT_NOTYPE	symbol's type not specified
STT_OBJECT	associated with a data object
STT_FUNC	associated with a function or other code
STT_SECTION	associated with a section
STT_FILE	source file name associated with object
STT_LOOS	OS-specific.
STT_HIOS	OS-specific.
STT_LOPROC	processor-specific
STT_HIPROC	processor-specific

Definition at line [925](#) of file [elf.h](#).

### 14.16.10 Kernel Interface Page API

Collaboration diagram for Kernel Interface Page API:



#### Files

- file [kip.h](#)

#### Macros

- `#define l4util_kip_for_each_feature(s)`  
*Cycle through kernel features given in the KIP.*

#### Functions

- `L4_BEGIN_DECLS int l4util_kip_kernel_has_feature (l4_kernel_info_t const *k, char const *str)`  
*Check if kernel supports a feature.*
- `unsigned long l4util_kip_kernel_abi_version (l4_kernel_info_t const *k)`  
*Return kernel ABI version.*

#### 14.16.10.1 Detailed Description

#### 14.16.10.2 Macro Definition Documentation

##### 14.16.10.2.1 l4util\_kip\_for\_each\_feature

```
#define l4util_kip_for_each_feature(  
    s)
```

#### Value:

`l4_kip_for_each_feature(s)`

Cycle through kernel features given in the KIP.

Cycles through all KIP kernel feature strings. `s` must be a character pointer (`char const *`) initialized with `l4_kip_version_string()`.

**Deprecated** Use `l4_kip_for_each_feature()`.

Definition at line 58 of file [kip.h](#).

### 14.16.10.3 Function Documentation

#### 14.16.10.3.1 l4util\_kip\_kernel\_abi\_version()

```
unsigned long l4util_kip_kernel_abi_version (  
    l4_kernel_info_t const * k)
```

Return kernel ABI version.

#### Parameters

---



<i>k</i>	Pointer to the kernel info page (KIP).
----------	----------------------------------------

**Returns**

Kernel ABI version.

References [L4\\_CV](#), and [L4\\_END\\_DECLS](#).

**14.16.10.3.2 l4util\_kip\_kernel\_has\_feature()**

```
L4_BEGIN_DECLS int l4util_kip_kernel_has_feature (  
    l4_kernel_info_t const * k,  
    char const * str)
```

Check if kernel supports a feature.

**Parameters**

<i>k</i>	Pointer to the kernel info page (KIP).
<i>str</i>	Feature name to check.

**Returns**

1 if the kernel supports the feature, 0 if not.

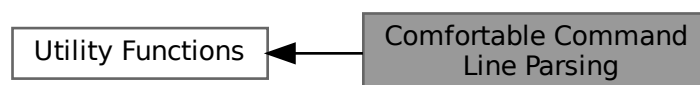
Checks the feature field in the KIP for the given string.

**Deprecated** Use [l4\\_kip\\_kernel\\_has\\_feature\(\)](#).

References [L4\\_CV](#).

**14.16.11 Comfortable Command Line Parsing**

Collaboration diagram for Comfortable Command Line Parsing:



## Typedefs

- typedef void(\* **parse\_cmd\_fn\_t**) (int)  
*Function type for PARSE\_CMD\_FN.*
- typedef void(\* **parse\_cmd\_fn\_arg\_t**) (int, const char \*, int)  
*Function type for PARSE\_CMD\_FN\_ARG.*

## Enumerations

- enum **parse\_cmd\_type**  
*Types for parsing.*

## Functions

- [L4\\_BEGIN\\_DECLS](#) int **parse\_cmdline** (int \*argc, const char \*\*\*argv, int arg0,...)  
*Parse the command-line for specified arguments and store the values into variables.*

### 14.16.11.1 Detailed Description

### 14.16.11.2 Function Documentation

#### 14.16.11.2.1 **parse\_cmdline()**

```
L4_BEGIN_DECLS int parse_cmdline (
    int * argc,
    const char *** argv,
    int arg0,
    ...)
```

Parse the command-line for specified arguments and store the values into variables.

This Functions gets the command-line, and a list of command-descriptors. Then, the command-line is parsed according to the given descriptors, storing strings, switches and numeric arguments at given addresses, and possibly calling specified functions. A default help descriptor is added. Its purpose is to present a short command overview in the case the given command-line does not fit to the descriptors.

Each command-descriptor has the following form:

*short option char, long option name, comment, type, val, addr.*

The *short option char* specifies the short form of the described option. The short form will be recognized after a single dash, or in a group of short options preceeded by a single dash. Specify ' ' if no short form should be used.

The *long option name* specifies the long form of the described option. The long form will be recognized after two dashes. Specify 0 if no long form should be used for this option.

The *comment* is a string that will be used when presenting the short command-line help.

The *type* specifies, if the option should be recognized as

- a number (PARSE\_CMD\_INT),

- a switch (`PARSE_CMD_SWITCH`),
- a string (`PARSE_CMD_STRING`),
- a function call (`PARSE_CMD_FN`, `PARSE_CMD_FN_ARG`),
- an increment/decrement operator (`PARSE_CMD_INC`, `PARSE_CMD_DEC`).

If `type` is `PARSE_CMD_INT`, the option requires a second argument on the command-line after the option. This argument is parsed as a number. It can be preceeded by 0x to present a hex-value or by 0 to present an octal form. `addr` is interpreted as an int-pointer. The scanned argument from the command-line is stored in this pointer.

If `type` is `PARSE_CMD_SWITCH`, `addr` must be a pointer to int, and the value from `val` is stored at this pointer.

With `PARSE_CMD_STRING`, an additional argument is expected at the cmdline. `addr` must be a pointer to `const char*`, and a pointer to the argument on the command line is stored at this pointer. The value in `val` is a default value, which is stored at `addr` if the corresponding option is not given on the command line.

With `PARSE_CMD_FN_ARG`, `addr` is interpreted as a function pointer of type `parse_cmd_fn_t`. It will be called with `val` as argument if the corresponding option is found.

If `type` is `PARSE_CMD_FN_ARG`, `addr` is as a function pointer of type `parse_cmd_fn_arg_t`, and handled similar to `PARSE_CMD_FN`. An additional argument is expected at the command line, however. It is given to the called function as 2nd argument, and parsed as an integer as with `PARSE_CMD_INT` as a third argument.

If `type` is `PARSE_CMD_INC` or `PARSE_CMD_DEC`, `addr` is interpreted as an int-pointer. The value of `val` is stored to this pointer first. For every occurrence of the option in the command line, the integer referenced by `addr` is incremented or decremented, respectively.

The list of command-descriptors is terminated by specifying a binary 0 for the short option char.

Note: The short option char 'h' and the long option name "help" must not be specified. They are used for the default help descriptor and produce a short command-options help when specified on the command-line.

### Parameters

<code>argc</code>	pointer to number of command line parameters as passed to main
<code>argv</code>	pointer to array of command line parameters as passed to main
<code>arg0</code>	format list describing the command line options to parse for

### Returns

0 if the command-line was successfully parsed, otherwise:

- -1 if the given descriptors are somehow wrong.
- -2 if not enough memory was available to hold temporary structs.
- -3 if the given command-line args did not meet the specified set.
- -4 if the help-option was given.

Upon return, `argc` and `argv` point to a list of arguments that were not scanned as arguments. See `getoptlong` for details on scanning.

References [L4\\_CV](#), and [L4\\_END\\_DECLS](#).

### 14.16.12 Random number support

Collaboration diagram for Random number support:



#### Functions

- [l4\\_uint32\\_t l4util\\_rand](#) (void)  
*Deliver next random number.*
- void [l4util\\_srand](#) ([l4\\_uint32\\_t](#) seed)  
*Initialize random number generator.*

#### 14.16.12.1 Detailed Description

#### 14.16.12.2 Function Documentation

##### 14.16.12.2.1 l4util\_rand()

```
l4_uint32_t l4util_rand (
    void )
```

Deliver next random number.

#### Returns

A new random number

References [L4\\_CV](#).

##### 14.16.12.2.2 l4util\_srand()

```
void l4util_srand (
    l4_uint32_t seed)
```

Initialize random number generator.

#### Parameters

<i>seed</i>	Value to initialize
-------------	---------------------

References [L4\\_END\\_DECLS](#).

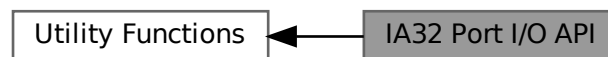
### 14.16.13 Low-Level Thread Functions

Collaboration diagram for Low-Level Thread Functions:



### 14.16.14 IA32 Port I/O API

Collaboration diagram for IA32 Port I/O API:



#### Functions

- [L4\\_BEGIN\\_DECLS](#) `int l4util_ioport_map (l4_cap_idx_t sigma0id, unsigned port_start, unsigned log2size)`  
*Map a range of I/O ports.*
- [l4\\_uint8\\_t](#) `l4util_in8 (l4_uint16_t port)`  
*Read byte from I/O port.*
- [l4\\_uint16\\_t](#) `l4util_in16 (l4_uint16_t port)`  
*Read 16-bit-value from I/O port.*
- [l4\\_uint32\\_t](#) `l4util_in32 (l4_uint16_t port)`  
*Read 32-bit-value from I/O port.*
- `void l4util_ins8 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)`  
*Read a block of 8-bit-values from I/O ports.*
- `void l4util_ins16 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)`  
*Read a block of 16-bit-values from I/O ports.*
- `void l4util_ins32 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)`  
*Read a block of 32-bit-values from I/O ports.*
- `void l4util_out8 (l4_uint8_t value, l4_uint16_t port)`  
*Write byte to I/O port.*
- `void l4util_out16 (l4_uint16_t value, l4_uint16_t port)`  
*Write 16-bit-value to I/O port.*
- `void l4util_out32 (l4_uint32_t value, l4_uint16_t port)`  
*Write 32-bit-value to I/O port.*
- `void l4util_outs8 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)`

*Write a block of bytes to I/O port.*

- void `l4util_outs16` (`l4_uint16_t` port, `l4_umword_t` addr, `l4_umword_t` count)

*Write a block of 16-bit-values to I/O port.*

- void `l4util_outs32` (`l4_uint16_t` port, `l4_umword_t` addr, `l4_umword_t` count)

*Write block of 32-bit-values to I/O port.*

- void `l4util_iodelay` (void)

*delay I/O port access by writing to port 0x80*

#### 14.16.14.1 Detailed Description

#### 14.16.14.2 Function Documentation

##### 14.16.14.2.1 `l4util_in16()`

```
l4_uint16_t l4util_in16 (
    l4_uint16_t port) [inline]
```

Read 16-bit-value from I/O port.

##### Parameters

<i>port</i>	I/O port address
-------------	------------------

##### Returns

value

Definition at line 176 of file `port_io.h`.

##### 14.16.14.2.2 `l4util_in32()`

```
l4_uint32_t l4util_in32 (
    l4_uint16_t port) [inline]
```

Read 32-bit-value from I/O port.

##### Parameters

<i>port</i>	I/O port address
-------------	------------------

##### Returns

value

Definition at line 184 of file `port_io.h`.

##### 14.16.14.2.3 `l4util_in8()`

```
L4_END_DECLS l4_uint8_t l4util_in8 (
    l4_uint16_t port) [inline]
```

Read byte from I/O port.

##### Parameters

<i>port</i>	I/O port address
-------------	------------------

**Returns**

value

Definition at line 168 of file [port\\_io.h](#).

**14.16.14.2.4 l4util\_ins16()**

```
void l4util_ins16 (  
    l4_uint16_t port,  
    l4_umword_t addr,  
    l4_umword_t count) [inline]
```

Read a block of 16-bit-values from I/O ports.

**Parameters**

<i>port</i>	I/O port address
<i>addr</i>	address of buffer
<i>count</i>	number of I/O operations

Definition at line 201 of file [port\\_io.h](#).

**14.16.14.2.5 l4util\_ins32()**

```
void l4util_ins32 (  
    l4_uint16_t port,  
    l4_umword_t addr,  
    l4_umword_t count) [inline]
```

Read a block of 32-bit-values from I/O ports.

**Parameters**

<i>port</i>	I/O port address
<i>addr</i>	address of buffer
<i>count</i>	number of I/O operations

Definition at line 210 of file [port\\_io.h](#).

**14.16.14.2.6 l4util\_ins8()**

```
void l4util_ins8 (  
    l4_uint16_t port,  
    l4_umword_t addr,  
    l4_umword_t count) [inline]
```

Read a block of 8-bit-values from I/O ports.

**Parameters**

<i>port</i>	I/O port address
<i>addr</i>	address of buffer
<i>count</i>	number of I/O operations

Definition at line 192 of file [port\\_io.h](#).

#### 14.16.14.2.7 l4util\_ioport\_map()

```
L4_END_DECLS int l4util_ioport_map (  
    l4_cap_idx_t sigma0id,  
    unsigned port_start,  
    unsigned log2size) [inline]
```

Map a range of I/O ports.

##### Parameters

<i>sigma0id</i>	I/O port service (sigma0).
<i>port_start</i>	(Start) Port to request.
<i>log2size</i>	Log2size of range to request.

##### Returns

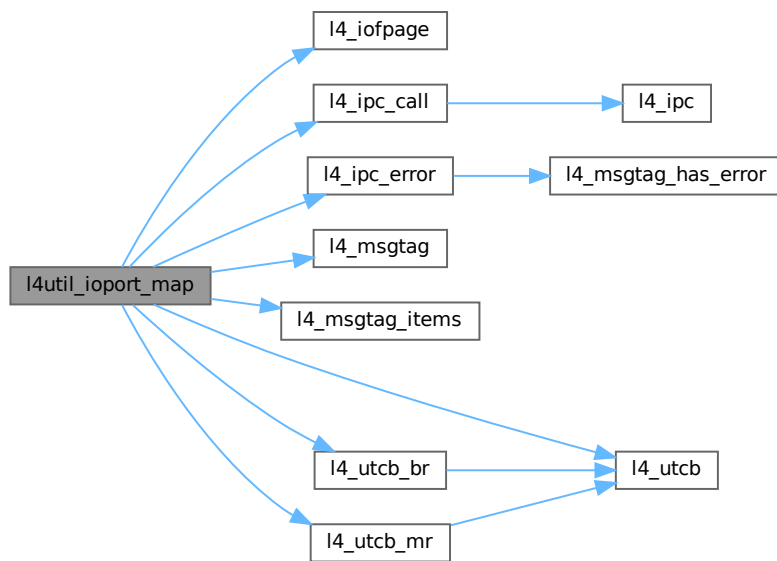
IPC result: 0 if the range could be successfully mapped on error: IPC failure, or -L4\_ENOENT if nothing mapped

Definition at line 54 of file [port\\_io.h](#).

References [l4\\_buf\\_regs\\_t::bdr](#), [l4\\_buf\\_regs\\_t::br](#), [L4\\_ENOENT](#), [l4\\_iofpage\(\)](#), [l4\\_ipc\\_call\(\)](#), [l4\\_ipc\\_error\(\)](#), [L4\\_IPC\\_NEVER](#), [L4\\_ITEM\\_MAP](#), [l4\\_msgtag\(\)](#), [l4\\_msgtag\\_items\(\)](#), [L4\\_PROTO\\_IO\\_PAGE\\_FAULT](#), [l4\\_utcb\(\)](#), [l4\\_utcb\\_br\(\)](#), [l4\\_utcb\\_mr\(\)](#), [l4\\_msg\\_regs\\_t::mr](#), and [l4\\_fpage\\_t::raw](#).



Here is the call graph for this function:



#### 14.16.14.2.8 l4util\_out16()

```
void l4util_out16 (
    l4_uint16_t value,
    l4_uint16_t port) [inline]
```

Write 16-bit-value to I/O port.

##### Parameters

<i>port</i>	I/O port address
<i>value</i>	value to write

Definition at line 225 of file [port\\_io.h](#).

#### 14.16.14.2.9 l4util\_out32()

```
void l4util_out32 (
    l4_uint32_t value,
    l4_uint16_t port) [inline]
```

Write 32-bit-value to I/O port.

##### Parameters

<i>port</i>	I/O port address
<i>value</i>	value to write

Definition at line 231 of file [port\\_io.h](#).

#### 14.16.14.2.10 l4util\_out8()

```
void l4util_out8 (  
    l4_uint8_t value,  
    l4_uint16_t port) [inline]
```

Write byte to I/O port.

##### Parameters

<i>port</i>	I/O port address
<i>value</i>	value to write

Definition at line 219 of file [port\\_io.h](#).

#### 14.16.14.2.11 l4util\_outs16()

```
void l4util_outs16 (  
    l4_uint16_t port,  
    l4_umword_t addr,  
    l4_umword_t count) [inline]
```

Write a block of 16-bit-values to I/O port.

##### Parameters

<i>port</i>	I/O port address
<i>addr</i>	address of buffer
<i>count</i>	number of I/O operations

Definition at line 246 of file [port\\_io.h](#).

#### 14.16.14.2.12 l4util\_outs32()

```
void l4util_outs32 (  
    l4_uint16_t port,  
    l4_umword_t addr,  
    l4_umword_t count) [inline]
```

Write block of 32-bit-values to I/O port.

##### Parameters

<i>port</i>	I/O port address
<i>addr</i>	address of buffer
<i>count</i>	number of I/O operations

Definition at line 255 of file [port\\_io.h](#).

#### 14.16.14.2.13 l4util\_outs8()

```
void l4util_outs8 (
    l4_uint16_t port,
    l4_umword_t addr,
    l4_umword_t count) [inline]
```

Write a block of bytes to I/O port.

#### Parameters

<i>port</i>	I/O port address
<i>addr</i>	address of buffer
<i>count</i>	number of I/O operations

Definition at line 237 of file [port\\_io.h](#).

## 14.17 Virtio Net Switch

A virtual network switch that can be used as defined in the virtio protocol.

#### Data Structures

- class [Mac\\_addr](#)  
*A wrapper class around the value of a MAC address.*
- class [Mac\\_table< Size >](#)  
*Mac\_table manages a 1:n association between ports and MAC addresses.*
- class [Switch\\_factory](#)  
*The IPC interface for creating ports.*
- class [L4virtio\\_port](#)  
*A Port on the Virtio Net Switch.*
- class [Net\\_transfer](#)  
*A network request to only a single destination.*
- class [Virtio\\_net\\_request](#)  
*Abstraction for a network request.*
- class [Virtio\\_switch](#)  
*The Virtio switch contains all ports and processes network requests.*
- struct [Buffer](#)  
*Data buffer used to transfer packets.*
- class [Virtio\\_vlan\\_mangle](#)  
*Class for VLAN packet rewriting.*

### 14.17.1 Detailed Description

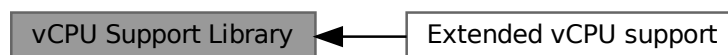
A virtual network switch that can be used as defined in the virtio protocol.

The abstraction of a single connection with a network device (also called client) from the switch's perspective is a port. A client can register multiple ports on the switch. The communication between a client and the switch happens via IRQs, MMIO and shared memory as defined by the Virtio protocol. The switch supports VLANs and ports can be either 'access' or 'trunk' ports. The optionally available monitor port receives network traffic from all ports, and the monitor can not send.

## 14.18 vCPU Support Library

vCPU handling functionality.

Collaboration diagram for vCPU Support Library:



### Topics

- [Extended vCPU support](#) . . . . . [739](#)  
*Extended vCPU handling functionality.*

### Data Structures

- class [L4vcpu::State](#)  
*C++ implementation of state word in the vCPU area.*
- class [L4vcpu::Vcpu](#)  
*C++ implementation of the vCPU save state area.*

### Functions

- void [l4vcpu\\_irq\\_disable](#) ([l4\\_vcpu\\_state\\_t](#) \*vcpu) [L4\\_NOTHROW](#)  
*Disable a vCPU for event delivery.*
- unsigned [l4vcpu\\_irq\\_disable\\_save](#) ([l4\\_vcpu\\_state\\_t](#) \*vcpu) [L4\\_NOTHROW](#)  
*Disable a vCPU for event delivery and return previous state.*
- void [l4vcpu\\_irq\\_enable](#) ([l4\\_vcpu\\_state\\_t](#) \*vcpu, [l4\\_utcb\\_t](#) \*utcb, [l4vcpu\\_event\\_hndl\\_t](#) do\_event\_work\_cb, [l4vcpu\\_setup\\_ipc\\_t](#) setup\_ipc) [L4\\_NOTHROW](#)  
*Enable a vCPU for event delivery.*
- void [l4vcpu\\_irq\\_restore](#) ([l4\\_vcpu\\_state\\_t](#) \*vcpu, unsigned s, [l4\\_utcb\\_t](#) \*utcb, [l4vcpu\\_event\\_hndl\\_t](#) do\_event\_work\_cb, [l4vcpu\\_setup\\_ipc\\_t](#) setup\_ipc) [L4\\_NOTHROW](#)

*Restore a previously saved IRQ/event state.*

- void [l4vcpu\\_wait\\_for\\_event](#) ([l4\\_vcpu\\_state\\_t](#) \*vcpu, [l4\\_utcb\\_t](#) \*utcb, [l4vcpu\\_event\\_hndl\\_t](#) do\_event\_work←\_cb, [l4vcpu\\_setup\\_ipc\\_t](#) setup\_ipc) [L4\\_NOTHROW](#)

*Wait for event.*

- void [l4vcpu\\_print\\_state](#) (const [l4\\_vcpu\\_state\\_t](#) \*vcpu, const char \*prefix) [L4\\_NOTHROW](#)

*Print the state of a vCPU.*

- int [l4vcpu\\_is\\_irq\\_entry](#) ([l4\\_vcpu\\_state\\_t](#) const \*vcpu) [L4\\_NOTHROW](#)

*Return whether the entry reason was an IRQ/IPC message.*

- int [l4vcpu\\_is\\_page\\_fault\\_entry](#) ([l4\\_vcpu\\_state\\_t](#) const \*vcpu) [L4\\_NOTHROW](#)

*Return whether the entry reason was a page fault.*

## 14.18.1 Detailed Description

vCPU handling functionality.

This library provides convenience functionality on top of the l4sys vCPU interface to ease programming. It wraps commonly used code and abstracts architecture depends parts as far as reasonable.

## 14.18.2 Function Documentation

### 14.18.2.1 l4vcpu\_irq\_disable()

```
void l4vcpu_irq_disable (
    l4\_vcpu\_state\_t * vcpu) [inline]
```

Disable a vCPU for event delivery.

#### Parameters

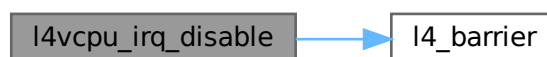
<i>vcpu</i>	Pointer to vCPU area.
-------------	-----------------------

Definition at line 201 of file [vcpu.h](#).

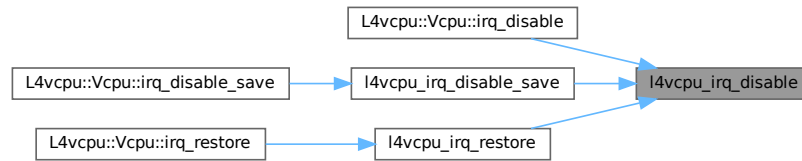
References [l4\\_barrier\(\)](#), [L4\\_NOTHROW](#), and [L4\\_VCPU\\_F\\_IRQ](#).

Referenced by [L4vcpu::Vcpu::irq\\_disable\(\)](#), [l4vcpu\\_irq\\_disable\\_save\(\)](#), and [l4vcpu\\_irq\\_restore\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 14.18.2.2 l4vcpu\_irq\_disable\_save()

```

unsigned l4vcpu_irq_disable_save (
    l4_vcpu_state_t * vcpu) [inline]
  
```

Disable a vCPU for event delivery and return previous state.

#### Parameters

<i>vcpu</i>	Pointer to vCPU area.
-------------	-----------------------

#### Returns

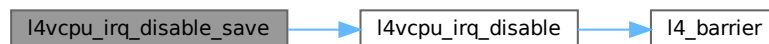
IRQ state before disabling IRQs.

Definition at line 209 of file [vcpu.h](#).

References [L4\\_NOTHROW](#), and [l4vcpu\\_irq\\_disable\(\)](#).

Referenced by [L4vcpu::Vcpu::irq\\_disable\\_save\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 14.18.2.3 l4vcpu\_irq\_enable()

```
void l4vcpu_irq_enable (
    l4_vcpu_state_t * vcpu,
    l4_utcb_t * utcb,
    l4vcpu_event_hndl_t do_event_work_cb,
    l4vcpu_setup_ipc_t setup_ipc) [inline]
```

Enable a vCPU for event delivery.

#### Parameters

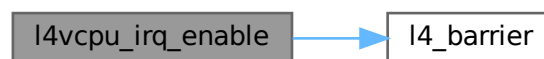
<i>vcpu</i>	Pointer to vCPU area.
<i>utcb</i>	Utcbl pointer of the calling vCPU.
<i>do_event_work_cb</i>	Call-back function that is called in case an event (such as an interrupt) is pending.
<i>setup_ipc</i>	Function call-back that is called right before any IPC operation, and before event delivery is enabled.

Definition at line 232 of file [vcpu.h](#).

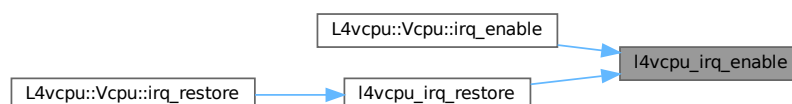
References [l4\\_barrier\(\)](#), [L4\\_IPC\\_BOTH\\_TIMEOUT\\_0](#), [L4\\_LIKELY](#), [L4\\_NOTHROW](#), [L4\\_VCPU\\_F\\_IRQ](#), and [L4\\_VCPU\\_SF\\_IRQ\\_PENDING](#).

Referenced by [L4vcpu::Vcpu::irq\\_enable\(\)](#), and [l4vcpu\\_irq\\_restore\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.18.2.4 l4vcpu\_irq\_restore()

```
void l4vcpu_irq_restore (
    l4_vcpu_state_t * vcpu,
    unsigned s,
    l4_utcb_t * utcb,
    l4vcpu_event_hndl_t do_event_work_cb,
    l4vcpu_setup_ipc_t setup_ipc) [inline]
```

Restore a previously saved IRQ/event state.

##### Parameters

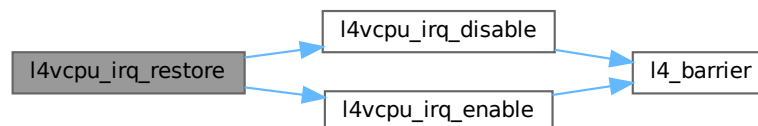
<i>vcpu</i>	Pointer to vCPU area.
<i>s</i>	IRQ state to be restored.
<i>utcb</i>	Utcbl pointer of the calling vCPU.
<i>do_event_work_cb</i>	Call-back function that is called in case an event (such as an interrupt) is pending after enabling.
<i>setup_ipc</i>	Function call-back that is called right before any IPC operation, and before event delivery is enabled.

Definition at line 257 of file [vcpu.h](#).

References [L4\\_NOTHROW](#), [L4\\_VCPU\\_F\\_IRQ](#), [l4vcpu\\_irq\\_disable\(\)](#), and [l4vcpu\\_irq\\_enable\(\)](#).

Referenced by [L4vcpu::Vcpu::irq\\_restore\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:





### 14.18.2.5 l4vcpu\_is\_irq\_entry()

```
int l4vcpu_is_irq_entry (  
    l4_vcpu_state_t const * vcpu) [inline]
```

Return whether the entry reason was an IRQ/IPC message.

#### Parameters

<i>vcpu</i>	Pointer to vCPU area.
-------------	-----------------------

return 0 if not, !=0 otherwise.

References [L4\\_CV](#), [L4\\_INLINE](#), and [L4\\_NOTHROW](#).

Referenced by [L4vcpu::Vcpu::is\\_irq\\_entry\(\)](#).

Here is the caller graph for this function:



### 14.18.2.6 l4vcpu\_is\_page\_fault\_entry()

```
int l4vcpu_is_page_fault_entry (  
    l4_vcpu_state_t const * vcpu) [inline]
```

Return whether the entry reason was a page fault.

#### Parameters

<i>vcpu</i>	Pointer to vCPU area.
-------------	-----------------------

return 0 if not, !=0 otherwise.

References [L4\\_CV](#), and [L4\\_NOTHROW](#).

Referenced by [L4vcpu::Vcpu::is\\_page\\_fault\\_entry\(\)](#).

Here is the caller graph for this function:



### 14.18.2.7 l4vcpu\_print\_state()

```
void l4vcpu_print_state (
    const l4_vcpu_state_t * vcpu,
    const char * prefix)
```

Print the state of a vCPU.

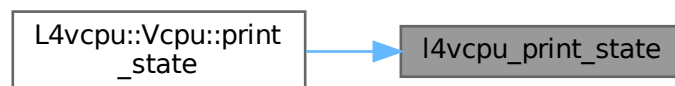
#### Parameters

<i>vcpu</i>	Pointer to vCPU area.
<i>prefix</i>	A prefix for each line printed.

References [L4\\_CV](#), [L4\\_INLINE](#), and [L4\\_NOTHROW](#).

Referenced by [L4vcpu::Vcpu::print\\_state\(\)](#).

Here is the caller graph for this function:



### 14.18.2.8 l4vcpu\_wait\_for\_event()

```
void l4vcpu_wait_for_event (
    l4_vcpu_state_t * vcpu,
    l4_utcb_t * utcb,
    l4vcpu_event_hndl_t do_event_work_cb,
    l4vcpu_setup_ipc_t setup_ipc) [inline]
```

Wait for event.

#### Parameters

<i>vcpu</i>	Pointer to vCPU area.
<i>utcb</i>	UtcB pointer of the calling vCPU.
<i>do_event_work_cb</i>	Call-back function that is called when the vCPU awakes and needs to handle an event/↔ IRQ.
<i>setup_ipc</i>	Function call-back that is called right before any IPC operation.

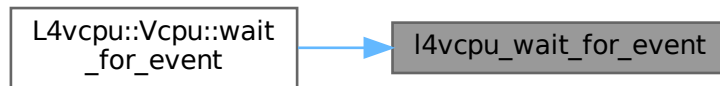
Note that event delivery remains disabled after this function returns.

Definition at line 270 of file [vcpu.h](#).

References [L4\\_IPC\\_NEVER](#), and [L4\\_NOTHROW](#).

Referenced by [L4vcpu::Vcpu::wait\\_for\\_event\(\)](#).

Here is the caller graph for this function:



### 14.18.3 Extended vCPU support

Extended vCPU handling functionality.

Collaboration diagram for Extended vCPU support:



#### Functions

- [int l4vcpu\\_ext\\_alloc](#) ([l4\\_vcpu\\_state\\_t](#) \*\*vcpu, [l4\\_addr\\_t](#) \*ext\_state, [l4\\_cap\\_idx\\_t](#) task, [l4\\_cap\\_idx\\_t](#) regmgr)  
[L4\\_NOTHROW](#)

*Allocate state area for an extended vCPU.*

#### 14.18.3.1 Detailed Description

Extended vCPU handling functionality.

#### 14.18.3.2 Function Documentation

##### 14.18.3.2.1 l4vcpu\_ext\_alloc()

```

int l4vcpu_ext_alloc (
    l4_vcpu_state_t ** vcpu,
    l4_addr_t * ext_state,
    l4_cap_idx_t task,
    l4_cap_idx_t regmgr)
  
```

Allocate state area for an extended vCPU.

#### Parameters

out	<i>vcpu</i>	Allocated vcpu-state area.
out	<i>ext_state</i>	Allocated extended vcpu-state area.
	<i>task</i>	Task to use for allocation.
	<i>regmgr</i>	Region manager to use for allocation.

**Returns**

0 for success, error code otherwise

References [L4\\_CV](#), [L4\\_INLINE](#), and [L4\\_NOTHROW](#).

# Chapter 15

## Namespace Documentation

### 15.1 cxx Namespace Reference

Our C++ library.

#### Namespaces

- namespace [Bits](#)  
*Internal helpers for the cxx package.*

#### Data Structures

- class [Avl\\_map](#)  
*AVL tree based associative container.*
- class [Avl\\_set](#)  
*AVL set for simple comparable items.*
- class [Avl\\_tree\\_node](#)  
*Node of an AVL tree.*
- class [Avl\\_tree](#)  
*A generic AVL tree.*
- class [Bitfield](#)  
*Definition for a member (part) of a bit field.*
- class [Bitmap\\_base](#)  
*Basic bitmap abstraction.*
- class [Bitmap](#)  
*A static bitmap.*
- class [H\\_list\\_item\\_t](#)  
*Basic element type for a double-linked [H\\_list](#).*
- class [H\\_list](#)  
*General double-linked list of unspecified [cxx::H\\_list\\_item](#) elements.*
- struct [H\\_list\\_t](#)  
*Double-linked list of typed [H\\_list\\_item\\_t](#) elements.*
- class [List\\_item](#)  
*Basic list item.*

- class [List](#)  
*Doubly linked list, with internal allocation.*
- class [List\\_alloc](#)  
*Standard list-based allocator.*
- struct [Pair](#)  
*Pair of two values.*
- class [Pair\\_first\\_compare](#)  
*Comparison functor for [Pair](#).*
- class [Ref\\_ptr](#)  
*A reference-counting pointer with automatic cleanup.*
- struct [Ref\\_obj\\_list\\_item](#)  
*Item for list linked via [cxx::Ref\\_ptr](#) with default reference counting.*
- class [Base\\_slab](#)  
*Basic slab allocator.*
- class [Slab](#)  
*Slab allocator for object of type `Type`.*
- class [Base\\_slab\\_static](#)  
*Merged slab allocator (allocators for objects of the same size are merged together).*
- class [Slab\\_static](#)  
*Merged slab allocator (allocators for objects of the same size are merged together).*
- class [S\\_list](#)  
*Simple single-linked list.*
- class [static\\_vector](#)  
*Simple encapsulation for a dynamically allocated array.*
- class [Nothrow](#)  
*Helper type to distinguish the `operator new` version that does not throw exceptions.*
- class [New\\_allocator](#)  
*Standard allocator based on `operator new ()`.*
- struct [Lt\\_functor](#)  
*Generic comparator class that defaults to the less-than operator.*
- class [String](#)  
*Allocation free string class with explicit length field.*
- class [Weak\\_ref\\_base](#)  
*Generic (base) weak reference to some object.*
- class [Weak\\_ref](#)  
*Typed weak reference to an object of type `T`.*

## Typedefs

- typedef [H\\_list\\_item\\_t](#)< void > [H\\_list\\_item](#)  
*Untyped list item.*
- template<typename T>  
using [Ref\\_ptr\\_list\\_item](#) = [Bits::Smart\\_ptr\\_list\\_item](#)<T, [cxx::Ref\\_ptr](#)<T> >  
*Item for list linked with [cxx::Ref\\_ptr](#).*
- template<typename T>  
using [Ref\\_ptr\\_list](#) = [Bits::Smart\\_ptr\\_list](#)<[Ref\\_ptr\\_list\\_item](#)<T> >  
*Single-linked list where elements are connected via a [cxx::Ref\\_ptr](#).*
- template<typename T>  
using [Unique\\_ptr\\_list\\_item](#) = [Bits::Smart\\_ptr\\_list\\_item](#)<T, [cxx::unique\\_ptr](#)<T> >  
*Item for list linked with [cxx::unique\\_ptr](#).*
- template<typename T>  
using [Unique\\_ptr\\_list](#) = [Bits::Smart\\_ptr\\_list](#)<[Unique\\_ptr\\_list\\_item](#)<T> >  
*Single-linked list where elements are connected with a [cxx::unique\\_ptr](#).*

## Functions

- `template<typename A, typename ... ARGS>`  
`constexpr A const & min (A const &a1, A const &a2, ARGS const &...a)`  
*Get the minimum of  $a_1$  and  $a_2$  up to  $a_N$ .*
- `template<typename A, typename ... ARGS>`  
`constexpr A const & min (cxx::identity_t< A > const &a1, cxx::identity_t< A > const &a2, ARGS const &...a)`  
*Get the minimum of  $a_1$  and  $a_2$  up to  $a_N$ .*
- `template<typename A, typename ... ARGS>`  
`constexpr A const & max (A const &a1, A const &a2, ARGS const &...a)`  
*Get the maximum of  $a_1$  and  $a_2$  up to  $a_N$ .*
- `template<typename A, typename ... ARGS>`  
`constexpr A const & max (cxx::identity_t< A > const &a1, cxx::identity_t< A > const &a2, ARGS const &...a)`  
*Get the maximum of  $a_1$  and  $a_2$  up to  $a_N$ .*
- `template<typename T1>`  
`T1 clamp (T1 v, T1 lo, T1 hi)`  
*Limit  $v$  to the range given by  $lo$  and  $hi$ .*
- `template<typename T>`  
`constexpr T gcd (T a, T b)`  
*Compute the greatest common divisor of two unsigned values.*
- `template<typename T>`  
`constexpr T lcm (T a, T b)`  
*Compute the least common multiple of two unsigned values.*
- `template<typename T>`  
`T access\_once (T const *a)`  
*Read the value at an address at most once.*
- `template<typename T, typename VAL>`  
`void write\_now (T *a, VAL &&val)`  
*Write a value at an address exactly once.*

### 15.1.1 Detailed Description

Our C++ library.

Small Low-Level C++ Library.

Strings.

Various kinds of C++ utilities.

### 15.1.2 Function Documentation

#### 15.1.2.1 `access_once()`

```
template<typename T>
T cxx::access_once (
    T const * a) [inline]
```

Read the value at an address at most once.

The read might be omitted if the result is not used by any code unless `typename` contains `volatile`. If the read operation has side effects and must not be omitted, use different means like [L4drivers::Mmio\\_register\\_block](#) or similar.

The compiler is disallowed to reuse a previous read at the same address, for example:

```
val1 = *a;
val2 = access_once(a); // compiler may not replace this by val2 = val1;
```

The compiler is also disallowed to repeat the read, for example:

```
val1 = access_once(a);
val2 = val1; // compiler may not replace this by val2 = *a;
```

The above implies that the compiler is also disallowed to move the read out of or into loops.

#### Note

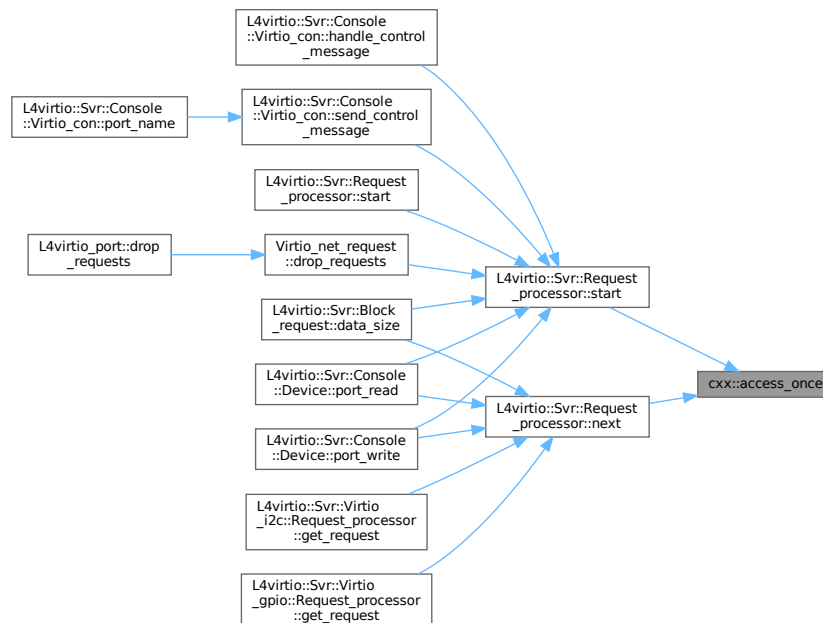
The read might still be moved relative to other code.

The value might be read from a hardware cache, not from RAM.

Definition at line 40 of file [utils](#).

Referenced by [L4virtio::Svr::Request\\_processor::next\(\)](#), and [L4virtio::Svr::Request\\_processor::start\(\)](#).

Here is the caller graph for this function:





### 15.1.2.2 gcd()

```
template<typename T>
T cxx::gcd (
    T a,
    T b) [constexpr]
```

Compute the greatest common divisor of two unsigned values.

This uses the Euclidean modulo algorithm.

#### Note

Contrary to the C++17 specification, this implementation requires the same unsigned type of both arguments and returns the same type (not a common type). This should be fine for most practical use cases.

Contrary to the C++17 specification, this implementation does not accept signed arguments and negative literals.

#### Template Parameters

<i>T</i>	Unsigned integer type of the values.
----------	--------------------------------------

#### Parameters

<i>a</i>	First input.
<i>b</i>	Second input.

#### Returns

Greatest common divisor of the input values.

Definition at line 34 of file [numeric](#).

Referenced by [lcm\(\)](#).

Here is the caller graph for this function:



### 15.1.2.3 lcm()

```
template<typename T>
T cxx::lcm (
    T a,
    T b) [constexpr]
```

Compute the least common multiple of two unsigned values.

This uses the greatest common divisor to compute the least common multiple.

#### Note

Contrary to the C++17 specification, this implementation requires the same unsigned type of both arguments and returns the same type (not a common type). This should be fine for most practical use cases.

Contrary to the C++17 specification, this implementation does not accept signed arguments and negative literals.

#### Template Parameters

<i>T</i>	Unsigned integer type of the values.
----------	--------------------------------------

#### Parameters

<i>a</i>	First input.
<i>b</i>	Second input.

#### Returns

Least common multiple of the input values.

Definition at line 68 of file [numeric](#).

References [gcd\(\)](#).

Here is the call graph for this function:



### 15.1.2.4 write\_now()

```
template<typename T, typename VAL>
void cxx::write_now (
    T * a,
    VAL && val) [inline]
```

Write a value at an address exactly once.

The compiler is disallowed to skip the write, for example:

```
*a = val;
write_now(a, val); // compiler may not skip this line
```

The compiler is also disallowed to repeat the write.

The above implies that the compiler is also disallowed to move the write out of or into loops.

#### Note

The write might still be moved relative to other code.

The value might be written just to a hardware cache for the moment, not immediately to RAM.

Definition at line 71 of file [utils](#).

## 15.2 cxx::Bits Namespace Reference

Internal helpers for the cxx package.

### Data Structures

- struct [Avl\\_map\\_get\\_key](#)  
*Key-getter for [Avl\\_map](#).*
- struct [Avl\\_set\\_get\\_key](#)  
*Internal, key-getter for [Avl\\_set](#) nodes.*
- class [Base\\_avl\\_set](#)  
*Internal: AVL set with internally managed nodes.*
- class [Bst](#)  
*Basic binary search tree (BST).*
- struct [Direction](#)  
*The direction to go in a binary search tree.*
- class [Bst\\_node](#)  
*Basic type of a node in a binary search tree (BST).*
- class [Basic\\_list](#)  
*Internal: Common functions for all head-based list implementations.*
- class [Smart\\_ptr\\_list\\_item](#)  
*List item for an arbitrary item in a [Smart\\_ptr\\_list](#).*
- class [Smart\\_ptr\\_list](#)  
*List of smart-pointer-managed objects.*

### 15.2.1 Detailed Description

Internal helpers for the cxx package.

## 15.3 L4 Namespace Reference

[L4](#) low-level kernel interface.

### Namespaces

- namespace [Typeid](#)  
*Definition of interface data-type helpers.*
- namespace [lpc](#)  
*IPC related functionality.*
- namespace [lpc\\_svr](#)  
*Helper classes for [L4::Server](#) instantiation.*
- namespace [Types](#)  
*[L4](#) basic type helpers for C++.*

### Data Structures

- struct [Type\\_info](#)  
*Dynamic Type Information for [L4Re](#) Interfaces.*
- struct [Kobject\\_typeid](#)  
*[Meta](#) object for handling access to type information of [Kobjects](#).*
- struct [Kobject\\_typeid< void >](#)  
*Minimalistic ID for `void` interface.*
- class [Kobject\\_t](#)  
*Helper class to create an [L4Re](#) interface class that is derived from a single base class.*
- class [Kobject\\_2t](#)  
*Helper class to create an [L4Re](#) interface class that is derived from two base classes (see [L4::Kobject\\_t](#)).*
- struct [Kobject\\_3t](#)  
*Helper class to create an [L4Re](#) interface class that is derived from three base classes (see [L4::Kobject\\_t](#)).*
- struct [Kobject\\_demand](#)  
*Get the combined server-side resource requirements for all type `T...`*
- struct [Proto\\_t](#)  
*Data type for defining protocol numbers.*
- struct [Kobject\\_x](#)  
*Generic [Kobject](#) inheritance template.*
- class [Vm](#)  
*Virtual machine host address space.*
- class [Arm\\_smccc](#)  
*Wrapper for function calls that follow the ARM SMC/HVC calling convention.*
- class [Cap](#)  
*C++ interface for capabilities.*
- class [Cap\\_base](#)  
*Base class for all kinds of capabilities.*
- struct [Epiface](#)

- Base class for interface implementations.*

  - struct [Epiface\\_t0](#)

*Epiface* mixin for generic Kobject-based interfaces.
  - struct [Irqep\\_t](#)

*Epiface* implementation for interrupt handlers.
  - class [Registry\\_iface](#)

Abstract interface for object registries.
  - struct [Epiface\\_t](#)

*Epiface* implementation for Kobject-based interface implementations.
  - class [Basic\\_registry](#)

This registry returns the corresponding server object based on the label of an [lpc\\_gate](#).
  - class [Server](#)

Basic server loop for handling client requests.
  - class [Debugger](#)

C++ kernel debugger API.
  - class [Exception](#)

*Exception* interface.
  - class [Factory](#)

C++ Factory interface, see [Factory](#) for the C interface.
  - class [Iommu](#)

Interface for IO-MMUs used for DMA remapping.
  - class [lpc\\_gate](#)

The C++ IPC gate interface, see [IPC-Gate API](#) for the C interface.
  - class [Irq\\_eoi](#)

Interface for sending an unmask message to an object.
  - struct [Triggerable](#)

Interface that allows an object to be triggered by some source.
  - class [Irq](#)

C++ *Irq* interface, see [IRQs](#) for the C interface.
  - class [Icu](#)

C++ *Icu* interface, see [Interrupt controller](#) for the C interface.
  - class [Kobject](#)

Base class for all kinds of kernel objects and remote objects, referenced by capabilities.
  - class [Meta](#)

*Meta* interface that shall be implemented by each [L4Re](#) object and gives access to the dynamic type information for [L4Re](#) objects.
  - class [Io\\_pager](#)

*Io\_pager* interface.
  - class [Pager](#)

*Pager* interface including the [Io\\_pager](#) interface.
  - class [Platform\\_control](#)

[L4](#) C++ interface for controlling platform-wide properties, see [Platform Control C API](#) for the C interface.
  - class [Rcv\\_endpoint](#)

Interface for kernel objects that allow to receive IPC from them.
  - class [Scheduler](#)

C++ interface of the [Scheduler](#) kernel object, see [Scheduler](#) for the C interface.
  - struct [Semaphore](#)

C++ Kernel-provided semaphore interface, see [Kernel-provided semaphore](#) for the C interface.
  - class [Smart\\_cap](#)

Smart capability class.
  - class [Task](#)

- C++ interface of the [Task](#) kernel object, see [Task](#) for the C interface.*
- class [Thread](#)

*C++ [L4](#) kernel thread interface, see [Thread](#) for the C interface.*
  - class [Thread\\_group](#)

*C++ [L4](#) kernel thread group interface, see [Thread groups](#) for the C interface.*
  - class [Vcon](#)

*C++ [L4](#) [Vcon](#) interface, see [Virtual Console](#) for the C interface.*
  - class [Poll\\_timeout\\_kipclock](#)

*A polling timeout based on the [L4Re](#) clock.*
  - class [Alloc\\_list](#)

*A simple list-based allocator.*
  - class [IOModifier](#)

*Modifier class for the IO stream.*
  - class [Exception\\_tracer](#)

*Back-trace support for exceptions.*
  - class [Base\\_exception](#)

*Base class for all exceptions, thrown by the [L4Re](#) framework.*
  - class [Runtime\\_error](#)

*[Exception](#) for an abstract runtime error.*
  - class [Out\\_of\\_memory](#)

*[Exception](#) signalling insufficient memory.*
  - class [Element\\_already\\_exists](#)

*[Exception](#) for duplicate element insertions.*
  - class [Unknown\\_error](#)

*[Exception](#) for an unknown condition.*
  - class [Element\\_not\\_found](#)

*[Exception](#) for a failed lookup (element not found).*
  - class [Invalid\\_capability](#)

*Indicates that an invalid object was invoked.*
  - class [Com\\_error](#)

*Error conditions during IPC.*
  - class [Bounds\\_error](#)

*Access out of bounds.*
  - class [Server\\_object](#)

*Abstract server object to be used with [L4::Server](#) and [L4::Basic\\_registry](#).*
  - struct [Server\\_object\\_t](#)

*Base class (template) for server implementing server objects.*
  - struct [Server\\_object\\_x](#)

*Helper class to implement p\_dispatch based server objects.*
  - struct [Irq\\_handler\\_object](#)

*[Server](#) object base class for handling IRQ messages.*
  - class [Lock\\_guard](#)

*Basic lock guard implementation that prevents forgotten unlocks on exit paths from a method or a block of code.*
  - class [String](#)

*A null-terminated string container class.*
  - class [Poll\\_timeout\\_counter](#)

*Evaluate an expression for a maximum number of times.*
  - class [Uart\\_apb](#)

*Driver for the Advanced Peripheral Bus (APB) UART from the Cortex-M System Design Kit (CMSDK).*
  - class [Uart](#)

*[Uart](#) driver abstraction.*

## Typedefs

- typedef int **Opcode**  
*Data type for RPC opcodes.*

## Enumerations

- enum { **PROTO\_ANY** = 0 , **PROTO\_EMPTY** = -19 }

## Functions

- template<typename T>  
**Type\_info** const \* **kobject\_typeid** () noexcept  
*Get the **L4::Type\_info** for the **L4Re** interface given in T.*
- template<typename T, typename F>  
**Cap**< T > **cap\_dynamic\_cast** (**Cap**< F > const &c) noexcept  
*dynamic\_cast for capabilities.*
- template<typename T, typename F>  
**Cap**< T > **cap\_cast** (**Cap**< F > const &c) noexcept  
*static\_cast for capabilities.*
- template<typename T, typename F>  
**Cap**< T > **cap\_reinterpret\_cast** (**Cap**< F > const &c) noexcept  
*reinterpret\_cast for capabilities.*
- template<typename T>  
constexpr T **trunc\_order** (T val, unsigned char order)  
*Round a value down so the given number of lsb is zero.*
- template<typename T>  
constexpr T **round\_order** (T val, unsigned char order)  
*Round a value up so the given number of lsb is zero.*
- template<typename T, typename F, typename SMART>  
**Smart\_cap**< T, SMART > **cap\_cast** (**Smart\_cap**< F, SMART > const &c) noexcept  
*static\_cast for (smart) capabilities.*
- template<typename T, typename F, typename SMART>  
**Smart\_cap**< T, SMART > **cap\_reinterpret\_cast** (**Smart\_cap**< F, SMART > const &c) noexcept  
*reinterpret\_cast for (smart) capabilities.*
- void **throw\_ipc\_exception** (**L4::Cap**< void > const &o, **l4\_msgtag\_t** const &err, **l4\_utcb\_t** \*utcb)  
*Throw an **L4** IPC error as exception.*
- void **throw\_ipc\_exception** (void const \*o, **l4\_msgtag\_t** const &err, **l4\_utcb\_t** \*utcb)  
*Throw an **L4** IPC error as exception.*

## Variables

- **IOModifier** const **hex**  
*Modifies the stream to print numbers as hexadecimal values.*
- **IOModifier** const **dec**  
*Modifies the stream to print numbers as decimal values.*
- **BasicOStream** **cout**  
*Standard output stream.*
- **BasicOStream** **cerr**  
*Standard error stream.*

### 15.3.1 Detailed Description

[L4](#) low-level kernel interface.

### 15.3.2 Enumeration Type Documentation

#### 15.3.2.1 anonymous enum

anonymous enum

##### Enumerator

PROTO_ANY	Default protocol used by <a href="#">Kobject_t</a> and <a href="#">Kobject_x</a> .
PROTO_EMPTY	Empty protocol for empty APIs.

Definition at line 44 of file [\\_\\_typeinfo.h](#).

### 15.3.3 Function Documentation

#### 15.3.3.1 cap\_cast() [1/2]

```
template<typename T, typename F>
Cap< T > L4::cap_cast (
    Cap< F > const & c) [inline], [noexcept]
```

static\_cast for capabilities.

##### Template Parameters

<i>T</i>	The target type of the capability
<i>F</i>	The source type (and is usually implicitly set)

##### Parameters

<i>c</i>	The source capability that shall be casted
----------	--------------------------------------------

##### Returns

A capability typed to the interface *T*.

The use of this cast operator is similar to the `static_cast<>()` for C++ pointers. It does the same type checking and adjustments like C++ does on pointers.

Example code:

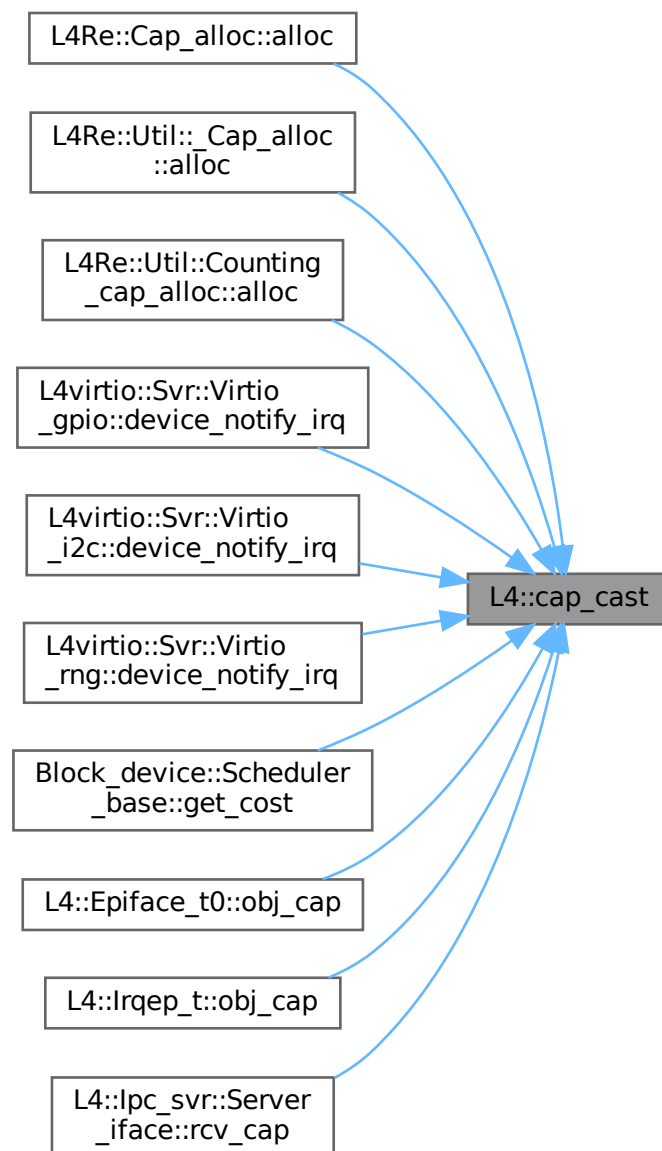


```
L4::Cap<L4::Kobject> obj = ... ;
L4::Cap<L4::Icu> icu = L4::cap_cast<L4::Icu>(obj);
```

Definition at line 416 of file [capability.h](#).

Referenced by [L4Re::Cap\\_alloc::alloc\(\)](#), [L4Re::Util::\\_Cap\\_alloc::alloc\(\)](#), [L4Re::Util::Counting\\_cap\\_alloc< COUNTERTYPE, Dbg >::a](#), [L4virtio::Svr::Virtio\\_gpio< Request\\_handler, Epiface >::device\\_notify\\_irq\(\)](#), [L4virtio::Svr::Virtio\\_i2c< Request\\_handler, Epiface >::d](#), [L4virtio::Svr::Virtio\\_rng< Rnd\\_state, Epiface >::device\\_notify\\_irq\(\)](#), [Block\\_device::Scheduler\\_base< DEV >::get\\_cost\(\)](#), [L4::Epiface\\_t0< RPC\\_IFACE, BASE >::obj\\_cap\(\)](#), [L4::Irqep\\_t< Derived, BASE, bool >::obj\\_cap\(\)](#), and [L4::lpc\\_svr::Server\\_iface::rcv](#).

Here is the caller graph for this function:



### 15.3.3.2 `cap_cast()` [2/2]

```
template<typename T, typename F, typename SMART>
Smart_cap< T, SMART > L4::cap_cast (
    Smart_cap< F, SMART > const & c) [inline], [noexcept]
```

`static_cast` for (smart) capabilities.

#### Template Parameters

<i>T</i>	Type to cast the capability to.
<i>F</i>	(implicit) Type of the passed capability.
<i>SMART</i>	(implicit) Class implementing the <a href="#">Smart_cap</a> interface.

#### Parameters

<i>c</i>	Capability to be casted.
----------	--------------------------

#### Returns

A smart capability with new type *T*.

Definition at line [192](#) of file [smart\\_capability](#).

### 15.3.3.3 `cap_dynamic_cast()`

```
template<typename T, typename F>
Cap< T > L4::cap_dynamic_cast (
    Cap< F > const & c) [inline], [noexcept]
```

`dynamic_cast` for capabilities.

#### Template Parameters

<i>T</i>	The target type of the capability.
<i>F</i>	The source type (is usually implicitly set).

#### Parameters

<i>c</i>	The source capability that shall be casted.
----------	---------------------------------------------

#### Return values

<a href="#">Cap&lt;T&gt;</a>	Capability of target interface <i>T</i> .
------------------------------	-------------------------------------------

<code>L4_INVALID_CAP</code>	<code>c</code> does not support the target interface <code>T</code> or the <a href="#">L4::Meta</a> interface.
-----------------------------	----------------------------------------------------------------------------------------------------------------

The use of this cast operator is similar to the `dynamic_cast<>()` for C++ pointers. It also induces overhead, because it uses the meta interface ([L4::Meta](#)) to do runtime type checking.

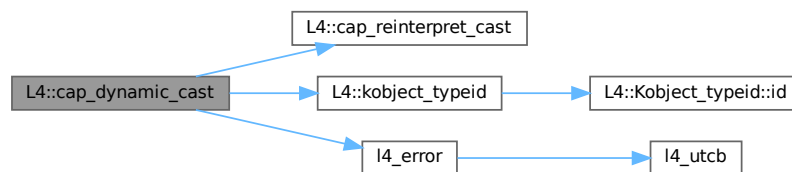
Example code:

```
L4::Cap<L4::Kobject> obj = ... ;
L4::Cap<L4::Icu> icu = L4::cap_dynamic_cast<L4::Icu>(obj);
```

Definition at line 116 of file [capability](#).

References [cap\\_reinterpret\\_cast\(\)](#), [L4::Cap\\_base::Invalid](#), [kobject\\_typeid\(\)](#), and [l4\\_error\(\)](#).

Here is the call graph for this function:



#### 15.3.3.4 cap\_reinterpret\_cast() [1/2]

```
template<typename T, typename F>
Cap< T > L4::cap_reinterpret_cast (
    Cap< F > const & c) [inline], [noexcept]
```

`reinterpret_cast` for capabilities.

##### Template Parameters

<i>T</i>	The target type of the capability
<i>F</i>	The source type (and is usually implicitly set)

##### Parameters

<i>c</i>	The source capability that shall be casted
----------	--------------------------------------------

**Returns**

A capability typed to the interface `T`.

The use of this cast operator is similar to the `reinterpret_cast<>()` for C++ pointers. It does not do any type checking or type adjustment.

Example code:

```
L4::Cap<L4::Kobject> obj = ... ;
L4::Cap<L4::Icu> icu = L4::cap_reinterpret_cast<L4::Icu>(obj);
```

Definition at line 447 of file [capability.h](#).

Referenced by [cap\\_dynamic\\_cast\(\)](#).

Here is the caller graph for this function:

**15.3.3.5 cap\_reinterpret\_cast() [2/2]**

```
template<typename T, typename F, typename SMART>
Smart_cap< T, SMART > L4::cap_reinterpret_cast (
    Smart_cap< F, SMART > const & c) [inline], [noexcept]
```

`reinterpret_cast` for (smart) capabilities.

**Template Parameters**

<i>T</i>	Type to cast the capability to.
<i>F</i>	(implicit) Type of the passed capability.
<i>SMART</i>	(implicit) Class implementing the <a href="#">Smart_cap</a> interface.

**Parameters**

<i>c</i>	Capability to be casted.
----------	--------------------------

**Returns**

A smart capability with new type `T`.

Definition at line 211 of file [smart\\_capability](#).

### 15.3.3.6 round\_order()

```
template<typename T>
T L4::round_order (
    T val,
    unsigned char order) [constexpr]
```

Round a value up so the given number of lsb is zero.

#### Template Parameters

<i>T</i>	The type of the value (shall be some integral type).
----------	------------------------------------------------------

#### Parameters

<i>val</i>	The value to round up to the next multiple of $2^{\text{order}}$ .
<i>order</i>	order ( $2^{\text{order}}$ ) to round up to.

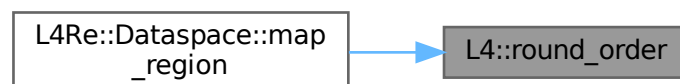
#### Returns

val rounded up to the next  $2^{\text{order}}$ .

Definition at line 32 of file [consts](#).

Referenced by [L4Re::Dataspace::map\\_region\(\)](#).

Here is the caller graph for this function:



### 15.3.3.7 trunc\_order()

```
template<typename T>
T L4::trunc_order (
    T val,
    unsigned char order) [constexpr]
```

Round a value down so the given number of lsb is zero.

#### Template Parameters

---

<i>T</i>	The type of the value (shall be some integral type).
----------	------------------------------------------------------

### Parameters

<i>val</i>	The value where the given lsb shall be masked.
<i>order</i>	the number of least significant bits (lsb) to mask.

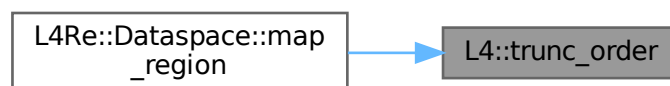
### Returns

val with order lsb masked to zero.

Definition at line 18 of file [consts](#).

Referenced by [L4Re::Dataspace::map\\_region\(\)](#).

Here is the caller graph for this function:



## 15.4 L4::lpc Namespace Reference

IPC related functionality.

### Namespaces

- namespace [Msg](#)  
*IPC Message related functionality.*

### Data Structures

- struct [Array\\_ref](#)  
*Array reference data type for arrays located in the message.*
- struct [Array](#)  
*Array data type for dynamically sized arrays in RPCs.*
- struct [Array\\_in\\_buf](#)  
*Server-side copy in buffer for [Array](#).*
- struct [Call](#)  
*RPC attribute for a standard RPC call.*

- struct [Call\\_zero\\_send\\_timeout](#)  
*RPC attribute for an RPC call, with zero send timeout.*
- struct [Call\\_t](#)  
*RPC attribute for an RPC call with required rights.*
- struct [Send\\_only](#)  
*RPC attribute for a send-only RPC.*
- struct [Ret\\_array](#)  
*Dynamically sized output array of type T.*
- struct [Out](#)  
*Mark an argument as a output value in an RPC signature.*
- struct [In\\_out](#)  
*Mark an argument as in-out argument.*
- struct [As\\_value](#)  
*Pass the argument as plain data value.*
- struct [Opt](#)  
*Attribute for defining an optional RPC argument.*
- class [Small\\_buf](#)  
*A receive item for receiving a single object capability.*
- class [Gen\\_fpage](#)  
*Generic RPC base for typed message items.*
- class [Snd\\_fpage](#)  
*Send item or return item.*
- class [Rcv\\_fpage](#)  
*Non-small receive item.*
- class [Cap](#)  
*Capability type for RPC interfaces (see [L4::Cap<T>](#)).*
- class [Varg](#)  
*Variably sized RPC argument.*
- class [Varg\\_list](#)  
*Self-contained list of variable-sized RPC parameters.*
- class [Varg\\_list\\_ref](#)  
*List of variable-sized RPC parameters as received by the server.*
- class [Str\\_cp\\_in](#)  
*Abstraction for extracting a zero-terminated string from an [lpc::lstream](#).*
- class [Msg\\_ptr](#)  
*Pointer to an element of type T in an [lpc::lstream](#).*
- class [lstream](#)  
*Input stream for IPC unmarshalling.*
- class [Ostream](#)  
*Output stream for IPC marshalling.*
- class [lostream](#)  
*Input/Output stream for IPC [un]marshalling.*

## Typedefs

- typedef unsigned short **Array\_len\_default**  
*Default type for passing length of an array.*

## Functions

- template<typename T>  
**Cap**< T > **make\_cap** (L4::Cap< T > cap, unsigned rights) noexcept  
*Make an L4::lpc::Cap<T> for the given capability and rights.*
- template<typename T>  
**Cap**< T > **make\_cap\_rw** (L4::Cap< T > cap) noexcept  
*Make an L4::lpc::Cap<T> for the given capability with L4\_CAP\_FPAGE\_RW rights.*
- template<typename T>  
**Cap**< T > **make\_cap\_rws** (L4::Cap< T > cap) noexcept  
*Make an L4::lpc::Cap<T> for the given capability with L4\_CAP\_FPAGE\_RWS rights.*
- template<typename T>  
**Cap**< T > **make\_cap\_full** (L4::Cap< T > cap) noexcept  
*Make an L4::IPC::Cap<T> for the given capability with full fpage and object-specific rights.*
- template<typename T>  
**Internal::Buf\_cp\_out**< T > **buf\_cp\_out** (T const \*v, unsigned long size)  
*Insert an array into an lpc::Ostream.*
- template<typename T>  
**Internal::Buf\_cp\_in**< T > **buf\_cp\_in** (T \*v, unsigned long &size)  
*Extract an array from an lpc::Istream.*
- template<typename T>  
**Str\_cp\_in**< T > **str\_cp\_in** (T \*v, unsigned long &size)  
*Create a Str\_cp\_in for the given values.*
- template<typename T>  
**Msg\_ptr**< T > **msg\_ptr** (T \*&p)  
*Create an Msg\_ptr to adjust the given pointer.*
- template<typename T>  
**Internal::Buf\_in**< T > **buf\_in** (T \*&v, unsigned long &size)  
*Return a pointer to stream array data.*
- template<typename T>  
**T read** (Istream &s)  
*Read a value out of a stream.*

### 15.4.1 Detailed Description

IPC related functionality.

### 15.4.2 Function Documentation

#### 15.4.2.1 buf\_cp\_in()

```
template<typename T>
Internal::Buf_cp_in< T > L4::Ipc::buf_cp_in (
    T * v,
    unsigned long & size)
```

Extract an array from an lpc::Istream.

#### Parameters

---



	<i>v</i>	Pointer to the array that shall receive the values from the <a href="#">lpc::lstream</a> .
<i>in, out</i>	<i>size</i>	Input: the number of elements the array can take at most Output: the number of elements found in the stream.

[buf\\_cp\\_in\(\)](#) can be used to extract an array from an [lpc::lstream](#). This is the counterpart [buf\\_cp\\_out\(\)](#). The data from the received message is thereby copied to the given buffer and size is set to the number of elements found in the stream. To avoid the copy operation [buf\\_in\(\)](#) may be used instead.

See also

[buf\\_in\(\)](#) and [buf\\_cp\\_out\(\)](#).

Definition at line 159 of file [ipc\\_stream](#).

#### 15.4.2.2 buf\_cp\_out()

```
template<typename T>
Internal::Buf_cp_out< T > L4::Ipc::buf_cp_out (
    T const * v,
    unsigned long size)
```

Insert an array into an [lpc::Ostream](#).

##### Parameters

<i>v</i>	Pointer to the array that shall be inserted into an <a href="#">lpc::Ostream</a> .
<i>size</i>	Number of elements in the array.

This function inserts an array (e.g. a string) into an [lpc::Ostream](#). The data is copied to the stream. On insertion into the [lpc::Ostream](#) exactly the given number of elements of type T are copied to the message buffer, this means the source buffer is no longer referenced after insertion into the stream.

See also

The counterpart is either [buf\\_cp\\_in\(\)](#) or [buf\\_in\(\)](#).

Definition at line 100 of file [ipc\\_stream](#).

#### 15.4.2.3 buf\_in()

```
template<typename T>
Internal::Buf_in< T > L4::Ipc::buf_in (
    T *& v,
    unsigned long & size)
```

Return a pointer to stream array data.

##### Parameters

out	<i>v</i>	Pointer to the array within the <a href="#">lpc::lstream</a> .
out	<i>size</i>	The number of elements found in the stream.

This routine provides a possibility to extract an array from an [lpc::lstream](#), without extra copy overhead. In contrast to [buf\\_cp\\_in\(\)](#) the data is not copied to a buffer, but a pointer to the array is returned. The user must make sure the UTCB is not used for other purposes while the returned pointer is still in use.

The mechanism is comparable to that of [Msg\\_ptr](#), however it handles arrays inserted with [buf\\_cp\\_out\(\)](#).

See also

[buf\\_cp\\_in\(\)](#) and [buf\\_cp\\_out\(\)](#).

Definition at line 310 of file [ipc\\_stream](#).

#### 15.4.2.4 make\_cap()

```
template<typename T>
Cap< T > L4::Ipc::make_cap (
    L4::Cap< T > cap,
    unsigned rights) [noexcept]
```

Make an [L4::lpc::Cap<T>](#) for the given capability and rights.

#### Template Parameters

<i>T</i>	(IMPLICIT) type of the referenced interface
----------	---------------------------------------------

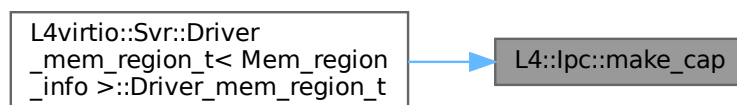
#### Parameters

<i>cap</i>	source capability ( <a href="#">L4::Cap&lt;T&gt;</a> )
<i>rights</i>	rights mask that shall be applied on transfer.

Definition at line 785 of file [ipc\\_types](#).

Referenced by [L4virtio::Svr::Driver\\_mem\\_region\\_t< Mem\\_region\\_info >::Driver\\_mem\\_region\\_t\(\)](#).

Here is the caller graph for this function:



### 15.4.2.5 make\_cap\_full()

```
template<typename T>
Cap< T > L4::Ipc::make_cap_full (
    L4::Cap< T > cap) [noexcept]
```

Make an L4::IPC::Cap<T> for the given capability with full fpage and object-specific rights.

#### Template Parameters

<i>T</i>	(implicit) type of the referenced interface
----------	---------------------------------------------

#### Parameters

<i>cap</i>	source capability (L4::Cap<T>)
------------	--------------------------------

#### See also

[L4\\_cap\\_fpage\\_rights](#)

[L4\\_obj\\_fpage\\_ctl](#)

#### Note

Full rights (including object-specific rights) are required when mapping an IPC gate where the receiver should become the server, i.e. where the receiver wants to call [L4::ipc\\_gate::bind\\_thread\(\)](#) or [L4::ipc\\_gate::bind\\_snd\\_destination\(\)](#).

Definition at line 824 of file [ipc\\_types](#).

References [L4\\_CAP\\_FPAGE\\_RWSD](#), and [L4\\_FPAGE\\_C\\_OBJ\\_RIGHTS](#).

### 15.4.2.6 make\_cap\_rw()

```
template<typename T>
Cap< T > L4::Ipc::make_cap_rw (
    L4::Cap< T > cap) [noexcept]
```

Make an L4::ipc::Cap<T> for the given capability with [L4\\_CAP\\_FPAGE\\_RW](#) rights.

#### Template Parameters

<i>T</i>	(IMPLICIT) type of the referenced interface
----------	---------------------------------------------

#### Parameters

<i>cap</i>	source capability (L4::Cap<T>)
------------	--------------------------------

## Examples

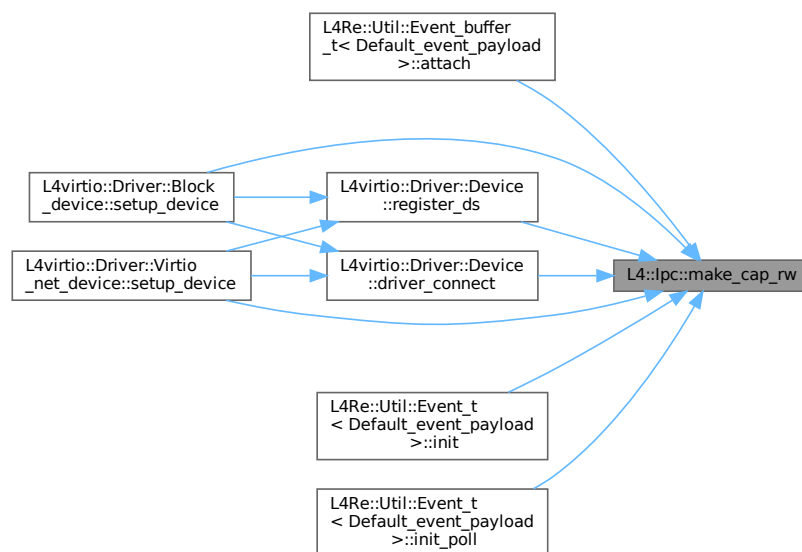
[examples/libs/l4re/c++/mem\\_alloc/ma+rm.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#), and [examples/libs/l4re/c++/shared\\_](#)

Definition at line 795 of file [ipc\\_types](#).

References [L4\\_CAP\\_FPAGE\\_RW](#).

Referenced by [L4Re::Util::Event\\_buffer\\_t< Default\\_event\\_payload >::attach\(\)](#), [L4virtio::Driver::Device::driver\\_connect\(\)](#), [L4Re::Util::Event\\_t< Default\\_event\\_payload >::init\(\)](#), [L4Re::Util::Event\\_t< Default\\_event\\_payload >::init\\_poll\(\)](#), [L4virtio::Driver::Device::register\\_ds\(\)](#), [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#), and [L4virtio::Driver::Virtio\\_net\\_device::setup\\_de](#)

Here is the caller graph for this function:



### 15.4.2.7 make\_cap\_rws()

```

template<typename T>
Cap< T > L4::Ipc::make_cap_rws (
    L4::Cap< T > cap) [noexcept]

```

Make an `L4::ipc::Cap<T>` for the given capability with `L4_CAP_FPAGE_RWS` rights.

#### Template Parameters

<i>T</i>	(IMPLICIT) type of the referenced interface
----------	---------------------------------------------

#### Parameters

---

<i>cap</i>	source capability ( <a href="#">L4::Cap&lt;T&gt;</a> )
------------	--------------------------------------------------------

Definition at line 805 of file [ipc\\_types](#).

References [L4\\_CAP\\_FPAGE\\_RWS](#).

#### 15.4.2.8 msg\_ptr()

```
template<typename T>
Msg_ptr< T > L4::IpC::msg_ptr (
    T *& p)
```

Create an [Msg\\_ptr](#) to adjust the given pointer.

This function makes it more convenient to extract pointers to data in the message buffer itself from an [IpC::Istream](#). This may be used to avoid copy out of large data structures. (See [Msg\\_ptr](#).)

Definition at line 252 of file [ipc\\_stream](#).

#### 15.4.2.9 read()

```
template<typename T>
T L4::IpC::read (
    Istream & s) [inline]
```

Read a value out of a stream.

##### Parameters

<i>s</i>	An <a href="#">Istream</a> .
----------	------------------------------

##### Returns

The value of type *T*.

The stream position is progressed accordingly.

Definition at line 1289 of file [ipc\\_stream](#).

#### 15.4.2.10 str\_cp\_in()

```
template<typename T>
Str_cp_in< T > L4::IpC::str_cp_in (
    T * v,
    unsigned long & size)
```

Create a [Str\\_cp\\_in](#) for the given values.

##### Parameters

	<i>v</i>	Pointer to the array that shall receive the values from the <a href="#">lpc::lstream</a> .
<i>in, out</i>	<i>size</i>	Input: the number of elements the array can take at most Output: the number of elements found in the stream.

This function makes it more convenient to extract arrays from an [lpc::lstream](#) (

See also

[Str\\_cp\\_in](#).)

Definition at line 213 of file [ipc\\_stream](#).

## 15.5 L4::lpc::Msg Namespace Reference

IPC Message related functionality.

### Data Structures

- struct [Elem< Array< A, LEN > >](#)  
*Array* as input arguments.
- struct [Elem< Array< A, LEN > & >](#)  
*Array* as output argument.
- struct [Elem< Array\\_ref< A, LEN > & >](#)  
*Array\_ref* as output argument.
- struct [Dir\\_in](#)  
*Marker type for input values.*
- struct [Dir\\_out](#)  
*Marker type for output values.*
- struct [Cls\\_data](#)  
*Marker type for data values.*
- struct [Cls\\_item](#)  
*Marker type for item values.*
- struct [Cls\\_buffer](#)  
*Marker type for receive buffer values.*
- struct [Do\\_in\\_data](#)  
*Marker for Input data.*
- struct [Do\\_out\\_data](#)  
*Marker for Output data.*
- struct [Do\\_in\\_items](#)  
*Marker for Input items.*
- struct [Do\\_out\\_items](#)  
*Marker for Output items.*
- struct [Do\\_rcv\\_buffers](#)  
*Marker for receive buffers.*
- struct [Svr\\_val\\_ops](#)  
*Defines client-side handling of 'MTYPE' as RPC argument.*
- struct [Is\\_valid\\_rpc\\_type](#)

*Type trait defining a valid RPC parameter type.*

- struct [Svr\\_arg\\_pack](#)

*Server-side RPC arguments data structure used to provide arguments to the server-side implementation of an RPC function.*

- struct [True](#)

*True meta value.*

- struct [False](#)

*False meta value.*

## Enumerations

- enum {  
[Word\\_bytes](#) = sizeof(l4\_umword\_t) , [Item\\_words](#) = 2 , [Item\\_bytes](#) = Word\_bytes \* Item\_words , [Mr\\_words](#) = L4\_UTCB\_GENERIC\_DATA\_SIZE ,  
[Mr\\_bytes](#) = Word\_bytes \* Mr\_words , [Br\\_bytes](#) = Word\_bytes \* L4\_UTCB\_GENERIC\_BUFFERS\_SIZE }

## Functions

- constexpr unsigned long [align\\_to](#) (unsigned long bytes, unsigned long align) noexcept  
*Pad bytes to the given alignment align (in bytes).*
- template<typename T>  
 constexpr unsigned long [align\\_to](#) (unsigned long bytes) noexcept  
*Pad bytes to the alignment of the type T.*
- template<typename T>  
 constexpr bool [check\\_size](#) (unsigned offset, unsigned limit) noexcept  
*Check if there is enough space for T from offset to limit.*
- template<typename T, typename CTYPE>  
 bool [check\\_size](#) (unsigned offset, unsigned limit, CTYPE cnt) noexcept  
*Check if there is enough space for an array of T from offset to limit.*
- template<typename T>  
 int [msg\\_add](#) (char \*msg, unsigned offs, unsigned limit, T v) noexcept  
*Add some data to a message at offs.*
- template<typename T>  
 int [msg\\_get](#) (char \*msg, unsigned offs, unsigned limit, T &v) noexcept  
*Get some data from a message at offs.*

### 15.5.1 Detailed Description

IPC Message related functionality.

### 15.5.2 Enumeration Type Documentation

#### 15.5.2.1 anonymous enum

anonymous enum

#### Enumerator

---

Word_bytes	number of bytes for one message word
Item_words	number of message words for one message item
Item_bytes	number of bytes for one message item
Mr_words	number of message words available in the UTCB
Mr_bytes	number of bytes available in the UTCB message registers
Br_bytes	number of bytes available in the UTCB buffer registers

Definition at line 85 of file [ipc\\_basics](#).

### 15.5.3 Function Documentation

#### 15.5.3.1 `align_to()` [1/2]

```
template<typename T>
unsigned long L4::Ipc::Msg::align_to (
    unsigned long bytes) [constexpr], [noexcept]
```

Pad *bytes* to the alignment of the type *T*.

#### Template Parameters

<i>T</i>	The data type used for the alignment
----------	--------------------------------------

#### Parameters

<i>bytes</i>	The value to add the padding to
--------------	---------------------------------

#### Returns

*bytes* padded to achieve the alignment of *T*.

Definition at line 40 of file [ipc\\_basics](#).

References [align\\_to\(\)](#).

Here is the call graph for this function:





### 15.5.3.2 align\_to() [2/2]

```
unsigned long L4::Ipc::Msg::align_to (
    unsigned long bytes,
    unsigned long align) [constexpr], [noexcept]
```

Pad bytes to the given alignment *align* (in bytes).

#### Parameters

<i>bytes</i>	The input value in bytes
<i>align</i>	The alignment value in bytes

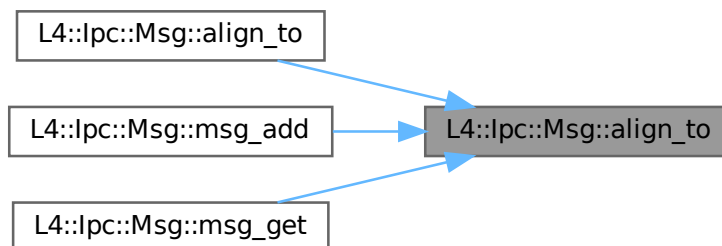
#### Returns

the result after padding *bytes* to *align*.

Definition at line 30 of file [ipc\\_basics](#).

Referenced by [align\\_to\(\)](#), [msg\\_add\(\)](#), and [msg\\_get\(\)](#).

Here is the caller graph for this function:



### 15.5.3.3 check\_size() [1/2]

```
template<typename T>
bool L4::Ipc::Msg::check_size (
    unsigned offset,
    unsigned limit) [constexpr], [noexcept]
```

Check if there is enough space for T from offset to limit.

#### Template Parameters

<i>T</i>	The data type that shall be fitted at <i>offset</i>
----------	-----------------------------------------------------

### Parameters

<i>offset</i>	The current offset in bytes (must already be padded if desired).
<i>limit</i>	The limit in bytes that must not be exceeded after adding the size of <i>T</i> .

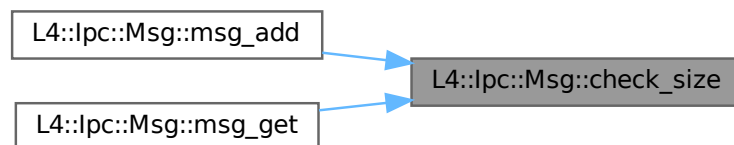
### Returns

true if the limit will not be exceeded, false else.

Definition at line 53 of file [ipc\\_basics](#).

Referenced by [msg\\_add\(\)](#), and [msg\\_get\(\)](#).

Here is the caller graph for this function:



#### 15.5.3.4 check\_size() [2/2]

```

template<typename T, typename CTYPE>
bool L4::Ipc::Msg::check_size (
    unsigned offset,
    unsigned limit,
    CTYPE cnt) [inline], [noexcept]
  
```

Check if there is enough space for an array of T from offset to limit.

### Template Parameters

<i>T</i>	The data type that shall be fitted at <i>offset</i>
<i>CTYPE</i>	Type of the <i>cnt</i> parameter

### Parameters

<i>offset</i>	The current offset in bytes (must already be padded if desired).
<i>limit</i>	The limit in bytes that must not be exceeded after adding <i>cnt</i> times the size of <i>T</i> .
<i>cnt</i>	The number of elements of type <i>T</i> that shall be put at <i>offset</i> .

**Returns**

true if the limit will not be exceeded, false else.

Definition at line 71 of file [ipc\\_basics](#).

References [L4\\_UNLIKELY](#).

**15.5.3.5 msg\_add()**

```
template<typename T>
int L4::IpC::Msg::msg_add (
    char * msg,
    unsigned offs,
    unsigned limit,
    T v) [inline], [noexcept]
```

Add some data to a message at offs.

**Template Parameters**

<i>T</i>	The type of the data to add
----------	-----------------------------

**Parameters**

<i>msg</i>	pointer to the start of the message
<i>offs</i>	The current offset within the message, this shall be padded to the alignment of <i>T</i> if <i>v</i> is added.
<i>limit</i>	The size limit in bytes that offset must not exceed.
<i>v</i>	The value to add to the message

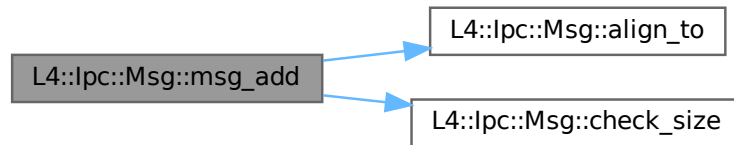
**Returns**

The new offset when successful, a negative value if the given limit will be exceeded.

Definition at line 114 of file [ipc\\_basics](#).

References [align\\_to\(\)](#), [check\\_size\(\)](#), [L4\\_MSGTOOLONG](#), and [L4\\_UNLIKELY](#).

Here is the call graph for this function:



### 15.5.3.6 msg\_get()

```

template<typename T>
int L4::Ipc::Msg::msg_get (
    char * msg,
    unsigned offs,
    unsigned limit,
    T & v) [inline], [noexcept]
  
```

Get some data from a message at offs.

#### Template Parameters

<i>T</i>	The type of the data to read
----------	------------------------------

#### Parameters

<i>msg</i>	Pointer to the start of the message
<i>offs</i>	The current offset within the message, this shall be padded to the alignment of <i>T</i> if a <i>v</i> can be read.
<i>limit</i>	The size limit in bytes that offset must not exceed.
<i>v</i>	A reference to receive the value from the message

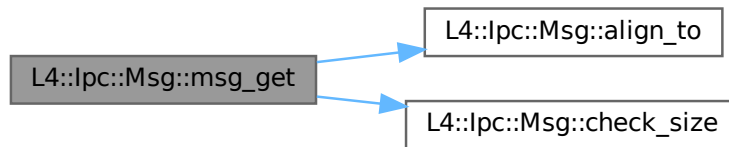
#### Returns

The new offset when successful, a negative value if the given limit will be exceeded.

Definition at line 135 of file `ipc_basics`.

References [align\\_to\(\)](#), [check\\_size\(\)](#), [L4\\_EMSGTOOSHORT](#), and [L4\\_UNLIKELY](#).

Here is the call graph for this function:



## 15.6 L4::lpc\_svr Namespace Reference

Helper classes for [L4::Server](#) instantiation.

### Data Structures

- class [Server\\_iface](#)  
*Interface for server-loop related functions.*
- struct [Ignore\\_errors](#)  
*Mix in for LOOP\_HOOKS to ignore IPC errors.*
- struct [Default\\_timeout](#)  
*Mix in for LOOP\_HOOKS to use a 0 send and a infinite receive timeout.*
- struct [Compound\\_reply](#)  
*Mix in for LOOP\_HOOKS to always use compound reply and wait.*
- struct [Default\\_setup\\_wait](#)  
*Mix in for LOOP\_HOOKS for setup\_wait no op.*
- struct [Direct\\_dispatch](#)  
*Direct dispatch helper, for forwarding dispatch calls to a registry R.*
- struct [Direct\\_dispatch< R \\* >](#)  
*Direct dispatch helper, for forwarding dispatch calls via a pointer to a registry R.*
- struct [Exc\\_dispatch](#)  
*Dispatch helper wrapping try {} catch {} around the dispatch call.*
- struct [Dbg\\_dispatch](#)  
*Dispatch helper that, in addition to what [Exc\\_dispatch](#) does, prints exception messages.*
- class [Br\\_manager\\_no\\_buffers](#)  
*Empty implementation of [Server\\_iface](#).*
- struct [Default\\_loop\\_hooks](#)  
*Default LOOP\_HOOKS.*
- class [Timeout](#)  
*Callback interface for [Timeout\\_queue](#).*
- class [Timeout\\_queue](#)  
*[Timeout](#) queue to be used in l4re server loop.*
- class [Timeout\\_queue\\_hooks](#)  
*Loop hooks mixin for integrating a timeout queue into the server loop.*

## Enumerations

- enum [Reply\\_mode](#) { [Reply\\_compound](#) , [Reply\\_separate](#) }  
*Reply mode for server loop.*

### 15.6.1 Detailed Description

Helper classes for [L4::Server](#) instantiation.

## 15.7 L4::Typeid Namespace Reference

Definition of interface data-type helpers.

### Data Structures

- struct [P\\_dispatch](#)  
*Use for protocol based dispatch stage.*
- struct [Raw\\_ipc](#)  
*RPCs list for passing raw incoming IPC to the server object.*
- struct [Rpc](#)  
*Standard list of RPCs of an interface.*
- struct [Rpc\\_code](#)  
*List of RPCs of an interface using a special opcode type.*
- struct [Rpc\\_nocode](#)  
*List of RPCs of an interface using a single operation without an opcode.*
- struct [Rpc\\_sys](#)  
*List of RPCs typically used for kernel interfaces.*

### 15.7.1 Detailed Description

Definition of interface data-type helpers.

#### Note

These type helpers are intended for internal use, if you look for standard C++ type traits use the `<type_traits>` header for the standard C++ library or use `<l4/cxx/type_traits>`.

## 15.8 L4::Types Namespace Reference

[L4](#) basic type helpers for C++.

## Data Structures

- class [Flags](#)  
*Template for defining typical [Flags](#) bitmaps.*
- struct [Int\\_for\\_size](#)  
*Metafunction to get an unsigned integral type for the given size.*
- struct [Int\\_for\\_type](#)  
*Metafunction to get an integral type of the same size as  $T$ .*
- struct [Flags\\_ops\\_t](#)  
*Mixin class to define a set of friend bitwise operators on  $DT$ .*
- struct [Flags\\_t](#)  
*Template type to define a flags type with bitwise operations.*
- struct [Bool](#)  
*Boolean meta type.*
- struct [False](#)  
*[False](#) meta value.*
- struct [True](#)  
*[True](#) meta value.*
- struct [Same](#)  
*Compare two data types for equality.*

### 15.8.1 Detailed Description

[L4](#) basic type helpers for C++.

## 15.9 L4Re Namespace Reference

[L4Re](#) C++ Interfaces.

## Namespaces

- namespace [Vfs](#)  
*Virtual file system for interfaces in POSIX libc.*
- namespace [Util](#)  
*Documentation of the [L4](#) Runtime Environment utility functionality in C++.*

## Data Structures

- class [Cap\\_alloc](#)  
*Capability allocator interface.*
- class [Smart\\_cap\\_auto](#)  
*Helper for [Unique\\_cap](#) and [Unique\\_del\\_cap](#).*
- class [Smart\\_count\\_cap](#)  
*Helper for [Ref\\_cap](#) and [Ref\\_del\\_cap](#).*
- class [Console](#)  
*[Console](#) class.*
- class [Dataspace](#)

- Interface for memory-like objects.*

  - class [Debug\\_obj](#)

*Debug interface.*
  - class [Dma\\_space](#)

*Managed DMA Address Space.*
  - class [Env](#)

*C++ interface of the initial environment that is provided to an [L4](#) task.*
  - class [Event](#)

*Event class.*
  - struct [Default\\_event\\_payload](#)

*Default event stream payload.*
  - class [Event\\_buffer\\_t](#)

*Event buffer class.*
  - class [Inhibitor](#)

*Set of inhibitor locks, which inhibit specific actions when held.*
  - class [Itas](#)

*Interface to the ITAS.*
  - class [Log](#)

*Log interface class.*
  - class [Mem\\_alloc](#)

*Memory allocation interface.*
  - struct [Mmio\\_space](#)

*Interface for memory-like address space accessible via IPC.*
  - class [Namespace](#)

*Name-space interface.*
  - class [Parent](#)

*Parent interface.*
  - struct [Random](#)

*Low-bandwidth interface for random number generators.*
  - class [Rm](#)

*Region map.*

## Typedefs

- template<typename T>
   
using [Shared\\_cap](#) = L4::Detail::Shared\_cap\_impl<T, [Smart\\_count\\_cap](#)<[L4\\_FP\\_ALL\\_SPACES](#)>>
   
*Shared capability that implements automatic free and unmap of the capability selector.*
- template<typename T>
   
using [shared\\_cap](#) = L4::Detail::Shared\_cap\_impl<T, [Smart\\_count\\_cap](#)<[L4\\_FP\\_ALL\\_SPACES](#)>>
   
*Shared capability that implements automatic free and unmap of the capability selector.*
- template<typename T>
   
using [Shared\\_del\\_cap](#) = L4::Detail::Shared\_cap\_impl<T, [Smart\\_count\\_cap](#)<[L4\\_FP\\_DELETE\\_OBJ](#)>>
   
*Shared capability that implements automatic free and unmap+delete of the capability selector.*
- template<typename T>
   
using [shared\\_del\\_cap](#) = L4::Detail::Shared\_cap\_impl<T, [Smart\\_count\\_cap](#)<[L4\\_FP\\_DELETE\\_OBJ](#)>>
   
*Shared capability that implements automatic free and unmap+delete of the capability selector.*
- template<typename T>
   
using [Unique\\_cap](#) = L4::Detail::Unique\_cap\_impl<T, [Smart\\_cap\\_auto](#)<[L4\\_FP\\_ALL\\_SPACES](#)>>
   
*Unique capability that implements automatic free and unmap of the capability selector.*
- template<typename T>
   
using [unique\\_cap](#) = L4::Detail::Unique\_cap\_impl<T, [Smart\\_cap\\_auto](#)<[L4\\_FP\\_ALL\\_SPACES](#)>>



*Unique capability that implements automatic free and unmap of the capability selector.*

- template<typename T>  
using [Unique\\_del\\_cap](#) = L4::Detail::Unique\_cap\_impl<T, [Smart\\_cap\\_auto](#)<[L4\\_FP\\_DELETE\\_OBJ](#)>>

*Unique capability that implements automatic free and unmap+delete of the capability selector.*

- template<typename T>  
using [unique\\_del\\_cap](#) = L4::Detail::Unique\_cap\_impl<T, [Smart\\_cap\\_auto](#)<[L4\\_FP\\_DELETE\\_OBJ](#)>>

*Unique capability that implements automatic free and unmap+delete of the capability selector.*

## Functions

- void [throw\\_error](#) (long err, char const \*extra="")  
*Generate C++ exception.*
- long [chksys](#) (long err, char const \*extra="", long ret=0)  
*Generate C++ exception on error.*
- long [chksys](#) ([l4\\_msgtag\\_t](#) const &t, char const \*extra="", [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)(), long ret=0)  
*Generate C++ exception on error.*
- long [chksys](#) ([l4\\_msgtag\\_t](#) const &t, [l4\\_utcb\\_t](#) \*utcb, char const \*extra="")  
*Generate C++ exception on error.*
- template<typename T>  
T [chkcap](#) (T &&cap, char const \*extra="", long err=-[L4\\_ENOMEM](#))  
*Check for valid capability or raise C++ exception.*
- [l4\\_msgtag\\_t](#) [chkipc](#) ([l4\\_msgtag\\_t](#) tag, char const \*extra="", [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)())  
*Test a message tag for IPC errors.*
- template<typename T>  
[Shared\\_cap](#)< T > [make\\_shared\\_cap](#) ([L4Re::Cap\\_alloc](#) \*ca)  
*Allocate a capability slot and wrap it in a [Shared\\_cap](#).*
- template<typename T>  
[Shared\\_del\\_cap](#)< T > [make\\_shared\\_del\\_cap](#) ([L4Re::Cap\\_alloc](#) \*ca)  
*Allocate a capability slot and wrap it in a [Shared\\_del\\_cap](#).*
- template<typename T>  
[Unique\\_cap](#)< T > [make\\_unique\\_cap](#) ([L4Re::Cap\\_alloc](#) \*ca)  
*Allocate a capability slot and wrap it in an [Unique\\_cap](#).*
- template<typename T>  
[Unique\\_del\\_cap](#)< T > [make\\_unique\\_del\\_cap](#) ([L4Re::Cap\\_alloc](#) \*ca)  
*Allocate a capability slot and wrap it in an [Unique\\_del\\_cap](#).*

### 15.9.1 Detailed Description

[L4Re](#) C++ Interfaces.

[L4](#) Runtime Environment.

### 15.9.2 Typedef Documentation

#### 15.9.2.1 Shared\_cap

```
template<typename T>
using L4Re::Shared\_cap = L4::Detail::Shared_cap_impl<T, Smart\_count\_cap<L4\_FP\_ALL\_SPACES>>
```

Shared capability that implements automatic free and unmap of the capability selector.

#### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	----------------------------------------------

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero.

#### Note

This type is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::Shared\\_cap](#).

Definition at line 33 of file [shared\\_cap](#).

### 15.9.2.2 shared\_cap

```
template<typename T>
using L4Re::shared_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES>>
```

Shared capability that implements automatic free and unmap of the capability selector.

#### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	----------------------------------------------

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero.

#### Note

This type is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::Shared\\_cap](#).

Definition at line 36 of file [shared\\_cap](#).

### 15.9.2.3 Shared\_del\_cap

```
template<typename T>
using L4Re::Shared_del_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>>
```

Shared capability that implements automatic free and unmap+delete of the capability selector.

#### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	----------------------------------------------

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero. The main difference to [Shared\\_cap](#) is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

#### Note

This type is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::Shared\\_del\\_cap](#).

Definition at line 69 of file [shared\\_cap](#).

#### 15.9.2.4 shared\_del\_cap

```
template<typename T>  
using L4Re::shared_del_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>>
```

Shared capability that implements automatic free and unmap+delete of the capability selector.

##### Template Parameters

---

<i>T</i>	Type of the object the capability refers to.
----------	----------------------------------------------

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero. The main difference to [Shared\\_cap](#) is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

#### Note

This type is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::Shared\\_del\\_cap](#).

Definition at line 72 of file [shared\\_cap](#).

### 15.9.2.5 Unique\_cap

```
template<typename T>
using L4Re::Unique_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>>
```

Unique capability that implements automatic free and unmap of the capability selector.

#### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	----------------------------------------------

The ownership of the capability is managed in the same way as `unique_ptr`.

#### Note

This type is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::Unique\\_cap](#).

Definition at line 31 of file [unique\\_cap](#).

### 15.9.2.6 unique\_cap

```
template<typename T>
using L4Re::unique_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>>
```

Unique capability that implements automatic free and unmap of the capability selector.

#### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	----------------------------------------------

The ownership of the capability is managed in the same way as `unique_ptr`.

#### Note

This type is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::Unique\\_cap](#).

Definition at line 34 of file [unique\\_cap](#).

### 15.9.2.7 Unique\_del\_cap

```
template<typename T>  
using L4Re::Unique_del_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>>
```

Unique capability that implements automatic free and unmap+delete of the capability selector.

#### Template Parameters

---

<i>T</i>	Type of the object the capability refers to.
----------	----------------------------------------------

The main difference to [Unique\\_cap](#) is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

#### Note

This type is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::Unique\\_del\\_cap](#).

Definition at line 64 of file [unique\\_cap](#).

### 15.9.2.8 unique\_del\_cap

```
template<typename T>
using L4Re::unique_del_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>>
```

Unique capability that implements automatic free and unmap+delete of the capability selector.

#### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	----------------------------------------------

The main difference to [Unique\\_cap](#) is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

#### Note

This type is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::Unique\\_del\\_cap](#).

Definition at line 67 of file [unique\\_cap](#).

## 15.9.3 Function Documentation

### 15.9.3.1 chkcap()

```
template<typename T>
T L4Re::chkcap (
    T && cap,
    char const * extra = "",
    long err = -L4_ENOMEM) [inline]
```

Check for valid capability or raise C++ exception.

#### Template Parameters

<i>T</i>	Type of object to check, must be capability-like ( <a href="#">L4::Cap</a> , <a href="#">L4Re::Util::Unique_cap</a> etc.)
----------	---------------------------------------------------------------------------------------------------------------------------

#### Parameters

---

<i>cap</i>	Capability value to check.
<i>extra</i>	Optional text for exception.
<i>err</i>	Error value for exception or 0 if the error code stored in the invalid capability should be used.

This function checks whether the capability is valid. If the capability is invalid, a C++ exception is generated, using *err* if *err* is not zero, otherwise the capability value is used. A valid capability will just be returned.

Definition at line 149 of file [error\\_helper](#).

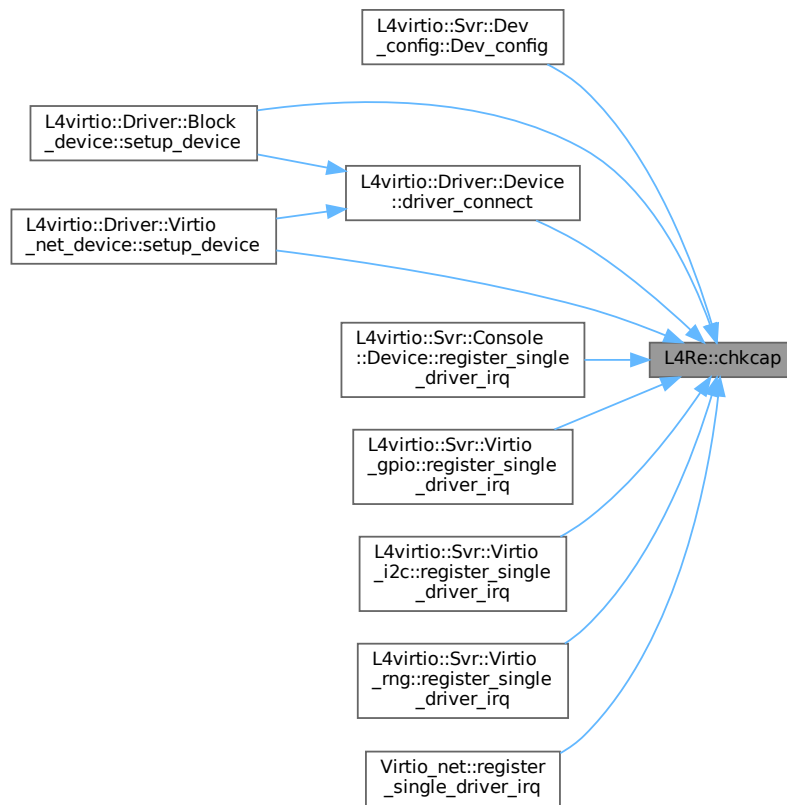
References [L4\\_ENOMEM](#), [L4\\_UNLIKELY](#), and [throw\\_error\(\)](#).

Referenced by [L4virtio::Svr::Dev\\_config::Dev\\_config\(\)](#), [L4virtio::Driver::Device::driver\\_connect\(\)](#), [L4virtio::Svr::Console::Device::register\\_single\\_driver\\_irq\(\)](#), [L4virtio::Svr::Virtio\\_gpio< Request\\_handler, Epiface >::register\\_single\\_driver\\_irq\(\)](#), [L4virtio::Svr::Virtio\\_i2c< Request\\_handler, Epiface >::register\\_single\\_driver\\_irq\(\)](#), [L4virtio::Svr::Virtio\\_rng< Rnd\\_state, Epiface >::register\\_single\\_driver\\_irq\(\)](#), [Virtio\\_net::register\\_single\\_driver\\_irq\(\)](#), [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#), and [L4virtio::Driver::Virtio\\_net\\_device::setup\\_device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.9.3.2 chkipc()

```

l4_msgtag_t L4Re::chkipc (
    l4_msgtag_t tag,
    char const * extra = "",
    l4_utcb_t * utcb = l4_utcb()) [inline]
  
```

Test a message tag for IPC errors.

#### Parameters

<i>tag</i>	Message tag returned by the IPC.
<i>extra</i>	Exception message in case of error.
<i>utcb</i>	The UTCB used in the IPC operation.

#### Returns

On IPC error an exception is thrown, otherwise `tag` is returned.

#### Exceptions



<code>L4::Runtime_exception</code>	with the translated IPC error code
------------------------------------	------------------------------------

This function does not check the message tag's label value.

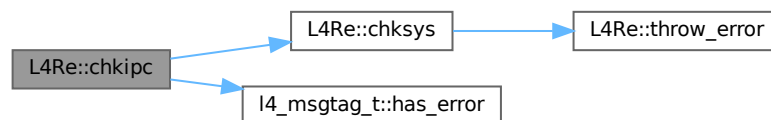
#### Note

This must be called on a message tag before the UTCB is changed.

Definition at line 180 of file [error\\_helper](#).

References [chksys\(\)](#), [l4\\_msgtag\\_t::has\\_error\(\)](#), and [L4\\_UNLIKELY](#).

Here is the call graph for this function:



### 15.9.3.3 chksys() [1/3]

```

long L4Re::chksys (
    l4_msgtag_t const & t,
    char const * extra = "",
    l4_utcb_t * utcb = l4_utcb(),
    long ret = 0) [inline]
  
```

Generate C++ exception on error.

#### Parameters

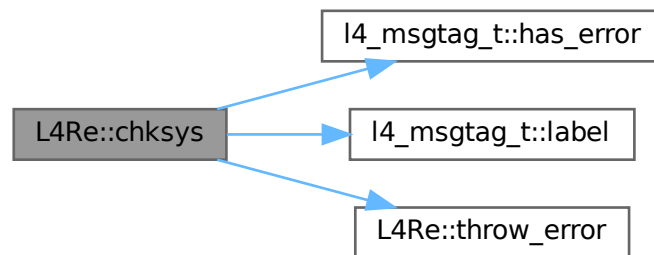
<i>t</i>	Message tag.
<i>extra</i>	Optional text for exception (default "")
<i>utcb</i>	Option UTCB
<i>ret</i>	Optional value for exception, default is error value (err)

This function throws an exception if the message tag contains an error or the label in the message tag is negative. Otherwise the label in the message tag is returned.

Definition at line 93 of file [error\\_helper](#).

References [l4\\_msgtag\\_t::has\\_error\(\)](#), [L4\\_UNLIKELY](#), [l4\\_msgtag\\_t::label\(\)](#), and [throw\\_error\(\)](#).

Here is the call graph for this function:



#### 15.9.3.4 chksys() [2/3]

```

long L4Re::chksys (
    l4_msgtag_t const & t,
    l4_utcb_t * utcb,
    char const * extra = "") [inline]
  
```

Generate C++ exception on error.

##### Parameters

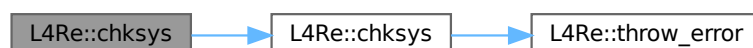
<i>t</i>	Message tag.
<i>utcb</i>	UTCB.
<i>extra</i>	Optional text for exception (default "")

This function throws an exception if the message tag contains an error or the label in the message tag is negative. Otherwise the label in the message tag is returned.

Definition at line 116 of file [error\\_helper](#).

References [chksys\(\)](#).

Here is the call graph for this function:



### 15.9.3.5 chksys() [3/3]

```
long L4Re::chksys (
    long err,
    char const * extra = "",
    long ret = 0) [inline]
```

Generate C++ exception on error.

#### Parameters

<i>err</i>	Error value, if negative exception will be thrown
<i>extra</i>	Optional text for exception (default "")
<i>ret</i>	Optional value for exception, default is error value (err)

This function throws an exception if the err is negative and otherwise returns err.

#### Examples

[examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#).

Definition at line 72 of file [error\\_helper](#).

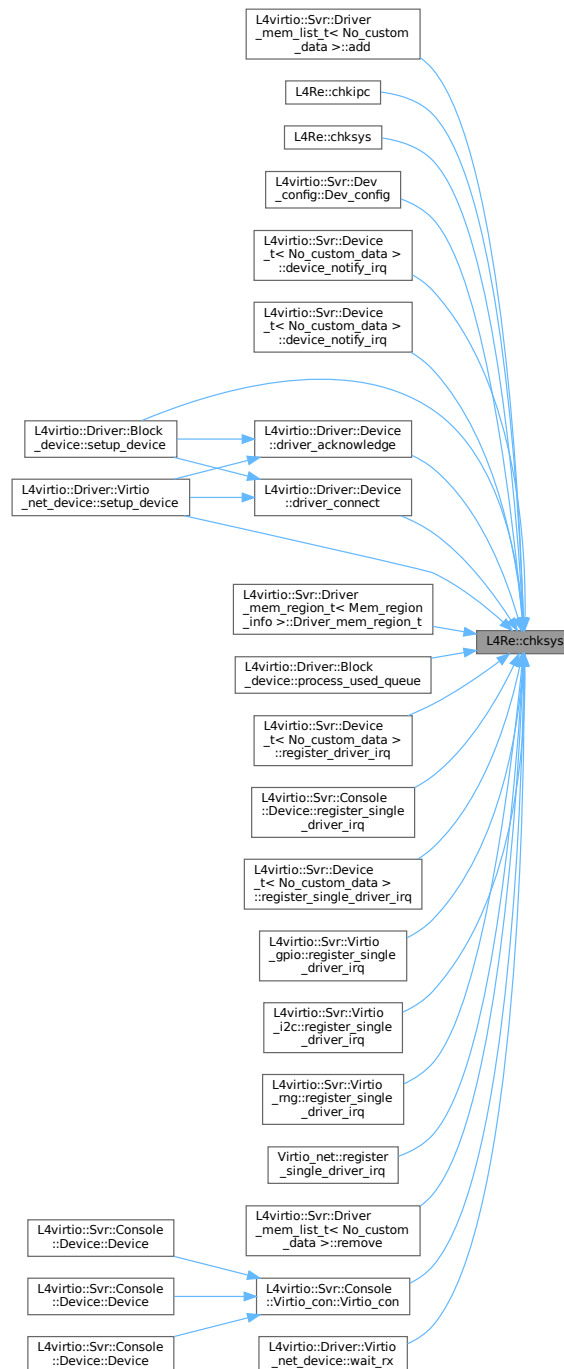
References [L4\\_UNLIKELY](#), and [throw\\_error\(\)](#).

Referenced by [L4virtio::Svr::Driver\\_mem\\_list\\_t< No\\_custom\\_data >::add\(\)](#), [chkipc\(\)](#), [chksys\(\)](#), [L4virtio::Svr::Dev\\_config::Dev\\_config\(\)](#), [L4virtio::Svr::Device\\_t< No\\_custom\\_data >::device\\_notify\\_irq\(\)](#), [L4virtio::Svr::Device\\_t< No\\_custom\\_data >::device\\_notify\\_irq\(\)](#), [L4virtio::Driver::Device::driver\\_acknowledge\(\)](#), [L4virtio::Driver::Device::driver\\_connect\(\)](#), [L4virtio::Svr::Driver\\_mem\\_region\\_t< Mem\\_region >::register\\_driver\\_irq\(\)](#), [L4virtio::Driver::Block\\_device::process\\_used\\_queue\(\)](#), [L4virtio::Svr::Device\\_t< No\\_custom\\_data >::register\\_driver\\_irq\(\)](#), [L4virtio::Svr::Console::Device::register\\_single\\_driver\\_irq\(\)](#), [L4virtio::Svr::Device\\_t< No\\_custom\\_data >::register\\_single\\_driver\\_irq\(\)](#), [L4virtio::Svr::Virtio\\_gpio< Request\\_handler, Epiface >::register\\_single\\_driver\\_irq\(\)](#), [L4virtio::Svr::Virtio\\_i2c< Request\\_handler, Epiface >::register\\_single\\_driver\\_irq\(\)](#), [L4virtio::Svr::Virtio\\_rng< Rnd\\_state, Epiface >::register\\_single\\_driver\\_irq\(\)](#), [Virtio\\_net::register\\_single\\_driver\\_irq\(\)](#), [L4virtio::Svr::Driver\\_mem\\_list\\_t< No\\_custom\\_data >::remove\(\)](#), [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#), [L4virtio::Driver::Virtio\\_net\\_device::setup\\_device\(\)](#), [L4virtio::Svr::Console::Virtio\\_con::Virtio\\_con\(\)](#), and [L4virtio::Driver::Virtio\\_net\\_device::Virtio\\_net\\_device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 15.9.3.6 make\_shared\_cap()

```

template<typename T>
Shared_cap< T > L4Re::make_shared_cap (
    L4Re::Cap_alloc * ca)

```

Allocate a capability slot and wrap it in a [Shared\\_cap](#).

#### Template Parameters

---

<i>T</i>	Type of the object the capability refers to.
----------	----------------------------------------------

### Parameters

<i>ca</i>	Capability allocator to use.
-----------	------------------------------

### Note

This function is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::make\\_shared\\_cap<T>\(\)](#).

Definition at line 49 of file [shared\\_cap](#).

References [L4Re::Cap\\_alloc::alloc\(\)](#).

Here is the call graph for this function:



### 15.9.3.7 make\_shared\_del\_cap()

```

template<typename T>
Shared_del_cap< T > L4Re::make_shared_del_cap (
    L4Re::Cap_alloc * ca)
  
```

Allocate a capability slot and wrap it in a [Shared\\_del\\_cap](#).

### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	----------------------------------------------

### Parameters

<i>ca</i>	Capability allocator to use.
-----------	------------------------------

**Note**

This function is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::make\\_shared\\_del\\_cap<T>\(\)](#).

Definition at line 85 of file [shared\\_cap](#).

References [L4Re::Cap\\_alloc::alloc\(\)](#).

Here is the call graph for this function:

**15.9.3.8 make\_unique\_cap()**

```

template<typename T>
Unique_cap< T > L4Re::make_unique_cap (
    L4Re::Cap_alloc * ca)
  
```

Allocate a capability slot and wrap it in an [Unique\\_cap](#).

**Template Parameters**

<i>T</i>	Type of the object the capability refers to.
----------	----------------------------------------------

**Parameters**

<i>ca</i>	Capability allocator to use.
-----------	------------------------------

**Note**

This function is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::make\\_unique\\_cap<T>\(\)](#).

Definition at line 47 of file [unique\\_cap](#).

References [L4Re::Cap\\_alloc::alloc\(\)](#).

Here is the call graph for this function:



### 15.9.3.9 make\_unique\_del\_cap()

```
template<typename T>
Unique_del_cap< T > L4Re::make_unique_del_cap (
    L4Re::Cap_alloc * ca)
```

Allocate a capability slot and wrap it in an [Unique\\_del\\_cap](#).

#### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	----------------------------------------------

#### Parameters

<i>ca</i>	Capability allocator to use.
-----------	------------------------------

#### Note

This function is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::make\\_unique\\_del\\_cap<T>\(\)](#).

Definition at line 80 of file [unique\\_cap](#).

References [L4Re::Cap\\_alloc::alloc\(\)](#).

Here is the call graph for this function:



### 15.9.3.10 throw\_error()

```
void L4Re::throw_error (
    long err,
    char const * extra = "") [inline]
```

Generate C++ exception.

#### Parameters

<i>err</i>	Error value
------------	-------------



<i>extra</i>	Optional text for exception (default "")
--------------	------------------------------------------

This function throws an [L4](#) exception. The exact exception type depends on the error value (`err`). This function does never return.

#### Examples

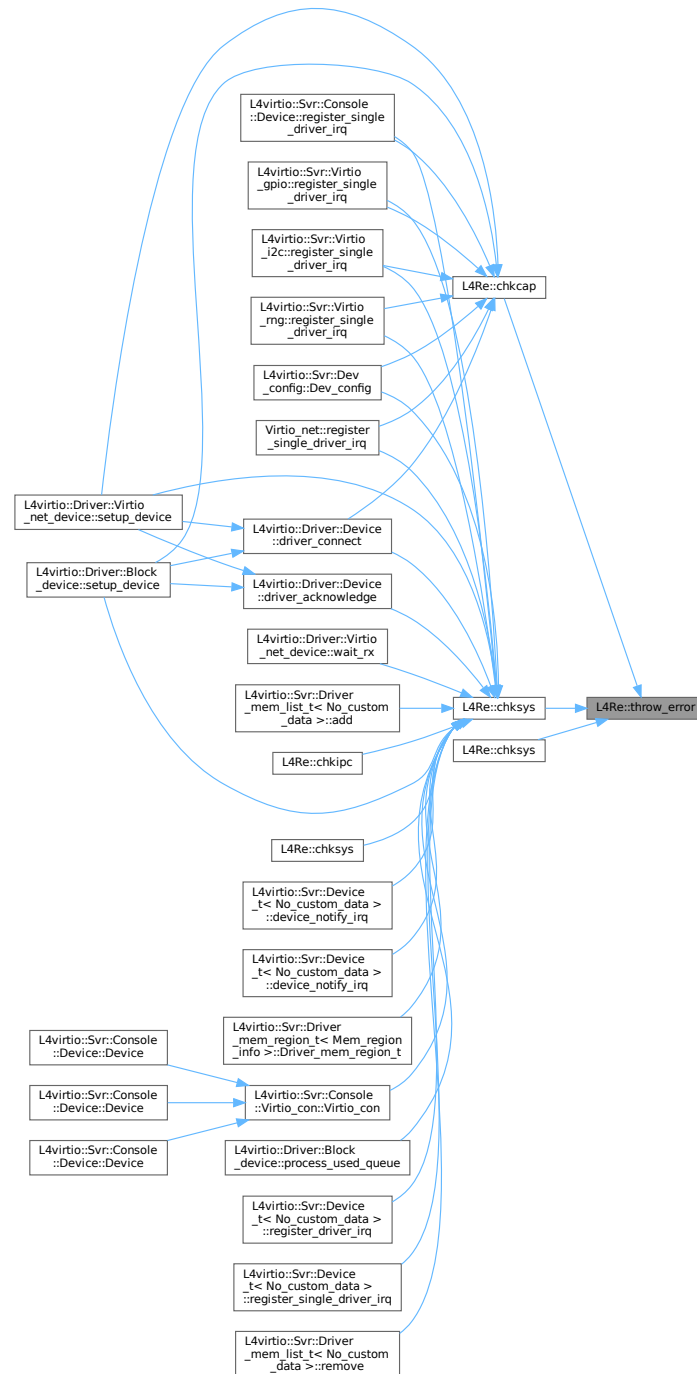
[examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#).

Definition at line [37](#) of file [error\\_helper](#).

References [L4\\_EEXIST](#), [L4\\_ENOENT](#), [L4\\_ENOMEM](#), and [L4\\_ERANGE](#).

Referenced by [chkcapp\(\)](#), [chksys\(\)](#), and [chksys\(\)](#).

Here is the caller graph for this function:



## 15.10 L4Re::Util Namespace Reference

Documentation of the [L4](#) Runtime Environment utility functionality in C++.

## Data Structures

- class [Cap\\_alloc\\_base](#)  
*Capability allocator.*
- class [Br\\_manager](#)  
*Buffer-register (BR) manager for [L4::Server](#).*
- struct [Br\\_manager\\_hooks](#)  
*Predefined server-loop hooks for a server loop using the [Br\\_manager](#).*
- struct [Br\\_manager\\_timeout\\_hooks](#)  
*Predefined server-loop hooks for a server with using the [Br\\_manager](#) and a timeout queue.*
- class [Smart\\_cap\\_auto](#)  
*Helper for [Unique\\_cap](#) and [Unique\\_del\\_cap](#).*
- class [Smart\\_count\\_cap](#)  
*Helper for [Ref\\_cap](#) and [Ref\\_del\\_cap](#).*
- struct [Ref\\_cap](#)  
*Automatic capability that implements automatic free and unmap of the capability selector.*
- struct [Ref\\_del\\_cap](#)  
*Automatic capability that implements automatic free and unmap+delete of the capability selector.*
- class [\\_Cap\\_alloc](#)  
*Adapter to expose the cap allocator implementation as [L4Re::Cap\\_alloc](#) compatible class.*
- struct [Counter](#)  
*Counter for [Counting\\_cap\\_alloc](#) with variable data width.*
- struct [Counter\\_atomic](#)  
*Thread safe version of counter for [Counting\\_cap\\_alloc](#).*
- class [Counting\\_cap\\_alloc](#)  
*Internal reference-counting cap allocator.*
- class [Dataspace\\_svr](#)  
*Dataspace server class.*
- class [Event\\_t](#)  
*Convenience wrapper for getting access to an event object.*
- class [Event\\_buffer\\_t](#)  
*Event\_buffer utility class.*
- class [Event\\_buffer\\_consumer\\_t](#)  
*An event buffer consumer.*
- class [Event\\_svr](#)  
*Convenience wrapper for implementing an event server.*
- class [Item\\_alloc\\_base](#)  
*Item allocator.*
- class [Object\\_registry](#)  
*A registry that manages server objects and their attached IPC gates for a single server loop for a specific thread.*
- class [Registry\\_server](#)  
*A server loop object which has a [Object\\_registry](#) included.*
- class [Vcon\\_svr](#)  
*Console server template class.*
- class [Bitmap\\_base](#)  
*Basic bitmap abstraction.*
- class [Bitmap](#)  
*A static bitmap.*

## Typedefs

- `template<typename T>`  
`using Shared_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES>>`  
*Shared capability that implements automatic free and unmap of the capability selector.*
- `template<typename T>`  
`using shared_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES>>`  
*Shared capability that implements automatic free and unmap of the capability selector.*
- `template<typename T>`  
`using Shared_del_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>>`  
*Shared capability that implements automatic free and unmap+delete of the capability selector.*
- `template<typename T>`  
`using shared_del_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>>`  
*Shared capability that implements automatic free and unmap+delete of the capability selector.*
- `template<typename T>`  
`using Unique_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>>`  
*Unique capability that implements automatic free and unmap of the capability selector.*
- `template<typename T>`  
`using unique_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>>`  
*Unique capability that implements automatic free and unmap of the capability selector.*
- `template<typename T>`  
`using Unique_del_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>>`  
*Unique capability that implements automatic free and unmap+delete of the capability selector.*
- `template<typename T>`  
`using unique_del_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>>`  
*Unique capability that implements automatic free and unmap+delete of the capability selector.*

## Functions

- `template<typename T>`  
`Ref_cap< T >::Cap make_ref_cap ()`  
*Allocate a capability slot and wrap it in a [Ref\\_cap](#).*
- `template<typename T>`  
`Ref_del_cap< T >::Cap make_ref_del_cap ()`  
*Allocate a capability slot and wrap it in a [Ref\\_del\\_cap](#).*
- `int kumem_alloc (l4_addr_t *mem, unsigned pages_order, L4::Cap< L4::Task > task=L4Re::Env::env() ->task(), L4::Cap< L4Re::Rm > rm=L4Re::Env::env() ->rm()) noexcept`  
*Allocate state area.*
- `template<typename T>`  
`Shared_cap< T > make_shared_cap ()`  
*Allocate a capability slot and wrap it in a [Shared\\_cap](#).*
- `template<typename T>`  
`Shared_del_cap< T > make_shared_del_cap ()`  
*Allocate a capability slot and wrap it in a [Shared\\_del\\_cap](#).*
- `template<typename T>`  
`Unique_cap< T > make_unique_cap ()`  
*Allocate a capability slot and wrap it in an [Unique\\_cap](#).*
- `template<typename T>`  
`Unique_del_cap< T > make_unique_del_cap ()`  
*Allocate a capability slot and wrap it in an [Unique\\_del\\_cap](#).*

## Variables

- [\\_Cap\\_alloc](#) & [cap\\_alloc](#)  
*Capability allocator.*

### 15.10.1 Detailed Description

Documentation of the [L4](#) Runtime Environment utility functionality in C++.

### 15.10.2 Typedef Documentation

#### 15.10.2.1 Shared\_cap

```
template<typename T>
using L4Re::Util::Shared\_cap = L4::Detail::Shared_cap_impl<T, Smart\_count\_cap<L4\_FP\_ALL\_SPACES>>
```

Shared capability that implements automatic free and unmap of the capability selector.

#### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	----------------------------------------------

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero.

Usage:

```
L4Re::Util::Shared_cap<L4Re::Dataspace> global_ds_cap;

{
    L4Re::Util::Shared_cap<L4Re::Dataspace>
        ds_cap = make_shared_cap<L4Re::Dataspace>();
    // reference count for the allocated cap selector is now 1

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

    global_ds_cap = ds_cap;
    // reference count is now 2
    ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
```

Definition at line 48 of file [shared\\_cap](#).

#### 15.10.2.2 shared\_cap

```
template<typename T>
using L4Re::Util::shared\_cap = L4::Detail::Shared_cap_impl<T, Smart\_count\_cap<L4\_FP\_ALL\_SPACES>>
```

Shared capability that implements automatic free and unmap of the capability selector.

#### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	----------------------------------------------

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero.

Usage:

```
L4Re::Util::Shared_cap<L4Re::Dataspace> global_ds_cap;

{
    L4Re::Util::Shared_cap<L4Re::Dataspace>
        ds_cap = make_shared_cap<L4Re::Dataspace>();
    // reference count for the allocated cap selector is now 1

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

    global_ds_cap = ds_cap;
    // reference count is now 2
    ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
```

Definition at line 51 of file [shared\\_cap](#).

### 15.10.2.3 Shared\_del\_cap

```
template<typename T>
using L4Re::Util::Shared_del_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>>
```

Shared capability that implements automatic free and unmap+delete of the capability selector.

#### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	----------------------------------------------

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero. The main difference to [Shared\\_cap](#) is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Usage:

```
L4Re::Util::Shared_del_cap<L4Re::Dataspace> global_ds_cap;

{
    L4Re::Util::Shared_del_cap<L4Re::Dataspace>
        ds_cap = make_shared_del_cap<L4Re::Dataspace>();
    // reference count for the allocated cap selector is now 1

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

    global_ds_cap = ds_cap;
    // reference count is now 2
    ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
...
global_ds_cap = L4_INVALID_CAP;
// reference count dropped to 0 (data space shall be deleted).
```

Definition at line 98 of file [shared\\_cap](#).

### 15.10.2.4 shared\_del\_cap

```
template<typename T>
using L4Re::Util::shared_del_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>>
```

Shared capability that implements automatic free and unmap+delete of the capability selector.

#### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	----------------------------------------------

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero. The main difference to [Shared\\_cap](#) is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

#### Usage:

```
L4Re::Util::Shared_del_cap<L4Re::Dataspace> global_ds_cap;

{
    L4Re::Util::Shared_del_cap<L4Re::Dataspace>
    ds_cap = make_shared_del_cap<L4Re::Dataspace>();
    // reference count for the allocated cap selector is now 1

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

    global_ds_cap = ds_cap;
    // reference count is now 2
    ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
...
global_ds_cap = L4_INVALID_CAP;
// reference count dropped to 0 (data space shall be deleted).
```

Definition at line 101 of file [shared\\_cap](#).

### 15.10.2.5 Unique\_cap

```
template<typename T>
using L4Re::Util::Unique_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>>
```

Unique capability that implements automatic free and unmap of the capability selector.

#### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	----------------------------------------------

The ownership of the capability is managed in the same way as `unique_ptr`.

#### Usage:

```

{
    L4Re::Util::Unique_cap<L4Re::Dataspace>
        ds_cap = L4Re::Util::make_unique_cap<L4Re::Dataspace>();

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(L4_PAGESIZE, ds_cap.get()));

    ...

    // At the end of the scope ds_cap is unmapped and the capability
    // selector is freed.
}

```

Definition at line 43 of file [unique\\_cap](#).

### 15.10.2.6 unique\_cap

```

template<typename T>
using L4Re::Util::unique_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>>

```

Unique capability that implements automatic free and unmap of the capability selector.

#### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	----------------------------------------------

The ownership of the capability is managed in the same way as `unique_ptr`.

Usage:

```

{
    L4Re::Util::Unique_cap<L4Re::Dataspace>
        ds_cap = L4Re::Util::make_unique_cap<L4Re::Dataspace>();

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(L4_PAGESIZE, ds_cap.get()));

    ...

    // At the end of the scope ds_cap is unmapped and the capability
    // selector is freed.
}

```

Definition at line 46 of file [unique\\_cap](#).

### 15.10.2.7 Unique\_del\_cap

```

template<typename T>
using L4Re::Util::Unique_del_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>>

```

Unique capability that implements automatic free and unmap+delete of the capability selector.

#### Template Parameters



<i>T</i>	Type of the object the capability refers to.
----------	----------------------------------------------

The main difference to [Unique\\_cap](#) is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Usage:

```
{
    L4Re::Util::Unique_del_cap<L4Re::Dataspace>
        ds_cap = make_unique_del_cap<L4Re::Dataspace>();

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(L4_PAGESIZE, ds_cap.get()));

    ...

    // At the end of the scope ds_cap is unmapped and the capability
    // selector is freed. Because the deletion flag is set the data space
    // shall also be deleted (even if there are other references to this
    // data space).
}
```

Definition at line 86 of file [unique\\_cap](#).

### 15.10.2.8 unique\_del\_cap

```
template<typename T>
using L4Re::Util::unique_del_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>>
```

Unique capability that implements automatic free and unmap+delete of the capability selector.

#### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	----------------------------------------------

The main difference to [Unique\\_cap](#) is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Usage:

```
{
    L4Re::Util::Unique_del_cap<L4Re::Dataspace>
        ds_cap = make_unique_del_cap<L4Re::Dataspace>();

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(L4_PAGESIZE, ds_cap.get()));

    ...

    // At the end of the scope ds_cap is unmapped and the capability
    // selector is freed. Because the deletion flag is set the data space
    // shall also be deleted (even if there are other references to this
    // data space).
}
```

Definition at line 89 of file [unique\\_cap](#).

### 15.10.3 Function Documentation

#### 15.10.3.1 `make_shared_cap()`

```
template<typename T>
Shared_cap< T > L4Re::Util::make_shared_cap ()
```

Allocate a capability slot and wrap it in a [Shared\\_cap](#).

##### Template Parameters

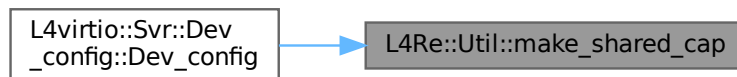
<i>T</i>	Type of the object the capability refers to.
----------	----------------------------------------------

Definition at line 60 of file [shared\\_cap](#).

References [cap\\_alloc](#).

Referenced by [L4virtio::Svr::Dev\\_config::Dev\\_config\(\)](#).

Here is the caller graph for this function:



#### 15.10.3.2 `make_shared_del_cap()`

```
template<typename T>
Shared_del_cap< T > L4Re::Util::make_shared_del_cap ()
```

Allocate a capability slot and wrap it in a [Shared\\_del\\_cap](#).

##### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	----------------------------------------------

Definition at line 110 of file [shared\\_cap](#).

References [cap\\_alloc](#).

#### 15.10.3.3 `make_unique_cap()`

```
template<typename T>
Unique_cap< T > L4Re::Util::make_unique_cap ()
```

Allocate a capability slot and wrap it in an [Unique\\_cap](#).

##### Template Parameters

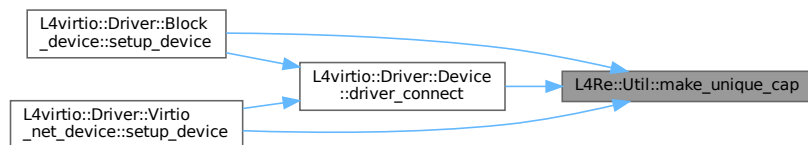
<i>T</i>	Type of the object the capability refers to.
----------	----------------------------------------------

Definition at line 55 of file [unique\\_cap](#).

References [cap\\_alloc](#).

Referenced by [L4virtio::Driver::Device::driver\\_connect\(\)](#), [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#), and [L4virtio::Driver::Virtio\\_net\\_device::setup\\_device\(\)](#).

Here is the caller graph for this function:



### 15.10.3.4 make\_unique\_del\_cap()

```
template<typename T>
Unique_del_cap< T > L4Re::Util::make_unique_del_cap ()
```

Allocate a capability slot and wrap it in an [Unique\\_del\\_cap](#).

#### Template Parameters

<i>T</i>	Type of the object the capability refers to.
----------	----------------------------------------------

Definition at line 98 of file [unique\\_cap](#).

References [cap\\_alloc](#).

## 15.11 L4Re::Vfs Namespace Reference

Virtual file system for interfaces in POSIX libc.

## Data Structures

- class [Be\\_file](#)  
*Boiler plate class for implementing an open file for [L4Re::Vfs](#).*
- class [Be\\_file\\_system](#)  
*Boilerplate class for implementing a [L4Re::Vfs::File\\_system](#).*
- class [Generic\\_file](#)  
*The common interface for an open POSIX file.*
- class [Directory](#)  
*Interface for a POSIX file that is a directory.*
- class [Regular\\_file](#)  
*Interface for a POSIX file that provides regular file semantics.*
- class [Special\\_file](#)  
*Interface for a POSIX file that provides special file semantics.*
- class [File](#)  
*The basic interface for an open POSIX file.*
- class [Mman](#)  
*Interface for POSIX memory management.*
- class [File\\_system](#)  
*Basic interface for an [L4Re::Vfs](#) file system.*
- class [Fs](#)  
*POSIX File-system related functionality.*
- class [Ops](#)  
*Interface for the POSIX backends of an application.*

## Functions

- [L4Re::Vfs::Ops](#) \*vfs\_ops **asm** ("l4re\_env\_posix\_vfs\_ops")  
*Reference to the applications [L4Re::Vfs::Ops](#) singleton.*

### 15.11.1 Detailed Description

Virtual file system for interfaces in POSIX libc.

## 15.12 L4vbus Namespace Reference

C++ interface of the [Vbus](#) API.

## Data Structures

- class [Pm](#)  
*Power-management API mixin.*
- class [Device](#)  
*Device on a [L4vbus::Vbus](#).*
- class [Icu](#)  
*Vbus Interrupt controller API.*
- class [Vbus](#)  
*The virtual bus ([Vbus](#)) interface.*
- class [Gpio\\_pin](#)  
*A GPIO pin.*
- class [Gpio\\_module](#)  
*A [Gpio\\_module](#) groups multiple GPIO pins together.*
- class [Pci\\_host\\_bridge](#)  
*A Pci host bridge.*
- class [Pci\\_dev](#)  
*A PCI device.*

### 15.12.1 Detailed Description

C++ interface of the [Vbus](#) API.

The virtual bus ([Vbus](#)) is a hierarchical (tree) structure of device nodes where each device has a set of resources attached to it. Each virtual bus provides an [Icu](#) ([Interrupt controller](#)) for interrupt handling.

The [Vbus](#) interface allows a client to find and query devices present on his virtual bus. After obtaining a device handle for a specific device the client can enumerate its resources.

Refer to [L4 Vbus functions](#) for the C API.

#### Include File

```
#include <l4/vbus/vbus>
```

#### Include File

```
#include <l4/vbus/vbus_gpio>
```

#### Include File

```
#include <l4/vbus/vbus_pci>
```

## 15.13 L4virtio Namespace Reference

L4-VIRTIO Transport C++ API.

## Data Structures

- class [Device](#)  
*IPC interface for virtio over [L4](#) IPC.*
- class [Ptr](#)  
*Pointer used in virtio descriptors.*
- class [Virtqueue](#)  
*Low-level [Virtqueue](#).*

### 15.13.1 Detailed Description

L4-VIRTIO Transport C++ API.

## Chapter 16

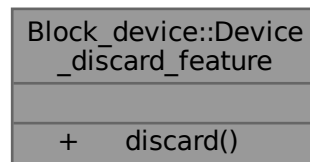
# Data Structure Documentation

### 16.1 Block\_device::Device\_discard\_feature Struct Reference

Partial interface for devices that offer discard functionality.

```
#include <device.h>
```

Collaboration diagram for Block\_device::Device\_discard\_feature:



#### Public Member Functions

- virtual int **discard** ([l4\\_uint64\\_t](#) offset, [Block\\_device::Inout\\_block](#) const &blocks, [Block\\_device::Inout\\_callback](#) const &cb, bool discard)=0  
*Issues one or more WRITE\_ZEROES or DISCARD commands.*

#### 16.1.1 Detailed Description

Partial interface for devices that offer discard functionality.

Definition at line [119](#) of file [device.h](#).

The documentation for this struct was generated from the following file:

- [l4/libblock-device/device.h](#)

## 16.2 Block\_device::Device\_mgr< DEV, FACTORY, SCHEDULER > Class Template Reference

Basic class that scans devices and handles client connections.

```
#include <block_device_mgr.h>
```

Collaboration diagram for Block\_device::Device\_mgr< DEV, FACTORY, SCHEDULER >:

Block_device::Device_mgr< DEV, FACTORY, SCHEDULER >	
+	check_clients()
+	shutdown_event()
+	parse_device_name()

### Public Member Functions

- void **check\_clients** ()  
*Remove clients where the client IPC gate is no longer valid.*
- void **shutdown\_event** (Shutdown\_type type)  
*Process a shutdown event on all connections.*

### Static Public Member Functions

- static int **parse\_device\_name** (std::string const &param, std::string &device)  
*Parse and verify a device string parameter.*

### 16.2.1 Detailed Description

```
template<typename DEV, typename FACTORY = Simple_factory<DEV>, typename SCHEDULER = Rr_↔
scheduler<typename FACTORY::Device_type>>
class Block_device::Device_mgr< DEV, FACTORY, SCHEDULER >
```

Basic class that scans devices and handles client connections.

### Template Parameters

<i>DEV</i>	Base class for all devices.
------------	-----------------------------



<i>FACTORY</i>	Class that creates clients and partitions. See Simple_factory for an example of the required interface.
<i>SCHEDULER</i>	Class that schedules VIRTIO block requests from all clients.

Definition at line 79 of file [block\\_device\\_mgr.h](#).

## 16.2.2 Member Function Documentation

### 16.2.2.1 parse\_device\_name()

```
template<typename DEV, typename FACTORY = Simple_factory<DEV>, typename SCHEDULER = Rr_↵
scheduler<typename FACTORY::Device_type>>
int Block_device::Device_mgr< DEV, FACTORY, SCHEDULER >::parse_device_name (
    std::string const & param,
    std::string & device) [inline], [static]
```

Parse and verify a device string parameter.

#### Parameters

in	<i>param</i>	Device string name parameter.
out	<i>device</i>	Device name extracted from parameter.

#### Returns

[L4](#) error code.

This function tests if 'param' contains one of the following variants of a device name and extracts it into 'device':

- "partlabel:<label>": 'device' contains "<label>" without conversion.
- "partuuid:<uuid>": Check if "<uuid>" is a valid UUID and return with error if not. In case of success, 'device' contains "<uuid>" with all characters converted to upper case.
- "<string>": Check if "<string>" is a valid UUID. If so, 'device' contains "<string>" with all characters converted to upper case. Otherwise, 'device' contains the unmodified "<string>".

Definition at line 383 of file [block\\_device\\_mgr.h](#).

References [L4\\_EINVAL](#), and [L4\\_EOK](#).

The documentation for this class was generated from the following file:

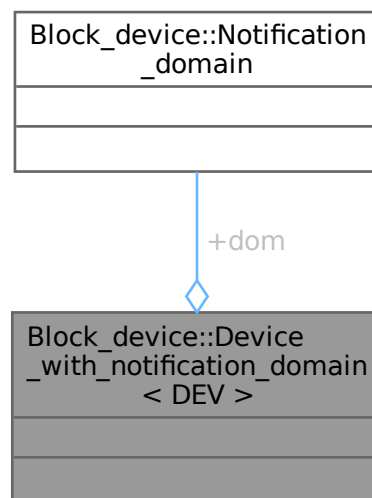
- l4/libblock-device/block\_device\_mgr.h

## 16.3 Block\_device::Device\_with\_notification\_domain< DEV > Struct Template Reference

Device with a per-device notification domain.

```
#include <device.h>
```

Collaboration diagram for Block\_device::Device\_with\_notification\_domain< DEV >:



### 16.3.1 Detailed Description

```
template<typename DEV>
struct Block_device::Device_with_notification_domain< DEV >
```

Device with a per-device notification domain.

Definition at line 109 of file [device.h](#).

The documentation for this struct was generated from the following file:

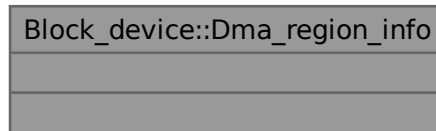
- I4/libblock-device/device.h

## 16.4 Block\_device::Dma\_region\_info Struct Reference

Base class used by the driver implementation to derive its own DMA mapping tracking structure.

```
#include <types.h>
```

Collaboration diagram for Block\_device::Dma\_region\_info:



### 16.4.1 Detailed Description

Base class used by the driver implementation to derive its own DMA mapping tracking structure.

Definition at line 43 of file [types.h](#).

The documentation for this struct was generated from the following file:

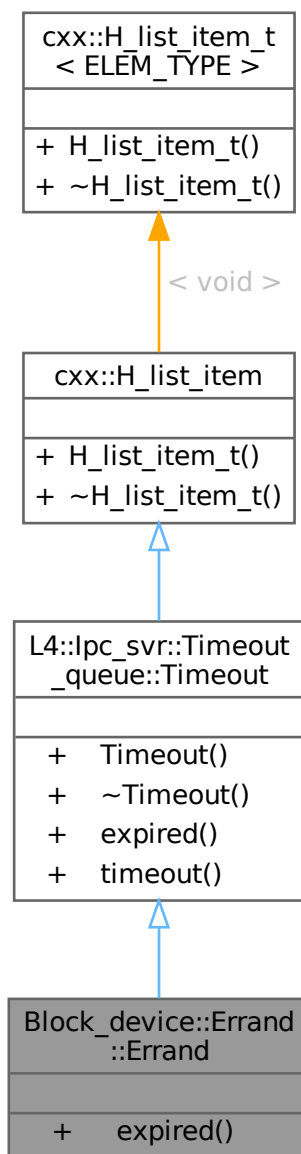
- I4/libblock-device/types.h

## 16.5 Block\_device::Errand::Errand Class Reference

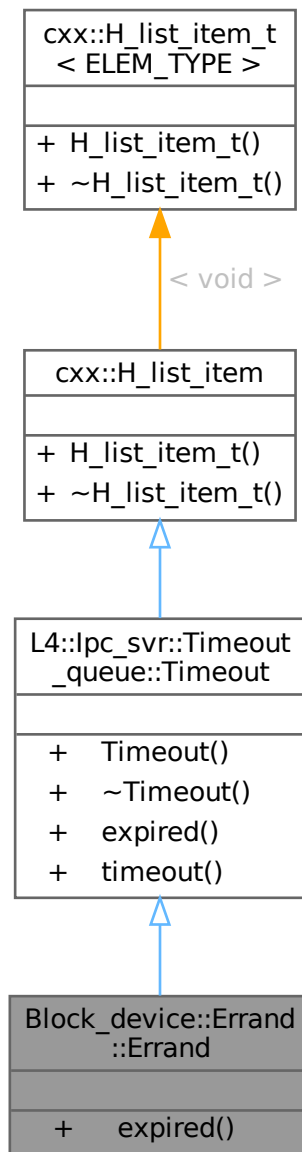
Wrapper for a small task executed asynchronously in the server loop.

```
#include <errand.h>
```

Inheritance diagram for Block\_device::Errand::Errand:



Collaboration diagram for Block\_device::Errand::Errand:



### Public Member Functions

- void `expired()` final  
*callback function to be called when timeout happened*

### Public Member Functions inherited from `L4::lpc_svr::Timeout`

- `Timeout()`  
*Make a timeout.*

- virtual `~Timeout()` = 0  
*Destroy a timeout.*
- `l4_kernel_clock_t timeout()` const  
*return absolute timeout of this callback.*

### Public Member Functions inherited from `cxx::H_list_item_t< void >`

- `H_list_item_t()`  
*Constructor.*
- `~H_list_item_t()` noexcept  
*Destructor.*

## 16.5.1 Detailed Description

Wrapper for a small task executed asynchronously in the server loop.

Errands are implemented as timeout tasks. They might be queued with the current timestamp, so that they are executed as soon as possible on the next iteration of the server loop or they might be scheduled with a timeout, which is particularly useful if the driver has to do a busy wait on the hardware.

Definition at line 98 of file [errand.h](#).

## 16.5.2 Member Function Documentation

### 16.5.2.1 `expired()`

```
void Block_device::Errand::Errand::expired() [inline], [final], [virtual]
```

callback function to be called when timeout happened

#### Note

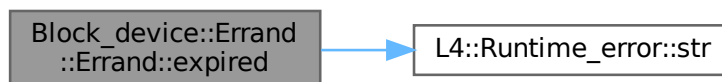
The timeout object is already dequeued when this function is called, this means the timeout may be safely queued again within the `expired()` function.

Implements [L4::lpc\\_svr::Timeout](#).

Definition at line 103 of file [errand.h](#).

References [L4::Runtime\\_error::str\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

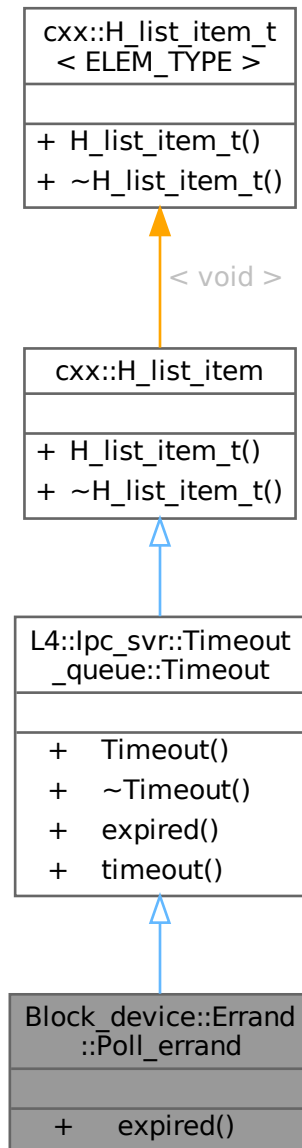
- [l4/libblock-device/errand.h](#)

## 16.6 Block\_device::Errand::Poll\_errand Class Reference

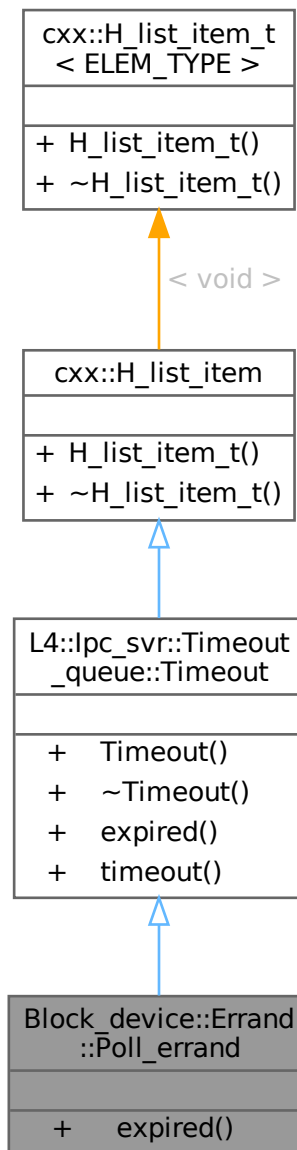
Wrapper for a regularly repeated task.

```
#include <errand.h>
```

Inheritance diagram for Block\_device::Errand::Poll\_errand:



Collaboration diagram for `Block_device::Errand::Poll_errand`:



### Public Member Functions

- void `expired()` final  
*callback function to be called when timeout happened*

### Public Member Functions inherited from `L4::lpc_svr::Timeout`

- `Timeout()`  
*Make a timeout.*



- virtual `~Timeout()` = 0  
*Destroy a timeout.*
- `l4_kernel_clock_t timeout()` const  
*return absolute timeout of this callback.*

### Public Member Functions inherited from `cxx::H_list_item_t< void >`

- `H_list_item_t()`  
*Constructor.*
- `~H_list_item_t()` noexcept  
*Destructor.*

## 16.6.1 Detailed Description

Wrapper for a regularly repeated task.

Definition at line 32 of file [errand.h](#).

## 16.6.2 Member Function Documentation

### 16.6.2.1 `expired()`

```
void Block_device::Errand::Poll_errand::expired () [inline], [final], [virtual]
```

callback function to be called when timeout happened

#### Note

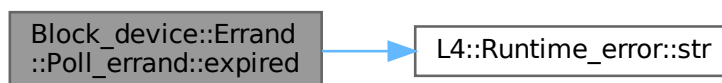
The timeout object is already dequeued when this function is called, this means the timeout may be safely queued again within the `expired()` function.

Implements [L4::lpc\\_svr::Timeout](#).

Definition at line 37 of file [errand.h](#).

References [L4::Runtime\\_error::str\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

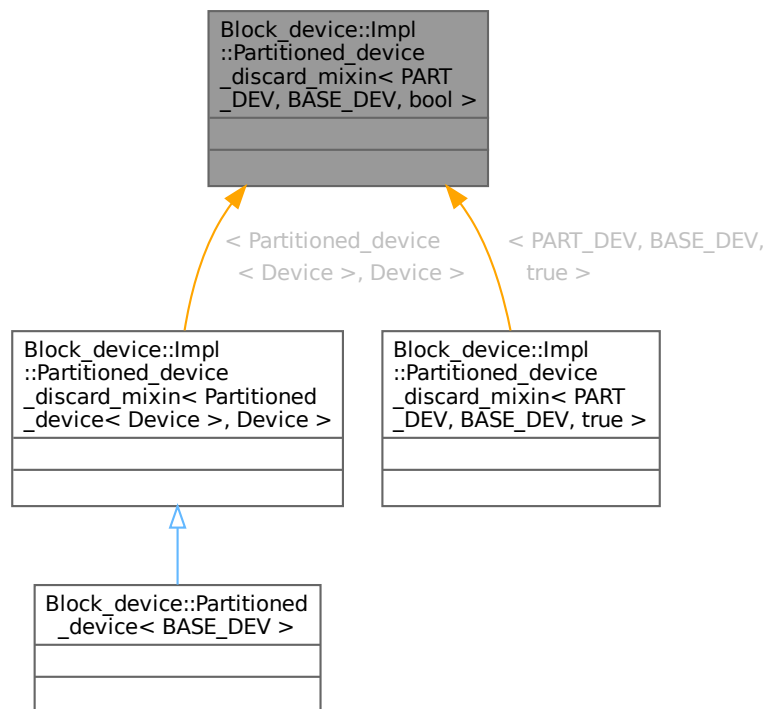
- `l4/libblock-device/errand.h`

## 16.7 Block\_device::Impl::Partitioned\_device\_discard\_mixin< PART\_DEV, BASE\_DEV, bool > Class Template Reference

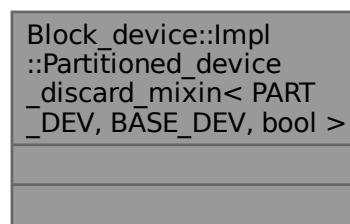
Dummy class used when the device class is not derived from [Device\\_discard\\_feature](#).

```
#include <part_device.h>
```

Inheritance diagram for Block\_device::Impl::Partitioned\_device\_discard\_mixin< PART\_DEV, BASE\_DEV, bool >:



Collaboration diagram for Block\_device::Impl::Partitioned\_device\_discard\_mixin< PART\_DEV, BASE\_DEV, bool >:



### 16.7.1 Detailed Description

```
template<typename PART_DEV, typename BASE_DEV, bool = std::is_base_of<Device_discard_feature,  
BASE_DEV>::value>  
class Block_device::Impl::Partitioned_device_discard_mixin< PART_DEV, BASE_DEV, bool >
```

Dummy class used when the device class is not derived from [Device\\_discard\\_feature](#).

Definition at line 28 of file [part\\_device.h](#).

The documentation for this class was generated from the following file:

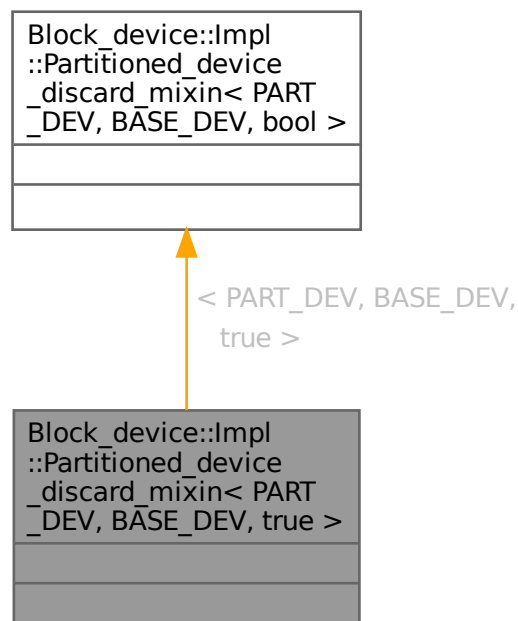
- I4/libblock-device/part\_device.h

## 16.8 Block\_device::Impl::Partitioned\_device\_discard\_mixin< PART\_DEV, BASE\_DEV, true > Class Template Reference

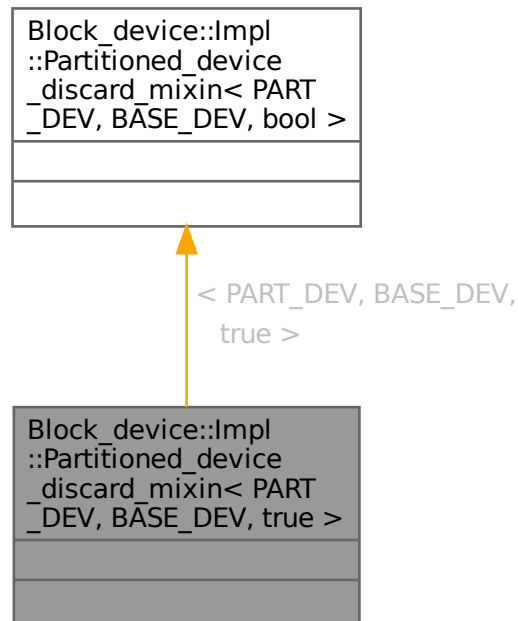
Mixin implementing discard for partition devices.

```
#include <part_device.h>
```

Inheritance diagram for Block\_device::Impl::Partitioned\_device\_discard\_mixin< PART\_DEV, BASE\_DEV, true >:



Collaboration diagram for `Block_device::Impl::Partitioned_device_discard_mixin< PART_DEV, BASE_DEV, true >`:



### 16.8.1 Detailed Description

```

template<typename PART_DEV, typename BASE_DEV>
class Block_device::Impl::Partitioned_device_discard_mixin< PART_DEV, BASE_DEV, true >

```

Mixin implementing discard for partition devices.

#### Template Parameters

<i>PART_DEV</i>	Class of the partition device
<i>BASE_DEV</i>	Class implementing the Device interface.

Definition at line 37 of file [part\\_device.h](#).

The documentation for this class was generated from the following file:

- `l4/libblock-device/part_device.h`

## 16.9 Block\_device::Inout\_block Struct Reference

Description of an inout block to be sent to the device.

```
#include <types.h>
```

Collaboration diagram for Block\_device::Inout\_block:

Block_device::Inout_block	
+	sector
+	flags

### Data Fields

- [l4\\_uint64\\_t](#) **sector** = 0  
*Initial sector. Used only by DISCARD / WRITE\_ZEROES requests.*
- [l4\\_uint32\\_t](#) **flags** = 0  
*Flags from Inout\_flags.*

### 16.9.1 Detailed Description

Description of an inout block to be sent to the device.

Block may be scatter gather in which case they are chained via the next pointer.

Definition at line 66 of file [types.h](#).

The documentation for this struct was generated from the following file:

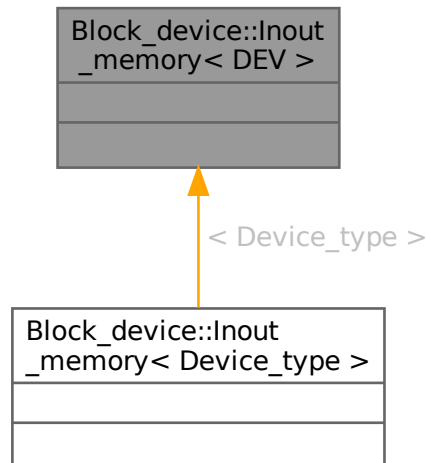
- [l4/libblock-device/types.h](#)

## 16.10 Block\_device::Inout\_memory< DEV > Class Template Reference

Helper class that temporarily allocates memory that can be used for in/out operations with the device.

```
#include <inout_memory.h>
```

Inheritance diagram for Block\_device::Inout\_memory< DEV >:



Collaboration diagram for Block\_device::Inout\_memory< DEV >:



### 16.10.1 Detailed Description

```
template<typename DEV>
class Block_device::Inout_memory< DEV >
```

Helper class that temporarily allocates memory that can be used for in/out operations with the device.

Definition at line 25 of file [inout\\_memory.h](#).

The documentation for this class was generated from the following file:

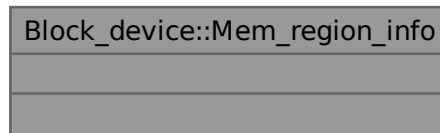
- `I4/libblock-device/inout_memory.h`

## 16.11 Block\_device::Mem\_region\_info Struct Reference

Additional info stored in each [L4virtio::Svr::Driver\\_mem\\_region\\_t](#) used for tracking dataspace-wide DMA mappings.

```
#include <types.h>
```

Collaboration diagram for Block\_device::Mem\_region\_info:



### 16.11.1 Detailed Description

Additional info stored in each [L4virtio::Svr::Driver\\_mem\\_region\\_t](#) used for tracking dataspace-wide DMA mappings.

Definition at line 52 of file [types.h](#).

The documentation for this struct was generated from the following file:

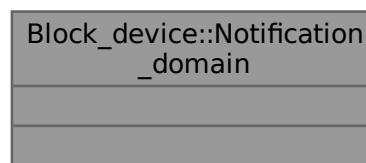
- l4/libblock-device/types.h

## 16.12 Block\_device::Notification\_domain Struct Reference

Opaque type for representing a notification domain.

```
#include <device.h>
```

Collaboration diagram for Block\_device::Notification\_domain:



### 16.12.1 Detailed Description

Opaque type for representing a notification domain.

Notification domains must be assigned to devices such that all devices that require a shared pool of resources to process their requests, also find themselves in the same notification domain. In particular, if two devices access common resources, then they must be in the same domain. An example of this are two partitions sharing the same parent device because processing of requests for one partition might depend on completion of request processing in another partition. On the other hand, independent disk devices will typically not share the same notification domain because their requests are completely independent of each other.

Definition at line 32 of file [device.h](#).

The documentation for this struct was generated from the following file:

- [l4/libblock-device/device.h](#)

## 16.13 Block\_device::Partition\_info Struct Reference

Information about a single partition.

```
#include <partition.h>
```

Collaboration diagram for Block\_device::Partition\_info:

Block_device::Partition_info	
+	guid
+	name
+	first
+	last
+	flags

### Data Fields

- char **guid** [37]  
*ID of the partition.*
- std::u16string **name**  
*UTF16 name of the partition.*
- [l4\\_uint64\\_t](#) **first**  
*First valid sector.*
- [l4\\_uint64\\_t](#) **last**  
*Last valid sector.*
- [l4\\_uint64\\_t](#) **flags**  
*Additional flags, depending on partition type.*



### 16.13.1 Detailed Description

Information about a single partition.

Definition at line 29 of file [partition.h](#).

The documentation for this struct was generated from the following file:

- l4/libblock-device/partition.h

## 16.14 Block\_device::Partition\_reader< DEV > Class Template Reference

Partition table reader for block devices.

```
#include <partition.h>
```

Collaboration diagram for Block\_device::Partition\_reader< DEV >:



### 16.14.1 Detailed Description

```
template<typename DEV>
class Block_device::Partition_reader< DEV >
```

Partition table reader for block devices.

Definition at line 43 of file [partition.h](#).

The documentation for this class was generated from the following file:

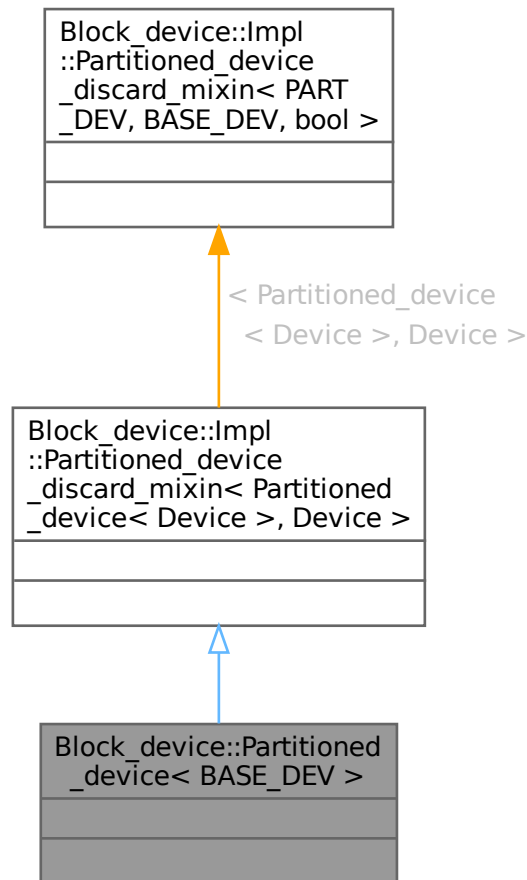
- l4/libblock-device/partition.h

## 16.15 Block\_device::Partitioned\_device< BASE\_DEV > Class Template Reference

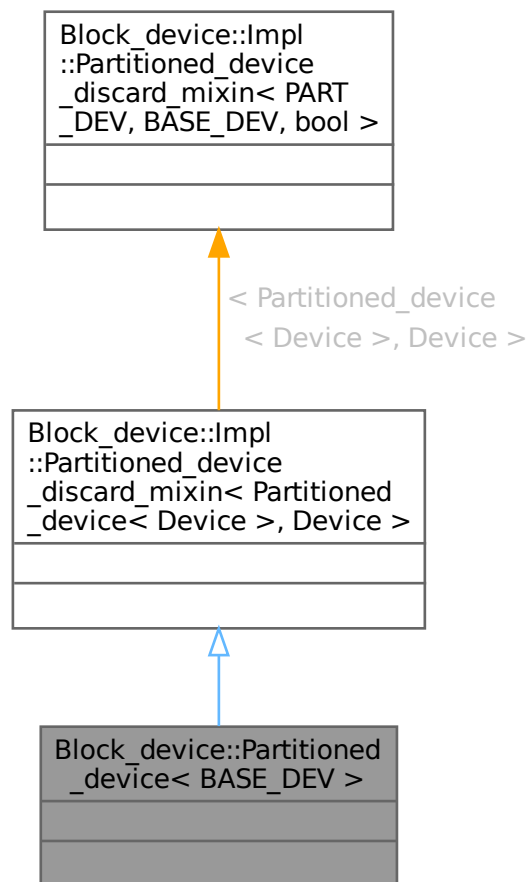
A partition device for the given device interface.

```
#include <part_device.h>
```

Inheritance diagram for Block\_device::Partitioned\_device< BASE\_DEV >:



Collaboration diagram for Block\_device::Partitioned\_device< BASE\_DEV >:



### 16.15.1 Detailed Description

```
template<typename BASE_DEV = Device>
class Block_device::Partitioned_device< BASE_DEV >
```

A partition device for the given device interface.

#### Template Parameters

<i>BASE_DEV</i>	Class defining the device interface. Attention: this is not the class implementing the device itself.
-----------------	-------------------------------------------------------------------------------------------------------

Definition at line 91 of file [part\\_device.h](#).

The documentation for this class was generated from the following file:

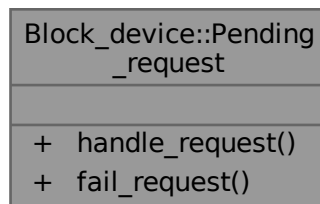
- `l4/libblock-device/part_device.h`

## 16.16 Block\_device::Pending\_request Struct Reference

Interface for pending requests.

```
#include <request.h>
```

Collaboration diagram for Block\_device::Pending\_request:



### Public Member Functions

- virtual int [handle\\_request](#) ()=0  
*Callback used when the request is ready for processing.*
- virtual void [fail\\_request](#) ()=0  
*Callback used when a request is dropped from the queue.*

### 16.16.1 Detailed Description

Interface for pending requests.

Definition at line 14 of file [request.h](#).

### 16.16.2 Member Function Documentation

#### 16.16.2.1 fail\_request()

```
virtual void Block_device::Pending_request::fail_request () [pure virtual]
```

Callback used when a request is dropped from the queue.

The function is called for notification only. The request will be destroyed.

#### 16.16.2.2 handle\_request()

```
virtual int Block_device::Pending_request::handle_request () [pure virtual]
```

Callback used when the request is ready for processing.

### Return values

<i>L4_EOK</i>	Request successfully issued. The callee has taken ownership of the request.
<i>-L4_EBUSY</i>	Device is still busy. The callee must not requeue the request as it will remain in the queue.
<	0 Other fatal error. The caller may dispose of the request.

The documentation for this struct was generated from the following file:

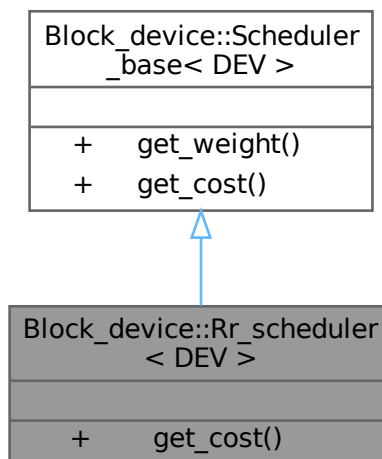
- l4/libblock-device/request.h

## 16.17 Block\_device::Rr\_scheduler< DEV > Struct Template Reference

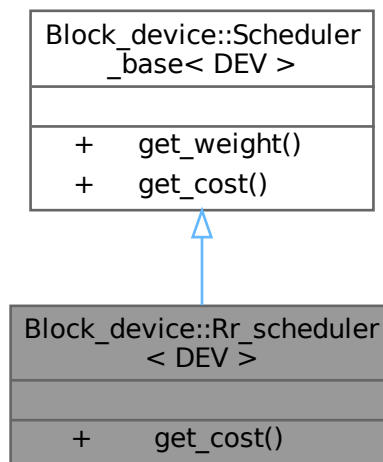
Round Robin scheduler class.

```
#include <scheduler.h>
```

Inheritance diagram for Block\_device::Rr\_scheduler< DEV >:



Collaboration diagram for `Block_device::Rr_scheduler< DEV >`:



### Public Member Functions

- `l4_size_t get_cost (Pending_request const &)` override  
*Return the cost of the pending request.*

### Public Member Functions inherited from `Block_device::Scheduler_base< DEV >`

- virtual `l4_size_t get_weight (Client_type const *)=0`  
*Return the weight of the client.*

## 16.17.1 Detailed Description

```
template<typename DEV>
struct Block_device::Rr_scheduler< DEV >
```

Round Robin scheduler class.

All clients have fixed weight of 1 and all requests have fixed cost of 1, giving thus each client one scheduling chance per scheduling round.

Definition at line 340 of file [scheduler.h](#).

The documentation for this struct was generated from the following file:

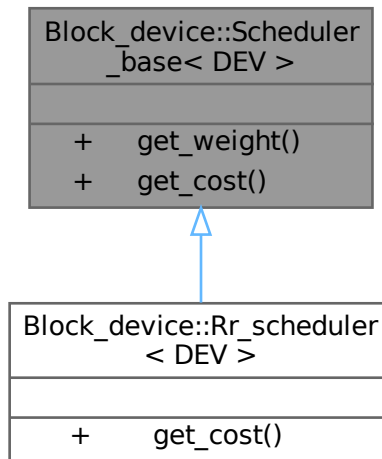
- `l4/libblock-device/scheduler.h`

## 16.18 Block\_device::Scheduler\_base< DEV > Class Template Reference

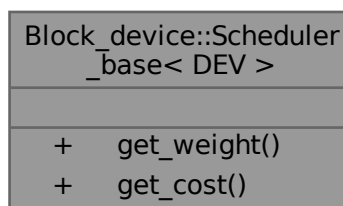
Scheduler base class.

```
#include <scheduler.h>
```

Inheritance diagram for Block\_device::Scheduler\_base< DEV >:



Collaboration diagram for Block\_device::Scheduler\_base< DEV >:



### Public Member Functions

- virtual `l4_size_t get_weight` (`Client_type const *`)=0  
*Return the weight of the client.*
- virtual `l4_size_t get_cost` (`Pending_request const &`)=0  
*Return the cost of the pending request.*

### 16.18.1 Detailed Description

```
template<typename DEV>
class Block_device::Scheduler_base< DEV >
```

Scheduler base class.

Derive from this class and override `get_weight()` and `get_cost()` to implement the desired scheduling algorithm.

The interpretation of the weight function depends on the definition of the cost function. For example, if the cost of each request is fixed to be 1, the weight then says how many requests per scheduling round the client can process. If the weight of each client is also fixed to be 1, it will result in the Round Robin scheduler. If the request cost derives from the size of data the request operates on, the weight determines a data limit.

Definition at line 35 of file [scheduler.h](#).

The documentation for this class was generated from the following file:

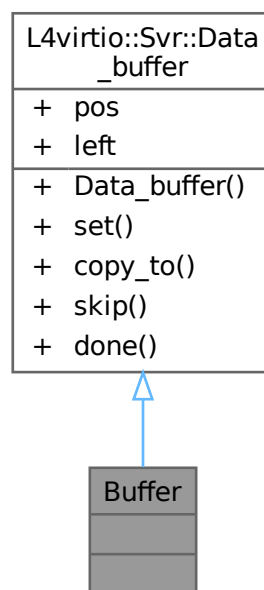
- [l4/libblock-device/scheduler.h](#)

## 16.19 Buffer Struct Reference

Data buffer used to transfer packets.

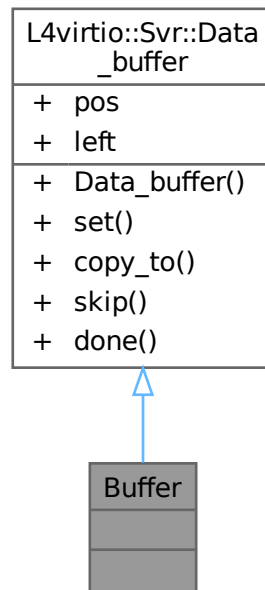
```
#include <virtio_net_buffer.h>
```

Inheritance diagram for Buffer:





Collaboration diagram for Buffer:



### Additional Inherited Members

### Public Member Functions inherited from `L4virtio::Svr::Data_buffer`

- `template<typename T>`  
`Data_buffer` (`T *p`)  
*Create buffer for object p.*
- `template<typename T>`  
`void set` (`T *p`)  
*Set buffer for object p.*
- `l4_uint32_t copy_to` (`Data_buffer *dst`, `l4_uint32_t max=UINT_MAX`)  
*Copy contents from this buffer to the destination buffer.*
- `l4_uint32_t skip` (`l4_uint32_t bytes`)  
*Skip given number of bytes in this buffer.*
- `bool done` () `const`  
*Check if there are no more bytes left in the buffer.*

### Data Fields inherited from `L4virtio::Svr::Data_buffer`

- `char * pos`  
*Current buffer position.*
- `l4_uint32_t left`  
*Bytes left in buffer.*

### 16.19.1 Detailed Description

Data buffer used to transfer packets.

Definition at line 19 of file [virtio\\_net\\_buffer.h](#).

The documentation for this struct was generated from the following file:

- pkg/virtio-net-switch/server/switch/virtio\_net\_buffer.h

## 16.20 `cxx::arith::Ld< V >` Struct Template Reference

Computes the binary logarithm of the given number at compile time.

```
#include <arith>
```

Collaboration diagram for `cxx::arith::Ld< V >`:



### 16.20.1 Detailed Description

```
template<unsigned long V>
struct cxx::arith::Ld< V >
```

Computes the binary logarithm of the given number at compile time.

#### Parameters

<i>val</i>	Number whose logarithm to compute, must be greater than zero.
------------	---------------------------------------------------------------

#### Returns

The binary logarithm of *val*.

Definition at line 48 of file [arith](#).

The documentation for this struct was generated from the following file:

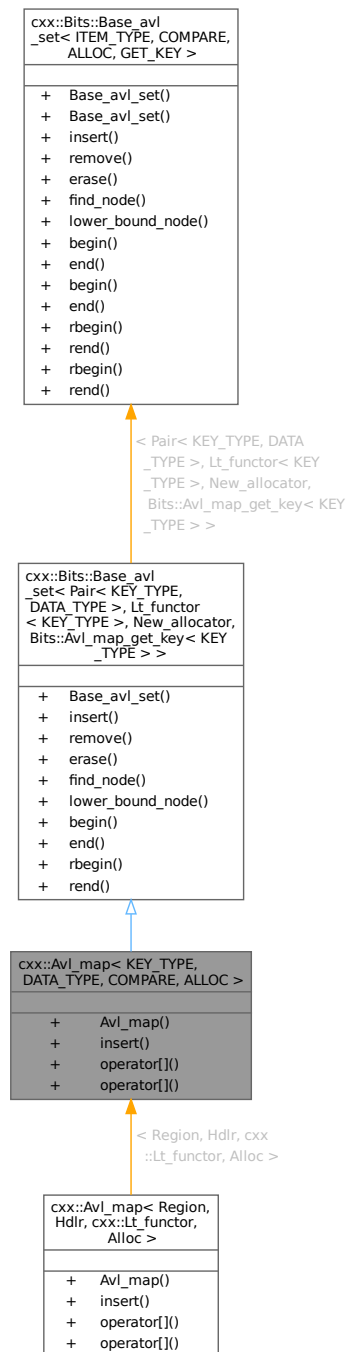
- l4/cxx/arith

## 16.21 `cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >` Class Template Reference

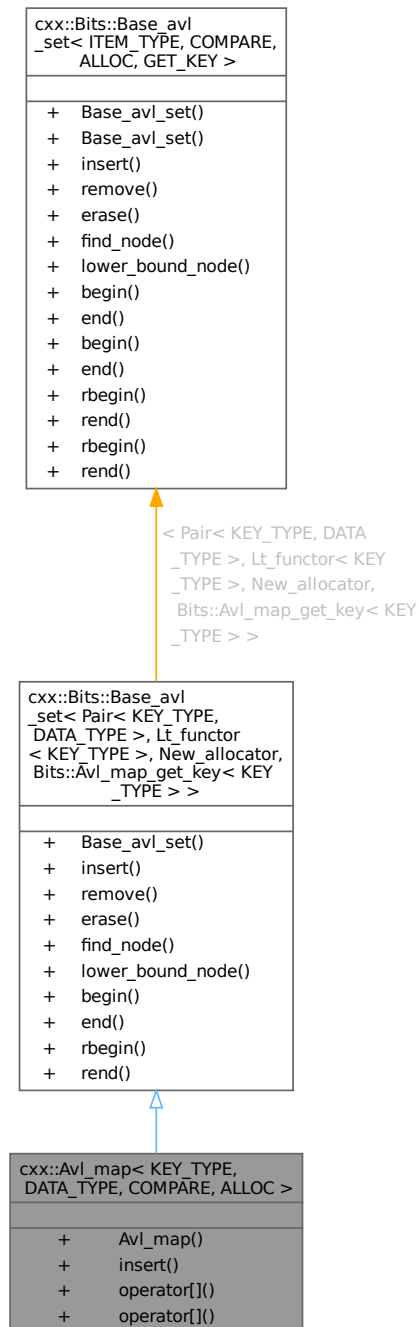
AVL tree based associative container.

```
#include <avl_map>
```

Inheritance diagram for `cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >`:



Collaboration diagram for `cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >`:



## Public Types

- `typedef COMPARE< KEY_TYPE > Key_compare`  
*Type of the comparison functor.*
- `typedef KEY_TYPE Key_type`  
*Type of the key values.*
- `typedef DATA_TYPE Data_type`

*Type of the data values.*

- typedef `Base_type::Node` **Node**

*Return type for find.*

- typedef `Base_type::Node_allocator` **Node\_allocator**

*Type of the allocator.*

## Public Types inherited from

`cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, Lt_functor< KEY_TYPE >, New_allocator,`

- enum  
*Return status constants.*
- typedef `Pair< KEY_TYPE, DATA_TYPE >` **Item\_type**  
*Type for the items store in the set.*
- typedef `Bits::Avl_map_get_key< KEY_TYPE >` **Get\_key**  
*Key-getter type to derive the sort key of an internal node.*
- typedef `Bits::Avl_map_get_key< KEY_TYPE >::Key_type` **Key\_type**  
*Type of the sort key used for the items.*
- typedef `Type_traits< Item_type >::Const_type` **Const\_item\_type**  
*Type used for const items within the set.*
- typedef `Lt_functor< KEY_TYPE >` **Item\_compare**  
*Type for the comparison functor.*
- typedef `New_allocator< _Node >` **Node\_allocator**  
*Type for the node allocator.*
- typedef `Avl_set_iter< _Node, Item_type, Fwd >` **Iterator**  
*Forward iterator for the set.*
- typedef `Avl_set_iter< _Node, Const_item_type, Fwd >` **Const\_iterator**  
*Constant forward iterator for the set.*
- typedef `Avl_set_iter< _Node, Item_type, Rev >` **Rev\_iterator**  
*Backward iterator for the set.*
- typedef `Avl_set_iter< _Node, Const_item_type, Rev >` **Const\_rev\_iterator**  
*Constant backward iterator for the set.*

## Public Member Functions

- `Avl_map (Node_allocator const &alloc=Node_allocator())`  
*Create an empty AVL-tree based map.*
- `cxx::Pair< Iterator, int > insert (Key_type const &key, Data_type const &data)`  
*Insert a <key, data> pair into the map.*
- `Data_type const & operator[] (Key_type const &key) const`  
*Get the data for the given key.*
- `Data_type & operator[] (Key_type const &key)`  
*Get or insert data for the given key.*

## Public Member Functions inherited from

**cxx::Bits::Base\_avl\_set**< **Pair**< **KEY\_TYPE**, **DATA\_TYPE** >, **Lt\_func**< **KEY\_TYPE** >, **New\_allocator**,

- **Base\_avl\_set** (**Node\_allocator** const &alloc=**Node\_allocator**())  
*Create a AVL-tree based set.*
- **cxx::Pair**< **Iterator**, int > **insert** (**Item\_type** const &item)  
*Insert an item into the set.*
- int **remove** (**Key\_type** const &item)  
*Remove an item from the set.*
- int **erase** (**Key\_type** const &item)  
*Erase the item with the given key.*
- **Node** **find\_node** (**Key\_type** const &item) const  
*Lookup a node equal to item.*
- **Node** **lower\_bound\_node** (**Key\_type** const &key) const  
*Find the first node greater or equal to key.*
- **Const\_iterator** **begin** () const  
*Get the constant forward iterator for the first element in the set.*
- **Const\_iterator** **end** () const  
*Get the end marker for the constant forward iterator.*
- **Const\_rev\_iterator** **rbegin** () const  
*Get the constant backward iterator for the last element in the set.*
- **Const\_rev\_iterator** **rend** () const  
*Get the end marker for the constant backward iterator.*

### 16.21.1 Detailed Description

```
template<typename KEY_TYPE, typename DATA_TYPE, template< typename A > class COMPARE = Lt_↔
functor, template< typename B > class ALLOC = New_allocator>
class cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >
```

AVL tree based associative container.

#### Template Parameters

<i>KEY_TYPE</i>	Type of the key values.
<i>DATA_TYPE</i>	Type of the data values.
<i>COMPARE</i>	Type comparison functor for the key values.
<i>ALLOC</i>	Type of the allocator used for the nodes.

Definition at line 45 of file [avl\\_map](#).

### 16.21.2 Constructor & Destructor Documentation

#### 16.21.2.1 Avl\_map()

```
template<typename KEY_TYPE, typename DATA_TYPE, template< typename A > class COMPARE = Lt_↔
functor, template< typename B > class ALLOC = New_allocator>
cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >::Avl_map (
    Node_allocator const & alloc = Node_allocator()) [inline]
```

Create an empty AVL-tree based map.

#### Parameters

<code>alloc</code>	The node allocator.
--------------------	---------------------

Definition at line 80 of file `avl_map`.

## 16.21.3 Member Function Documentation

### 16.21.3.1 `insert()`

```
template<typename KEY_TYPE, typename DATA_TYPE, template< typename A > class COMPARE = Lt_↔
functor, template< typename B > class ALLOC = New_allocator>
cxx::Pair< Iterator, int > cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >::insert (
    Key_type const & key,
    Data_type const & data) [inline]
```

Insert a `<key, data>` pair into the map.

#### Parameters

<code>key</code>	The key value.
<code>data</code>	The data value to insert.

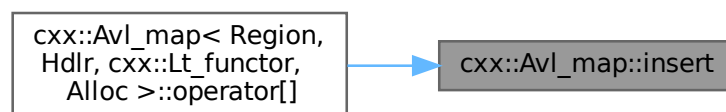
#### Returns

A pair of iterator (`first`) and return value (`second`). `second` will be 0 if the element was inserted into the set and `-#E_exist` if the key was already in the set and the set was therefore not updated. In both cases, `first` contains an iterator that points to the element. `second` may also be `-#E_nomem` when memory for the new node could not be allocated. `first` is then invalid.

Definition at line 99 of file `avl_map`.

Referenced by `cxx::Avl_map< Region, Hdlr, cxx::Lt_functor, Alloc >::operator[]()`.

Here is the caller graph for this function:



### 16.21.3.2 `operator[]()` [1/2]

```
template<typename KEY_TYPE, typename DATA_TYPE, template< typename A > class COMPARE = Lt_↔
functor, template< typename B > class ALLOC = New_allocator>
Data_type & cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >::operator[] (
    Key_type const & key) [inline]
```

Get or insert data for the given key.

#### Parameters

<i>key</i>	The key value to use for lookup.
------------	----------------------------------

**Returns**

If the item already exists, a reference to the data item. Otherwise a new data item is default-constructed and inserted under the given key before a reference is returned.

Definition at line 123 of file [avl\\_map](#).

**16.21.3.3 operator[]() [2/2]**

```
template<typename KEY_TYPE, typename DATA_TYPE, template< typename A > class COMPARE = Lt_↵
functor, template< typename B > class ALLOC = New_allocator>
Data_type const & cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >::operator[] (
    Key_type const & key) const [inline]
```

Get the data for the given key.

**Parameters**

<i>key</i>	The key value to use for lookup.
------------	----------------------------------

**Precondition**

A <key, data> pair for the given key value must exist.

Definition at line 111 of file [avl\\_map](#).

The documentation for this class was generated from the following file:

- I4/cxx/[avl\\_map](#)

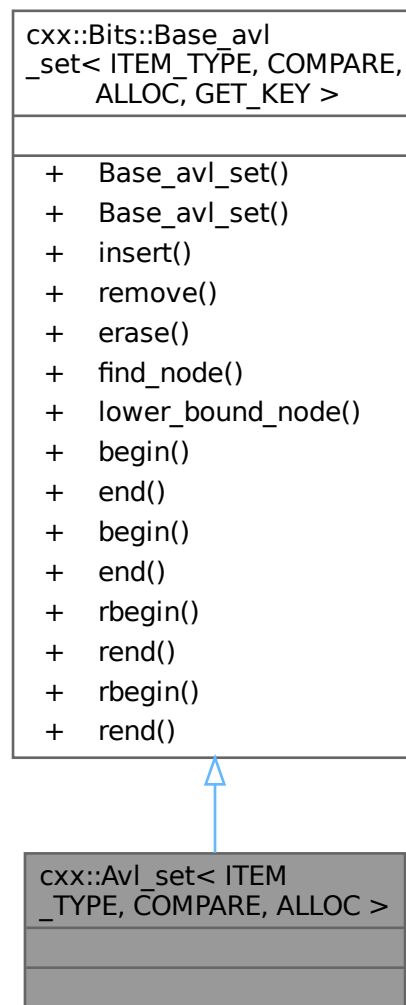
## 16.22 cxx::Avl\_set< ITEM\_TYPE, COMPARE, ALLOC > Class Template Reference

AVL set for simple comparable items.

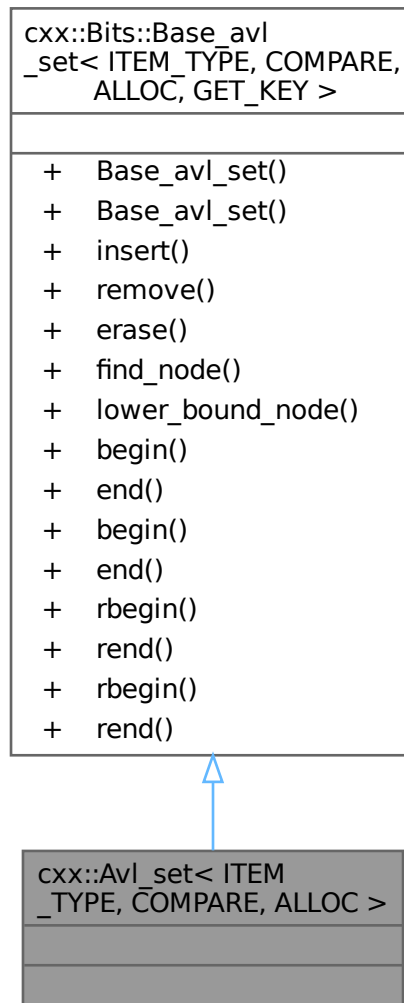
```
#include <avl_set>
```



Inheritance diagram for cxx::Avl\_set< ITEM\_TYPE, COMPARE, ALLOC >:



Collaboration diagram for `cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >`:



#### Additional Inherited Members

#### Public Types inherited from

`cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >`

- enum { `E_noent` = 2 , `E_exist` = 17 , `E_nomem` = 12 , `E_inval` = 22 }
- *Return status constants.*
- typedef `ITEM_TYPE` **Item\_type**
- *Type for the items store in the set.*
- typedef `GET_KEY` **Get\_key**
- *Key-getter type to derive the sort key of an internal node.*
- typedef `GET_KEY::Key_type` **Key\_type**
- *Type of the sort key used for the items.*

- `typedef Type_traits< Item\_type >::Const_type Const_item_type`  
*Type used for const items within the set.*
- `typedef COMPARE Item_compare`  
*Type for the comparison functor.*
- `typedef ALLOC< _Node > Node_allocator`  
*Type for the node allocator.*
- `typedef Avl_set_iter< _Node, Item\_type, Fwd > Iterator`  
*Forward iterator for the set.*
- `typedef Avl_set_iter< _Node, Const\_item\_type, Fwd > Const_iterator`  
*Constant forward iterator for the set.*
- `typedef Avl_set_iter< _Node, Item\_type, Rev > Rev_iterator`  
*Backward iterator for the set.*
- `typedef Avl_set_iter< _Node, Const\_item\_type, Rev > Const_rev_iterator`  
*Constant backward iterator for the set.*

### Public Member Functions inherited from

#### `cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >`

- `Base_avl_set (Node_allocator const &alloc=Node_allocator())`  
*Create a AVL-tree based set.*
- `Base_avl_set (Base_avl_set const &o)`  
*Create a copy of an AVL-tree based set.*
- `cxx::Pair< Iterator, int > insert (Item\_type const &item)`  
*Insert an item into the set.*
- `int remove (Key\_type const &item)`  
*Remove an item from the set.*
- `int erase (Key\_type const &item)`  
*Erase the item with the given key.*
- `Node find_node (Key\_type const &item) const`  
*Lookup a node equal to *item*.*
- `Node lower_bound_node (Key\_type const &key) const`  
*Find the first node greater or equal to *key*.*
- `Const_iterator begin () const`  
*Get the constant forward iterator for the first element in the set.*
- `Const_iterator end () const`  
*Get the end marker for the constant forward iterator.*
- `Iterator begin ()`  
*Get the mutable forward iterator for the first element of the set.*
- `Iterator end ()`  
*Get the end marker for the mutable forward iterator.*
- `Const_rev_iterator rbegin () const`  
*Get the constant backward iterator for the last element in the set.*
- `Const_rev_iterator rend () const`  
*Get the end marker for the constant backward iterator.*
- `Rev_iterator rbegin ()`  
*Get the mutable backward iterator for the last element of the set.*
- `Rev_iterator rend ()`  
*Get the end marker for the mutable backward iterator.*

### 16.22.1 Detailed Description

```
template<typename ITEM_TYPE, class COMPARE = Lt_functor<ITEM_TYPE>, template< typename A >
class ALLOC = New_allocator>
class cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >
```

AVL set for simple comparable items.

The AVL set can store any kind of items where a partial order is defined. The default relation is defined by the '<' operator.

#### Template Parameters

<i>ITEM_TYPE</i>	The type of the items to be stored in the set.
<i>COMPARE</i>	The relation to define the partial order, default is to use operator '<'.
<i>ALLOC</i>	The allocator to use for the nodes of the AVL set.

Definition at line [465](#) of file [avl\\_set](#).

The documentation for this class was generated from the following file:

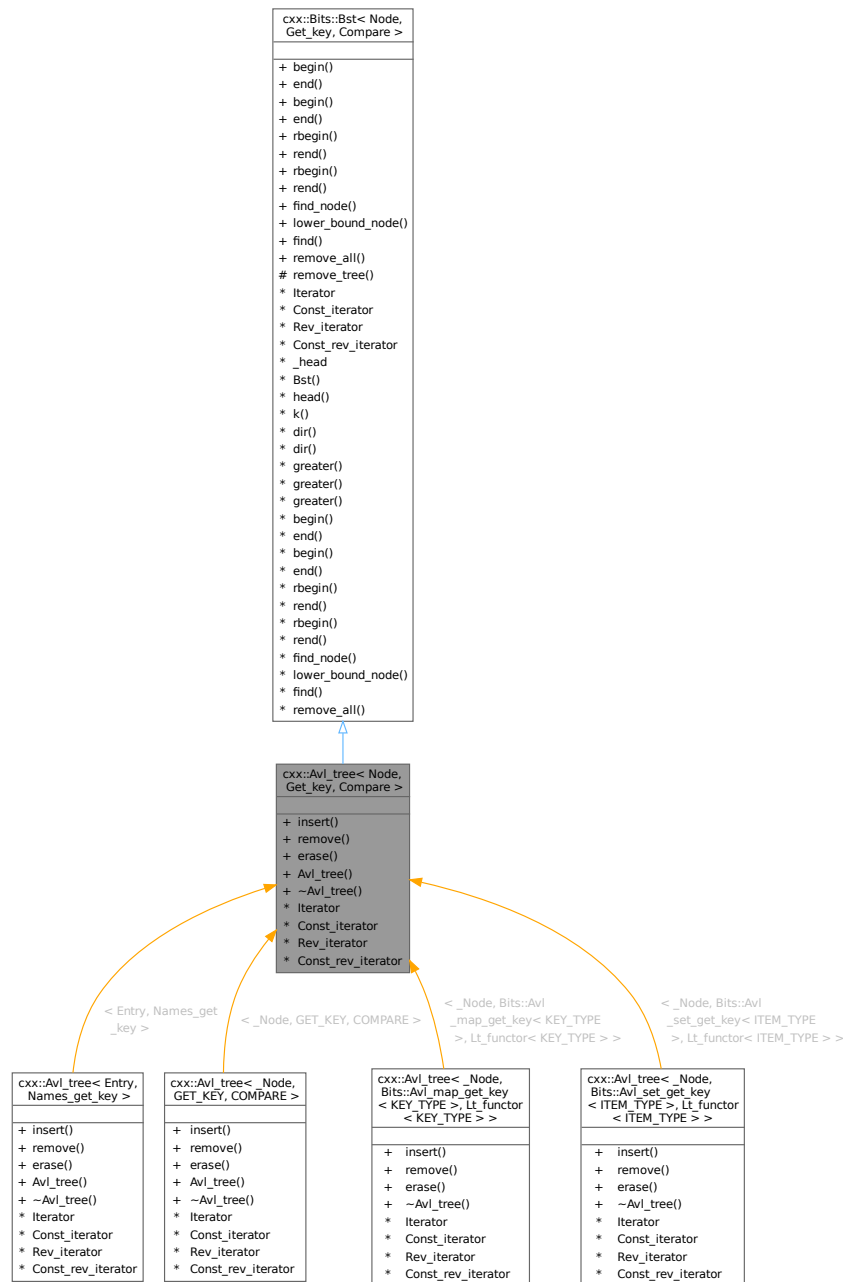
- [l4/cxx/avl\\_set](#)

## 16.23 cxx::Avl\_tree< Node, Get\_key, Compare > Class Template Reference

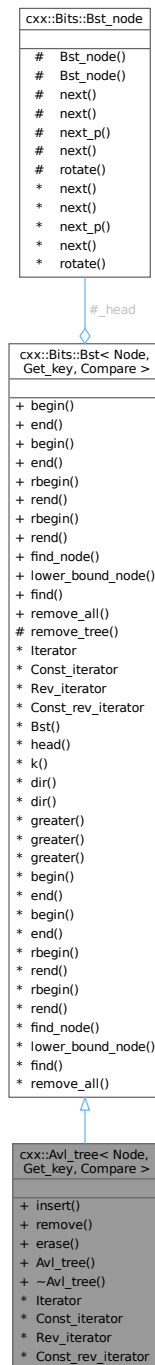
A generic AVL tree.

```
#include <avl_tree>
```

Inheritance diagram for cxx::Avl\_tree< Node, Get\_key, Compare >:



Collaboration diagram for `cxx::Avl_tree< Node, Get_key, Compare >`:



## Public Types

- typedef `Bst::Iterator` `Iterator`

- typedef [Bst::Const\\_iterator](#) **Const\_iterator**  
*Constant forward iterator for the tree.*
- typedef [Bst::Rev\\_iterator](#) **Rev\_iterator**  
*Backward iterator for the tree.*
- typedef [Bst::Const\\_rev\\_iterator](#) **Const\_rev\_iterator**  
*Constant backward iterator for the tree.*

## Public Types inherited from [cxx::Bits::Bst< Node, Get\\_key, Compare >](#)

- typedef [Get\\_key::Key\\_type](#) **Key\_type**  
*The type of key values used to generate the total order of the elements.*
- typedef [Type\\_traits< \[Key\\\_type\]\(#\) >::Param\\_type](#) **Key\_param\_type**  
*The type for key parameters.*
- typedef [Fwd](#) **Fwd\_iter\_ops**  
*Helper for building forward iterators for different wrapper classes.*
- typedef [Rev](#) **Rev\_iter\_ops**  
*Helper for building reverse iterators for different wrapper classes.*
- typedef [\\_\\_Bst\\_iter< Node, Node, Fwd >](#) **Iterator**  
*Forward iterator.*
- typedef [\\_\\_Bst\\_iter< Node, Node const, Fwd >](#) **Const\_iterator**  
*Constant forward iterator.*
- typedef [\\_\\_Bst\\_iter< Node, Node, Rev >](#) **Rev\_iterator**  
*Backward iterator.*
- typedef [\\_\\_Bst\\_iter< Node, Node const, Rev >](#) **Const\_rev\_iterator**  
*Constant backward.*

## Public Member Functions

- [Pair< Node \\*, bool >](#) [insert](#) (Node \*new\_node)  
*Insert a new node into this AVL tree.*
- Node \* [remove](#) (Key\_param\_type key)  
*Remove the node with key from the tree.*
- Node \* [erase](#) (Key\_param\_type key)  
*An alias for [remove\(\)](#).*
- [Avl\\_tree](#) ()=default  
*Create an empty AVL tree.*
- [~Avl\\_tree](#) () noexcept  
*Destroy the tree.*

## Public Member Functions inherited from `cxx::Bits::Bst< Node, Get_key, Compare >`

- `Const_iterator begin ()` const  
*Get the constant forward iterator for the first element in the set.*
- `Const_iterator end ()` const  
*Get the end marker for the constant forward iterator.*
- `Iterator begin ()`  
*Get the mutable forward iterator for the first element of the set.*
- `Iterator end ()`  
*Get the end marker for the mutable forward iterator.*
- `Const_rev_iterator rbegin ()` const  
*Get the constant backward iterator for the last element in the set.*
- `Const_rev_iterator rend ()` const  
*Get the end marker for the constant backward iterator.*
- `Rev_iterator rbegin ()`  
*Get the mutable backward iterator for the last element of the set.*
- `Rev_iterator rend ()`  
*Get the end marker for the mutable backward iterator.*
  
- `Node * find_node (Key_param_type key)` const  
*find the node with the given key.*
- `Node * lower_bound_node (Key_param_type key)` const  
*Find the first node with a key not less than the given `key`.*
- `Const_iterator find (Key_param_type key)` const  
*find the node with the given key.*
- `template<typename FUNC>`  
`void remove_all (FUNC &&callback)`  
*Clear the tree.*

## Additional Inherited Members

- `Bst ()`  
*Create an empty tree.*
  
- `Node * head ()` const  
*Access the head node as object of type `Node`.*

## Static Protected Member Functions inherited from `cxx::Bits::Bst< Node, Get_key, Compare >`

- `template<typename FUNC>`  
`static void remove_tree (Bst_node *head, FUNC &&callback)`  
*Remove all elements in the subtree of head.*
  
- `static Key_type k (Bst_node const *n)`  
*Get the key value of `n`.*
  
- `static Dir dir (Key_param_type l, Key_param_type r)`



*Get the direction to go from  $l$  to search for  $r$ .*

- static `Dir dir (Key_param_type l, Bst_node const *r)`

*Get the direction to go from  $l$  to search for  $r$ .*

- static bool **greater** (Key\_param\_type l, Key\_param\_type r)

*Is  $l$  greater than  $r$ .*

- static bool **greater** (Key\_param\_type l, Bst\_node const \*r)

*Is  $l$  greater than  $r$ .*

- static bool **greater** (Bst\_node const \*l, Bst\_node const \*r)

*Is  $l$  greater than  $r$ .*

- `Bst_node * _head`

*The head pointer of the tree.*

### 16.23.1 Detailed Description

```
template<typename Node, typename Get_key, typename Compare = Lt_functor<typename Get_key::Key↵
_type>>
class cxx::Avl_tree< Node, Get_key, Compare >
```

A generic AVL tree.

#### Template Parameters

<i>Node</i>	The data type of the nodes (must inherit from <a href="#">Avl_tree_node</a> ).
<i>Get_key</i>	The meta function to get the key value from a node. The implementation uses <code>Get_key::key↵_of(ptr_to_node)</code> . The type of the key values must be defined in <code>Get_key::Key_type</code> .
<i>Compare</i>	Binary relation to establish a total order for the nodes of the tree. <code>Compare() (l, r)</code> must return true if the key $l$ is smaller than the key $r$ .

This implementation does not provide any memory management. It is the responsibility of the caller to allocate nodes before inserting them and to free them when they are removed or when the tree is destroyed. Conversely, the caller must also ensure that nodes are removed from the tree before they are destroyed.

#### Examples

[tmpfs/lib/src/fs.cc](#).

Definition at line 100 of file [avl\\_tree](#).

## 16.23.2 Member Typedef Documentation

### 16.23.2.1 Iterator

```
template<typename Node, typename Get_key, typename Compare = It_functor<typename Get_key::←
Key_type>>
typedef Bst::Iterator cxx::Avl_tree< Node, Get_key, Compare >::Iterator
```

Forward iterator for the tree.

Definition at line 130 of file [avl\\_tree](#).

## 16.23.3 Member Function Documentation

### 16.23.3.1 insert()

```
template<typename Node, typename Get_key, class Compare>
Pair< Node *, bool > cxx::Avl_tree< Node, Get_key, Compare >::insert (
    Node * new_node)
```

Insert a new node into this AVL tree.

#### Parameters

<i>new_node</i>	A pointer to the new node.
-----------------	----------------------------

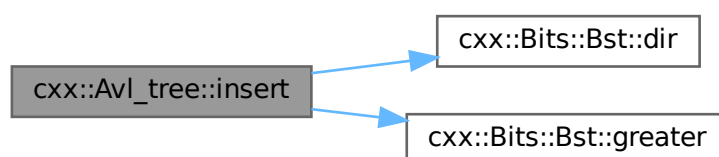
#### Returns

A pair, with *second* set to `true` and *first* pointing to *new\_node*, on success. If there is already a node with the same key then *first* points to this node and *second* is 'false'.

Definition at line 220 of file [avl\\_tree](#).

References [cxx::Bits::Bst< Node, Get\\_key, Compare >::dir\(\)](#), [cxx::Bits::Bst< Node, Get\\_key, Compare >::greater\(\)](#), and [cxx::Bits::Direction::N](#).

Here is the call graph for this function:



### 16.23.3.2 remove()

```
template<typename Node, typename Get_key, class Compare>
Node * cxx::Avl_tree< Node, Get_key, Compare >::remove (
    Key_param_type key) [inline]
```

Remove the node with *key* from the tree.

#### Parameters

---

<i>key</i>	The key to the node to remove.
------------	--------------------------------

#### Returns

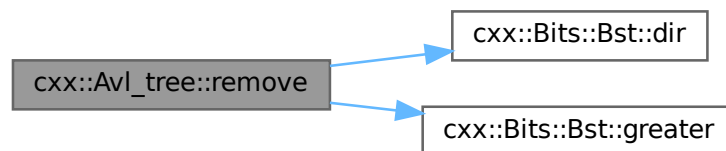
The pointer to the removed node on success, or 0 if no node with the *key* exists.

Definition at line 282 of file [avl\\_tree](#).

References [cxx::Bits::Bst< Node, Get\\_key, Compare >::dir\(\)](#), [cxx::Bits::Bst< Node, Get\\_key, Compare >::greater\(\)](#), [cxx::Bits::Direction::L](#), and [cxx::Bits::Direction::N](#).

Referenced by [cxx::Avl\\_tree< Entry, Names\\_get\\_key >::erase\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

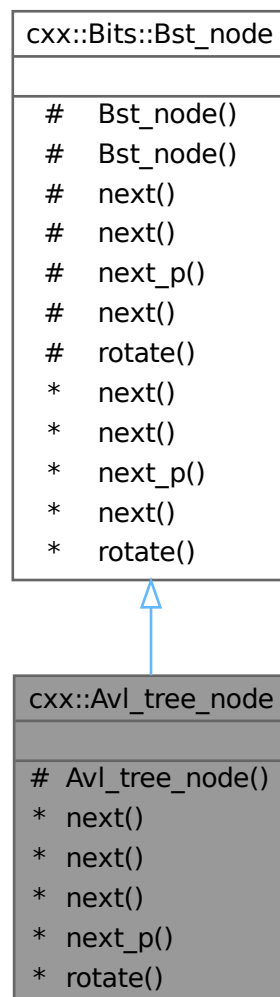
- [I4/cxx/avl\\_tree](#)

## 16.24 cxx::Avl\_tree\_node Class Reference

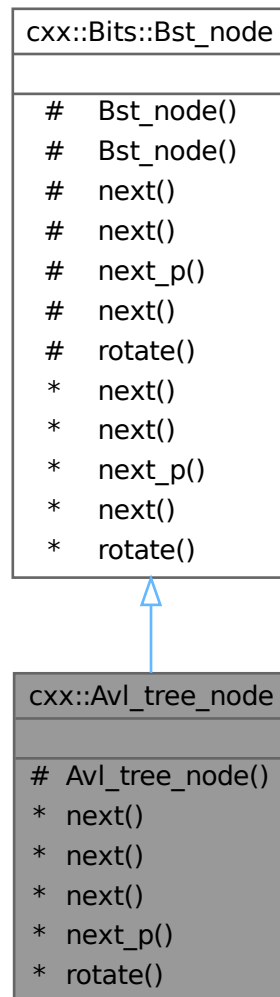
Node of an AVL tree.

```
#include <avl_tree>
```

Inheritance diagram for cxx::Avl\_tree\_node:



Collaboration diagram for `cxx::Avl_tree_node`:



### Protected Member Functions

- **`Avl_tree_node()`**=default

*Create an uninitialized node, this is what you should do.*

### Protected Member Functions inherited from `cxx::Bits::Bst_node`

- **`Bst_node()`**

*Create uninitialized node.*

- **`Bst_node(bool)`**

*Create initialized node.*

### Additional Inherited Members

### Static Protected Member Functions inherited from `cxx::Bits::Bst_node`

- static `Bst_node * next` (`Bst_node` const \*p, `Direction` d)  
*Get next node in direction d.*
- static void `next` (`Bst_node` \*p, `Direction` d, `Bst_node` \*n)  
*Set next node of p in direction d to n.*
- static `Bst_node ** next_p` (`Bst_node` \*p, `Direction` d)  
*Get pointer to link in direction d.*
- template<typename Node>  
static Node \* `next` (`Bst_node` const \*p, `Direction` d)  
*Get next node in direction d as type Node.*
- static void `rotate` (`Bst_node` \*\*t, `Direction` idir)  
*Rotate subtree t in the opposite direction of idir.*

### 16.24.1 Detailed Description

Node of an AVL tree.

#### Examples

[tmpfs/lib/src/fs.cc](#).

Definition at line 29 of file [avl\\_tree](#).

The documentation for this class was generated from the following file:

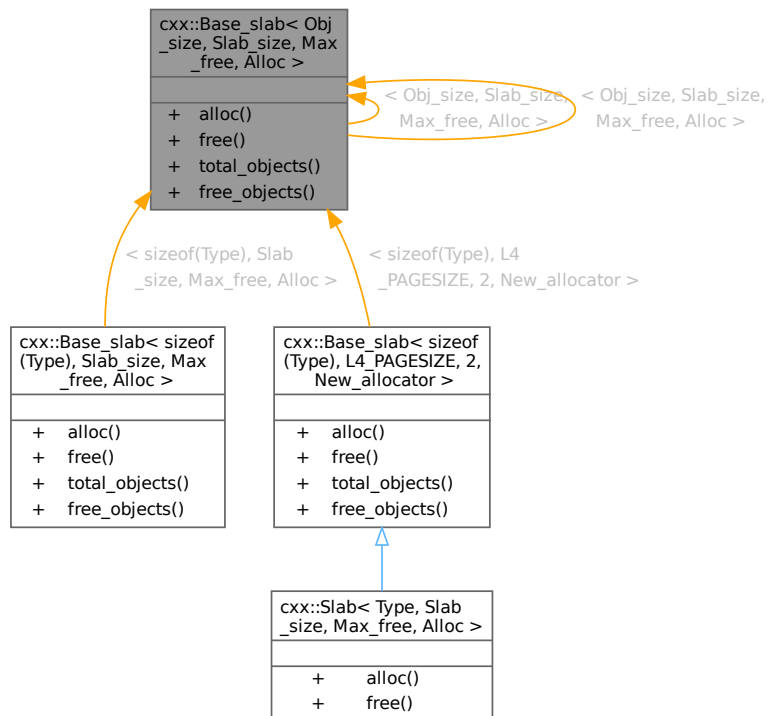
- [l4/cxx/avl\\_tree](#)

## 16.25 `cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >` Class Template Reference

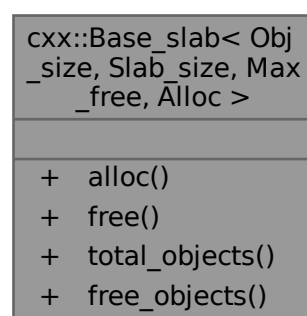
Basic slab allocator.

```
#include <slab_alloc>
```

Inheritance diagram for `cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >`:



Collaboration diagram for `cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >`:



## Data Structures

- struct [Slab\\_i](#)  
*Type of a slab.*



## Public Types

- enum { `object_size` = `Obj_size` , `slab_size` = `Slab_size` , `objects_per_slab` = (`Slab_size` - `sizeof(Slab_head)`) / `object_size` , `max_free_slabs` = `Max_free` }
- typedef `Alloc< Slab_i >` **`Slab_alloc`**  
*Type of the backend allocator.*

## Public Member Functions

- void \* `alloc` () noexcept  
*Allocate a new object.*
- void `free` (void \* `_o`) noexcept  
*Free the given object (`_o`).*
- unsigned `total_objects` () const noexcept  
*Get the total number of objects managed by the slab allocator.*
- unsigned `free_objects` () const noexcept  
*Get the number of objects which can be allocated before a new empty slab needs to be added to the slab allocator.*

### 16.25.1 Detailed Description

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class Alloc
= New_allocator>
class cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >
```

Basic slab allocator.

#### Template Parameters

<i>Obj_size</i>	The size of the objects managed by the allocator (in bytes).
<i>Slab_size</i>	The size of a slab (in bytes).
<i>Max_free</i>	The maximum number of free slabs. When this limit is reached slabs are freed, provided that the backend allocator supports allocated memory to be freed.
<i>Alloc</i>	The backend allocator used to allocate slabs.

Definition at line 31 of file [slab\\_alloc](#).

### 16.25.2 Member Enumeration Documentation

#### 16.25.2.1 anonymous enum

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
anonymous enum
```

#### Enumerator

object_size	Size of an object.
slab_size	Size of a slab.
objects_per_slab	Objects per slab.
max_free_slabs	Maximum number of free slabs.

Definition at line 65 of file [slab\\_alloc](#).

## 16.25.3 Member Function Documentation

### 16.25.3.1 alloc()

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
void * cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::alloc () [inline], [noexcept]
```

Allocate a new object.

#### Returns

A pointer to the new object if the allocation succeeds, or 0 on failure to acquire memory from the backend allocator when the slab cache memory is already exhausted.

#### Note

The user is responsible for initializing the object.

Definition at line 207 of file [slab\\_alloc](#).

### 16.25.3.2 free()

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
void cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::free (
    void * _o) [inline], [noexcept]
```

Free the given object (\_o).

#### Precondition

The object must have been allocated with this allocator.

Definition at line 246 of file [slab\\_alloc](#).

### 16.25.3.3 `free_objects()`

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
unsigned cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::free_objects () const [inline],
[noexcept]
```

Get the number of objects which can be allocated before a new empty slab needs to be added to the slab allocator.

#### Returns

The number of free objects in the slab allocator.

Definition at line 308 of file [slab\\_alloc](#).

### 16.25.3.4 `total_objects()`

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
unsigned cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::total_objects () const [inline],
[noexcept]
```

Get the total number of objects managed by the slab allocator.

#### Returns

The number of objects managed by the allocator (including the free objects).

Definition at line 299 of file [slab\\_alloc](#).

The documentation for this class was generated from the following file:

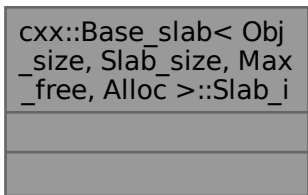
- `l4/cxx/slab_alloc`

## 16.26 `cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_i` Struct Reference

Type of a slab.

```
#include <slab_alloc>
```

Collaboration diagram for `cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_i`:



```
classDiagram
    class cxx__Base_slab__Obj_size_Slab_size_Max_free_Alloc__Slab_i {
    }
    cxx__Base_slab__Obj_size_Slab_size_Max_free_Alloc__Slab_i -- cxx__Base_slab__Obj_size_Slab_size_Max_free_Alloc__Slab_i
```

### 16.26.1 Detailed Description

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class Alloc
= New_allocator>
struct cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_i
```

Type of a slab.

Definition at line 86 of file [slab\\_alloc](#).

The documentation for this struct was generated from the following file:

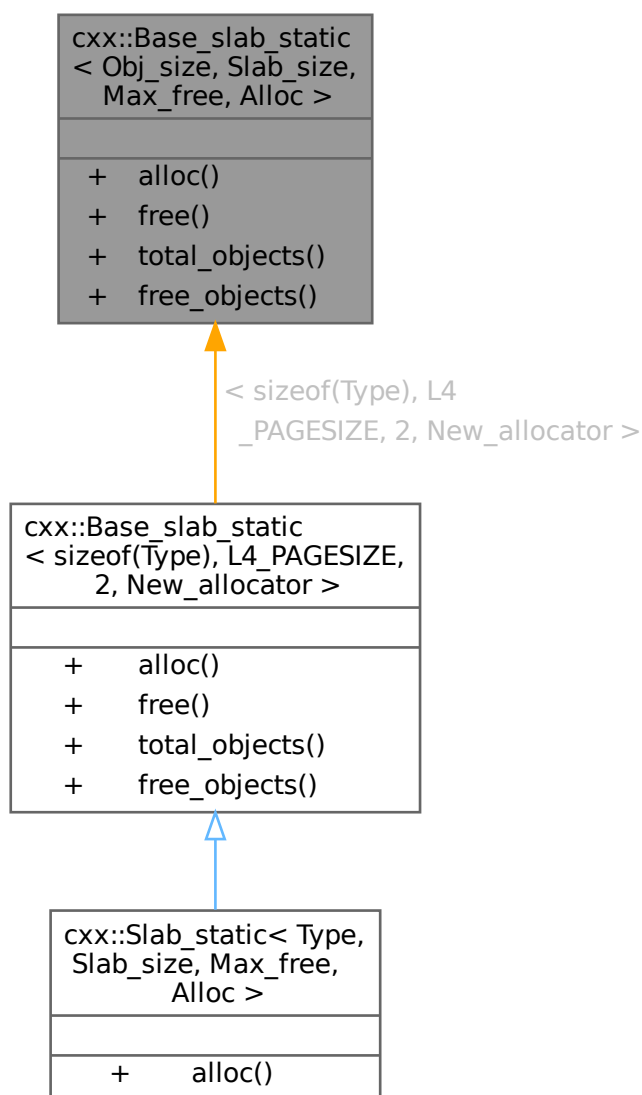
- l4/cxx/slab\_alloc

## 16.27 cxx::Base\_slab\_static< Obj\_size, Slab\_size, Max\_free, Alloc > Class Template Reference

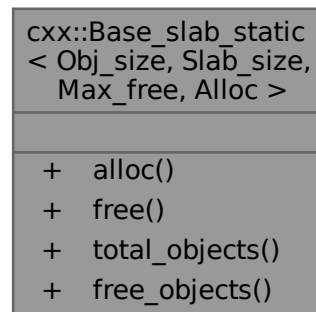
Merged slab allocator (allocators for objects of the same size are merged together).

```
#include <slab_alloc>
```

Inheritance diagram for cxx::Base\_slab\_static< Obj\_size, Slab\_size, Max\_free, Alloc >:



Collaboration diagram for `cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >`:



## Public Types

- enum { `object_size` = `Obj_size` , `slab_size` = `Slab_size` , `objects_per_slab` = `_A::objects_per_slab` , `max_free_slabs` = `Max_free` }

## Public Member Functions

- void \* `alloc` () noexcept  
*Allocate an object.*
- void `free` (void \*p) noexcept  
*Free the given object (p).*
- unsigned `total_objects` () const noexcept  
*Get the total number of objects managed by the slab allocator.*
- unsigned `free_objects` () const noexcept  
*Get the number of free objects in the slab allocator.*

## 16.27.1 Detailed Description

`template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class Alloc = New_allocator>`

`class cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >`

Merged slab allocator (allocators for objects of the same size are merged together).

## Template Parameters

<i>Obj_size</i>	The size of an object managed by the slab allocator.
<i>Slab_size</i>	The size of a slab.
<i>Max_free</i>	The maximum number of free slabs.

<i>Alloc</i>	The allocator for the slabs.
--------------	------------------------------

This slab allocator class is useful for merging slab allocators with the same parameters (equal `Obj_size`, `Slab_size`, `Max_free`, and `Alloc` parameters) together and share the overhead for the slab caches among all equal-sized objects.

Definition at line 388 of file [slab\\_alloc](#).

## 16.27.2 Member Enumeration Documentation

### 16.27.2.1 anonymous enum

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
anonymous enum
```

#### Enumerator

<code>object_size</code>	Size of an object.
<code>slab_size</code>	Size of a slab.
<code>objects_per_slab</code>	Number of objects per slab.
<code>max_free_slabs</code>	Maximum number of free slabs.

Definition at line 395 of file [slab\\_alloc](#).

## 16.27.3 Member Function Documentation

### 16.27.3.1 `alloc()`

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
void * cxx::Base\_slab\_static< Obj_size, Slab_size, Max_free, Alloc >::alloc () [inline],
[noexcept]
```

Allocate an object.

#### Note

The user is responsible for initializing the object.

Definition at line 412 of file [slab\\_alloc](#).

### 16.27.3.2 `free()`

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
void cxx::Base\_slab\_static< Obj_size, Slab_size, Max_free, Alloc >::free (
    void * p) [inline], [noexcept]
```

Free the given object (`p`).

#### Parameters

<i>p</i>	The pointer to the object to free.
----------	------------------------------------

**Precondition**

*p* must be a pointer to an object allocated by this allocator.

Definition at line 420 of file [slab\\_alloc](#).

**16.27.3.3 free\_objects()**

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
unsigned cxx::Base\_slab\_static< Obj_size, Slab_size, Max_free, Alloc >::free_objects () const
[inline], [noexcept]
```

Get the number of free objects in the slab allocator.

**Returns**

The number of free objects in all free and partially used slabs managed by this allocator.

**Note**

The value is the merged value for all equal parameterized [Base\\_slab\\_static](#) instances.

Definition at line 440 of file [slab\\_alloc](#).

**16.27.3.4 total\_objects()**

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
unsigned cxx::Base\_slab\_static< Obj_size, Slab_size, Max_free, Alloc >::total_objects () const
[inline], [noexcept]
```

Get the total number of objects managed by the slab allocator.

**Returns**

The number of objects managed by the allocator (including the free objects).

**Note**

The value is the merged value for all equal parameterized [Base\\_slab\\_static](#) instances.

Definition at line 430 of file [slab\\_alloc](#).

The documentation for this class was generated from the following file:

- I4/cxx/slab\_alloc



## 16.28 `cxx::Bitfield< T, LSB, MSB >` Class Template Reference

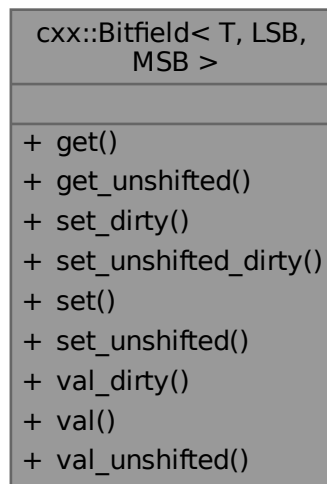
Definition for a member (part) of a bit field.

```
#include <bitfield>
```

Inheritance diagram for `cxx::Bitfield< T, LSB, MSB >`:



Collaboration diagram for `cxx::Bitfield< T, LSB, MSB >`:



### Data Structures

- class [Value\\_base](#)  
*Internal helper type.*
- class [Value](#)  
*Internal helper type.*
- class [Value\\_unshifted](#)  
*Internal helper type.*

## Public Types

- enum { **Bits** = MSB + 1 - LSB , **Lsb** = LSB , **Msb** = MSB }
- enum **Masks** : Base\_type { **Low\_mask** = static\_cast<Base\_type>(~0ULL) >> (sizeof(Base\_type)\*8 - Bits) , **Mask** = Low\_mask << Lsb }
- *Masks for bitwise operation on internal parts of a bitfield.*
- typedef Best\_type< **Bits** >::Type **Bits\_type**  
Type to hold at least **Bits** bits.
- typedef Best\_type< **Bits**+**Lsb** >::Type **Shift\_type**  
Type to hold at least **Bits** + **Lsb** bits.
- typedef Value< Base\_type & > **Ref**  
Reference type to access the bits inside a raw bit field.
- typedef Value< Base\_type volatile & > **Ref\_volatile**  
Volatile reference type to access the bits inside a raw bit field.
- typedef Value< Base\_type const > **Val**  
Value type to access the bits inside a raw bit field.
- typedef Value\_unshifted< Base\_type & > **Ref\_unshifted**  
Reference type to access the bits inside a raw bit field (in place).
- typedef Value\_unshifted< Base\_type volatile & > **Ref\_unshifted\_volatile**  
Volatile reference type to access the bits inside a raw bit field (in place).
- typedef Value\_unshifted< Base\_type const > **Val\_unshifted**  
Value type to access the bits inside a raw bit field (in place).

## Static Public Member Functions

- static constexpr **Bits\_type** get (Shift\_type val)  
Get the bits out of *val*.
- static constexpr Base\_type get\_unshifted (Shift\_type val)  
Get the bits in place out of *val*.
- static constexpr Base\_type set\_dirty (Base\_type dest, Shift\_type val)  
Set the bits corresponding to *val*.
- static constexpr Base\_type set\_unshifted\_dirty (Base\_type dest, Shift\_type val)  
Set the bits corresponding to *val*.
- static Base\_type set (Base\_type dest, Bits\_type val)  
Set the bits corresponding to *val*.
- static Base\_type set\_unshifted (Base\_type dest, Shift\_type val)  
Set the bits corresponding to *val*.
- static constexpr Base\_type val\_dirty (Shift\_type val)  
Get the shifted bits for *val*.
- static constexpr Base\_type val (Bits\_type val)  
Get the shifted bits for *val*.
- static constexpr Base\_type val\_unshifted (Shift\_type val)  
Get the shifted bits for *val*.

## 16.28.1 Detailed Description

```
template<typename T, unsigned LSB, unsigned MSB>
class cxx::Bitfield< T, LSB, MSB >
```

Definition for a member (part) of a bit field.

### Parameters

<i>T</i>	The underlying type of the bit field.
<i>LSB</i>	The least significant bit of our bits.
<i>MSB</i>	The most significant bit of our bits.

Definition at line 24 of file [bitfield](#).

## 16.28.2 Member Typedef Documentation

### 16.28.2.1 Bits\_type

```
template<typename T, unsigned LSB, unsigned MSB>
typedef Best_type<Bits>::Type cxx::Bitfield< T, LSB, MSB >::Bits_type
```

Type to hold at least [Bits](#) bits.

This type can handle all values that can be stored in this part of the bit field.

Definition at line 74 of file [bitfield](#).

### 16.28.2.2 Shift\_type

```
template<typename T, unsigned LSB, unsigned MSB>
typedef Best_type<Bits+Lsb>::Type cxx::Bitfield< T, LSB, MSB >::Shift_type
```

Type to hold at least [Bits](#) + [Lsb](#) bits.

This type can handle all values that can be stored in this part of the bit field when they are at the target location ([Lsb](#) bits shifted to the left).

Definition at line 82 of file [bitfield](#).

## 16.28.3 Member Enumeration Documentation

### 16.28.3.1 anonymous enum

```
template<typename T, unsigned LSB, unsigned MSB>
anonymous enum
```

#### Enumerator

Bits	Number of bits.
Lsb	index of the LSB
Msb	index of the MSB

Definition at line 52 of file [bitfield](#).

### 16.28.3.2 Masks

```
template<typename T, unsigned LSB, unsigned MSB>
enum cxx::Bitfield::Masks : Base_type
```

[Masks](#) for bitwise operation on internal parts of a bitfield.

#### Enumerator

---

Low_mask	Mask value to get <a href="#">Bits</a> bits.
Mask	Mask value to the bits out of a T.

Definition at line 60 of file [bitfield](#).

## 16.28.4 Member Function Documentation

### 16.28.4.1 `get()`

```
template<typename T, unsigned LSB, unsigned MSB>
constexpr Bits\_type cxx::Bitfield< T, LSB, MSB >::get (
    Shift\_type val) [inline], [static], [constexpr]
```

Get the bits out of [val](#).

#### Parameters

<a href="#">val</a>	The raw value of the whole bit field.
---------------------	---------------------------------------

#### Returns

The bits form [Lsb](#) to [Msb](#) shifted to the right.

Definition at line 99 of file [bitfield](#).

### 16.28.4.2 `get_unshifted()`

```
template<typename T, unsigned LSB, unsigned MSB>
constexpr Base_type cxx::Bitfield< T, LSB, MSB >::get_unshifted (
    Shift\_type val) [inline], [static], [constexpr]
```

Get the bits in place out of [val](#).

#### Parameters

<a href="#">val</a>	The raw value of the whole bit field.
---------------------	---------------------------------------

#### Returns

The bits from [Lsb](#) to [Msb](#) (unshifted).

This means other bits are masked out, however the result is not shifted to the right.

Definition at line 112 of file [bitfield](#).

### 16.28.4.3 `set()`

```
template<typename T, unsigned LSB, unsigned MSB>
Base_type cxx::Bitfield< T, LSB, MSB >::set (
    Base_type dest,
    Bits\_type val) [inline], [static]
```

Set the bits corresponding to [val](#).

#### Parameters

<i>dest</i>	The current value of the whole bit field.
<i>val</i>	The value to set into the bits.

**Returns**

The new value of the whole bit field.

Definition at line 161 of file [bitfield](#).

**16.28.4.4 `set_dirty()`**

```
template<typename T, unsigned LSB, unsigned MSB>
constexpr Base_type cxx::Bitfield< T, LSB, MSB >::set_dirty (
    Base_type dest,
    Shift_type val) [inline], [static], [constexpr]
```

Set the bits corresponding to `val`.

**Parameters**

<i>dest</i>	The current value of the whole bit field.
<i>val</i>	The value to set into the bits.

**Returns**

The new value of the whole bit field.

**Precondition**

`val` must not contain more than `Bits` bits.

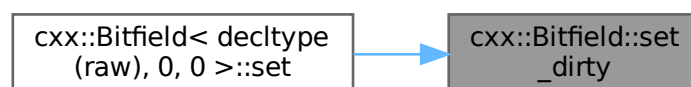
**Note**

This function does not mask `val` to the right number of bits.

Definition at line 127 of file [bitfield](#).

Referenced by `cxx::Bitfield< decltype(raw), 0, 0 >::set()`.

Here is the caller graph for this function:



#### 16.28.4.5 set\_unshifted()

```
template<typename T, unsigned LSB, unsigned MSB>
Base_type cxx::Bitfield< T, LSB, MSB >::set_unshifted (
    Base_type dest,
    Shift_type val) [inline], [static]
```

Set the bits corresponding to *val*.

##### Parameters

<i>dest</i>	The current value of the whole bit field.
<i>val</i>	The value shifted <i>Lsb</i> bits to the left that shall be set into the bit field.

##### Returns

the new value of the whole bit field.

Definition at line 173 of file [bitfield](#).

#### 16.28.4.6 set\_unshifted\_dirty()

```
template<typename T, unsigned LSB, unsigned MSB>
constexpr Base_type cxx::Bitfield< T, LSB, MSB >::set_unshifted_dirty (
    Base_type dest,
    Shift_type val) [inline], [static], [constexpr]
```

Set the bits corresponding to *val*.

##### Parameters

<i>dest</i>	The current value of the whole bit field.
<i>val</i>	The value shifted <i>Lsb</i> bits to the left that shall be set into the bits.

##### Returns

The new value of the whole bit field.

##### Precondition

*val* must not contain more than *Bits* bits shifted *Lsb* bits to the left.

**Note**

This function does not mask `val` to the right number of bits.

Definition at line 147 of file `bitfield`.

Referenced by `cxx::Bitfield< decltype(raw), 0, 0 >::set_unshifted()`.

Here is the caller graph for this function:

**16.28.4.7 val()**

```

template<typename T, unsigned LSB, unsigned MSB>
constexpr Base_type cxx::Bitfield< T, LSB, MSB >::val (
    Bits_type val) [inline], [static], [constexpr]
  
```

Get the shifted bits for `val`.

**Parameters**

<code>val</code>	The value to set into the bits.
------------------	---------------------------------

**Returns**

The raw bit field value.

Definition at line 196 of file `bitfield`.

**16.28.4.8 val\_dirty()**

```

template<typename T, unsigned LSB, unsigned MSB>
constexpr Base_type cxx::Bitfield< T, LSB, MSB >::val_dirty (
    Shift_type val) [inline], [static], [constexpr]
  
```

Get the shifted bits for `val`.

**Parameters**

<i>val</i>	The value to set into the bits.
------------	---------------------------------

**Returns**

The raw bit field value.

**Precondition**

*val* must not contain more than *Bits* bits.

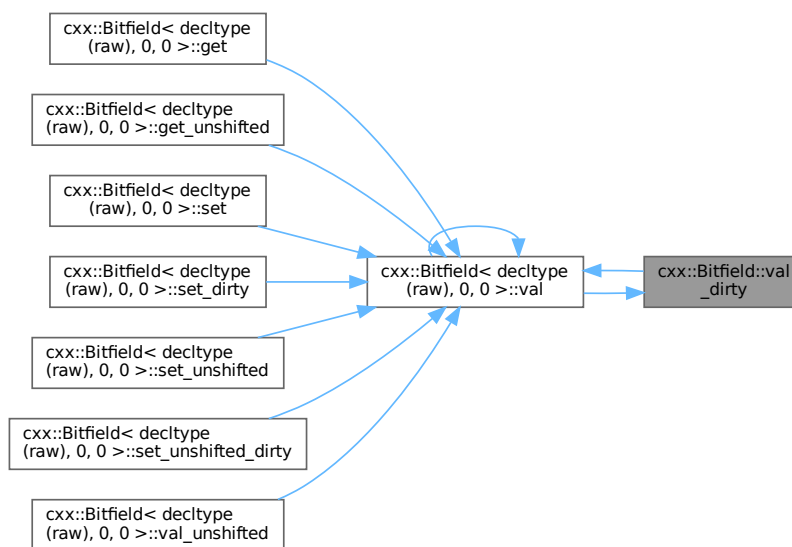
**Note**

This function does not mask *val* to the right number of bits.

Definition at line 187 of file [bitfield](#).

Referenced by [cxx::Bitfield< decltype\(raw\), 0, 0 >::val\(\)](#).

Here is the caller graph for this function:

**16.28.4.9 val\_unshifted()**

```

template<typename T, unsigned LSB, unsigned MSB>
constexpr Base_type cxx::Bitfield< T, LSB, MSB >::val_unshifted (
    Shift_type val) [inline], [static], [constexpr]

```

Get the shifted bits for *val*.

**Parameters**



<i>val</i>	The value shifted <code>Lsb</code> bits to the left that shall be set into the bits.
------------	--------------------------------------------------------------------------------------

**Returns**

The raw bit field value.

Definition at line 206 of file [bitfield](#).

The documentation for this class was generated from the following file:

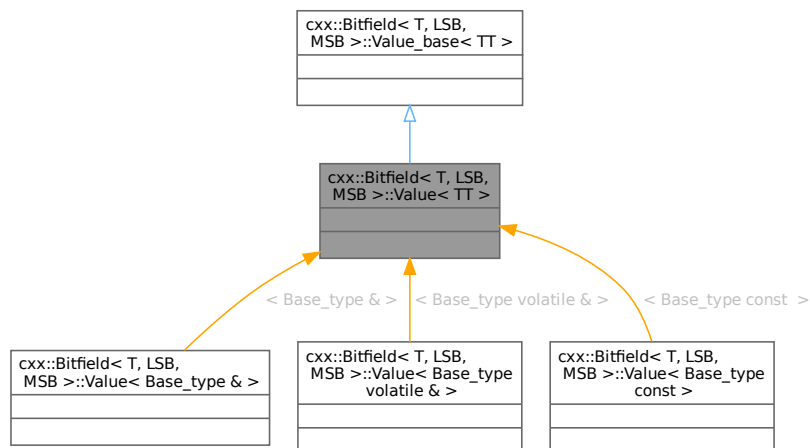
- `I4/cxx/bitfield`

## 16.29 `cxx::Bitfield< T, LSB, MSB >::Value< TT >` Class Template Reference

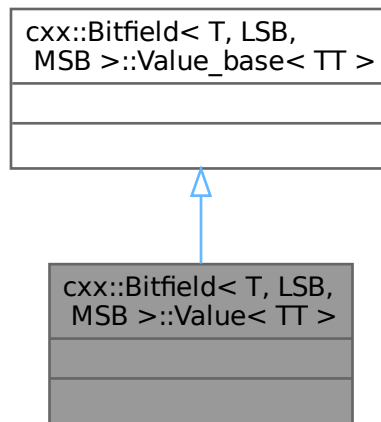
Internal helper type.

```
#include <bitfield>
```

Inheritance diagram for `cxx::Bitfield< T, LSB, MSB >::Value< TT >`:



Collaboration diagram for `cxx::Bitfield< T, LSB, MSB >::Value< TT >`:



### 16.29.1 Detailed Description

```

template<typename T, unsigned LSB, unsigned MSB>
template<typename TT>
class cxx::Bitfield< T, LSB, MSB >::Value< TT >
  
```

Internal helper type.

Definition at line 228 of file [bitfield](#).

The documentation for this class was generated from the following file:

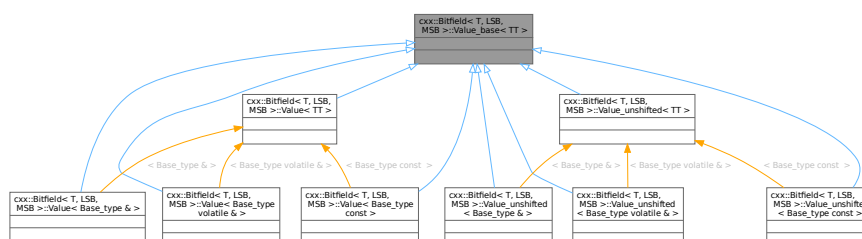
- `I4/cxx/bitfield`

## 16.30 `cxx::Bitfield< T, LSB, MSB >::Value_base< TT >` Class Template Reference

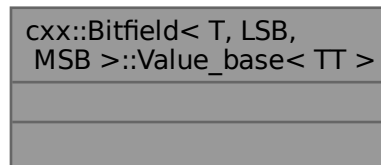
Internal helper type.

```
#include <bitfield>
```

Inheritance diagram for `cxx::Bitfield< T, LSB, MSB >::Value_base< TT >`:



Collaboration diagram for `cxx::Bitfield< T, LSB, MSB >::Value_base< TT >`:



### 16.30.1 Detailed Description

```

template<typename T, unsigned LSB, unsigned MSB>
template<typename TT>
class cxx::Bitfield< T, LSB, MSB >::Value_base< TT >

```

Internal helper type.

Definition at line 210 of file [bitfield](#).

The documentation for this class was generated from the following file:

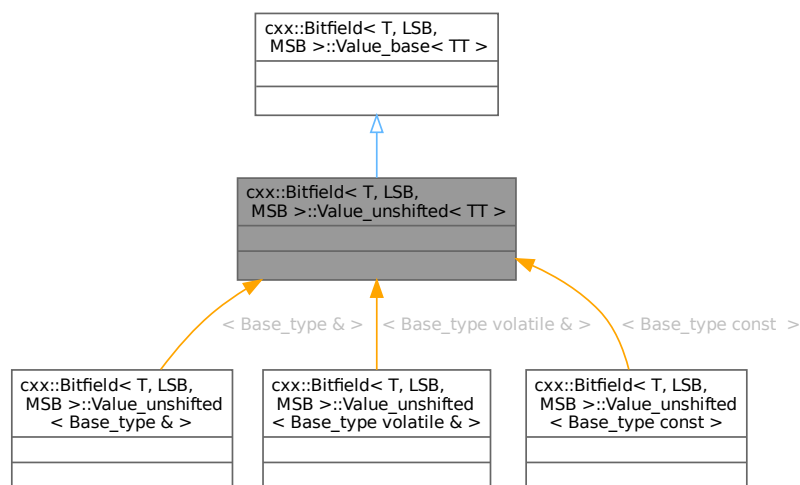
- `I4/cxx/bitfield`

## 16.31 `cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >` Class Template Reference

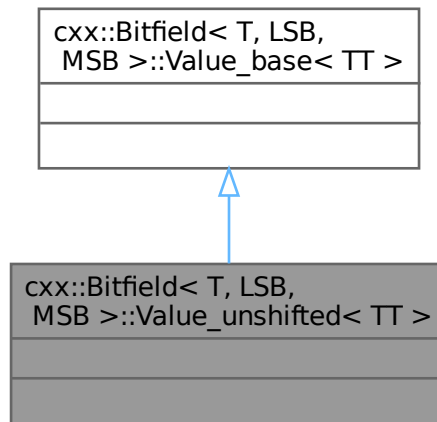
Internal helper type.

```
#include <bitfield>
```

Inheritance diagram for `cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >`:



Collaboration diagram for `cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >`:



### 16.31.1 Detailed Description

```

template<typename T, unsigned LSB, unsigned MSB>
template<typename TT>
class cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >

```

Internal helper type.

Definition at line 241 of file [bitfield](#).

The documentation for this class was generated from the following file:

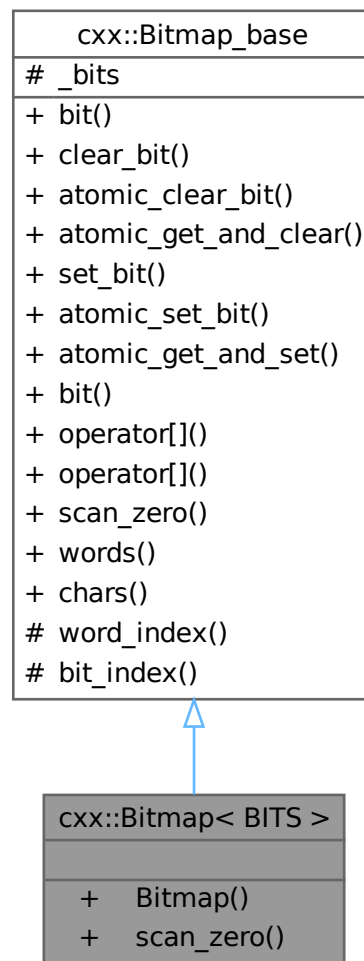
- `I4/cxx/bitfield`

## 16.32 `cxx::Bitmap< BITS >` Class Template Reference

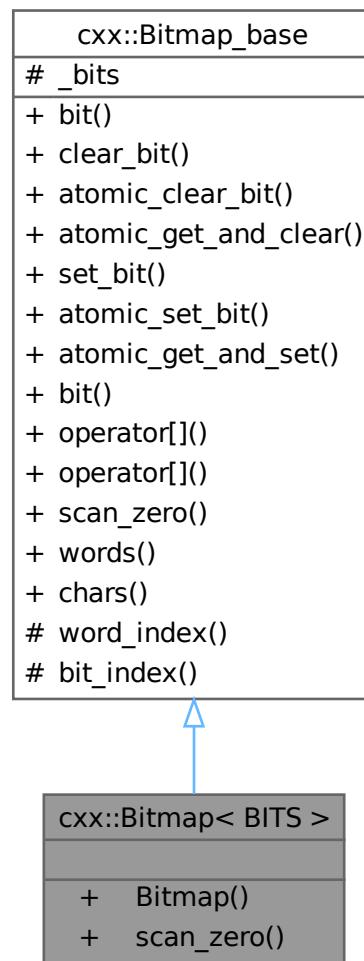
A static bitmap.

```
#include <bitmap>
```

Inheritance diagram for cxx::Bitmap< BITS >:



Collaboration diagram for `cxx::Bitmap< BITS >`:



## Public Member Functions

- **Bitmap** () noexcept  
Create a bitmap with *BITS* bits.
- long **scan\_zero** (long start\_bit=0) const noexcept  
Scan for the first zero bit.

## Public Member Functions inherited from `cxx::Bitmap_base`

- void **bit** (long bit, bool on) noexcept  
Set the value of bit *bit* to on.
- void **clear\_bit** (long bit) noexcept  
Clear bit *bit*.
- void **atomic\_clear\_bit** (long bit) noexcept

- Clear bit `bit` atomically.*
- `word_type atomic_get_and_clear` (long `bit`) noexcept  
*Clear bit `bit` atomically and return old state.*
- void `set_bit` (long `bit`) noexcept  
*Set bit `bit`.*
- void `atomic_set_bit` (long `bit`) noexcept  
*Set bit `bit` atomically.*
- `word_type atomic_get_and_set` (long `bit`) noexcept  
*Set bit `bit` atomically and return old state.*
- `word_type bit` (long `bit`) const noexcept  
*Get the truth value of a bit.*
- `word_type operator[]` (long `bit`) const noexcept  
*Get the bit at index `bit`.*
- `Bit operator[]` (long `bit`) noexcept  
*Get the lvalue for the bit at index `bit`.*
- long `scan_zero` (long `max_bit`, long `start_bit`=0) const noexcept  
*Scan for the first zero bit.*

### Additional Inherited Members

### Static Public Member Functions inherited from `cx::Bitmap_base`

- static long `words` (long bits) noexcept  
*Get the number of `Words` that are used for the bitmap.*
- static long `chars` (long bits) noexcept  
*Get the number of `chars` that are used for the bitmap.*

### Protected Types inherited from `cx::Bitmap_base`

- enum { `W_bits` = sizeof(`word_type`) \* 8 , `C_bits` = 8 }
- typedef unsigned long `word_type`  
*Data type for each element of the bit buffer.*

### Static Protected Member Functions inherited from `cx::Bitmap_base`

- static unsigned `word_index` (unsigned `bit`)  
*Get the word index for the given bit.*
- static unsigned `bit_index` (unsigned `bit`)  
*Get the bit index within `word_type` for the given bit.*

### Protected Attributes inherited from `cx::Bitmap_base`

- `word_type * _bits`  
*Pointer to the buffer storing the bits.*

## 16.32.1 Detailed Description

```
template<int BITS>
class cx::Bitmap< BITS >
```

A static bitmap.

### Template Parameters

<i>BITS</i>	The number of bits that shall be in the bitmap.
-------------	-------------------------------------------------

Definition at line 220 of file [bitmap](#).

## 16.32.2 Member Function Documentation

### 16.32.2.1 `scan_zero()`

```
template<int BITS>
long cxx::Bitmap< BITS >::scan_zero (
    long start_bit = 0) const [inline], [noexcept]
```

Scan for the first zero bit.

#### Parameters

<i>start_bit</i>	Hint at the number of the first bit to look at. Zero bits below <code>start_bit</code> may or may not be taken into account by the implementation.
------------------	----------------------------------------------------------------------------------------------------------------------------------------------------

#### Return values

<code>&gt;=</code>	0 Number of first zero bit found.
<code>-1</code>	All bits at <code>start_bit</code> or higher are set.

Compared to [Bitmap\\_base::scan\\_zero\(\)](#), the upper bound is set to BITS.

Definition at line 365 of file [bitmap](#).

References [cxx::Bitmap\\_base::scan\\_zero\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- `I4/cxx/bitmap`

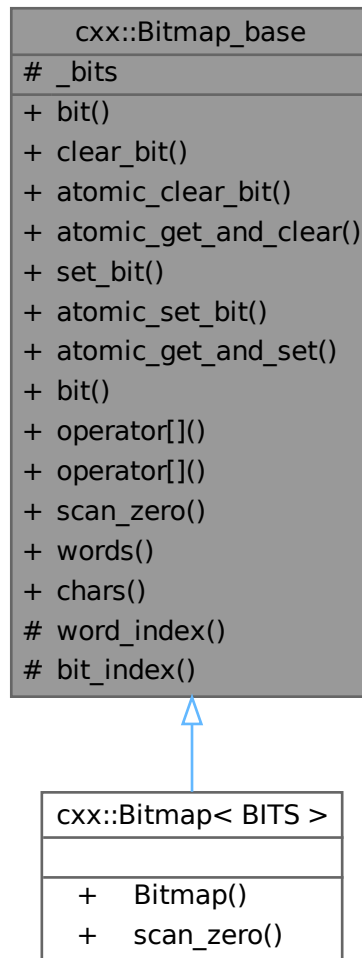


## 16.33 cxx::Bitmap\_base Class Reference

Basic bitmap abstraction.

```
#include <bitmap>
```

Inheritance diagram for cxx::Bitmap\_base:



Collaboration diagram for `cxx::Bitmap_base`:

<code>cxx::Bitmap_base</code>
<code># _bits</code>
<code>+ bit()</code>
<code>+ clear_bit()</code>
<code>+ atomic_clear_bit()</code>
<code>+ atomic_get_and_clear()</code>
<code>+ set_bit()</code>
<code>+ atomic_set_bit()</code>
<code>+ atomic_get_and_set()</code>
<code>+ bit()</code>
<code>+ operator[]()</code>
<code>+ operator[]()</code>
<code>+ scan_zero()</code>
<code>+ words()</code>
<code>+ chars()</code>
<code># word_index()</code>
<code># bit_index()</code>

## Data Structures

- class [Bit](#)  
*A writable bit in a bitmap.*
- class [Word](#)  
*Helper abstraction for a word contained in the bitmap.*
- class [Char](#)  
*Helper abstraction for a byte contained in the bitmap.*

## Public Member Functions

- void [bit](#) (long [bit](#), bool on) noexcept  
*Set the value of bit [bit](#) to on.*
- void [clear\\_bit](#) (long [bit](#)) noexcept  
*Clear bit [bit](#).*
- void [atomic\\_clear\\_bit](#) (long [bit](#)) noexcept  
*Clear bit [bit](#) atomically.*
- [word\\_type](#) [atomic\\_get\\_and\\_clear](#) (long [bit](#)) noexcept  
*Clear bit [bit](#) atomically and return old state.*
- void [set\\_bit](#) (long [bit](#)) noexcept  
*Set bit [bit](#).*

- void `atomic_set_bit` (long `bit`) noexcept  
*Set bit `bit` atomically.*
- `word_type` `atomic_get_and_set` (long `bit`) noexcept  
*Set bit `bit` atomically and return old state.*
- `word_type` `bit` (long `bit`) const noexcept  
*Get the truth value of a bit.*
- `word_type` `operator[]` (long `bit`) const noexcept  
*Get the bit at index `bit`.*
- `Bit` `operator[]` (long `bit`) noexcept  
*Get the lvalue for the bit at index `bit`.*
- long `scan_zero` (long `max_bit`, long `start_bit=0`) const noexcept  
*Scan for the first zero bit.*

### Static Public Member Functions

- static long `words` (long `bits`) noexcept  
*Get the number of `Words` that are used for the bitmap.*
- static long `chars` (long `bits`) noexcept  
*Get the number of chars that are used for the bitmap.*

### Protected Types

- enum { `W_bits` = sizeof(`word_type`) \* 8 , `C_bits` = 8 }
- typedef unsigned long `word_type`  
*Data type for each element of the bit buffer.*

### Static Protected Member Functions

- static unsigned `word_index` (unsigned `bit`)  
*Get the word index for the given bit.*
- static unsigned `bit_index` (unsigned `bit`)  
*Get the bit index within `word_type` for the given bit.*

### Protected Attributes

- `word_type` \* `_bits`  
*Pointer to the buffer storing the bits.*

## 16.33.1 Detailed Description

Basic bitmap abstraction.

This abstraction keeps a pointer to a memory area that is used as bitmap.

Definition at line 18 of file `bitmap`.

## 16.33.2 Member Enumeration Documentation

### 16.33.2.1 anonymous enum

```
anonymous enum [protected]
```

#### Enumerator

---

W_bits	number of bits in <a href="#">word_type</a>
C_bits	number of bits in char

Definition at line 26 of file [bitmap](#).

### 16.33.3 Member Function Documentation

#### 16.33.3.1 `atomic_clear_bit()`

```
void cxx::Bitmap_base::atomic_clear_bit (
    long bit) [inline], [noexcept]
```

Clear bit [bit](#) atomically.

Use this function for multi-threaded access to the bitmap.

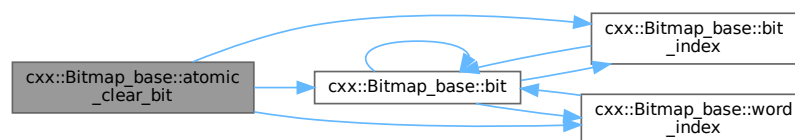
##### Parameters

<i>bit</i>	The number of the bit to clear.
------------	---------------------------------

Definition at line 269 of file [bitmap](#).

References [\\_bits](#), [bit\(\)](#), [bit\\_index\(\)](#), and [word\\_index\(\)](#).

Here is the call graph for this function:



#### 16.33.3.2 `atomic_get_and_clear()`

```
Bitmap_base::word_type cxx::Bitmap_base::atomic_get_and_clear (
    long bit) [inline], [noexcept]
```

Clear bit [bit](#) atomically and return old state.

Use this function for multi-threaded access to the bitmap.

##### Parameters

<i>bit</i>	The number of the bit to clear.
------------	---------------------------------

Definition at line 279 of file [bitmap](#).

References [\\_bits](#), [bit\(\)](#), [bit\\_index\(\)](#), and [word\\_index\(\)](#).

Here is the call graph for this function:



### 16.33.3.3 atomic\_get\_and\_set()

```

Bitmap_base::word_type cxx::Bitmap_base::atomic_get_and_set (
    long bit) [inline], [noexcept]

```

Set bit [bit](#) atomically and return old state.

Use this function for multi-threaded access to the bitmap.

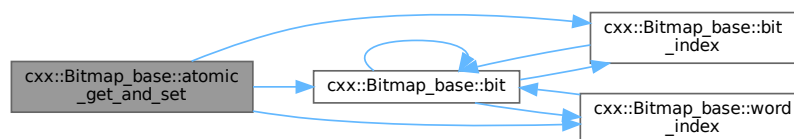
#### Parameters

<i>bit</i>	The number of the bit to set.
------------	-------------------------------

Definition at line 308 of file [bitmap](#).

References [\\_bits](#), [bit\(\)](#), [bit\\_index\(\)](#), and [word\\_index\(\)](#).

Here is the call graph for this function:



### 16.33.3.4 atomic\_set\_bit()

```

void cxx::Bitmap_base::atomic_set_bit (
    long bit) [inline], [noexcept]

```

Set bit [bit](#) atomically.

Use this function for multi-threaded access to the bitmap.

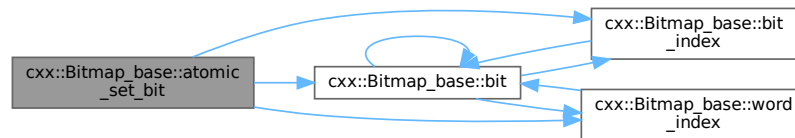
#### Parameters

<i>bit</i>	The number of the bit to set.
------------	-------------------------------

Definition at line 298 of file [bitmap](#).

References [\\_bits](#), [bit\(\)](#), [bit\\_index\(\)](#), and [word\\_index\(\)](#).

Here is the call graph for this function:



### 16.33.3.5 bit() [1/2]

```

Bitmap_base::word_type cxx::Bitmap_base::bit (
    long bit) const [inline], [noexcept]

```

Get the truth value of a bit.

#### Parameters

<i>bit</i>	The number of the bit to read.
------------	--------------------------------

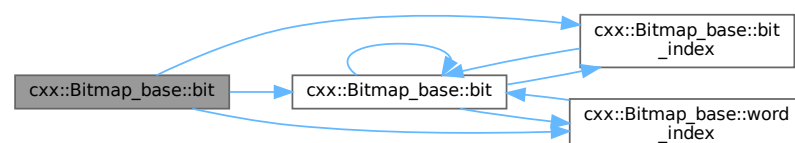
#### Return values

0	Bit is not set.
!= 0	Bit is set.

Definition at line 318 of file [bitmap](#).

References [\\_bits](#), [bit\(\)](#), [bit\\_index\(\)](#), and [word\\_index\(\)](#).

Here is the call graph for this function:



## 16.33.3.6 bit() [2/2]

```
void cxx::Bitmap_base::bit (
    long bit,
    bool on) [inline], [noexcept]
```

Set the value of bit `bit` to on.

## Parameters

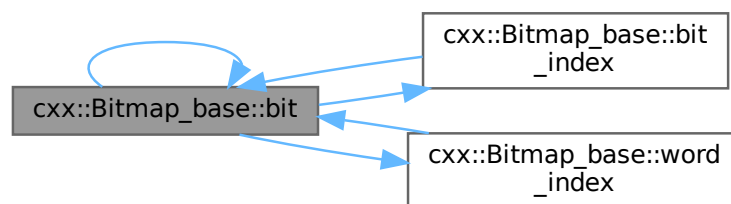
<i>bit</i>	The number of the bit.
<i>on</i>	The boolean value that shall be assigned to the bit.

Definition at line 251 of file [bitmap](#).

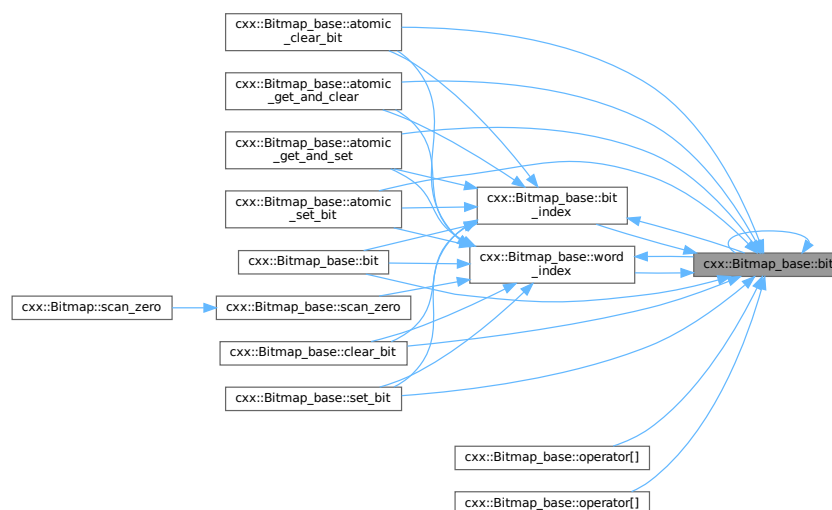
References [\\_bits](#), [bit\(\)](#), [bit\\_index\(\)](#), and [word\\_index\(\)](#).

Referenced by [atomic\\_clear\\_bit\(\)](#), [atomic\\_get\\_and\\_clear\(\)](#), [atomic\\_get\\_and\\_set\(\)](#), [atomic\\_set\\_bit\(\)](#), [bit\(\)](#), [bit\(\)](#), [bit\\_index\(\)](#), [clear\\_bit\(\)](#), [operator\[\]\(\)](#), [operator\[\]\(\)](#), [set\\_bit\(\)](#), and [word\\_index\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.33.3.7 bit\_index()

```
unsigned cxx::Bitmap_base::bit_index (
    unsigned bit) [inline], [static], [protected]
```

Get the bit index within [word\\_type](#) for the given bit.

#### Parameters

<i>bit</i>	The bit index in the bitmap.
------------	------------------------------

#### Returns

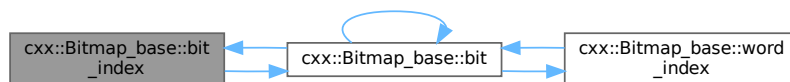
the bit index within [word\\_type](#) ( $\text{bit} \% W\_bits$ ).

Definition at line 53 of file [bitmap](#).

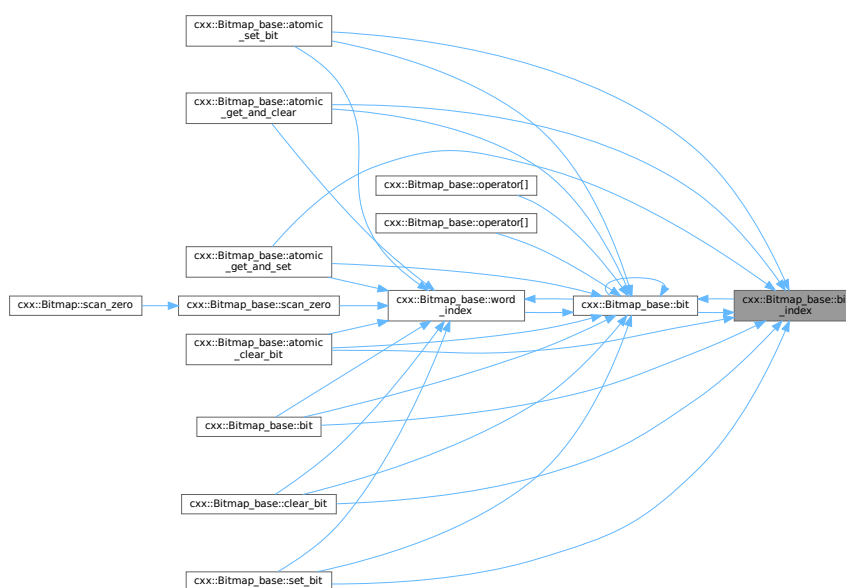
References [bit\(\)](#), and [W\\_bits](#).

Referenced by [atomic\\_clear\\_bit\(\)](#), [atomic\\_get\\_and\\_clear\(\)](#), [atomic\\_get\\_and\\_set\(\)](#), [atomic\\_set\\_bit\(\)](#), [bit\(\)](#), [bit\(\)](#), [clear\\_bit\(\)](#), and [set\\_bit\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:





## 16.33.3.8 clear\_bit()

```
void cxx::Bitmap_base::clear_bit (
    long bit) [inline], [noexcept]
```

Clear bit [bit](#).

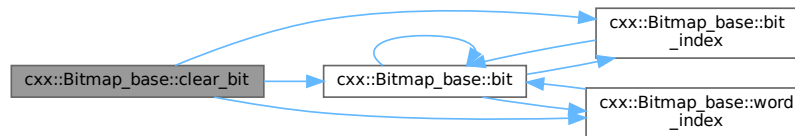
## Parameters

<i>bit</i>	The number of the bit to clear.
------------	---------------------------------

Definition at line 260 of file [bitmap](#).

References [\\_bits](#), [bit\(\)](#), [bit\\_index\(\)](#), and [word\\_index\(\)](#).

Here is the call graph for this function:



## 16.33.3.9 operator[]() [1/2]

```
word_type cxx::Bitmap_base::operator[] (
    long bit) const [inline], [noexcept]
```

Get the bit at index [bit](#).

## Parameters

<i>bit</i>	The number of the bit to read.
------------	--------------------------------

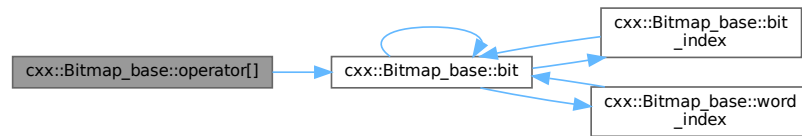
## Return values

0	<a href="#">Bit</a> is not set.
!= 0	<a href="#">Bit</a> is set.

Definition at line 181 of file [bitmap](#).

References [bit\(\)](#).

Here is the call graph for this function:



### 16.33.3.10 operator[]() [2/2]

```

Bit cxx::Bitmap_base::operator[] (
    long bit) [inline], [noexcept]
  
```

Get the lvalue for the bit at index `bit`.

#### Parameters

<i>bit</i>	The number.
------------	-------------

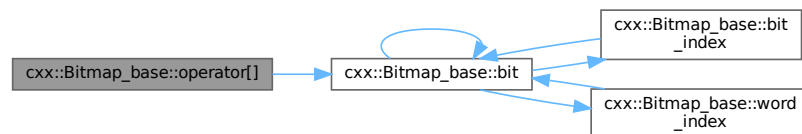
#### Returns

lvalue for `bit`

Definition at line 191 of file `bitmap`.

References `bit()`.

Here is the call graph for this function:



### 16.33.3.11 scan\_zero()

```

long cxx::Bitmap_base::scan_zero (
    long max_bit,
    long start_bit = 0) const [inline], [noexcept]
  
```

Scan for the first zero bit.

#### Parameters

<i>max_bit</i>	Upper bound (exclusive) for the scanning operation.
<i>start_bit</i>	Hint at the number of the first bit to look at. Zero bits below <i>start_bit</i> may or may not be taken into account by the implementation.

### Return values

<i>&gt;=</i>	0 Number of first zero bit found.
<i>-1</i>	All bits between <i>start_bit</i> and <i>max_bit</i> are set.

Definition at line 339 of file [bitmap](#).

References [\\_bits](#), [W\\_bits](#), and [word\\_index\(\)](#).

Referenced by [cxx::Bitmap< BITS >::scan\\_zero\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.33.3.12 set\_bit()

```
void cxx::Bitmap_base::set_bit (
    long bit) [inline], [noexcept]
```

Set bit [bit](#).

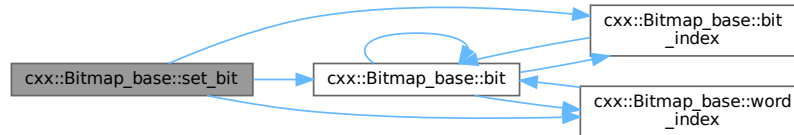
### Parameters

<i>bit</i>	The number of the bit to set.
------------	-------------------------------

Definition at line 289 of file [bitmap](#).

References [\\_bits](#), [bit\(\)](#), [bit\\_index\(\)](#), and [word\\_index\(\)](#).

Here is the call graph for this function:



### 16.33.3.13 word\_index()

```
unsigned cxx::Bitmap_base::word_index (
    unsigned bit) [inline], [static], [protected]
```

Get the word index for the given bit.

#### Parameters

<i>bit</i>	The index of the bit in question.
------------	-----------------------------------

#### Returns

the index in [Bitmap\\_base::\\_bits](#) for the given bit (`bit / W_bits`).

Definition at line 44 of file [bitmap](#).

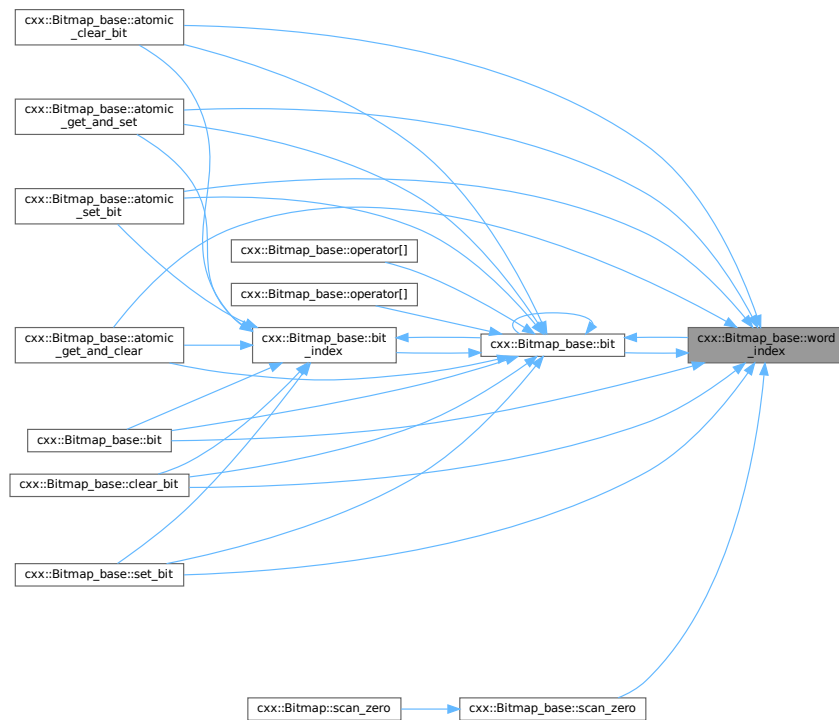
References [bit\(\)](#), and [W\\_bits](#).

Referenced by [atomic\\_clear\\_bit\(\)](#), [atomic\\_get\\_and\\_clear\(\)](#), [atomic\\_get\\_and\\_set\(\)](#), [atomic\\_set\\_bit\(\)](#), [bit\(\)](#), [bit\(\)](#), [clear\\_bit\(\)](#), [scan\\_zero\(\)](#), and [set\\_bit\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

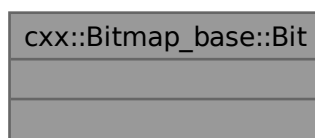
- `I4/cxx/bitmap`

## 16.34 cxx::Bitmap\_base::Bit Class Reference

A writable bit in a bitmap.

```
#include <bitmap>
```

Collaboration diagram for `cxx::Bitmap_base::Bit`:



### 16.34.1 Detailed Description

A writable bit in a bitmap.

Definition at line 58 of file [bitmap](#).

The documentation for this class was generated from the following file:

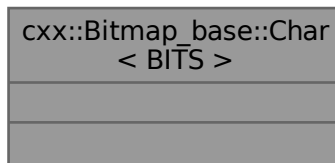
- I4/cxx/bitmap

## 16.35 cxx::Bitmap\_base::Char< BITS > Class Template Reference

Helper abstraction for a byte contained in the bitmap.

```
#include <bitmap>
```

Collaboration diagram for cxx::Bitmap\_base::Char< BITS >:



### 16.35.1 Detailed Description

```
template<long BITS>
class cxx::Bitmap_base::Char< BITS >
```

Helper abstraction for a byte contained in the bitmap.

Definition at line 95 of file [bitmap](#).

The documentation for this class was generated from the following file:

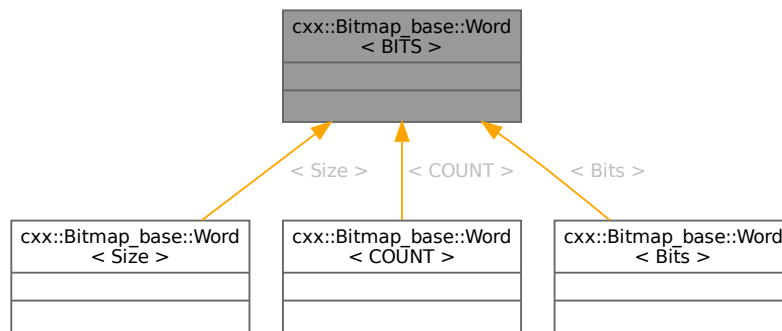
- I4/cxx/bitmap

## 16.36 cxx::Bitmap\_base::Word< BITS > Class Template Reference

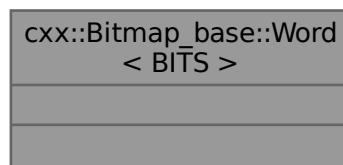
Helper abstraction for a word contained in the bitmap.

```
#include <bitmap>
```

Inheritance diagram for cxx::Bitmap\_base::Word< BITS >:



Collaboration diagram for cxx::Bitmap\_base::Word< BITS >:



### 16.36.1 Detailed Description

```
template<long BITS>
class cxx::Bitmap_base::Word< BITS >
```

Helper abstraction for a word contained in the bitmap.

Definition at line 79 of file [bitmap](#).

The documentation for this class was generated from the following file:

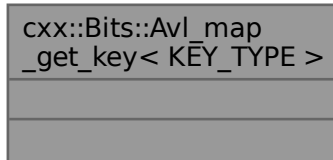
- `I4/cxx/bitmap`

## 16.37 `cxx::Bits::Avl_map_get_key< KEY_TYPE >` Struct Template Reference

Key-getter for [Avl\\_map](#).

```
#include <avl_map>
```

Collaboration diagram for `cxx::Bits::Avl_map_get_key< KEY_TYPE >`:



### 16.37.1 Detailed Description

```
template<typename KEY_TYPE>
struct cxx::Bits::Avl_map_get_key< KEY_TYPE >
```

Key-getter for [Avl\\_map](#).

Definition at line 25 of file [avl\\_map](#).

The documentation for this struct was generated from the following file:

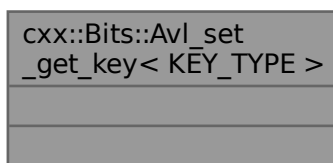
- [l4/cxx/avl\\_map](#)

## 16.38 `cxx::Bits::Avl_set_get_key< KEY_TYPE >` Struct Template Reference

Internal, key-getter for [Avl\\_set](#) nodes.

```
#include <avl_set>
```

Collaboration diagram for `cxx::Bits::Avl_set_get_key< KEY_TYPE >`:





### 16.38.1 Detailed Description

```
template<typename KEY_TYPE>
struct cxx::Bits::Avl_set_get_key< KEY_TYPE >
```

Internal, key-getter for [Avl\\_set](#) nodes.

Definition at line 98 of file [avl\\_set](#).

The documentation for this struct was generated from the following file:

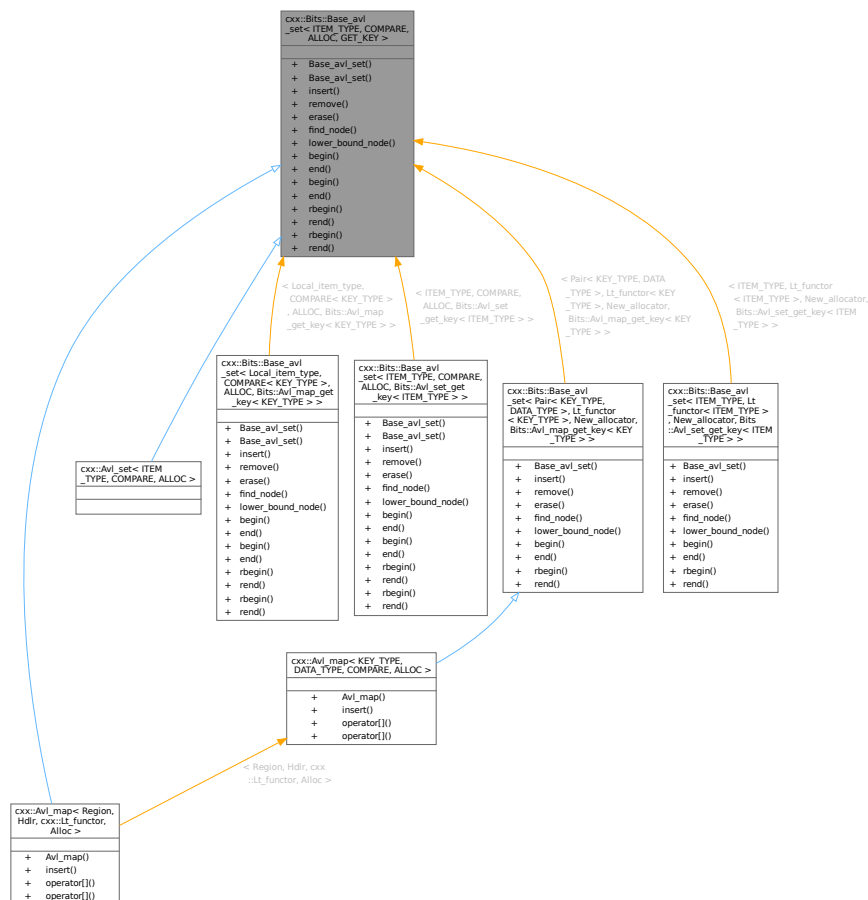
- [l4/cxx/avl\\_set](#)

## 16.39 cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY > Class Template Reference

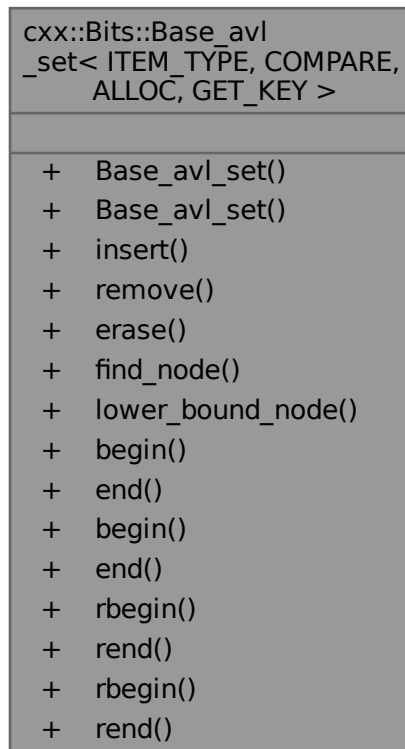
Internal: AVL set with internally managed nodes.

```
#include <avl_set>
```

Inheritance diagram for cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >:



Collaboration diagram for `cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >`:



## Data Structures

- class [Node](#)

*A smart pointer to a tree item.*

## Public Types

- enum { [E\\_noent](#) = 2 , [E\\_exist](#) = 17 , [E\\_nomem](#) = 12 , [E\\_inval](#) = 22 }
- Return status constants.*
- typedef ITEM\_TYPE **Item\_type**
- Type for the items store in the set.*
- typedef GET\_KEY **Get\_key**
- Key-getter type to derive the sort key of an internal node.*
- typedef GET\_KEY::Key\_type **Key\_type**
- Type of the sort key used for the items.*
- typedef Type\_traits< [Item\\_type](#) >::Const\_type **Const\_item\_type**
- Type used for const items within the set.*
- typedef COMPARE **Item\_compare**
- Type for the comparison functor.*
- typedef ALLOC< \_Node > **Node\_allocator**

*Type for the node allocator.*

- typedef Avl\_set\_iter< \_Node, [Item\\_type](#), Fwd > **Iterator**

*Forward iterator for the set.*

- typedef Avl\_set\_iter< \_Node, [Const\\_item\\_type](#), Fwd > **Const\_iterator**

*Constant forward iterator for the set.*

- typedef Avl\_set\_iter< \_Node, [Item\\_type](#), Rev > **Rev\_iterator**

*Backward iterator for the set.*

- typedef Avl\_set\_iter< \_Node, [Const\\_item\\_type](#), Rev > **Const\_rev\_iterator**

*Constant backward iterator for the set.*

## Public Member Functions

- [Base\\_avl\\_set](#) ([Node\\_allocator](#) const &alloc=[Node\\_allocator](#)())

*Create a AVL-tree based set.*

- [Base\\_avl\\_set](#) ([Base\\_avl\\_set](#) const &o)

*Create a copy of an AVL-tree based set.*

- [cxx::Pair](#)< [Iterator](#), int > [insert](#) ([Item\\_type](#) const &item)

*Insert an item into the set.*

- int [remove](#) ([Key\\_type](#) const &item)

*Remove an item from the set.*

- int [erase](#) ([Key\\_type](#) const &item)

*Erase the item with the given key.*

- [Node](#) [find\\_node](#) ([Key\\_type](#) const &item) const

*Lookup a node equal to *item*.*

- [Node](#) [lower\\_bound\\_node](#) ([Key\\_type](#) const &key) const

*Find the first node greater or equal to *key*.*

- [Const\\_iterator](#) [begin](#) () const

*Get the constant forward iterator for the first element in the set.*

- [Const\\_iterator](#) [end](#) () const

*Get the end marker for the constant forward iterator.*

- [Iterator](#) [begin](#) ()

*Get the mutable forward iterator for the first element of the set.*

- [Iterator](#) [end](#) ()

*Get the end marker for the mutable forward iterator.*

- [Const\\_rev\\_iterator](#) [rbegin](#) () const

*Get the constant backward iterator for the last element in the set.*

- [Const\\_rev\\_iterator](#) [rend](#) () const

*Get the end marker for the constant backward iterator.*

- [Rev\\_iterator](#) [rbegin](#) ()

*Get the mutable backward iterator for the last element of the set.*

- [Rev\\_iterator](#) [rend](#) ()

*Get the end marker for the mutable backward iterator.*

## 16.39.1 Detailed Description

template<typename ITEM\_TYPE, class COMPARE, template< typename A > class ALLOC, typename GET\_KEY>

class cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >

Internal: AVL set with internally managed nodes.

Use [Avl\\_set](#), [Avl\\_map](#), or [Avl\\_tree](#) in applications.

## Template Parameters

<i>ITEM_TYPE</i>	The type of the items to be stored in the set.
<i>COMPARE</i>	The relation to define the partial order, default is to use operator '<'.
<i>ALLOC</i>	The allocator to use for the nodes of the AVL set.
<i>GET_KEY</i>	Sort-key getter (must provide the <a href="#">Key_type</a> and sort-key for an item (of <i>ITEM_TYPE</i> )).

Definition at line [122](#) of file [avl\\_set](#).

## 16.39.2 Member Enumeration Documentation

### 16.39.2.1 anonymous enum

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
anonymous enum
```

Return status constants.

These constants are compatible with the [L4](#) error codes, see [l4\\_error\\_code\\_t](#).

#### Enumerator

E_noent	Item does not exist.
E_exist	Item exists already.
E_nomem	Memory allocation failed.
E_inval	Internal error.

Definition at line [133](#) of file [avl\\_set](#).

## 16.39.3 Constructor & Destructor Documentation

### 16.39.3.1 Base\_avl\_set() [1/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Base_avl_set (
    Node_allocator const & alloc = Node_allocator()) [inline], [explicit]
```

Create a AVL-tree based set.

#### Parameters

<i>alloc</i>	<a href="#">Node</a> allocator.
--------------	---------------------------------

Create an empty set (AVL-tree based).

Definition at line [243](#) of file [avl\\_set](#).

Referenced by [Base\\_avl\\_set\(\)](#).

Here is the caller graph for this function:



### 16.39.3.2 Base\_avl\_set() [2/2]

```
template<typename Item, class Compare, template< typename A > class Alloc, typename KEY_TYPE>
cxx::Bits::Base_avl_set< Item, Compare, Alloc, KEY_TYPE >::Base_avl_set (
    Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY > const & o) [inline]
```

Create a copy of an AVL-tree based set.

#### Parameters

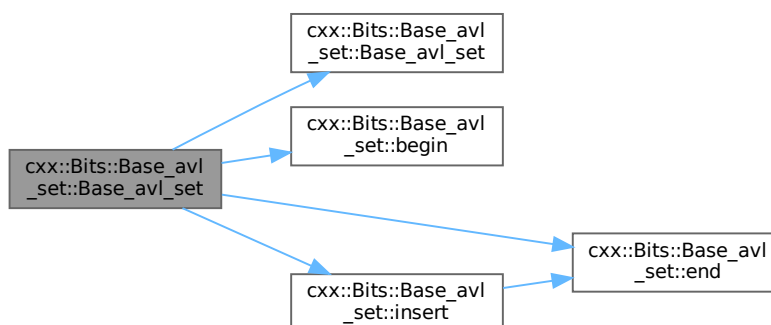
<i>o</i>	The set to copy.
----------	------------------

Creates a deep copy of the set with all its items.

Definition at line 402 of file [avl\\_set](#).

References [Base\\_avl\\_set\(\)](#), [begin\(\)](#), [end\(\)](#), and [insert\(\)](#).

Here is the call graph for this function:



## 16.39.4 Member Function Documentation

### 16.39.4.1 `begin()` [1/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
```

```
Iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::begin () [inline]
```

Get the mutable forward iterator for the first element of the set.

#### Returns

The mutable forward iterator for the first element of the set.

Definition at line 356 of file `avl_set`.

### 16.39.4.2 `begin()` [2/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
```

```
Const_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::begin () const
[inline]
```

Get the constant forward iterator for the first element in the set.

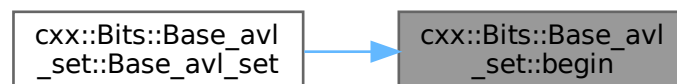
#### Returns

Constant forward iterator for the first element in the set.

Definition at line 345 of file `avl_set`.

Referenced by `Base_avl_set()`.

Here is the caller graph for this function:



**16.39.4.3 end()** [1/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
Iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::end () [inline]
```

Get the end marker for the mutable forward iterator.

**Returns**

The end marker for mutable forward iterator.

Definition at line 361 of file [avl\\_set](#).

**16.39.4.4 end()** [2/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
Const_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::end () const
[inline]
```

Get the end marker for the constant forward iterator.

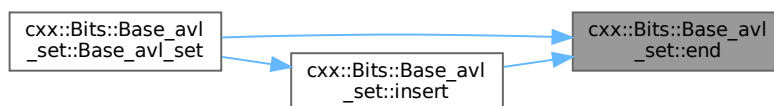
**Returns**

The end marker for the constant forward iterator.

Definition at line 350 of file [avl\\_set](#).

Referenced by [Base\\_avl\\_set\(\)](#), and [insert\(\)](#).

Here is the caller graph for this function:

**16.39.4.5 erase()**

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
int cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::erase (
    Key_type const & item) [inline]
```

Erase the item with the given key.

**Parameters**

<i>item</i>	The key of the item to remove.
-------------	--------------------------------

Definition at line 313 of file [avl\\_set](#).

#### 16.39.4.6 find\_node()

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
Node cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::find_node (
    Key_type const & item) const [inline]
```

Lookup a node equal to *item*.

##### Parameters

<i>item</i>	The value to search for.
-------------	--------------------------

##### Returns

A smart pointer to the element found. If no element was found the smart pointer will be invalid.

Definition at line 324 of file [avl\\_set](#).

#### 16.39.4.7 insert()

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
Pair< typename Base_avl_set< Item, Compare, Alloc, KEY_TYPE >::Iterator, int > cxx::Bits::Base_avl_set<
Item, Compare, Alloc, KEY_TYPE >::insert (
    Item_type const & item)
```

Insert an item into the set.

##### Parameters

<i>item</i>	The item to insert.
-------------	---------------------

##### Returns

A pair of iterator (*first*) and return value (*second*). *second* will be 0 if the element was inserted into the set and `-#E_exist` if the element was already in the set and the set was therefore not updated. In both cases, *first* contains an iterator that points to the element. *second* may also be `-#E_nomem` when memory for the node could not be allocated. *first* is then invalid.



Insert a new item into the set, each item can only be once in the set.

Definition at line 412 of file `avl_set`.

References `E_exist`, `E_nomem`, `end()`, `cxx::Pair< First, Second >::first`, and `cxx::Pair< First, Second >::second`.

Referenced by `Base_avl_set()`.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 16.39.4.8 `lower_bound_node()`

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GET_KEY>
```

```
Node cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::lower_bound_node (
    Key_type const & key) const [inline]
```

Find the first node greater or equal to `key`.

##### Parameters

<i>key</i>	Minimum key to look for.
------------	--------------------------

##### Returns

Smart pointer to the first node greater or equal to `key`. Will be invalid if no such element was found.

Definition at line 335 of file `avl_set`.

**16.39.4.9 rbegin()** [1/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
Rev_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::rbegin () [inline]
```

Get the mutable backward iterator for the last element of the set.

**Returns**

The mutable backward iterator for the last element of the set.

Definition at line 378 of file [avl\\_set](#).

**16.39.4.10 rbegin()** [2/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
Const_rev_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::rbegin ()
const [inline]
```

Get the constant backward iterator for the last element in the set.

**Returns**

The constant backward iterator for the last element in the set.

Definition at line 367 of file [avl\\_set](#).

**16.39.4.11 remove()**

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
int cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::remove (
    Key_type const & item) [inline]
```

Remove an item from the set.

**Parameters**

<i>item</i>	The item to remove.
-------------	---------------------

**Return values**

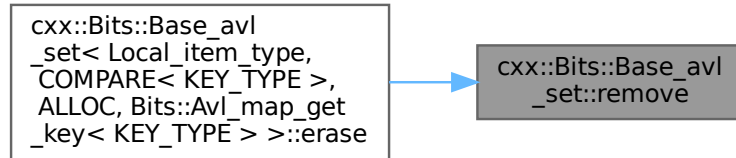
0	Success
-E_noent	Item does not exist

Definition at line 295 of file [avl\\_set](#).

## 16.39 cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY > Class Template Reference 007

Referenced by [cxx::Bits::Base\\_avl\\_set< Local\\_item\\_type, COMPARE< KEY\\_TYPE >, ALLOC, Bits::Avl\\_map\\_get\\_key< KEY\\_TYPE >>](#)

Here is the caller graph for this function:



### 16.39.4.12 `rend()` [1/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GET_KEY>
Rev_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::rend () [inline]
```

Get the end marker for the mutable backward iterator.

#### Returns

The end marker for mutable backward iterator.

Definition at line 383 of file [avl\\_set](#).

### 16.39.4.13 `rend()` [2/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GET_KEY>
Const_rev_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::rend ()
const [inline]
```

Get the end marker for the constant backward iterator.

#### Returns

The end marker for the constant backward iterator.

Definition at line 372 of file [avl\\_set](#).

The documentation for this class was generated from the following file:

- [I4/cxx/avl\\_set](#)

## 16.40 cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >::Node Class Reference

A smart pointer to a tree item.

```
#include <avl_set>
```

Collaboration diagram for cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >::Node:

cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node	
+	Node()
+	operator*()
+	operator->()
+	valid()
+	operator Item_type const *()

### Public Member Functions

- **Node ()**  
*Default construction for NIL pointer.*
- **Item\_type const & operator\* ()**  
*Dereference the pointer.*
- **Item\_type const \* operator-> ()**  
*Dereferenced member access.*
- **bool valid () const**  
*Validity check.*
- **operator Item\_type const \* ()**  
*Cast to a real item pointer.*

### 16.40.1 Detailed Description

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GET_KEY>
class cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node
```

A smart pointer to a tree item.

Definition at line 172 of file [avl\\_set](#).

## 16.40.2 Member Function Documentation

### 16.40.2.1 `operator*()`

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
Item_type const & cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node::operator*
() [inline]
```

Dereference the pointer.

#### Precondition

`Node` is valid.

Definition at line 189 of file `avl_set`.

### 16.40.2.2 `operator->()`

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
Item_type const * cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node::operator->
() [inline]
```

Dereferenced member access.

#### Precondition

`Node` is valid.

Definition at line 195 of file `avl_set`.

### 16.40.2.3 `valid()`

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
bool cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node::valid () const
[inline]
```

Validity check.

#### Returns

false if the pointer is NIL, true if valid.

Definition at line 201 of file `avl_set`.

The documentation for this class was generated from the following file:

- `I4/cxx/avl_set`



## Public Member Functions

- `bool empty () const`  
*Check if the list is empty.*
- `Value_type front () const`  
*Return the first element in the list.*
- `void clear ()`  
*Remove all elements from the list.*
- `Iterator begin ()`  
*Return an iterator to the beginning of the list.*
- `Const_iterator begin () const`  
*Return a const iterator to the beginning of the list.*
- `Const_iterator end () const`  
*Return a const iterator to the end of the list.*
- `Iterator end ()`  
*Return an iterator to the end of the list.*

## Static Public Member Functions

- `static Const_iterator iter (Const_value_type c)`  
*Return a const iterator that begins at the given element.*

## Protected Attributes

- `POLICY::Head_type _f`  
*Pointer to front of the list.*

### 16.41.1 Detailed Description

```
template<typename POLICY>
class cxx::Bits::Basic_list< POLICY >
```

Internal: Common functions for all head-based list implementations.

Definition at line 39 of file [list\\_basics.h](#).

### 16.41.2 Member Function Documentation

#### 16.41.2.1 `clear()`

```
template<typename POLICY>
void cxx::Bits::Basic_list< POLICY >::clear () [inline]
```

Remove all elements from the list.

After the operation the state of the elements is undefined.

Definition at line 135 of file [list\\_basics.h](#).

References [\\_f](#).

### 16.41.2.2 iter()

```
template<typename POLICY>
Const_iterator cxx::Bits::Basic_list< POLICY >::iter (
    Const_value_type c) [inline], [static]
```

Return a const iterator that begins at the given element.

#### Parameters

---



<code>c</code>	Element where the iterator should start.
----------------	------------------------------------------

#### Precondition

The element `c` must already be in a list.

Definition at line 148 of file [list\\_basics.h](#).

The documentation for this class was generated from the following file:

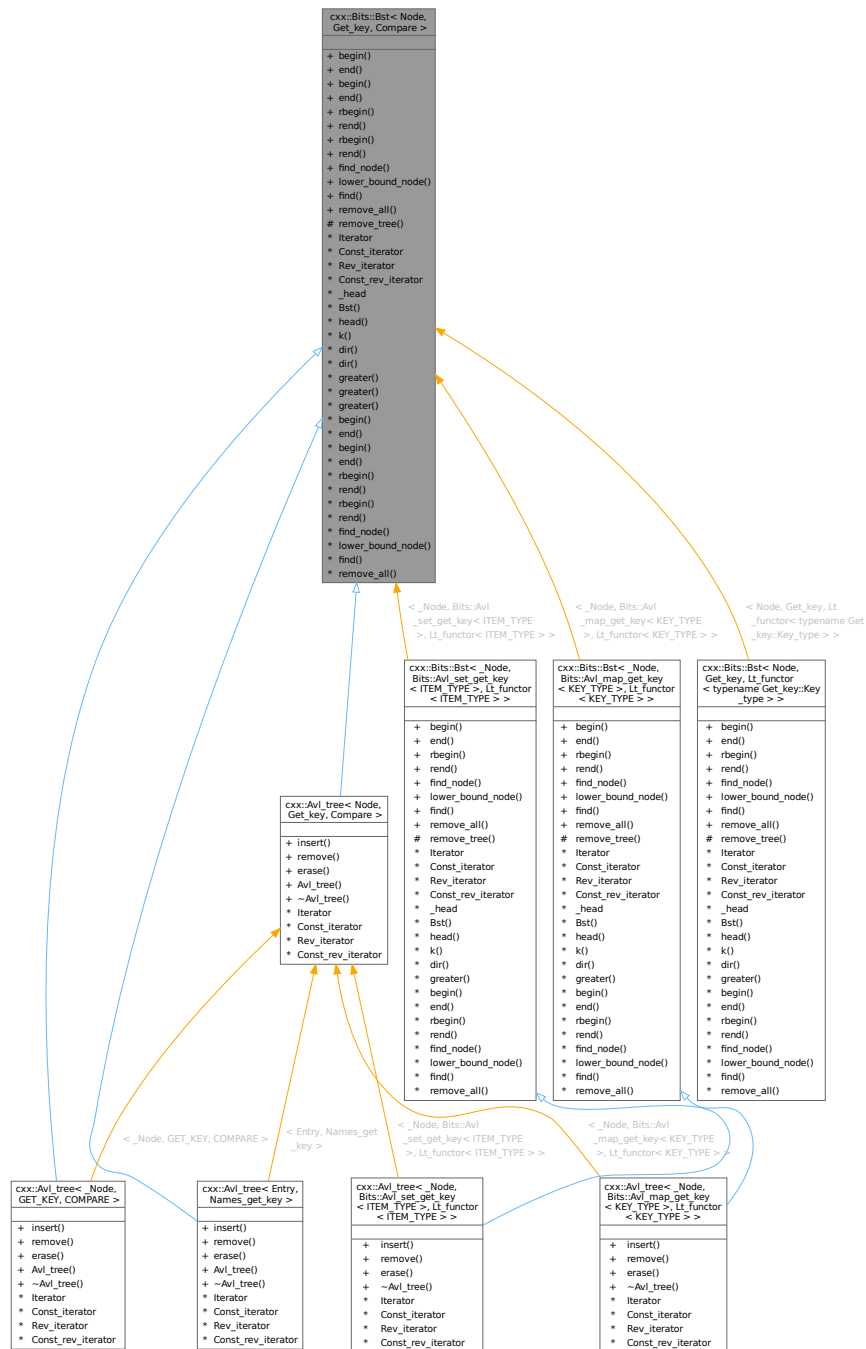
- `I4/cxx/bits/list_basics.h`

## 16.42 `cxx::Bits::Bst< Node, Get_key, Compare >` Class Template Reference

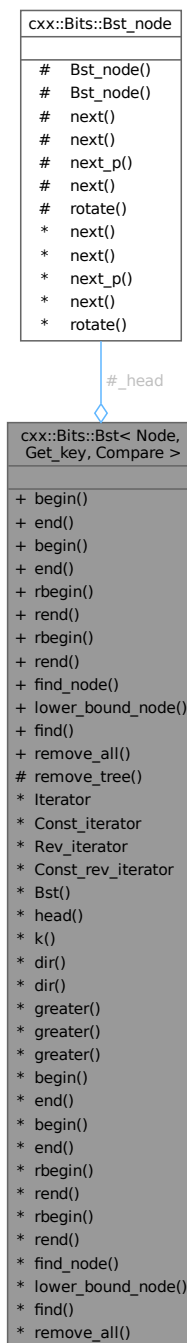
Basic binary search tree (BST).

```
#include <bst.h>
```

Inheritance diagram for `cxx::Bits::Bst< Node, Get_key, Compare >`:



Collaboration diagram for cxx::Bits::Bst< Node, Get\_key, Compare >:



## Public Types

- typedef Get\_key::Key\_type **Key\_type**  
*The type of key values used to generate the total order of the elements.*
- typedef Type\_traits< Key\_type >::Param\_type **Key\_param\_type**  
*The type for key parameters.*
- typedef Fwd **Fwd\_iter\_ops**

*Helper for building forward iterators for different wrapper classes.*

- typedef Rev **Rev\_iter\_ops**

*Helper for building reverse iterators for different wrapper classes.*

## Iterators

- typedef \_\_Bst\_iter< Node, Node, Fwd > **Iterator**  
*Forward iterator.*
- typedef \_\_Bst\_iter< Node, Node const, Fwd > **Const\_iterator**  
*Constant forward iterator.*
- typedef \_\_Bst\_iter< Node, Node, Rev > **Rev\_iterator**  
*Backward iterator.*
- typedef \_\_Bst\_iter< Node, Node const, Rev > **Const\_rev\_iterator**  
*Constant backward.*

## Public Member Functions

### Get default iterators for the ordered tree.

- [Const\\_iterator begin](#) () const  
*Get the constant forward iterator for the first element in the set.*
- [Const\\_iterator end](#) () const  
*Get the end marker for the constant forward iterator.*
- [Iterator begin](#) ()  
*Get the mutable forward iterator for the first element of the set.*
- [Iterator end](#) ()  
*Get the end marker for the mutable forward iterator.*
- [Const\\_rev\\_iterator rbegin](#) () const  
*Get the constant backward iterator for the last element in the set.*
- [Const\\_rev\\_iterator rend](#) () const  
*Get the end marker for the constant backward iterator.*
- [Rev\\_iterator rbegin](#) ()  
*Get the mutable backward iterator for the last element of the set.*
- [Rev\\_iterator rend](#) ()  
*Get the end marker for the mutable backward iterator.*

### Lookup functions.

- Node \* [find\\_node](#) (Key\_param\_type key) const  
*find the node with the given key.*
- Node \* [lower\\_bound\\_node](#) (Key\_param\_type key) const  
*Find the first node with a key not less than the given key.*
- [Const\\_iterator find](#) (Key\_param\_type key) const  
*find the node with the given key.*
- template<typename FUNC>  
void [remove\\_all](#) (FUNC &&callback)  
*Clear the tree.*

## Static Protected Member Functions

- template<typename FUNC>  
static void [remove\\_tree](#) (Bst\_node \*head, FUNC &&callback)  
*Remove all elements in the subtree of head.*

**Interior access for descendants.**

As this class is an intended base class we provide protected access to our interior, use 'using' to make this private in concrete implementations.

- [Bst\\_node](#) \* **\_head**  
*The head pointer of the tree.*
- **Bst** ()  
*Create an empty tree.*
- Node \* **head** () const  
*Access the head node as object of type Node.*
- static [Key\\_type](#) **k** ([Bst\\_node](#) const \*n)  
*Get the key value of n.*
- static [Dir](#) **dir** ([Key\\_param\\_type](#) l, [Key\\_param\\_type](#) r)  
*Get the direction to go from l to search for r.*
- static [Dir](#) **dir** ([Key\\_param\\_type](#) l, [Bst\\_node](#) const \*r)  
*Get the direction to go from l to search for r.*
- static bool **greater** ([Key\\_param\\_type](#) l, [Key\\_param\\_type](#) r)  
*Is l greater than r.*
- static bool **greater** ([Key\\_param\\_type](#) l, [Bst\\_node](#) const \*r)  
*Is l greater than r.*
- static bool **greater** ([Bst\\_node](#) const \*l, [Bst\\_node](#) const \*r)  
*Is l greater than r.*

**16.42.1 Detailed Description**

```
template<typename Node, typename Get_key, typename Compare>
class cxx::Bits::Bst< Node, Get_key, Compare >
```

Basic binary search tree (BST).

This class is intended as a base class for concrete binary search trees, such as an AVL tree. This class already provides the basic lookup methods and iterator definitions for a BST.

Definition at line 31 of file [bst.h](#).

**16.42.2 Member Function Documentation****16.42.2.1 begin() [1/2]**

```
template<typename Node, typename Get_key, typename Compare>
Iterator cxx::Bits::Bst< Node, Get_key, Compare >::begin () [inline]
```

Get the mutable forward iterator for the first element of the set.

**Returns**

The mutable forward iterator for the first element of the set.

Definition at line 183 of file [bst.h](#).

References [head\(\)](#).

Here is the call graph for this function:

**16.42.2.2 begin() [2/2]**

```
template<typename Node, typename Get_key, typename Compare>
Const_iterator cxx::Bits::Bst< Node, Get_key, Compare >::begin () const [inline]
```

Get the constant forward iterator for the first element in the set.

**Returns**

Constant forward iterator for the first element in the set.

Definition at line 172 of file [bst.h](#).

References [head\(\)](#).

Here is the call graph for this function:

**16.42.2.3 dir() [1/2]**

```
template<typename Node, typename Get_key, typename Compare>
Dir cxx::Bits::Bst< Node, Get_key, Compare >::dir (
    Key_param_type l,
    Bst_node const * r) [inline], [static], [protected]
```

Get the direction to go from `l` to search for `r`.

**Parameters**

<i>l</i>	is the key to look for.
<i>r</i>	is the node at the current position.

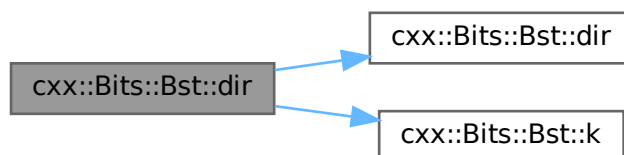
**Return values**

<a href="#"><i>Direction::L</i></a>	For left.
<a href="#"><i>Direction::R</i></a>	For right.
<a href="#"><i>Direction::N</i></a>	If <i>l</i> is equal to <i>r</i> .

Definition at line 128 of file [bst.h](#).

References [dir\(\)](#), and [k\(\)](#).

Here is the call graph for this function:

**16.42.2.4 dir() [2/2]**

```

template<typename Node, typename Get_key, typename Compare>
Dir cxx::Bits::Bst< Node, Get_key, Compare >::dir (
    Key_param_type l,
    Key_param_type r) [inline], [static], [protected]
  
```

Get the direction to go from *l* to search for *r*.

**Parameters**

<i>l</i>	is the key to look for.
<i>r</i>	is the key at the current position.

**Return values**

<a href="#"><i>Direction::L</i></a>	for left
<a href="#"><i>Direction::R</i></a>	for right

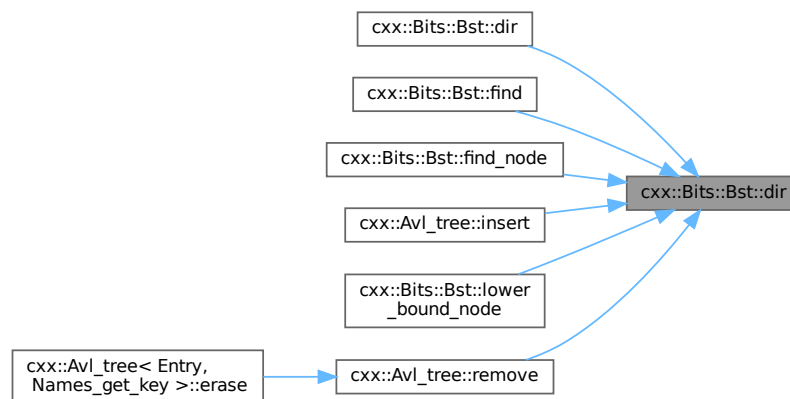
<i>Direction::N</i>	if <code>l</code> is equal to <code>r</code> .
---------------------	------------------------------------------------

Definition at line 111 of file [bst.h](#).

References [cxx::Bits::Direction::L](#), and [cxx::Bits::Direction::N](#).

Referenced by [dir\(\)](#), [find\(\)](#), [find\\_node\(\)](#), [cxx::Avl\\_tree< Node, Get\\_key, Compare >::insert\(\)](#), [lower\\_bound\\_node\(\)](#), and [cxx::Avl\\_tree< Node, Get\\_key, Compare >::remove\(\)](#).

Here is the caller graph for this function:



#### 16.42.2.5 `end()` [1/2]

```
template<typename Node, typename Get_key, typename Compare>
Iterator cxx::Bits::Bst< Node, Get_key, Compare >::end () [inline]
```

Get the end marker for the mutable forward iterator.

##### Returns

The end marker for mutable forward iterator.

Definition at line 188 of file [bst.h](#).

#### 16.42.2.6 `end()` [2/2]

```
template<typename Node, typename Get_key, typename Compare>
Const_iterator cxx::Bits::Bst< Node, Get_key, Compare >::end () const [inline]
```

Get the end marker for the constant forward iterator.

##### Returns

The end marker for the constant forward iterator.

Definition at line 177 of file [bst.h](#).



## 16.42.2.7 find()

```
template<typename Node, typename Get_key, class Compare>
Bst< Node, Get_key, Compare >::Const_iterator cxx::Bits::Bst< Node, Get_key, Compare >::find
(
    Key_param_type key) const [inline]
```

find the node with the given *key*.

## Parameters

<i>key</i>	The key value of the element to search.
------------	-----------------------------------------

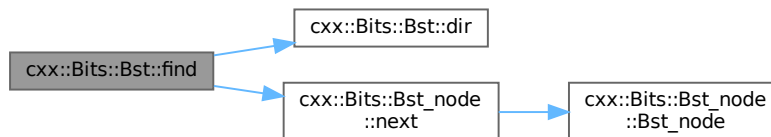
## Returns

A valid iterator for the node with the given *key*, or an invalid iterator if *key* was not found.

Definition at line 305 of file [bst.h](#).

References [\\_head](#), [dir\(\)](#), [cxx::Bits::Direction::L](#), [cxx::Bits::Direction::N](#), and [cxx::Bits::Bst\\_node::next\(\)](#).

Here is the call graph for this function:



## 16.42.2.8 find\_node()

```
template<typename Node, typename Get_key, class Compare>
Node * cxx::Bits::Bst< Node, Get_key, Compare >::find_node (
    Key_param_type key) const [inline]
```

find the node with the given *key*.

## Parameters

<i>key</i>	The key value of the element to search.
------------	-----------------------------------------

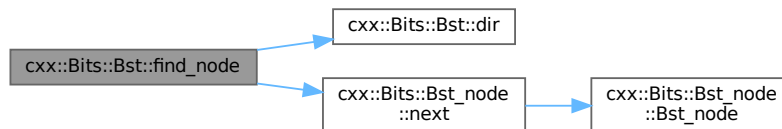
**Returns**

A pointer to the node with the given *key*, or `NULL` if *key* was not found.

Definition at line 269 of file [bst.h](#).

References [\\_head](#), [dir\(\)](#), [cxx::Bits::Direction::N](#), and [cxx::Bits::Bst\\_node::next\(\)](#).

Here is the call graph for this function:

**16.42.2.9 lower\_bound\_node()**

```

template<typename Node, typename Get_key, class Compare>
Node * cxx::Bits::Bst< Node, Get_key, Compare >::lower_bound_node (
    Key_param_type key) const [inline]
  
```

Find the first node with a key not less than the given key.

**Parameters**

<i>key</i>	The key used for the search.
------------	------------------------------

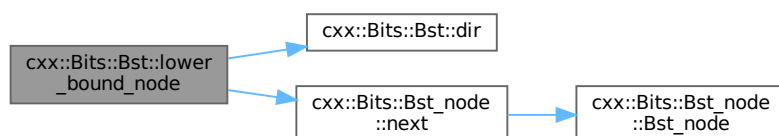
**Returns**

A pointer to the found node, or `NULL` if no node was found.

Definition at line 285 of file [bst.h](#).

References [\\_head](#), [dir\(\)](#), [cxx::Bits::Direction::L](#), [cxx::Bits::Direction::N](#), and [cxx::Bits::Bst\\_node::next\(\)](#).

Here is the call graph for this function:



**16.42.2.10 rbegin()** [1/2]

```
template<typename Node, typename Get_key, typename Compare>
Rev_iterator cxx::Bits::Bst< Node, Get_key, Compare >::rbegin () [inline]
```

Get the mutable backward iterator for the last element of the set.

**Returns**

The mutable backward iterator for the last element of the set.

Definition at line 205 of file [bst.h](#).

References [head\(\)](#).

Here is the call graph for this function:

**16.42.2.11 rbegin()** [2/2]

```
template<typename Node, typename Get_key, typename Compare>
Const_rev_iterator cxx::Bits::Bst< Node, Get_key, Compare >::rbegin () const [inline]
```

Get the constant backward iterator for the last element in the set.

**Returns**

The constant backward iterator for the last element in the set.

Definition at line 194 of file [bst.h](#).

References [head\(\)](#).

Here is the call graph for this function:



#### 16.42.2.12 remove\_all()

```
template<typename Node, typename Get_key, typename Compare>
template<typename FUNC>
void cxx::Bits::Bst< Node, Get_key, Compare >::remove_all (
    FUNC && callback) [inline]
```

Clear the tree.

#### Parameters

---

<i>callback</i>	Optional function to be called on each removed element.
-----------------	---------------------------------------------------------

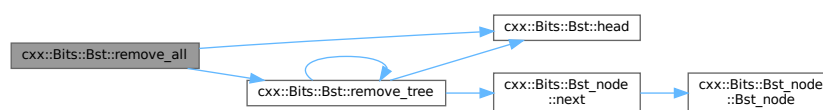
The callback may delete the elements. The function guarantees that the elements are no longer used after the callback has been called.

Definition at line 251 of file [bst.h](#).

References [\\_head](#), [head\(\)](#), and [remove\\_tree\(\)](#).

Referenced by [cxx::Avl\\_tree< Entry, Names\\_get\\_key >::~~Avl\\_tree\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.42.2.13 remove\_tree()

```

template<typename Node, typename Get_key, typename Compare>
template<typename FUNC>
void cxx::Bits::Bst< Node, Get_key, Compare >::remove_tree (
    Bst_node * head,
    FUNC && callback) [inline], [static], [protected]

```

Remove all elements in the subtree of head.

#### Parameters

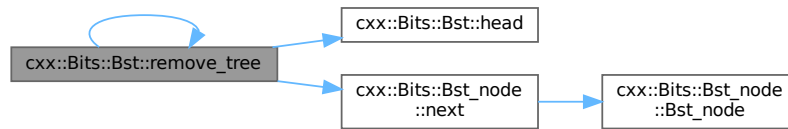
<i>head</i>	Head of the the subtree to remove
<i>callback</i>	Optional function called on each removed element.

Definition at line 151 of file [bst.h](#).

References [head\(\)](#), [cxx::Bits::Direction::L](#), [cxx::Bits::Bst\\_node::next\(\)](#), [cxx::Bits::Direction::R](#), and [remove\\_tree\(\)](#).

Referenced by [remove\\_all\(\)](#), and [remove\\_tree\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 16.42.2.14 `rend()` [1/2]

```
template<typename Node, typename Get_key, typename Compare>
Rev_iterator cxx::Bits::Bst< Node, Get_key, Compare >::rend () [inline]
```

Get the end marker for the mutable backward iterator.

##### Returns

The end marker for mutable backward iterator.

Definition at line 210 of file [bst.h](#).

#### 16.42.2.15 `rend()` [2/2]

```
template<typename Node, typename Get_key, typename Compare>
Const_rev_iterator cxx::Bits::Bst< Node, Get_key, Compare >::rend () const [inline]
```

Get the end marker for the constant backward iterator.

##### Returns

The end marker for the constant backward iterator.

Definition at line 199 of file [bst.h](#).

The documentation for this class was generated from the following file:

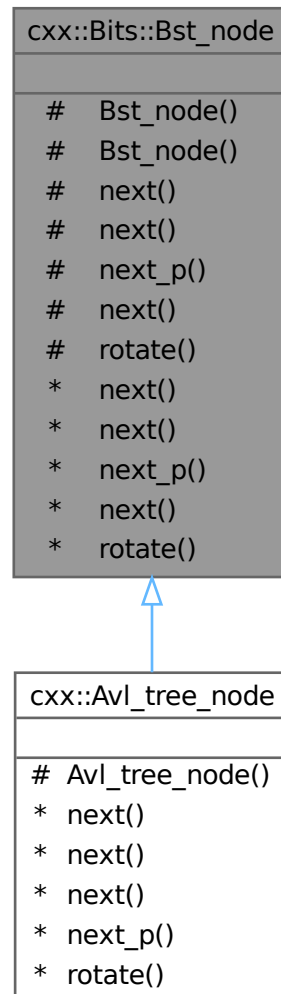
- [I4/cxx/bits/bst.h](#)

## 16.43 cxx::Bits::Bst\_node Class Reference

Basic type of a node in a binary search tree (BST).

```
#include <bst_base.h>
```

Inheritance diagram for cxx::Bits::Bst\_node:



Collaboration diagram for `cxx::Bits::Bst_node`:

cxx::Bits::Bst_node	
#	Bst_node()
#	Bst_node()
#	next()
#	next()
#	next_p()
#	next()
#	rotate()
*	next()
*	next()
*	next_p()
*	next()
*	rotate()

### Protected Member Functions

- **Bst\_node** ()  
*Create uninitialized node.*
- **Bst\_node** (bool)  
*Create initialized node.*

### Static Protected Member Functions

#### Access to BST linkage.

Provide access to the tree linkage to inherited classes. Inherited nodes, such as AVL nodes should make these methods private via 'using'

- static `Bst_node` \* **next** (`Bst_node` const \*p, `Direction` d)  
*Get next node in direction d.*
- static void **next** (`Bst_node` \*p, `Direction` d, `Bst_node` \*n)  
*Set next node of p in direction d to n.*
- static `Bst_node` \*\* **next\_p** (`Bst_node` \*p, `Direction` d)  
*Get pointer to link in direction d.*
- template<typename Node>  
static Node \* **next** (`Bst_node` const \*p, `Direction` d)  
*Get next node in direction d as type Node.*
- static void **rotate** (`Bst_node` \*\*t, `Direction` idir)  
*Rotate subtree t in the opposite direction of idir.*



### 16.43.1 Detailed Description

Basic type of a node in a binary search tree (BST).

Definition at line 70 of file [bst\\_base.h](#).

The documentation for this class was generated from the following file:

- [l4/cxx/bits/bst\\_base.h](#)

## 16.44 cxx::Bits::Direction Struct Reference

The direction to go in a binary search tree.

```
#include <bst_base.h>
```

Collaboration diagram for cxx::Bits::Direction:

cxx::Bits::Direction	
+	Direction()
+	Direction()
+	Direction()
+	operator!()
+	operator==(())
+	operator!=(())
+	operator==(())
+	operator!=(())
*	operator==(())
*	operator!=(())
*	operator==(())
*	operator!=(())

### Public Types

- enum [Direction\\_e](#) { [L](#) = 0 , [R](#) = 1 , [N](#) = 2 }

*The literal direction values.*

## Public Member Functions

- **Direction** ()=default  
*Uninitialized direction.*
- **Direction** ([Direction\\_e](#) d)  
*Convert a literal direction ([L](#), [R](#), [N](#)) to an object.*
- **Direction** (bool b)  
*Convert a boolean to a direction (false == [L](#), true == [R](#)).*
- **Direction operator!** () const  
*Negate the direction.*

## Comparison operators (equality and inequality)

- bool **operator==** ([Direction\\_e](#) o) const  
*Compare for equality.*
- bool **operator!=** ([Direction\\_e](#) o) const  
*Compare for inequality.*
- bool **operator==** ([Direction](#) o) const  
*Compare for equality.*
- bool **operator!=** ([Direction](#) o) const  
*Compare for inequality.*

### 16.44.1 Detailed Description

The direction to go in a binary search tree.

Definition at line 28 of file [bst\\_base.h](#).

### 16.44.2 Member Enumeration Documentation

#### 16.44.2.1 Direction\_e

```
enum cxx::Bits::Direction::Direction\_e
```

The literal direction values.

#### Enumerator

L	Go to the left child.
R	Go to the right child.
N	Stop.

Definition at line 31 of file [bst\\_base.h](#).

### 16.44.3 Member Function Documentation

#### 16.44.3.1 operator"!")()

```
Direction cxx::Bits::Direction::operator!  () const [inline]
```

Negate the direction.

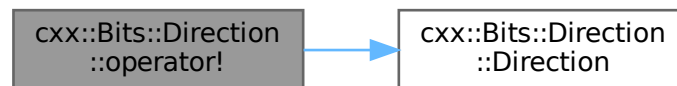
##### Note

This is only defined for a current value of [L](#) or [R](#)

Definition at line [52](#) of file [bst\\_base.h](#).

References [Direction\(\)](#).

Here is the call graph for this function:



The documentation for this struct was generated from the following file:

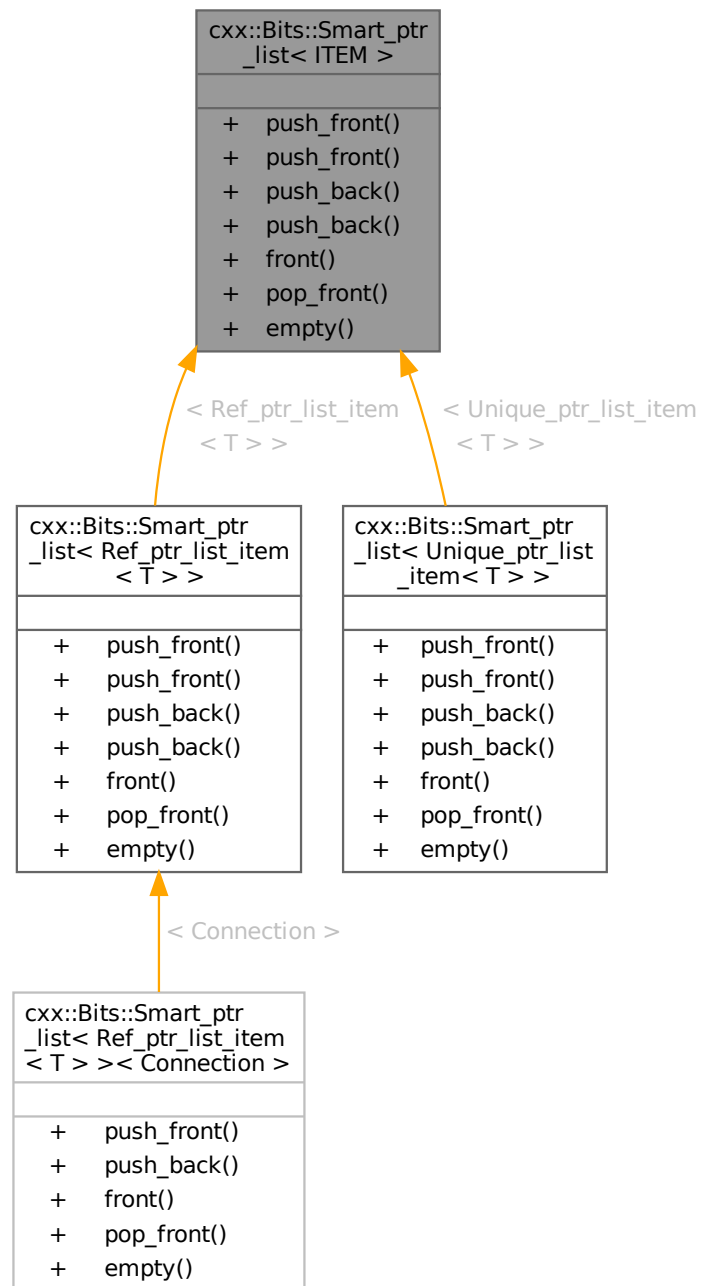
- [l4/cxx/bits/bst\\_base.h](#)

## 16.45 cxx::Bits::Smart\_ptr\_list< ITEM > Class Template Reference

[List](#) of smart-pointer-managed objects.

```
#include <smart_ptr_list.h>
```

Inheritance diagram for `cxx::Bits::Smart_ptr_list< ITEM >`:



Collaboration diagram for cxx::Bits::Smart\_ptr\_list< ITEM >:

cxx::Bits::Smart_ptr_list< ITEM >	
+	push_front()
+	push_front()
+	push_back()
+	push_back()
+	front()
+	pop_front()
+	empty()

### Public Member Functions

- void **push\_front** (Next\_type &&e)  
*Add an element to the front of the list.*
- void **push\_front** (Next\_type const &e)  
*Add an element to the front of the list.*
- void **push\_back** (Next\_type &&e)  
*Add an element at the end of the list.*
- void **push\_back** (Next\_type const &e)  
*Add an element at the end of the list.*
- Value\_type \* **front** () const  
*Return a pointer to the first element in the list.*
- Next\_type **pop\_front** ()  
*Remove the element in front of the list and return it.*
- bool **empty** () const  
*Check if the list is empty.*

### 16.45.1 Detailed Description

```
template<typename ITEM>
class cxx::Bits::Smart_ptr_list< ITEM >
```

List of smart-pointer-managed objects.

### Template Parameters

<i>ITEM</i>	Type of the list items.
-------------	-------------------------

The list is implemented as a single-linked list connected via smart pointers, so that they are automatically cleaned up when they are removed from the list.

Definition at line 46 of file [smart\\_ptr\\_list.h](#).

## 16.45.2 Member Function Documentation

### 16.45.2.1 pop\_front()

```
template<typename ITEM>
Next_type cxx::Bits::Smart_ptr_list< ITEM >::pop_front () [inline]
```

Remove the element in front of the list and return it.

#### Returns

The element that was previously in front of the list as a managed pointer or a nullptr-equivalent when the list was already empty.

Definition at line 149 of file [smart\\_ptr\\_list.h](#).

The documentation for this class was generated from the following file:

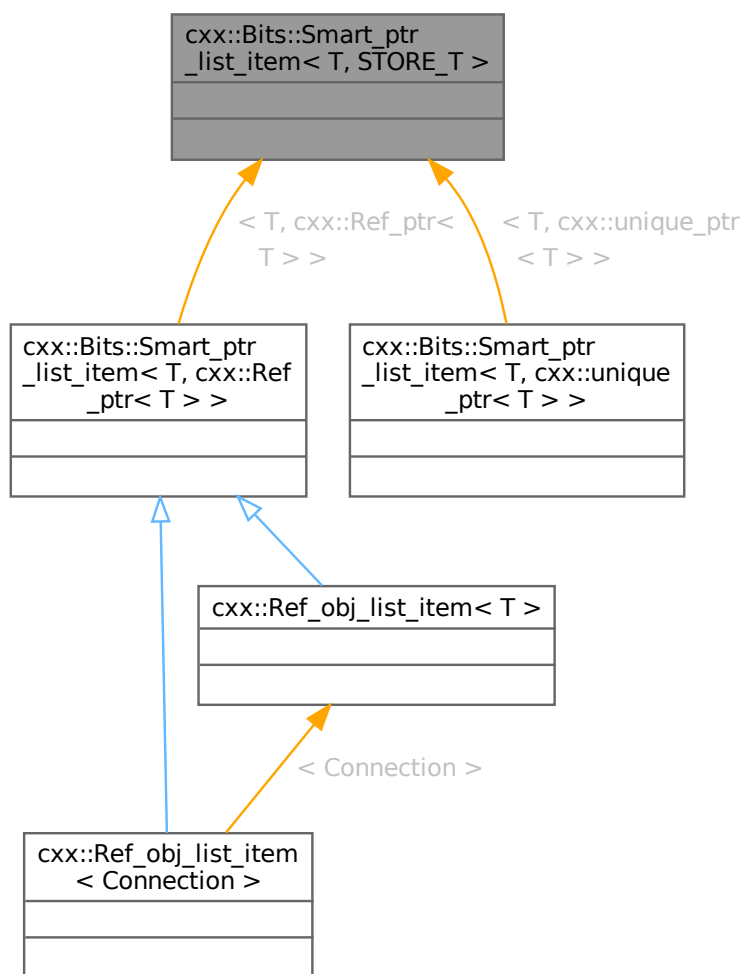
- [l4/cxx/bits/smart\\_ptr\\_list.h](#)

## 16.46 cxx::Bits::Smart\_ptr\_list\_item< T, STORE\_T > Class Template Reference

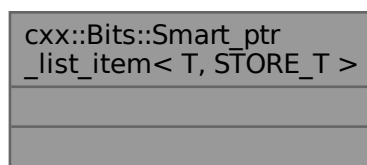
[List](#) item for an arbitrary item in a [Smart\\_ptr\\_list](#).

```
#include <smart_ptr_list.h>
```

Inheritance diagram for cxx::Bits::Smart\_ptr\_list\_item< T, STORE\_T >:



Collaboration diagram for cxx::Bits::Smart\_ptr\_list\_item< T, STORE\_T >:



### 16.46.1 Detailed Description

```
template<typename T, typename STORE_T>
class cxx::Bits::Smart_ptr_list_item< T, STORE_T >
```

List item for an arbitrary item in a [Smart\\_ptr\\_list](#).

#### Template Parameters

<i>T</i>	Type of object to be stored in the list.
<i>STORE_T</i>	Storage type for pointer to next item. The class must implement a <code>get()</code> function that returns a pointer to the stored object and destroy the stored object when the item goes out of scope.

Definition at line 27 of file [smart\\_ptr\\_list.h](#).

The documentation for this class was generated from the following file:

- [I4/cxx/bits/smart\\_ptr\\_list.h](#)

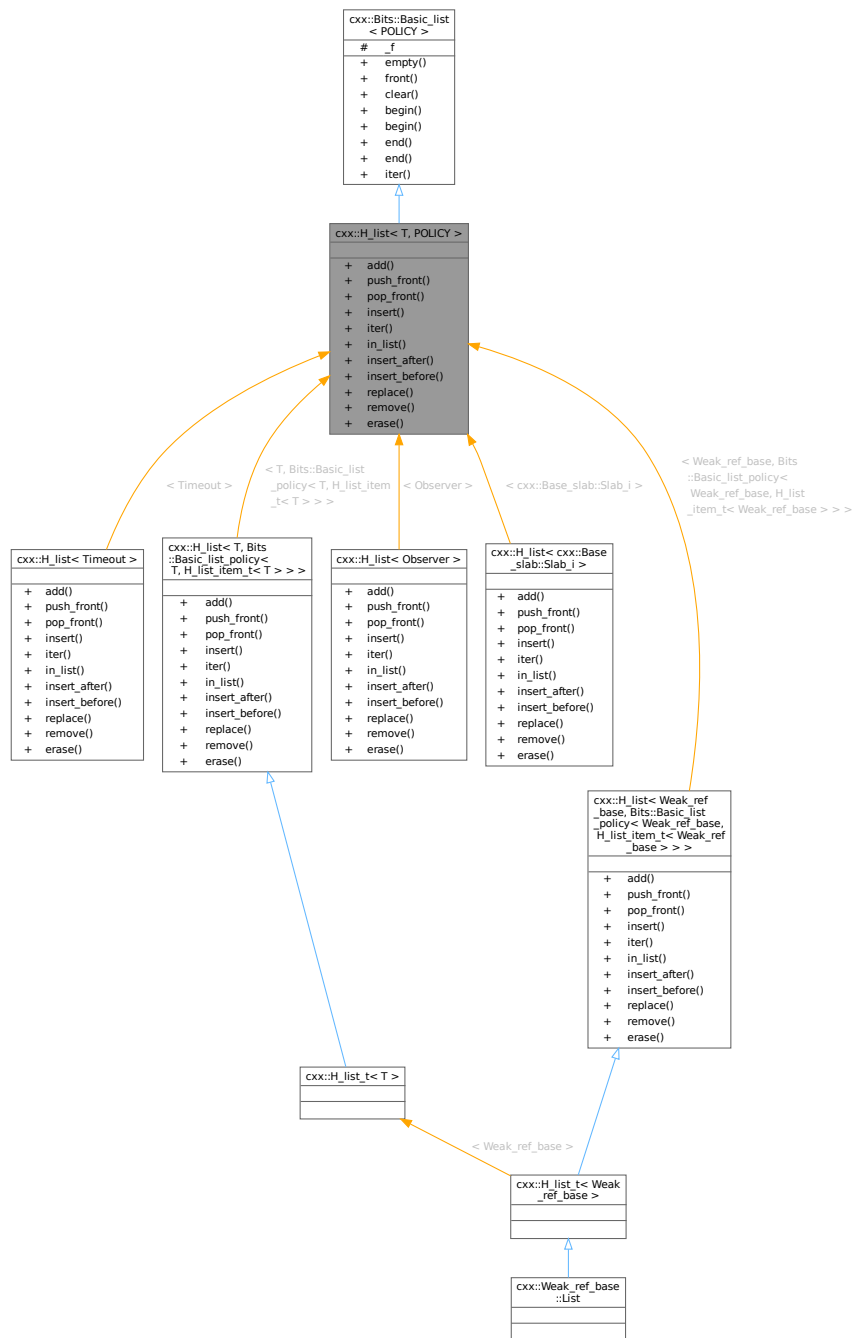
## 16.47 `cxx::H_list< T, POLICY >` Class Template Reference

General double-linked list of unspecified [cxx::H\\_list\\_item](#) elements.

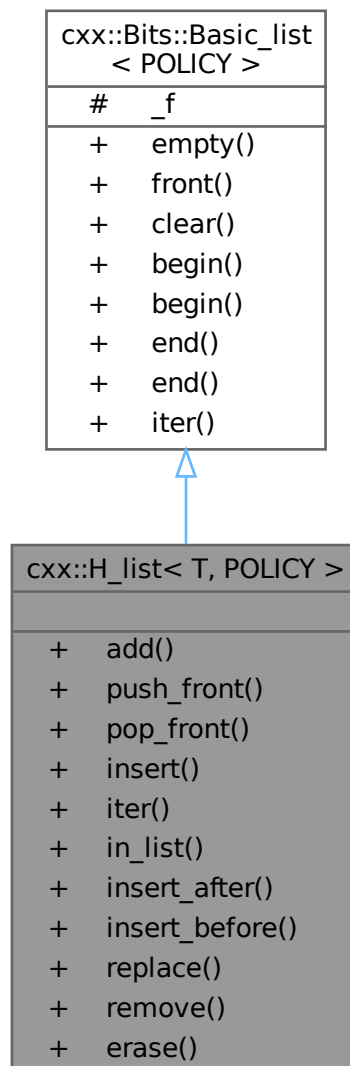
```
#include <hlist>
```



Inheritance diagram for cxx::H\_list< T, POLICY >:



Collaboration diagram for `cxx::H_list< T, POLICY >`:



## Public Member Functions

- void **add** (T \*e)  
*Add element to the front of the list.*
- void **push\_front** (T \*e)  
*Add element to the front of the list.*
- T \* **pop\_front** ()  
*Remove and return the head element of the list.*
- Iterator **insert** (T \*e, Iterator const &pred)  
*Insert an element at the iterator position.*

**Public Member Functions inherited from `cxx::Bits::Basic_list< POLICY >`**

- `bool empty () const`  
*Check if the list is empty.*
- `Value_type front () const`  
*Return the first element in the list.*
- `void clear ()`  
*Remove all elements from the list.*
- `Iterator begin ()`  
*Return an iterator to the beginning of the list.*
- `Const_iterator begin () const`  
*Return a const iterator to the beginning of the list.*
- `Const_iterator end () const`  
*Return a const iterator to the end of the list.*
- `Iterator end ()`  
*Return an iterator to the end of the list.*

**Static Public Member Functions**

- `static Iterator iter (T *c)`  
*Return an iterator for an arbitrary list element.*
- `static bool in_list (T const *e)`  
*Check if the given element is currently part of a list.*
- `static Iterator insert_after (T *e, Iterator const &pred)`  
*Insert an element after the iterator position.*
- `static void insert_before (T *e, Iterator const &succ)`  
*Insert an element before the iterator position.*
- `static void replace (T *p, T *e)`  
*Replace an element in a list with a new element.*
- `static void remove (T *e)`  
*Remove the given element from its list.*
- `static Iterator erase (Iterator const &e)`  
*Remove the element at the given iterator position.*

**Static Public Member Functions inherited from `cxx::Bits::Basic_list< POLICY >`**

- `static Const_iterator iter (Const_value_type c)`  
*Return a const iterator that begins at the given element.*

**Additional Inherited Members****Protected Attributes inherited from `cxx::Bits::Basic_list< POLICY >`**

- `POLICY::Head_type _f`  
*Pointer to front of the list.*

### 16.47.1 Detailed Description

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
class cxx::H_list< T, POLICY >
```

General double-linked list of unspecified [cxx::H\\_list\\_item](#) elements.

Most of the time, you want to use [H\\_list\\_t](#).

Definition at line 69 of file [hlist](#).

### 16.47.2 Member Function Documentation

#### 16.47.2.1 erase()

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
Iterator cxx::H_list< T, POLICY >::erase (
    Iterator const & e) [inline], [static]
```

Remove the element at the given iterator position.

##### Parameters

<i>e</i>	Iterator pointing to the element to be removed. Must not point to <a href="#">end()</a> .
----------	-------------------------------------------------------------------------------------------

##### Returns

New iterator pointing to the element after the removed one.

##### Note

The hlist implementation guarantees that the original iterator is still valid after the element has been removed. In fact, the iterator returned is the same as the one supplied in the *e* parameter.

Definition at line 236 of file [hlist](#).

#### 16.47.2.2 insert()

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
Iterator cxx::H_list< T, POLICY >::insert (
    T * e,
    Iterator const & pred) [inline]
```

Insert an element at the iterator position.

##### Parameters

<i>e</i>	New Element to be inserted
<i>pred</i>	Iterator pointing to the element after which the element will be inserted. If <code>end()</code> is given, the element will be inserted at the beginning of the queue.

**Returns**

Iterator pointing to the newly inserted element.

Definition at line 133 of file [hlist](#).

**16.47.2.3 `insert_after()`**

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
Iterator cxx::H_list< T, POLICY >::insert_after (
    T * e,
    Iterator const & pred) [inline], [static]
```

Insert an element after the iterator position.

**Parameters**

<i>e</i>	New element to be inserted.
<i>pred</i>	Iterator pointing to the element after which the element will be inserted. Must not be <code>end()</code> .

**Returns**

Iterator pointing to the newly inserted element.

**Precondition**

The list must not be empty.

Definition at line 160 of file [hlist](#).

**16.47.2.4 `insert_before()`**

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
void cxx::H_list< T, POLICY >::insert_before (
    T * e,
    Iterator const & succ) [inline], [static]
```

Insert an element before the iterator position.

**Parameters**

<i>e</i>	New element to be inserted.
<i>succ</i>	Iterator pointing to the element before which the element will be inserted. Must not be <a href="#">end()</a> .

Definition at line 180 of file [hlist](#).

#### 16.47.2.5 `iter()`

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
Iterator cxx::H\_list< T, POLICY >::iter (
    T * c)    [inline], [static]
```

Return an iterator for an arbitrary list element.

#### Parameters

<i>c</i>	<a href="#">List</a> element to start the iteration.
----------	------------------------------------------------------

#### Returns

A mutable forward iterator.

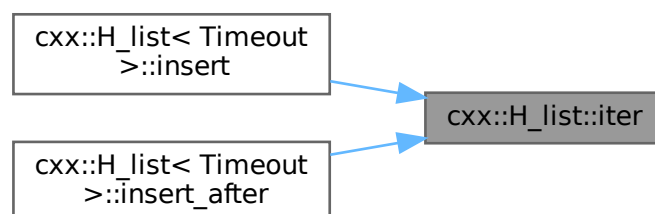
#### Precondition

The element must be in a list.

Definition at line 93 of file [hlist](#).

Referenced by [cxx::H\\_list< Timeout >::insert\(\)](#), and [cxx::H\\_list< Timeout >::insert\\_after\(\)](#).

Here is the caller graph for this function:



**16.47.2.6 `pop_front()`**

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
T * cxx::H_list< T, POLICY >::pop_front () [inline]
```

Remove and return the head element of the list.

**Precondition**

The list must not be empty or the behaviour will be undefined.

Definition at line 116 of file `hlist`.

**16.47.2.7 `remove()`**

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
void cxx::H_list< T, POLICY >::remove (
    T * e) [inline], [static]
```

Remove the given element from its list.

**Parameters**

<code>e</code>	Element to be removed. Must be in a list.
----------------	-------------------------------------------

Definition at line 220 of file `hlist`.

Referenced by `cxx::H_list< Timeout >::pop_front()`.

Here is the caller graph for this function:

**16.47.2.8 `replace()`**

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
void cxx::H_list< T, POLICY >::replace (
    T * p,
    T * e) [inline], [static]
```

Replace an element in a list with a new element.

**Parameters**

<i>p</i>	Element in list to be replaced.
<i>e</i>	Replacement element, must not yet be in a list.

**Precondition**

*p* and *e* must not be NULL.

After the operation the *p* element is no longer in the list and may be reused.

Definition at line 204 of file [hlist](#).

The documentation for this class was generated from the following file:

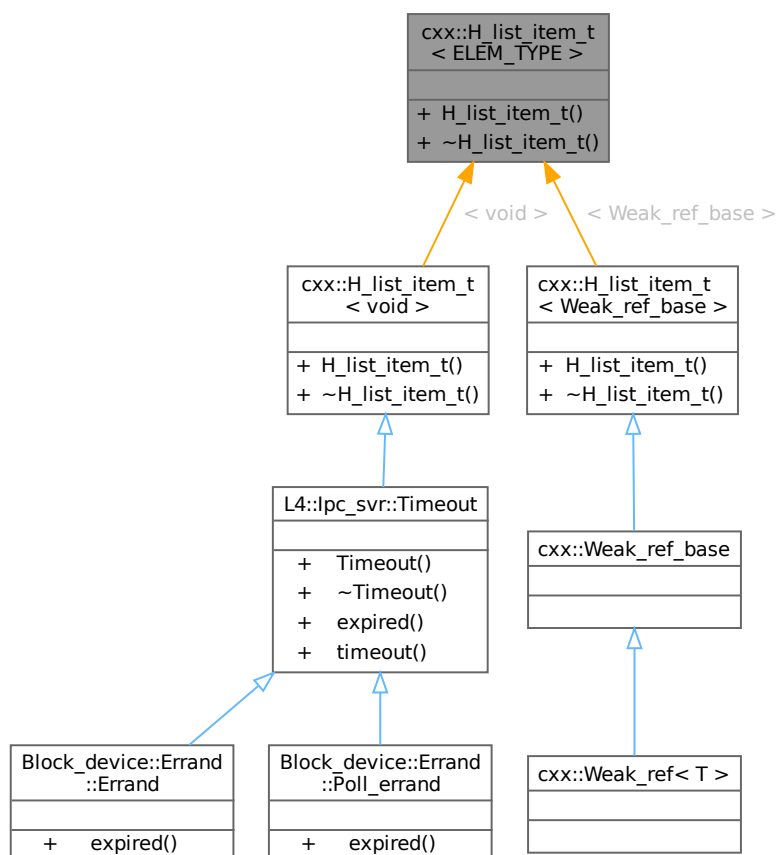
- l4/cxx/hlist

## 16.48 cxx::H\_list\_item\_t< ELEM\_TYPE > Class Template Reference

Basic element type for a double-linked [H\\_list](#).

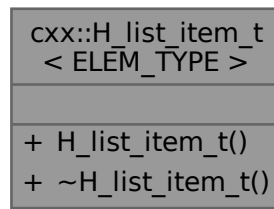
```
#include <hlist>
```

Inheritance diagram for cxx::H\_list\_item\_t< ELEM\_TYPE >:





Collaboration diagram for `cxx::H_list_item_t< ELEM_TYPE >`:



### Public Member Functions

- [H\\_list\\_item\\_t\(\)](#)  
*Constructor.*
- [~H\\_list\\_item\\_t\(\)](#) noexcept  
*Destructor.*

### 16.48.1 Detailed Description

```
template<typename ELEM_TYPE>
class cxx::H_list_item_t< ELEM_TYPE >
```

Basic element type for a double-linked [H\\_list](#).

### Template Parameters

<code>ELEM_TYPE</code>	Base class of the list element.
------------------------	---------------------------------

Definition at line 22 of file [hlist](#).

### 16.48.2 Constructor & Destructor Documentation

#### 16.48.2.1 H\_list\_item\_t()

```
template<typename ELEM_TYPE>
cxx::H_list_item_t< ELEM_TYPE >::H_list_item_t () [inline]
```

Constructor.

Creates an element that is not in any list.

Definition at line 30 of file [hlist](#).

### 16.48.2.2 ~H\_list\_item\_t()

```
template<typename ELEM_TYPE>
cxx::H_list_item_t< ELEM_TYPE >::~~H_list_item_t () [inline], [noexcept]
```

Destructor.

Automatically removes the element from any list it still might be enchainned in.

Definition at line 37 of file [hlist](#).

The documentation for this class was generated from the following file:

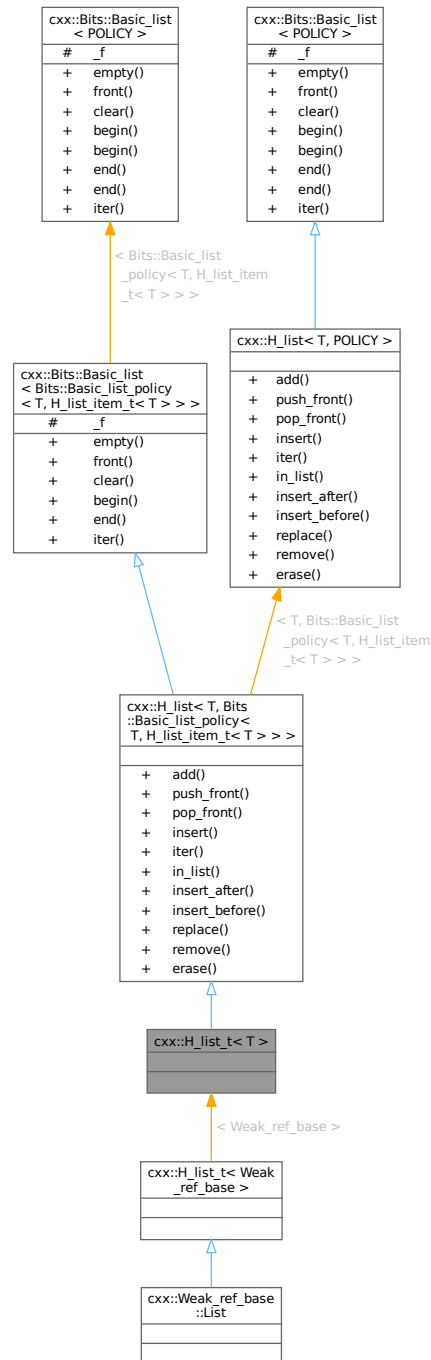
- I4/cxx/hlist

## 16.49 cxx::H\_list\_t< T > Struct Template Reference

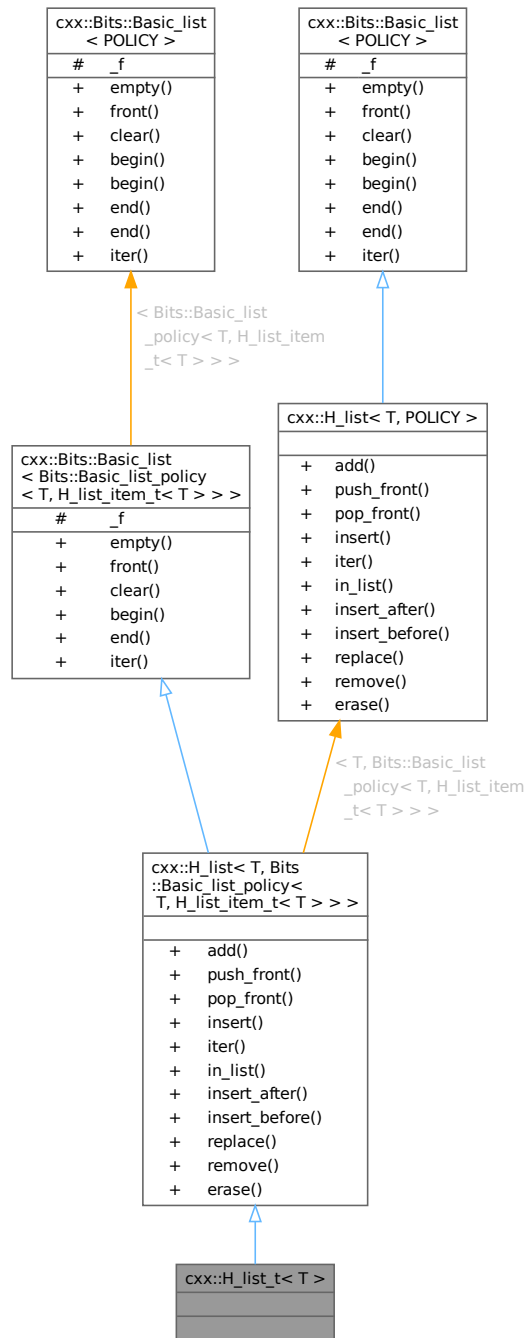
Double-linked list of typed [H\\_list\\_item\\_t](#) elements.

```
#include <hlist>
```

Inheritance diagram for cxx::H\_list\_t< T >:



Collaboration diagram for `cxx::H_list_t< T >`:



### Additional Inherited Members

### Public Member Functions inherited from

`cxx::H_list< T, Bits::Basic_list_policy< T, H_list_item_t< T > > >`

- void **add** (T \*e)

- *Add element to the front of the list.*
- void **push\_front** (T \*e)  
*Add element to the front of the list.*
- T \* **pop\_front** ()  
*Remove and return the head element of the list.*
- Iterator **insert** (T \*e, Iterator const &pred)  
*Insert an element at the iterator position.*

### Public Member Functions inherited from

`cxx::Bits::Basic_list< Bits::Basic_list_policy< T, H_list_item_t< T > > >`

- bool **empty** () const  
*Check if the list is empty.*
- Value\_type **front** () const  
*Return the first element in the list.*
- void **clear** ()  
*Remove all elements from the list.*
- Iterator **begin** ()  
*Return an iterator to the beginning of the list.*
- Const\_iterator **end** () const  
*Return a const iterator to the end of the list.*

### Static Public Member Functions inherited from

`cxx::H_list< T, Bits::Basic_list_policy< T, H_list_item_t< T > > >`

- static Iterator **iter** (T \*c)  
*Return an iterator for an arbitrary list element.*
- static bool **in\_list** (T const \*e)  
*Check if the given element is currently part of a list.*
- static Iterator **insert\_after** (T \*e, Iterator const &pred)  
*Insert an element after the iterator position.*
- static void **insert\_before** (T \*e, Iterator const &succ)  
*Insert an element before the iterator position.*
- static void **replace** (T \*p, T \*e)  
*Replace an element in a list with a new element.*
- static void **remove** (T \*e)  
*Remove the given element from its list.*
- static Iterator **erase** (Iterator const &e)  
*Remove the element at the given iterator position.*

### Static Public Member Functions inherited from

`cxx::Bits::Basic_list< Bits::Basic_list_policy< T, H_list_item_t< T > > >`

- static Const\_iterator **iter** (Const\_value\_type c)  
*Return a const iterator that begins at the given element.*

**Protected Attributes inherited from****cxx::Bits::Basic\_list< Bits::Basic\_list\_policy< T, H\_list\_item\_t< T > > >**

- Bits::Basic\_list\_policy< T, H\_list\_item\_t< T > >::Head\_type \_f

*Pointer to front of the list.***16.49.1 Detailed Description**

```
template<typename T>
struct cxx::H_list_t< T >
```

Double-linked list of typed [H\\_list\\_item\\_t](#) elements.

**Note**

H\_lists are not self-cleaning. Elements that are still chained during destruction are not removed and will therefore be in an undefined state after the destruction.

Definition at line [248](#) of file [hlist](#).

The documentation for this struct was generated from the following file:

- I4/cxx/hlist

**16.50 cxx::List< D, Alloc > Class Template Reference**

Doubly linked list, with internal allocation.

```
#include <list>
```

Collaboration diagram for cxx::List< D, Alloc >:

cxx::List< D, Alloc >	
+	push_back()
+	push_front()
+	remove()
+	size()
+	operator[]()
+	operator[]()
+	items()

## Data Structures

- class `Iter`  
*Iterator.*

## Public Member Functions

- void **push\_back** (D const &d) noexcept  
*Add element at the end of the list.*
- void **push\_front** (D const &d) noexcept  
*Add element at the beginning of the list.*
- void **remove** (Iter const &i) noexcept  
*Remove element pointed to by the iterator.*
- unsigned long **size** () const noexcept  
*Get the length of the list.*
- D const & **operator[]** (unsigned long idx) const noexcept  
*Random access.*
- D & **operator[]** (unsigned long idx) noexcept  
*Random access.*
- `Iter` **items** () noexcept  
*Get iterator for the list elements.*

### 16.50.1 Detailed Description

`template<typename D, template< typename A > class Alloc = New_allocator>`  
`class cxx::List< D, Alloc >`

Doubly linked list, with internal allocation.

Container for items of type D, implemented by a doubly linked list. Alloc defines the allocator policy.

Definition at line 323 of file `list`.

### 16.50.2 Member Function Documentation

#### 16.50.2.1 `operator[]()` [1/2]

```
template<typename D, template< typename A > class Alloc = New_allocator>
D const & cxx::List< D, Alloc >::operator[] (
    unsigned long idx) const [inline], [noexcept]
```

Random access.

Complexity is O(n).

Definition at line 393 of file `list`.

### 16.50.2.2 operator[]() [2/2]

```
template<typename D, template< typename A > class Alloc = New_allocator>
D & cxx::List< D, Alloc >::operator[] (
    unsigned long idx) [inline], [noexcept]
```

Random access.

Complexity is O(n).

Definition at line 397 of file [list](#).

The documentation for this class was generated from the following file:

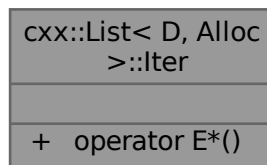
- l4/cxx/list

## 16.51 cxx::List< D, Alloc >::Iter Class Reference

Iterator.

```
#include <list>
```

Collaboration diagram for cxx::List< D, Alloc >::Iter:



### Public Member Functions

- **operator E\* ()** const noexcept  
*operator for testing validity (syntactically equal to pointers)*

### 16.51.1 Detailed Description

```
template<typename D, template< typename A > class Alloc = New_allocator>
class cxx::List< D, Alloc >::Iter
```

Iterator.

Forward and backward iterable.

Definition at line 343 of file [list](#).

The documentation for this class was generated from the following file:

- l4/cxx/list

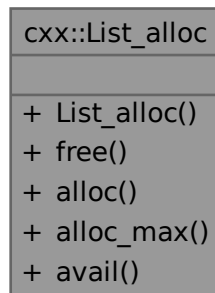


## 16.52 cxx::List\_alloc Class Reference

Standard list-based allocator.

```
#include <list_alloc>
```

Collaboration diagram for cxx::List\_alloc:



### Public Member Functions

- [List\\_alloc\(\)](#)  
*Initializes an empty list allocator.*
- void [free](#) (void \*block, unsigned long size, bool initial\_free=false)  
*Return a free memory block to the allocator.*
- void \* [alloc](#) (unsigned long size, unsigned long align, unsigned long lower=0, unsigned long upper=~0UL)  
*Allocate a memory block.*
- void \* [alloc\\_max](#) (unsigned long min, unsigned long \*max, unsigned long align, unsigned granularity, unsigned long lower=0, unsigned long upper=~0UL)  
*Allocate a memory block of  $min \leq size \leq max$ .*
- unsigned long [avail](#) ()  
*Get the amount of available memory.*

### 16.52.1 Detailed Description

Standard list-based allocator.

Definition at line 21 of file [list\\_alloc](#).

### 16.52.2 Constructor & Destructor Documentation

#### 16.52.2.1 List\_alloc()

```
cxx::List_alloc::List_alloc () [inline]
```

Initializes an empty list allocator.

#### Note

To initialize the allocator with available memory use the [free\(\)](#) function.

Definition at line 46 of file [list\\_alloc](#).

## 16.52.3 Member Function Documentation

### 16.52.3.1 alloc()

```
void * cxx::List_alloc::alloc (
    unsigned long size,
    unsigned long align,
    unsigned long lower = 0,
    unsigned long upper = ~0UL) [inline]
```

Allocate a memory block.

#### Parameters

<i>size</i>	Size of the memory block.
<i>align</i>	Alignment constraint.
<i>lower</i>	Lower bound of the physical region the memory block should be allocated from.
<i>upper</i>	Upper bound of the physical region the memory block should be allocated from, value is inclusive.

#### Returns

Pointer to memory block

#### Precondition

$0 < \text{size} \leq \sim 0\text{UL} - 32$ .

Definition at line 389 of file [list\\_alloc](#).

### 16.52.3.2 alloc\_max()

```
void * cxx::List_alloc::alloc_max (
    unsigned long min,
    unsigned long * max,
    unsigned long align,
    unsigned granularity,
    unsigned long lower = 0,
    unsigned long upper = ~0UL) [inline]
```

Allocate a memory block of  $\text{min} \leq \text{size} \leq \text{max}$ .

#### Parameters

	<i>min</i>	Minimal size to allocate (in bytes).
<i>in, out</i>	<i>max</i>	Maximum size to allocate (in bytes). The actual allocated size is returned here.
	<i>align</i>	Alignment constraint.
	<i>granularity</i>	Granularity to use for the allocation (power of 2).

	<i>lower</i>	Lower bound of the physical region the memory block should be allocated from.
	<i>upper</i>	Upper bound of the physical region the memory block should be allocated from, value is inclusive.

**Returns**

Pointer to memory block

**Precondition**

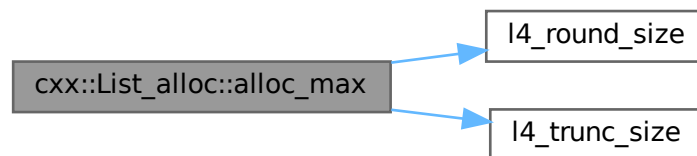
$0 < \text{min} \leq \sim 0\text{UL} - 32$ .

$0 < \text{max}$ .

Definition at line 269 of file `list_alloc`.

References `l4_round_size()`, and `l4_trunc_size()`.

Here is the call graph for this function:

**16.52.3.3 avail()**

```
unsigned long cxx::List_alloc::avail () [inline]
```

Get the amount of available memory.

**Returns**

Available memory in bytes

Definition at line 477 of file `list_alloc`.

**16.52.3.4 free()**

```
void cxx::List_alloc::free (
    void * block,
    unsigned long size,
    bool initial_free = false) [inline]
```

Return a free memory block to the allocator.

**Parameters**

<i>block</i>	Pointer to memory block.
<i>size</i>	Size of memory block.
<i>initial_free</i>	Set to true for putting fresh memory to the allocator. This will enforce alignment on that memory.

**Precondition**

block must not be NULL.  
 $2 * \text{sizeof}(\text{void} *) \leq \text{size} \leq \sim 0\text{UL} - 32.$

Definition at line 228 of file [list\\_alloc](#).

The documentation for this class was generated from the following file:

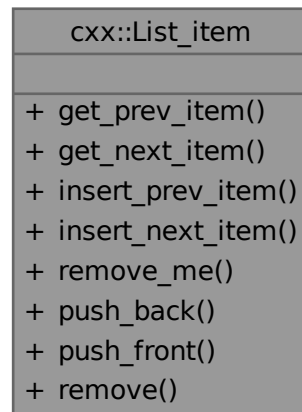
- [l4/cxx/list\\_alloc](#)

## 16.53 cxx::List\_item Class Reference

Basic list item.

```
#include <list>
```

Collaboration diagram for cxx::List\_item:

**Data Structures**

- class [Iter](#)  
*Iterator for a list of ListItem-s.*
- class [T\\_iter](#)  
*Iterator for derived classes from ListItem.*

## Public Member Functions

- [List\\_item](#) \* **get\_prev\_item** () const noexcept  
*Get previous item.*
- [List\\_item](#) \* **get\_next\_item** () const noexcept  
*Get next item.*
- void **insert\_prev\_item** ([List\\_item](#) \*p) noexcept  
*Insert item p before this item.*
- void **insert\_next\_item** ([List\\_item](#) \*p) noexcept  
*Insert item p after this item.*
- void **remove\_me** () noexcept  
*Remove this item from the list.*

## Static Public Member Functions

- template<typename C, typename N>  
static C \* **push\_back** (C \*head, N \*p) noexcept  
*Append item to a list.*
- template<typename C, typename N>  
static C \* **push\_front** (C \*head, N \*p) noexcept  
*Prepend item to a list.*
- template<typename C, typename N>  
static C \* **remove** (C \*head, N \*p) noexcept  
*Remove item from a list.*

### 16.53.1 Detailed Description

Basic list item.

Basic item that can be member of a doubly linked, cyclic list.

Definition at line 26 of file [list](#).

### 16.53.2 Member Function Documentation

#### 16.53.2.1 push\_back()

```
template<typename C, typename N>
C * cxx::List_item::push_back (
    C * head,
    N * p) [inline], [static], [noexcept]
```

Append item to a list.

Convenience function for empty-head corner case.

#### Parameters

---

<i>head</i>	Pointer to the current list head.
<i>p</i>	Pointer to new item.

**Returns**

the pointer to the new head.

Definition at line 237 of file [list](#).

Referenced by [cxx::List< D, Alloc >::push\\_back\(\)](#).

Here is the caller graph for this function:

**16.53.2.2 push\_front()**

```

template<typename C, typename N>
C * cxx::List_item::push_front (
    C * head,
    N * p) [inline], [static], [noexcept]
  
```

Prepend item to a list.

Convenience function for empty-head corner case.

**Parameters**

<i>head</i>	pointer to the current list head.
<i>p</i>	pointer to new item.

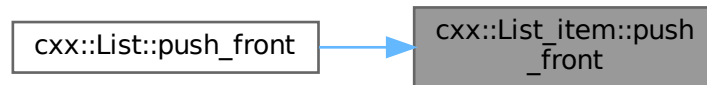
**Returns**

the pointer to the new head.

Definition at line 248 of file [list](#).

Referenced by [cxx::List< D, Alloc >::push\\_front\(\)](#).

Here is the caller graph for this function:



### 16.53.2.3 remove()

```

template<typename C, typename N>
C * cxx::List_item::remove (
    C * head,
    N * p) [inline], [static], [noexcept]
  
```

Remove item from a list.

Convenience function for remove-head corner case.

#### Parameters

<i>head</i>	pointer to the current list head.
<i>p</i>	pointer to the item to remove.

#### Returns

the pointer to the new head.

Definition at line 258 of file [list](#).

Referenced by [cxx::List< D, Alloc >::remove\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

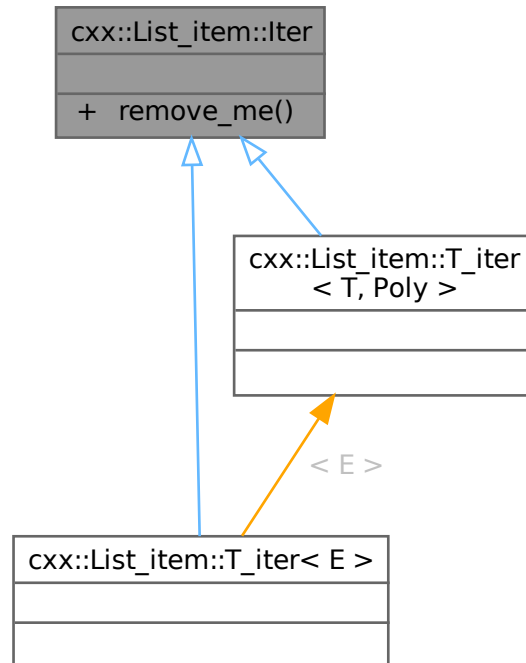
- `I4/cxx/list`

## 16.54 cxx::List\_item::Iter Class Reference

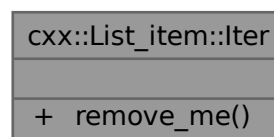
Iterator for a list of ListItem-s.

```
#include <list>
```

Inheritance diagram for cxx::List\_item::Iter:



Collaboration diagram for cxx::List\_item::Iter:



### Public Member Functions

- [List\\_item](#) \* **remove\_me** () noexcept  
Remove item pointed to by iterator, and return pointer to element.



### 16.54.1 Detailed Description

Iterator for a list of ListItem-s.

The Iterator iterates till it finds the first element again.

Definition at line 34 of file [list](#).

The documentation for this class was generated from the following file:

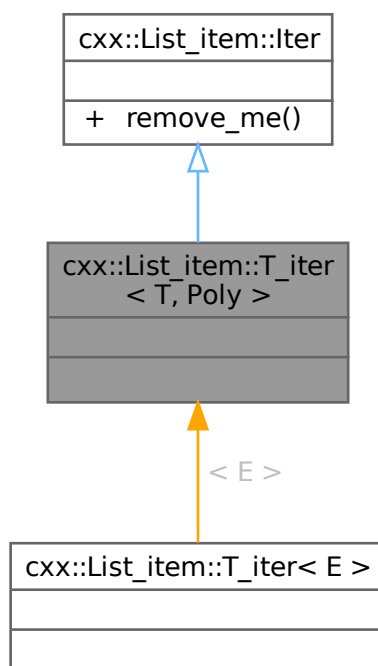
- l4/cxx/list

## 16.55 cxx::List\_item::T\_iter< T, Poly > Class Template Reference

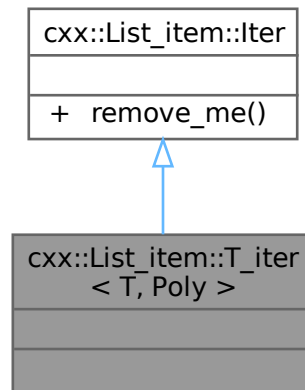
Iterator for derived classes from ListItem.

```
#include <list>
```

Inheritance diagram for cxx::List\_item::T\_iter< T, Poly >:



Collaboration diagram for `cxx::List_item::T_iter< T, Poly >`:



#### Additional Inherited Members

#### Public Member Functions inherited from `cxx::List_item::Iter`

- `List_item * remove_me ()` noexcept  
*Remove item pointed to by iterator, and return pointer to element.*

### 16.55.1 Detailed Description

```
template<typename T, bool Poly = false>
class cxx::List_item::T_iter< T, Poly >
```

Iterator for derived classes from `ListItem`.

Allows direct access to derived classes by `*` operator.

Example: `class Foo : public ListItem { public: typedef T_iter<Foo> Iter; ... };`

Definition at line 108 of file `list`.

The documentation for this class was generated from the following file:

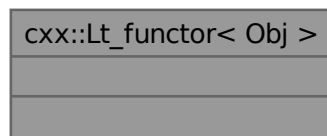
- `I4/cxx/list`

## 16.56 cxx::Lt\_functor< Obj > Struct Template Reference

Generic comparator class that defaults to the less-than operator.

```
#include <std_ops>
```

Collaboration diagram for cxx::Lt\_functor< Obj >:



### 16.56.1 Detailed Description

```
template<typename Obj>
struct cxx::Lt_functor< Obj >
```

Generic comparator class that defaults to the less-than operator.

Definition at line 18 of file [std\\_ops](#).

The documentation for this struct was generated from the following file:

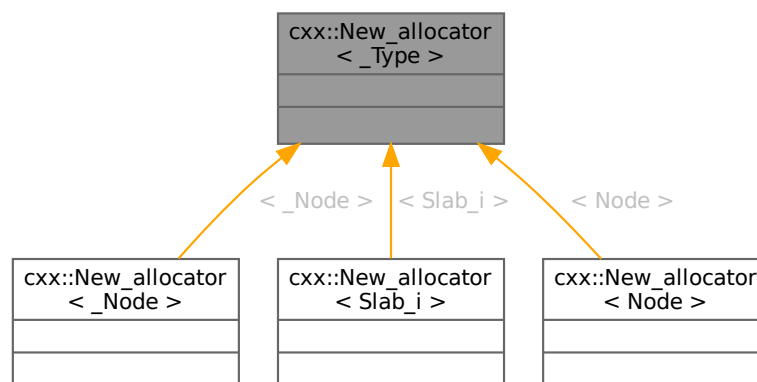
- l4/cxx/std\_ops

## 16.57 cxx::New\_allocator< \_Type > Class Template Reference

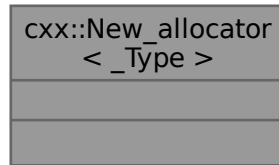
Standard allocator based on `operator new ()`.

```
#include <std_alloc>
```

Inheritance diagram for cxx::New\_allocator< \_Type >:



Collaboration diagram for `cxx::New_allocator<_Type>`:



### 16.57.1 Detailed Description

```
template<typename _Type>
class cxx::New_allocator<_Type>
```

Standard allocator based on `operator new ()`.

This allocator is the default allocator used for the *cxx Containers*, such as [cxx::Avl\\_set](#) and [cxx::Avl\\_map](#), to allocate the internal data structures.

Definition at line 56 of file [std\\_alloc](#).

The documentation for this class was generated from the following file:

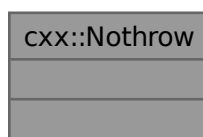
- `I4/cxx/std_alloc`

## 16.58 cxx::Nothrow Class Reference

Helper type to distinguish the `operator new` version that does not throw exceptions.

```
#include <std_alloc>
```

Collaboration diagram for `cxx::Nothrow`:



### 16.58.1 Detailed Description

Helper type to distinguish the `operator new` version that does not throw exceptions.

Definition at line 19 of file [std\\_alloc](#).

The documentation for this class was generated from the following file:

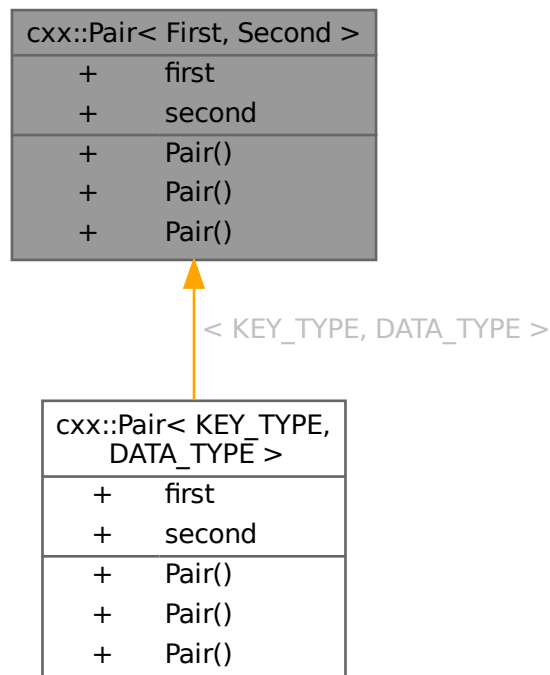
- `l4/cxx/std_alloc`

## 16.59 cxx::Pair< First, Second > Struct Template Reference

[Pair](#) of two values.

```
#include <pair>
```

Inheritance diagram for `cxx::Pair< First, Second >`:



Collaboration diagram for `cxx::Pair< First, Second >`:

<code>cxx::Pair&lt; First, Second &gt;</code>	
+	<code>first</code>
+	<code>second</code>
+	<code>Pair()</code>
+	<code>Pair()</code>
+	<code>Pair()</code>

### Public Types

- typedef First **First\_type**  
*Type of first value.*
- typedef Second **Second\_type**  
*Type of second value.*

### Public Member Functions

- `template<typename A1, typename A2>`  
`Pair (A1 &&first, A2 &&second)`  
*Create a pair from the two values.*
- `template<typename A1>`  
`Pair (A1 &&first)`  
*Create a pair, default constructing the second value.*
- `Pair ()=default`  
*Default construction.*

### Data Fields

- First **first**  
*First value.*
- Second **second**  
*Second value.*

## 16.59.1 Detailed Description

`template<typename First, typename Second>`  
`struct cxx::Pair< First, Second >`

`Pair` of two values.

Standard container for a pair of values.

### Parameters

---

<i>First</i>	Type of the first value.
<i>Second</i>	Type of the second value.

Definition at line 27 of file [pair](#).

## 16.59.2 Constructor & Destructor Documentation

### 16.59.2.1 `Pair()` [1/2]

```
template<typename First, typename Second>
template<typename A1, typename A2>
cxx::Pair< First, Second >::Pair (
    A1 && first,
    A2 && second) [inline]
```

Create a pair from the two values.

#### Parameters

<i>first</i>	The first value.
<i>second</i>	The second value.

Definition at line 45 of file [pair](#).

### 16.59.2.2 `Pair()` [2/2]

```
template<typename First, typename Second>
template<typename A1>
cxx::Pair< First, Second >::Pair (
    A1 && first) [inline]
```

Create a pair, default constructing the second value.

#### Parameters

<i>first</i>	The first value.
--------------	------------------

Definition at line 53 of file [pair](#).

The documentation for this struct was generated from the following file:

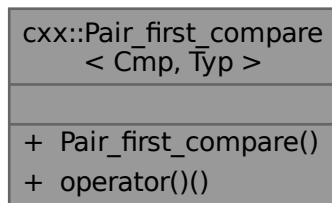
- [I4/cxx/pair](#)

## 16.60 `cxx::Pair_first_compare< Cmp, Typ >` Class Template Reference

Comparison functor for [Pair](#).

```
#include <pair>
```

Collaboration diagram for `cxx::Pair_first_compare< Cmp, Typ >`:



### Public Member Functions

- [Pair\\_first\\_compare](#) (Cmp const &cmp=Cmp())  
*Construction.*
- bool [operator\(\)](#) (Typ const &l, Typ const &r) const  
*Do the comparison based on the first value.*

### 16.60.1 Detailed Description

```
template<typename Cmp, typename Typ>
class cxx::Pair_first_compare< Cmp, Typ >
```

Comparison functor for [Pair](#).

#### Parameters

<i>Cmp</i>	Comparison functor for the first value of the pair.
<i>Typ</i>	The pair type.

This functor can be used to compare [Pair](#) values with respect to the first value.

Definition at line 74 of file [pair](#).

### 16.60.2 Constructor & Destructor Documentation

#### 16.60.2.1 `Pair_first_compare()`

```
template<typename Cmp, typename Typ>
cxx::Pair_first_compare< Cmp, Typ >::Pair_first_compare (
    Cmp const & cmp = Cmp()) [inline]
```

Construction.

#### Parameters



<code>cmp</code>	The comparison functor used for the first value.
------------------	--------------------------------------------------

Definition at line 84 of file [pair](#).

## 16.60.3 Member Function Documentation

### 16.60.3.1 `operator>()()`

```
template<typename Cmp, typename Typ>
bool cxx::Pair_first_compare< Cmp, Typ >::operator() (
    Typ const & l,
    Typ const & r) const [inline]
```

Do the comparison based on the first value.

#### Parameters

<code>l</code>	The lefthand value.
<code>r</code>	The righthand value.

Definition at line 91 of file [pair](#).

The documentation for this class was generated from the following file:

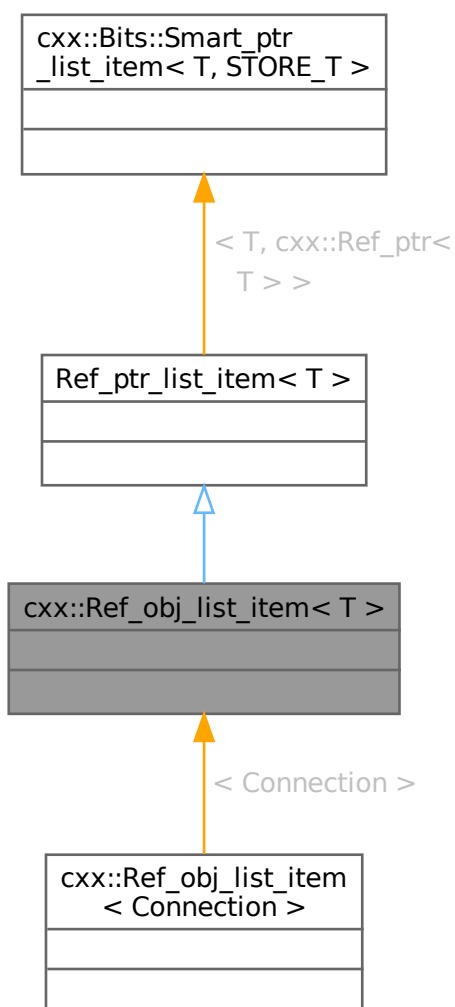
- [l4/cxx/pair](#)

## 16.61 `cxx::Ref_obj_list_item< T >` Struct Template Reference

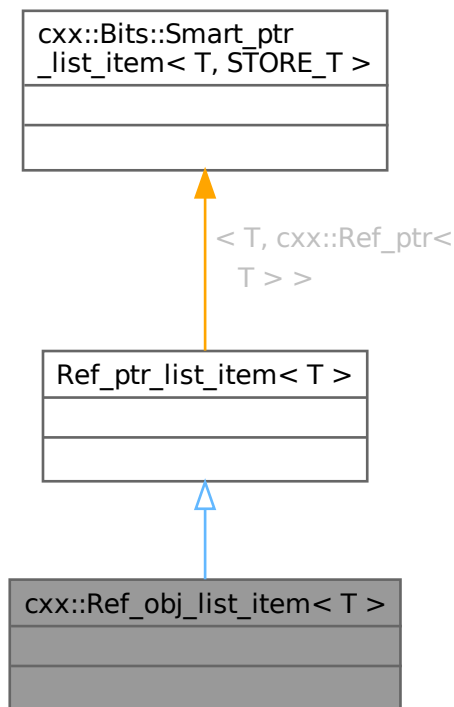
Item for list linked via [cxx::Ref\\_ptr](#) with default refence counting.

```
#include <ref_ptr_list>
```

Inheritance diagram for `cxx::Ref_obj_list_item< T >`:



Collaboration diagram for `cxx::Ref_obj_list_item< T >`:



### 16.61.1 Detailed Description

```
template<typename T>
struct cxx::Ref_obj_list_item< T >
```

Item for list linked via `cxx::Ref_ptr` with default reference counting.

Definition at line 26 of file `ref_ptr_list`.

The documentation for this struct was generated from the following file:

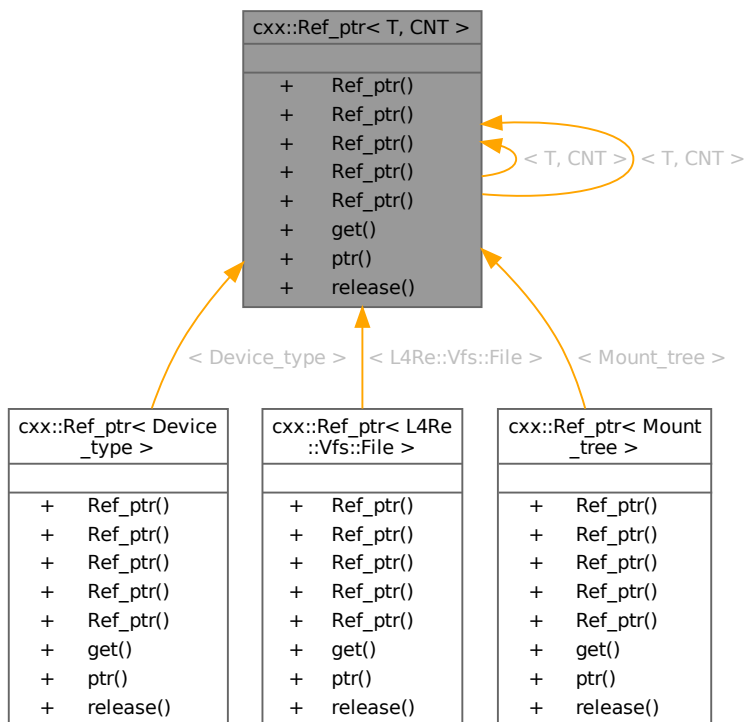
- `I4/cxx/ref_ptr_list`

## 16.62 `cxx::Ref_ptr< T, CNT >` Class Template Reference

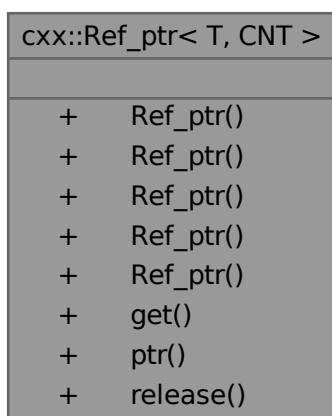
A reference-counting pointer with automatic cleanup.

```
#include <ref_ptr>
```

Inheritance diagram for `cxx::Ref_ptr< T, CNT >`:



Collaboration diagram for `cxx::Ref_ptr< T, CNT >`:



## Public Member Functions

- `Ref_ptr()` noexcept

- Default constructor creates a pointer with no managed object.*
- `Ref_ptr (Wp const &o)` noexcept  
*Create a shared pointer from a weak pointer.*
- `Ref_ptr (decltype(nullptr) n)` noexcept  
*allow creation from `nullptr`*
- `template<typename X>`  
`Ref_ptr (X *o)` noexcept  
*Create a shared pointer from a raw pointer.*
- `Ref_ptr (T *o, bool d)` noexcept  
*Create a shared pointer from a raw pointer without creating a new reference.*
- `T * get ()` const noexcept  
*Return a raw pointer to the object this shared pointer points to.*
- `T * ptr ()` const noexcept  
*Return a raw pointer to the object this shared pointer points to.*
- `T * release ()` noexcept  
*Release the shared pointer without removing the reference.*

### 16.62.1 Detailed Description

`template<typename T = void, template< typename X > class CNT = Default_ref_counter>`  
`class cxx::Ref_ptr< T, CNT >`

A reference-counting pointer with automatic cleanup.

#### Template Parameters

<i>T</i>	Type of object the pointer points to.
<i>CNT</i>	Type of management class that manages the life time of the object.

This pointer is similar to the standard C++-11 `shared_ptr` but it does the reference counting directly in the object being pointed to, so that no additional management structures need to be allocated from the heap.

Classes that use this pointer type must implement two functions:

```
int remove_ref()
```

is called when a reference is removed and must return 0 when there are no further references to the object.

```
void add_ref()
```

is called when another `ref_ptr` to the object is created.

`Ref_obj` provides a simple implementation of this interface from which classes may inherit.

#### Examples

[tmpfs/lib/src/fs.cc](#).

Definition at line 70 of file [ref\\_ptr](#).

## 16.62.2 Constructor & Destructor Documentation

### 16.62.2.1 Ref\_ptr() [1/3]

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
cxx::Ref_ptr< T, CNT >::Ref_ptr (
    Wp const & o) [inline], [noexcept]
```

Create a shared pointer from a weak pointer.

Increases references.

Definition at line 88 of file [ref\\_ptr](#).

### 16.62.2.2 Ref\_ptr() [2/3]

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
template<typename X>
cxx::Ref_ptr< T, CNT >::Ref_ptr (
    X * o) [inline], [explicit], [noexcept]
```

Create a shared pointer from a raw pointer.

In contrast to C++11 `shared_ptr` it is safe to use this constructor multiple times and have the same reference counter.

Definition at line 101 of file [ref\\_ptr](#).

### 16.62.2.3 Ref\_ptr() [3/3]

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
cxx::Ref_ptr< T, CNT >::Ref_ptr (
    T * o,
    bool d) [inline], [noexcept]
```

Create a shared pointer from a raw pointer without creating a new reference.

#### Parameters

<i>o</i>	Pointer to the object.
<i>d</i>	Dummy parameter to select this constructor at compile time. The value may be true or false.

This is the counterpart to [release\(\)](#).

Definition at line 114 of file [ref\\_ptr](#).

## 16.62.3 Member Function Documentation

### 16.62.3.1 get()

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
T * cxx::Ref_ptr< T, CNT >::get () const [inline], [noexcept]
```

Return a raw pointer to the object this shared pointer points to.

This does not release the pointer or decrease the reference count.

Definition at line 121 of file [ref\\_ptr](#).

### 16.62.3.2 ptr()

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
T * cxx::Ref_ptr< T, CNT >::ptr () const [inline], [noexcept]
```

Return a raw pointer to the object this shared pointer points to.

This does not release the pointer or decrease the reference count.

Definition at line 127 of file [ref\\_ptr](#).

### 16.62.3.3 release()

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
T * cxx::Ref_ptr< T, CNT >::release () [inline], [noexcept]
```

Release the shared pointer without removing the reference.

#### Returns

A raw pointer to the managed object.

Definition at line 138 of file [ref\\_ptr](#).

The documentation for this class was generated from the following file:

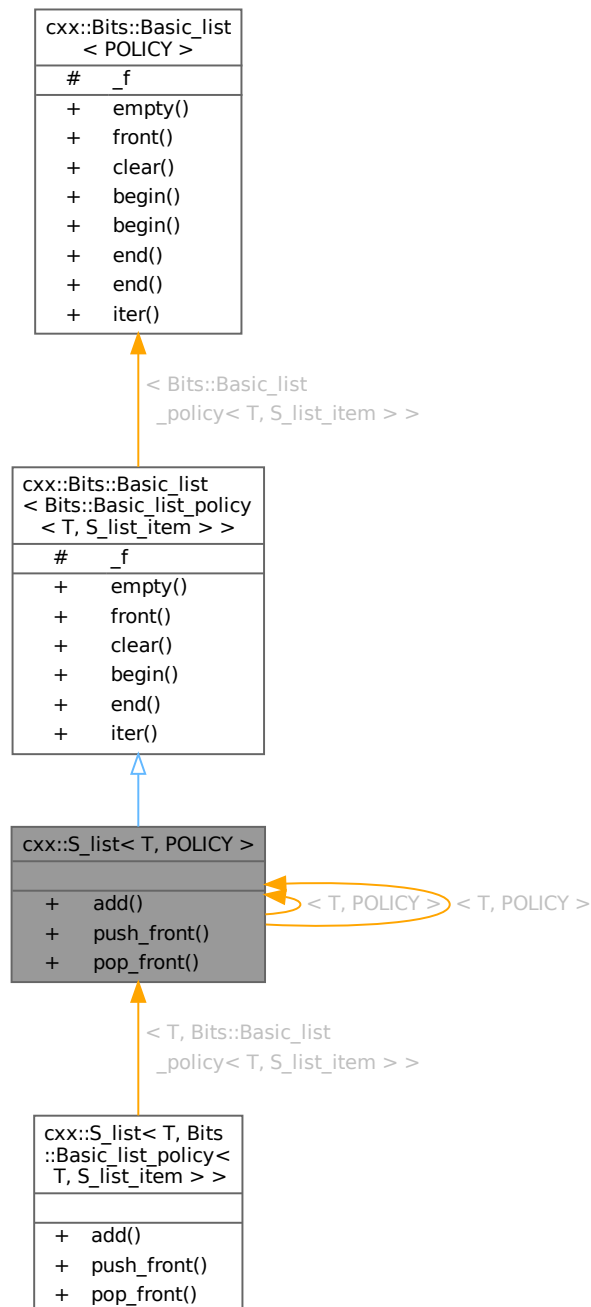
- l4/cxx/ref\_ptr

## 16.63 cxx::S\_list< T, POLICY > Class Template Reference

Simple single-linked list.

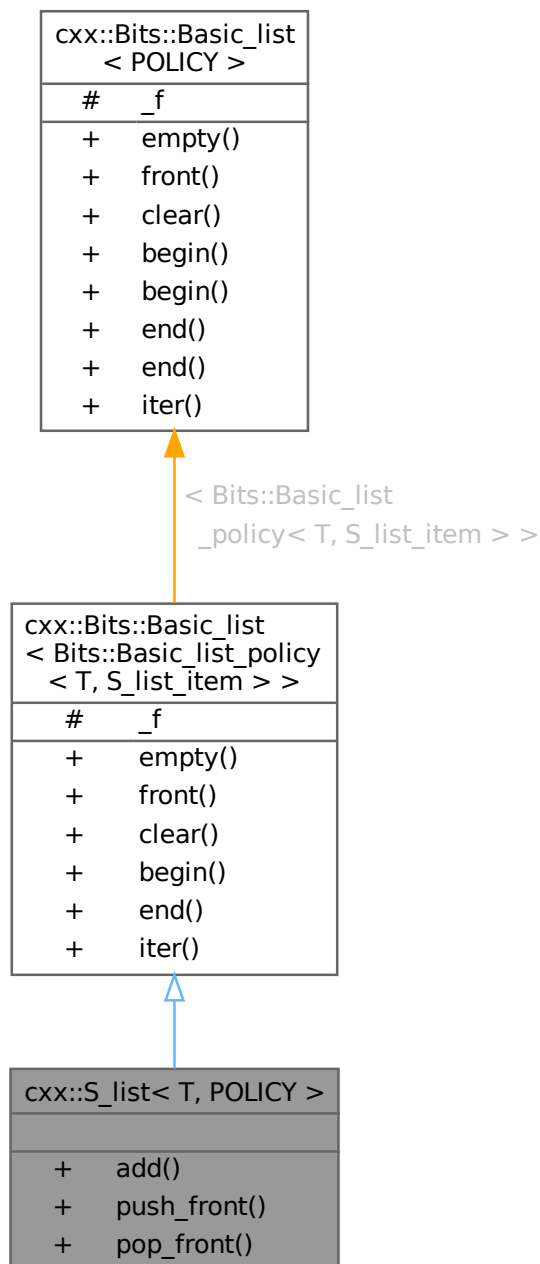
```
#include <slist>
```

Inheritance diagram for cxx::S\_list< T, POLICY >:





Collaboration diagram for cxx::S\_list< T, POLICY >:



### Public Member Functions

- void **add** (T \*e)  
*Add an element to the front of the list.*
- void **push\_front** (T \*e)  
*Add an element to the front of the list.*
- T \* **pop\_front** ()  
*Remove and return the head element of the list.*

**Public Member Functions inherited from****cxx::Bits::Basic\_list< Bits::Basic\_list\_policy< T, S\_list\_item > >**

- bool **empty** () const  
*Check if the list is empty.*
- Value\_type **front** () const  
*Return the first element in the list.*
- void **clear** ()  
*Remove all elements from the list.*
- Iterator **begin** ()  
*Return an iterator to the beginning of the list.*
- Const\_iterator **end** () const  
*Return a const iterator to the end of the list.*

**Additional Inherited Members****Static Public Member Functions inherited from****cxx::Bits::Basic\_list< Bits::Basic\_list\_policy< T, S\_list\_item > >**

- static Const\_iterator **iter** (Const\_value\_type c)  
*Return a const iterator that begins at the given element.*

**Protected Attributes inherited from****cxx::Bits::Basic\_list< Bits::Basic\_list\_policy< T, S\_list\_item > >**

- Bits::Basic\_list\_policy< T, S\_list\_item >::Head\_type **\_f**  
*Pointer to front of the list.*

**16.63.1 Detailed Description**

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, S_list_item >>
class cxx::S_list< T, POLICY >
```

Simple single-linked list.

**Template Parameters**

<i>T</i>	Type of elements saved in the list. Must inherit from cxx::S_list_item
----------	------------------------------------------------------------------------

Definition at line 40 of file [slist](#).

## 16.63.2 Member Function Documentation

### 16.63.2.1 `pop_front()`

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, S_list_item >>
T * cxx::S_list< T, POLICY >::pop_front () [inline]
```

Remove and return the head element of the list.

#### Precondition

The list must not be empty or the behaviour will be undefined.

Definition at line 89 of file `slist`.

The documentation for this class was generated from the following file:

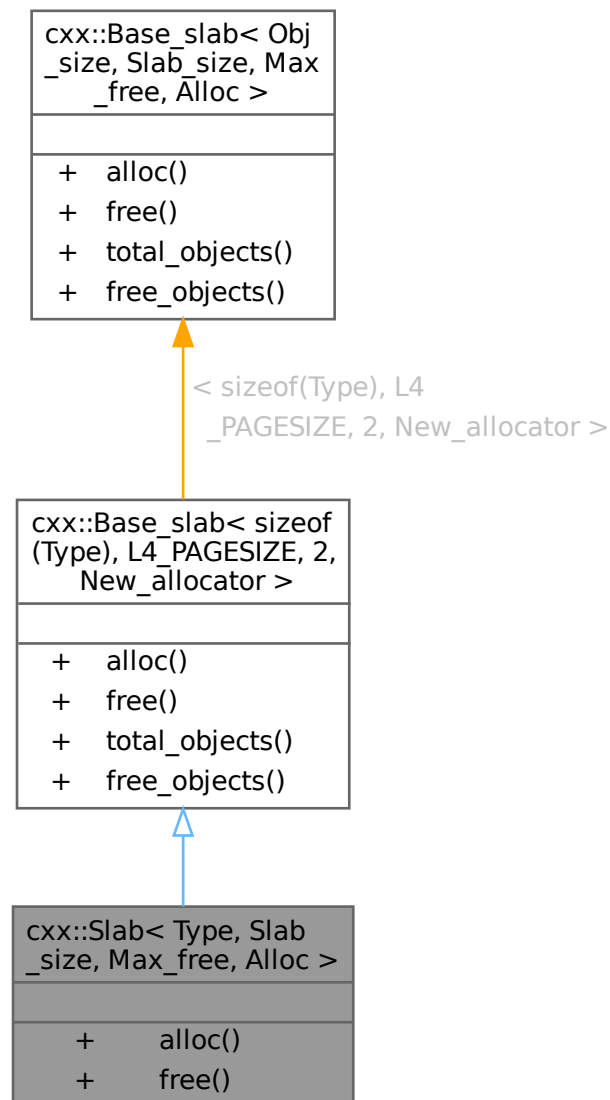
- `l4/cxx/slist`

## 16.64 `cxx::Slab< Type, Slab_size, Max_free, Alloc >` Class Template Reference

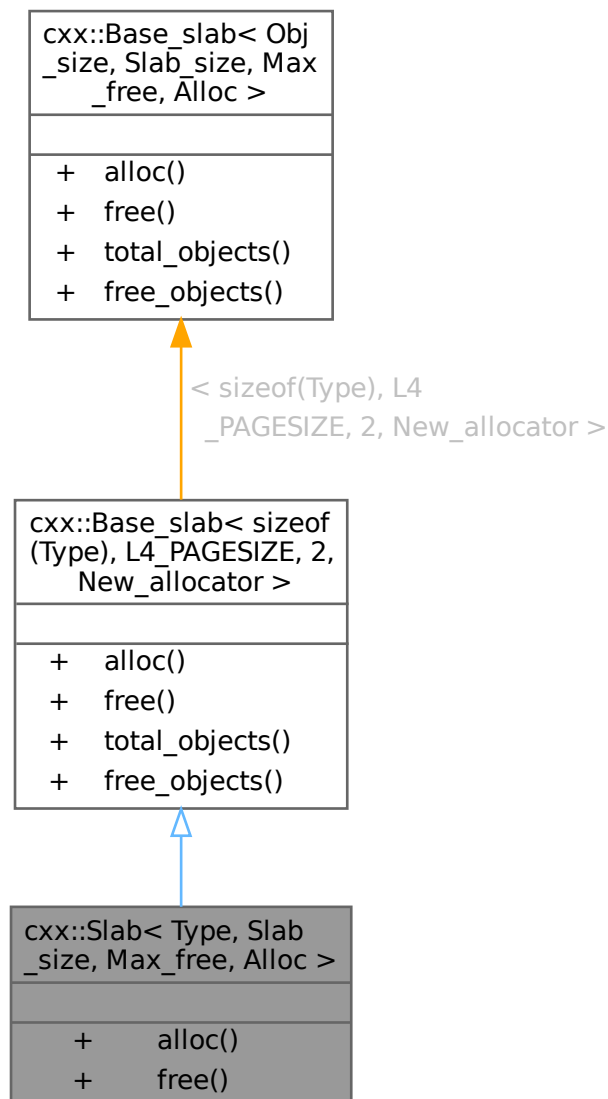
`Slab` allocator for object of type `Type`.

```
#include <slab_alloc>
```

Inheritance diagram for `cxx::Slab< Type, Slab_size, Max_free, Alloc >`:



Collaboration diagram for `cxx::Slab< Type, Slab_size, Max_free, Alloc >`:



### Public Member Functions

- `Type * alloc ()` noexcept  
*Allocate an object of type `Type`.*
- `void free (Type *o)` noexcept  
*Free the object addressed by `o`.*

### Public Member Functions inherited from

`cxx::Base_slab< sizeof(Type), L4_PAGESIZE, 2, New_allocator >`

- `void * alloc ()` noexcept

- *Allocate a new object.*  
void [free](#) (void \*\_o) noexcept
- *Free the given object (\_o).*  
unsigned [total\\_objects](#) () const noexcept
- *Get the total number of objects managed by the slab allocator.*  
unsigned [free\\_objects](#) () const noexcept
- *Get the number of objects which can be allocated before a new empty slab needs to be added to the slab allocator.*

### Additional Inherited Members

### Public Types inherited from

[cxx::Base\\_slab](#)< [sizeof\(Type\)](#), [L4\\_PAGESIZE](#), [2](#), [New\\_allocator](#) >

- typedef [New\\_allocator](#)< Slab\_i > [Slab\\_alloc](#)  
*Type of the backend allocator.*

## 16.64.1 Detailed Description

```
template<typename Type, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class
Alloc = New_allocator>
class cxx::Slab< Type, Slab_size, Max_free, Alloc >
```

[Slab](#) allocator for object of type [Type](#).

### Template Parameters

<i>Type</i>	The type of the objects to manage.
<i>Slab_size</i>	Size of a slab.
<i>Max_free</i>	The maximum number of free slabs.
<i>Alloc</i>	The allocator for the slabs.

Definition at line [335](#) of file [slab\\_alloc](#).

## 16.64.2 Member Function Documentation

### 16.64.2.1 [alloc\(\)](#)

```
template<typename Type, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
Type * cxx::Slab< Type, Slab_size, Max_free, Alloc >::alloc () [inline], [noexcept]
```

Allocate an object of type [Type](#).

### Returns

A pointer to the object just allocated, or 0 on failure.

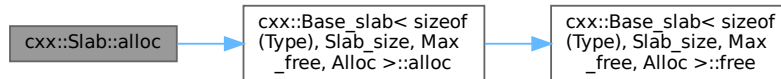
**Note**

The user is responsible for initializing the object.

Definition at line 355 of file [slab\\_alloc](#).

References [cxx::Base\\_slab< sizeof\(Type\), Slab\\_size, Max\\_free, Alloc >::alloc\(\)](#).

Here is the call graph for this function:

**16.64.2.2 free()**

```

template<typename Type, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
void cxx::Slab< Type, Slab_size, Max_free, Alloc >::free (
    Type * o) [inline], [noexcept]
  
```

Free the object addressed by `o`.

**Parameters**

<code>o</code>	The pointer to the object to free.
----------------	------------------------------------

**Precondition**

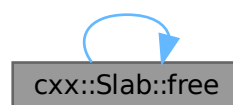
The object must have been allocated with this allocator.

Definition at line 366 of file [slab\\_alloc](#).

References [free\(\)](#).

Referenced by [free\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- `l4/cxx/slab_alloc`

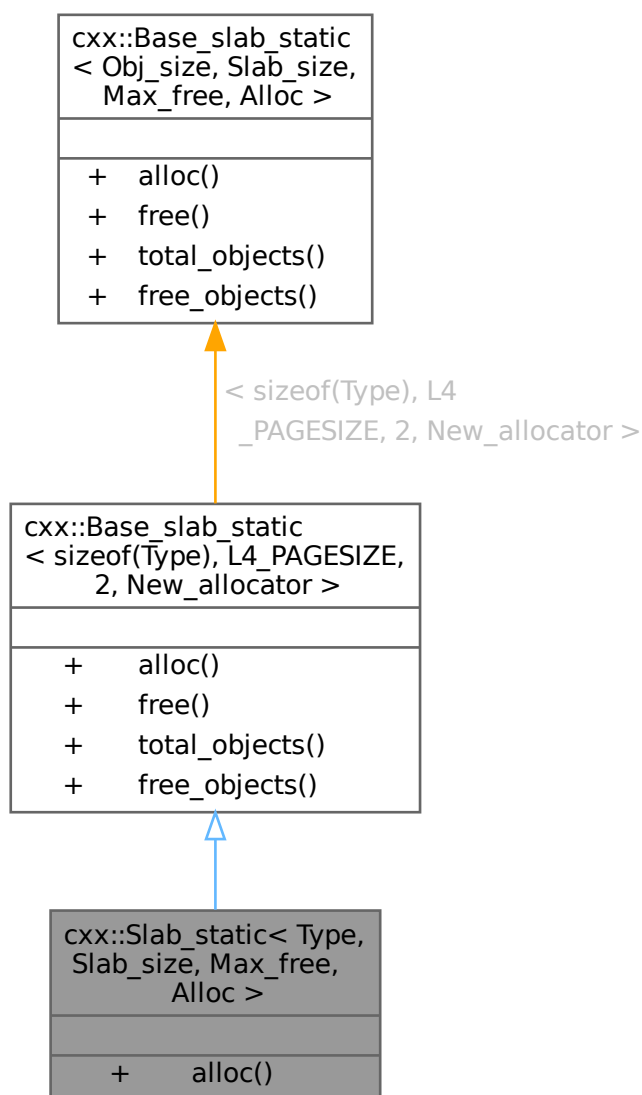
## 16.65 `cxx::Slab_static< Type, Slab_size, Max_free, Alloc > Class` Template Reference

Merged slab allocator (allocators for objects of the same size are merged together).

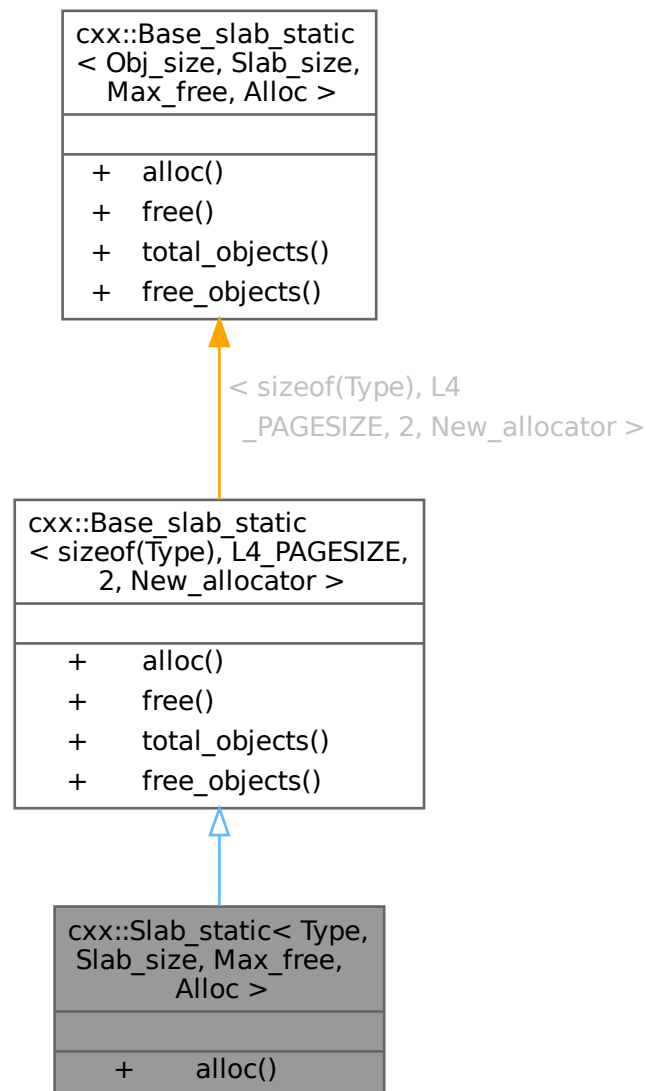
```
#include <slab_alloc>
```



Inheritance diagram for cxx::Slab\_static< Type, Slab\_size, Max\_free, Alloc >:



Collaboration diagram for `cxx::Slab_static< Type, Slab_size, Max_free, Alloc >`:



### Public Member Functions

- `Type * alloc ()` noexcept  
*Allocate an object of type `Type`.*

### Public Member Functions inherited from

`cxx::Base_slab_static< sizeof(Type), L4_PAGESIZE, 2, New_allocator >`

- `void * alloc ()` noexcept  
*Allocate an object.*

- void `free` (void \*p) noexcept  
*Free the given object (p).*
- unsigned `total_objects` () const noexcept  
*Get the total number of objects managed by the slab allocator.*
- unsigned `free_objects` () const noexcept  
*Get the number of free objects in the slab allocator.*

### Additional Inherited Members

### Public Types inherited from

`cxx::Base_slab_static< sizeof(Type), L4_PAGESIZE, 2, New_allocator >`

## 16.65.1 Detailed Description

```
template<typename Type, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class
Alloc = New_allocator>
class cxx::Slab_static< Type, Slab_size, Max_free, Alloc >
```

Merged slab allocator (allocators for objects of the same size are merged together).

### Template Parameters

<i>Type</i>	The type of the objects to manage.
<i>Slab_size</i>	The size of a slab.
<i>Max_free</i>	The maximum number of free slabs.
<i>Alloc</i>	The allocator for the slabs.

This slab allocator class is useful for merging slab allocators with the same parameters (equal `sizeof(Type)`, `Slab_size`, `Max_free`, and `Alloc` parameters) together and share the overhead for the slab caches among all equal-sized objects.

Definition at line 465 of file `slab_alloc`.

## 16.65.2 Member Function Documentation

### 16.65.2.1 `alloc()`

```
template<typename Type, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
Type * cxx::Slab_static< Type, Slab_size, Max_free, Alloc >::alloc () [inline], [noexcept]
```

Allocate an object of type `Type`.

### Returns

A pointer to the just allocated object, or 0 on failure.

**Note**

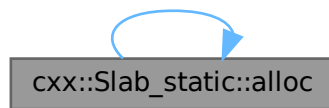
The object is not zeroed out by the slab allocator.

Definition at line 478 of file [slab\\_alloc](#).

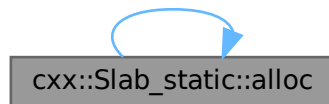
References [alloc\(\)](#).

Referenced by [alloc\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

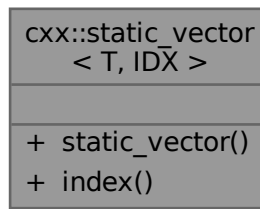
- `I4/cxx/slab_alloc`

## 16.66 `cxx::static_vector< T, IDX >` Class Template Reference

Simple encapsulation for a dynamically allocated array.

```
#include <static_vector>
```

Collaboration diagram for `cxx::static_vector< T, IDX >`:



### Public Member Functions

- `template<typename X, typename = enable_if_t<is_convertible<X, T>::value>>`  
**static\_vector** ([static\\_vector](#)< X, IDX > const &o)  
*Conversion from compatible arrays.*
- `index_type` **index** (value\_type const \*o) const  
*Get the index of the given element of the array.*

### 16.66.1 Detailed Description

```
template<typename T, typename IDX = unsigned>
class cxx::static_vector< T, IDX >
```

Simple encapsulation for a dynamically allocated array.

The main purpose of this class is to support C++11 range for for simple dynamically allocated array with static size.

Definition at line 16 of file [static\\_vector](#).

The documentation for this class was generated from the following file:

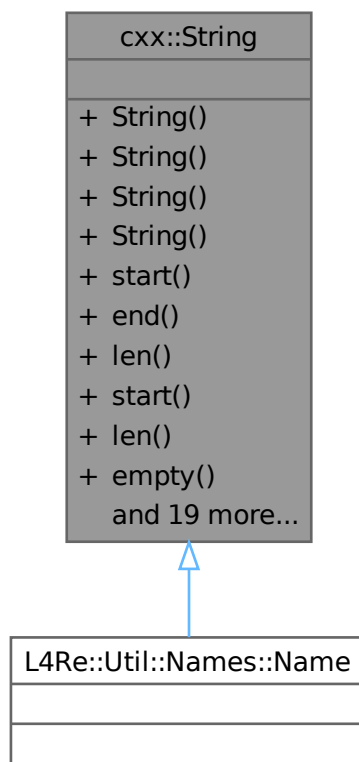
- `I4/cxx/static_vector`

## 16.67 cxx::String Class Reference

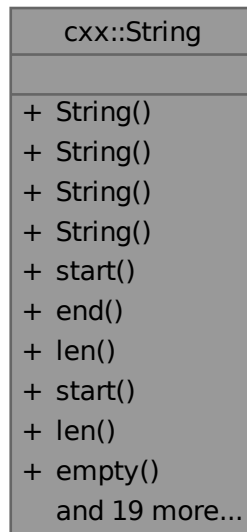
Allocation free string class with explicit length field.

```
#include <string>
```

Inheritance diagram for `cxx::String`:



Collaboration diagram for `cxx::String`:



## Public Types

- typedef char const \* **Index**  
*Character index type.*

## Public Member Functions

- **String** (char const \*s) noexcept  
*Initialize from a zero-terminated string.*
- **String** (char const \*s, unsigned long len) noexcept  
*Initialize from a pointer to first character and a length.*
- **String** (char const \*s, char const \*e) noexcept  
*Initialize with start and end pointer.*
- **String** ()  
*Zero-initialize. Create an invalid string.*
- **Index start** () const  
*Pointer to first character.*
- **Index end** () const  
*Pointer to first byte behind the string.*
- int **len** () const  
*Length.*
- void **start** (char const \*s)  
*Set start.*
- void **len** (unsigned long len)  
*Set length.*
- bool **empty** () const

- Check if the string has length zero.*
- **String head** (**Index end**) const
  - Return prefix up to index.*
- **String head** (unsigned long **end**) const
  - Prefix of length **end**.*
- **String substr** (unsigned long **idx**, unsigned long **len**=~0UL) const
  - Substring of length **len** starting at **idx**.*
- **String substr** (char const \***start**, unsigned long **len**=0) const
  - Substring of length **len** starting at **start**.*
- template<typename F>
  - char const \* **find\_match** (F &&match) const
    - Find matching character. **match** should be a function such as **isspace**.*
- char const \* **find** (char const \*c) const
  - Find character. Return **end()** if not found.*
- char const \* **find** (int c) const
  - Find character. Return **end()** if not found.*
- char const \* **rfind** (char const \*c) const
  - Find right-most character. Return **end()** if not found.*
- **Index starts\_with** (cxx::String const &c) const
  - Check if **c** is a prefix of string.*
- char const \* **find** (int c, char const \*s) const
  - Find character **c** starting at position **s**. Return **end()** if not found.*
- char const \* **find** (char const \*c, char const \*s) const
  - Find character set at position.*
- char const & **operator[]** (unsigned long **idx**) const
  - Get character at **idx**.*
- char const & **operator[]** (int **idx**) const
  - Get character at **idx**.*
- char const & **operator[]** (**Index idx**) const
  - Get character at **idx**.*
- bool **eof** (char const \*s) const
  - Check if pointer **s** points behind string.*
- template<typename INT>
  - int **from\_dec** (INT \*v) const
    - Convert decimal string to integer.*
- template<typename INT>
  - int **from\_hex** (INT \*v) const
    - Convert hex string to integer.*
- bool **operator==** (String const &o) const
  - Equality.*
- bool **operator!=** (String const &o) const
  - Inequality.*

### 16.67.1 Detailed Description

Allocation free string class with explicit length field.

This class is used to group characters of a string which belong to one syntactical token types number, identifier, string, whitespace or another single character.

Stings in this class can contain null bytes and may denote parts of other strings.

#### Examples

[tmpfs/lib/src/fs.cc](#).

Definition at line 30 of file [string](#).



## 16.67.2 Constructor & Destructor Documentation

### 16.67.2.1 String()

```
cxx::String::String (
    char const * s,
    char const * e) [inline], [noexcept]
```

Initialize with start and end pointer.

#### Parameters

<i>s</i>	first character of the string
<i>e</i>	pointer to first byte behind the string

Definition at line 48 of file [string](#).

## 16.67.3 Member Function Documentation

### 16.67.3.1 find()

```
char const * cxx::String::find (
    char const * c,
    char const * s) const [inline]
```

Find character set at position.

#### Parameters

<i>c</i>	zero-terminated string of characters to search for
<i>s</i>	start position of search in string

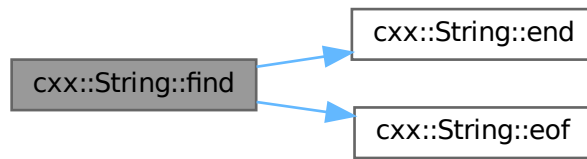
#### Return values

<a href="#">end()</a>	if no char in <i>c</i> is contained in string at or behind <i>s</i> .
<i>position</i>	in string of some character in <i>c</i> .

Definition at line 191 of file [string](#).

References [end\(\)](#), and [eof\(\)](#).

Here is the call graph for this function:



### 16.67.3.2 from\_dec()

```
template<typename INT>
int cxx::String::from_dec (
    INT * v) const [inline]
```

Convert decimal string to integer.

#### Template Parameters

<i>INT</i>	result integer type
------------	---------------------

#### Parameters

out	v	conversion result
-----	---	-------------------

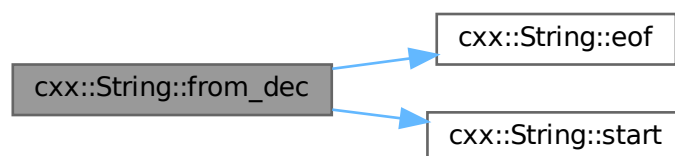
#### Returns

position of first character not converted.

Definition at line 228 of file [string](#).

References [eof\(\)](#), and [start\(\)](#).

Here is the call graph for this function:



### 16.67.3.3 from\_hex()

```
template<typename INT>
int cxx::String::from_hex (
    INT * v) const [inline]
```

Convert hex string to integer.

#### Template Parameters

<i>INT</i>	result integer type
------------	---------------------

#### Parameters

out	<i>v</i>	conversion result
-----	----------	-------------------

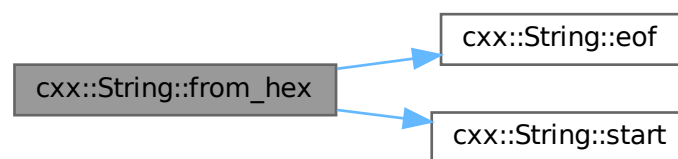
#### Return values

<i>-1</i>	if the maximal amount of digits fitting into <i>INT</i> have been read,
<i>position</i>	of first character not converted otherwise.

Definition at line 257 of file [string](#).

References [eof\(\)](#), and [start\(\)](#).

Here is the call graph for this function:



### 16.67.3.4 starts\_with()

```
Index cxx::String::starts_with (
    cxx::String const & c) const [inline]
```

Check if `c` is a prefix of string.

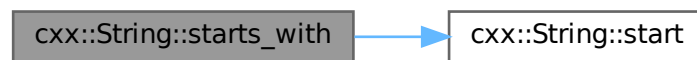
**Returns**

0 if `c` is not a prefix, if it is a prefix, return first position not in `c` (which might be [end\(\)](#)).

Definition at line 155 of file [string](#).

References [start\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

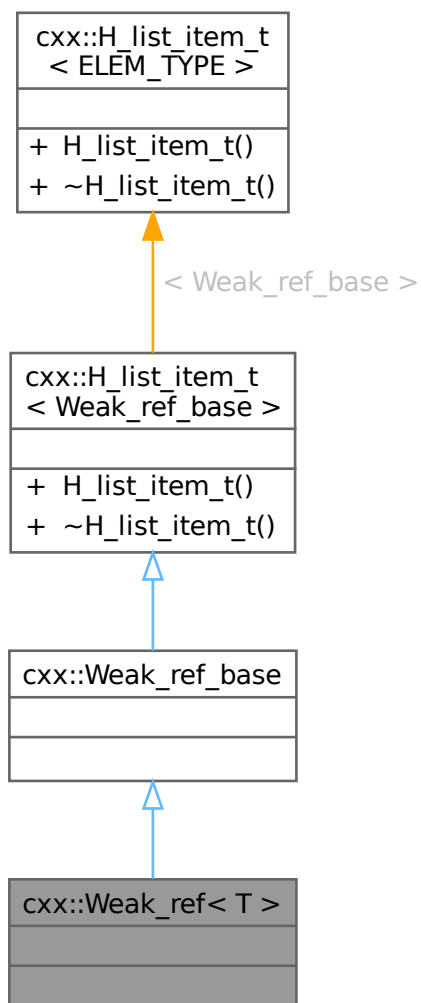
- `I4/cxx/string`

## 16.68 `cxx::Weak_ref< T >` Class Template Reference

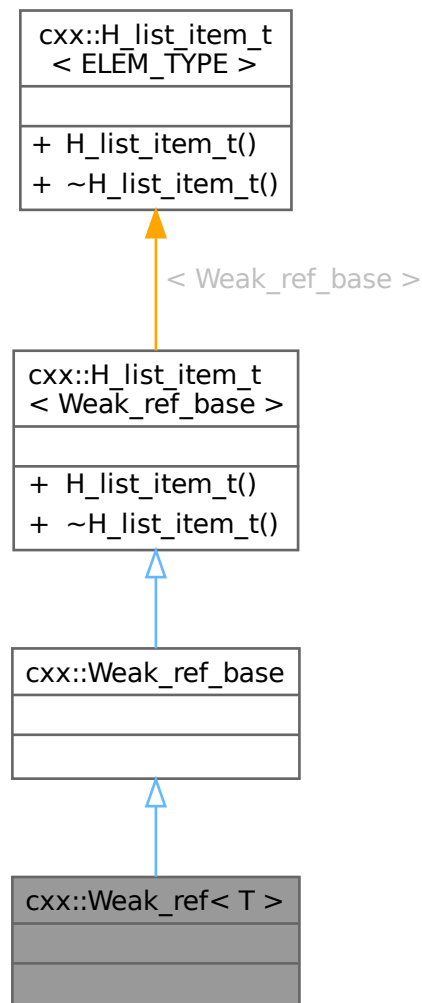
Typed weak reference to an object of type `T`.

```
#include <weak_ref>
```

Inheritance diagram for cxx::Weak\_ref< T >:



Collaboration diagram for `cxx::Weak_ref< T >`:



#### Additional Inherited Members

#### Public Member Functions inherited from `cxx::H_list_item_t< Weak_ref_base >`

- `H_list_item_t()`  
*Constructor.*
- `~H_list_item_t()` noexcept  
*Destructor.*

### 16.68.1 Detailed Description

```
template<typename T>
class cxx::Weak_ref< T >
```

Typed weak reference to an object of type `T`.

## Template Parameters

---

<i>T</i>	The type of the referenced object.
----------	------------------------------------

A weak reference is a reference that is invalidated when the referenced object is about to be deleted. All weak references to an object are kept in a linked list (see [Weak\\_ref\\_base::List](#)) and all the weak references are iterated and reset by the [Weak\\_ref\\_base::List](#) destructor or [Weak\\_ref\\_base::List::reset\(\)](#).

The type *T* must provide two methods that handle the housekeeping of weak references: `remove_weak_ref(Weak_ref_base *)` and `add_weak_ref(Weak_ref_base *)`. These functions must handle the insertion and removal of the weak reference into the respective [Weak\\_ref\\_base::List](#) object. For convenience one can use the `cxx::Weak_ref_obj` as a base class that handles weak references for you.

For example:

```
class C : public cxx::Weak_ref_obj {};

int main()
{
    cxx::Weak_ref<C> r; // r is nullptr
    {
        C c;
        r = &c; // now r points to c
    } // c is destructed, which implies resetting all weak references to c
    // now r is nullptr
    return 0;
}
```

#### Note

Weak references have no effect on the lifetime of the referenced object. Hence, a referenced object is *not* deleted when all weak references for it are gone. If automatic deletion is needed, see [cxx::Ref\\_ptr](#).

Definition at line 95 of file [weak\\_ref](#).

The documentation for this class was generated from the following file:

- `I4/cxx/weak_ref`

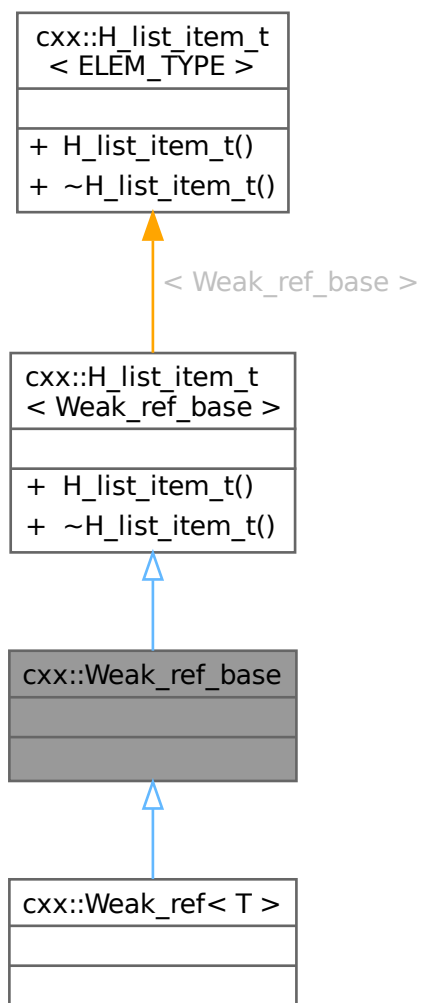
## 16.69 cxx::Weak\_ref\_base Class Reference

Generic (base) weak reference to some object.

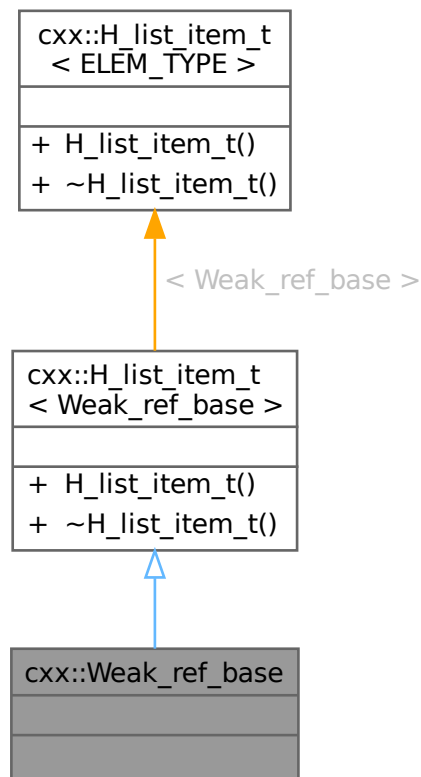
```
#include <weak_ref>
```



Inheritance diagram for cxx::Weak\_ref\_base:



Collaboration diagram for `cxx::Weak_ref_base`:



## Data Structures

- struct [List](#)

*The list type for keeping all weak references to an object.*

## Additional Inherited Members

## Public Member Functions inherited from `cxx::H_list_item_t< Weak_ref_base >`

- [H\\_list\\_item\\_t\(\)](#)  
*Constructor.*
- [~H\\_list\\_item\\_t\(\)](#) noexcept  
*Destructor.*

### 16.69.1 Detailed Description

Generic (base) weak reference to some object.

A weak reference is a reference that gets reset to NULL when the object shall be deleted. All weak references to the same object are kept in a linked list of weak references.

For typed weak references see [cxx::Weak\\_ref](#).

Definition at line 24 of file [weak\\_ref](#).

The documentation for this class was generated from the following file:

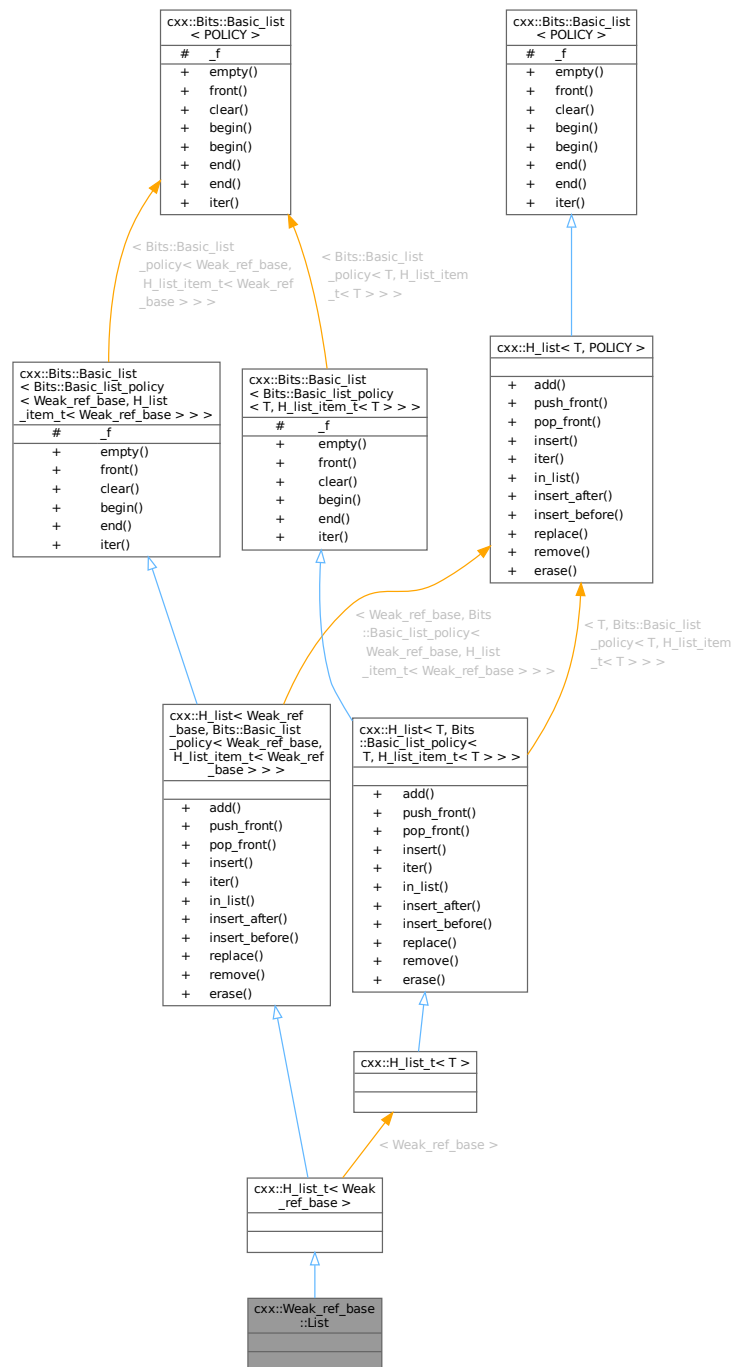
- `I4/cxx/weak_ref`

## 16.70 `cxx::Weak_ref_base::List` Struct Reference

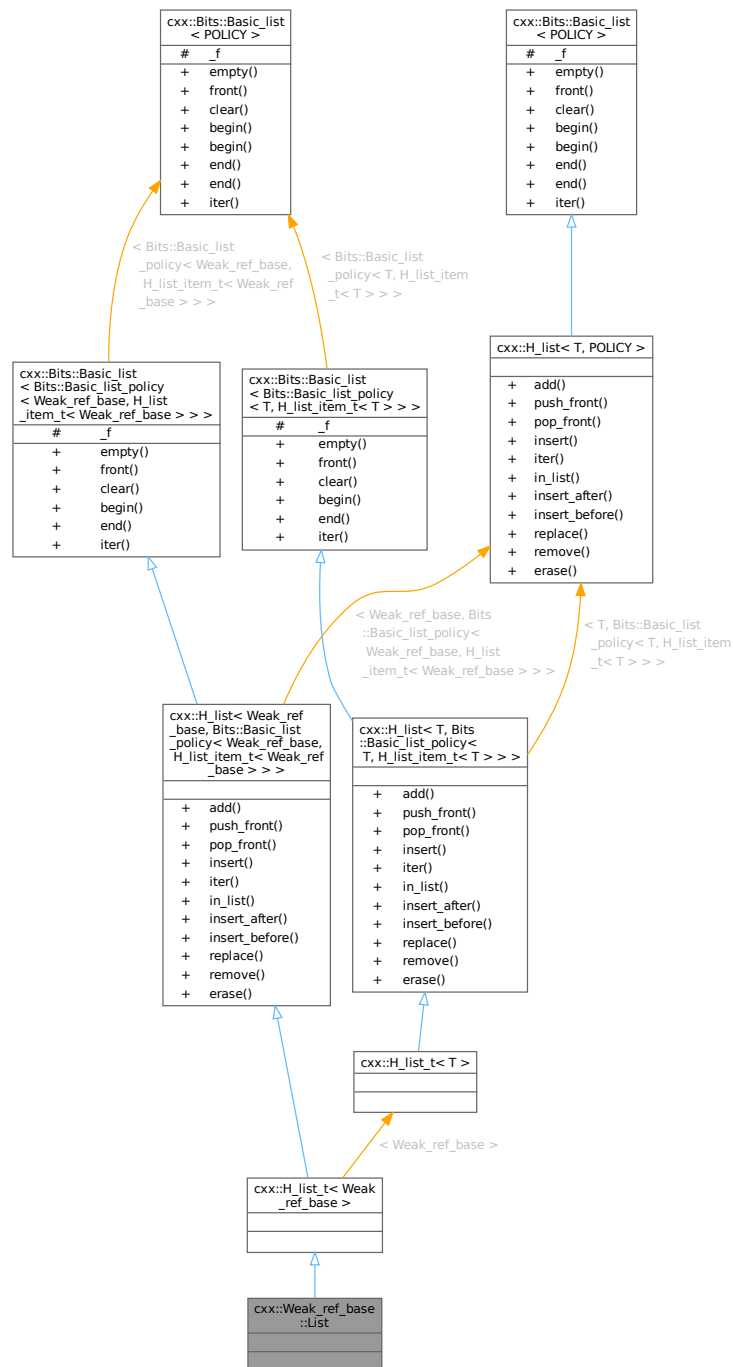
The list type for keeping all weak references to an object.

```
#include <weak_ref>
```

Inheritance diagram for `cxx::Weak_ref_base::List`:



Collaboration diagram for cxx::Weak\_ref\_base::List:



### Additional Inherited Members

### Public Member Functions inherited from

`cxx::H_list< Weak_ref_base, Bits::Basic_list_policy< Weak_ref_base, H_list_item_t< Weak_ref_base >`

- void **add** (`Weak_ref_base *e`)

- *Add element to the front of the list.*
- void **push\_front** ([Weak\\_ref\\_base](#) \*e)  
*Add element to the front of the list.*
- [Weak\\_ref\\_base](#) \* **pop\_front** ()  
*Remove and return the head element of the list.*
- Iterator **insert** ([Weak\\_ref\\_base](#) \*e, Iterator const &pred)  
*Insert an element at the iterator position.*

### Public Member Functions inherited from

[cxx::Bits::Basic\\_list](#)< [Bits::Basic\\_list\\_policy](#)< [Weak\\_ref\\_base](#), [H\\_list\\_item\\_t](#)< [Weak\\_ref\\_base](#) > > >

- bool **empty** () const  
*Check if the list is empty.*
- Value\_type **front** () const  
*Return the first element in the list.*
- void **clear** ()  
*Remove all elements from the list.*
- Iterator **begin** ()  
*Return an iterator to the beginning of the list.*
- Const\_iterator **end** () const  
*Return a const iterator to the end of the list.*

### Static Public Member Functions inherited from

[cxx::H\\_list](#)< [Weak\\_ref\\_base](#), [Bits::Basic\\_list\\_policy](#)< [Weak\\_ref\\_base](#), [H\\_list\\_item\\_t](#)< [Weak\\_ref\\_base](#) > > >

- static Iterator **iter** ([Weak\\_ref\\_base](#) \*c)  
*Return an iterator for an arbitrary list element.*
- static bool **in\_list** ([Weak\\_ref\\_base](#) const \*e)  
*Check if the given element is currently part of a list.*
- static Iterator **insert\_after** ([Weak\\_ref\\_base](#) \*e, Iterator const &pred)  
*Insert an element after the iterator position.*
- static void **insert\_before** ([Weak\\_ref\\_base](#) \*e, Iterator const &succ)  
*Insert an element before the iterator position.*
- static void **replace** ([Weak\\_ref\\_base](#) \*p, [Weak\\_ref\\_base](#) \*e)  
*Replace an element in a list with a new element.*
- static void **remove** ([Weak\\_ref\\_base](#) \*e)  
*Remove the given element from its list.*
- static Iterator **erase** (Iterator const &e)  
*Remove the element at the given iterator position.*

### Static Public Member Functions inherited from

[cxx::Bits::Basic\\_list](#)< [Bits::Basic\\_list\\_policy](#)< [Weak\\_ref\\_base](#), [H\\_list\\_item\\_t](#)< [Weak\\_ref\\_base](#) > > >

- static Const\_iterator **iter** (Const\_value\_type c)  
*Return a const iterator that begins at the given element.*

**Protected Attributes inherited from****`cxx::Bits::Basic_list< Bits::Basic_list_policy< Weak_ref_base, H_list_item_t< Weak_ref_base > > >`**

- `Bits::Basic_list_policy< Weak_ref_base, H_list_item_t< Weak_ref_base > >::Head_type_f`

*Pointer to front of the list.***16.70.1 Detailed Description**

The list type for keeping all weak references to an object.

On destruction of a list, all weak references to the respective object are set to `nullptr`.

Definition at line 38 of file [weak\\_ref](#).

The documentation for this struct was generated from the following file:

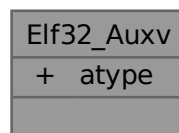
- `l4/cxx/weak_ref`

**16.71 Elf32\_Auxv Struct Reference**

Auxiliary vector (32-bit).

```
#include <elf.h>
```

Collaboration diagram for Elf32\_Auxv:

**Data Fields**

- [Elf32\\_Word atype](#)

**16.71.1 Detailed Description**

Auxiliary vector (32-bit).

Definition at line 962 of file [elf.h](#).

## 16.71.2 Field Documentation

### 16.71.2.1 atype

[Elf32\\_Word](#) `Elf32_Auxv::atype`

See also

[Elf\\_ATs](#)

Definition at line [964](#) of file [elf.h](#).

The documentation for this struct was generated from the following file:

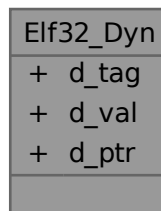
- [l4/util/elf.h](#)

## 16.72 Elf32\_Dyn Struct Reference

ELF32 dynamic entry.

```
#include <elf.h>
```

Collaboration diagram for Elf32\_Dyn:



### Data Fields

- [Elf32\\_Sword](#) `d_tag`

### 16.72.1 Detailed Description

ELF32 dynamic entry.

Definition at line [513](#) of file [elf.h](#).



## 16.72.2 Field Documentation

### 16.72.2.1 d\_tag

[Elf32\\_Sword](#) `Elf32_Dyn::d_tag`

See also

[Elf\\_DTs](#)

Definition at line 515 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

## 16.73 Elf32\_Ehdr Struct Reference

ELF32 header.

```
#include <elf.h>
```

Collaboration diagram for Elf32\_Ehdr:

Elf32_Ehdr
+ e_ident
+ e_type
+ e_machine
+ e_version
+ e_entry
+ e_phoff
+ e_shoff
+ e_flags
+ e_ehsize
+ e_phentsize
+ e_phnum
+ e_shentsize
+ e_shnum
+ e_shstrndx

## Data Fields

- unsigned char **e\_ident** [[EI\\_NIDENT](#)]  
*see [Elf\\_EI](#)s*
- [Elf32\\_Half](#) **e\_type**  
*type of ELF file*
- [Elf32\\_Half](#) **e\_machine**  
*required architecture*
- [Elf32\\_Word](#) **e\_version**  
*file version*
- [Elf32\\_Addr](#) **e\_entry**  
*initial program counter*
- [Elf32\\_Off](#) **e\_phoff**  
*offset of program header table*
- [Elf32\\_Off](#) **e\_shoff**  
*offset of file header table*
- [Elf32\\_Word](#) **e\_flags**  
*processor-specific flags*
- [Elf32\\_Half](#) **e\_ehsize**  
*size of ELF header*
- [Elf32\\_Half](#) **e\_phentsize**  
*size of program header entry*
- [Elf32\\_Half](#) **e\_phnum**  
*number of entries in program header table*
- [Elf32\\_Half](#) **e\_shentsize**  
*size of section header entry*
- [Elf32\\_Half](#) **e\_shnum**  
*number of entries in section header table*
- [Elf32\\_Half](#) **e\_shstrndx**  
*section header table index of strtab*

### 16.73.1 Detailed Description

ELF32 header.

Definition at line [125](#) of file [elf.h](#).

### 16.73.2 Field Documentation

#### 16.73.2.1 e\_flags

[Elf32\\_Word](#) [Elf32\\_Ehdr::e\\_flags](#)

processor-specific flags

See also

[Elf\\_EF\\_ARM\\_s](#)

Definition at line [134](#) of file [elf.h](#).

### 16.73.2.2 e\_machine

`Elf32_Half Elf32_Ehdr::e_machine`

required architecture

See also

[Elf\\_EMs](#)

Definition at line 129 of file [elf.h](#).

### 16.73.2.3 e\_type

`Elf32_Half Elf32_Ehdr::e_type`

type of ELF file

See also

[Elf\\_ETs](#)

Definition at line 128 of file [elf.h](#).

### 16.73.2.4 e\_version

`Elf32_Word Elf32_Ehdr::e_version`

file version

See also

[Elf\\_EVs](#)

Definition at line 130 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

## 16.74 Elf32\_Phdr Struct Reference

ELF32 program header.

```
#include <elf.h>
```

Collaboration diagram for Elf32\_Phdr:

Elf32_Phdr
+ p_type
+ p_offset
+ p_vaddr
+ p_paddr
+ p_filesz
+ p_memsz
+ p_flags
+ p_align

### Data Fields

- [Elf32\\_Word p\\_type](#)  
*type of program section*
- [Elf32\\_Off p\\_offset](#)  
*file offset of program section*
- [Elf32\\_Addr p\\_vaddr](#)  
*memory address of prog section*
- [Elf32\\_Addr p\\_paddr](#)  
*physical address (ignored)*
- [Elf32\\_Word p\\_filesz](#)  
*file size of program section*
- [Elf32\\_Word p\\_memsz](#)  
*memory size of program section*
- [Elf32\\_Word p\\_flags](#)  
*flags*
- [Elf32\\_Word p\\_align](#)  
*alignment of section*

### 16.74.1 Detailed Description

ELF32 program header.

Definition at line 425 of file [elf.h](#).

## 16.74.2 Field Documentation

### 16.74.2.1 p\_flags

`Elf32_Word Elf32_Phdr::p_flags`

flags

See also

[Elf\\_PFs](#)

Definition at line [433](#) of file [elf.h](#).

### 16.74.2.2 p\_type

`Elf32_Word Elf32_Phdr::p_type`

type of program section

See also

[Elf\\_PT](#)

Definition at line [427](#) of file [elf.h](#).

The documentation for this struct was generated from the following file:

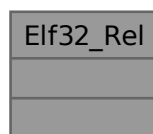
- [l4/util/elf.h](#)

## 16.75 Elf32\_Rel Struct Reference

ELF32 relocation entry w/o addend.

```
#include <elf.h>
```

Collaboration diagram for Elf32\_Rel:



### 16.75.1 Detailed Description

ELF32 relocation entry w/o addend.

Definition at line 631 of file [elf.h](#).

The documentation for this struct was generated from the following file:

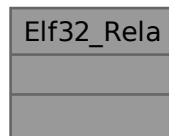
- [l4/util/elf.h](#)

## 16.76 Elf32\_Rela Struct Reference

ELF32 relocation entry w/ addend.

```
#include <elf.h>
```

Collaboration diagram for Elf32\_Rela:



### 16.76.1 Detailed Description

ELF32 relocation entry w/ addend.

Definition at line 638 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

## 16.77 Elf32\_Shdr Struct Reference

ELF32 section header.

```
#include <elf.h>
```

Collaboration diagram for Elf32\_Shdr:

Elf32_Shdr
+ sh_name
+ sh_type
+ sh_flags
+ sh_addr
+ sh_offset
+ sh_size
+ sh_link
+ sh_info
+ sh_addralign
+ sh_entsize

### Data Fields

- [Elf32\\_Word](#) **sh\_name**  
*name of sect (idx into strtab)*
- [Elf32\\_Word](#) **sh\_type**  
*section's type*
- [Elf32\\_Word](#) **sh\_flags**  
*section's flags*
- [Elf32\\_Addr](#) **sh\_addr**  
*memory address of section*
- [Elf32\\_Off](#) **sh\_offset**  
*file offset of section*
- [Elf32\\_Word](#) **sh\_size**  
*file size of section*
- [Elf32\\_Word](#) **sh\_link**  
*idx to associated header section*
- [Elf32\\_Word](#) **sh\_info**  
*extra info of header section*
- [Elf32\\_Word](#) **sh\_addralign**  
*address alignment constraints*
- [Elf32\\_Word](#) **sh\_entsize**  
*size of entry if sect is table*

### 16.77.1 Detailed Description

ELF32 section header.

Definition at line 347 of file [elf.h](#).

### 16.77.2 Field Documentation

#### 16.77.2.1 sh\_flags

[Elf32\\_Word](#) `Elf32_Shdr::sh_flags`

section's flags

See also

[Elf\\_SHFs](#)

Definition at line 351 of file [elf.h](#).

#### 16.77.2.2 sh\_type

[Elf32\\_Word](#) `Elf32_Shdr::sh_type`

section's type

See also

[Elf\\_SHTs](#)

Definition at line 350 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

## 16.78 Elf32\_Sym Struct Reference

ELF32 symbol table entry.

```
#include <elf.h>
```

Collaboration diagram for `Elf32_Sym`:

Elf32_Sym
+ st_name
+ st_value
+ st_size
+ st_info
+ st_other
+ st_shndx



## Data Fields

- [Elf32\\_Word](#) **st\_name**  
*name of symbol (idx symstrtab)*
- [Elf32\\_Addr](#) **st\_value**  
*value of associated symbol*
- [Elf32\\_Word](#) **st\_size**  
*size of associated symbol*
- unsigned char **st\_info**  
*type and binding info*
- unsigned char **st\_other**  
*undefined*
- [Elf32\\_Half](#) **st\_shndx**  
*associated section header*

### 16.78.1 Detailed Description

ELF32 symbol table entry.

Definition at line 871 of file [elf.h](#).

The documentation for this struct was generated from the following file:

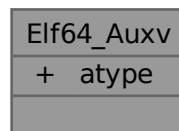
- [l4/util/elf.h](#)

## 16.79 Elf64\_Auxv Struct Reference

Auxiliary vector (64-bit).

```
#include <elf.h>
```

Collaboration diagram for Elf64\_Auxv:



## Data Fields

- [Elf64\\_Word](#) **atype**

### 16.79.1 Detailed Description

Auxiliary vector (64-bit).

Definition at line 969 of file [elf.h](#).

### 16.79.2 Field Documentation

#### 16.79.2.1 atype

[Elf64\\_Word](#) [Elf64\\_Auxv::atype](#)

See also

[Elf\\_ATs](#)

Definition at line 971 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

## 16.80 Elf64\_Dyn Struct Reference

ELF64 dynamic entry.

```
#include <elf.h>
```

Collaboration diagram for Elf64\_Dyn:



#### Data Fields

- [Elf64\\_Sxword](#) [d\\_tag](#)

### 16.80.1 Detailed Description

ELF64 dynamic entry.

Definition at line 524 of file [elf.h](#).

### 16.80.2 Field Documentation

#### 16.80.2.1 d\_tag

`Elf64_Sxword Elf64_Dyn::d_tag`

See also

[Elf\\_DTs](#)

Definition at line 526 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

## 16.81 Elf64\_Ehdr Struct Reference

ELF64 header.

```
#include <elf.h>
```

Collaboration diagram for Elf64\_Ehdr:

Elf64_Ehdr
+ e_ident
+ e_type
+ e_machine
+ e_version
+ e_entry
+ e_phoff
+ e_shoff
+ e_flags
+ e_ehsize
+ e_phentsize
+ e_phnum
+ e_shentsize
+ e_shnum
+ e_shstrndx

## Data Fields

- unsigned char **e\_ident** [[EI\\_NIDENT](#)]  
*see [Elf\\_EI](#)s*
- [Elf64\\_Half](#) **e\_type**  
*type of ELF file*
- [Elf64\\_Half](#) **e\_machine**  
*required architecture*
- [Elf64\\_Word](#) **e\_version**  
*file version*
- [Elf64\\_Addr](#) **e\_entry**  
*initial program counter*
- [Elf64\\_Off](#) **e\_phoff**  
*offset of program header table*
- [Elf64\\_Off](#) **e\_shoff**  
*offset of file header table*
- [Elf64\\_Word](#) **e\_flags**  
*processor-specific flags*
- [Elf64\\_Half](#) **e\_ehsize**  
*size of ELF header*
- [Elf64\\_Half](#) **e\_phentsize**  
*size of program header entry*
- [Elf64\\_Half](#) **e\_phnum**  
*number of entries in program header table*
- [Elf64\\_Half](#) **e\_shentsize**  
*size of section header entry*
- [Elf64\\_Half](#) **e\_shnum**  
*number of entries in section header table*
- [Elf64\\_Half](#) **e\_shstrndx**  
*section header table index of strtab*

### 16.81.1 Detailed Description

ELF64 header.

Definition at line [146](#) of file [elf.h](#).

### 16.81.2 Field Documentation

#### 16.81.2.1 e\_flags

[Elf64\\_Word](#) [Elf64\\_Ehdr::e\\_flags](#)

processor-specific flags

See also

[Elf\\_EF\\_ARM\\_s](#)

Definition at line [155](#) of file [elf.h](#).

### 16.81.2.2 e\_machine

`Elf64_Half Elf64_Ehdr::e_machine`

required architecture

See also

[Elf\\_EMs](#)

Definition at line 150 of file [elf.h](#).

### 16.81.2.3 e\_type

`Elf64_Half Elf64_Ehdr::e_type`

type of ELF file

See also

[Elf\\_ETs](#)

Definition at line 149 of file [elf.h](#).

### 16.81.2.4 e\_version

`Elf64_Word Elf64_Ehdr::e_version`

file version

See also

[Elf\\_EVs](#)

Definition at line 151 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

## 16.82 Elf64\_Phdr Struct Reference

ELF64 program header.

```
#include <elf.h>
```

Collaboration diagram for Elf64\_Phdr:

Elf64_Phdr
+ p_type
+ p_flags
+ p_offset
+ p_vaddr
+ p_paddr
+ p_filesz
+ p_memsz
+ p_align

### Data Fields

- [Elf64\\_Word p\\_type](#)  
*type of program section*
- [Elf64\\_Word p\\_flags](#)  
*flags*
- [Elf64\\_Off p\\_offset](#)  
*file offset of program section*
- [Elf64\\_Addr p\\_vaddr](#)  
*memory address of prog section*
- [Elf64\\_Addr p\\_paddr](#)  
*physical address (ignored)*
- [Elf64\\_Xword p\\_filesz](#)  
*file size of program section*
- [Elf64\\_Xword p\\_memsz](#)  
*memory size of program section*
- [Elf64\\_Xword p\\_align](#)  
*alignment of section*

### 16.82.1 Detailed Description

ELF64 program header.

Definition at line 438 of file [elf.h](#).

## 16.82.2 Field Documentation

### 16.82.2.1 p\_flags

`Elf64_Word Elf64_Phdr::p_flags`

flags

See also

`Elf_PFs`

Definition at line 441 of file [elf.h](#).

### 16.82.2.2 p\_type

`Elf64_Word Elf64_Phdr::p_type`

type of program section

See also

[Elf\\_PT](#)s

Definition at line 440 of file [elf.h](#).

The documentation for this struct was generated from the following file:

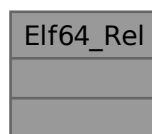
- [l4/util/elf.h](#)

## 16.83 Elf64\_Rel Struct Reference

ELF64 relocation entry w/o addend.

```
#include <elf.h>
```

Collaboration diagram for Elf64\_Rel:



### 16.83.1 Detailed Description

ELF64 relocation entry w/o addend.

Definition at line [646](#) of file [elf.h](#).

The documentation for this struct was generated from the following file:

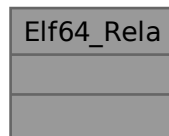
- [l4/util/elf.h](#)

## 16.84 Elf64\_Rela Struct Reference

ELF64 relocation entry w/ addend.

```
#include <elf.h>
```

Collaboration diagram for Elf64\_Rela:



### 16.84.1 Detailed Description

ELF64 relocation entry w/ addend.

Definition at line [653](#) of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)



## 16.85 Elf64\_Shdr Struct Reference

ELF64 section header.

```
#include <elf.h>
```

Collaboration diagram for Elf64\_Shdr:

Elf64_Shdr
+ sh_name
+ sh_type
+ sh_flags
+ sh_addr
+ sh_offset
+ sh_size
+ sh_link
+ sh_info
+ sh_addralign
+ sh_entsize

### Data Fields

- [Elf64\\_Word](#) **sh\_name**  
*name of sect (idx into strtab)*
- [Elf64\\_Word](#) **sh\_type**  
*section's type*
- [Elf64\\_Xword](#) **sh\_flags**  
*section's flags*
- [Elf64\\_Addr](#) **sh\_addr**  
*memory address of section*
- [Elf64\\_Off](#) **sh\_offset**  
*file offset of section*
- [Elf64\\_Xword](#) **sh\_size**  
*file size of section*
- [Elf64\\_Word](#) **sh\_link**  
*idx to associated header section*
- [Elf64\\_Word](#) **sh\_info**  
*extra info of header section*
- [Elf64\\_Xword](#) **sh\_addralign**  
*address alignment constraints*
- [Elf64\\_Xword](#) **sh\_entsize**  
*size of entry if sect is table*

### 16.85.1 Detailed Description

ELF64 section header.

Definition at line 362 of file [elf.h](#).

### 16.85.2 Field Documentation

#### 16.85.2.1 sh\_flags

[Elf64\\_Xword](#) Elf64\_Shdr::sh\_flags

section's flags

See also

[Elf\\_SHFs](#)

Definition at line 366 of file [elf.h](#).

#### 16.85.2.2 sh\_type

[Elf64\\_Word](#) Elf64\_Shdr::sh\_type

section's type

See also

[Elf\\_SHTs](#)

Definition at line 365 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

## 16.86 Elf64\_Sym Struct Reference

ELF64 symbol table entry.

```
#include <elf.h>
```

Collaboration diagram for Elf64\_Sym:

Elf64_Sym
+ st_name
+ st_info
+ st_other
+ st_shndx
+ st_value
+ st_size

## Data Fields

- [Elf64\\_Word](#) **st\_name**  
*name of symbol (idx symstrtab)*
- unsigned char **st\_info**  
*type and binding info*
- unsigned char **st\_other**  
*undefined*
- [Elf64\\_Half](#) **st\_shndx**  
*associated section header*
- [Elf64\\_Addr](#) **st\_value**  
*value of associated symbol*
- [Elf64\\_Xword](#) **st\_size**  
*size of associated symbol*

### 16.86.1 Detailed Description

ELF64 symbol table entry.

Definition at line 882 of file [elf.h](#).

The documentation for this struct was generated from the following file:

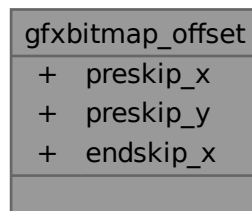
- [l4/util/elf.h](#)

## 16.87 gfxbitmap\_offset Struct Reference

offsets in pmap[] and bmap[]

```
#include <bitmap.h>
```

Collaboration diagram for gfxbitmap\_offset:



## Data Fields

- [l4\\_uint32\\_t](#) **preskip\_x**  
*skip pixels at beginning of line*
- [l4\\_uint32\\_t](#) **preskip\_y**  
*skip lines*
- [l4\\_uint32\\_t](#) **endskip\_x**  
*skip pixels at end of line*

### 16.87.1 Detailed Description

offsets in `pmap[]` and `bmap[]`

Definition at line 68 of file [bitmap.h](#).

The documentation for this struct was generated from the following file:

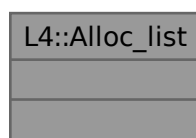
- [l4/libgfxbitmap/bitmap.h](#)

## 16.88 L4::Alloc\_list Class Reference

A simple list-based allocator.

```
#include <alloc.h>
```

Collaboration diagram for L4::Alloc\_list:



### 16.88.1 Detailed Description

A simple list-based allocator.

Definition at line 20 of file [alloc.h](#).

The documentation for this class was generated from the following file:

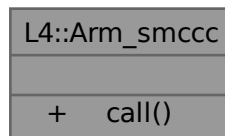
- [l4/cxx/alloc.h](#)

## 16.89 L4::Arm\_smccc Class Reference

Wrapper for function calls that follow the ARM SMC/HVC calling convention.

```
#include <arm_smccc>
```

Collaboration diagram for L4::Arm\_smccc:



### Public Member Functions

- [l4\\_msgtag\\_t](#) [call](#) ([l4\\_umword\\_t](#) func, [l4\\_umword\\_t](#) in0, [l4\\_umword\\_t](#) in1, [l4\\_umword\\_t](#) in2, [l4\\_umword\\_t](#) in3, [l4\\_umword\\_t](#) in4, [l4\\_umword\\_t](#) in5, [l4\\_umword\\_t](#) \*out0, [l4\\_umword\\_t](#) \*out1, [l4\\_umword\\_t](#) \*out2, [l4\\_umword\\_t](#) \*out3, [l4\\_umword\\_t](#) client\_id)

*ARM SMC/HVC function call.*

### 16.89.1 Detailed Description

Wrapper for function calls that follow the ARM SMC/HVC calling convention.

See [l4\\_arm\\_smccc\\_call\(\)](#) for the corresponding C interface.

Definition at line 23 of file [arm\\_smccc](#).

### 16.89.2 Member Function Documentation

#### 16.89.2.1 call()

```

l4_msgtag_t L4::Arm_smccc::call (
    l4_umword_t func,
    l4_umword_t in0,
    l4_umword_t in1,
    l4_umword_t in2,
    l4_umword_t in3,
    l4_umword_t in4,
    l4_umword_t in5,
    l4_umword_t * out0,
    l4_umword_t * out1,
    l4_umword_t * out2,
    l4_umword_t * out3,
    l4_umword_t client_id)
  
```

ARM SMC/HVC function call.

The input parameters consist of a function identifier, 6 arguments and a client id. Results are returned in 4 output parameters.

#### Parameters

	<i>func</i>	Function identifier. <ul style="list-style-type: none"> <li>• Bit 31 has to be set: This marks the call as <i>Fast Call</i>. <i>Yielding Calls</i> (bit 31 unset) are rejected by the kernel.</li> <li>• Bit 30 defines the calling convention:</li> <li>• Bit 30 == 1: 64-bit calling convention.</li> <li>• Bit 30 == 0: 32-bit calling convention.</li> <li>• Bits 24..29 determine the service call ID. The permitted IDs are set in the kernel configuration. By default only service IDs <math>\geq 0x30000000</math> (<i>Trusted Application Calls</i> and <i>Trusted OS Calls</i>) are allowed.</li> </ul>
in	<i>in0</i>	First input parameter.
in	<i>in1</i>	Second input parameter.
in	<i>in2</i>	Third input parameter.
in	<i>in3</i>	Fourth input parameter.
in	<i>in4</i>	Fifth input parameter.
in	<i>in5</i>	Sixth input parameter.
out	<i>out0</i>	First output parameter.
out	<i>out1</i>	Second output parameter.
out	<i>out2</i>	Third output parameter.
out	<i>out3</i>	Fourth output parameter.
in	<i>client↔ _id</i>	Client ID. According to the specification, this value might be ignored by certain functions.

### Return values

<code>-L4_ENOSYS</code>	Either bit 31 of the function call not set or service ID outside the range permitted by kernel configuration.
<code>-L4_EINVAL</code>	Invalid number of parameters.
<code>&lt; 0</code>	Other <a href="#">L4</a> error.
<code>0</code>	Success.

The documentation for this class was generated from the following file:

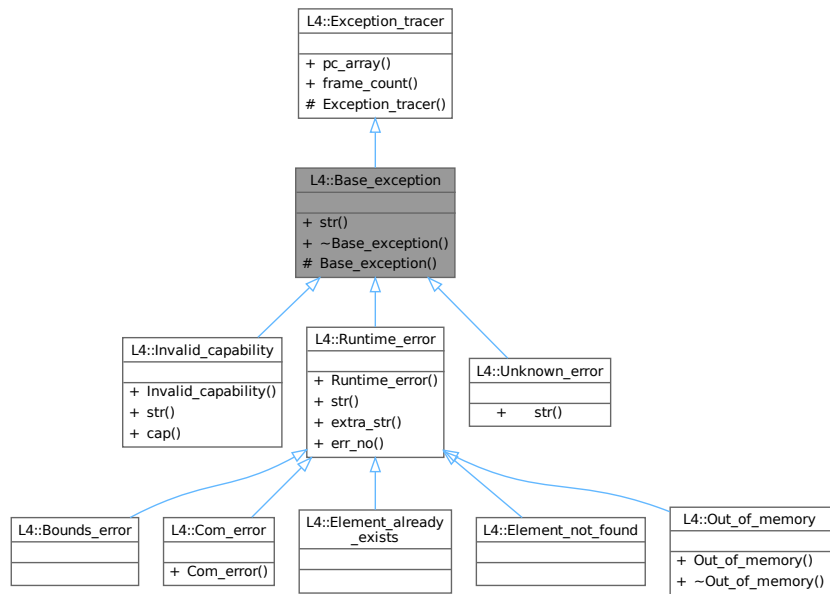
- `l4/sys/arm_smccc`

## 16.90 L4::Base\_exception Class Reference

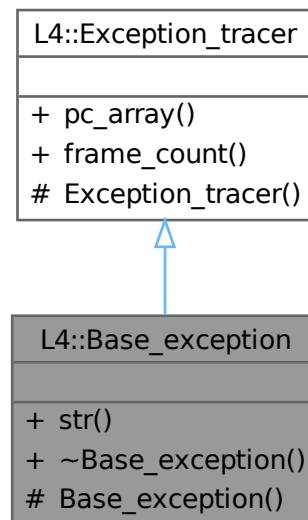
Base class for all exceptions, thrown by the [L4Re](#) framework.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Base\_exception:



Collaboration diagram for L4::Base\_exception:



## Public Member Functions

- virtual char const \* **str** () const noexcept=0

*Return a human readable string for the exception.*

- virtual `~Base_exception ()` noexcept

*Destruction.*

### Public Member Functions inherited from [L4::Exception\\_tracer](#)

- void const \*const \* `pc_array ()` const noexcept

*Get the array containing the call trace.*

- int `frame_count ()` const noexcept

*Get the number of entries that are valid in the call trace.*

### Protected Member Functions

- `Base_exception ()` noexcept

*Create a base exception.*

### Protected Member Functions inherited from [L4::Exception\\_tracer](#)

- `Exception_tracer ()` noexcept

*Create a back trace.*

## 16.90.1 Detailed Description

Base class for all exceptions, thrown by the [L4Re](#) framework.

This is the abstract base of all exceptions thrown within the [L4Re](#) framework. It is basically also a good idea to use it as base of all user defined exceptions.

Definition at line [105](#) of file [exceptions](#).

The documentation for this class was generated from the following file:

- [l4/cxx/exceptions](#)

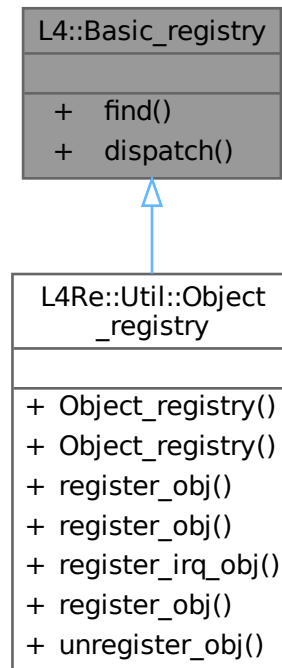


## 16.91 L4::Basic\_registry Class Reference

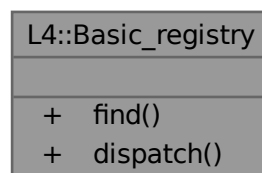
This registry returns the corresponding server object based on the label of an [lpc\\_gate](#).

```
#include <ipc_epiface>
```

Inheritance diagram for L4::Basic\_registry:



Collaboration diagram for L4::Basic\_registry:



## Static Public Member Functions

- static [Value](#) \* [find](#) ([l4\\_umword\\_t](#) label)  
*Get the server object for an [lpc\\_gate](#) label.*
- static [l4\\_msgtag\\_t](#) [dispatch](#) ([l4\\_msgtag\\_t](#) tag, [l4\\_umword\\_t](#) label, [l4\\_utcb\\_t](#) \*utcb)  
*The dispatch function called by the server loop.*

### 16.91.1 Detailed Description

This registry returns the corresponding server object based on the label of an [lpc\\_gate](#).

Definition at line 529 of file [ipc\\_epiface](#).

### 16.91.2 Member Function Documentation

#### 16.91.2.1 dispatch()

```
l4\_msgtag\_t L4::Basic_registry::dispatch (
    l4\_msgtag\_t tag,
    l4\_umword\_t label,
    l4\_utcb\_t * utcb) [inline], [static]
```

The dispatch function called by the server loop.

This function forwards the message to the server object identified by the given *label*.

#### Parameters

<i>tag</i>	The message tag used for the invocation.
<i>label</i>	The label used to find the object including the rights bits of the invoked capability.
<i>utcb</i>	The UTCB used for the invocation.

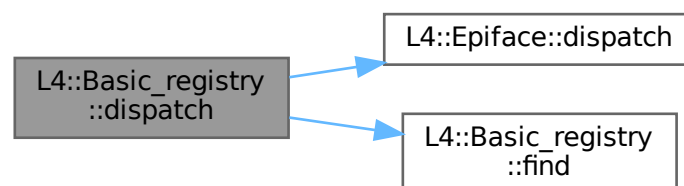
#### Returns

The return code from the object's dispatch function or -L4\_ENOENT if the object does not exist.

Definition at line 554 of file [ipc\\_epiface](#).

References [L4::Epiface::dispatch\(\)](#), and [find\(\)](#).

Here is the call graph for this function:



### 16.91.2.2 find()

```
Value * L4::Basic_registry::find (  
    l4_umword_t label) [inline], [static]
```

Get the server object for an [lpc\\_gate](#) label.

#### Parameters

<i>label</i>	The label usually stored in an <a href="#">lpc_gate</a> .
--------------	-----------------------------------------------------------

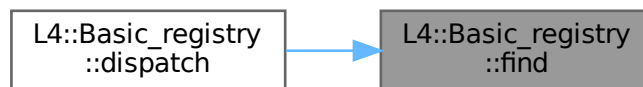
#### Returns

A pointer to the [Epiface](#) identified by the given label.

Definition at line 538 of file [ipc\\_epiface](#).

Referenced by [dispatch\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

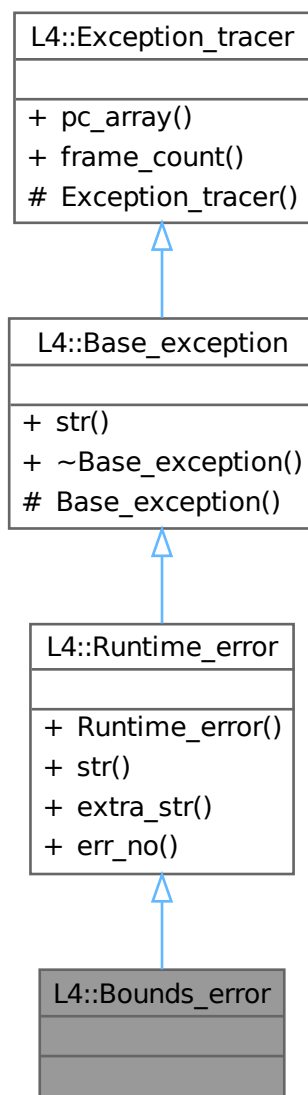
- `l4/sys/cxx/ipc_epiface`

## 16.92 L4::Bounds\_error Class Reference

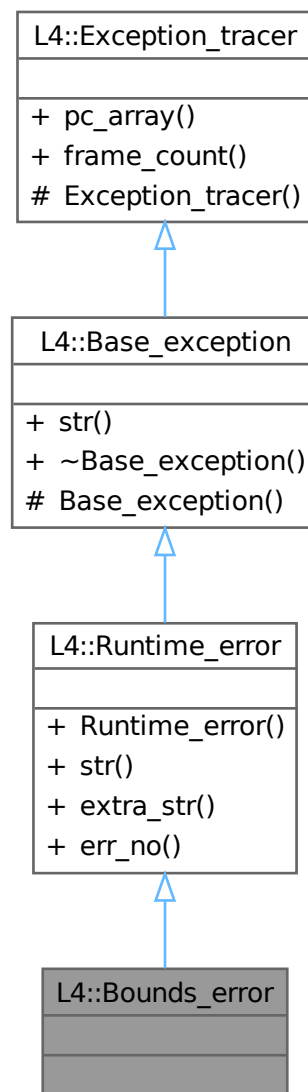
Access out of bounds.

```
#include <exceptions>
```

Inheritance diagram for L4::Bounds\_error:



Collaboration diagram for L4::Bounds\_error:



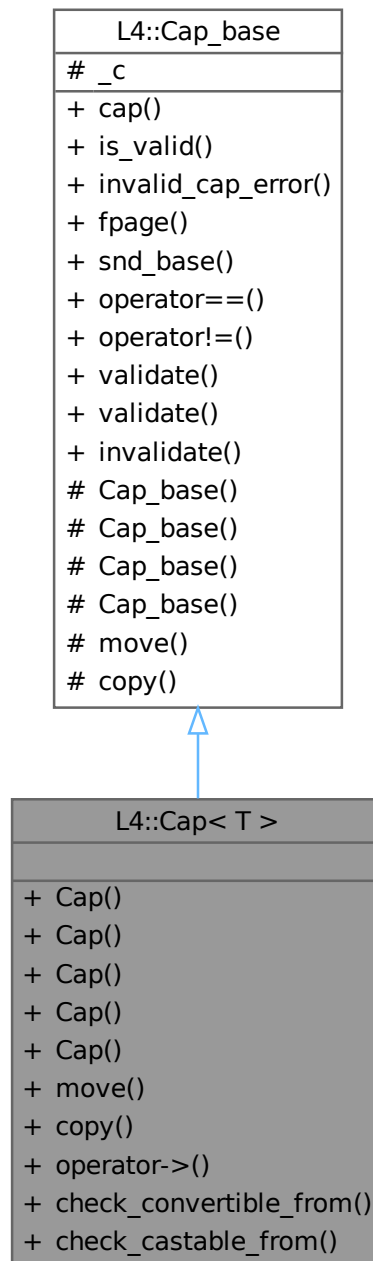
### Additional Inherited Members

### Public Member Functions inherited from [L4::Runtime\\_error](#)

- [Runtime\\_error](#) (long [err\\_no](#), char const \*extra=0) noexcept  
Create a new [Runtime\\_error](#).
- char const \* [str](#) () const noexcept override  
Return a human readable string for the exception.
- char const \* [extra\\_str](#) () const  
Get the description text for this runtime error.
- long [err\\_no](#) () const noexcept  
Get the error value for this runtime error.



Collaboration diagram for L4::Cap< T >:



## Public Member Functions

- `template<typename O>`  
`Cap (Cap< O > const &o) noexcept`  
*Create a copy from o, supporting implicit type casting.*
- `Cap (Cap_type cap) noexcept`  
*Constructor to create an invalid capability selector.*

- [Cap](#) ([l4\\_default\\_caps\\_t cap](#)) noexcept  
*Initialize capability with one of the default capability selectors.*
- [Cap](#) ([l4\\_cap\\_idx\\_t idx=L4\\_INVALID\\_CAP](#)) noexcept  
*Initialize capability, defaults to the invalid capability selector.*
- [Cap](#) ([No\\_init\\_type](#)) noexcept  
*Create an uninitialized cap selector.*
- [Cap move](#) ([Cap](#) const &src) const  
*Move a capability to this cap slot.*
- [Cap copy](#) ([Cap](#) const &src) const  
*Copy a capability to this cap slot.*
- [T](#) \* **operator->** () const noexcept  
*Member access of a T.*

## Public Member Functions inherited from [L4::Cap\\_base](#)

- [l4\\_cap\\_idx\\_t cap](#) () const noexcept  
*Return capability selector.*
- bool [is\\_valid](#) () const noexcept  
*Test whether the capability is a valid capability index (i.e., not L4\_INVALID\_CAP).*
- int [invalid\\_cap\\_error](#) () const noexcept  
*Return the transported error code in an invalid capability index.*
- [l4\\_fpage\\_t fpage](#) (unsigned rights=[L4\\_CAP\\_FPAGE\\_RWS](#)) const noexcept  
*Return flexpage for the capability.*
- [l4\\_umword\\_t snd\\_base](#) (unsigned grant=[L4\\_MAP\\_ITEM\\_MAP](#), [l4\\_cap\\_idx\\_t base=L4\\_INVALID\\_CAP](#)) const noexcept  
*Return send base.*
- bool **operator==** ([Cap\\_base](#) const &o) const noexcept  
*Test if two capabilities are equal.*
- bool **operator!=** ([Cap\\_base](#) const &o) const noexcept  
*Test if two capabilities are not equal.*
- [l4\\_msgtag\\_t validate](#) ([l4\\_utcb\\_t \\*u=l4\\_utcb\(\)](#)) const noexcept  
*Check whether a capability is present (refers to an object).*
- [l4\\_msgtag\\_t validate](#) ([Cap](#)< [Task](#) > task, [l4\\_utcb\\_t \\*u=l4\\_utcb\(\)](#)) const noexcept  
*Check whether a capability is present (refers to an object).*
- void [invalidate](#) () noexcept  
*Set this capability to invalid (L4\_INVALID\_CAP).*

## Static Public Member Functions

- template<typename From>  
static void [check\\_convertible\\_from](#) () noexcept  
*Perform the type conversion that needs to compile in order for a capability of type From to be convertible to one of type T.*
- template<typename From>  
static void [check\\_castable\\_from](#) () noexcept  
*Perform the type conversion that needs to compile in order for a capability of type From to be castable (via the correct cap\_cast) to one of type T.*

## Friends

- class [L4::Kobject](#)



## Additional Inherited Members

### Public Types inherited from L4::Cap\_base

- enum [No\\_init\\_type](#) { [No\\_init](#) }  
*Special value for uninitialized capability objects.*
- enum [Cap\\_type](#) { [Invalid](#) = L4\_INVALID\_CAP }  
*Invalid capability type.*

### Protected Member Functions inherited from L4::Cap\_base

- [Cap\\_base](#) ([l4\\_cap\\_idx\\_t](#) c) noexcept  
*Generate a capability from its C representation.*
- [Cap\\_base](#) ([Cap\\_type](#) cap) noexcept  
*Constructor to create an invalid capability.*
- [Cap\\_base](#) ([l4\\_default\\_caps\\_t](#) cap) noexcept  
*Initialize capability with one of the default capabilities.*
- [Cap\\_base](#) () noexcept  
*Create an uninitialized instance.*
- void [move](#) ([Cap\\_base](#) const &src) const  
*Replace this capability with the contents of `src`.*
- void [copy](#) ([Cap\\_base](#) const &src) const  
*Copy a capability.*

### Protected Attributes inherited from L4::Cap\_base

- [l4\\_cap\\_idx\\_t\\_c](#)  
*The C representation of a capability selector.*

## 16.93.1 Detailed Description

```
template<typename T>
class L4::Cap< T >
```

C++ interface for capabilities.

### Template Parameters

<a href="#">T</a>	Type of the object the capability points to.
-------------------	----------------------------------------------

The C++ version of a capability is comparable to a pointer, in fact it is a kind of smart pointer for our kernel objects and the objects derived from the kernel objects ([L4::Kobject](#)).

Add

```
#include <l4/sys/capability>
```

to your code to use the capability interface.

### Examples

[examples/clntsrv/src/client.cc](#), [examples/libs/l4re/c++/mem\\_alloc/ma+rm.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#), [examples/libs/l4re/streammap/client.cc](#), and [examples/sys/migrate/thread\\_migrate](#)

Definition at line 223 of file [capability.h](#).

## 16.93.2 Constructor & Destructor Documentation

### 16.93.2.1 Cap() [1/4]

```
template<typename T>
template<typename O>
L4::Cap< T >::Cap (
    Cap< O > const & o) [inline], [noexcept]
```

Create a copy from `o`, supporting implicit type casting.

#### Parameters

<code>o</code>	The source selector that shall be copied (and casted).
----------------	--------------------------------------------------------

Definition at line 275 of file [capability.h](#).

### 16.93.2.2 Cap() [2/4]

```
template<typename T>
L4::Cap< T >::Cap (
    Cap_type cap) [inline], [noexcept]
```

Constructor to create an invalid capability selector.

#### Parameters

<code>cap</code>	Capability selector.
------------------	----------------------

Definition at line 282 of file [capability.h](#).

### 16.93.2.3 Cap() [3/4]

```
template<typename T>
L4::Cap< T >::Cap (
    l4_default_caps_t cap) [inline], [noexcept]
```

Initialize capability with one of the default capability selectors.

#### Parameters

<code>cap</code>	Capability selector.
------------------	----------------------

Definition at line 288 of file [capability.h](#).

### 16.93.2.4 Cap() [4/4]

```
template<typename T>
L4::Cap< T >::Cap (
    l4_cap_idx_t idx = L4_INVALID_CAP) [inline], [explicit], [noexcept]
```

Initialize capability, defaults to the invalid capability selector.

#### Parameters

---

<i>idx</i>	Capability selector.
------------	----------------------

Definition at line 294 of file [capability.h](#).

## 16.93.3 Member Function Documentation

### 16.93.3.1 check\_castable\_from()

```
template<typename T>
template<typename From>
void L4::Cap< T >::check_castable_from () [inline], [static], [noexcept]
```

Perform the type conversion that needs to compile in order for a capability of type From to be castable (via the correct `cap_cast`) to one of type T.

#### Template Parameters

<i>From</i>	Type to convert from
-------------	----------------------

Definition at line 264 of file [capability.h](#).

### 16.93.3.2 check\_convertible\_from()

```
template<typename T>
template<typename From>
void L4::Cap< T >::check_convertible_from () [inline], [static], [noexcept]
```

Perform the type conversion that needs to compile in order for a capability of type From to be convertible to one of type T.

#### Template Parameters

<i>From</i>	Type to convert from
-------------	----------------------

Definition at line 251 of file [capability.h](#).

### 16.93.3.3 copy()

```
template<typename T>
Cap L4::Cap< T >::copy (
    Cap< T > const & src) const [inline]
```

Copy a capability to this cap slot.

#### Parameters

---

<i>src</i>	the source capability slot.
------------	-----------------------------

Definition at line 317 of file [capability.h](#).

### 16.93.3.4 move()

```
template<typename T>
Cap L4::Cap< T >::move (
    Cap< T > const & src) const [inline]
```

Move a capability to this cap slot.

#### Parameters

<i>src</i>	the source capability slot.
------------	-----------------------------

After this operation the source slot is no longer valid.

Definition at line 307 of file [capability.h](#).

The documentation for this class was generated from the following file:

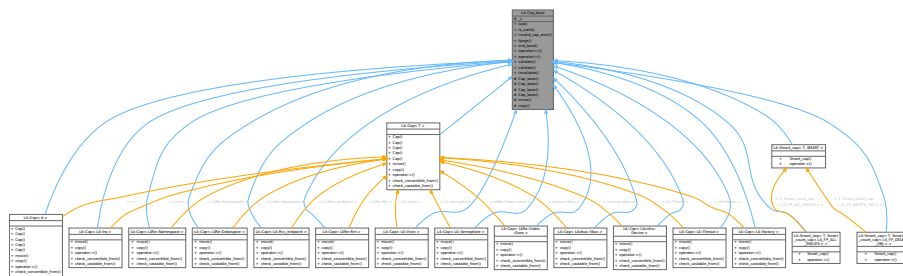
- l4/sys/cxx/capability.h

## 16.94 L4::Cap\_base Class Reference

Base class for all kinds of capabilities.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Cap\_base:



Collaboration diagram for L4::Cap\_base:

L4::Cap_base
# _c
+ cap()
+ is_valid()
+ invalid_cap_error()
+ fpage()
+ snd_base()
+ operator==()
+ operator!=()
+ validate()
+ validate()
+ invalidate()
# Cap_base()
# Cap_base()
# Cap_base()
# Cap_base()
# move()
# copy()

## Public Types

- enum [No\\_init\\_type](#) { [No\\_init](#) }  
*Special value for uninitialized capability objects.*
- enum [Cap\\_type](#) { [Invalid](#) = L4\_INVALID\_CAP }  
*Invalid capability type.*

## Public Member Functions

- [l4\\_cap\\_idx\\_t](#) [cap](#) () const noexcept  
*Return capability selector.*
- bool [is\\_valid](#) () const noexcept  
*Test whether the capability is a valid capability index (i.e., not L4\_INVALID\_CAP).*
- int [invalid\\_cap\\_error](#) () const noexcept  
*Return the transported error code in an invalid capability index.*
- [l4\\_fpage\\_t](#) [fpage](#) (unsigned rights=[L4\\_CAP\\_FPAGE\\_RWS](#)) const noexcept  
*Return flexpage for the capability.*
- [l4\\_umword\\_t](#) [snd\\_base](#) (unsigned grant=[L4\\_MAP\\_ITEM\\_MAP](#), [l4\\_cap\\_idx\\_t](#) base=[L4\\_INVALID\\_CAP](#)) const noexcept  
*Return send base.*

- bool **operator==** ([Cap\\_base](#) const &o) const noexcept  
*Test if two capabilities are equal.*
- bool **operator!=** ([Cap\\_base](#) const &o) const noexcept  
*Test if two capabilities are not equal.*
- [l4\\_msgtag\\_t validate](#) ([l4\\_utcb\\_t](#) \*u=[l4\\_utcb](#)()) const noexcept  
*Check whether a capability is present (refers to an object).*
- [l4\\_msgtag\\_t validate](#) ([Cap](#)< [Task](#) > task, [l4\\_utcb\\_t](#) \*u=[l4\\_utcb](#)()) const noexcept  
*Check whether a capability is present (refers to an object).*
- void **invalidate** () noexcept  
*Set this capability to invalid ([L4\\_INVALID\\_CAP](#)).*

### Protected Member Functions

- [Cap\\_base](#) ([l4\\_cap\\_idx\\_t](#) c) noexcept  
*Generate a capability from its C representation.*
- **Cap\_base** ([Cap\\_type](#) cap) noexcept  
*Constructor to create an invalid capability.*
- [Cap\\_base](#) ([l4\\_default\\_caps\\_t](#) cap) noexcept  
*Initialize capability with one of the default capabilities.*
- **Cap\_base** () noexcept  
*Create an uninitialized instance.*
- void [move](#) ([Cap\\_base](#) const &src) const  
*Replace this capability with the contents of *src*.*
- void [copy](#) ([Cap\\_base](#) const &src) const  
*Copy a capability.*

### Protected Attributes

- [l4\\_cap\\_idx\\_t \\_c](#)  
*The C representation of a capability selector.*

## 16.94.1 Detailed Description

Base class for all kinds of capabilities.

#### Attention

This class is not for direct use, use [L4::Cap](#) instead.

This class contains all the things that are independent of the type of the object referred by the capability.

#### See also

[L4::Cap](#) for typed capabilities.

Definition at line 25 of file [capability.h](#).

## 16.94.2 Member Enumeration Documentation

### 16.94.2.1 Cap\_type

```
enum L4::Cap\_base::Cap\_type
```

Invalid capability type.

#### Enumerator

Invalid	Invalid capability selector.
---------	------------------------------

Definition at line 40 of file [capability.h](#).

### 16.94.2.2 No\_init\_type

```
enum L4::Cap_base::No_init_type
```

Special value for uninitialized capability objects.

#### Enumerator

No_init	Special value for constructing uninitialized <a href="#">Cap</a> objects.
---------	---------------------------------------------------------------------------

Definition at line 29 of file [capability.h](#).

## 16.94.3 Constructor & Destructor Documentation

### 16.94.3.1 Cap\_base() [1/2]

```
L4::Cap_base::Cap_base (
    l4_cap_idx_t c) [inline], [explicit], [protected], [noexcept]
```

Generate a capability from its C representation.

#### Parameters

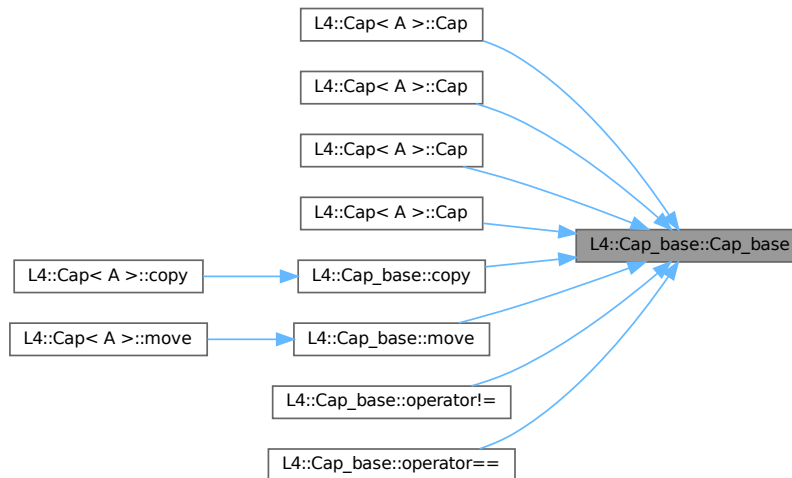
<i>c</i>	The C capability
----------	------------------

Definition at line 149 of file [capability.h](#).

References [\\_c](#).

Referenced by [L4::Cap< A >::Cap\(\)](#), [L4::Cap< A >::Cap\(\)](#), [L4::Cap< A >::Cap\(\)](#), [L4::Cap< A >::Cap\(\)](#), [copy\(\)](#), [move\(\)](#), [operator!=\(\)](#), and [operator==\(\)](#).

Here is the caller graph for this function:



### 16.94.3.2 Cap\_base() [2/2]

```
L4::Cap_base::Cap_base (
    l4_default_caps_t cap) [inline], [explicit], [protected], [noexcept]
```

Initialize capability with one of the default capabilities.

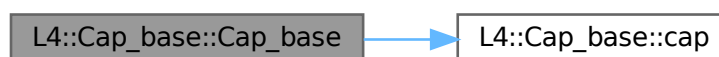
#### Parameters

<i>cap</i>	Capability.
------------	-------------

Definition at line 160 of file [capability.h](#).

References [\\_c](#), and [cap\(\)](#).

Here is the call graph for this function:





## 16.94.4 Member Function Documentation

### 16.94.4.1 cap()

```
l4_cap_idx_t L4::Cap_base::cap () const [inline], [noexcept]
```

Return capability selector.

#### Returns

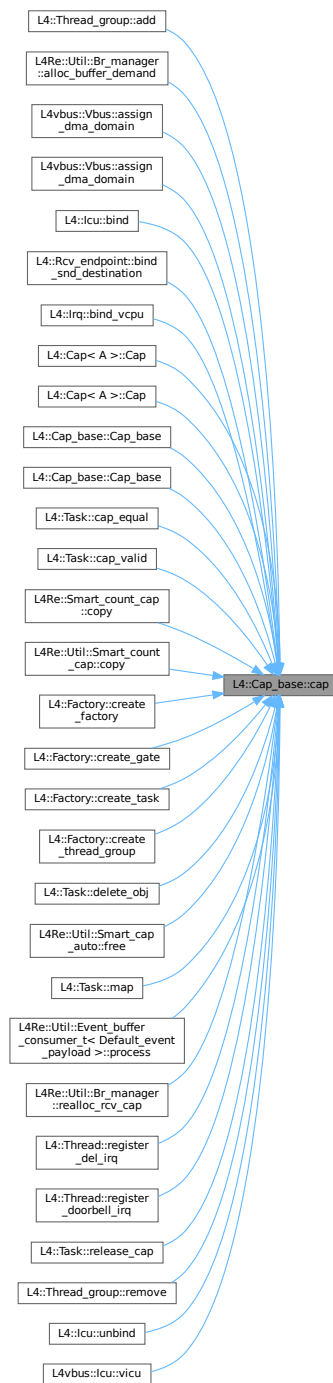
Capability selector.

Definition at line 49 of file [capability.h](#).

References [\\_c](#).

Referenced by [L4::Thread\\_group::add\(\)](#), [L4Re::Util::Br\\_manager::alloc\\_buffer\\_demand\(\)](#), [L4vbus::Vbus::assign\\_dma\\_domain\(\)](#), [L4vbus::Vbus::assign\\_dma\\_domain\(\)](#), [L4::lcu::bind\(\)](#), [L4::Rcv\\_endpoint::bind\\_snd\\_destination\(\)](#), [L4::Irq::bind\\_vcpu\(\)](#), [L4::Cap< A >::Cap\(\)](#), [L4::Cap< A >::Cap\(\)](#), [Cap\\_base\(\)](#), [Cap\\_base\(\)](#), [L4::Task::cap\\_equal\(\)](#), [L4::Task::cap\\_valid\(\)](#), [L4Re::Smart\\_count\\_cap< Unmap\\_flags >::copy\(\)](#), [L4Re::Util::Smart\\_count\\_cap< Unmap\\_flags >::copy\(\)](#), [L4::Factory::create\\_factory\(\)](#), [L4::Factory::create\\_gate\(\)](#), [L4::Factory::create\\_task\(\)](#), [L4::Factory::create\\_thread\\_group\(\)](#), [L4::Task::delete\\_obj\(\)](#), [L4Re::Util::Smart\\_cap\\_auto< Unmap\\_flags >::free\(\)](#), [L4::Task::map\(\)](#), [L4Re::Util::Event\\_buffer\\_consumer\\_t< T >::map\(\)](#), [L4Re::Util::Br\\_manager::realloc\\_rcv\\_cap\(\)](#), [L4::Thread::register\\_del\\_irq\(\)](#), [L4::Thread::register\\_doorbell\\_irq\(\)](#), [L4::Task::release\\_cap\(\)](#), [L4::Thread\\_group::remove\(\)](#), [L4::lcu::unbind\(\)](#), and [L4vbus::lcu::vicu\(\)](#).

Here is the caller graph for this function:



### 16.94.4.2 copy()

```
void L4::Cap_base::copy (
    Cap_base const & src) const [inline], [protected]
```

Copy a capability.

## Parameters

---

<i>src</i>	the source capability.
------------	------------------------

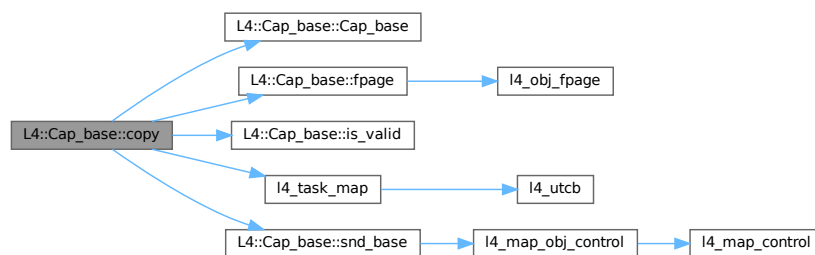
After this operation this capability refers to the same object as `src`.

Definition at line 192 of file `capability.h`.

References `Cap_base()`, `fpage()`, `is_valid()`, `L4_BASE_TASK_CAP`, `L4_CAP_FPAGE_RWSD`, `L4_FPAGE_C_OBJ_RIGHTS`, `l4_task_map()`, and `snd_base()`.

Referenced by `L4::Cap< A >::copy()`.

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.94.4.3 fpage()

```

l4_fpage_t L4::Cap_base::fpage (
    unsigned rights = L4_CAP_FPAGE_RWS) const [inline], [noexcept]
  
```

Return flexpage for the capability.

#### Parameters

<i>rights</i>	Rights, defaults to 'rws'
---------------	---------------------------

**Returns**

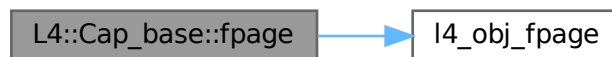
flexpage

Definition at line 74 of file [capability.h](#).

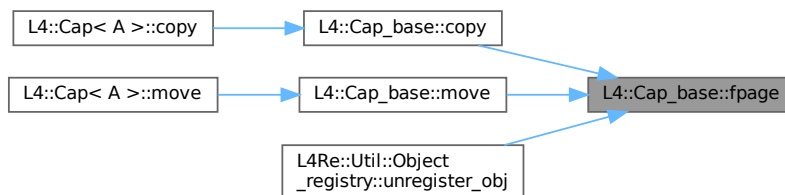
References [\\_c](#), [L4\\_CAP\\_FPAGE\\_RWS](#), and [l4\\_obj\\_fpage\(\)](#).

Referenced by [copy\(\)](#), [move\(\)](#), and [L4Re::Util::Object\\_registry::unregister\\_obj\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**16.94.4.4 is\_valid()**

```
bool L4::Cap_base::is_valid () const [inline], [noexcept]
```

Test whether the capability is a valid capability index (i.e., not `L4_INVALID_CAP`).

**Returns**

True if capability is not invalid, false if invalid

**Examples**

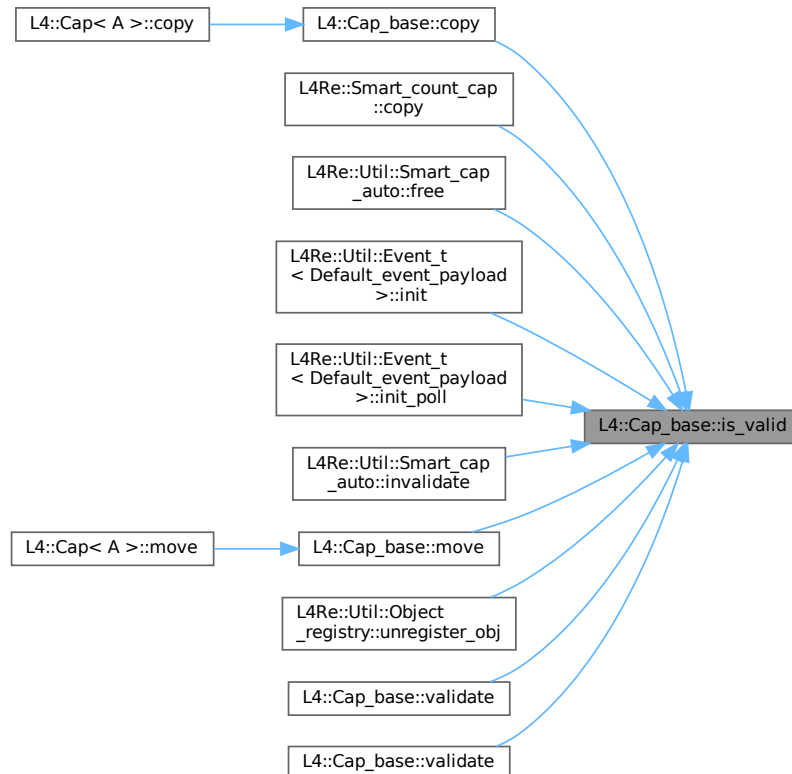
[examples/libs/l4re/c++/mem\\_alloc/ma+rm.cc](#), and [examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#).

Definition at line 57 of file [capability.h](#).

References [\\_c](#).

Referenced by [copy\(\)](#), [L4Re::Smart\\_count\\_cap< Unmap\\_flags >::copy\(\)](#), [L4Re::Util::Smart\\_cap\\_auto< Unmap\\_flags >::free\(\)](#), [L4Re::Util::Event\\_t< Default\\_event\\_payload >::init\(\)](#), [L4Re::Util::Event\\_t< Default\\_event\\_payload >::init\\_poll\(\)](#), [L4Re::Util::Smart\\_cap\\_auto< Unmap\\_flags >::invalidate\(\)](#), [move\(\)](#), [L4Re::Util::Object\\_registry::unregister\\_obj\(\)](#), [validate\(\)](#), and [validate\(\)](#).

Here is the caller graph for this function:



#### 16.94.4.5 move()

```
void L4::Cap_base::move (
    Cap_base const & src) const [inline], [protected]
```

Replace this capability with the contents of `src`.

##### Parameters

<code>src</code>	the source capability.
------------------	------------------------

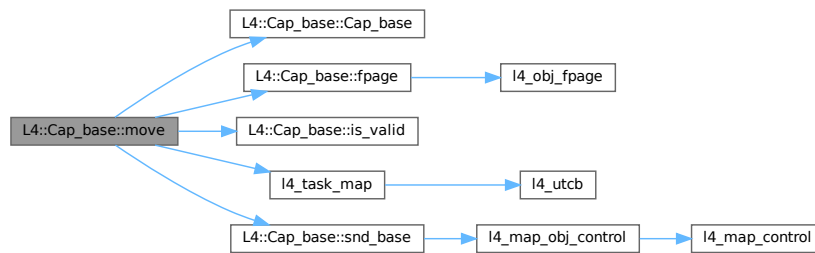
After the operation this capability refers to the object formerly referred to by the source capability `src`, and the source capability no longer refers to an object.

Definition at line 176 of file [capability.h](#).

References [Cap\\_base\(\)](#), [fpage\(\)](#), [is\\_valid\(\)](#), [L4\\_BASE\\_TASK\\_CAP](#), [L4\\_CAP\\_FPAGE\\_RWSD](#), [L4\\_FPAGE\\_C\\_OBJ\\_RIGHTS](#), [L4\\_MAP\\_ITEM\\_GRANT](#), [l4\\_task\\_map\(\)](#), and [snd\\_base\(\)](#).

Referenced by [L4::Cap< A >::move\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 16.94.4.6 snd\_base()

```

l4_umword_t L4::Cap_base::snd_base (
    unsigned grant = L4_MAP_ITEM_MAP,
    l4_cap_idx_t base = L4_INVALID_CAP) const [inline], [noexcept]
  
```

Return send base.

##### Parameters

<i>grant</i>	Indicates if object shall be granted. Allowed values: <a href="#">L4_MAP_ITEM_MAP</a> , <a href="#">L4_MAP_ITEM_GRANT</a> .
<i>base</i>	Base capability (first in a bundle of aligned capabilities)

**Returns**

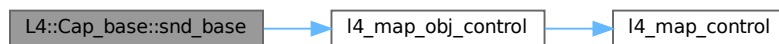
Map object.

Definition at line 86 of file [capability.h](#).

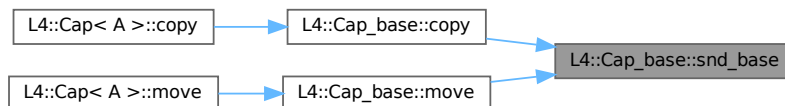
References [\\_c](#), [L4\\_INVALID\\_CAP](#), [L4\\_MAP\\_ITEM\\_MAP](#), and [l4\\_map\\_obj\\_control\(\)](#).

Referenced by [copy\(\)](#), and [move\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**16.94.4.7 validate() [1/2]**

```

l4_msgtag_t L4::Cap_base::validate (
    Cap< Task > task,
    l4_utcb_t * u = l4_utcb()) const [inline], [noexcept]
  
```

Check whether a capability is present (refers to an object).

**Parameters**

<i>task</i>	<a href="#">Task</a> to check the capability in.
<i>u</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

**Return values**

<code>l4_msgtag_t::label() &gt; 0</code>	Capability is present (refers to an object).
<code>l4_msgtag_t::label() == 0</code>	No capability present (void object or invalid capability slot).

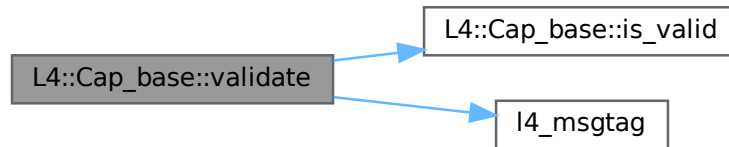
A capability is considered present when it refers to an existing kernel object.



Definition at line 74 of file [capability](#).

References [\\_c](#), [is\\_valid\(\)](#), and [l4\\_msgtag\(\)](#).

Here is the call graph for this function:



#### 16.94.4.8 validate() [2/2]

```
l4_msgtag_t L4::Cap_base::validate (
    l4_utcb_t * u = l4_utcb()) const [inline], [noexcept]
```

Check whether a capability is present (refers to an object).

##### Parameters

<i>u</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .
----------	------------------------------------------------------------------------------------------------------------

##### Return values

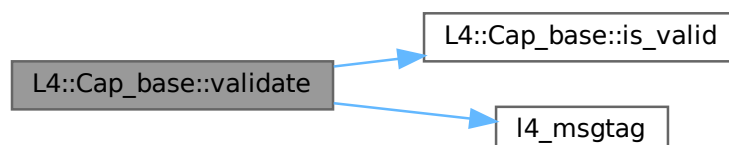
<i>l4_msgtag_t::label()</i> > 0	Capability is present (refers to an object).
<i>l4_msgtag_t::label()</i> == 0	No capability present (void object or invalid capability slot).

A capability is considered present when it refers to an existing kernel object.

Definition at line 81 of file [capability](#).

References [\\_c](#), [is\\_valid\(\)](#), [L4\\_BASE\\_TASK\\_CAP](#), and [l4\\_msgtag\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

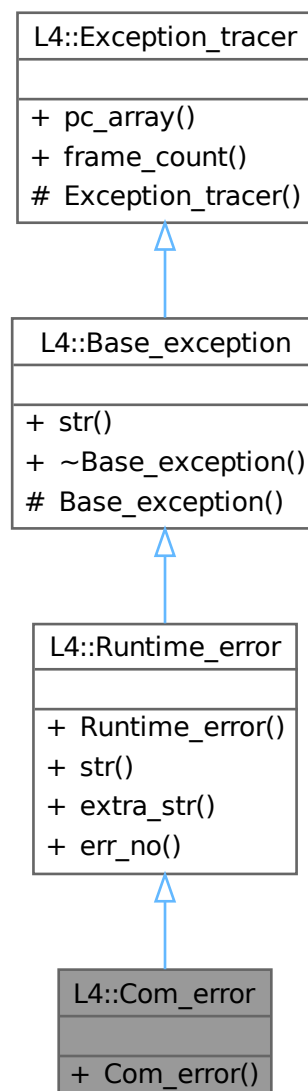
- [l4/sys/cxx/capability.h](#)
- [l4/sys/capability](#)

## 16.95 L4::Com\_error Class Reference

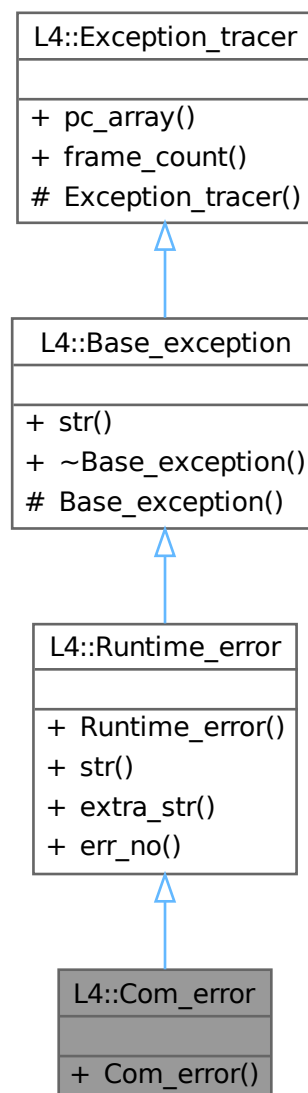
Error conditions during IPC.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Com\_error:



Collaboration diagram for L4::Com\_error:



### Public Member Functions

- [Com\\_error](#) (long err) noexcept  
Create a [Com\\_error](#) for the given [L4](#) IPC error code.

### Public Member Functions inherited from [L4::Runtime\\_error](#)

- [Runtime\\_error](#) (long [err\\_no](#), char const \*extra=0) noexcept  
Create a new [Runtime\\_error](#).
- char const \* [str](#) () const noexcept override

*Return a human readable string for the exception.*

- char const \* [extra\\_str](#) () const

*Get the description text for this runtime error.*

- long [err\\_no](#) () const noexcept

*Get the error value for this runtime error.*

## Public Member Functions inherited from [L4::Base\\_exception](#)

- virtual ~**Base\_exception** () noexcept

*Destruction.*

## Public Member Functions inherited from [L4::Exception\\_tracer](#)

- void const \*const \* **pc\_array** () const noexcept

*Get the array containing the call trace.*

- int **frame\_count** () const noexcept

*Get the number of entries that are valid in the call trace.*

## Additional Inherited Members

## Protected Member Functions inherited from [L4::Base\\_exception](#)

- **Base\_exception** () noexcept

*Create a base exception.*

## Protected Member Functions inherited from [L4::Exception\\_tracer](#)

- **Exception\_tracer** () noexcept

*Create a back trace.*

## 16.95.1 Detailed Description

Error conditions during IPC.

This exception encapsulates all IPC error conditions of [L4](#) IPC.

Definition at line [263](#) of file [exceptions](#).

## 16.95.2 Constructor & Destructor Documentation

### 16.95.2.1 Com\_error()

```
L4::Com_error::Com_error (
    long err) [inline], [explicit], [noexcept]
```

Create a [Com\\_error](#) for the given [L4](#) IPC error code.

#### Parameters

---

<i>err</i>	The <a href="#">L4</a> IPC error code (l4_ipc... return value).
------------	-----------------------------------------------------------------

Definition at line 270 of file [exceptions](#).

The documentation for this class was generated from the following file:

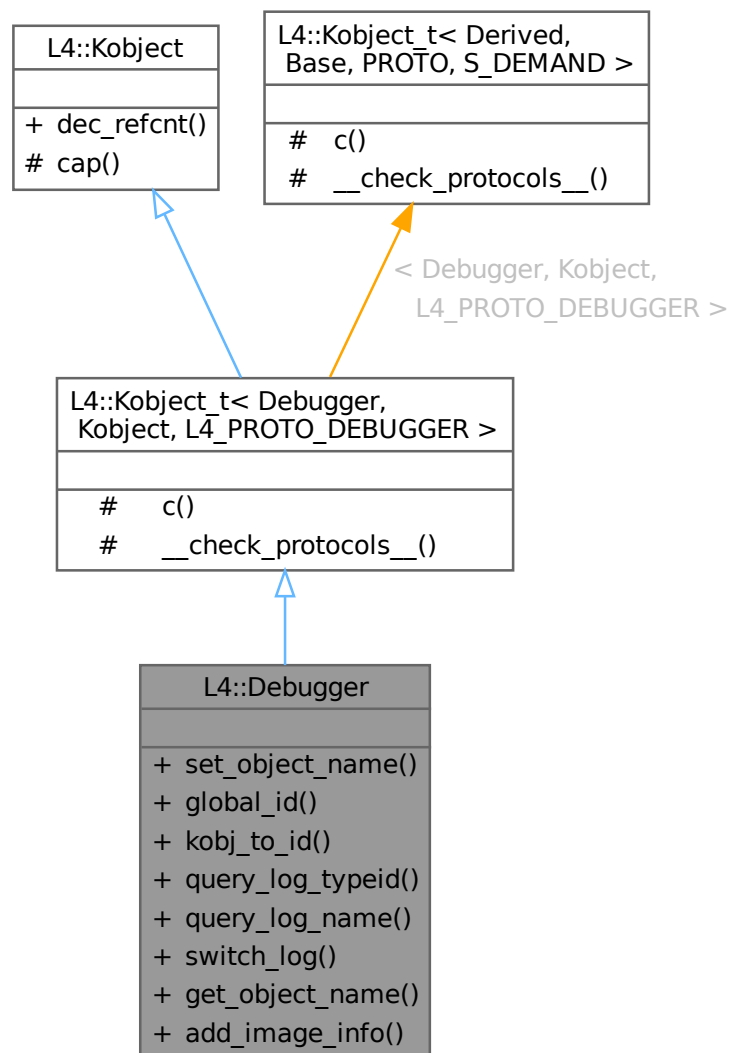
- [l4/cxx/exceptions](#)

## 16.96 L4::Debugger Class Reference

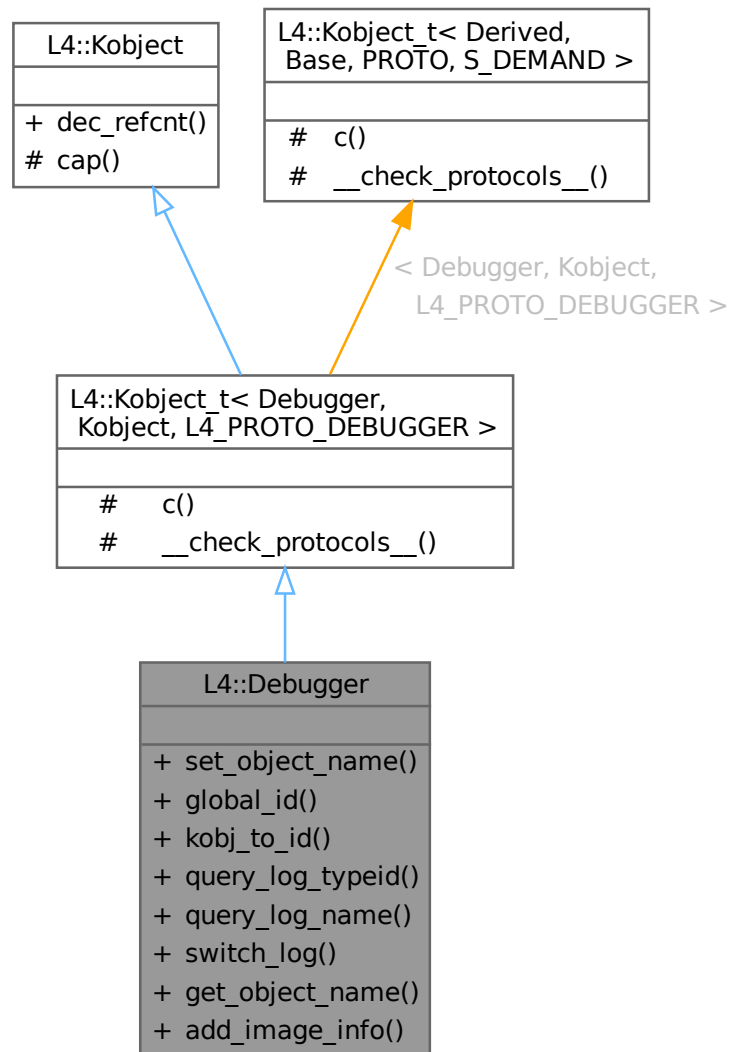
C++ kernel debugger API.

```
#include <debugger>
```

Inheritance diagram for L4::Debugger:



Collaboration diagram for L4::Debugger:



## Public Member Functions

- [l4\\_msgtag\\_t set\\_object\\_name](#) (const char \*name, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Set the name of a kernel object.*
- unsigned long [global\\_id](#) ([l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Get the globally unique ID of the object behind a capability.*
- unsigned long [kobj\\_to\\_id](#) ([l4\\_addr\\_t](#) kobjp, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Get the globally unique ID of the object behind the kobject pointer.*
- long [query\\_log\\_typeid](#) (const char \*name, unsigned idx, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Query the log-id for a log type.*
- long [query\\_log\\_name](#) (unsigned idx, char \*name, unsigned namelen, char \*shortname, unsigned short-namelen, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept

*Query the name of a log type given the ID.*

- [l4\\_msgtag\\_t switch\\_log](#) (const char \*name, unsigned on\_off, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept

*Set or unset log.*

- [l4\\_msgtag\\_t get\\_object\\_name](#) (unsigned id, char \*name, unsigned size, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept

*Get name of object with Id *i* d.*

- [l4\\_msgtag\\_t add\\_image\\_info](#) ([l4\\_addr\\_t](#) base, const char \*name, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept

*Add loaded image information for a task.*

## Public Member Functions inherited from [L4::Kobject](#)

- [l4\\_msgtag\\_t dec\\_refcnt](#) ([l4\\_mword\\_t](#) diff, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#))

*Decrement the in kernel reference counter for the object.*

## Additional Inherited Members

## Protected Types inherited from

[L4::Kobject\\_t](#)< [Debugger](#), [Kobject](#), [L4\\_PROTO\\_DEBUGGER](#) >

- typedef [Debugger](#) **Class**

*The target interface type (inheriting from [Kobject\\_t](#)).*

- typedef Typeid::Iface< [PROTO](#), [Debugger](#) > **\_\_iface**

*The interface description for the derived class.*

- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_iface** >, typename [Kobject](#)::\_\_iface\_list > **\_\_iface\_list**

*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Member Functions inherited from

[L4::Kobject\\_t](#)< [Debugger](#), [Kobject](#), [L4\\_PROTO\\_DEBUGGER](#) >

- [L4::Cap](#)< **Class** > **c** () const noexcept

*Get the capability to ourselves.*

## Protected Member Functions inherited from [L4::Kobject](#)

- [l4\\_cap\\_idx\\_t cap](#) () const noexcept

*Return capability selector.*

## Static Protected Member Functions inherited from

[L4::Kobject\\_t](#)< [Debugger](#), [Kobject](#), [L4\\_PROTO\\_DEBUGGER](#) >

- static void **\_\_check\_protocols** () noexcept

*Helper to check for protocol conflicts.*

## 16.96.1 Detailed Description

C++ kernel debugger API.

### Attention

This API is subject to change! Do not rely on it in production code.

This API is to be used for debugging exclusively.

This is the API for accessing kernel-debugger functionality from user-level programs. Specifically, it provides functionality to enrich the kernel debugger with insights into the program. The purpose is to facilitate debugging with the kernel debugger. For instance, a developer might choose to name the threads of her program so that she can find them in the kernel debugger thread list.

This API interacts with a kernel object that interfaces with the kernel debugger, the jdb-kernel object. The jdb-kernel object is fix and only available when the kernel debugger is built into the microkernel. The developer needs to pass the capability through to her program.

### Include File

```
#include <l4/sys/debugger>
```

Definition at line 42 of file [debugger](#).

## 16.96.2 Member Function Documentation

### 16.96.2.1 add\_image\_info()

```
l4_msgtag_t L4::Debugger::add_image_info (
    l4_addr_t base,
    const char * name,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Add loaded image information for a task.

### Parameters

<i>base</i>	Image load base address
<i>name</i>	Image name
<i>utcb</i>	The UTCB to use for the operation.

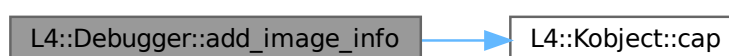
### Returns

System call return tag.

Definition at line 161 of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:





### 16.96.2.2 get\_object\_name()

```
l4_msgtag_t L4::Debugger::get_object_name (
    unsigned id,
    char * name,
    unsigned size,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Get name of object with Id `id`.

#### Parameters

	<i>id</i>	Id of the object whose name is asked.
out	<i>name</i>	Buffer to copy the name into. The buffer must be allocated by the caller.
	<i>size</i>	Length of the <code>name</code> buffer.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

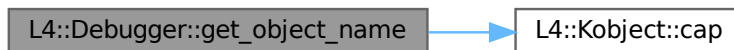
#### Returns

Syscall return tag

Definition at line 148 of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 16.96.2.3 global\_id()

```
unsigned long L4::Debugger::global_id (
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Get the globally unique ID of the object behind a capability.

#### Parameters

<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .
-------------	--------------------------------------------------------------------------------------------------------------------

#### Return values

$\sim 0UL$	The capability is invalid.
$\geq 0$	The global debugger id.

Definition at line 71 of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



#### 16.96.2.4 kobj\_to\_id()

```

unsigned long L4::Debugger::kobj_to_id (
    l4_addr_t kobjp,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Get the globally unique ID of the object behind the kobject pointer.

##### Parameters

<i>kobjp</i>	<a href="#">Kobject</a> pointer
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

##### Return values

$\sim 0UL$	The capability or the <a href="#">Kobject</a> pointer are invalid.
$\geq 0$	The globally unique id.

Definition at line 83 of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 16.96.2.5 query\_log\_name()

```
long L4::Debugger::query_log_name (
    unsigned idx,
    char * name,
    unsigned namelen,
    char * shortname,
    unsigned shortnamelen,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Query the name of a log type given the ID.

#### Parameters

	<i>idx</i>	ID to query.
out	<i>name</i>	Buffer to copy name to. The buffer must be allocated by the caller.
	<i>namelen</i>	Buffer length of name.
out	<i>shortname</i>	Buffer to copy shortname to. The buffer must be allocated by the caller.
	<i>shortnamelen</i>	Buffer length of shortname.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

#### Return values

0	Success
<0	Error

Definition at line 116 of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 16.96.2.6 query\_log\_typeid()

```
long L4::Debugger::query_log_typeid (
    const char * name,
    unsigned idx,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Query the log-id for a log type.

#### Parameters

<i>name</i>	Name to query for.
<i>idx</i>	Idx to start searching, start with 0
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

### Return values

$\geq 0$	Id
$< 0$	Error

Definition at line 97 of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 16.96.2.7 set\_object\_name()

```

l4_msgtag_t L4::Debugger::set_object_name (
    const char * name,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Set the name of a kernel object.

### Parameters

<i>name</i>	Name
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

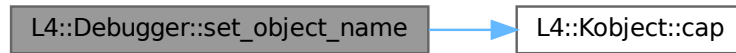
### Returns

System call return tag.

Definition at line 59 of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 16.96.2.8 switch\_log()

```

l4_msgtag_t L4::Debugger::switch_log (
    const char * name,
    unsigned on_off,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Set or unset log.

#### Parameters

<i>name</i>	Name of the log type.
<i>on_off</i>	1: turn log on, 0: turn log off
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

#### Returns

Syscall return tag

Definition at line 133 of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

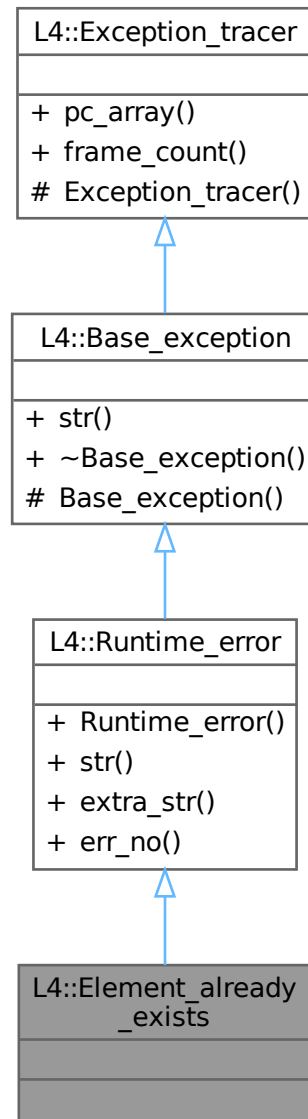
- [l4/sys/debugger](#)

## 16.97 L4::Element\_already\_exists Class Reference

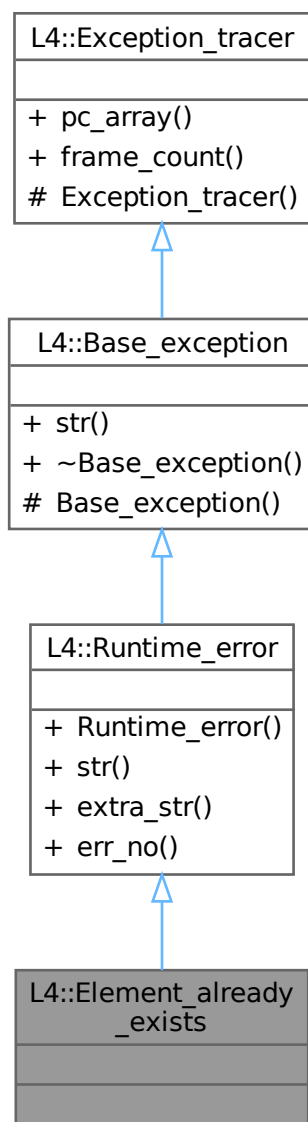
[Exception](#) for duplicate element insertions.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Element\_already\_exists:



Collaboration diagram for L4::Element\_already\_exists:



### Additional Inherited Members

#### Public Member Functions inherited from [L4::Runtime\\_error](#)

- [Runtime\\_error](#) (long [err\\_no](#), char const \*extra=0) noexcept  
*Create a new [Runtime\\_error](#).*
- char const \* [str](#) () const noexcept override  
*Return a human readable string for the exception.*
- char const \* [extra\\_str](#) () const  
*Get the description text for this runtime error.*
- long [err\\_no](#) () const noexcept  
*Get the error value for this runtime error.*

### Public Member Functions inherited from [L4::Base\\_exception](#)

- virtual `~Base_exception ()` noexcept  
*Destruction.*

### Public Member Functions inherited from [L4::Exception\\_tracer](#)

- void const \*const \* `pc_array ()` const noexcept  
*Get the array containing the call trace.*
- int `frame_count ()` const noexcept  
*Get the number of entries that are valid in the call trace.*

### Protected Member Functions inherited from [L4::Base\\_exception](#)

- `Base_exception ()` noexcept  
*Create a base exception.*

### Protected Member Functions inherited from [L4::Exception\\_tracer](#)

- `Exception_tracer ()` noexcept  
*Create a back trace.*

## 16.97.1 Detailed Description

[Exception](#) for duplicate element insertions.

Definition at line 192 of file [exceptions](#).

The documentation for this class was generated from the following file:

- [l4/cxx/exceptions](#)

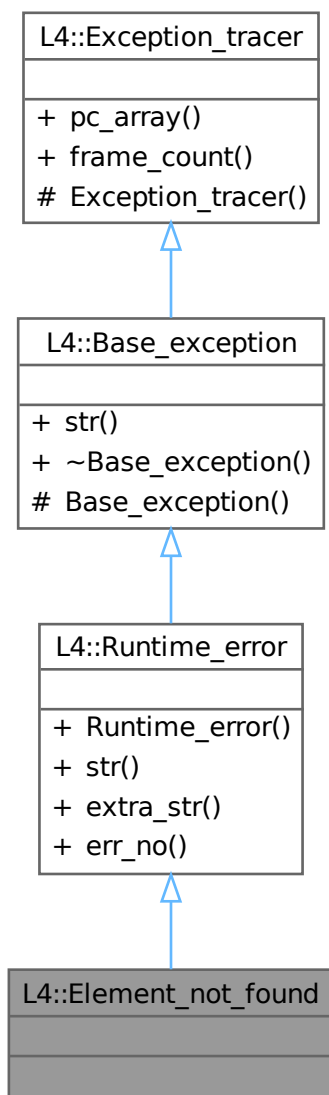
## 16.98 [L4::Element\\_not\\_found](#) Class Reference

[Exception](#) for a failed lookup (element not found).

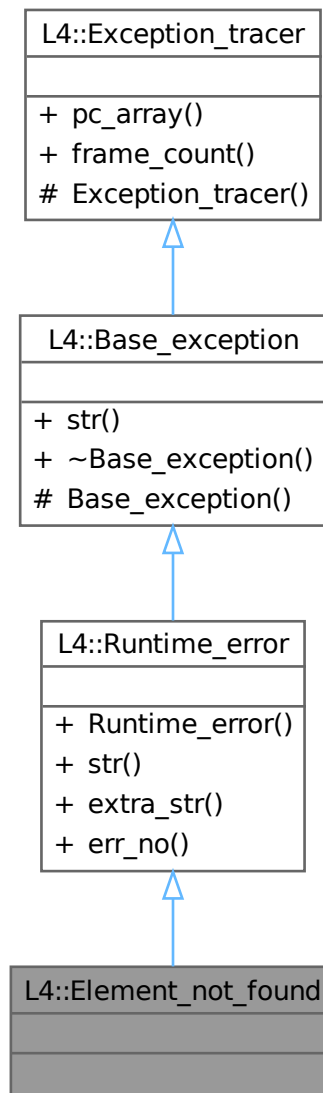
```
#include <l4/cxx/exceptions>
```



Inheritance diagram for L4::Element\_not\_found:



Collaboration diagram for L4::Element\_not\_found:



#### Additional Inherited Members

#### Public Member Functions inherited from [L4::Runtime\\_error](#)

- [Runtime\\_error](#) (long [err\\_no](#), char const \*extra=0) noexcept  
*Create a new [Runtime\\_error](#).*
- char const \* [str](#) () const noexcept override  
*Return a human readable string for the exception.*
- char const \* [extra\\_str](#) () const  
*Get the description text for this runtime error.*
- long [err\\_no](#) () const noexcept  
*Get the error value for this runtime error.*

### Public Member Functions inherited from L4::Base\_exception

- virtual ~**Base\_exception** () noexcept  
*Destruction.*

## Public Member Functions inherited from L4::Exception\_tracer

- `void const *const * pc_array () const noexcept`  
*Get the array containing the call trace.*
- `int frame_count () const noexcept`  
*Get the number of entries that are valid in the call trace.*

### Protected Member Functions inherited from L4::Base\_exception

- **Base\_exception ()** noexcept  
*Create a base exception.*

### Protected Member Functions inherited from [L4::Exception\\_tracer](#)

- **Exception\_tracer ()** noexcept  
*Create a back trace.*

### 16.98.1 Detailed Description

**Exception** for a failed lookup (element not found).

Definition at line 220 of file exceptions.

The documentation for this class was generated from the following file:

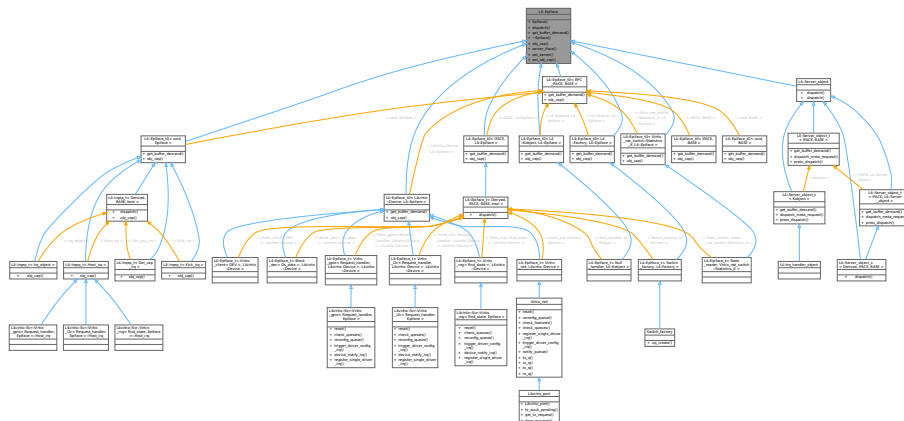
- l4/cxx/exceptions

## 16.99 L4::Epiface Struct Reference

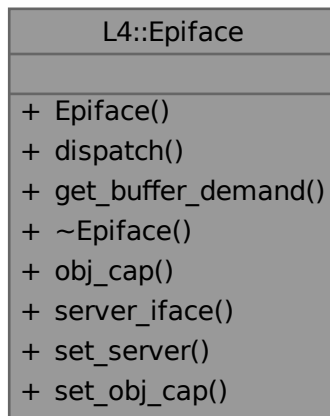
Base class for interface implementations.

```
#include <ipc_epiface>
```

Inheritance diagram for L4::Epiface:



Collaboration diagram for L4::Epiface:



## Public Types

- typedef [lpc\\_svr::Server\\_iface](#) **Server\_iface**  
*Type for abstract server interface.*
- typedef [lpc\\_svr::Server\\_iface::Demand](#) **Demand**  
*Type for server-side receive buffer demand.*

## Public Member Functions

- **Epiface ()**  
*Make a server object.*
- virtual [l4\\_msgtag\\_t](#) **dispatch** ([l4\\_msgtag\\_t](#) tag, unsigned rights, [l4\\_utcb\\_t](#) \*utcb)=0  
*The abstract handler for client requests to the object.*
- virtual [Demand](#) **get\_buffer\_demand** () const =0  
*Get the server-side receive buffer demand for this object.*
- virtual **~Epiface** ()=0  
*Destroy the object.*
- Stored\_cap **obj\_cap** () const  
*Get the capability to the kernel object belonging to this object.*
- [Server\\_iface](#) \* **server\_iface** () const  
*Get pointer to server interface at which the object is currently registered.*
- int **set\_server** ([Server\\_iface](#) \*srv, [Cap](#)< void > cap, bool managed=false)  
*Set server registration info for the object.*
- void **set\_obj\_cap** ([Cap](#)< void > const &cap)  
*Deprecated server registration function.*

## 16.99.1 Detailed Description

Base class for interface implementations.

An [Epiface](#) is the base interface of objects registered in the server loop. Incoming IPC gets dispatched to the appropriate [Epiface](#) object where the call is then handled appropriately.

### Note

[Server](#) loops are allowed to internally keep raw pointers to [Epiface](#) objects for dispatching calls. Instances must therefore never be copied or moved.

Definition at line 145 of file [ipc\\_epiface](#).

## 16.99.2 Member Function Documentation

### 16.99.2.1 dispatch()

```
virtual l4_msgtag_t L4::Epiface::dispatch (
    l4_msgtag_t tag,
    unsigned rights,
    l4_utcb_t * utcb) [pure virtual]
```

The abstract handler for client requests to the object.

#### Parameters

<i>tag</i>	The message tag for this invocation.
<i>rights</i>	The rights bits in the invoked capability.
<i>utcb</i>	The UTCB used for the invocation.

#### Return values

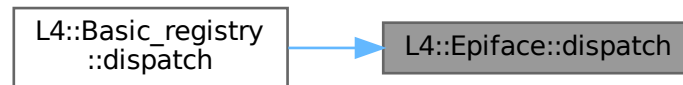
<code>-L4_ENOREPLY</code>	No reply message is send.
<code>&lt;0</code>	Error, reply with error code.
<code>&gt;=0</code>	Success, reply with return value.

This function must be implemented by application specific server objects.

Implemented in [L4::Epiface\\_t< Derived, IFACE, BASE, bool >](#), [L4::Epiface\\_t< Block\\_dev< Ds\\_data >, L4virtio::Device >](#), [L4::Epiface\\_t< Null\\_handler, L4::Kobject >](#), [L4::Epiface\\_t< Stats\\_reader, Virtio\\_net\\_switch::Statistics\\_if >](#), [L4::Epiface\\_t< Switch\\_factory, L4::Factory >](#), [L4::Epiface\\_t< Virtio\\_client< DEV >, L4virtio::Device >](#), [L4::Epiface\\_t< Virtio\\_gpio< Virtio\\_i2c< Request\\_handler, L4virtio::Device >, L4virtio::Device >](#), [L4::Epiface\\_t< Virtio\\_net, L4virtio::Device >](#), [L4::Epiface\\_t< Virtio\\_rng< Rnd\\_state >, L4virtio::Device >](#), [L4::lrqep\\_t< Derived, BASE, bool >](#), [L4::lrqep\\_t< Del\\_cap\\_irq >](#), [L4::lrqep\\_t< Host\\_irq >](#), [L4::lrqep\\_t< Irq\\_object >](#), [L4::lrqep\\_t< Kick\\_irq >](#), and [L4::Server\\_object](#).

Referenced by [L4::Basic\\_registry::dispatch\(\)](#).

Here is the caller graph for this function:



### 16.99.2.2 `get_buffer_demand()`

```
virtual Demand L4::Epiface::get_buffer_demand () const [pure virtual]
```

Get the server-side receive buffer demand for this object.

#### Note

This function is usually not implemented directly, but by using `Server_object_t` template with an IPC interface definition.

#### Returns

The needed server-side receive buffers for this object

Implemented in `L4::Epiface_t0< RPC_IFACE, BASE >`, `L4::Epiface_t0< IFACE, L4::Epiface >`, `L4::Epiface_t0< L4::Factory, L4::Epiface >`, `L4::Epiface_t0< L4::Kobject, L4::Epiface >`, `L4::Epiface_t0< L4virtio::Device, L4::Epiface >`, `L4::Epiface_t0< Virtio_net_switch::State, L4::Epiface >`, `L4::Epiface_t0< void, Epiface >`, `L4::Server_object_t< IFACE, BASE >`, `L4::Server_object_t< IFACE, L4::Server_object >`, and `L4::Server_object_t< Kobject >`.

### 16.99.2.3 `obj_cap()`

```
Stored_cap L4::Epiface::obj_cap () const [inline]
```

Get the capability to the kernel object belonging to this object.

#### Returns

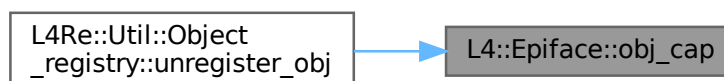
Capability for the kernel object behind the server.

This is usually either an `lpc_gate` or an `lrc`.

Definition at line 206 of file `ipc_epiface`.

Referenced by `L4Re::Util::Object_registry::unregister_obj()`.

Here is the caller graph for this function:



### 16.99.2.4 server\_iface()

```
Server_iface * L4::Epiface::server_iface () const [inline]
```

Get pointer to server interface at which the object is currently registered.

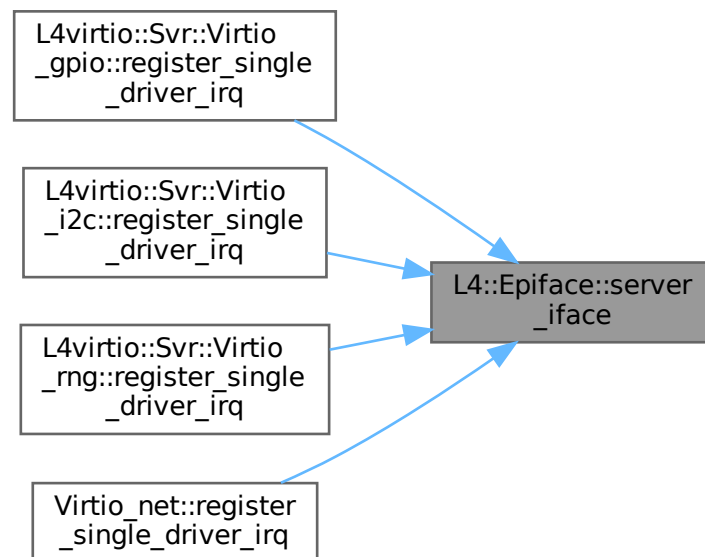
#### Returns

Pointer to the server at which the object is currently registered, NULL if the object is not registered at any server.

Definition at line 213 of file [ipc\\_epiface](#).

Referenced by [L4virtio::Svr::Virtio\\_gpio< Request\\_handler, Epiface >::register\\_single\\_driver\\_irq\(\)](#), [L4virtio::Svr::Virtio\\_i2c< Request\\_handler, Epiface >::register\\_single\\_driver\\_irq\(\)](#), [L4virtio::Svr::Virtio\\_rng< Rnd\\_state, Epiface >::register\\_single\\_driver\\_irq\(\)](#), and [Virtio\\_net::register\\_single\\_driver\\_irq\(\)](#).

Here is the caller graph for this function:



### 16.99.2.5 set\_server()

```
int L4::Epiface::set_server (
    Server_iface * srv,
    Cap< void > cap,
    bool managed = false) [inline]
```

Set server registration info for the object.

#### Parameters

<i>srv</i>	The server to register at
<i>cap</i>	The capability that connects the object.
<i>managed</i>	Mark the capability as managed or unmanaged. Typical server implementations use this flag to remember whether the capability was internally allocated or not.

#### Returns

0 on success, -L4\_EINVAL if the *srv* and *cap* are not consistent.

Definition at line 224 of file [ipc\\_epiface](#).

References [L4\\_EINVAL](#).

Referenced by [L4Re::Util::Object\\_registry::unregister\\_obj\(\)](#).

Here is the caller graph for this function:



The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_epiface`

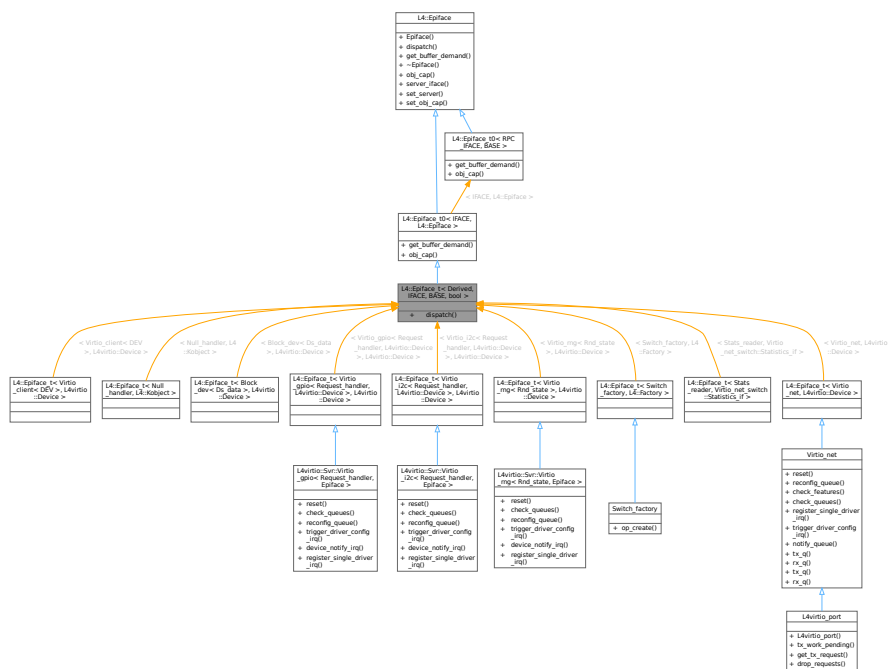
## 16.100 L4::Epiface\_t< Derived, IFACE, BASE, bool > Struct Template Reference

[Epiface](#) implementation for Kobject-based interface implementations.

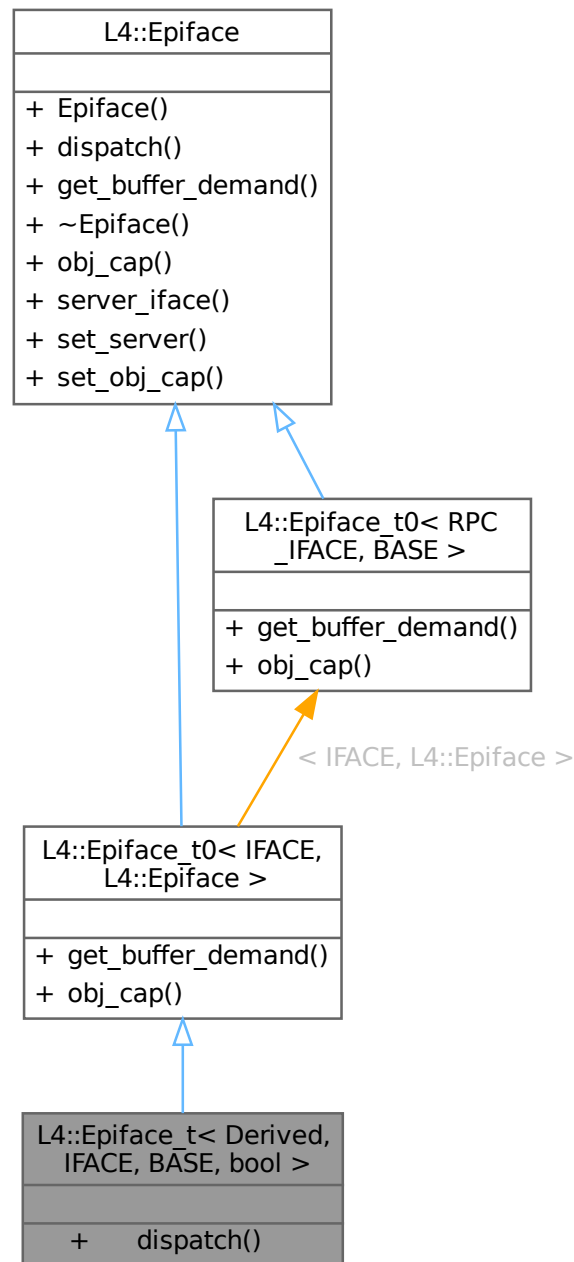
```
#include <ipc_epiface>
```



Inheritance diagram for L4::Epiface\_t< Derived, IFACE, BASE, bool >:



Collaboration diagram for L4::Epiface\_t< Derived, IFACE, BASE, bool >:



### Public Member Functions

- `l4_msgtag_t dispatch (l4_msgtag_t tag, unsigned rights, l4_utcb_t *utcb) final`  
*The abstract handler for client requests to the object.*

### Public Member Functions inherited from **L4::Epiface\_t0< IFACE, L4::Epiface >**

- `Type_info::Demand get_buffer_demand () const`

*Get the server-side buffer demand based in IFACE.*

- [Cap](#)< IFACE > [obj\\_cap](#) () const

*Get the (typed) capability to this object.*

## Public Member Functions inherited from [L4::Epiface](#)

- [Epiface](#) ()

*Make a server object.*

- virtual [~Epiface](#) ()=0

*Destroy the object.*

- Stored\_cap [obj\\_cap](#) () const

*Get the capability to the kernel object belonging to this object.*

- [Server\\_iface](#) \* [server\\_iface](#) () const

*Get pointer to server interface at which the object is currently registered.*

- int [set\\_server](#) ([Server\\_iface](#) \*srv, [Cap](#)< void > cap, bool managed=false)

*Set server registration info for the object.*

- void [set\\_obj\\_cap](#) ([Cap](#)< void > const &cap)

*Deprecated server registration function.*

## Additional Inherited Members

## Public Types inherited from [L4::Epiface\\_t](#)< IFACE, [L4::Epiface](#) >

- typedef IFACE **Interface**

*Data type of the IPC interface definition.*

## Public Types inherited from [L4::Epiface](#)

- typedef [ipc\\_svr::Server\\_iface](#) **Server\_iface**

*Type for abstract server interface.*

- typedef [ipc\\_svr::Server\\_iface::Demand](#) **Demand**

*Type for server-side receive buffer demand.*

### 16.100.1 Detailed Description

```
template<typename Derived, typename IFACE, typename BASE = L4::Epiface, bool = cxx::is_↵
polymorphic<BASE>::value>
struct L4::Epiface_t< Derived, IFACE, BASE, bool >
```

[Epiface](#) implementation for Kobject-based interface implementations.

## Template Parameters

<i>Derived</i>	Class providing the interface implementations.
<i>BASE</i>	<a href="#">Epiface</a> base class.

## Examples

[examples/clntsrv/src/server.cc](#).

Definition at line 503 of file [ipc\\_epiface](#).

## 16.100.2 Member Function Documentation

### 16.100.2.1 dispatch()

```
template<typename Derived, typename IFACE, typename BASE = L4::Epiface, bool = cxx::is_↵
polymorphic<BASE>::value>
l4_msgtag_t L4::Epiface_t< Derived, IFACE, BASE, bool >::dispatch (
    l4_msgtag_t tag,
    unsigned rights,
    l4_utcb_t * utcb) [inline], [final], [virtual]
```

The abstract handler for client requests to the object.

#### Parameters

<i>tag</i>	The message tag for this invocation.
<i>rights</i>	The rights bits in the invoked capability.
<i>utcb</i>	The UTCB used for the invocation.

#### Return values

<code>-L4_ENOREPLY</code>	No reply message is send.
<code>&lt; 0</code>	Error, reply with error code.
<code>&gt;= 0</code>	Success, reply with return value.

This function must be implemented by application specific server objects.

Implements [L4::Epiface](#).

Definition at line [506](#) of file [ipc\\_epiface](#).

The documentation for this struct was generated from the following file:

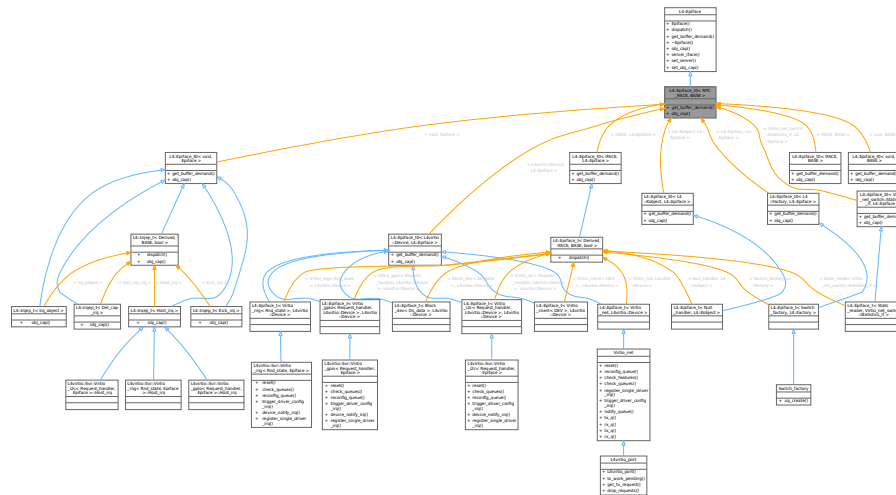
- `l4/sys/cxx/ipc_epiface`

## 16.101 L4::Epiface\_t0< RPC\_IFACE, BASE > Struct Template Reference

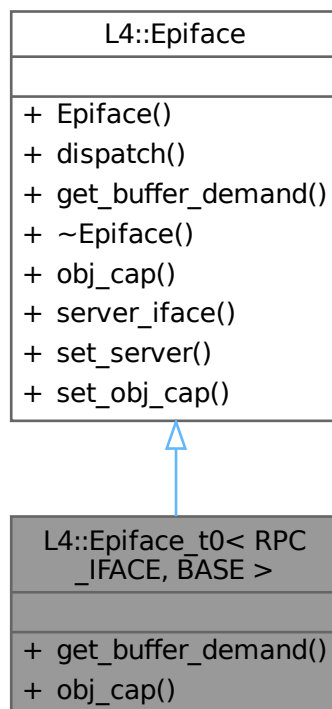
[Epiface](#) mixin for generic Kobject-based interfaces.

```
#include <ipc_epiface>
```

Inheritance diagram for L4::Epiface\_t0< RPC\_IFACE, BASE >:



Collaboration diagram for L4::Epiface\_t0< RPC\_IFACE, BASE >:



## Public Types

- typedef `RPC_IFACE` **Interface**  
Data type of the IPC interface definition.

## Public Types inherited from [L4::Epiface](#)

- typedef [lpc\\_svr::Server\\_iface](#) **Server\_iface**  
*Type for abstract server interface.*
- typedef [lpc\\_svr::Server\\_iface::Demand](#) **Demand**  
*Type for server-side receive buffer demand.*

## Public Member Functions

- [Type\\_info::Demand](#) **get\_buffer\_demand** () const  
*Get the server-side buffer demand based in IFACE.*
- [Cap](#)< [RPC\\_IFACE](#) > **obj\_cap** () const  
*Get the (typed) capability to this object.*

## Public Member Functions inherited from [L4::Epiface](#)

- **Epiface** ()  
*Make a server object.*
- virtual [l4\\_msgtag\\_t](#) **dispatch** ([l4\\_msgtag\\_t](#) tag, unsigned rights, [l4\\_utcb\\_t](#) \*utcb)=0  
*The abstract handler for client requests to the object.*
- virtual ~**Epiface** ()=0  
*Destroy the object.*
- Stored\_cap **obj\_cap** () const  
*Get the capability to the kernel object belonging to this object.*
- [Server\\_iface](#) \* **server\_iface** () const  
*Get pointer to server interface at which the object is currently registered.*
- int **set\_server** ([Server\\_iface](#) \*srv, [Cap](#)< void > cap, bool managed=false)  
*Set server registration info for the object.*
- void **set\_obj\_cap** ([Cap](#)< void > const &cap)  
*Deprecated server registration function.*

## 16.101.1 Detailed Description

```
template<typename RPC_IFACE, typename BASE = Epiface>
struct L4::Epiface_t0< RPC_IFACE, BASE >
```

[Epiface](#) mixin for generic Kobject-based interfaces.

### Template Parameters

<i>RPC_IFACE</i>	Data type of the IPC interface definition.
<i>BASE</i>	Base <a href="#">Epiface</a> class.

Definition at line 256 of file [ipc\\_epiface](#).

## 16.101.2 Member Function Documentation

### 16.101.2.1 obj\_cap()

```
template<typename RPC_IFACE, typename BASE = Epiface>
Cap< RPC_IFACE > L4::Epiface_t0< RPC_IFACE, BASE >::obj_cap () const [inline]
```

Get the (typed) capability to this object.

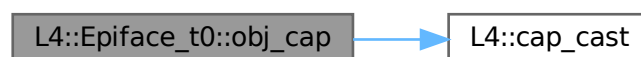
#### Returns

Capability for the kernel object behind the server.

Definition at line 269 of file [ipc\\_epiface](#).

References [L4::cap\\_cast\(\)](#).

Here is the call graph for this function:



The documentation for this struct was generated from the following file:

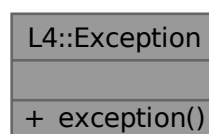
- `l4/sys/cxx/ipc_epiface`

## 16.102 L4::Exception Class Reference

[Exception](#) interface.

```
#include <exception>
```

Collaboration diagram for L4::Exception:



## Public Member Functions

- [l4\\_msgtag\\_t exception](#) ([L4::lpc::In\\_out](#)< [l4\\_exc\\_regs\\_t](#) \* > *regs*, [L4::lpc::Rcv\\_fpage](#) *rwin*, [L4::lpc::Opt](#)< [L4::lpc::Snd\\_fpage](#) & > *fp*)  
*Exception call.*

### 16.102.1 Detailed Description

[Exception](#) interface.

This class defines the interface for handling exception IPC. When an exception occurs during program execution, for example due to a division by zero, the kernel will synthesise an exception IPC and send it to the thread's exception handler, who can then handle it.

The exception handler is set with the [L4::Thread::control](#) interface.

Definition at line 31 of file [exception](#).

### 16.102.2 Member Function Documentation

#### 16.102.2.1 exception()

```
l4\_msgtag\_t L4::Exception::exception (
    L4::Ipc::In\_out< l4\_exc\_regs\_t * > regs,
    L4::Ipc::Rcv\_fpage rwin,
    L4::Ipc::Opt< L4::Ipc::Snd\_fpage & > fp)
```

[Exception](#) call.

#### Parameters

	<i>regs</i>	Register state of the faulting thread.
	<i>rwin</i>	Receive window in the address space.
out	<i>fp</i>	Optional flexpage to resolve the exception.

#### Returns

Message tag containing error code.

The documentation for this class was generated from the following file:

- [l4/sys/exception](#)

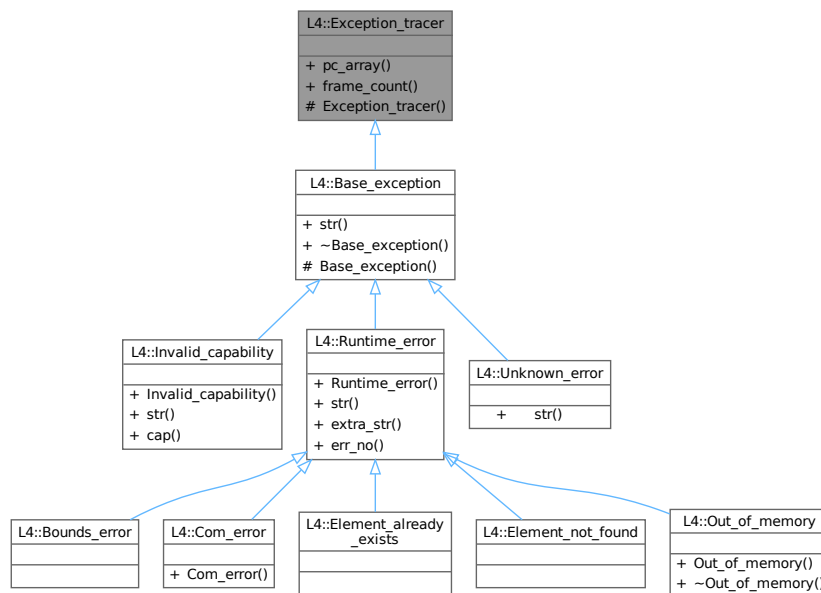


## 16.103 L4::Exception\_tracer Class Reference

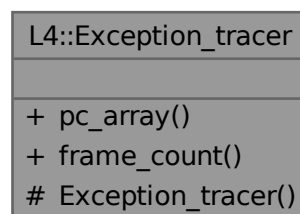
Back-trace support for exceptions.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Exception\_tracer:



Collaboration diagram for L4::Exception\_tracer:



### Public Member Functions

- void const \*const \* **pc\_array** () const noexcept  
*Get the array containing the call trace.*
- int **frame\_count** () const noexcept  
*Get the number of entries that are valid in the call trace.*

### Protected Member Functions

- **Exception\_tracer** () noexcept  
*Create a back trace.*

## 16.103.1 Detailed Description

Back-trace support for exceptions.

This class holds an array of at most [L4\\_CXX\\_EXCEPTION\\_BACKTRACE](#) instruction pointers containing the call trace at the instant when an exception was thrown.

Definition at line 51 of file [exceptions](#).

The documentation for this class was generated from the following file:

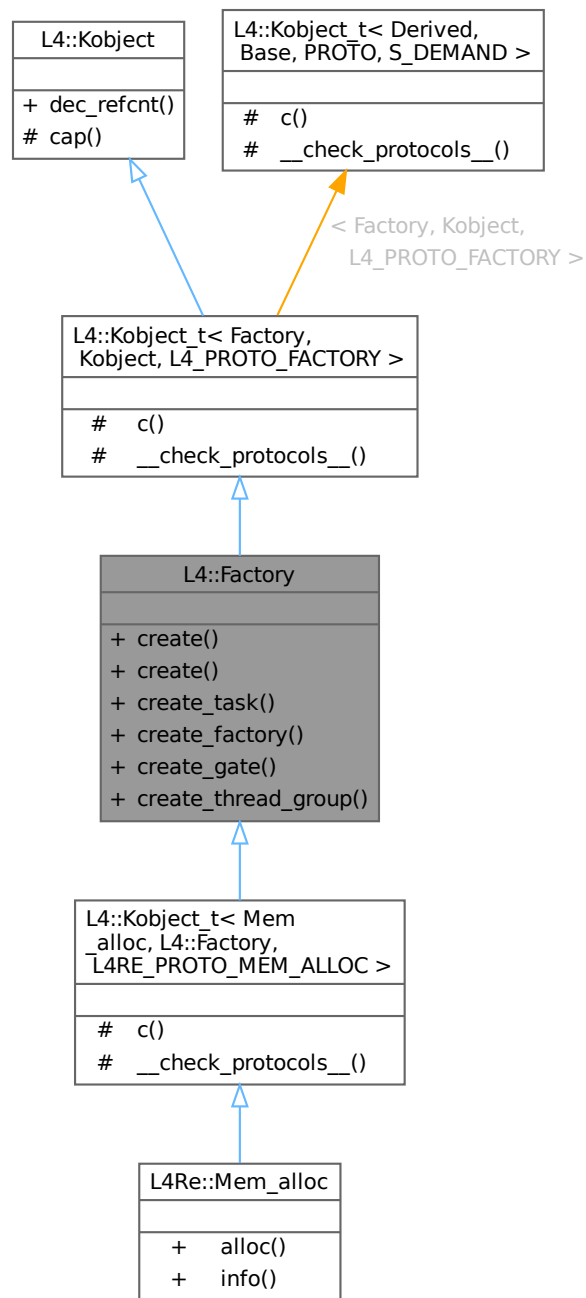
- [l4/cxx/exceptions](#)

## 16.104 L4::Factory Class Reference

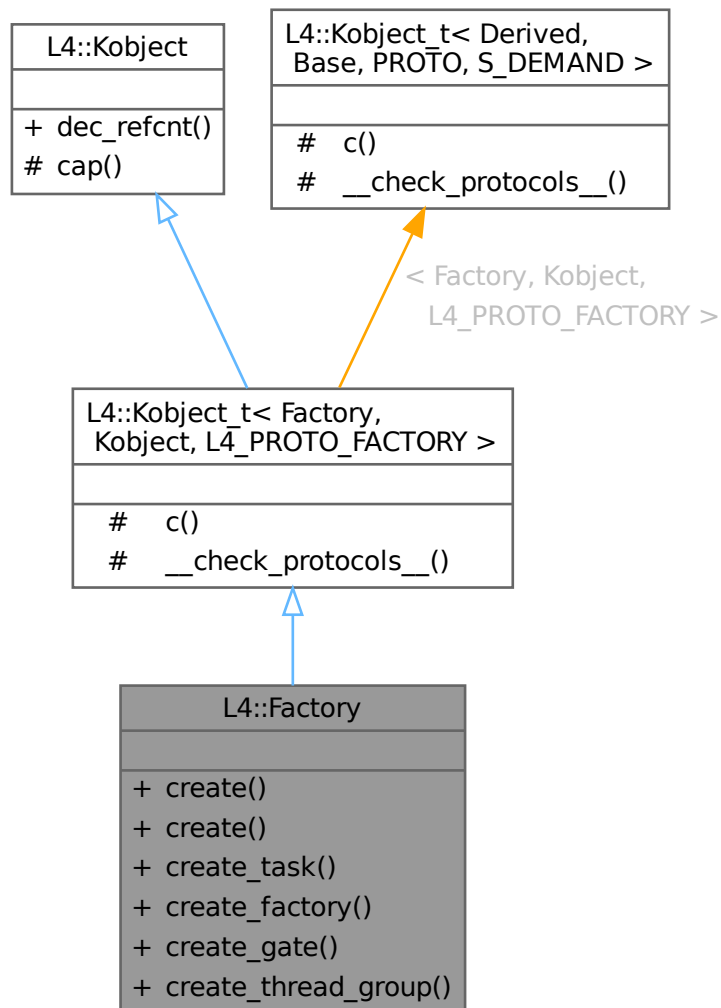
C++ Factory interface, see [Factory](#) for the C interface.

```
#include <factory>
```

Inheritance diagram for L4::Factory:



Collaboration diagram for L4::Factory:



## Data Structures

- struct [Nil](#)  
*Special type to add a void argument into the factory create stream.*
- struct [Lstr](#)  
*Special type to add a pascal string into the factory create stream.*
- class [S](#)  
*Stream class for the [create\(\)](#) argument stream.*

## Public Member Functions

- [S create](#) ([Cap](#)< void > target, long obj, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Generic create call to the factory.*

- `template<typename OBJ>`  
`S create (Cap< OBJ > target, l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Create call for typed capabilities.*
- `l4_msgtag_t create_task (Cap< Task > const &target_cap, l4_fpage_t *utcb_area, l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Create a new task.*
- `l4_msgtag_t create_factory (Cap< Factory > const &target_cap, unsigned long limit, l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Create a new factory.*
- `l4_msgtag_t create_gate (Cap< void > const &target_cap, Cap< Snd_destination > const &snd_dst_cap, l4_umword_t label, l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Create a new IPC gate, optionally bound to a send destination (a thread or thread group).*
- `l4_msgtag_t create_thread_group (Cap< Thread_group > const &target_cap, unsigned policy, l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Create a new thread group.*

## Public Member Functions inherited from L4::Kobject

- `l4_msgtag_t dec_refcnt (l4_mword_t diff, l4_utcb_t *utcb=l4_utcb())`  
*Decrement the in kernel reference counter for the object.*

## Additional Inherited Members

## Protected Types inherited from L4::Kobject\_t< Factory, Kobject, L4\_PROTO\_FACTORY >

- `typedef Factory Class`  
*The target interface type (inheriting from Kobject\_t).*
- `typedef Typeid::Iface< PROTO, Factory > __iface`  
*The interface description for the derived class.*
- `typedef Typeid::Merge_list< Typeid::Iface_list< __iface >, typename Kobject::__iface_list > __iface_list`  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Member Functions inherited from L4::Kobject\_t< Factory, Kobject, L4\_PROTO\_FACTORY >

- `L4::Cap< Class > c () const noexcept`  
*Get the capability to ourselves.*

## Protected Member Functions inherited from L4::Kobject

- `l4_cap_idx_t cap () const noexcept`  
*Return capability selector.*

## Static Protected Member Functions inherited from L4::Kobject\_t< Factory, Kobject, L4\_PROTO\_FACTORY >

- `static void __check_protocols__ () noexcept`  
*Helper to check for protocol conflicts.*

### 16.104.1 Detailed Description

C++ Factory interface, see [Factory](#) for the C interface.

Factories provide an interface to create objects which are accessed via capabilities.

For additional information about which objects can be created via this interface, see server-specific information in [Kernel Factory](#) and [L4Re Servers](#).

#### Include File

```
#include <l4/sys/factory>
```

For the C interface refer to [Factory](#).

Definition at line 38 of file [factory](#).

### 16.104.2 Member Function Documentation

#### 16.104.2.1 create() [1/2]

```
template<typename OBJ>
S L4::Factory::create (
    Cap< OBJ > target,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Create call for typed capabilities.

#### Template Parameters

<i>OBJ</i>	Capability type of the object to be created.
------------	----------------------------------------------

#### Parameters

out	<i>target</i>	Capability of type OBJ.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

#### Returns

A create stream that allows additional arguments to be passed to the `create()` call via the left-shift (`<<`) operator (see `#S::operator <<`).

This method does not directly invoke the factory. The factory is invoked when the create stream returned by this method is converted to an [l4\\_msgtag\\_t](#) (see `S::operator l4_msgtag_t()`), or otherwise when the stream goes out of scope (not recommended; see `#S::~~S()`).

**Precondition**

The invoked [Factory](#) capability must have the permission [L4\\_CAP\\_FPAGE\\_S](#), otherwise the later factory IPC will fail with [L4\\_EPERM](#) (see [S::operator l4\\_msgtag\\_t\(\)](#)).

**Note**

The create stream uses the UTCB to store parameters for the service call. During the lifetime of a create stream or, until it is converted to an [l4\\_msgtag\\_t](#), other UTCB-using operations must not be used.

**Usage:**

```
L4::Cap<L4Re::Dataspace> ds = L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>();
factory->create(ds) << l4_mword_t(size_in_bytes);
```

Definition at line 329 of file [factory](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:

**16.104.2.2 create() [2/2]**

```
S L4::Factory::create (
    Cap< void > target,
    long obj,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Generic create call to the factory.

**Parameters**

out	<i>target</i>	Capability selector for the new object. The caller must allocate the capability slot. The kernel stores the new objects's capability into this slot.
	<i>obj</i>	The protocol ID that specifies which kind of object shall be created.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

**Returns**

A create stream that allows additional arguments to be passed to the `create()` call via the left-shift (`<<`) operator (see [#S::operator <<](#)).

This method does not directly invoke the factory. The factory is invoked when the create stream returned by this method is converted to an [l4\\_msgtag\\_t](#) (see [S::operator l4\\_msgtag\\_t\(\)](#)), or otherwise when the stream goes out of scope (not recommended; see [#S::~~S\(\)](#)).

**Precondition**

The invoked [Factory](#) capability must have the permission [L4\\_CAP\\_FPAGE\\_S](#), otherwise the later factory IPC will fail with [L4\\_EPERM](#) (see [S::operator l4\\_msgtag\\_t\(\)](#)).

**Note**

The create stream uses the UTCB to store parameters for the service call. During the lifetime of a create stream or, until it is converted to an [l4\\_msgtag\\_t](#), other UTCB-using operations must not be used.

**See also**

[create\(Cap<OBJ>, l4\\_utcb\\_t \\*\)](#)

Definition at line [292](#) of file [factory](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:

**16.104.2.3 create\_factory()**

```

l4_msgtag_t L4::Factory::create_factory (
    Cap< Factory > const & target_cap,
    unsigned long limit,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Create a new factory.

**Parameters**

out	<i>target_cap</i>	The kernel stores the new factory's capability into this slot.
	<i>limit</i>	Limit for the new factory in bytes.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

**Returns**

Syscall return tag

**Return values**



<code>L4_EOK</code>	No error occurred.
<code>-L4_EPERM</code>	Insufficient permissions; see precondition.
<code>&lt;0</code>	Error code.

**Precondition**

The invoked [Factory](#) capability must have the permission [L4\\_CAP\\_FPAGE\\_S](#).

**Note**

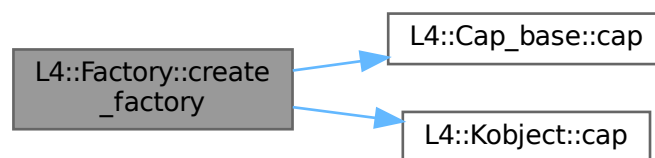
In addition to memory needed for internal data structures, the `limit` (quota) of the new factory is counted towards the quota of the creating factory. The `limit` must be within  $1 \leq \text{limit} \leq 2^{(8 * \text{sizeof}(\text{l4\_umword\_t}) - 1) - 2}$  otherwise the behavior is undefined.

This method is only guaranteed to work with the [Kernel Factory](#). For other services, use the generic [create\(\)](#) method and consult the service documentation for information on the arguments that need to be passed to the create stream.

Definition at line [404](#) of file [factory](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:

**16.104.2.4 create\_gate()**

```

l4_msgtag_t L4::Factory::create_gate (
    Cap< void > const & target_cap,
    Cap< Snd_destination > const & snd_dst_cap,
    l4_umword_t label,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Create a new IPC gate, optionally bound to a send destination (a thread or thread group).

**Parameters**

<code>out</code>	<code>target_cap</code>	The kernel stores the new IPC gate's capability into this slot.
------------------	-------------------------	-----------------------------------------------------------------

	<i>snd_dst_cap</i>	Optional capability selector of a thread or thread group to bind the gate to. Use <a href="#">L4_INVALID_CAP</a> to create an unbound IPC gate.
	<i>label</i>	Optional label of the gate (precisely used if <i>snd_dst_cap</i> is valid). If <i>snd_dst_cap</i> is valid, <i>label</i> must be present.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

### Returns

Syscall return tag containing one of the following return codes.

### Return values

<i>L4_EOK</i>	No error occurred.
<i>-L4_ENOMEM</i>	Out-of-memory during allocation of the <a href="#">lpc_gate</a> object.
<i>-L4_EINVAL</i>	<i>snd_dst_cap</i> is void or points to something that is not a thread or thread group.
<i>-L4_EPERM</i>	Insufficient permissions; see precondition.

### Precondition

The invoked [Factory](#) capability must have the permission [L4\\_CAP\\_FPAGE\\_S](#). Also *snd\_dst\_cap* (if *valid*) must have the permission [L4\\_CAP\\_FPAGE\\_S](#).

An unbound IPC gate can be bound to a thread or thread group using [L4::lpc\\_gate::bind\\_thread\(\)](#) or [bind\\_snd\\_destination\(\)](#).

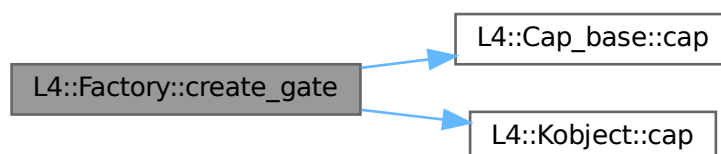
### See also

[L4::lpc\\_gate](#)

Definition at line 440 of file [factory](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



## 16.104.2.5 create\_task()

```
l4_msgtag_t L4::Factory::create_task (
    Cap< Task > const & target_cap,
    l4_fpage_t * utcb_area,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Create a new task.

## Parameters

out	<i>target_cap</i>	The kernel stores the new task's capability into this slot.
in, out	<i>utcb_area</i>	Flexpage that describes an area in the address space of the new task, where the kernel should map the kernel-allocated kernel-user memory to. The kernel uses the kernel-user memory to store UTCBs and vCPU state-save-areas of the new task.

On systems without MMU, the flexpage is adjusted to reflect the actually allocated physical address.

## Parameters

<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .
-------------	--------------------------------------------------------------------------------------------------------------------

## Returns

Syscall return tag

## Return values

<i>L4_EOK</i>	No error occurred.
<i>-L4_EPERM</i>	Insufficient permissions; see precondition.
<i>&lt;0</i>	Error code.

## Precondition

The invoked [Factory](#) capability must have the permission [L4\\_CAP\\_FPAGE\\_S](#).

## Note

The size of the UTCB area specifies indirectly the number of UTCBs available for this task. Refer to [L4::Task::add\\_ku\\_mem](#) for adding more of this type of memory.

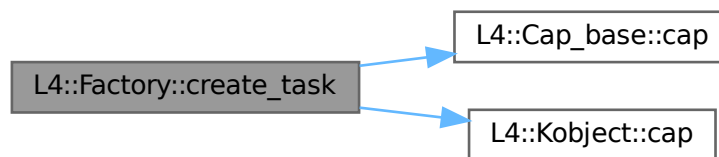
See also

[L4::Task](#)

Definition at line 370 of file [factory](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



#### 16.104.2.6 create\_thread\_group()

```

l4_msgtag_t L4::Factory::create_thread_group (
    Cap< Thread_group > const & target_cap,
    unsigned policy,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Create a new thread group.

An IPC endpoint can be bound to a thread group. When a message arrives at the IPC endpoint, a specific thread of the thread group is selected to actually receive the message. A thread group is a send destination for an IPC endpoint.

##### Parameters

out	<i>target_cap</i>	The kernel stores the new thread group's capability into this slot.
	<i>policy</i>	Policy parameter for the thread group. See <code>L4_thread_group_policy</code> for a list of supported values.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

##### Returns

Syscall return tag containing one of the following return codes.

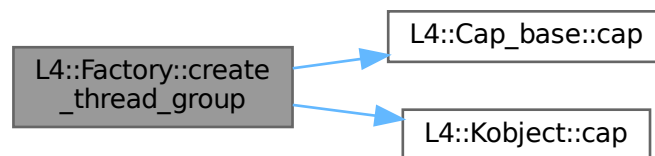
##### Return values

<i>L4_EOK</i>	No error occurred.
<i>-L4_ENOMEM</i>	Out-of-memory during allocation of the <a href="#">Thread_group</a> object.
<i>-L4_EINVAL</i>	Invalid policy parameter.
<i>-L4_EPERM</i>	The factory instance requires <a href="#">L4_CAP_FPAGE_S</a> rights on the invoked capability and <a href="#">L4_CAP_FPAGE_S</a> is not present.

Definition at line 475 of file [factory](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

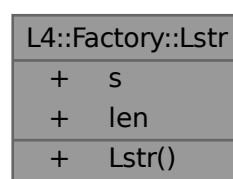
- [l4/sys/factory](#)

## 16.105 L4::Factory::Lstr Struct Reference

Special type to add a pascal string into the factory create stream.

```
#include <factory>
```

Collaboration diagram for L4::Factory::Lstr:



## Public Member Functions

- [Lstr](#) (char const \*[s](#), unsigned [len](#)) noexcept

## Data Fields

- char const \* **s**  
*The character buffer.*
- unsigned **len**  
*The number of characters in the buffer.*

### 16.105.1 Detailed Description

Special type to add a pascal string into the factory create stream.

This encapsulates a string that has an explicit length.

Definition at line [54](#) of file [factory](#).

### 16.105.2 Constructor & Destructor Documentation

#### 16.105.2.1 Lstr()

```
L4::Factory::Lstr::Lstr (  
    char const * s,  
    unsigned len) [inline], [noexcept]
```

#### Parameters

<i>s</i>	Pointer to the c-style string.
<i>len</i>	Length in number of characters of the string <i>s</i> .

Definition at line [70](#) of file [factory](#).

References [len](#), and [s](#).

The documentation for this struct was generated from the following file:

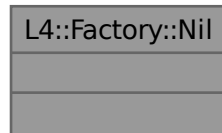
- [l4/sys/factory](#)

## 16.106 L4::Factory::Nil Struct Reference

Special type to add a void argument into the factory create stream.

```
#include <factory>
```

Collaboration diagram for L4::Factory::Nil:



### 16.106.1 Detailed Description

Special type to add a void argument into the factory create stream.

Definition at line 47 of file [factory](#).

The documentation for this struct was generated from the following file:

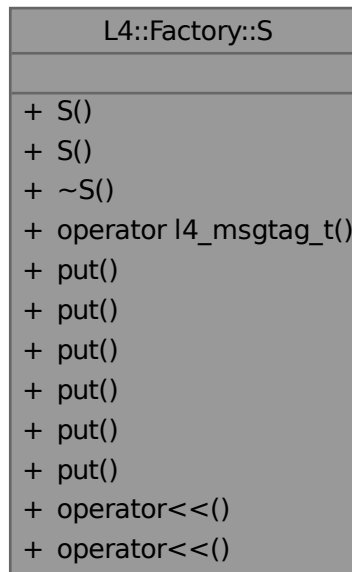
- [l4/sys/factory](#)

## 16.107 L4::Factory::S Class Reference

Stream class for the [create\(\)](#) argument stream.

```
#include <factory>
```

Collaboration diagram for L4::Factory::S:



### Public Member Functions

- **S** (**S** &&o) noexcept  
*Move constructor.*
- **S** (**l4\_cap\_idx\_t** f, long obj, **L4::Cap**< void > target, **l4\_utcb\_t** \*utcb) noexcept  
*Create a stream for a specific [create\(\)](#) call.*
- **~S** () noexcept  
*Commit the [create\(\)](#) operation if not already done explicitly via [operator l4\\_msgtag\\_t\(\)](#).*
- **operator l4\_msgtag\_t** () noexcept  
*Explicitly commits the operation and returns the result.*
- void **put** (**l4\_mword\_t** i) noexcept  
*Put a single [l4\\_mword\\_t](#) as next argument.*
- void **put** (**l4\_umword\_t** i) noexcept  
*Put a single [l4\\_umword\\_t](#) as next argument.*
- void **put** (char const \*s) &noexcept  
*Add a zero-terminated string as next argument.*
- void **put** (**Lstr** const &s) &noexcept  
*Add a pascal string as next argument.*
- void **put** (**Nil**) &noexcept  
*Add an empty argument.*
- void **put** (**l4\_fpage\_t** d) &noexcept  
*Add a flexpage as next argument.*
- template<typename T>  
**S** & **operator<<** (T const &d) &noexcept  
*Add next argument.*
- template<typename T>  
**S** && **operator<<** (T const &d) &&noexcept  
*Add next argument.*



## 16.107.1 Detailed Description

Stream class for the [create\(\)](#) argument stream.

This stream allows a variable number of arguments to be added to a [create\(\)](#) call.

Definition at line 79 of file [factory](#).

## 16.107.2 Constructor & Destructor Documentation

### 16.107.2.1 S() [1/2]

```
L4::Factory::S::S (
    S && o) [inline], [noexcept]
```

Move constructor.

#### Parameters

<i>o</i>	Instance of <a href="#">S</a> to move.
----------	----------------------------------------

Definition at line 98 of file [factory](#).

### 16.107.2.2 S() [2/2]

```
L4::Factory::S::S (
    l4_cap_idx_t f,
    long obj,
    L4::Cap< void > target,
    l4_utcb_t * utcb) [inline], [noexcept]
```

Create a stream for a specific [create\(\)](#) call.

#### Parameters

	<i>f</i>	The capability for the factory object ( <a href="#">L4::Factory</a> ).
	<i>obj</i>	The protocol ID to describe the type of the object that shall be created.
out	<i>target</i>	The capability selector for the new object. The caller must allocate the capability slot. The kernel stores the new object's capability into this slot.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

#### Precondition

The capability *f* must have the permission [L4\\_CAP\\_FPAGE\\_S](#), otherwise the later factory IPC will fail with [L4\\_EPERM](#).

Definition at line 125 of file [factory](#).

### 16.107.2.3 ~S()

```
L4::Factory::S::~~S () [inline], [noexcept]
```

Commit the [create\(\)](#) operation if not already done explicitly via [operator l4\\_msgtag\\_t\(\)](#).

#### Warning

If the commit is deferred until destruction, potential errors are silently ignored. It is therefore recommended to commit explicitly via [operator l4\\_msgtag\\_t\(\)](#) and check the return value.

Definition at line 139 of file [factory](#).

## 16.107.3 Member Function Documentation

### 16.107.3.1 operator l4\_msgtag\_t()

```
L4::Factory::S::operator l4_msgtag_t () [inline], [noexcept]
```

Explicitly commits the operation and returns the result.

#### Returns

The result of the [create\(\)](#) operation.

#### Return values

<i>L4_EOK</i>	No error occurred.
<i>-L4_EPERM</i>	Insufficient permissions; see precondition.
<i>&lt;0</i>	Error code.

#### Precondition

The invoked [Factory](#) capability must have the permission [L4\\_CAP\\_FPAGE\\_S](#).

Definition at line 157 of file [factory](#).

### 16.107.3.2 operator<<() [1/2]

```
template<typename T>
S && L4::Factory::S::operator<< (
    T const & d) && [inline], [noexcept]
```

Add next argument.

#### Template Parameters

<i>T</i>	The argument type. Compilation succeeds only if it is a possible argument type for <code>S::put()</code> .
----------	------------------------------------------------------------------------------------------------------------

### Parameters

<i>d</i>	The value to add as next argument.
----------	------------------------------------

Definition at line 252 of file [factory](#).

References [put\(\)](#).

Here is the call graph for this function:



### 16.107.3.3 operator<<() [2/2]

```
template<typename T>
S & L4::Factory::S::operator<< (
    T const & d) & [inline], [noexcept]
```

Add next argument.

### Template Parameters

<i>T</i>	The argument type. Compilation succeeds only if it is a possible argument type for <code>S::put()</code> .
----------	------------------------------------------------------------------------------------------------------------

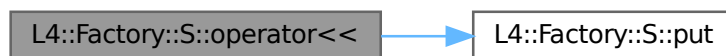
### Parameters

<i>d</i>	The value to add as next argument.
----------	------------------------------------

Definition at line 237 of file [factory](#).

References [put\(\)](#).

Here is the call graph for this function:



**16.107.3.4 put()** [1/5]

```
void L4::Factory::S::put (
    char const * s) & [inline], [noexcept]
```

Add a zero-terminated string as next argument.

**Parameters**

<i>s</i>	The string to add as next argument.
----------	-------------------------------------

The string will be added with the zero-terminator.

Definition at line 191 of file [factory](#).

**16.107.3.5 put()** [2/5]

```
void L4::Factory::S::put (
    l4_fpage_t d) & [inline], [noexcept]
```

Add a flexpage as next argument.

**Parameters**

<i>d</i>	The flexpage to add (there will be no map operation).
----------	-------------------------------------------------------

Definition at line 223 of file [factory](#).

**16.107.3.6 put()** [3/5]

```
void L4::Factory::S::put (
    l4_mword_t i) [inline], [noexcept]
```

Put a single [l4\\_mword\\_t](#) as next argument.

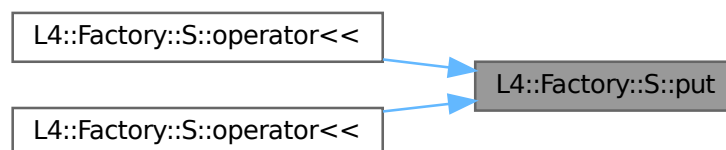
**Parameters**

<i>i</i>	The value to add as next argument.
----------	------------------------------------

Definition at line 169 of file [factory](#).

Referenced by [operator<<\(\)](#), and [operator<<\(\)](#).

Here is the caller graph for this function:



**16.107.3.7 put()** [4/5]

```
void L4::Factory::S::put (
    l4_umword_t i) [inline], [noexcept]
```

Put a single [l4\\_umword\\_t](#) as next argument.

**Parameters**

<i>i</i>	The value to add as next argument.
----------	------------------------------------

Definition at line [179](#) of file [factory](#).

**16.107.3.8 put()** [5/5]

```
void L4::Factory::S::put (
    Lstr const & s) & [inline], [noexcept]
```

Add a pascal string as next argument.

**Parameters**

<i>s</i>	The string to add as next argument.
----------	-------------------------------------

The string will be added with the exact length given. It is the responsibility of the caller to make sure that the string is zero- terminated when that is required by the server.

Definition at line [205](#) of file [factory](#).

The documentation for this class was generated from the following file:

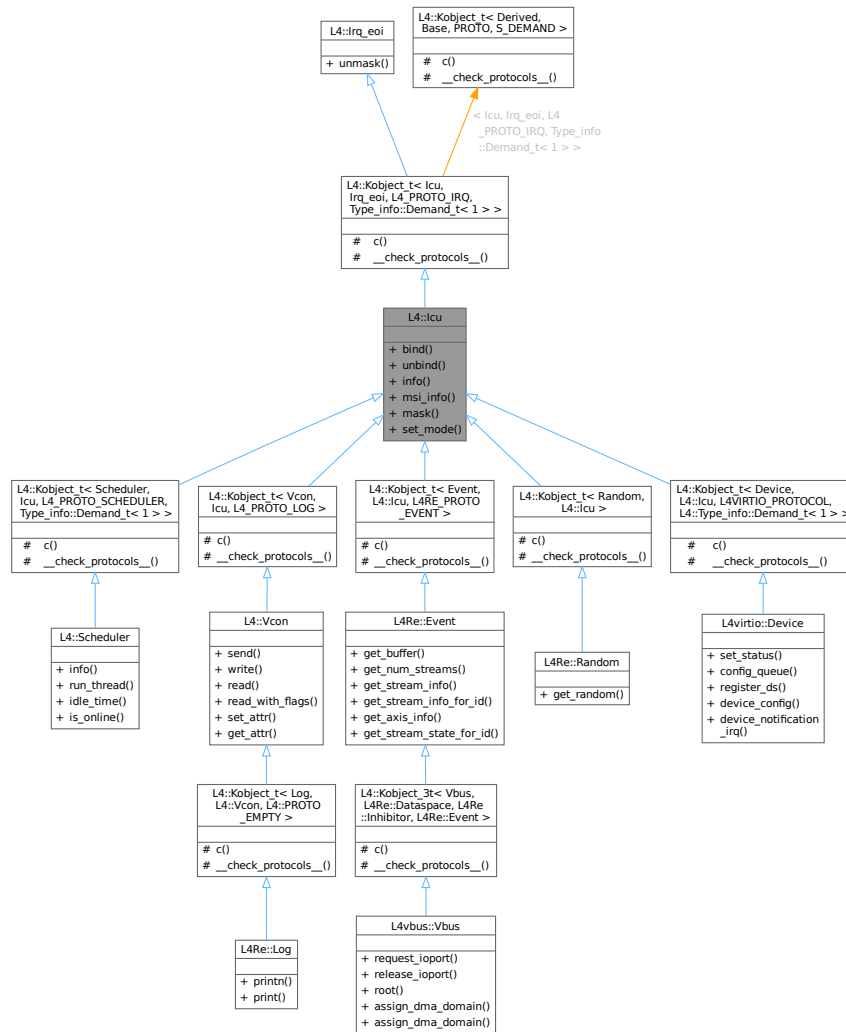
- [l4/sys/factory](#)

**16.108 L4::lcu Class Reference**

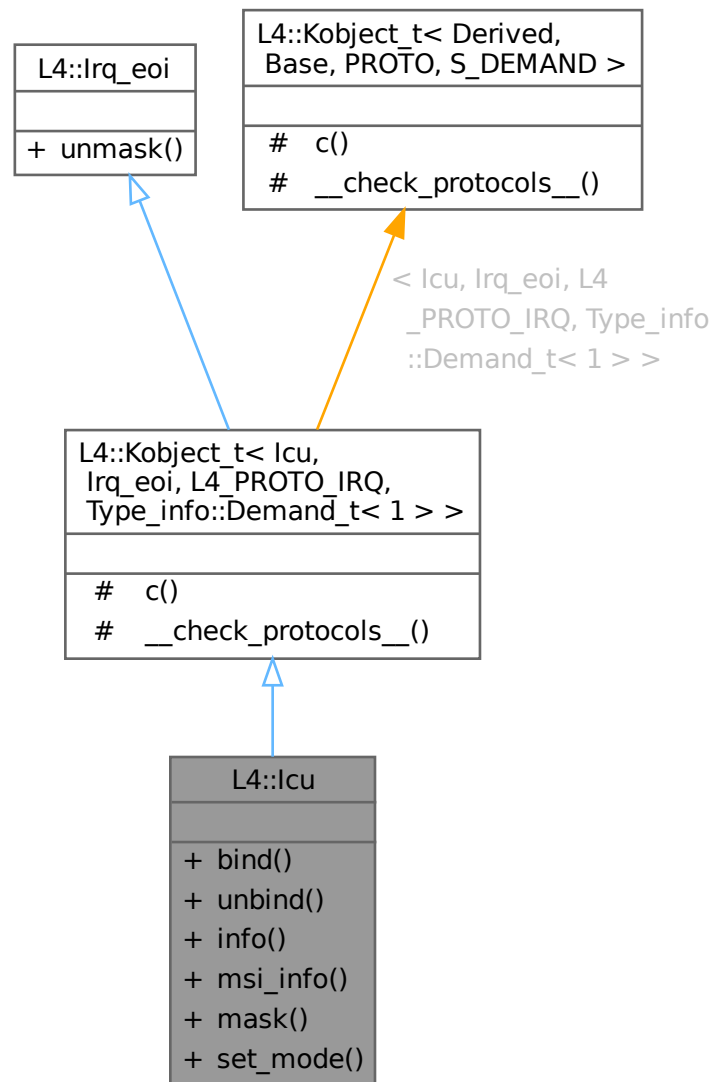
C++ [lcu](#) interface, see [Interrupt controller](#) for the C interface.

```
#include <irq>
```

Inheritance diagram for L4::Icu:



Collaboration diagram for L4::Icu:



## Data Structures

- class [Info](#)

*This class encapsulates information about an ICU.*

## Public Member Functions

- [I4\\_msgtag\\_t bind](#) (unsigned irqnum, [L4::Cap<Triggerable>](#) irq, [I4\\_utcb\\_t](#) \*utcb=[I4\\_utcb\(\)](#)) noexcept  
*Bind an interrupt line of an interrupt controller to an interrupt object.*
- [I4\\_msgtag\\_t unbind](#) (unsigned irqnum, [L4::Cap<Triggerable>](#) irq, [I4\\_utcb\\_t](#) \*utcb=[I4\\_utcb\(\)](#)) noexcept  
*Remove binding of an interrupt line from the interrupt controller object.*

- `l4_msgtag_t info` (`l4_icu_info_t *info`, `l4_utcb_t *utcb=l4_utcb()`) noexcept  
*Get information about the ICU features.*
- `l4_msgtag_t msi_info` (`l4_umword_t irqnum`, `l4_uint64_t source`, `l4_icu_msi_info_t *msi_info`)  
*Get MSI info about IRQ.*
- `l4_msgtag_t mask` (`unsigned irqnum`, `l4_umword_t *label=0`, `l4_timeout_t to=L4_IPC_NEVER`, `l4_utcb_t *utcb=l4_utcb()`) noexcept  
*Mask an IRQ line.*
- `l4_msgtag_t set_mode` (`unsigned irqnum`, `l4_umword_t mode`, `l4_utcb_t *utcb=l4_utcb()`) noexcept  
*Set interrupt mode.*

## Public Member Functions inherited from `L4::Irq_eoi`

- `l4_msgtag_t unmask` (`unsigned irqnum`, `l4_umword_t *label=0`, `l4_timeout_t to=L4_IPC_NEVER`, `l4_utcb_t *utcb=l4_utcb()`) noexcept  
*Unmask the given interrupt line.*

## Additional Inherited Members

## Protected Types inherited from

`L4::Kobject_t< Icu, Irq_eoi, L4_PROTO_IRQ, Type_info::Demand_t< 1 > >`

- typedef `Icu Class`  
*The target interface type (inheriting from `Kobject_t`).*
- typedef `Typeid::Iface< PROTO, Icu > __Iface`  
*The interface description for the derived class.*
- typedef `Typeid::Merge_list< Typeid::Iface_list< __Iface >, typename Irq_eoi::__Iface_list > __Iface_list`  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Member Functions inherited from

`L4::Kobject_t< Icu, Irq_eoi, L4_PROTO_IRQ, Type_info::Demand_t< 1 > >`

- `L4::Cap< Class > c` () const noexcept  
*Get the capability to ourselves.*

## Static Protected Member Functions inherited from

`L4::Kobject_t< Icu, Irq_eoi, L4_PROTO_IRQ, Type_info::Demand_t< 1 > >`

- static void `__check_protocols` () noexcept  
*Helper to check for protocol conflicts.*



## 16.108.1 Detailed Description

C++ [Icu](#) interface, see [Interrupt controller](#) for the C interface.

### Note

"ICU" is short for "interrupt control unit".

This class defines the interface for interrupt controllers. It defines functions for binding [L4::Irq](#) objects to interrupt lines and other interrupt sources, as well as functions for masking and unmasking of interrupts.

To setup an interrupt line the following steps are required:

1. [set\\_mode\(\)](#) (optional if interrupt has a default mode)
2. [L4::Rcv\\_endpoint::bind\\_thread\(\)](#) or [L4::Rcv\\_endpoint::bind\\_snd\\_destination\(\)](#) to attach the [L4::Irq](#) object to a send destination.
3. [bind\(\)](#)
4. [unmask\(\)](#) to receive the first interrupt

For certain interrupt sources only some of these steps are necessary and supported, see [L4::Scheduler](#) and [L4::Vcon](#).

At most one [L4::Irq](#) object can be bound to a certain interrupt source and a certain [L4::Irq](#) object can be bound to at most one interrupt source.

### Include File

```
#include <l4/sys/icu>
```

Definition at line 250 of file [irq](#).

## 16.108.2 Member Function Documentation

### 16.108.2.1 bind()

```
l4_msgtag_t L4::Icu::bind (
    unsigned irqnum,
    L4::Cap< Triggerable > irq,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Bind an interrupt line of an interrupt controller to an interrupt object.

### Parameters

<i>irqnum</i>	IRQ line at the ICU.
<i>irq</i>	IRQ object for the given IRQ line to bind to this ICU.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

### Returns

Syscall return tag. The caller should check the return value using [l4\\_error\(\)](#) to check for errors and to identify the correct method for unmasking the interrupt. Return values < 0 indicate an error. A return value of 0 means a direct unmask via the IRQ object using [L4::Irq::unmask](#). A return value of 1 means that the interrupt has to be unmasked via the ICU using [L4::Icu::unmask](#).

### Return values

<code>-L4_EINVAL</code>	<code>irq</code> is bound to an interrupt source.
<code>-L4_EPERM</code>	Insufficient permissions; see precondition.

### Precondition

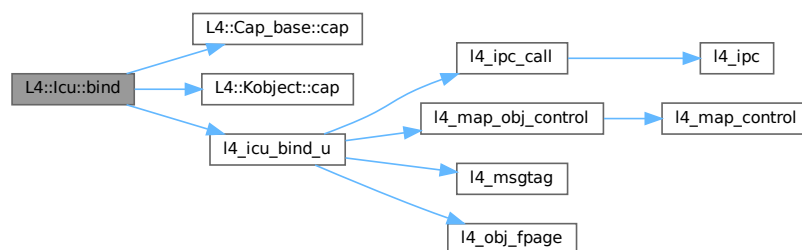
The capability `irq` must have the permission [L4\\_CAP\\_FPAGE\\_W](#).

In case the `irq` is already bound to an interrupt source, it is unbound first. In case the `irq` is bound and the interrupt source is bound to a different [L4::Irq](#) object, only the unbinding happens. An [L4::Irq](#) object that is bound to an interrupt source will get unbound if the [L4::Irq](#) object is deleted.

Definition at line 310 of file [irq](#).

References [L4::Cap\\_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4\\_icu\\_bind\\_u\(\)](#).

Here is the call graph for this function:



### 16.108.2.2 info()

```

l4_msgtag_t L4::Icu::info (
    l4_icu_info_t * info,
    l4_utcb_t * utcb = l4_utcb())  [inline], [noexcept]

```

Get information about the ICU features.

### Parameters

<code>out</code>	<code>info</code>	<a href="#">Info</a> structure to be filled with information.
	<code>utcb</code>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

## Returns

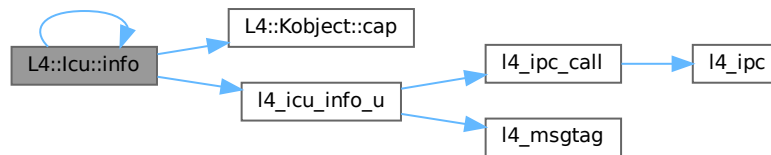
Syscall return tag

Definition at line 345 of file [irq](#).

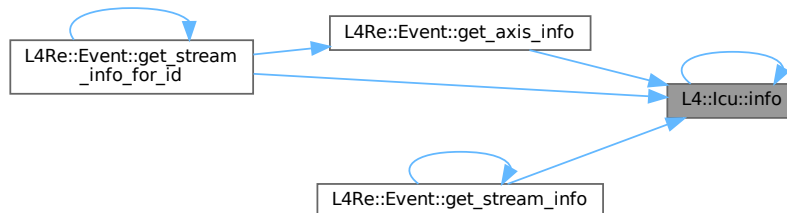
References [L4::Kobject::cap\(\)](#), [info\(\)](#), and [l4\\_icu\\_info\\_u\(\)](#).

Referenced by [L4Re::Event::get\\_axis\\_info\(\)](#), [L4Re::Event::get\\_stream\\_info\(\)](#), [L4Re::Event::get\\_stream\\_info\\_for\\_id\(\)](#), and [info\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.108.2.3 mask()

```

l4_msgtag_t L4::Icu::mask (
    unsigned irqnum,
    l4_umword_t * label = 0,
    l4_timeout_t to = L4_IPC_NEVER,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Mask an IRQ line.

## Parameters

<i>irqnum</i>	IRQ line at the ICU.
---------------	----------------------

<i>label</i>	If NULL, this function is a send-only message to the ICU. If not NULL, this function will enter an open wait after sending the mask message and the received label is returned here.
<i>to</i>	The timeout-pair (send and receive) that shall be used for this operation. The receive timeout is used with a non-NULL <code>label</code> only.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

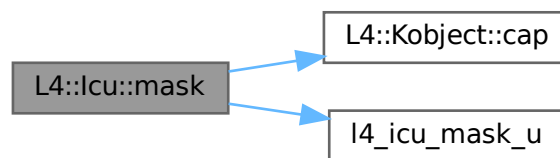
#### Returns

Syscall return tag. If `label` is NULL, this function performs an IPC send-only operation and there is no return value except [L4\\_MSGTAG\\_ERROR](#) indicating success or failure of the send operation. In this case use [l4\\_ipc\\_error\(\)](#) to check for errors and **do not** use [l4\\_error\(\)](#).

Definition at line 393 of file [irq](#).

References [L4::Kobject::cap\(\)](#), [l4\\_icu\\_mask\\_u\(\)](#), and [L4\\_IPC\\_NEVER](#).

Here is the call graph for this function:



#### 16.108.2.4 msi\_info()

```

l4_msgtag_t L4::Icu::msi_info (
    l4_umword_t irqnum,
    l4_uint64_t source,
    l4_icu_msi_info_t * msi_info)
  
```

Get MSI info about IRQ.

#### Parameters

	<i>irqnum</i>	IRQ line at the ICU.
	<i>source</i>	Platform dependent requester ID for MSIs. On IA32 we use a 20bit source filter value as described in the Intel IRQ remapping specification.
out	<i>msi_info</i>	A <a href="#">l4_icu_msi_info_t</a> structure receiving the address and the data value to trigger this MSI.

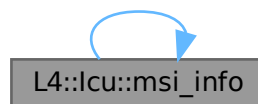
**Returns**

Syscall return tag

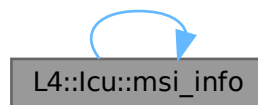
References [msi\\_info\(\)](#).

Referenced by [msi\\_info\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**16.108.2.5 set\_mode()**

```
l4_msgtag_t L4::Icu::set_mode (
    unsigned irqnum,
    l4_unword_t mode,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Set interrupt mode.

**Parameters**

<i>irqnum</i>	IRQ line at the ICU.
<i>mode</i>	Mode, see <a href="#">L4_irq_mode</a> .
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

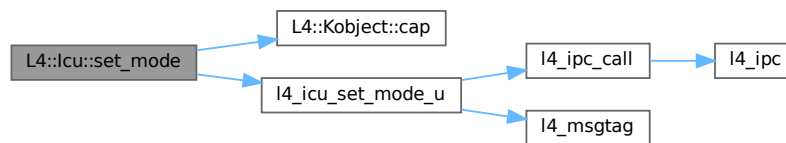
**Returns**

Syscall return tag

Definition at line 421 of file [irq](#).

References [L4::Kobject::cap\(\)](#), and [l4\\_icu\\_set\\_mode\\_u\(\)](#).

Here is the call graph for this function:

**16.108.2.6 unbind()**

```

l4_msgtag_t L4::Icu::unbind (
    unsigned irqnum,
    L4::Cap< Triggerable > irq,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Remove binding of an interrupt line from the interrupt controller object.

**Parameters**

<i>irqnum</i>	IRQ line at the ICU.
<i>irq</i>	IRQ object to remove from the ICU.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

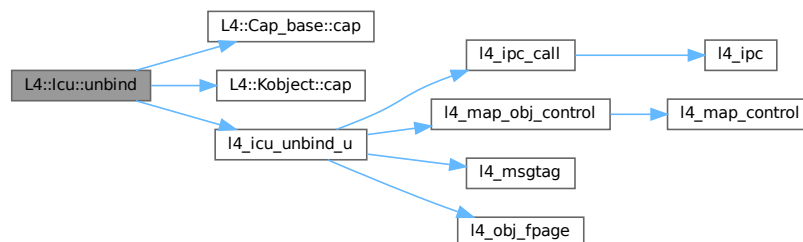
**Returns**

Syscall return tag

Definition at line 328 of file [irq](#).

References [L4::Cap\\_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4\\_icu\\_unbind\\_u\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

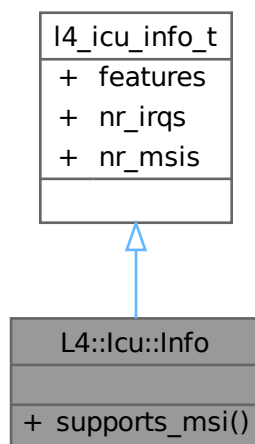
- [l4/sys/irq](#)

## 16.109 L4::lcu::Info Class Reference

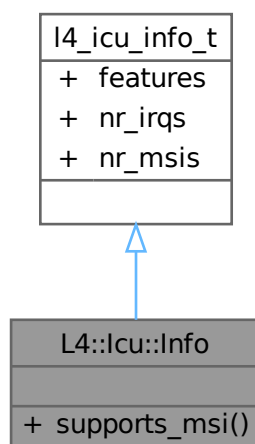
This class encapsulates information about an ICU.

```
#include <irq>
```

Inheritance diagram for L4::lcu::Info:



Collaboration diagram for L4::lcu::Info:



### Public Member Functions

- bool **supports\_msi** () const noexcept  
*True, if the ICU has support for MSIs.*

### Additional Inherited Members

### Data Fields inherited from [l4\\_icu\\_info\\_t](#)

- unsigned [features](#)  
*Feature flags.*
- unsigned **nr\_irqs**  
*The number of IRQ lines supported by the ICU,.*
- unsigned **nr\_msis**  
*The number of MSI vectors supported by the ICU,.*

## 16.109.1 Detailed Description

This class encapsulates information about an ICU.

Definition at line [277](#) of file [irq](#).

The documentation for this class was generated from the following file:

- [l4/sys/irq](#)

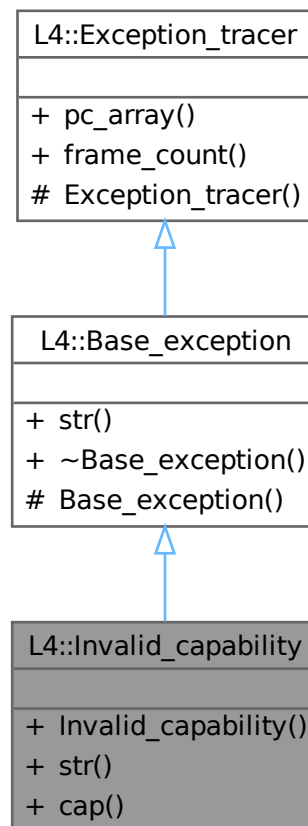
## 16.110 L4::Invalid\_capability Class Reference

Indicates that an invalid object was invoked.

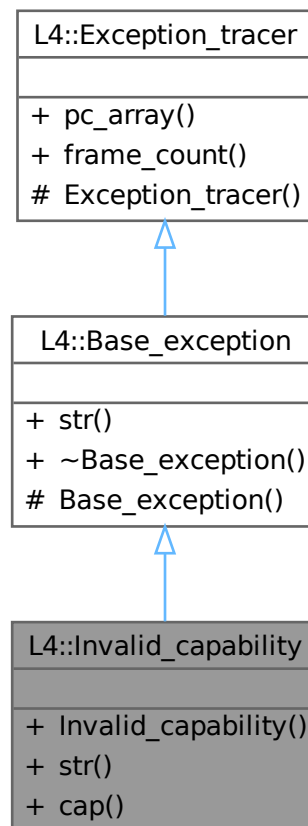
```
#include <l4/cxx/exceptions>
```



Inheritance diagram for L4::Invalid\_capability:



Collaboration diagram for L4::Invalid\_capability:



### Public Member Functions

- [Invalid\\_capability](#) ([Cap](#)< void > const &o) noexcept  
*Create an Invalid\_object exception for the Object o.*
- char const \* **str** () const noexcept override  
*Return a human readable string for the exception.*
- [Cap](#)< void > const & [cap](#) () const noexcept  
*Get the object that caused the error.*

### Public Member Functions inherited from [L4::Base\\_exception](#)

- virtual **~Base\_exception** () noexcept  
*Destruction.*

### Public Member Functions inherited from [L4::Exception\\_tracer](#)

- void const \*const \* **pc\_array** () const noexcept  
*Get the array containing the call trace.*
- int **frame\_count** () const noexcept  
*Get the number of entries that are valid in the call trace.*

## Additional Inherited Members

## Protected Member Functions inherited from [L4::Base\\_exception](#)

- **Base\_exception** () noexcept  
*Create a base exception.*

## Protected Member Functions inherited from [L4::Exception\\_tracer](#)

- **Exception\_tracer** () noexcept  
*Create a back trace.*

### 16.110.1 Detailed Description

Indicates that an invalid object was invoked.

An Object is invalid if it has L4\_INVALID\_ID as server [L4](#) UID, or if the server does not know the object ID.

Definition at line [234](#) of file [exceptions](#).

### 16.110.2 Constructor & Destructor Documentation

#### 16.110.2.1 Invalid\_capability()

```
L4::Invalid_capability::Invalid_capability (
    Cap< void > const & o) [inline], [explicit], [noexcept]
```

Create an Invalid\_object exception for the Object o.

#### Parameters

<i>o</i>	The object that caused the server side error.
----------	-----------------------------------------------

Definition at line [244](#) of file [exceptions](#).

### 16.110.3 Member Function Documentation

#### 16.110.3.1 cap()

```
Cap< void > const & L4::Invalid_capability::cap () const [inline], [noexcept]
```

Get the object that caused the error.

#### Returns

The object that caused the error on invocation.

Definition at line [253](#) of file [exceptions](#).

The documentation for this class was generated from the following file:

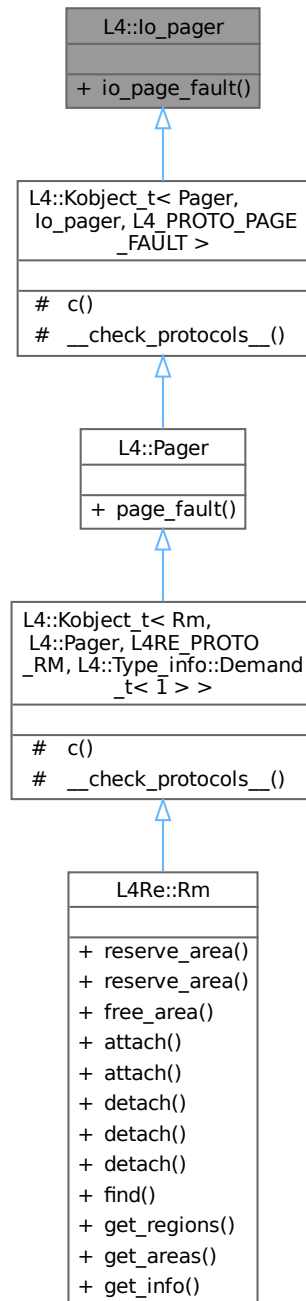
- [l4/cxx/exceptions](#)

## 16.111 L4::lo\_pager Class Reference

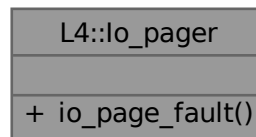
[lo\\_pager](#) interface.

```
#include <pager>
```

Inheritance diagram for L4::lo\_pager:



Collaboration diagram for L4::io\_pager:



### Public Member Functions

- [l4\\_msgtag\\_t](#) [io\\_page\\_fault](#) ([l4\\_fpage\\_t](#) io\_pfa, [l4\\_umword\\_t](#) pc, [L4::lpc::Rcv\\_fpage](#) rwin, [L4::lpc::Opt<L4::lpc::Snd\\_fpage & >](#) fp)

*IO page fault protocol message.*

### 16.111.1 Detailed Description

[io\\_pager](#) interface.

#### Note

This interface is IA32 specific.

This class defines the interface for handling IO page faults. IO page faults happen when a thread tries to access an IO port that it does not currently have access to.

Depending on the microkernel's implementation, IO page faults can be handled in two ways.

If the microkernel does not support IO page faults, this IO pagefault interface is not used. Instead, the microkernel sends an exception IPC to the thread's exception handler ([L4::Exception](#)), indicating a GP (exception number 13). The exception handler must consult the faulting instruction to determine the cause of the exception. This is the default in Fiasco.OC.

In contrast, if the microkernel supports IO page faults, the microkernel will generate an IO page fault message and send it to the thread's page fault handler (pager). The page fault handler can implement this interface to handle the IO page faults.

#### Note

A program may use this mechanism to implement a lazy IO port access scheme.

The page fault and exception handlers are set with the [L4::Thread::control](#) interface.

Definition at line 50 of file [pager](#).

## 16.111.2 Member Function Documentation

### 16.111.2.1 io\_page\_fault()

```
l4_msgtag_t L4::Io_pager::io_page_fault (
    l4_fpage_t io_pfa,
    l4_umword_t pc,
    L4::Ipc::Rcv_fpage rwin,
    L4::Ipc::Opt< L4::Ipc::Snd_fpage & > fp)
```

IO page fault protocol message.

#### Parameters

---

	<i>io_pfa</i>	Flexpage describing the faulting IO-port.
	<i>pc</i>	Faulting program counter.
	<i>rwin</i>	The receive window for a flexpage mapping.
out	<i>fp</i>	Optional: flexpage descriptor to send to the task raising the page fault.

#### Returns

System call message tag; use [l4\\_error\(\)](#) to check for errors.

IO-port fault messages are usually generated by the kernel and an IO-page-fault handler needs to be in place to handle such faults and generate a reply, potentially filling in `fp`.

The documentation for this class was generated from the following file:

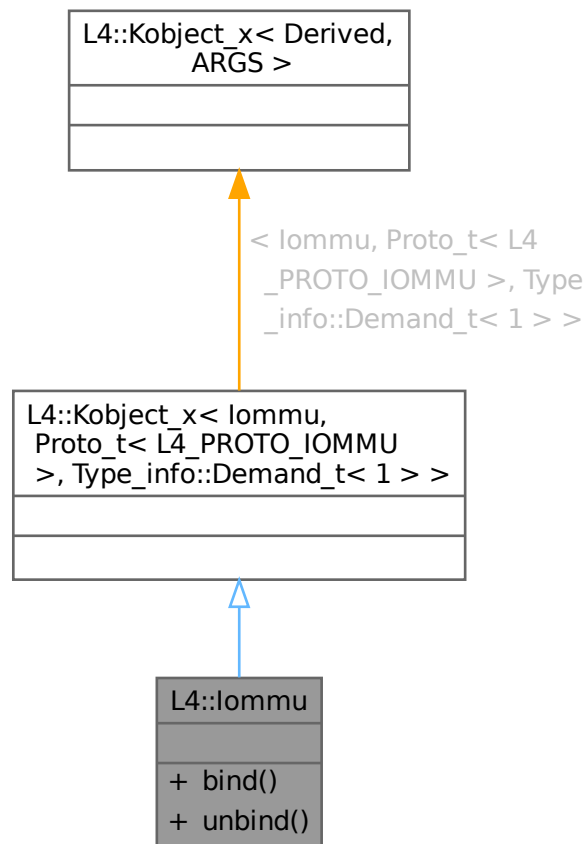
- [l4/sys/pager](#)

## 16.112 L4::lommu Class Reference

Interface for IO-MMUs used for DMA remapping.

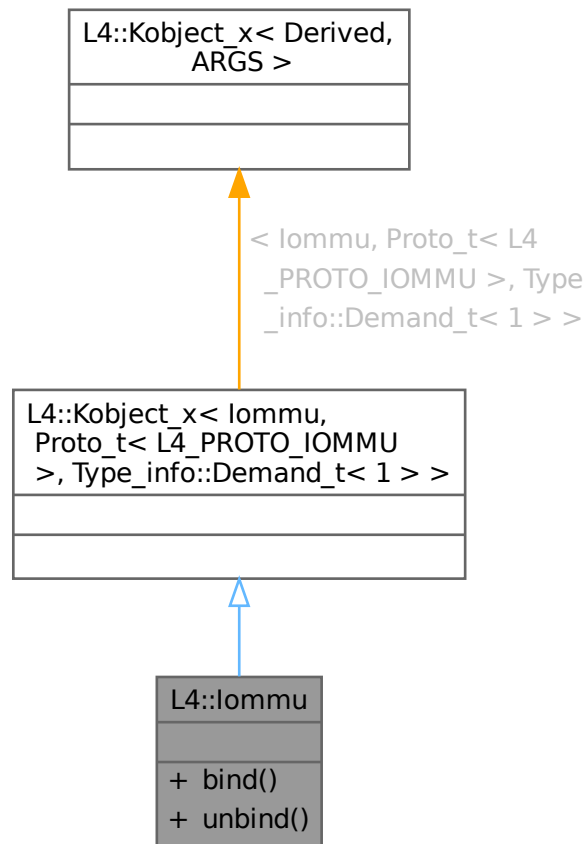
```
#include <iommu>
```

Inheritance diagram for L4::lomu:





Collaboration diagram for L4::lommu:



### Public Member Functions

- `l4_msgtag_t bind (l4_uint64_t src_id, lpc::Cap< Task > dma_space)`  
Associate *dma\_space* with the set of device(s) specified by *src\_id*.
- `l4_msgtag_t unbind (l4_uint64_t src_id, lpc::Cap< Task > dma_space)`  
Remove the association of the given DMA address space from the device(s) specified by *src\_id*.

### 16.112.1 Detailed Description

Interface for IO-MMUs used for DMA remapping.

#### Note

This interface is only present in the kernel if the kernel detected an IOMMU during boot.

This interface allows to associate a DMA address space with a platform dependent set of devices. The kernel automatically keeps the memory spaces of associated DMA spaces in sync with the respective page table structures in the IOMMU.

Definition at line 21 of file [iommu](#).

## 16.112.2 Member Function Documentation

### 16.112.2.1 bind()

```
l4_msgtag_t L4::Iommu::bind (
    l4_uint64_t src_id,
    Ipc::Cap< Task > dma_space)
```

Associate `dma_space` with the set of device(s) specified by `src_id`.

Updates the respective page table structures in the IOMMU and keeps them in sync when memory is mapped to the `dma_space` or revoked from it.

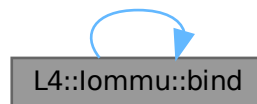
#### Parameters

<i>src_id</i>	Platform dependent source ID specifying the set of devices that shall use <code>dma_space</code> for DMA remapping.
<i>dma_space</i>	The DMA space ( <a href="#">L4::Task</a> created with <code>L4_PROTO_DMA_SPACE</code> ) providing the mappings that shall be used for the device(s).

References [bind\(\)](#).

Referenced by [bind\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



16.112.2.2 `unbind()`

```
l4_msgtag_t L4::Iommu::unbind (
    l4_uint64_t src_id,
    Ipc::Cap< Task > dma_space)
```

Remove the association of the given DMA address space from the device(s) specified by `src_id`.

Clear the respective page stable structures in the IOMMU.

**Parameters**

<code>src_id</code>	Platform dependent source ID specifying the set of devices that shall no longer use <code>dma_space</code> for DMA remapping.
<code>dma_space</code>	The DMA space formerly associated with <code>bind()</code> .

References `unbind()`.

Referenced by `unbind()`.

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

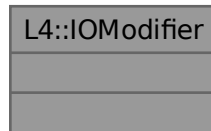
- `l4/sys/iommu`

## 16.113 L4::IOModifier Class Reference

Modifier class for the IO stream.

```
#include <basic_ostream>
```

Collaboration diagram for L4::IOModifier:



### 16.113.1 Detailed Description

Modifier class for the IO stream.

An IO Modifier can be used to change properties of an IO stream for example the number format.

Definition at line 22 of file [basic\\_ostream](#).

The documentation for this class was generated from the following file:

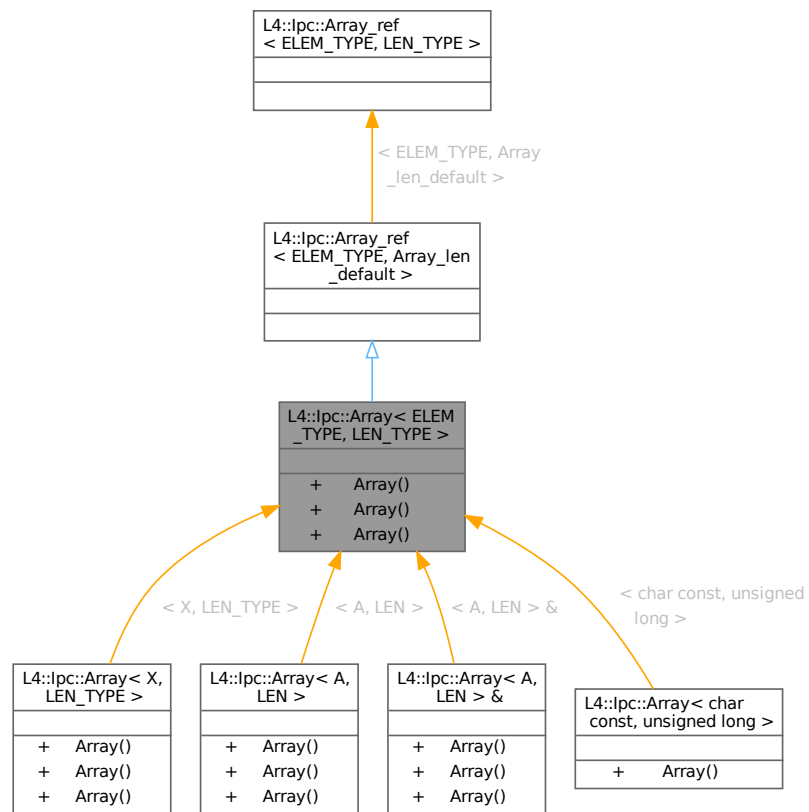
- [l4/cxx/basic\\_ostream](#)

## 16.114 L4::lpc::Array< ELEM\_TYPE, LEN\_TYPE > Struct Template Reference

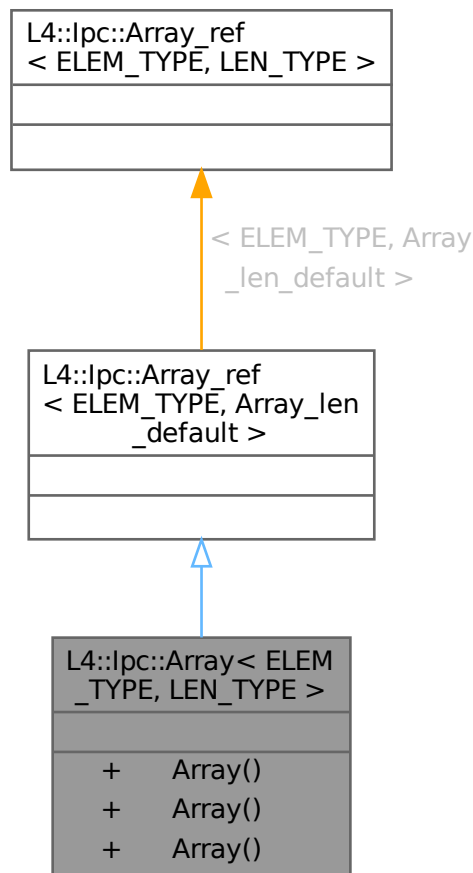
[Array](#) data type for dynamically sized arrays in RPCs.

```
#include <ipc_array>
```

Inheritance diagram for L4::lpc::Array< ELEM\_TYPE, LEN\_TYPE >:



Collaboration diagram for L4::lpc::Array< ELEM\_TYPE, LEN\_TYPE >:



## Public Member Functions

- **Array ()**  
*Make array.*
- **Array (LEN\_TYPE length, ELEM\_TYPE \*data)**  
*Make array from length and data pointer.*
- **Array (typename Non\_const< ELEM\_TYPE >::type const &other)**  
*Make a const array from a non-const array.*

## 16.114.1 Detailed Description

```
template<typename ELEM_TYPE, typename LEN_TYPE = Array_len_default>
struct L4::lpc::Array< ELEM_TYPE, LEN_TYPE >
```

[Array](#) data type for dynamically sized arrays in RPCs.

## Template Parameters

<i>ELEM_TYPE</i>	The data type of an array element, should be 'const' when used as input.
<i>LEN_TYPE</i>	Data type used to store the number of elements in the array.

An [Array](#) generally encapsulates a data pointer and a length (number of elements). [Array](#) does *not* provide any storage for the data itself. The storage is either provided by a client-side caller or in the case of [Array\\_ref](#) is the message itself.

Arrays can be used as input or as output arguments, when used as input *ELEM\_TYPE* should be qualified *const*, when used as output a reference to an array must be used and the *ELEM\_TYPE* must *not* be qualified *const*. It is the caller's responsibility to provide an array buffer of sufficient length. If a message from the server is too large it will be silently truncated.

If backward compatibility with `lpc::Stream` is required, then *LEN\_TYPE* must be `unsigned long`.

Definition at line 81 of file [ipc\\_array](#).

The documentation for this struct was generated from the following file:

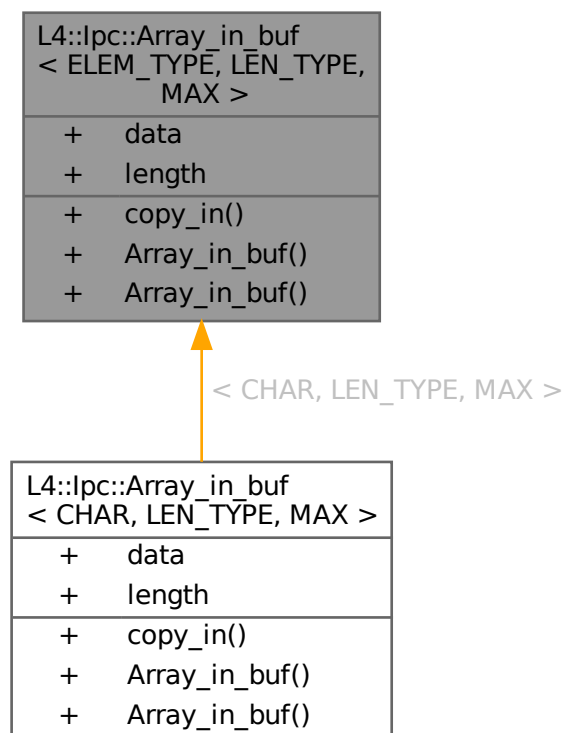
- `l4/sys/cxx/ipc_array`

## 16.115 L4::lpc::Array\_in\_buf< ELEM\_TYPE, LEN\_TYPE, MAX > Struct Template Reference

Server-side copy in buffer for [Array](#).

```
#include <ipc_array>
```

Inheritance diagram for `L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX >`:



Collaboration diagram for L4::lpc::Array\_in\_buf< ELEM\_TYPE, LEN\_TYPE, MAX >:

L4::lpc::Array_in_buf < ELEM_TYPE, LEN_TYPE, MAX >	
+	data
+	length
+	copy_in()
+	Array_in_buf()
+	Array_in_buf()

## Public Member Functions

- void **copy\_in** ([const\\_array](#) a)  
*copy in data from a source array*
- **Array\_in\_buf** ([const\\_array](#) a)  
*Make [Array\\_in\\_buf](#) from a const array.*
- **Array\_in\_buf** ([array](#) a)  
*Make [Array\\_in\\_buf](#) from a non-const array.*

## Data Fields

- ELEM\_TYPE **data** [MAX]  
*The data elements.*
- LEN\_TYPE **length**  
*The length of the array.*

### 16.115.1 Detailed Description

```
template<typename ELEM_TYPE, typename LEN_TYPE = Array_len_default, LEN_TYPE MAX = (L4_↔
UTCB_GENERIC_DATA_SIZE * sizeof(I4_umword_t)) / sizeof(ELEM_TYPE)>
struct L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX >
```

Server-side copy in buffer for [Array](#).

## Template Parameters

<i>ELEM_TYPE</i>	Data type of an array element.
<i>LEN_TYPE</i>	Data type for the number of elements in the array.



<b>MAX</b>	The maximum number of elements in the buffer. If the actual message is longer than the buffer, it will be silently truncated.
------------	-------------------------------------------------------------------------------------------------------------------------------

This type is assignment compatible to [Array\\_ref<ELEM\\_TYPE, LEN\\_TYPE>](#) and provides a transparent server-side copy-in mechanism for array parameters. The [Array\\_in\\_buf](#) provides the storage for the array data and receives a copy of the data passed to the server-function.

Definition at line 126 of file [ipc\\_array](#).

The documentation for this struct was generated from the following file:

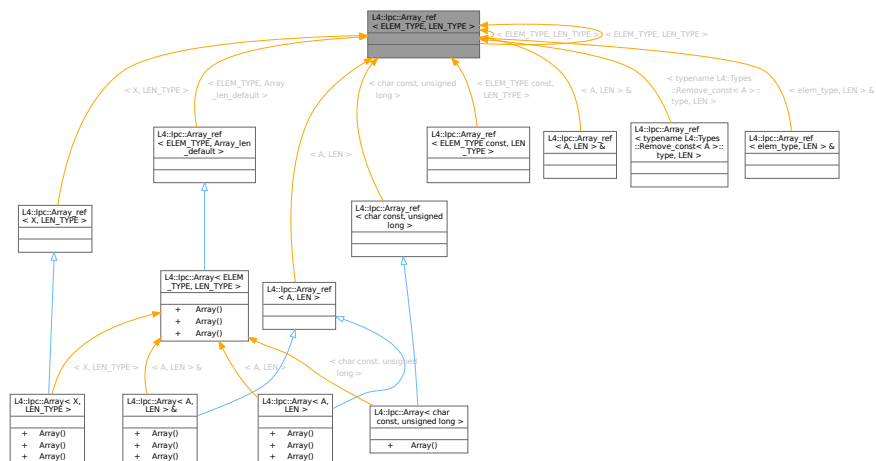
- [l4/sys/cxx/ipc\\_array](#)

## 16.116 L4::lpc::Array\_ref< ELEM\_TYPE, LEN\_TYPE > Struct Template Reference

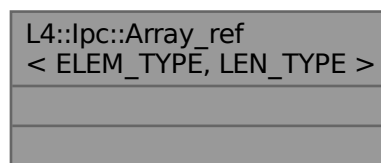
[Array](#) reference data type for arrays located in the message.

```
#include <ipc_array>
```

Inheritance diagram for L4::lpc::Array\_ref< ELEM\_TYPE, LEN\_TYPE >:



Collaboration diagram for L4::lpc::Array\_ref< ELEM\_TYPE, LEN\_TYPE >:



### 16.116.1 Detailed Description

```
template<typename ELEM_TYPE, typename LEN_TYPE = Array_len_default>
struct L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE >
```

[Array](#) reference data type for arrays located in the message.

#### Note

Use [Array](#) for normal RPC interfaces, [Array\\_ref](#) is usually used as server-side argument, see [Array](#).

#### Template Parameters

<i>ELEM_TYPE</i>	The data type of an array element, should be 'const' when used as input.
<i>LEN_TYPE</i>	Data type used to store the number of elements in the array.

Definition at line 28 of file [ipc\\_array](#).

The documentation for this struct was generated from the following file:

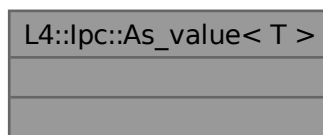
- `l4/sys/cxx/ipc_array`

## 16.117 L4::lpc::As\_value< T > Struct Template Reference

Pass the argument as plain data value.

```
#include <ipc_types>
```

Collaboration diagram for L4::lpc::As\_value< T >:



### 16.117.1 Detailed Description

```
template<typename T>
struct L4::lpc::As_value< T >
```

Pass the argument as plain data value.

#### Template Parameters

<i>T</i>	The type of the original argument.
----------	------------------------------------

[As\\_value<T>](#) is used when *T* would be otherwise interpreted specially, for example as flexpage. When using [As\\_value<>](#) then the argument is transmitted as plain data element.

Definition at line 116 of file [ipc\\_types](#).

The documentation for this struct was generated from the following file:

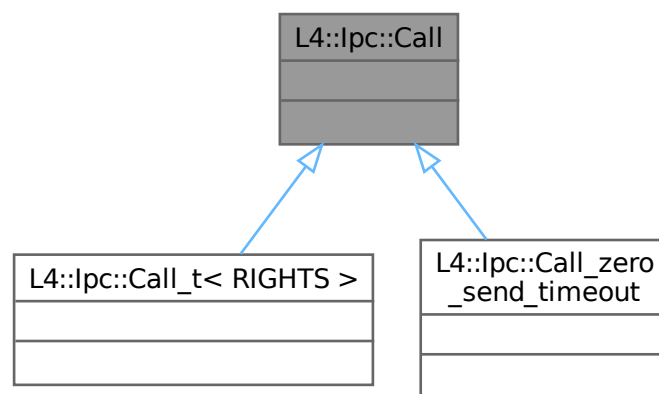
- [l4/sys/cxx/ipc\\_types](#)

## 16.118 L4::lpc::Call Struct Reference

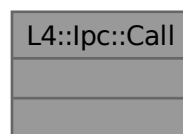
RPC attribute for a standard RPC call.

```
#include <ipc_iface>
```

Inheritance diagram for L4::lpc::Call:



Collaboration diagram for L4::lpc::Call:



### 16.118.1 Detailed Description

RPC attribute for a standard RPC call.

This is the default for the *FLAGS* parameter for `L4::lpc::Msg::Rpc_call` `L4::lpc::Msg::Rpc_inline_call` templates and declares the RPC to have default call semantics and timeouts.

Examples:

```
L4_RPC(long, send, (unsigned value), L4::Ipc::Call);
```

which is equivalent to:

```
L4_RPC(long, send, (unsigned value));
```

Definition at line 239 of file [ipc\\_iface](#).

The documentation for this struct was generated from the following file:

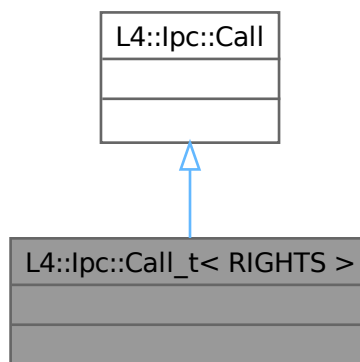
- [l4/sys/cxx/ipc\\_iface](#)

## 16.119 L4::lpc::Call\_t< RIGHTS > Struct Template Reference

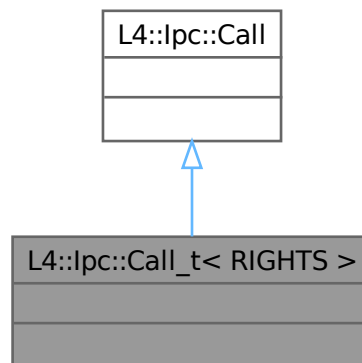
RPC attribute for an RPC call with required rights.

```
#include <ipc_iface>
```

Inheritance diagram for `L4::lpc::Call_t< RIGHTS >`:



Collaboration diagram for L4::lpc::Call\_t< RIGHTS >:



### 16.119.1 Detailed Description

```
template<unsigned RIGHTS>
struct L4::lpc::Call_t< RIGHTS >
```

RPC attribute for an RPC call with required rights.

#### Template Parameters

<i>RIGHTS</i>	The capability rights required for this call. <a href="#">L4_CAP_FPAGE_W</a> and <a href="#">L4_CAP_FPAGE_S</a> are checked within the server (and <a href="#">-L4_EPERM</a> shall be returned if the caller has insufficient rights). <a href="#">L4_CAP_FPAGE_R</a> is always on but might be specified for documentation purposes. Other rights cannot be used in this context, because they cannot be checked at the server side.
---------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Examples:

```
L4_RPC(long, func, (unsigned value), L4::lpc::Call_t<L4_CAP_FPAGE_RW>);
```

Definition at line 270 of file [ipc\\_iface](#).

The documentation for this struct was generated from the following file:

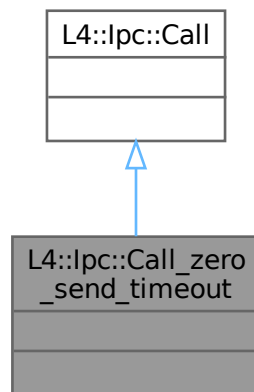
- [l4/sys/cxx/ipc\\_iface](#)

## 16.120 L4::lpc::Call\_zero\_send\_timeout Struct Reference

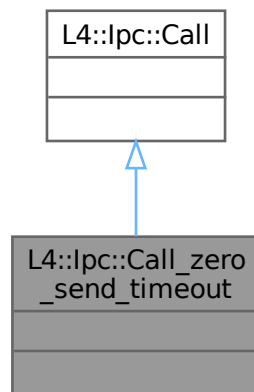
RPC attribute for an RPC call, with zero send timeout.

```
#include <ipc_iface>
```

Inheritance diagram for L4::lpc::Call\_zero\_send\_timeout:



Collaboration diagram for L4::lpc::Call\_zero\_send\_timeout:



### 16.120.1 Detailed Description

RPC attribute for an RPC call, with zero send timeout.

Definition at line 249 of file [ipc\\_iface](#).

The documentation for this struct was generated from the following file:

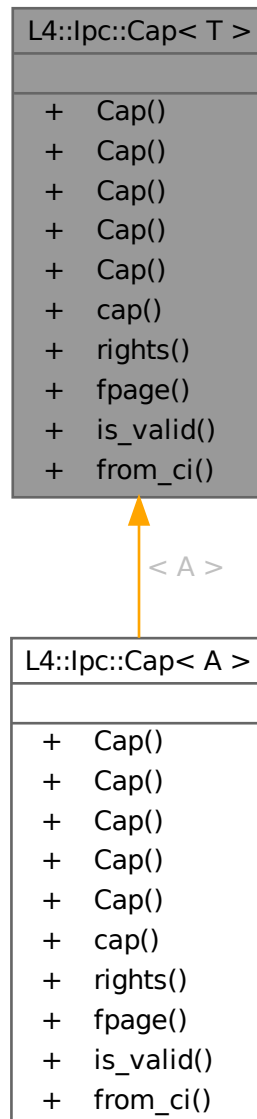
- [l4/sys/cxx/ipc\\_iface](#)

## 16.121 L4::lpc::Cap< T > Class Template Reference

Capability type for RPC interfaces (see [L4::Cap<T>](#)).

```
#include <ipc_types>
```

Inheritance diagram for L4::lpc::Cap< T >:



Collaboration diagram for L4::lpc::Cap< T >:

L4::lpc::Cap< T >
<ul style="list-style-type: none"> <li>+ Cap()</li> <li>+ Cap()</li> <li>+ Cap()</li> <li>+ Cap()</li> <li>+ Cap()</li> <li>+ cap()</li> <li>+ rights()</li> <li>+ fpage()</li> <li>+ is_valid()</li> <li>+ from_ci()</li> </ul>

## Public Types

- enum { [Rights\\_mask](#) = 0xff , [Cap\\_mask](#) = L4\_CAP\_MASK }

## Public Member Functions

- template<typename O>  
**Cap** ([Cap](#)< O > const &o) noexcept  
*Make copy with conversion.*
- Cap** ([L4::Cap](#)< T > [cap](#)) noexcept  
*Make a [Cap](#) from [L4::Cap<T>](#), with minimal rights.*
- template<typename O>  
**Cap** ([L4::Cap](#)< O > [cap](#)) noexcept  
*Make IPC [Cap](#) from [L4::Cap](#) with conversion (and minimal rights).*
- Cap** () noexcept  
*Make an invalid cap.*
- Cap** ([L4::Cap](#)< T > [cap](#), unsigned char [rights](#)) noexcept  
*Make a [Cap](#) from [L4::Cap<T>](#) with the given rights.*
- [L4::Cap](#)< T > **cap** () const noexcept  
*Return the [L4::Cap<T>](#) of this [Cap](#).*
- unsigned **rights** () const noexcept  
*Return the rights bits stored in this IPC cap.*
- [L4::lpc::Snd\\_fpage](#) **fpage** () const noexcept  
*Return the send flexpage for this [Cap](#) (see [l4\\_fpage\\_t](#)).*
- bool **is\_valid** () const noexcept  
*Return true if this [Cap](#) is valid.*



### Static Public Member Functions

- static [Cap from\\_ci](#) ([l4\\_cap\\_idx\\_t](#) c) noexcept  
*Create an IPC capability from a C capability index plus rights.*

## 16.121.1 Detailed Description

```
template<typename T>
class L4::lpc::Cap< T >
```

Capability type for RPC interfaces (see [L4 : :Cap<T>](#)).

### Template Parameters

<i>T</i>	type of the interface referenced by the capability.
----------	-----------------------------------------------------

In contrast to [L4 : :Cap<T>](#) this type additionally stores a rights mask that shall be used when the capability is transferred to the receiver. This allows to apply restrictions to the transferred capability in the form of a subset of the rights possessed by the sender.

See also

[L4::lpc::make\\_cap\(\)](#)

Definition at line 698 of file [ipc\\_types](#).

## 16.121.2 Member Enumeration Documentation

### 16.121.2.1 anonymous enum

```
template<typename T>
anonymous enum
```

#### Enumerator

Rights_mask	Mask for rights bits stored internally. <a href="#">L4_FPAGE_RIGHTS_MASK</a>   <a href="#">L4_FPAGE_C_NO_REF_CNT</a>   <a href="#">L4_FPAGE_C_OBJ_RIGHTS</a> ).
Cap_mask	Mask for significant capability bits. (incl. the invalid bit to support invalid caps)

Definition at line 704 of file [ipc\\_types](#).

## 16.121.3 Constructor & Destructor Documentation

### 16.121.3.1 Cap()

```
template<typename T>
L4::lpc::Cap< T >::Cap (
    L4::Cap< T > cap,
    unsigned char rights) [inline], [noexcept]
```

Make a [Cap](#) from [L4::Cap<T>](#) with the given rights.

#### Parameters

<i>cap</i>	Capability to be sent.
<i>rights</i>	Rights to be sent. Consists of <a href="#">L4_fpage_rights</a> and <a href="#">L4_obj_fpage_ctl</a> .

Definition at line [750](#) of file [ipc\\_types](#).

## 16.121.4 Member Function Documentation

### 16.121.4.1 from\_ci()

```
template<typename T>
Cap L4::Ipc::Cap< T >::from_ci (
    l4_cap_idx_t c) [inline], [static], [noexcept]
```

Create an IPC capability from a C capability index plus rights.

#### Parameters

<i>c</i>	C capability index with the lowest 8 bits used as rights for the map operation (see <a href="#">L4_fpage_rights</a> ).
----------	------------------------------------------------------------------------------------------------------------------------

Definition at line [758](#) of file [ipc\\_types](#).

The documentation for this class was generated from the following file:

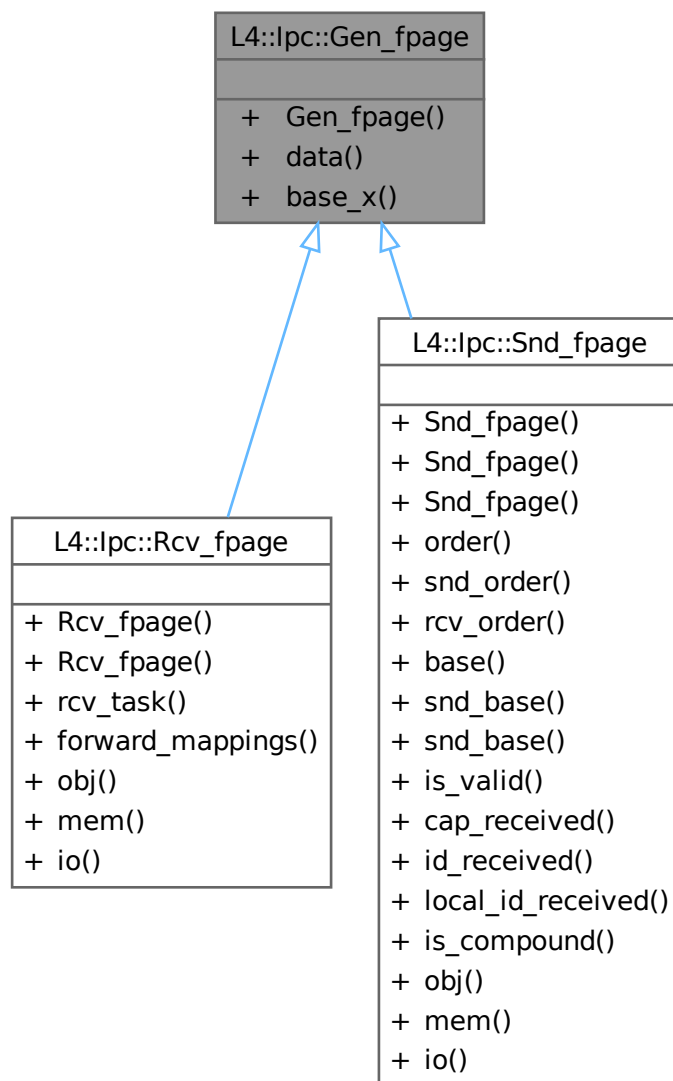
- [l4/sys/cxx/ipc\\_types](#)

## 16.122 L4::Ipc::Gen\_fpage Class Reference

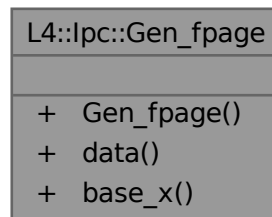
Generic RPC base for typed message items.

```
#include <ipc_types>
```

Inheritance diagram for L4::lpc::Gen\_fpage:



Collaboration diagram for L4::Ipc::Gen\_fpage:



## Public Types

- enum [Type](#) { [Special](#) = L4\_FPAGE\_SPECIAL << 4 , [Memory](#) = L4\_FPAGE\_MEMORY << 4 , [Io](#) = L4\_FPAGE\_IO << 4 , [Obj](#) = L4\_FPAGE\_OBJ << 4 }
- Type of mapping object, see [L4\\_fpage\\_type](#).*

## Public Member Functions

- **Gen\_fpage** ([l4\\_umword\\_t](#) base, [l4\\_umword\\_t](#) data) noexcept  
*Construct from raw values.*
- [l4\\_umword\\_t](#) **data** () const noexcept  
*Return the raw flexpage descriptor.*
- [l4\\_umword\\_t](#) **base\_x** () const noexcept  
*Return the raw base descriptor.*

### 16.122.1 Detailed Description

Generic RPC base for typed message items.

Definition at line 286 of file [ipc\\_types](#).

### 16.122.2 Member Enumeration Documentation

#### 16.122.2.1 Type

enum [L4::Ipc::Gen\\_fpage::Type](#)

[Type](#) of mapping object, see [L4\\_fpage\\_type](#).

## Enumerator

Special	Special flexpage, either <a href="#">l4_fpage_invalid()</a> or <a href="#">l4_fpage_all()</a> ; only supported by selected interfaces.
Memory	Flexpage for memory spaces.
Io	Flexpage for I/O port spaces.
Obj	Flexpage for object spaces.

Definition at line [290](#) of file [ipc\\_types](#).

The documentation for this class was generated from the following file:

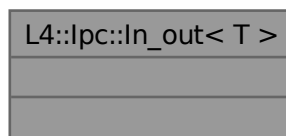
- [l4/sys/cxx/ipc\\_types](#)

## 16.123 L4::lpc::In\_out< T > Struct Template Reference

Mark an argument as in-out argument.

```
#include <ipc_types>
```

Collaboration diagram for L4::lpc::In\_out< T >:



### 16.123.1 Detailed Description

```
template<typename T>
struct L4::lpc::In_out< T >
```

Mark an argument as in-out argument.

#### Template Parameters

<i>T</i>	The original argument type, usually a pointer or a reference.
----------	---------------------------------------------------------------

[In\\_out<>](#) is used when an otherwise output-only value shall also be used as input value.

Definition at line [41](#) of file [ipc\\_types](#).

The documentation for this struct was generated from the following file:

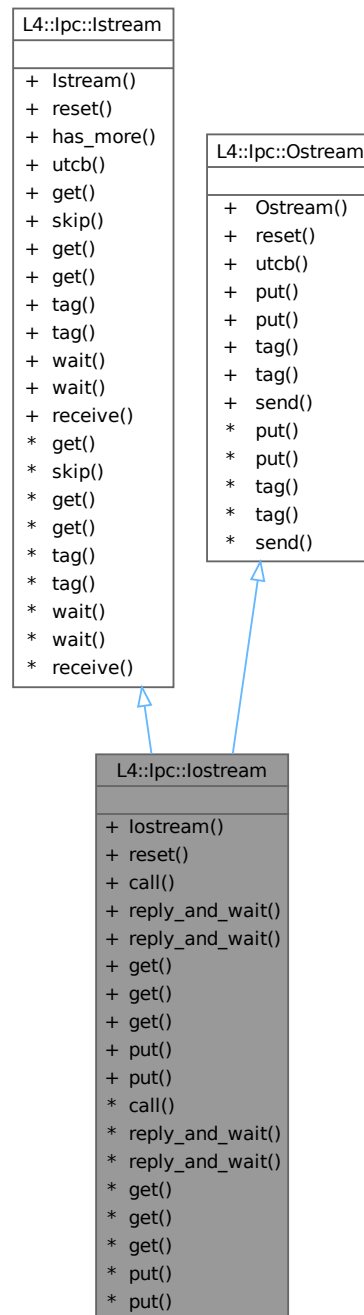
- [l4/sys/cxx/ipc\\_types](#)

## 16.124 L4::lpc::lostream Class Reference

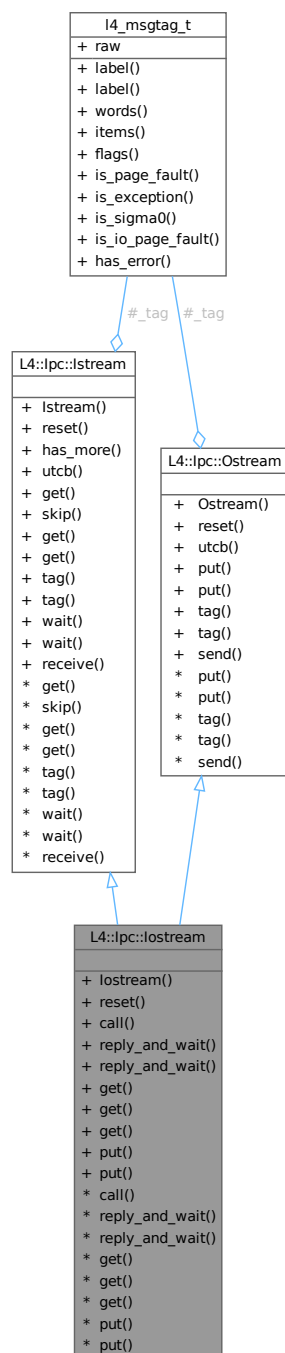
Input/Output stream for IPC [un]marshalling.

```
#include <ipc_stream>
```

Inheritance diagram for L4::lpc::lostream:



Collaboration diagram for L4::lpc::lostream:



## Public Member Functions

- `lostream (l4_utcb_t *utcb)`  
Create an IPC IO stream with a single message buffer.
- `void reset ()`  
Reset the stream to its initial state.

**IPC operations.**

- `l4_msgtag_t call (l4_cap_idx_t dst, l4_timeout_t timeout, long proto=0)`  
*Do an IPC call using the message in the output stream and receive the reply in the input stream.*
- `l4_msgtag_t reply_and_wait (l4_umword_t *src_dst, long proto=0)`  
*Do an IPC reply and wait.*
- `l4_msgtag_t reply_and_wait (l4_umword_t *src_dst, l4_timeout_t timeout, long proto=0)`  
*Do an IPC reply and wait.*

**Get/Put functions.**

*These functions are basically used to implement the insertion operators (<<) and should not be called directly.*

- `template<typename T>`  
`unsigned long get (T *buf, unsigned long elems)`  
*Copy out an array of type T with size elements.*
- `template<typename T>`  
`unsigned long get (Msg_ptr< T > const &buf, unsigned long elems=1)`  
*Read one size elements of type T from the stream and return a pointer.*
- `template<typename T>`  
`bool get (T &v)`  
*Extract a single element of type T from the stream.*
- `template<typename T>`  
`bool put (T *buf, unsigned long size)`  
*Put an array with size elements of type T into the stream.*
- `template<typename T>`  
`bool put (T const &v)`  
*Insert an element of type T into the stream.*

**Public Member Functions inherited from `L4::lpc::lstream`**

- `lstream (l4_utcb_t *utcb)`  
*Create an input stream for the given message buffer.*
- `void reset ()`  
*Reset the stream to empty, and ready for `receive()/wait()`.*
- `template<typename T>`  
`bool has_more (unsigned long count=1)`  
*Check whether a value of type T can be obtained from the stream.*
- `l4_utcb_t * utcb () const`  
*Return utcb pointer.*
- `template<typename T>`  
`unsigned long get (T *buf, unsigned long elems)`  
*Copy out an array of type T with size elements.*
- `template<typename T>`  
`void skip (unsigned long elems)`  
*Skip size elements of type T in the stream.*
- `template<typename T>`  
`unsigned long get (Msg_ptr< T > const &buf, unsigned long elems=1)`  
*Read one size elements of type T from the stream and return a pointer.*
- `template<typename T>`  
`bool get (T &v)`  
*Extract a single element of type T from the stream.*
- `l4_msgtag_t tag () const`



- Get the message tag of a received IPC.  
 • [l4\\_msgtag\\_t & tag \(\)](#)  
 Get the message tag of a received IPC.
- [l4\\_msgtag\\_t wait \(l4\\_umword\\_t \\*src\)](#)  
 Wait for an incoming message from any sender.
- [l4\\_msgtag\\_t wait \(l4\\_umword\\_t \\*src, l4\\_timeout\\_t timeout\)](#)  
 Wait for an incoming message from any sender.
- [l4\\_msgtag\\_t receive \(l4\\_cap\\_idx\\_t src\)](#)  
 Wait for a message from the specified sender.

## Public Member Functions inherited from L4::lpc::Ostream

- **Ostream** ([l4\\_utcb\\_t \\*utcb](#))  
 Create an IPC output stream using the given message buffer *utcb*.
- void **reset** ()  
 Reset the stream to empty, same state as a newly created stream.
- [l4\\_utcb\\_t \\* utcb](#) () const  
 Return *utcb* pointer.
- template<typename T>  
 bool **put** (T \*buf, unsigned long size)  
 Put an array with *size* elements of type *T* into the stream.
- template<typename T>  
 bool **put** (T const &v)  
 Insert an element of type *T* into the stream.
- [l4\\_msgtag\\_t tag](#) () const  
 Extract the *L4* message tag from the stream.
- [l4\\_msgtag\\_t & tag](#) ()  
 Extract a reference to the *L4* message tag from the stream.
- [l4\\_msgtag\\_t send](#) ([l4\\_cap\\_idx\\_t](#) dst, long proto=0, unsigned flags=0)  
 Send the message via IPC to the given receiver.

### 16.124.1 Detailed Description

Input/Output stream for IPC [un]marshalling.

The [lpc::lostream](#) is part of the AW Env IPC framework as well as [lpc::lstream](#) and [lpc::Ostream](#). In particular an [lpc::lostream](#) is a combination of an [lpc::lstream](#) and an [lpc::Ostream](#). It can use either a single message buffer for receiving and sending messages or a pair of a receive and a send buffer. The stream also supports combined IPC operations such as [call\(\)](#) and [reply\\_and\\_wait\(\)](#), which can be used to implement RPC functionality.

#### Examples

[examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#), [examples/libs/l4re/streammap/client.cc](#), and [examples/libs/l4re/streammap/server.cc](#).

Definition at line 789 of file [ipc\\_stream](#).

## 16.124.2 Constructor & Destructor Documentation

### 16.124.2.1 `Iostream()`

```
L4::Ipc::Iostream::Iostream (  
    l4_utcb_t * utcb) [inline], [explicit]
```

Create an IPC IO stream with a single message buffer.

#### Parameters

---

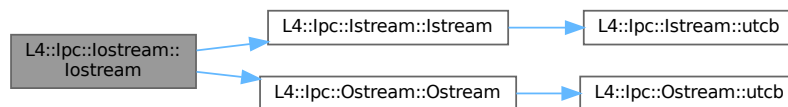
<i>utcb</i>	The message buffer used as backing store.
-------------	-------------------------------------------

The created IO stream uses the same message buffer for sending and receiving IPC messages.

Definition at line 801 of file [ipc\\_stream](#).

References [L4::ipc::Istream::Istream\(\)](#), and [L4::ipc::Ostream::Ostream\(\)](#).

Here is the call graph for this function:



## 16.124.3 Member Function Documentation

### 16.124.3.1 call()

```

l4_msgtag_t L4::Ipc::Iostream::call (
    l4_cap_idx_t dst,
    l4_timeout_t timeout,
    long proto = 0) [inline]

```

Do an IPC call using the message in the output stream and receive the reply in the input stream.

#### Parameters

<i>dst</i>	The destination to call.
<i>timeout</i>	The IPC timeout for the call.
<i>proto</i>	The protocol value to use in the message tag.

#### Returns

The result tag of the IPC operation.

This is a combined IPC operation consisting of a send and a receive to/from the given destination *dst*.

A call is usually used by clients for RPCs to a server.

#### Examples

[examples/libs/l4re/streammap/client.cc](#).

Definition at line 966 of file [ipc\\_stream](#).

References [l4\\_ipc\\_call\(\)](#), and [L4::ipc::Istream::tag\(\)](#).

Here is the call graph for this function:



### 16.124.3.2 `get()` [1/3]

```

template<typename T>
unsigned long L4::IpC::Istream::get (
    Msg_ptr< T > const & buf,
    unsigned long elems = 1) [inline]
  
```

Read one size elements of type T from the stream and return a pointer.

#### Parameters

<i>buf</i>	A <a href="#">Msg_ptr</a> that is actually set to point to the element in the stream.
<i>elems</i>	Number of elements to extract (default is 1).

#### Returns

The number of elements extracted.

In contrast to a normal `get`, this version does actually not copy the data but returns a pointer to the data.

See [operator>>\(\)](#)

Definition at line 439 of file [ipc\\_stream](#).

### 16.124.3.3 `get()` [2/3]

```

template<typename T>
bool L4::IpC::Istream::get (
    T & v) [inline]
  
```

Extract a single element of type T from the stream.

#### Parameters

<i>out</i>	<i>v</i>	The element.
------------	----------	--------------

**Return values**

<i>true</i>	An element was successfully extracted.
<i>false</i>	An element could not be extracted.

See [operator>>\(\)](#)

Definition at line 464 of file [ipc\\_stream](#).

**16.124.3.4 get() [3/3]**

```
template<typename T>
unsigned long L4::Ipc::Istream::get (
    T * buf,
    unsigned long elems) [inline]
```

Copy out an array of type T with *size* elements.

**Parameters**

<i>buf</i>	Pointer to a buffer for <i>size</i> elements of type T.
<i>elems</i>	Number of elements of type T to copy out.

**Returns**

The number of elements copied out.

See [operator>>\(\)](#)

Definition at line 394 of file [ipc\\_stream](#).

**16.124.3.5 put() [1/2]**

```
template<typename T>
bool L4::Ipc::Ostream::put (
    T * buf,
    unsigned long size) [inline]
```

Put an array with *size* elements of type T into the stream.

**Parameters**

<i>buf</i>	A pointer to the array to insert into the buffer.
------------	---------------------------------------------------

<i>size</i>	The number of elements in the array.
-------------	--------------------------------------

Definition at line 660 of file [ipc\\_stream](#).

### 16.124.3.6 put() [2/2]

```
template<typename T>
bool L4::IpC::Ostream::put (
    T const & v) [inline]
```

Insert an element of type T into the stream.

#### Parameters

<i>v</i>	The element to insert.
----------	------------------------

Definition at line 678 of file [ipc\\_stream](#).

### 16.124.3.7 reply\_and\_wait() [1/2]

```
l4_msgtag_t L4::IpC::Iostream::reply_and_wait (
    l4_umword_t * src_dst,
    l4_timeout_t timeout,
    long proto = 0) [inline]
```

Do an IPC reply and wait.

#### Parameters

<i>in, out</i>	<i>src_dst</i>	Input: the destination for the send operation. Output: the source of the received message.
	<i>timeout</i>	Timeout used for IPC.
	<i>proto</i>	Protocol to use.

#### Returns

The result tag of the IPC operation.

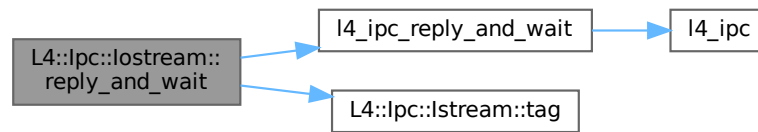
This is a combined IPC operation consisting of a send operation and an open wait for any message.

A reply and wait is usually used by servers that reply to a client and wait for the next request by any other client.

Definition at line 981 of file [ipc\\_stream](#).

References [l4\\_ipc\\_reply\\_and\\_wait\(\)](#), and [L4::IpC::Istream::tag\(\)](#).

Here is the call graph for this function:



### 16.124.3.8 reply\_and\_wait() [2/2]

```

l4_msgtag_t L4::Ipc::Iostream::reply_and_wait (
    l4_umword_t * src_dst,
    long proto = 0) [inline]
  
```

Do an IPC reply and wait.

#### Parameters

<i>in, out</i>	<i>src_dst</i>	Input: the destination for the send operation. Output: the source of the received message.
	<i>proto</i>	Protocol to use.

#### Returns

The result tag of the IPC operation.

This is a combined IPC operation consisting of a send operation and an open wait for any message.

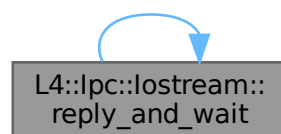
A reply and wait is usually used by servers that reply to a client and wait for the next request by any other client.

Definition at line 874 of file [ipc\\_stream](#).

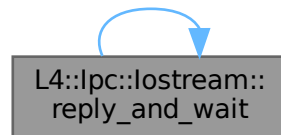
References [L4\\_IPC\\_SEND\\_TIMEOUT\\_0](#), and [reply\\_and\\_wait\(\)](#).

Referenced by [reply\\_and\\_wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 16.124.3.9 reset()

```
void L4::Ipc::Iostream::reset () [inline]
```

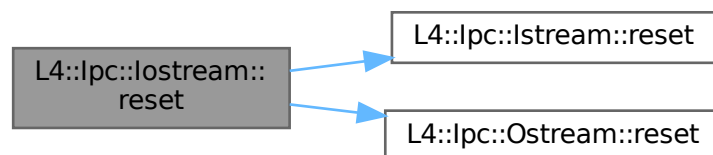
Reset the stream to its initial state.

Input as well as the output stream are reset.

Definition at line 815 of file [ipc\\_stream](#).

References [L4::lpc::Istream::reset\(\)](#), and [L4::lpc::Ostream::reset\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- [l4/cxx/ipc\\_stream](#)

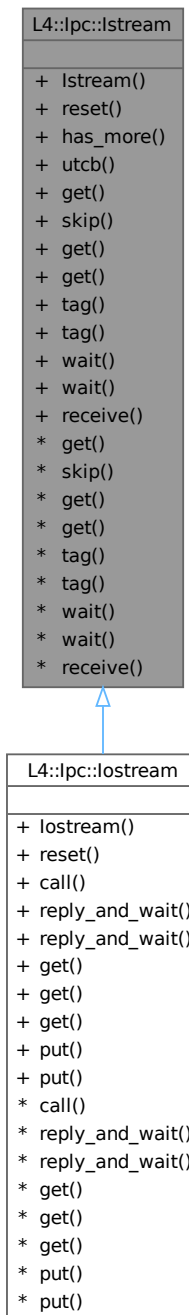


## 16.125 L4::lpc::Istream Class Reference

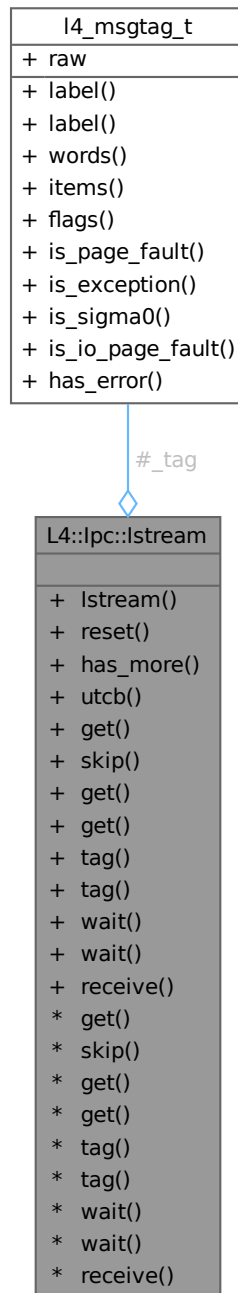
Input stream for IPC unmarshalling.

```
#include <ipc_stream>
```

Inheritance diagram for L4::lpc::Istream:



Collaboration diagram for L4::lpc::lstream:



## Public Member Functions

- [lstream](#) ([l4\\_utcb\\_t](#) \*utcb)  
*Create an input stream for the given message buffer.*
- void [reset](#) ()  
*Reset the stream to empty, and ready for [receive\(\)](#)/[wait\(\)](#).*

- `template<typename T>`  
`bool has_more (unsigned long count=1)`  
*Check whether a value of type T can be obtained from the stream.*
- `l4_utcb_t * utcb () const`  
*Return utcb pointer.*

#### Get/Put Functions.

- `template<typename T>`  
`unsigned long get (T *buf, unsigned long elems)`  
*Copy out an array of type T with size elements.*
- `template<typename T>`  
`void skip (unsigned long elems)`  
*Skip size elements of type T in the stream.*
- `template<typename T>`  
`unsigned long get (Msg_ptr< T > const &buf, unsigned long elems=1)`  
*Read one size elements of type T from the stream and return a pointer.*
- `template<typename T>`  
`bool get (T &v)`  
*Extract a single element of type T from the stream.*
- `l4_msgtag_t tag () const`  
*Get the message tag of a received IPC.*
- `l4_msgtag_t & tag ()`  
*Get the message tag of a received IPC.*

#### IPC operations.

- `l4_msgtag_t wait (l4_umword_t *src)`  
*Wait for an incoming message from any sender.*
- `l4_msgtag_t wait (l4_umword_t *src, l4_timeout_t timeout)`  
*Wait for an incoming message from any sender.*
- `l4_msgtag_t receive (l4_cap_idx_t src)`  
*Wait for a message from the specified sender.*

### 16.125.1 Detailed Description

Input stream for IPC unmarshalling.

[lpc::Istream](#) is part of the dynamic IPC marshalling infrastructure, as well as [lpc::Ostream](#) and [lpc::Iostream](#).

[lpc::Istream](#) is an input stream supporting extraction of values from an IPC message buffer. A received IPC message can be unmarshalled using the usual extraction operator (>>).

There exist some special wrapper classes to extract arrays (see [lpc\\_buf\\_cp\\_in](#) and [lpc\\_buf\\_in](#)) and indirect strings (see [Msg\\_in\\_buffer](#) and [Msg\\_io\\_buffer](#)).

Definition at line 334 of file [ipc\\_stream](#).

### 16.125.2 Constructor & Destructor Documentation

#### 16.125.2.1 Istream()

```
L4::Ipc::Istream::Istream (
    l4_utcb_t * utcb) [inline]
```

Create an input stream for the given message buffer.

The given message buffer is used for IPC operations [wait\(\)](#)/[receive\(\)](#) and received data can be extracted using the >> operator afterwards. In the case of indirect message parts a buffer of type [Msg\\_in\\_buffer](#) must be inserted into the stream before the IPC operation and contains received data afterwards.

#### Parameters

<i>utcb</i>	The message buffer to receive IPC messages.
-------------	---------------------------------------------

Definition at line 348 of file [ipc\\_stream](#).

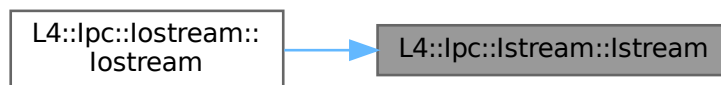
References [utcb\(\)](#).

Referenced by [L4::lpc::lstream::lstream\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



## 16.125.3 Member Function Documentation

### 16.125.3.1 `get()` [1/3]

```

template<typename T>
unsigned long L4::Ipc::Istream::get (
    Msg\_ptr< T > const & buf,
    unsigned long elems = 1) [inline]
  
```

Read one size elements of type T from the stream and return a pointer.

#### Parameters

<i>buf</i>	A <a href="#">Msg_ptr</a> that is actually set to point to the element in the stream.
<i>elems</i>	Number of elements to extract (default is 1).

**Returns**

The number of elements extracted.

In contrast to a normal get, this version does actually not copy the data but returns a pointer to the data.

See [operator>>\(\)](#)

Definition at line 439 of file [ipc\\_stream](#).

References [has\\_more\(\)](#), and [L4\\_UNLIKELY](#).

Here is the call graph for this function:

**16.125.3.2 get() [2/3]**

```

template<typename T>
bool L4::lpc::Istream::get (
    T & v) [inline]
  
```

Extract a single element of type T from the stream.

**Parameters**

out	v	The element.
-----	---	--------------

**Return values**

<i>true</i>	An element was successfully extracted.
<i>false</i>	An element could not be extracted.

See [operator>>\(\)](#)

Definition at line 464 of file [ipc\\_stream](#).

References [has\\_more\(\)](#), and [L4\\_UNLIKELY](#).

Here is the call graph for this function:



### 16.125.3.3 `get()` [3/3]

```
template<typename T>
unsigned long L4::Ipc::Istream::get (
    T * buf,
    unsigned long elems) [inline]
```

Copy out an array of type `T` with `size` elements.

#### Parameters

<i>buf</i>	Pointer to a buffer for <code>size</code> elements of type <code>T</code> .
<i>elems</i>	Number of elements of type <code>T</code> to copy out.

#### Returns

The number of elements copied out.

See [operator>>\(\)](#)

Definition at line 394 of file [ipc\\_stream](#).

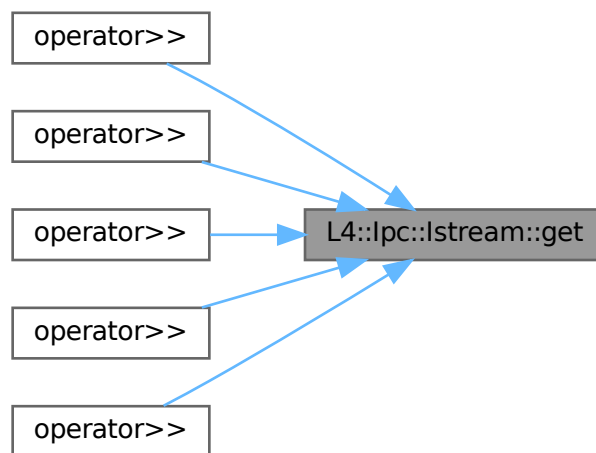
References [has\\_more\(\)](#), and [L4\\_UNLIKELY](#).

Referenced by [operator>>\(\)](#), [operator>>\(\)](#), [operator>>\(\)](#), [operator>>\(\)](#), and [operator>>\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 16.125.3.4 receive()

```
l4_msgtag_t L4::Ipc::Istream::receive (
    l4_cap_idx_t src) [inline]
```

Wait for a message from the specified sender.

##### Parameters

<code>src</code>	The sender id to receive from.
------------------	--------------------------------

##### Returns

The IPC result tag (`l4_msgtag_t`).

This is commonly known as 'closed wait'.

Definition at line 572 of file `ipc_stream`.

References `L4_IPC_NEVER`, and `receive()`.

Referenced by `receive()`.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 16.125.3.5 reset()

```
void L4::Ipc::Istream::reset () [inline]
```

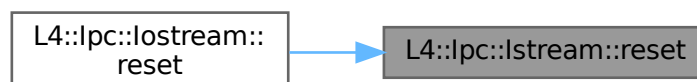
Reset the stream to empty, and ready for [receive\(\)/wait\(\)](#).

The stream is reset to the same state as on its creation.

Definition at line [358](#) of file [ipc\\_stream](#).

Referenced by [L4::lpc::Istream::reset\(\)](#).

Here is the caller graph for this function:





### 16.125.3.6 skip()

```
template<typename T>
void L4::Ipc::Istream::skip (
    unsigned long elems) [inline]
```

Skip size elements of type T in the stream.

#### Parameters

---

<i>elems</i>	Number of elements to skip.
--------------	-----------------------------

Definition at line 414 of file [ipc\\_stream](#).

References [has\\_more\(\)](#), and [L4\\_UNLIKELY](#).

Here is the call graph for this function:



### 16.125.3.7 tag() [1/2]

```
l4_msgtag_t & L4::Ipc::Istream::tag () [inline]
```

Get the message tag of a received IPC.

#### Returns

A reference to the [L4](#) message tag for the received IPC.

This is in particular useful for handling page faults or exceptions.

See [operator>>\(\)](#)

Definition at line 517 of file [ipc\\_stream](#).

### 16.125.3.8 tag() [2/2]

```
l4_msgtag_t L4::Ipc::Istream::tag () const [inline]
```

Get the message tag of a received IPC.

**Returns**

The [L4](#) message tag for the received IPC.

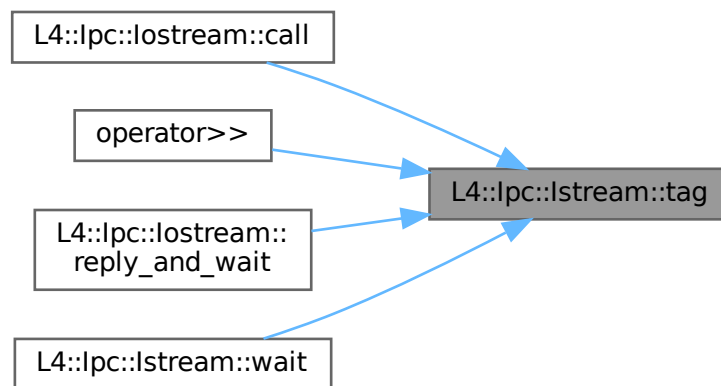
This is in particular useful for handling page faults or exceptions.

See [operator>>\(\)](#)

Definition at line 505 of file [ipc\\_stream](#).

Referenced by [L4::lpc::Istream::call\(\)](#), [operator>>\(\)](#), [L4::lpc::Istream::reply\\_and\\_wait\(\)](#), and [wait\(\)](#).

Here is the caller graph for this function:

**16.125.3.9 wait() [1/2]**

```
l4_msgtag_t L4::Ipc::Istream::wait (
    l4_umword_t * src) [inline]
```

Wait for an incoming message from any sender.

**Parameters**

out	src	Contains the sender after a successful IPC operation.
-----	-----	-------------------------------------------------------

**Returns**

Syscall return tag.

This wait is actually known as 'open wait'.

Definition at line 548 of file [ipc\\_stream](#).

References [L4\\_IPC\\_NEVER](#), and [wait\(\)](#).

Referenced by [wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**16.125.3.10 wait() [2/2]**

```

l4_msgtag_t L4::Ipc::Istream::wait (
    l4_umword_t * src,
    l4_timeout_t timeout) [inline]
  
```

Wait for an incoming message from any sender.

**Parameters**

out	src	Contains the sender after a successful IPC operation.
	timeout	Timeout used for IPC.

### Returns

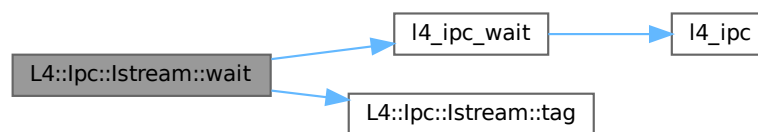
The IPC result tag ([l4\\_msgtag\\_t](#)).

This wait is actually known as 'open wait'.

Definition at line 1013 of file [ipc\\_stream](#).

References [l4\\_ipc\\_wait\(\)](#), and [tag\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

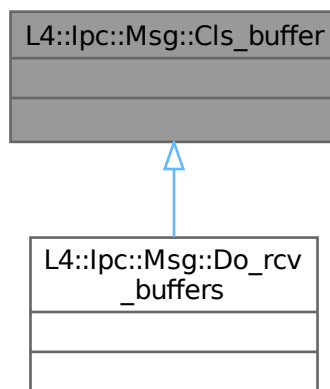
- [l4/cxx/ipc\\_stream](#)

## 16.126 L4::lpc::Msg::Cls\_buffer Struct Reference

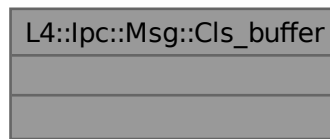
Marker type for receive buffer values.

```
#include <ipc_basics>
```

Inheritance diagram for `L4::lpc::Msg::Cls_buffer`:



Collaboration diagram for L4::lpc::Msg::Cls\_buffer:



### 16.126.1 Detailed Description

Marker type for receive buffer values.

Definition at line 154 of file [ipc\\_basics](#).

The documentation for this struct was generated from the following file:

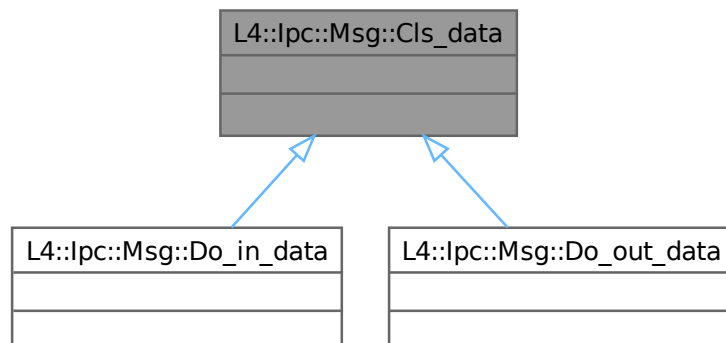
- [l4/sys/cxx/ipc\\_basics](#)

## 16.127 L4::lpc::Msg::Cls\_data Struct Reference

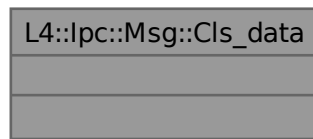
Marker type for data values.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Cls\_data:



Collaboration diagram for L4::lpc::Msg::Cls\_data:



### 16.127.1 Detailed Description

Marker type for data values.

Definition at line 150 of file [ipc\\_basics](#).

The documentation for this struct was generated from the following file:

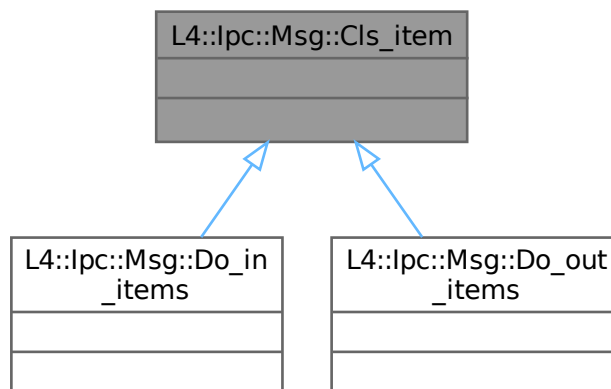
- `l4/sys/cxx/ipc_basics`

## 16.128 L4::lpc::Msg::Cls\_item Struct Reference

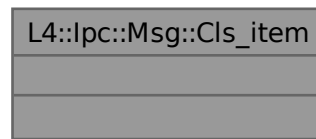
Marker type for item values.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Cls\_item:



Collaboration diagram for L4::lpc::Msg::Cls\_item:



### 16.128.1 Detailed Description

Marker type for item values.

Definition at line 152 of file [ipc\\_basics](#).

The documentation for this struct was generated from the following file:

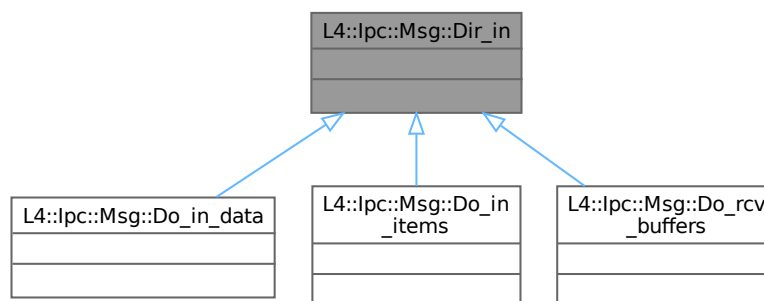
- `l4/sys/cxx/ipc_basics`

## 16.129 L4::lpc::Msg::Dir\_in Struct Reference

Marker type for input values.

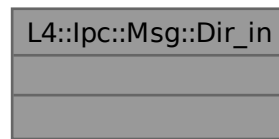
```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Dir\_in:





Collaboration diagram for L4::lpc::Msg::Dir\_in:



### 16.129.1 Detailed Description

Marker type for input values.

Definition at line 145 of file [ipc\\_basics](#).

The documentation for this struct was generated from the following file:

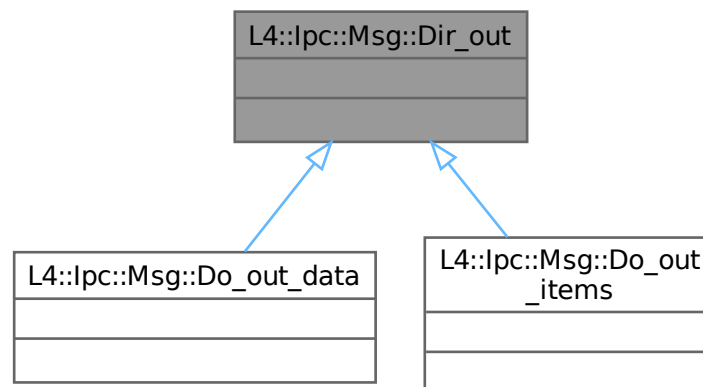
- `l4/sys/cxx/ipc_basics`

## 16.130 L4::lpc::Msg::Dir\_out Struct Reference

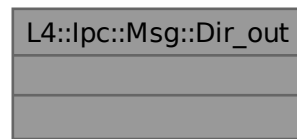
Marker type for output values.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Dir\_out:



Collaboration diagram for L4::lpc::Msg::Dir\_out:



### 16.130.1 Detailed Description

Marker type for output values.

Definition at line 147 of file [ipc\\_basics](#).

The documentation for this struct was generated from the following file:

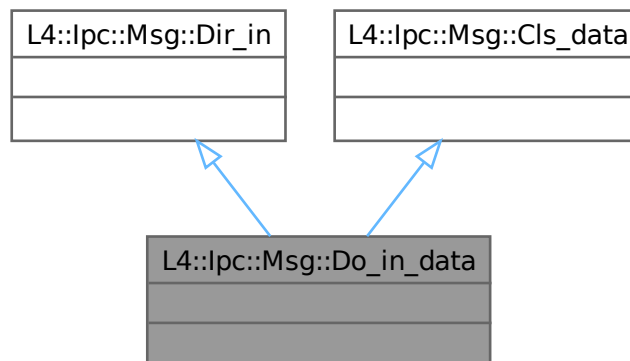
- `l4/sys/cxx/ipc_basics`

## 16.131 L4::lpc::Msg::Do\_in\_data Struct Reference

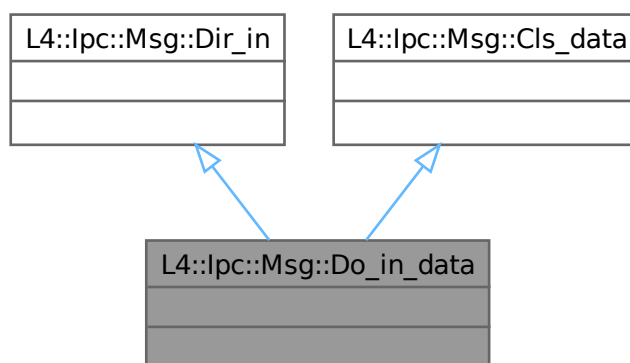
Marker for Input data.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Do\_in\_data:



Collaboration diagram for L4::lpc::Msg::Do\_in\_data:



### 16.131.1 Detailed Description

Marker for Input data.

Definition at line 158 of file [ipc\\_basics](#).

The documentation for this struct was generated from the following file:

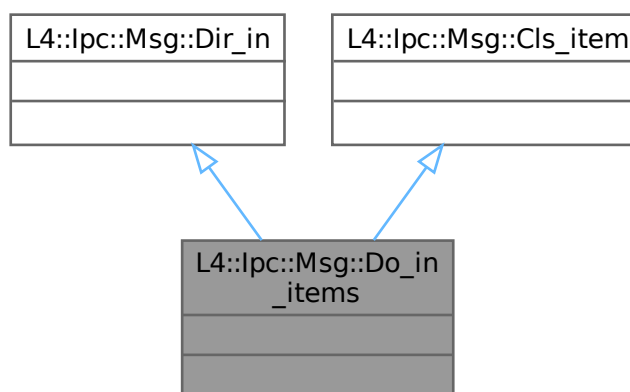
- `l4/sys/cxx/ipc_basics`

## 16.132 L4::lpc::Msg::Do\_in\_items Struct Reference

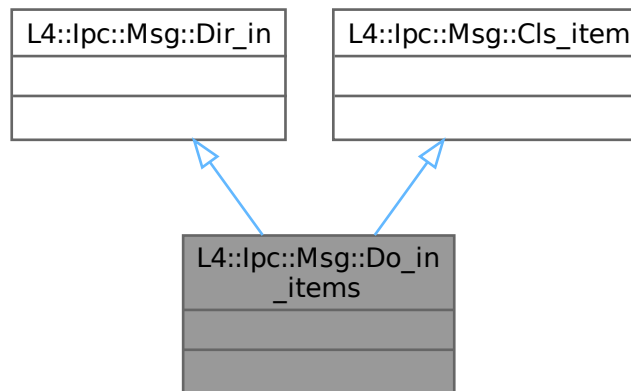
Marker for Input items.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Do\_in\_items:



Collaboration diagram for L4::lpc::Msg::Do\_in\_items:



### 16.132.1 Detailed Description

Marker for Input items.

Definition at line 162 of file [ipc\\_basics](#).

The documentation for this struct was generated from the following file:

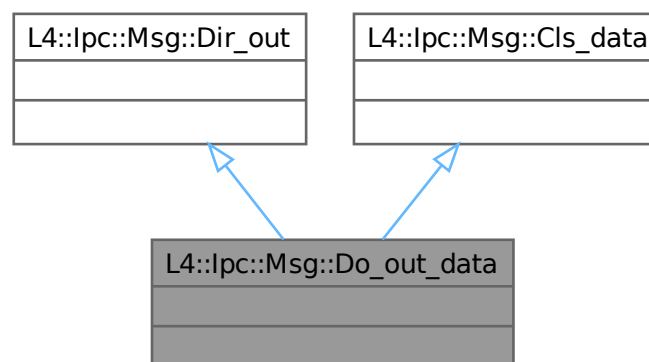
- l4/sys/cxx/ipc\_basics

## 16.133 L4::lpc::Msg::Do\_out\_data Struct Reference

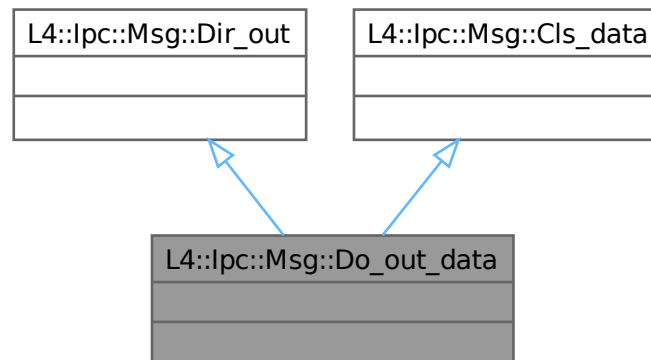
Marker for Output data.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Do\_out\_data:



Collaboration diagram for L4::lpc::Msg::Do\_out\_data:



### 16.133.1 Detailed Description

Marker for Output data.

Definition at line 160 of file [ipc\\_basics](#).

The documentation for this struct was generated from the following file:

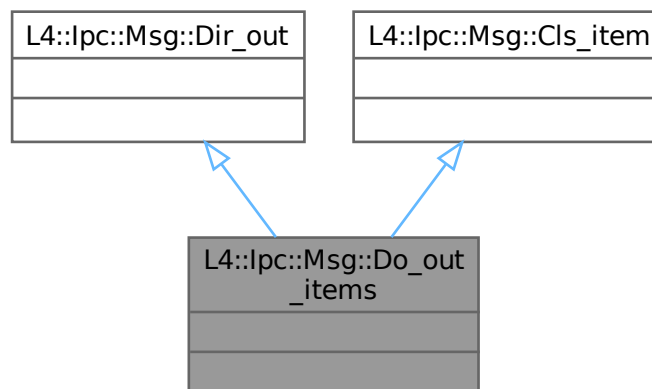
- `l4/sys/cxx/ipc_basics`

## 16.134 L4::lpc::Msg::Do\_out\_items Struct Reference

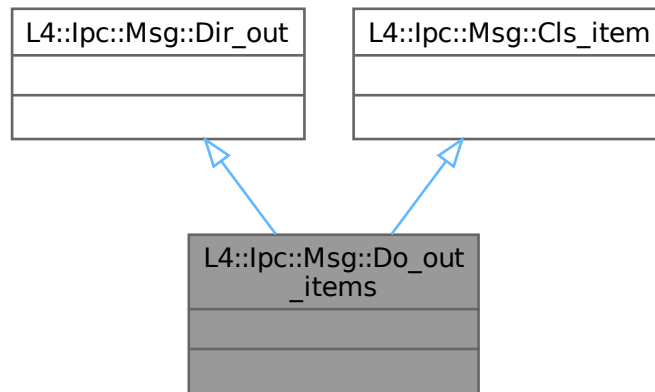
Marker for Output items.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Do\_out\_items:



Collaboration diagram for L4::lpc::Msg::Do\_out\_items:



### 16.134.1 Detailed Description

Marker for Output items.

Definition at line 164 of file [ipc\\_basics](#).

The documentation for this struct was generated from the following file:

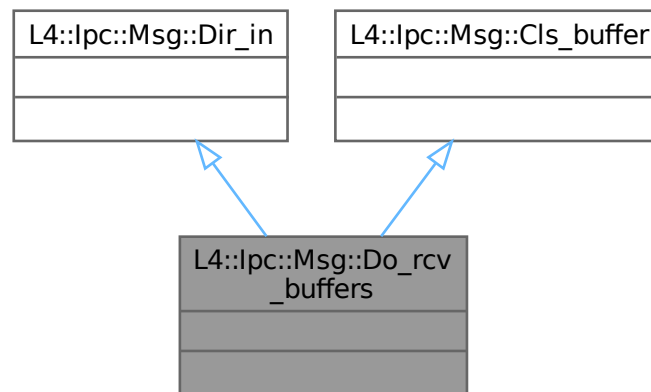
- l4/sys/cxx/ipc\_basics

## 16.135 L4::lpc::Msg::Do\_rcv\_buffers Struct Reference

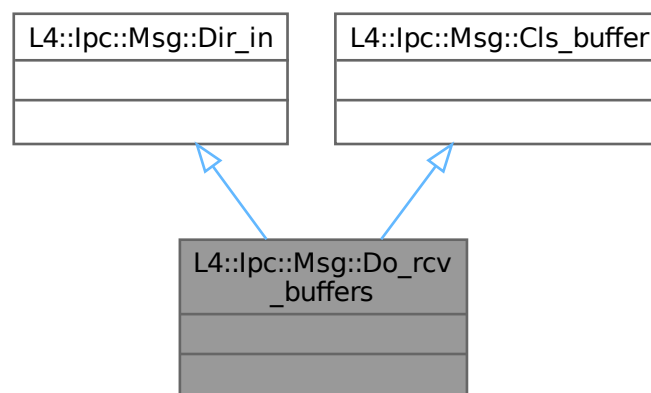
Marker for receive buffers.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Do\_rcv\_buffers:



Collaboration diagram for L4::lpc::Msg::Do\_rcv\_buffers:



### 16.135.1 Detailed Description

Marker for receive buffers.

Definition at line 166 of file [ipc\\_basics](#).

The documentation for this struct was generated from the following file:

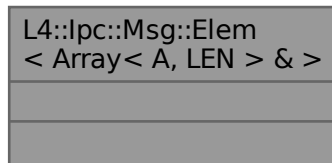
- `l4/sys/cxx/ipc_basics`

## 16.136 L4::lpc::Msg::Elem< Array< A, LEN > & > Struct Template Reference

[Array](#) as output argument.

```
#include <ipc_array>
```

Collaboration diagram for L4::lpc::Msg::Elem< Array< A, LEN > & >:



### Public Types

- typedef [Array](#)< A, LEN > & **arg\_type**  
[Array](#)<> & at the interface.
- typedef [Array\\_ref](#)< A, LEN > **svr\_type**  
[Array\\_ref](#)<> as server storage type.
- typedef [svr\\_type](#) & **svr\_arg\_type**  
[Array\\_ref](#)<> & at the server side.

### 16.136.1 Detailed Description

```
template<typename A, typename LEN>
struct L4::lpc::Msg::Elem< Array< A, LEN > & >
```

[Array](#) as output argument.

Definition at line 170 of file [ipc\\_array](#).

The documentation for this struct was generated from the following file:

- `I4/sys/cxx/ipc_array`

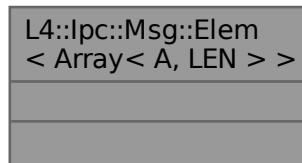


## 16.137 L4::lpc::Msg::Elem< Array< A, LEN > > Struct Template Reference

[Array](#) as input arguments.

```
#include <ipc_array>
```

Collaboration diagram for L4::lpc::Msg::Elem< Array< A, LEN > >:



### Public Types

- typedef [Array](#)< A, LEN > **arg\_type**  
[Array](#)<> as argument at the interface.
- typedef [Array\\_ref](#)< A, LEN > **svr\_type**  
[Array\\_ref](#)<> at the server side.

### 16.137.1 Detailed Description

```
template<typename A, typename LEN>
struct L4::lpc::Msg::Elem< Array< A, LEN > >
```

[Array](#) as input arguments.

Definition at line 158 of file [ipc\\_array](#).

The documentation for this struct was generated from the following file:

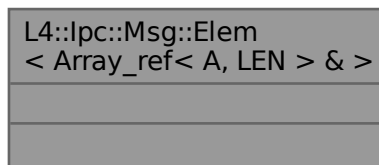
- l4/sys/cxx/ipc\_array

## 16.138 L4::lpc::Msg::Elem< Array\_ref< A, LEN > & > Struct Template Reference

[Array\\_ref](#) as output argument.

```
#include <ipc_array>
```

Collaboration diagram for L4::lpc::Msg::Elem< Array\_ref< A, LEN > & >:



### Public Types

- typedef [Array\\_ref](#)< A, LEN > & **arg\_type**  
*[Array\\_ref](#)<> at the interface.*
- typedef [Array\\_ref](#)< typename L4::Types::Remove\_const< A >::type, LEN > **svr\_type**  
*[Array\\_ref](#)<> as server storage.*
- typedef **svr\_type** & **svr\_arg\_type**  
*[Array\\_ref](#)<> & as server argument.*

### 16.138.1 Detailed Description

```
template<typename A, typename LEN>
struct L4::lpc::Msg::Elem< Array_ref< A, LEN > & >
```

[Array\\_ref](#) as output argument.

Definition at line 183 of file [ipc\\_array](#).

The documentation for this struct was generated from the following file:

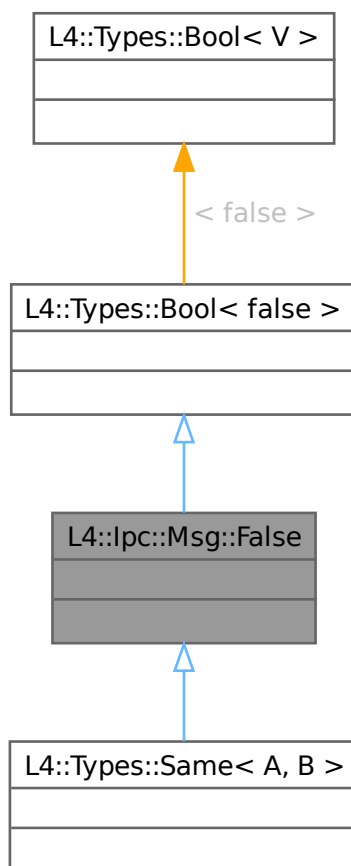
- I4/sys/cxx/ipc\_array

## 16.139 L4::lpc::Msg::False Struct Reference

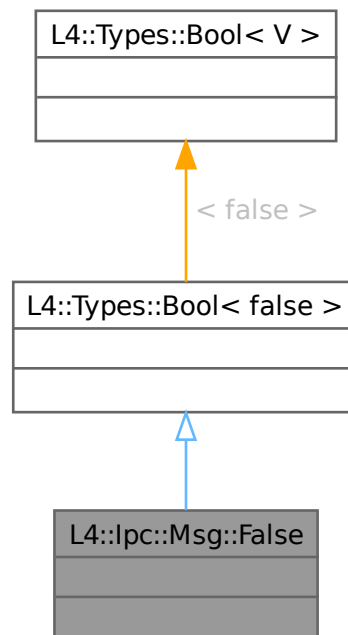
False meta value.

```
#include <types>
```

Inheritance diagram for L4::lpc::Msg::False:



Collaboration diagram for L4::ipc::Msg::False:



#### Additional Inherited Members

#### Public Types inherited from [L4::Types::Bool< false >](#)

- typedef [Bool< V >](#) **type**  
*The meta type itself.*

#### 16.139.1 Detailed Description

[False](#) meta value.

Definition at line [296](#) of file [types](#).

The documentation for this struct was generated from the following file:

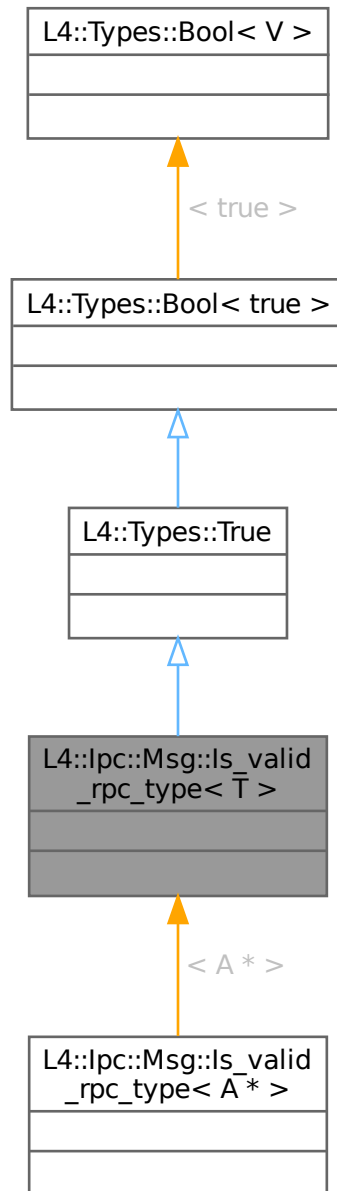
- [l4/sys/cxx/types](#)

## 16.140 L4::lpc::Msg::ls\_valid\_rpc\_type< T > Struct Template Reference

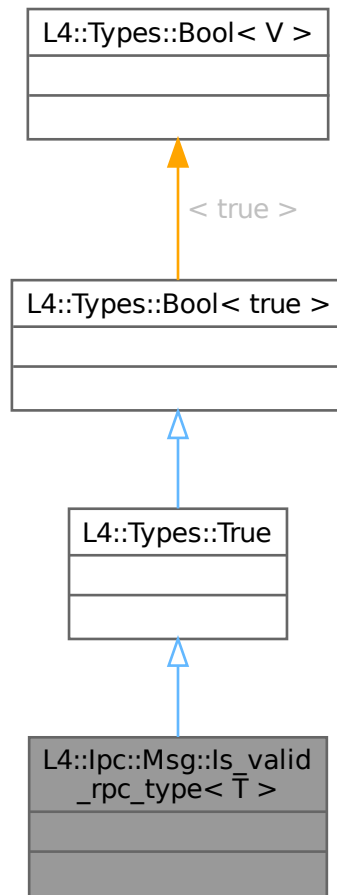
Type trait defining a valid RPC parameter type.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::ls\_valid\_rpc\_type< T >:



Collaboration diagram for `L4::lpc::Msg::ls_valid_rpc_type< T >`:



#### Additional Inherited Members

#### Public Types inherited from `L4::Types::Bool< true >`

- typedef `Bool< V >` type  
*The meta type itself.*

#### 16.140.1 Detailed Description

```
template<typename T>
struct L4::lpc::Msg::ls_valid_rpc_type< T >
```

Type trait defining a valid RPC parameter type.

Definition at line 339 of file `ipc_basics`.

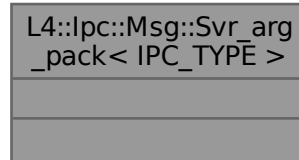
The documentation for this struct was generated from the following file:

- `I4/sys/cxx/ipc_basics`

## 16.141 L4::lpc::Msg::Svr\_arg\_pack< IPC\_TYPE > Struct Template Reference

Server-side RPC arguments data structure used to provide arguments to the server-side implementation of an RPC function.

Collaboration diagram for L4::lpc::Msg::Svr\_arg\_pack< IPC\_TYPE >:



### 16.141.1 Detailed Description

```
template<typename IPC_TYPE>
struct L4::lpc::Msg::Svr_arg_pack< IPC_TYPE >
```

Server-side RPC arguments data structure used to provide arguments to the server-side implementation of an RPC function.

Definition at line 144 of file [ipc\\_server](#).

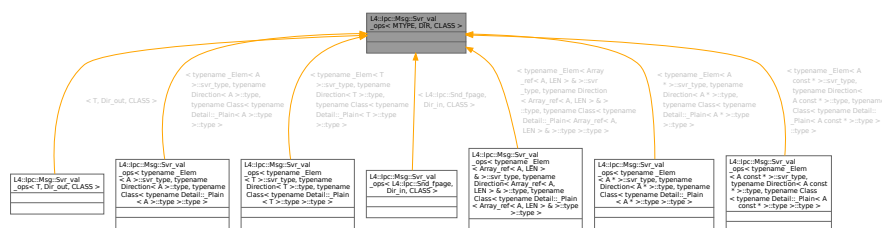
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_server`

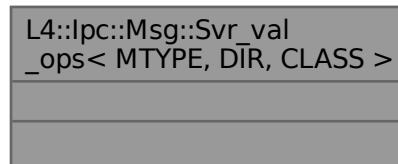
## 16.142 L4::lpc::Msg::Svr\_val\_ops< MTYPE, DIR, CLASS > Struct Template Reference

Defines client-side handling of 'MTYPE' as RPC argument.

Inheritance diagram for L4::lpc::Msg::Svr\_val\_ops< MTYPE, DIR, CLASS >:



Collaboration diagram for L4::lpc::Msg::Svr\_val\_ops< MTYPE, DIR, CLASS >:



### 16.142.1 Detailed Description

```
template<typename MTYPE, typename DIR, typename CLASS>
struct L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS >
```

Defines client-side handling of `MTYPE` as RPC argument.

#### Template Parameters

<i>MTYPE</i>	Elem<T>::arg_type (where T is the type used in the RPC definition)
<i>DIR</i>	<a href="#">Dir_in</a> (client -> server), or <a href="#">Dir_out</a> (server -> client)
<i>CLASS</i>	<a href="#">Cls_data</a> , <a href="#">Cls_item</a> , or <a href="#">Cls_buffer</a> */ template<typename MTYPE, typename DIR, typename CLASS> struct Clnt_val_ops;

```
template<typename T> struct Clnt_noops { template<typename A, typename B> static constexpr int to_msg(char
*, unsigned offset, unsigned, T, A, B) noexcept { return offset; } }
```

```
/ copy data from the message to the client reference template<typename A, typename B> static constexpr int
from_msg(char *, unsigned offset, unsigned, long, T const &, A, B) noexcept { return offset; } };
```

```
template<typename T> struct Svr_noops { template<typename A, typename B> static constexpr int from_svr(char
*, unsigned offset, unsigned, long, T, A, B) noexcept { return offset; } }
```

```
/ copy data from the message to the client reference template<typename A, typename B> static constexpr int
to_svr(char *, unsigned offset, unsigned, T, A, B) noexcept { return offset; } };
```

```
template<typename MTYPE, typename CLASS> struct Clnt_val_ops<MTYPE, Dir_in, CLASS> : Clnt_noops<↵
MTYPE> { using Clnt_noops<MTYPE>::to_msg; / Copy a T into the message static int to_msg(char *msg, un-
signed offset, unsigned limit, MTYPE arg, Dir_in, CLASS) noexcept { return msg_add<MTYPE>(msg, offset, limit,
arg); } };
```

```
template<typename MTYPE, typename CLASS> struct Clnt_val_ops<MTYPE, Dir_out, CLASS> : Clnt_↵
noops<MTYPE> { using Clnt_noops<MTYPE>::from_msg; / copy data from the message to the client reference
static int from_msg(char *msg, unsigned offset, unsigned limit, long, MTYPE &arg, Dir_out, CLASS) noexcept {
return msg_get<MTYPE>(msg, offset, limit, arg); } };
```

/\*\* Defines server-side handling for MTYPE server arguments.

#### Template Parameters



<i>MTYPE</i>	Elem<T>::svr_type (where T is the type used in the RPC definition)
<i>DIR</i>	<a href="#">Dir_in</a> (client -> server), or <a href="#">Dir_out</a> (server -> client)
<i>CLASS</i>	<a href="#">Cls_data</a> , <a href="#">Cls_item</a> , or <a href="#">Cls_buffer</a>

Definition at line 264 of file [ipc\\_basics](#).

The documentation for this struct was generated from the following file:

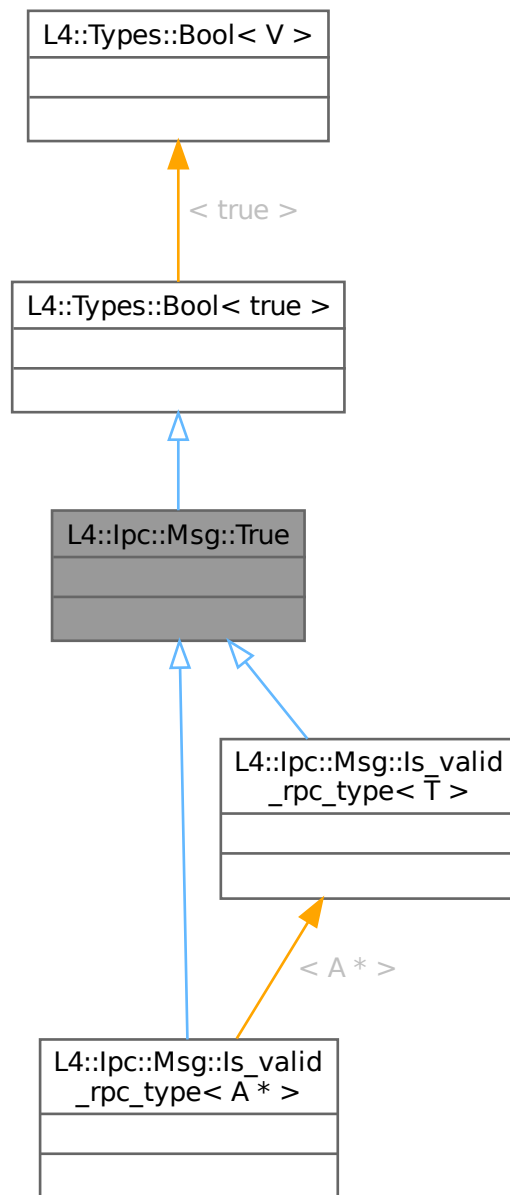
- l4/sys/cxx/ipc\_basics

## 16.143 L4::lpc::Msg::True Struct Reference

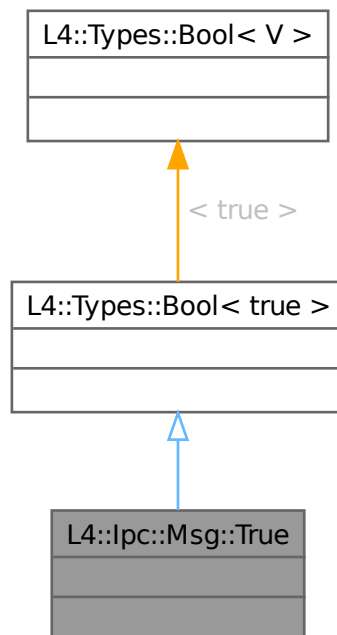
[True](#) meta value.

```
#include <types>
```

Inheritance diagram for L4::lpc::Msg::True:



Collaboration diagram for L4::lpc::Msg::True:



#### Additional Inherited Members

#### Public Types inherited from `L4::Types::Bool< true >`

- typedef `Bool< V > type`  
*The meta type itself.*

#### 16.143.1 Detailed Description

`True` meta value.

Definition at line 300 of file `types`.

The documentation for this struct was generated from the following file:

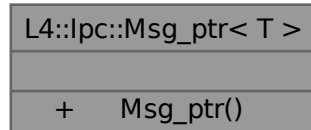
- `l4/sys/cxx/types`

## 16.144 L4::lpc::Msg\_ptr< T > Class Template Reference

Pointer to an element of type T in an [lpc::lstream](#).

```
#include <ipc_stream>
```

Collaboration diagram for L4::lpc::Msg\_ptr< T >:



### Public Member Functions

- [Msg\\_ptr](#) (T \*&p)

Create a [Msg\\_ptr](#) object that set pointer p to point into the message buffer.

### 16.144.1 Detailed Description

```
template<typename T>
class L4::lpc::Msg_ptr< T >
```

Pointer to an element of type T in an [lpc::lstream](#).

This wrapper can be used to extract an element of type T from an [lpc::lstream](#), whereas the data is not copied out, but a pointer into the message buffer itself is returned. With is mechanism it is possible to avoid an extra copy of large data structures from a received IPC message, instead the returned pointer gives direct access to the data in the message.

See [msg\\_ptr\(\)](#).

Definition at line 229 of file [ipc\\_stream](#).

### 16.144.2 Constructor & Destructor Documentation

#### 16.144.2.1 Msg\_ptr()

```
template<typename T>
L4::lpc::Msg_ptr< T >::Msg_ptr (
    T *& p) [inline], [explicit]
```

Create a [Msg\\_ptr](#) object that set pointer p to point into the message buffer.

#### Parameters

---

*p* | The pointer that is adjusted to point into the message buffer.

Definition at line 240 of file [ipc\\_stream](#).

The documentation for this class was generated from the following file:

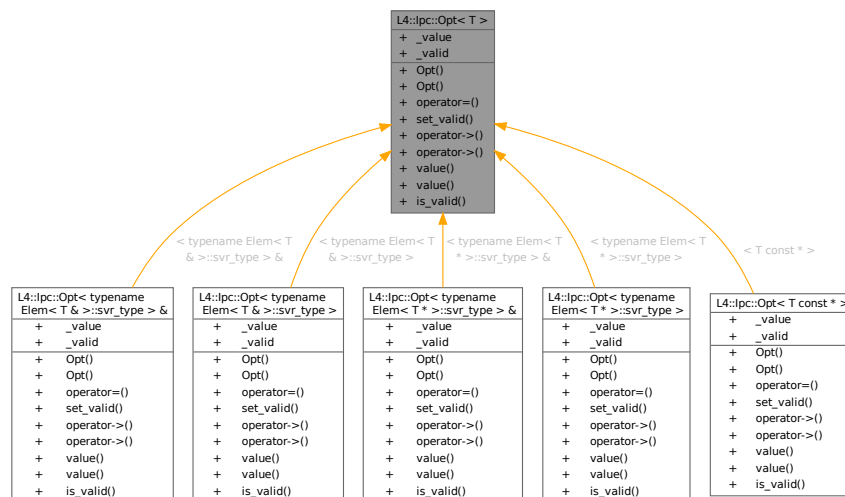
- [l4/cxx/ipc\\_stream](#)

## 16.145 L4::lpc::Opt< T > Struct Template Reference

Attribute for defining an optional RPC argument.

```
#include <ipc_types>
```

Inheritance diagram for L4::lpc::Opt< T >:



Collaboration diagram for L4::lpc::Opt< T >:

L4::lpc::Opt< T >
+ _value
+ _valid
+ Opt()
+ Opt()
+ operator=()
+ set_valid()
+ operator->()
+ operator->()
+ value()
+ value()
+ is_valid()

## Public Member Functions

- **Opt** () noexcept  
*Make an absent optional argument.*
- **Opt** (T **value**) noexcept  
*Make a present optional argument with the given value.*
- **Opt & operator=** (T **value**) noexcept  
*Assign a value to the optional argument (makes the argument present).*
- void **set\_valid** (bool valid=true) noexcept  
*Set the argument to present or absent.*
- T \* **operator->** () noexcept  
*Get the pointer to the value.*
- T const \* **operator->** () const noexcept  
*Get the const pointer to the value.*
- T **value** () const noexcept  
*Get the value.*
- T & **value** () noexcept  
*Get the value.*
- bool **is\_valid** () const noexcept  
*Get true if present, false if not.*

## Data Fields

- T **\_value**  
*The value.*
- bool **\_valid**  
*True if the optional argument is present, false else.*

### 16.145.1 Detailed Description

```
template<typename T>  
struct L4::lpc::Opt< T >
```

Attribute for defining an optional RPC argument.

Definition at line 136 of file [ipc\\_types](#).

The documentation for this struct was generated from the following file:

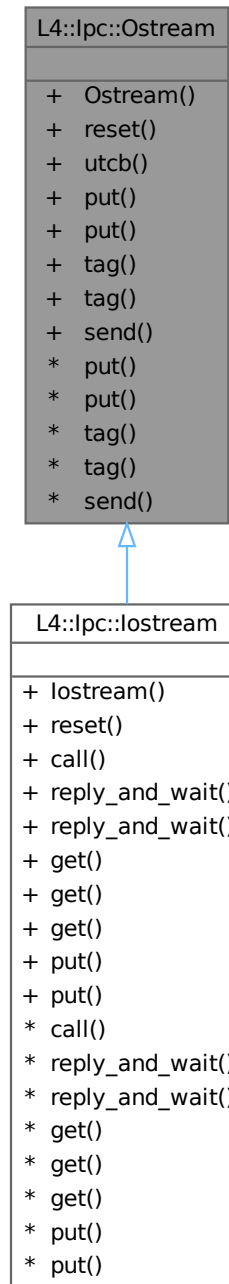
- [l4/sys/cxx/ipc\\_types](#)

## 16.146 L4::lpc::Ostream Class Reference

Output stream for IPC marshalling.

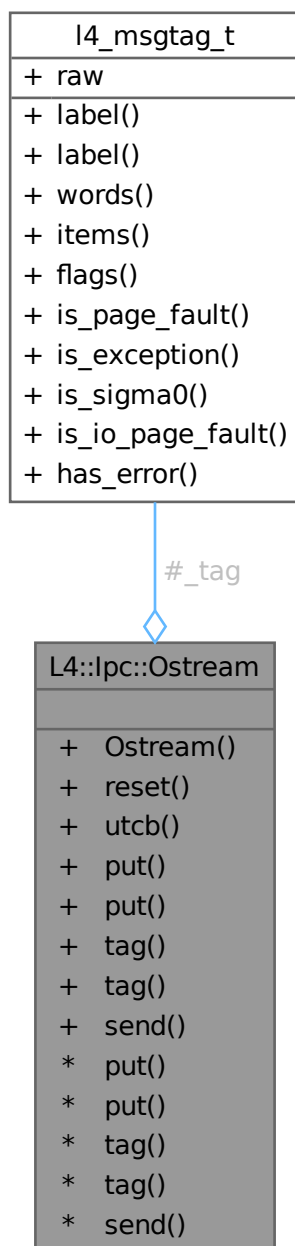
```
#include <ipc_stream>
```

Inheritance diagram for L4::ipc::Ostream:





Collaboration diagram for L4::lpc::Ostream:



## Public Member Functions

- **Ostream** ([l4\\_utcb\\_t](#) \*utcb)  
Create an IPC output stream using the given message buffer [utcb](#).
- void **reset** ()  
Reset the stream to empty, same state as a newly created stream.
- [l4\\_utcb\\_t](#) \* **utcb** () const

*Return utcb pointer.*

### Get/Put functions.

*These functions are basically used to implement the insertion operators (<<) and should not be called directly.*

- `template<typename T>`  
`bool put (T *buf, unsigned long size)`  
*Put an array with `size` elements of type `T` into the stream.*
- `template<typename T>`  
`bool put (T const &v)`  
*Insert an element of type `T` into the stream.*
- `l4_msgtag_t tag () const`  
*Extract the `L4` message tag from the stream.*
- `l4_msgtag_t & tag ()`  
*Extract a reference to the `L4` message tag from the stream.*

### IPC operations.

- `l4_msgtag_t send (l4_cap_idx_t dst, long proto=0, unsigned flags=0)`  
*Send the message via IPC to the given receiver.*

## 16.146.1 Detailed Description

Output stream for IPC marshalling.

`lpc::Ostream` is part of the dynamic IPC marshalling infrastructure, as well as `lpc::Istream` and `lpc::lostream`.

`lpc::Ostream` is an output stream supporting insertion of values into an IPC message buffer. A IPC message can be marshalled using the usual insertion operator <<, see [IPC stream operators](#) .

There exist some special wrapper classes to insert arrays (see `lpc::Buf_cp_out`) and indirect strings (see `Msg_↔out_buffer` and `Msg_io_buffer`).

Definition at line 623 of file [ipc\\_stream](#).

## 16.146.2 Member Function Documentation

### 16.146.2.1 put() [1/2]

```
template<typename T>
bool L4::Ipc::Ostream::put (
    T * buf,
    unsigned long size) [inline]
```

Put an array with `size` elements of type `T` into the stream.

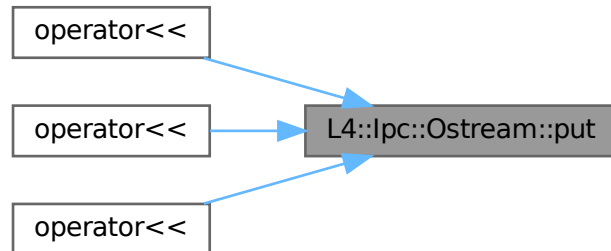
#### Parameters

<i>buf</i>	A pointer to the array to insert into the buffer.
<i>size</i>	The number of elements in the array.

Definition at line 660 of file [ipc\\_stream](#).

Referenced by [operator<<\(\)](#), [operator<<\(\)](#), and [operator<<\(\)](#).

Here is the caller graph for this function:



### 16.146.2.2 put() [2/2]

```
template<typename T>
bool L4::Ipc::Ostream::put (
    T const & v) [inline]
```

Insert an element of type T into the stream.

#### Parameters

<i>v</i>	The element to insert.
----------	------------------------

Definition at line 678 of file [ipc\\_stream](#).

### 16.146.2.3 send()

```
l4_msgtag_t L4::Ipc::Ostream::send (
    l4_cap_idx_t dst,
    long proto = 0,
    unsigned flags = 0) [inline]
```

Send the message via IPC to the given receiver.

#### Parameters

<i>dst</i>	The destination for the message.
<i>proto</i>	Protocol to use.

<i>flags</i>	Flags to use.
--------------	---------------

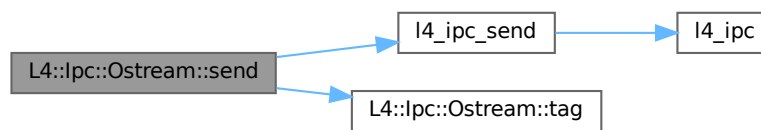
#### Returns

The syscall return tag.

Definition at line 959 of file [ipc\\_stream](#).

References [L4\\_IPC\\_NEVER](#), [l4\\_ipc\\_send\(\)](#), [L4\\_MSGTAG\\_FLAGS](#), and [tag\(\)](#).

Here is the call graph for this function:



#### 16.146.2.4 tag() [1/2]

```
l4_msgtag_t & L4::Ipc::Ostream::tag () [inline]
```

Extract a reference to the [L4](#) message tag from the stream.

#### Returns

A reference to the [L4](#) message tag.

Definition at line 713 of file [ipc\\_stream](#).

#### 16.146.2.5 tag() [2/2]

```
l4_msgtag_t L4::Ipc::Ostream::tag () const [inline]
```

Extract the [L4](#) message tag from the stream.

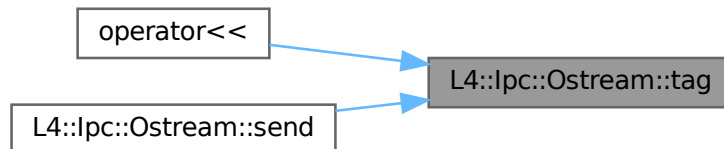
**Returns**

The extracted [L4](#) message tag.

Definition at line [706](#) of file [ipc\\_stream](#).

Referenced by [operator<<\(\)](#), and [send\(\)](#).

Here is the caller graph for this function:



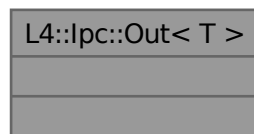
The documentation for this class was generated from the following file:

- [l4/cxx/ipc\\_stream](#)

## 16.147 L4::lpc::Out< T > Struct Template Reference

Mark an argument as a output value in an RPC signature.

Collaboration diagram for L4::lpc::Out< T >:



### 16.147.1 Detailed Description

```
template<typename T>
struct L4::lpc::Out< T >
```

Mark an argument as a output value in an RPC signature.

#### Template Parameters

<i>T</i>	The original type of the argument.
----------	------------------------------------

**Note**

The use of `Out<>` is usually not needed, because typical out-put data types in C++ (pointers to non-const objects or non-const references are interpreted as output values anyway. However, there are some data types, such as returned capabilities that can be marked as such by using `Out<>`.

Definition at line 31 of file [ipc\\_types](#).

The documentation for this struct was generated from the following file:

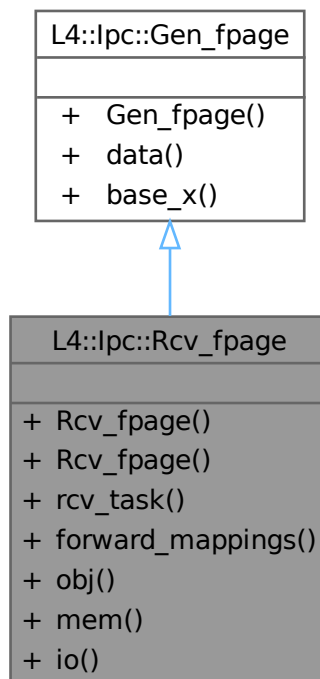
- [l4/sys/cxx/ipc\\_types](#)

## 16.148 L4::lpc::Rcv\_fpage Class Reference

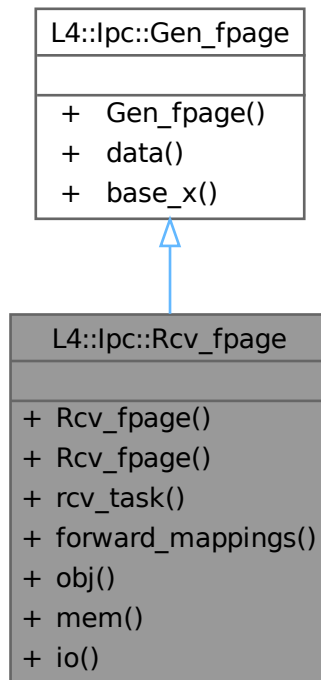
Non-small receive item.

```
#include <ipc_types>
```

Inheritance diagram for L4::lpc::Rcv\_fpage:



Collaboration diagram for L4::lpc::Rcv\_fpage:



### Public Member Functions

- **Rcv\_fpage** () noexcept  
*Construct a void receive item.*
- **Rcv\_fpage** (l4\_fpage\_t const &fp, l4\_addr\_t snd\_base=0, l4\_cap\_idx\_t rcv\_task=L4\_INVALID\_CAP) noexcept  
*Construct a non-small receive item.*
- **l4\_cap\_idx\_t rcv\_task** () const  
*Get the capability index of the destination task for received capabilities.*
- **bool forward\_mappings** () const noexcept  
*Check if rcv\_task() shall be used as destination for received capabilities.*

### Public Member Functions inherited from L4::lpc::Gen\_fpage

- **Gen\_fpage** (l4\_umword\_t base, l4\_umword\_t data) noexcept  
*Construct from raw values.*
- **l4\_umword\_t data** () const noexcept  
*Return the raw flexpage descriptor.*
- **l4\_umword\_t base\_x** () const noexcept  
*Return the raw base descriptor.*

## Static Public Member Functions

- static [Rcv\\_fpage obj](#) ([l4\\_cap\\_idx\\_t](#) base, int order, [l4\\_addr\\_t](#) snd\_base=0, [L4::Cap](#)< void > [rcv\\_task](#)=[L4::Cap](#)< void >::Invalid) noexcept  
*Construct a non-small receive item for the object space.*
- static [Rcv\\_fpage mem](#) ([l4\\_addr\\_t](#) base, int order, [l4\\_addr\\_t](#) snd\_base=0, [L4::Cap](#)< void > [rcv\\_task](#)=[L4::Cap](#)< void >::Invalid) noexcept  
*Construct a receive item for the memory space.*
- static [Rcv\\_fpage io](#) (unsigned long base, int order, [l4\\_addr\\_t](#) snd\_base=0, [L4::Cap](#)< void > [rcv\\_task](#)=[L4::Cap](#)< void >::Invalid) noexcept  
*Construct a receive item for the I/O port space.*

## Additional Inherited Members

## Public Types inherited from [L4::lpc::Gen\\_fpage](#)

- enum [Type](#) { [Special](#) = L4\_FPAGE\_SPECIAL << 4 , [Memory](#) = L4\_FPAGE\_MEMORY << 4 , [Io](#) = L4\_FPAGE\_IO << 4 , [Obj](#) = L4\_FPAGE\_OBJ << 4 }
- Type of mapping object, see [L4\\_fpage\\_type](#).*

### 16.148.1 Detailed Description

Non-small receive item.

This class represents a non-small receive item. A receive item is a message item in the buffer registers of the UTCB of the receiver (see [l4\\_utcb\\_br\(\)](#)).

Definition at line 544 of file [ipc\\_types](#).

### 16.148.2 Constructor & Destructor Documentation

#### 16.148.2.1 [Rcv\\_fpage\(\)](#)

```
L4::Ipc::Rcv_fpage::Rcv_fpage (
    l4\_fpage\_t const & fp,
    l4\_addr\_t snd_base = 0,
    l4\_cap\_idx\_t rcv_task = L4\_INVALID\_CAP) [inline], [noexcept]
```

Construct a non-small receive item.

#### Parameters

<i>fp</i>	Flexpage defining where and which kind of capabilities may be received.
<i>snd_base</i>	Reserved; should be zero.
<i>rcv_task</i>	Optional destination task for received capabilities. If invalid, capabilities are received in the invoking task.

Definition at line 561 of file [ipc\\_types](#).



References [L4\\_INVALID\\_CAP](#), [L4\\_ITEM\\_MAP](#), [L4\\_RCV\\_ITEM\\_FORWARD\\_MAPPINGS](#), and [rcv\\_task\(\)](#).

Here is the call graph for this function:



## 16.148.3 Member Function Documentation

### 16.148.3.1 io()

```

Rcv_fpage L4::Ipc::Rcv_fpage::io (
    unsigned long base,
    int order,
    l4_addr_t snd_base = 0,
    L4::Cap< void > rcv_task = L4::Cap<void>::Invalid) [inline], [static], [noexcept]
  
```

Construct a receive item for the I/O port space.

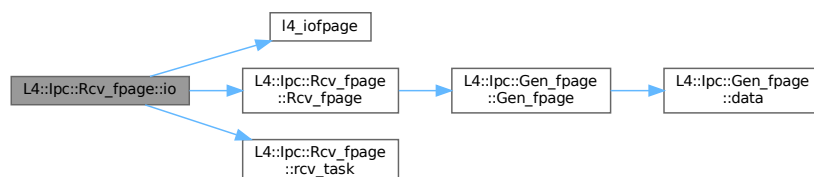
#### Parameters

<i>base</i>	Start of flexpage (see <a href="#">l4_iofpage()</a> ).
<i>order</i>	Log <sub>2</sub> size of flexpage (see <a href="#">l4_iofpage()</a> ).
<i>snd_base</i>	Reserved; should be zero.
<i>rcv_task</i>	Optional destination task for received capabilities. If invalid, capabilities are received in the invoking task.

Definition at line 609 of file [ipc\\_types](#).

References [L4::Cap\\_base::Invalid](#), [l4\\_iofpage\(\)](#), [Rcv\\_fpage\(\)](#), and [rcv\\_task\(\)](#).

Here is the call graph for this function:



### 16.148.3.2 mem()

```
Rcv_fpage L4::Ipc::Rcv_fpage::mem (
    l4_addr_t base,
    int order,
    l4_addr_t snd_base = 0,
    L4::Cap< void > rcv_task = L4::Cap<void>::Invalid) [inline], [static], [noexcept]
```

Construct a receive item for the memory space.

#### Parameters

<i>base</i>	Start of flexpage (see <a href="#">l4_fpage()</a> ).
<i>order</i>	Log <sub>2</sub> size of flexpage (see <a href="#">l4_fpage()</a> ).
<i>snd_base</i>	Reserved; should be zero.
<i>rcv_task</i>	Optional destination task for received capabilities. If invalid, capabilities are received in the invoking task.

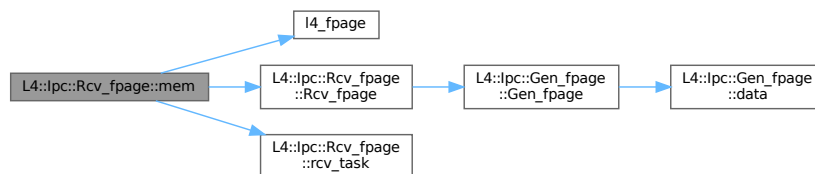
#### Examples

[examples/libs/l4re/streammap/client.cc](#).

Definition at line 594 of file [ipc\\_types](#).

References [L4::Cap\\_base::Invalid](#), [l4\\_fpage\(\)](#), [Rcv\\_fpage\(\)](#), and [rcv\\_task\(\)](#).

Here is the call graph for this function:



### 16.148.3.3 obj()

```
Rcv_fpage L4::Ipc::Rcv_fpage::obj (
    l4_cap_idx_t base,
    int order,
    l4_addr_t snd_base = 0,
    L4::Cap< void > rcv_task = L4::Cap<void>::Invalid) [inline], [static], [noexcept]
```

Construct a non-small receive item for the object space.

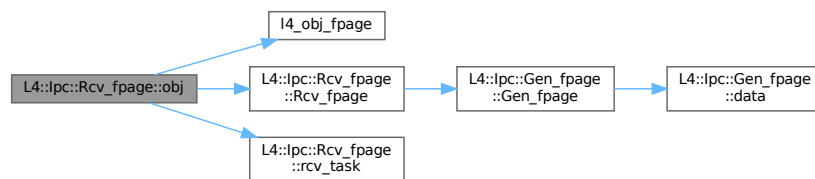
#### Parameters

<i>base</i>	Start of flexpage (see <a href="#">l4_obj_fpage()</a> ).
<i>order</i>	$\log_2$ size of flexpage (see <a href="#">l4_obj_fpage()</a> ).
<i>snd_base</i>	Reserved; should be zero.
<i>rcv_task</i>	Optional destination task for received capabilities. If invalid, capabilities are received in the invoking task.

Definition at line 578 of file [ipc\\_types](#).

References [L4::Cap\\_base::Invalid](#), [l4\\_obj\\_fpage\(\)](#), [Rcv\\_fpage\(\)](#), and [rcv\\_task\(\)](#).

Here is the call graph for this function:



#### 16.148.3.4 rcv\_task()

```
l4_cap_idx_t L4::lpc::Rcv_fpage::rcv_task () const [inline]
```

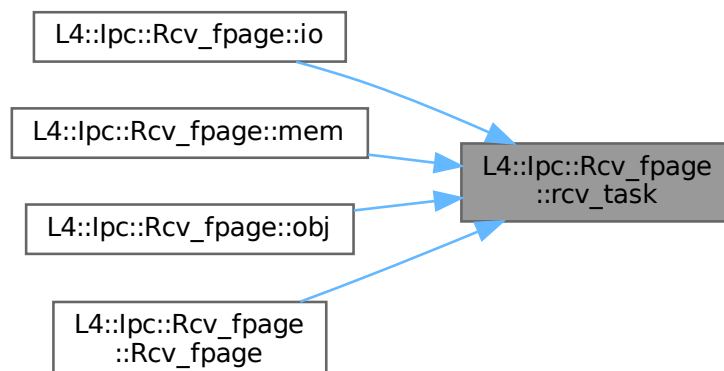
Get the capability index of the destination task for received capabilities.

Only relevant if [forward\\_mappings\(\)](#) is true.

Definition at line 620 of file [ipc\\_types](#).

Referenced by [io\(\)](#), [mem\(\)](#), [obj\(\)](#), and [Rcv\\_fpage\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

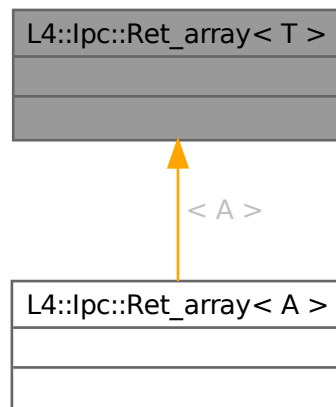
- [l4/sys/cxx/ipc\\_types](#)

## 16.149 L4::lpc::Ret\_array< T > Struct Template Reference

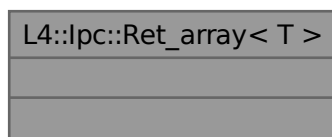
Dynamically sized output array of type T.

```
#include <ipc_ret_array>
```

Inheritance diagram for L4::lpc::Ret\_array< T >:



Collaboration diagram for L4::lpc::Ret\_array< T >:



### 16.149.1 Detailed Description

```
template<typename T>
struct L4::lpc::Ret_array< T >
```

Dynamically sized output array of type T.

#### Template Parameters

---

<i>T</i>	The data-type of each array element.
----------	--------------------------------------

[Ret\\_array<>](#) is a special dynamically sized output array where the number of transmitted elements is passed in the return value of the call (if positive)

Definition at line 23 of file [ipc\\_ret\\_array](#).

The documentation for this struct was generated from the following file:

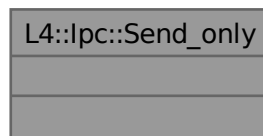
- [l4/sys/cxx/ipc\\_ret\\_array](#)

## 16.150 L4::lpc::Send\_only Struct Reference

RPC attribute for a send-only RPC.

```
#include <ipc_iface>
```

Collaboration diagram for L4::lpc::Send\_only:



### 16.150.1 Detailed Description

RPC attribute for a send-only RPC.

This class can be used as FLAGS parameter to [L4::lpc::Msg::Rpc\\_call](#) and [L4::lpc::Msg::Rpc\\_inline\\_call](#) templates and declares the RPC to use send-only semantics and timeouts.

Examples:

```
L4\_RPC(long, send, (unsigned value), L4::lpc::Send_only);
```

Definition at line 287 of file [ipc\\_iface](#).

The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc\\_iface](#)

## 16.151 L4::lpc::Small\_buf Class Reference

A receive item for receiving a single object capability.

```
#include <ipc_types>
```

Collaboration diagram for L4::lpc::Small\_buf:

L4::lpc::Small_buf
<ul style="list-style-type: none"> <li>+ Small_buf()</li> <li>+ Small_buf()</li> <li>+ raw()</li> </ul>

### Public Member Functions

- [Small\\_buf](#) ([L4::Cap](#)< void > cap, unsigned long flags=0) noexcept  
*Create a receive item from a C++ cap.*
- [Small\\_buf](#) ([l4\\_cap\\_idx\\_t](#) cap, unsigned long flags=0) noexcept  
*Create a receive item from a C cap.*
- [l4\\_umword\\_t](#) **raw** () const noexcept  
*Return the raw data.*

### 16.151.1 Detailed Description

A receive item for receiving a single object capability.

This class is the main abstraction for receiving object capabilities via [lpc::lstream](#). To receive an object capability, an instance of [Small\\_buf](#) that refers to an empty capability slot must be inserted into the [lpc::lstream](#) before the receive operation.

Definition at line 257 of file [ipc\\_types](#).

### 16.151.2 Constructor & Destructor Documentation

#### 16.151.2.1 Small\_buf() [1/2]

```
L4::Ipc::Small_buf::Small_buf (
    L4::Cap< void > cap,
    unsigned long flags = 0) [inline], [explicit], [noexcept]
```

Create a receive item from a C++ cap.

#### Parameters

<i>cap</i>	Capability slot where to save the capability.
<i>flags</i>	Receive buffer flags, see <a href="#">l4_msg_item_consts_t</a> . <a href="#">L4_RCV_ITEM_SINGLE_CAP</a> will always be set.

Definition at line 267 of file [ipc\\_types](#).

References [L4\\_RCV\\_ITEM\\_SINGLE\\_CAP](#).

### 16.151.2.2 Small\_buf() [2/2]

```
L4::Ipc::Small_buf::Small_buf (
    l4_cap_idx_t cap,
    unsigned long flags = 0) [inline], [explicit], [noexcept]
```

Create a receive item from a C cap.

#### Parameters

<i>cap</i>	Capability slot where to save the capability.
<i>flags</i>	Receive buffer flags, see <a href="#">l4_msg_item_consts_t</a> . <a href="#">L4_RCV_ITEM_SINGLE_CAP</a> will always be set.

Definition at line 274 of file [ipc\\_types](#).

References [L4\\_RCV\\_ITEM\\_SINGLE\\_CAP](#).

The documentation for this class was generated from the following file:

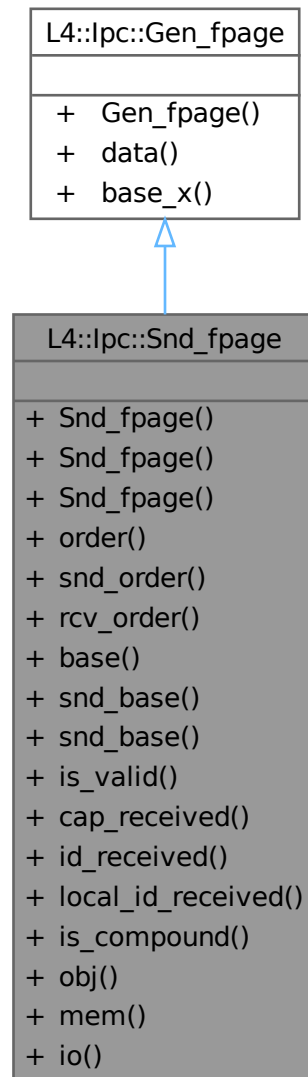
- [l4/sys/cxx/ipc\\_types](#)

## 16.152 L4::Ipc::Snd\_fpage Class Reference

Send item or return item.

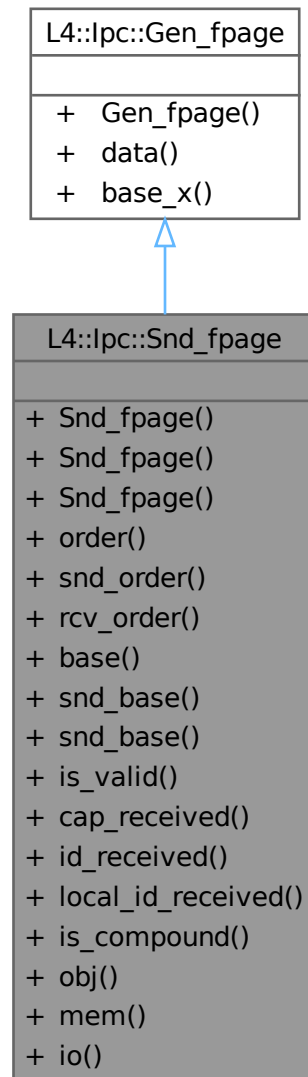
```
#include <ipc_types>
```

Inheritance diagram for L4::lpc::Snd\_fpage:





Collaboration diagram for L4::lpc::Snd\_fpage:



## Public Types

- enum [Map\\_type](#) { [Map](#) = L4\_MAP\_ITEM\_MAP , [Grant](#) = L4\_MAP\_ITEM\_GRANT }  
(Defined for send items only.) Kind of mapping.
- enum [Cacheopt](#) { [None](#) = 0 , [Cached](#) = L4\_FPAGE\_CACHEABLE << 4 , [Buffered](#) = L4\_FPAGE\_BUFFERABLE << 4 , [Uncached](#) = L4\_FPAGE\_UNCACHEABLE << 4 }  
(Defined for memory send items only.) Caching options, see [l4\\_fpage\\_cacheability\\_opt\\_t](#).
- enum [Continue](#) { [Single](#) = 0 , [Last](#) = 0 , [More](#) = L4\_ITEM\_CONT , [Compound](#) = L4\_ITEM\_CONT }  
Specify if the following item is associated with the same receive item as this one, see [L4\\_ITEM\\_CONT](#).

## Public Types inherited from [L4::lpc::Gen\\_fpage](#)

- enum [Type](#) { [Special](#) = L4\_FPAGE\_SPECIAL << 4 , [Memory](#) = L4\_FPAGE\_MEMORY << 4 , [Io](#) = L4\_FPAGE\_IO << 4 , [Obj](#) = L4\_FPAGE\_OBJ << 4 }

*Type of mapping object, see [L4\\_fpage\\_type](#).*

## Public Member Functions

- [Snd\\_fpage](#) ([l4\\_umword\\_t](#) base=0, [l4\\_umword\\_t](#) data=0) noexcept  
*Construct from raw values.*
- [Snd\\_fpage](#) ([l4\\_fpage\\_t](#) const &fp, [l4\\_addr\\_t](#) snd\_base=0, [Map\\_type](#) map\_type=[Map](#), [Cacheopt](#) cache=[None](#), [Continue](#) cont=[Last](#)) noexcept  
*Construct a send item for the memory space.*
- [Snd\\_fpage](#) ([L4::Cap](#)< void > cap, unsigned rights, [Map\\_type](#) map\_type=[Map](#)) noexcept  
*Construct a send item for the object space.*
- unsigned [order](#) () const noexcept  
*(Defined only if send item or if [local\\_id\\_received\(\)](#) is true.) Get log<sub>2</sub> size.*
- unsigned [snd\\_order](#) () const noexcept  
*(Defined only if send item or if [local\\_id\\_received\(\)](#) is true.) Get log<sub>2</sub> size.*
- unsigned [rcv\\_order](#) () const noexcept  
*(Defined for return items only.) Get log<sub>2</sub> size.*
- [l4\\_addr\\_t](#) [base](#) () const noexcept  
*(Defined only if send item or if [local\\_id\\_received\(\)](#) is true.) Get the start of the item (i.e., the start of its flexpage).*
- [l4\\_addr\\_t](#) [snd\\_base](#) () const noexcept  
*Get the position in receive window for the case that this item has a different size than the corresponding receive item.*
- void [snd\\_base](#) ([l4\\_addr\\_t](#) b) noexcept  
*Set the position in receive window for the case that this item has a different size than the corresponding receive item.*
- bool [is\\_valid](#) () const noexcept  
*Check if the capability is valid.*
- bool [cap\\_received](#) () const noexcept  
*(Defined for return items only.) Check if at least one object capability has been mapped for this item.*
- bool [id\\_received](#) () const noexcept  
*(Defined for return items only.) Check if an IPC gate label has been received instead of a mapping.*
- bool [local\\_id\\_received](#) () const noexcept  
*(Defined for return items only.) Check if a raw object flexpage has been received instead of a mapping.*
- bool [is\\_compound](#) () const noexcept  
*Check if the item has the compound bit set, see [Continue](#).*

## Public Member Functions inherited from [L4::lpc::Gen\\_fpage](#)

- [Gen\\_fpage](#) ([l4\\_umword\\_t](#) base, [l4\\_umword\\_t](#) data) noexcept  
*Construct from raw values.*
- [l4\\_umword\\_t](#) [data](#) () const noexcept  
*Return the raw flexpage descriptor.*
- [l4\\_umword\\_t](#) [base\\_x](#) () const noexcept  
*Return the raw base descriptor.*

## Static Public Member Functions

- static [Snd\\_fpage obj](#) ([l4\\_cap\\_idx\\_t](#) base, int [order](#), unsigned char rights, [l4\\_addr\\_t](#) snd\_base=0, [Map\\_type](#) map\_type=[Map](#), [Continue](#) cont=[Last](#)) noexcept  
Construct a send item for the object space.
- static [Snd\\_fpage mem](#) ([l4\\_addr\\_t](#) base, int [order](#), unsigned char rights, [l4\\_addr\\_t](#) snd\_base=0, [Map\\_type](#) map\_type=[Map](#), [Cacheopt](#) cache=[None](#), [Continue](#) cont=[Last](#)) noexcept  
Construct a send item for the memory space.
- static [Snd\\_fpage io](#) (unsigned long base, int [order](#), unsigned char rights, [l4\\_addr\\_t](#) snd\_base=0, [Map\\_type](#) map\_type=[Map](#), [Continue](#) cont=[Last](#)) noexcept  
Construct a send item for the I/O port space.

### 16.152.1 Detailed Description

Send item or return item.

This class represents a typed message item in the message registers of the UTCB. If it is provided by the sender, then it is a *send item*. If it is provided by the kernel during IPC, it is a *return item*.

Note that some members are dedicated for send items only or return items only.

Definition at line 323 of file [ipc\\_types](#).

### 16.152.2 Member Enumeration Documentation

#### 16.152.2.1 Cacheopt

```
enum L4::Ipc::Snd_fpage::Cacheopt
```

(Defined for memory send items only.) Caching options, see [l4\\_fpage\\_cacheability\\_opt\\_t](#).

#### Enumerator

None	Copy options from sender.
Cached	Cacheability option to enable caches for the mapping.
Buffered	Cacheability option to enable buffered writes for the mapping.
Uncached	Cacheability option to disable caching for the mapping.

Definition at line 336 of file [ipc\\_types](#).

#### 16.152.2.2 Continue

```
enum L4::Ipc::Snd_fpage::Continue
```

Specify if the following item is associated with the same receive item as this one, see [L4\\_ITEM\\_CONT](#).

#### Enumerator

Single	Inverse of <a href="#">Compound</a> .
Last	Inverse of <a href="#">More</a> .
More	Alias for <a href="#">Compound</a> .
Compound	Denote that the following item shall be put into the same receive item as this one.

Definition at line [346](#) of file [ipc\\_types](#).

### 16.152.2.3 Map\_type

```
enum L4::Ipc::Snd_fpage::Map_type
```

(Defined for send items only.) Kind of mapping.

#### Enumerator

Map	Flag as usual <i>map</i> operation.
Grant	<p>Flag as <i>grant</i> instead of <i>map</i> operation. This means, the sender delegates access to the receiver and the kernel removes the rights from the sender (basically a move operation). The mapping in the receiver gets the new parent of any child mappings of the mapping of the sender. Rights revocation via send item/flexpage is <i>not</i> guaranteed to be applied to descendant mappings in case of grant. See <a href="#">Spaces and Mappings</a> for more details on map/grant.</p> <p><b>Note</b></p> <p>The grant operation is not performed if the resulting rights of the receiver mapping would not contain the <a href="#">L4_CAP_FPAGE_R</a> bit (for object capabilities) or none of the <a href="#">L4_FPAGE_RWX</a> bits (memory and IO ports). In that case, the mapping is not created in the receiver space and not removed from the sender space.</p> <p>If the removal of the whole mapping from the sender is not possible because the size of the mapped frame at the sender exceeds the size defined by the send or receive flexpage, the grant operation is turned into a regular map operation and the mapping is <i>not</i> removed from the sender. This would happen if, for example, a smaller part of an <a href="#">L4</a> superpage mapping shall be granted.</p>

Definition at line [328](#) of file [ipc\\_types](#).

## 16.152.3 Constructor & Destructor Documentation

### 16.152.3.1 Snd\_fpage() [1/2]

```
L4::Ipc::Snd_fpage::Snd_fpage (
    l4_fpage_t const & fp,
    l4_addr_t snd_base = 0,
    Map_type map_type = Map,
    Cacheopt cache = None,
    Continue cont = Last) [inline], [noexcept]
```

Construct a send item for the memory space.

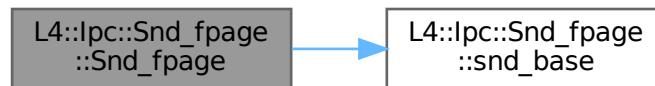
#### Parameters

<i>fp</i>	Memory flexpage defining which range and kind of capabilities shall be sent (see <a href="#">l4_fpage()</a> ).
<i>snd_base</i>	Position in receive window in case it has a different size than <i>fp</i> .
<i>map_type</i>	See <a href="#">Map_type</a> .
<i>cache</i>	See <a href="#">Cacheopt</a> .
<i>cont</i>	See <a href="#">Continue</a> .

Definition at line 370 of file [ipc\\_types](#).

References [L4\\_ITEM\\_MAP](#), [Last](#), [Map](#), [None](#), and [snd\\_base\(\)](#).

Here is the call graph for this function:



### 16.152.3.2 Snd\_fpage() [2/2]

```

L4::Ipc::Snd_fpage::Snd_fpage (
    L4::Cap< void > cap,
    unsigned rights,
    Map_type map_type = Map) [inline], [noexcept]
  
```

Construct a send item for the object space.

#### Parameters

<i>cap</i>	Capability to be sent.
<i>rights</i>	Permissions to be transferred. See <a href="#">L4_cap_fpage_rights</a> and <a href="#">L4_obj_fpage_ctl</a> .
<i>map_type</i>	See <a href="#">Map_type</a> .

Definition at line 386 of file [ipc\\_types](#).

References [L4\\_ITEM\\_MAP](#), and [Map](#).

## 16.152.4 Member Function Documentation

### 16.152.4.1 cap\_received()

```
bool L4::Ipc::Snd_fpage::cap_received () const [inline], [noexcept]
```

(Defined for return items only.) Check if at least one object capability has been mapped for this item.

The capabilities themselves can then be retrieved from the cap slots that have been provided in the receive operation.

#### Note

If this function returns `true` and the receive window covers more than one capability slot, then it is not possible to determine which slots actually got capabilities mapped from the sender.

If the received capabilities do not have type object (see [L4\\_FPAGE\\_OBJ](#)), then this function returns `false`.

Definition at line 496 of file [ipc\\_types](#).

### 16.152.4.2 id\_received()

```
bool L4::Ipc::Snd_fpage::id_received () const [inline], [noexcept]
```

(Defined for return items only.) Check if an IPC gate label has been received instead of a mapping.

If the [L4\\_RCV\\_ITEM\\_LOCAL\\_ID](#) flag has been set by the receiver, the conditions for [local\\_id\\_received\(\)](#) do not apply, the sender sent an IPC gate capability, and the receiving thread is in the same task as the thread that is attached to the IPC gate, then no mapping is done for this item and only the bitwise OR (|) of the label of the IPC gate and the special and write permission ([L4\\_CAP\\_FPAGE\\_S](#) and [L4\\_CAP\\_FPAGE\\_W](#)) that would have been mapped is received.

The bitwise OR of the label and the permissions can be retrieved with [Gen\\_fpage::data\(\)](#).

Definition at line 512 of file [ipc\\_types](#).

### 16.152.4.3 io()

```
Snd_fpage L4::Ipc::Snd_fpage::io (
    unsigned long base,
    int order,
    unsigned char rights,
    l4_addr_t snd_base = 0,
    Map_type map_type = Map,
    Continue cont = Last) [inline], [static], [noexcept]
```

Construct a send item for the I/O port space.

#### Parameters

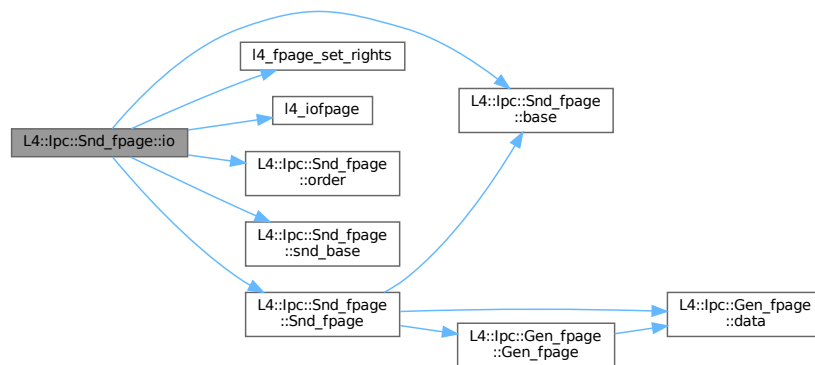
<i>base</i>	Start of flexpage (see <a href="#">l4_iofpage()</a> ).
-------------	--------------------------------------------------------

<i>order</i>	Log <sub>2</sub> size of flexpage (see <a href="#">l4_iofpage()</a> ).
<i>rights</i>	Permissions of flexpage (see <a href="#">L4_fpage_rights</a> ).
<i>snd_base</i>	Position in receive window in case it has a different size than 1 << <a href="#">order</a> .
<i>map_type</i>	See <a href="#">Map_type</a> .
<i>cont</i>	See <a href="#">Continue</a> .

Definition at line 445 of file [ipc\\_types](#).

References [base\(\)](#), [l4\\_fpage\\_set\\_rights\(\)](#), [l4\\_iofpage\(\)](#), [Last](#), [Map](#), [None](#), [order\(\)](#), [snd\\_base\(\)](#), and [Snd\\_fpage\(\)](#).

Here is the call graph for this function:



#### 16.152.4.4 is\_compound()

```
bool L4::Ipc::Snd_fpage::is_compound () const [inline], [noexcept]
```

Check if the item has the compound bit set, see [Continue](#).

A set compound bit means the next message item of the same type will be mapped to the same receive buffer as this message item.

Definition at line 535 of file [ipc\\_types](#).

#### 16.152.4.5 local\_id\_received()

```
bool L4::Ipc::Snd_fpage::local_id_received () const [inline], [noexcept]
```

(Defined for return items only.) Check if a raw object flexpage has been received instead of a mapping.

If the [L4\\_RCV\\_ITEM\\_LOCAL\\_ID](#) flag has been set by the receiver, and sender and receiver are in the same task, then no mapping is done for this item and only the raw flexpage ([l4\\_fpage\\_t](#)) is received.

This function checks if this is the case and if it is an object flexpage.

The flexpage can be retrieved with [Gen\\_fpage::data\(\)](#).

#### Note

If a raw flexpage was received, but it does not have type object (see [L4\\_FPAGE\\_OBJ](#)), then this function returns `false`.

Definition at line 528 of file [ipc\\_types](#).

### 16.152.4.6 mem()

```
Snd_fpage L4::Ipc::Snd_fpage::mem (
    l4_addr_t base,
    int order,
    unsigned char rights,
    l4_addr_t snd_base = 0,
    Map_type map_type = Map,
    Cacheopt cache = None,
    Continue cont = Last) [inline], [static], [noexcept]
```

Construct a send item for the memory space.

#### Parameters

<i>base</i>	Start of flexpage (see <a href="#">l4_fpage()</a> ).
<i>order</i>	Log <sub>2</sub> size of flexpage (see <a href="#">l4_fpage()</a> ).
<i>rights</i>	Permissions of flexpage (see <a href="#">l4_fpage()</a> ).
<i>snd_base</i>	Position in receive window in case it has a different size than 1 << <a href="#">order</a> .
<i>map_type</i>	See <a href="#">Map_type</a> .
<i>cache</i>	See <a href="#">Cacheopt</a> .
<i>cont</i>	See <a href="#">Continue</a> .

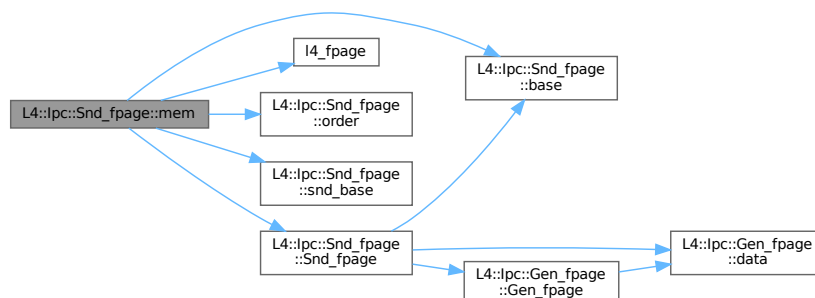
#### Examples

[examples/libs/l4re/streammap/server.cc](#).

Definition at line [424](#) of file [ipc\\_types](#).

References [base\(\)](#), [l4\\_fpage\(\)](#), [Last](#), [Map](#), [None](#), [order\(\)](#), [snd\\_base\(\)](#), and [Snd\\_fpage\(\)](#).

Here is the call graph for this function:





## 16.152.4.7 obj()

```

Snd_fpage L4::Ipc::Snd_fpage::obj (
    l4_cap_idx_t base,
    int order,
    unsigned char rights,
    l4_addr_t snd_base = 0,
    Map_type map_type = Map,
    Continue cont = Last) [inline], [static], [noexcept]

```

Construct a send item for the object space.

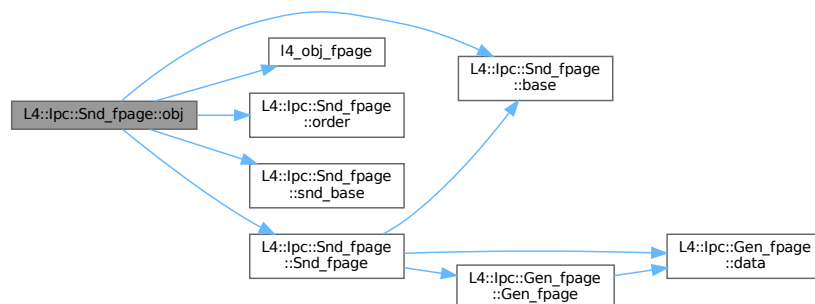
## Parameters

<i>base</i>	Start of flexpage (see <a href="#">l4_obj_fpage()</a> ).
<i>order</i>	$\log_2$ size of flexpage (see <a href="#">l4_obj_fpage()</a> ).
<i>rights</i>	Permissions of flexpage (see <a href="#">l4_obj_fpage()</a> ).
<i>snd_base</i>	Position in receive window in case it has a different size than $1 \ll order$ .
<i>map_type</i>	See <a href="#">Map_type</a> .
<i>cont</i>	See <a href="#">Continue</a> .

Definition at line 402 of file [ipc\\_types](#).

References [base\(\)](#), [l4\\_obj\\_fpage\(\)](#), [Last](#), [Map](#), [None](#), [order\(\)](#), [snd\\_base\(\)](#), and [Snd\\_fpage\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

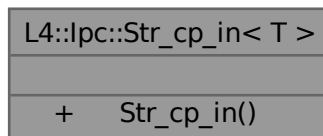
- [l4/sys/cxx/ipc\\_types](#)

## 16.153 L4::lpc::Str\_cp\_in< T > Class Template Reference

Abstraction for extracting a zero-terminated string from an [lpc::lstream](#).

```
#include <ipc_stream>
```

Collaboration diagram for L4::lpc::Str\_cp\_in< T >:



### Public Member Functions

- [Str\\_cp\\_in](#) (T \*v, unsigned long &size)  
Create a buffer for extracting an array from an [lpc::lstream](#).

### 16.153.1 Detailed Description

```
template<typename T>
class L4::lpc::Str_cp_in< T >
```

Abstraction for extracting a zero-terminated string from an [lpc::lstream](#).

An instance of [Str\\_cp\\_in](#) can be used to extract a zero-terminated string an [lpc::lstream](#). The data from the received message is thereby copied to the given buffer and size is set to the number of characters found in the stream. The string is zero terminated in any circumstances. When the given buffer is smaller than the received string the last byte in the buffer will be the zero terminator. In the case the received string is shorter than the given buffer the zero termination will be placed behind the received data. This provides a zero-terminated result even in cases where the sender did not provide proper termination or in cases of too small receiver buffers.

See also

[str\\_cp\\_in\(\)](#).

Definition at line 178 of file [ipc\\_stream](#).

## 16.153.2 Constructor & Destructor Documentation

### 16.153.2.1 Str\_cp\_in()

```
template<typename T>
L4::lpc::Str_cp_in< T >::Str_cp_in (
    T * v,
    unsigned long & size) [inline]
```

Create a buffer for extracting an array from an [lpc::lstream](#).

### Parameters

	<i>v</i>	The buffer for string.
<i>in, out</i>	<i>size</i>	Input: The number of bytes available in <i>v</i> Output: The number of bytes received (including the terminator).

Definition at line 189 of file [ipc\\_stream](#).

The documentation for this class was generated from the following file:

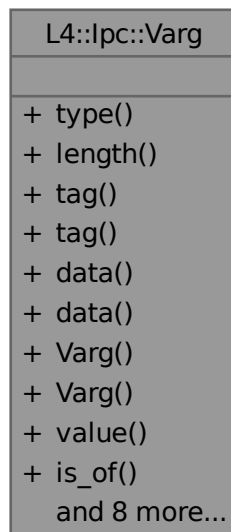
- [l4/cxx/ipc\\_stream](#)

## 16.154 L4::lpc::Varg Class Reference

Variably sized RPC argument.

```
#include <ipc_varg>
```

Collaboration diagram for L4::lpc::Varg:



### Public Types

- typedef [l4\\_umword\\_t](#) **Tag**  
*The data type for the tag.*

## Public Member Functions

- L4\_varg\_type [type](#) () const
- unsigned [length](#) () const  
*Get the size of the RPC argument.*
- [Tag](#) tag () const
- void **tag** ([Tag](#) tag)  
*Set [Varg](#) tag (usually from message).*
- void **data** (char const \*d)  
*Set [Varg](#) to indirect data value (usually in UTCB).*
- char const \* [data](#) () const
- **Varg** ()=default  
*Make uninitialized [Varg](#).*
- **Varg** (L4\_varg\_type t, void const \*v, int len)  
*Make an indirect varg.*
- template<typename V>  
Va\_type< V >::Ret\_value [value](#) () const
- template<typename T>  
bool [is\\_of](#) () const
- bool [is\\_nil](#) () const
- bool [is\\_of\\_int](#) () const
- template<typename T>  
bool [get\\_value](#) (typename Va\_type< T >::Value \*v) const  
*Get the value of the [Varg](#) as type T.*
- template<typename T>  
void **set\_value** (void const \*d)  
*Set to indirect value of type T.*
- template<typename T>  
void **set\_direct\_value** (T val, typename L4::Types::Enable\_if< sizeof(T)<=sizeof(char const \*)>, bool >::type=true)  
*Set to directly stored value of type T.*
- template<typename T>  
**Varg** (T const \*[data](#))  
*Make [Varg](#) from indirect value (pointer).*
- **Varg** (char const \*[data](#))  
*Make [Varg](#) from null-terminated string.*
- template<typename T>  
**Varg** (T [data](#), typename L4::Types::Enable\_if< sizeof(T)<=sizeof(char const \*)>, bool >::type=true)  
*Make [Varg](#) from direct value.*

### 16.154.1 Detailed Description

Variably sized RPC argument.

Definition at line 96 of file [ipc\\_varg](#).

### 16.154.2 Member Function Documentation

#### 16.154.2.1 data()

```
char const * L4::Ipc::Varg::data () const [inline]
```

#### Returns

pointer to the data, also safe for direct data

Definition at line 123 of file [ipc\\_varg](#).

### 16.154.2.2 get\_value()

```
template<typename T>
bool L4::Ipc::Varg::get_value (
    typename Va_type< T >::Value * v) const [inline]
```

Get the value of the [Varg](#) as type T.

#### Template Parameters

<i>T</i>	The expected type of the <a href="#">Varg</a> .
----------	-------------------------------------------------

#### Parameters

<i>v</i>	Pointer to store the value
----------	----------------------------

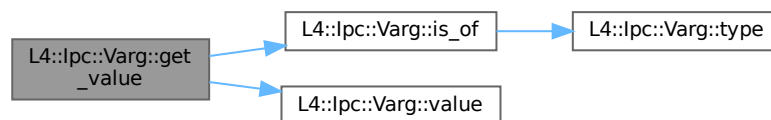
#### Returns

true when the [Varg](#) is of type T, false if not

Definition at line 185 of file [ipc\\_varg](#).

References [is\\_of\(\)](#), and [value\(\)](#).

Here is the call graph for this function:



### 16.154.2.3 is\_nil()

```
bool L4::Ipc::Varg::is_nil () const [inline]
```

**Returns**

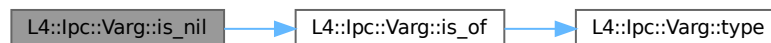
true if the [Varg](#) is of nil type.

Definition at line 172 of file [ipc\\_varg](#).

References [is\\_of\(\)](#).

Referenced by [L4::Ipc::Varg\\_list\\_ref::Iterator::equals\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**16.154.2.4 is\_of()**

```
template<typename T>
bool L4::Ipc::Varg::is_of () const [inline]
```

**Returns**

true if the [Varg](#) is of type T

Definition at line 169 of file [ipc\\_varg](#).

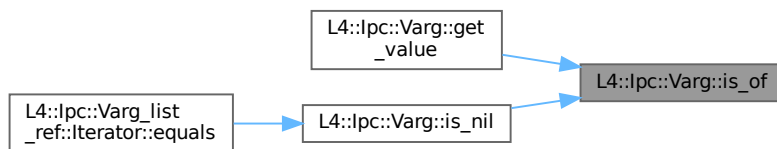
References [type\(\)](#).

Referenced by [get\\_value\(\)](#), and [is\\_nil\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 16.154.2.5 is\_of\_int()

```
bool L4::Ipc::Varg::is_of_int () const [inline]
```

##### Returns

true if the [Varg](#) is an integer type (signed or unsigned).

Definition at line 175 of file [ipc\\_varg](#).

References [type\(\)](#).

Here is the call graph for this function:



### 16.154.2.6 length()

```
unsigned L4::Ipc::Varg::length () const [inline]
```

Get the size of the RPC argument.

#### Returns

The size of the RPC argument

Definition at line 114 of file [ipc\\_varg](#).

### 16.154.2.7 tag()

```
Tag L4::Ipc::Varg::tag () const [inline]
```

#### Returns

the tag value (the Direct\_data bit masked)

Definition at line 116 of file [ipc\\_varg](#).

Referenced by [tag\(\)](#).

Here is the caller graph for this function:



### 16.154.2.8 type()

```
L4_varg_type L4::Ipc::Varg::type () const [inline]
```

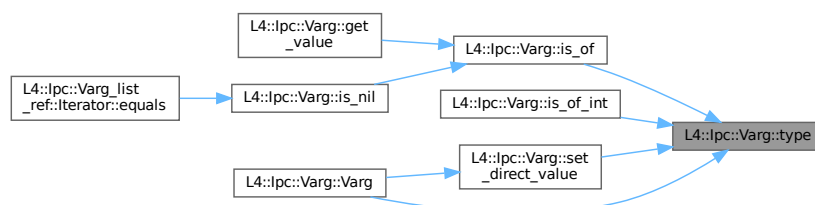
#### Returns

the type field of the tag

Definition at line 109 of file [ipc\\_varg](#).

Referenced by [is\\_of\(\)](#), [is\\_of\\_int\(\)](#), [set\\_direct\\_value\(\)](#), and [Varg\(\)](#).

Here is the caller graph for this function:





**16.154.2.9 value()**

```
template<typename V>
Va_type< V >::Ret_value L4::Ipc::Varg::value () const [inline]
```

**Template Parameters**

<b>V</b>	The data type of the value to retrieve.
----------	-----------------------------------------

**Precondition**

The [Varg](#) must be of type *V* (otherwise the result is unpredictable).

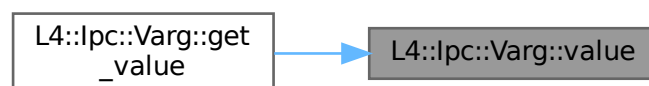
**Returns**

The value of the [Varg](#) as type *V*.

Definition at line [155](#) of file [ipc\\_varg](#).

Referenced by [get\\_value\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

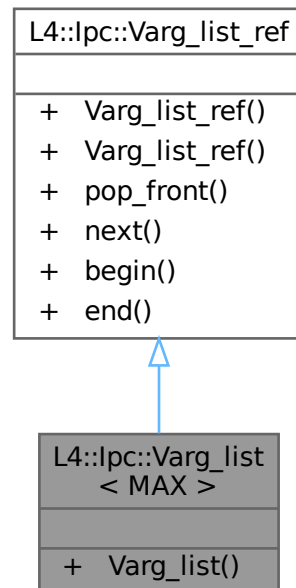
- [l4/sys/cxx/ipc\\_varg](#)

**16.155 L4::Ipc::Varg\_list< MAX > Class Template Reference**

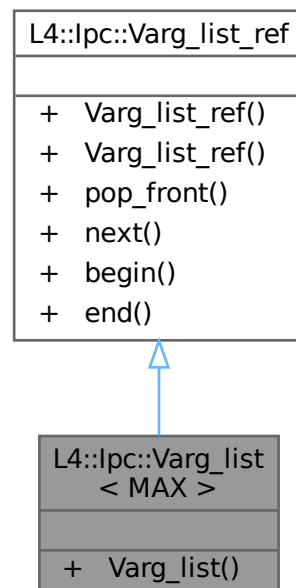
Self-contained list of variable-sized RPC parameters.

```
#include <ipc_varg>
```

Inheritance diagram for L4::lpc::Varg\_list< MAX >:



Collaboration diagram for L4::lpc::Varg\_list< MAX >:



## Public Member Functions

- **Varg\_list** ([Varg\\_list\\_ref](#) const &r)  
*Create a parameter list as a copy from a referencing list.*

## Public Member Functions inherited from [L4::lpc::Varg\\_list\\_ref](#)

- **Varg\_list\_ref** ()=default  
*Create an empty parameter list.*
- [Varg\\_list\\_ref](#) (void const \*start, void const \*end)  
*Create a parameter list over a given memory region.*
- [Varg](#) **pop\_front** ()  
*Get the next parameter in the list.*
- [Varg](#) **next** ()  
*Get the next parameter in the list.*
- [Iterator](#) **begin** () const  
*Returns an iterator to the first [Varg](#).*
- [Iterator](#) **end** () const  
*Returns the end of the list.*

### 16.155.1 Detailed Description

```
template<unsigned MAX>
class L4::lpc::Varg_list< MAX >
```

Self-contained list of variable-sized RPC parameters.

Works like [Varg\\_list\\_ref](#) but contains a full copy of the data. Use this as a parameter in server functions, if the handler function needs to use the UTCB (e.g. while sending further IPC).

Definition at line 411 of file [ipc\\_varg](#).

The documentation for this class was generated from the following file:

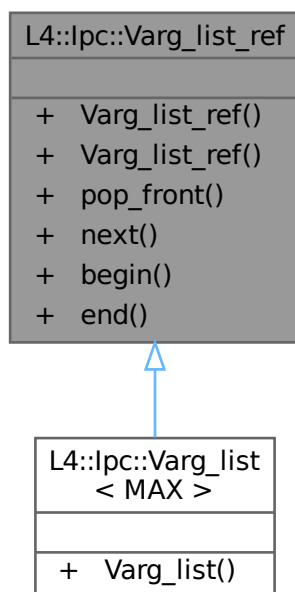
- I4/sys/cxx/ipc\_varg

## 16.156 L4::lpc::Varg\_list\_ref Class Reference

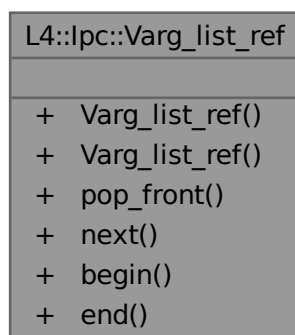
List of variable-sized RPC parameters as received by the server.

```
#include <ipc_varg>
```

Inheritance diagram for L4::lpc::Varg\_list\_ref:



Collaboration diagram for L4::lpc::Varg\_list\_ref:



## Data Structures

- class [Iterator](#)  
*Iterator* for *Valists*.

## Public Member Functions

- **Varg\_list\_ref** ()=default  
*Create an empty parameter list.*
- **Varg\_list\_ref** (void const \*start, void const \*end)  
*Create a parameter list over a given memory region.*
- **Varg pop\_front** ()  
*Get the next parameter in the list.*
- **Varg next** ()  
*Get the next parameter in the list.*
- **Iterator begin** () const  
*Returns an iterator to the first [Varg](#).*
- **Iterator end** () const  
*Returns the end of the list.*

### 16.156.1 Detailed Description

List of variable-sized RPC parameters as received by the server.

The list can be traversed exactly once using [next\(\)](#).

This is a reference list, where the returned [Varg](#) point to data in the underlying storage, conventionally the UTCB. This type should only be used in server functions when the implementation can ensure that all content is read before the UTCB is reused (e.g. for IPC), otherwise use [Varg\\_list](#).

Definition at line 253 of file [ipc\\_varg](#).

### 16.156.2 Constructor & Destructor Documentation

#### 16.156.2.1 Varg\_list\_ref()

```
L4::Ipc::Varg_list_ref::Varg_list_ref (
    void const * start,
    void const * end) [inline]
```

Create a parameter list over a given memory region.

#### Parameters

<i>start</i>	Pointer to start of the parameter list.
<i>end</i>	Pointer to end of the list (inclusive).

Definition at line 332 of file [ipc\\_varg](#).

References [end\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

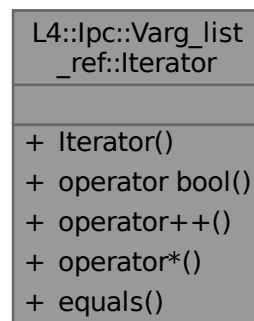
- l4/sys/cxx/ipc\_varg

## 16.157 L4::ipc::Varg\_list\_ref::Iterator Class Reference

[Iterator](#) for Valists.

```
#include <ipc_varg>
```

Collaboration diagram for L4::ipc::Varg\_list\_ref::Iterator:



### Public Member Functions

- **Iterator** (Iter\_state const &s)  
*Create a new iterator.*
- **operator bool** () const  
*validity check for the iterator*
- **Iterator** & **operator++** ()  
*increment iterator to the next arg*
- **Varg** **operator\*** () const  
*dereference the iterator, get [Varg](#)*
- **bool** **equals** ([Iterator](#) const &o) const  
*check for equality*

### 16.157.1 Detailed Description

[Iterator](#) for Valists.

Definition at line 338 of file [ipc\\_varg](#).

The documentation for this class was generated from the following file:

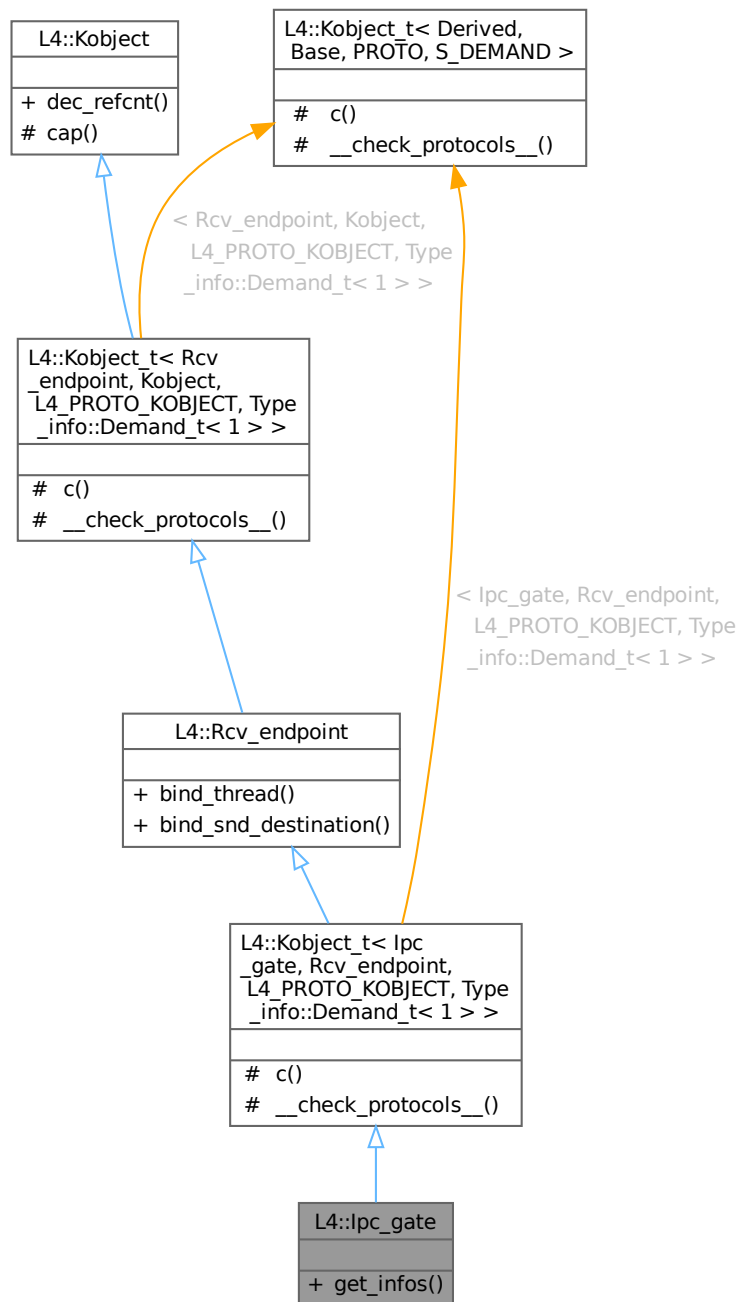
- l4/sys/cxx/ipc\_varg

## 16.158 L4::ipc\_gate Class Reference

The C++ IPC gate interface, see [IPC-Gate API](#) for the C interface.

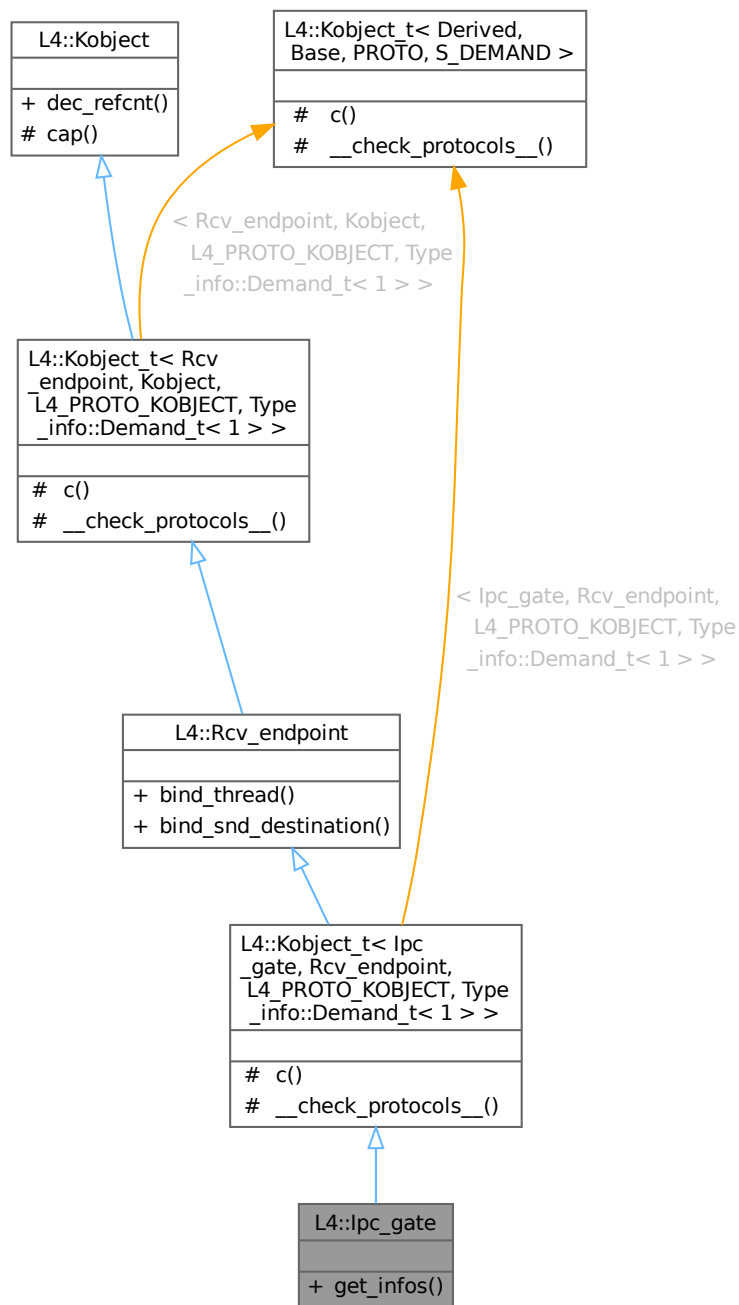
```
#include <ipc_gate>
```

Inheritance diagram for L4::lpc\_gate:





Collaboration diagram for L4::lpc\_gate:



### Public Member Functions

- `l4_msgtag_t get_infos (l4_umword_t *label)`  
Get information about the IPC-gate.

### Public Member Functions inherited from L4::Rcv\_endpoint

- `l4_msgtag_t bind_thread (lpc::Cap< Thread > t, l4_umword_t label)`

*Bind the IPC receive endpoint to a thread.*

- `l4_msgtag_t bind_snd_destination (Cap< Snd_destination > snd_dst, l4_umword_t label)`

*Bind a send destination (a thread or thread group) to an IPC receive endpoint.*

## Public Member Functions inherited from `L4::Kobject`

- `l4_msgtag_t dec_refcnt (l4_mword_t diff, l4_utcb_t *utcb=l4_utcb())`

*Decrement the in kernel reference counter for the object.*

## Additional Inherited Members

## Protected Types inherited from

`L4::Kobject_t< lpc_gate, Rcv_endpoint, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > >`

- typedef `lpc_gate` **Class**

*The target interface type (inheriting from `Kobject_t`).*

- typedef `Typeid::Iface< PROTO, lpc_gate > __Iface`

*The interface description for the derived class.*

- typedef `Typeid::Merge_list< Typeid::Iface_list< __Iface >, typename Rcv_endpoint::__Iface_list > __Iface_list`

*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Types inherited from

`L4::Kobject_t< Rcv_endpoint, Kobject, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > >`

- typedef `Rcv_endpoint` **Class**

*The target interface type (inheriting from `Kobject_t`).*

- typedef `Typeid::Iface< PROTO, Rcv_endpoint > __Iface`

*The interface description for the derived class.*

- typedef `Typeid::Merge_list< Typeid::Iface_list< __Iface >, typename Kobject::__Iface_list > __Iface_list`

*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Member Functions inherited from

`L4::Kobject_t< lpc_gate, Rcv_endpoint, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > >`

- `L4::Cap< Class > c () const noexcept`

*Get the capability to ourselves.*

## Protected Member Functions inherited from

`L4::Kobject_t< Rcv_endpoint, Kobject, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > >`

- `L4::Cap< Class > c () const noexcept`

*Get the capability to ourselves.*

## Protected Member Functions inherited from L4::Kobject

- [l4\\_cap\\_idx\\_t cap](#) () const noexcept  
*Return capability selector.*

## Static Protected Member Functions inherited from

[L4::Kobject\\_t< lpc\\_gate, Rcv\\_endpoint, L4\\_PROTO\\_KOBJECT, Type\\_info::Demand\\_t< 1 > >](#)

- static void [\\_\\_check\\_protocols\\_\\_](#) () noexcept  
*Helper to check for protocol conflicts.*

## Static Protected Member Functions inherited from

[L4::Kobject\\_t< Rcv\\_endpoint, Kobject, L4\\_PROTO\\_KOBJECT, Type\\_info::Demand\\_t< 1 > >](#)

- static void [\\_\\_check\\_protocols\\_\\_](#) () noexcept  
*Helper to check for protocol conflicts.*

### 16.158.1 Detailed Description

The C++ IPC gate interface, see [IPC-Gate API](#) for the C interface.

IPC gates are used to create secure communication channels between protection domains. An IPC gate can be created using the [L4::Factory](#) interface.

Depending on the permissions of the capability used, an IPC gate forwards IPC to the [L4::Thread](#) or [L4::Thread\\_group](#) the IPC gate is *bound* to (cf. [bind\\_thread\(\)](#) and [bind\\_snd\\_destination\(\)](#)). If the capability has the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission, only IPC using a protocol different from the [L4\\_PROTO\\_KOBJECT](#) protocol is forwarded. Without the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission, all IPC is forwarded. The latter is the usual case for a client in a client/server scenario. When not bound to a thread or thread group yet, the forwarded IPC blocks until the IPC gate is bound to a thread or thread group, or the IPC times out.

Forwarded IPC is always forwarded to the userland of the thread the IPC gate is bound to, either directly or indirectly using a thread group. That means, the [L4::Thread](#) interface of that thread is not accessible via an IPC gate. The [L4::lpc\\_gate](#) interface of an IPC gate is only accessible if the capability used has the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission (cf. previous paragraph). Conversely that means, if the capability used lacks the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission, [L4::lpc\\_gate](#) interface calls are forwarded to the thread or thread group the IPC gate is bound to instead of being processed by the IPC gate itself. In a client/server scenario, a client should only get IPC gate capabilities without [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission so the client cannot tamper with the IPC gate.

When binding an IPC gate to a thread or thread group, a user-defined, kernel protected, machine-word sized payload called the IPC gate's *label* is assigned to the IPC gate (note that the two least significant bits of the label must be zero; cf. [bind\\_thread\(\)](#) and [bind\\_snd\\_destination\(\)](#)). When a send-only IPC or call IPC is forwarded via an IPC gate, the label provided by the sender is ignored and replaced by the IPC gate's label where the two least significant bits are set to the [L4\\_CAP\\_FPAGE\\_S](#) and [L4\\_CAP\\_FPAGE\\_W](#) permissions of the capability used. The replaced label is only visible to the thread the IPC gate is bound to (or to the selected thread of the thread group the IPC gate is bound to) upon receive. However, the configured label of an IPC gate can also be queried via [get\\_infos\(\)](#) if the capability used has the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission.

When deleting an IPC gate or when unbinding it from a thread or thread group, the label of IPC already in flight won't be changed. To ensure that no IPC from this IPC gate is received by a thread with an unexpected label, [L4::Thread::modify\\_senders\(\)](#) shall be used to change the labels of every pending IPC to that gate. This is also required if the label of an already bound IPC gate is changed. It is not necessary after binding the IPC gate to a thread or thread group for the first time.

When binding a currently bound IPC gate to a new thread or thread group, the same label should be used that was used with the old thread. Otherwise the old and the new thread need to synchronize to avoid IPC messages with unexpected labels.

**Include File**

```
#include <l4/sys/ipc_gate>
```

For the C interface refer to the C [IPC-Gate API](#).

**See also**

[Object Invocation](#)

Definition at line 85 of file [ipc\\_gate](#).

**16.158.2 Member Function Documentation****16.158.2.1 get\_infos()**

```
l4_msgtag_t L4::Ipc_gate::get_infos (
    l4_umword_t * label)
```

Get information about the IPC-gate.

**Parameters**

out	<i>label</i>	The label of the IPC gate is returned here.
-----	--------------	---------------------------------------------

**Returns**

System call return tag.

**Precondition**

If the IPC gate capability used to invoke this operation does not possess the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) right, the kernel will not perform the operation. Instead, the underlying IPC message will be forwarded to the thread or thread group the IPC gate is bound to, blocking the caller if the IPC gate is not bound yet.

References [get\\_infos\(\)](#).

Referenced by [get\\_infos\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- [l4/sys/ipc\\_gate](#)

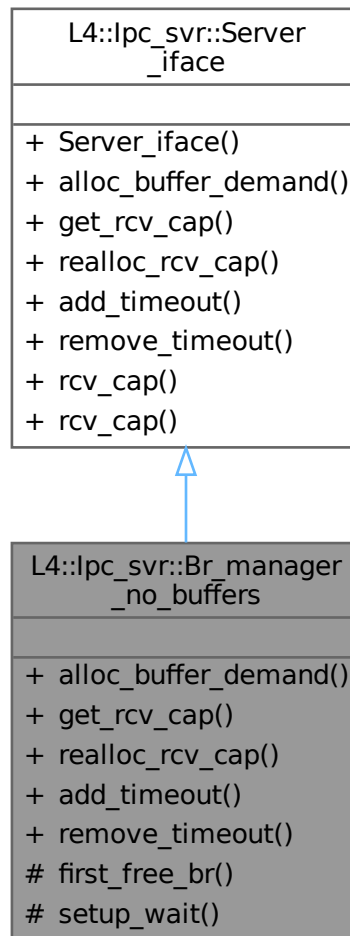
## 16.159 L4::lpc\_svr::Br\_manager\_no\_buffers Class Reference

Empty implementation of [Server\\_iface](#).

```
#include <ipc_server_loop>
```



Collaboration diagram for L4::lpc\_svr::Br\_manager\_no\_buffers:



## Public Member Functions

- int `alloc_buffer_demand` (`Demand` const &demand) override  
*Tells the server to allocate buffers for the given demand.*
- L4::Cap< void > `get_rcv_cap` (int) const override  
*Returns L4::Cap<void>::Invalid, we have no buffer management.*
- int `realloc_rcv_cap` (int) override  
*Returns -L4\_ENOMEM, we have no buffer management.*
- int `add_timeout` (`Timeout` \*, `l4_kernel_clock_t`) override  
*Returns -L4\_ENOSYS, we have no timeout queue.*
- int `remove_timeout` (`Timeout` \*) override  
*Returns -L4\_ENOSYS, we have no timeout queue.*

## Public Member Functions inherited from [L4::lpc\\_svr::Server\\_iface](#)

- **Server\_iface** ()  
*Make a server interface.*
- `template<typename T>`  
[L4::Cap](#)< T > [rcv\\_cap](#) (int index) const  
*Get given receive buffer as typed capability.*
- [L4::Cap](#)< void > [rcv\\_cap](#) (int index) const  
*Get receive cap with the given index as generic (void) type.*

## Protected Member Functions

- unsigned **first\_free\_br** () const  
*Returns 1 as first free buffer.*
- void **setup\_wait** ([l4\\_utcb\\_t](#) \*utcb, [L4::lpc\\_svr::Reply\\_mode](#))  
*Setup wait function for the server loop ([Server](#)<>).*

## Additional Inherited Members

## Public Types inherited from [L4::lpc\\_svr::Server\\_iface](#)

- typedef [L4::Type\\_info::Demand](#) **Demand**  
*Data type expressing server-side demand for receive buffers.*

### 16.159.1 Detailed Description

Empty implementation of [Server\\_iface](#).

This implementation of [Server\\_iface](#) provides no buffer or timeout management at all it just returns errors for all calls that express other than empty demands. However, this may be useful for very simple servers that serve simple server objects only.

Definition at line 233 of file [ipc\\_server\\_loop](#).

### 16.159.2 Member Function Documentation

#### 16.159.2.1 [alloc\\_buffer\\_demand\(\)](#)

```
int L4::Ipc_svr::Br_manager_no_buffers::alloc_buffer_demand (
    Demand const & demand) [inline], [override], [virtual]
```

Tells the server to allocate buffers for the given demand.

#### Parameters

<i>demand</i>	The total server-side demand of receive buffers needed for a given interface, see <a href="#">Demand</a> .
---------------	------------------------------------------------------------------------------------------------------------

This function is not called by user applications directly. Usually the server implementation or the registry implementation calls this function whenever a new object is registered at the server.



**Returns**

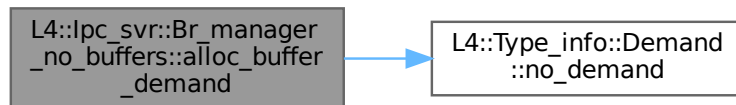
success (0) if demand is empty, -L4\_ENOMEM else.

Implements [L4::lpc\\_svr::Server\\_iface](#).

Definition at line 240 of file [ipc\\_server\\_loop](#).

References [L4\\_ENOMEM](#), [L4\\_EOK](#), and [L4::Type\\_info::Demand::no\\_demand\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

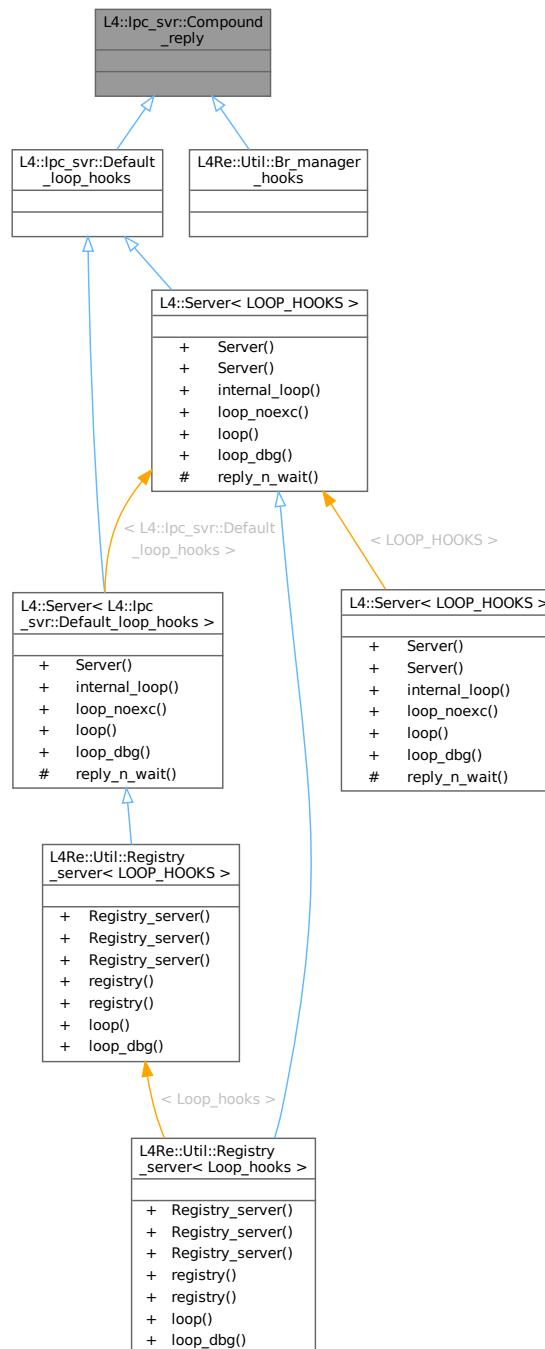
- `l4/sys/cxx/ipc_server_loop`

## 16.160 L4::lpc\_svr::Compound\_reply Struct Reference

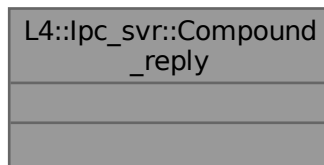
Mix in for LOOP\_HOOKS to always use compound reply and wait.

```
#include <ipc_server_loop>
```

Inheritance diagram for L4::ipc\_svr::Compound\_reply:



Collaboration diagram for L4::lpc\_svr::Compound\_reply:



### 16.160.1 Detailed Description

Mix in for LOOP\_HOOKS to always use compound reply and wait.

Definition at line 73 of file [ipc\\_server\\_loop](#).

The documentation for this struct was generated from the following file:

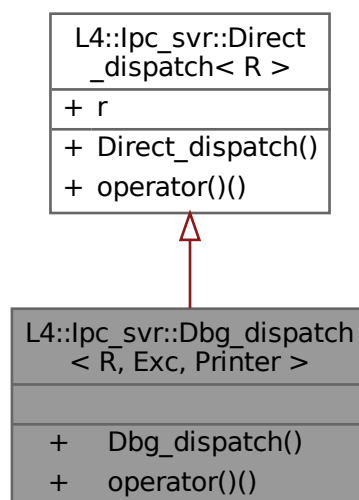
- l4/sys/cxx/ipc\_server\_loop

## 16.161 L4::lpc\_svr::Dbg\_dispatch< R, Exc, Printer > Struct Template Reference

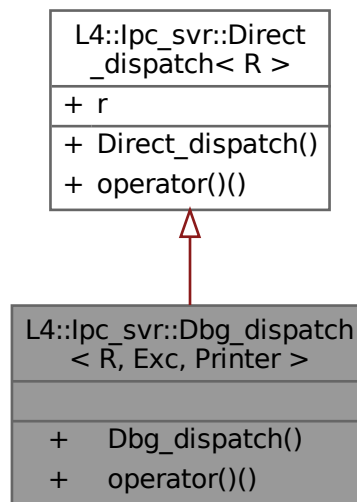
Dispatch helper that, in addition to what [Exc\\_dispatch](#) does, prints exception messages.

```
#include <ipc_server_loop>
```

Inheritance diagram for L4::lpc\_svr::Dbg\_dispatch< R, Exc, Printer >:



Collaboration diagram for `L4::lpc_svr::Dbg_dispatch< R, Exc, Printer >`:



### Public Member Functions

- **Dbg\_dispatch** (`R r`, `Printer p`)  
*Make an exception handling dispatcher.*
- `l4_msgtag_t` **operator()** (`l4_msgtag_t tag`, `l4_umword_t obj`, `l4_utcb_t *utcb`)  
*Dispatch the call via [Direct\\_dispatch<R>\(\)](#), handle exceptions, and print the exception message.*

### 16.161.1 Detailed Description

```
template<typename R, typename Exc, typename Printer>
struct L4::lpc_svr::Dbg_dispatch< R, Exc, Printer >
```

Dispatch helper that, in addition to what [Exc\\_dispatch](#) does, prints exception messages.

### Template Parameters

<i>R</i>	Data type of the registry used for dispatching to objects.
<i>Exc</i>	Data type of the exceptions that shall be caught. This data type must provide a member <code>err_no()</code> that returns the negative integer (int) error code for the exception. In addition, methods <code>str()</code> and <code>extra_str()</code> are required that return a c-string describing the error.
<i>Printer</i>	A type that provides a <code>printf()</code> member that is used (with the usual format string syntax) to print error messages.

This dispatcher wraps `Direct_dispatch<R>` with a try-catch (`Exc`).

Definition at line 184 of file [ipc\\_server\\_loop](#).

The documentation for this struct was generated from the following file:

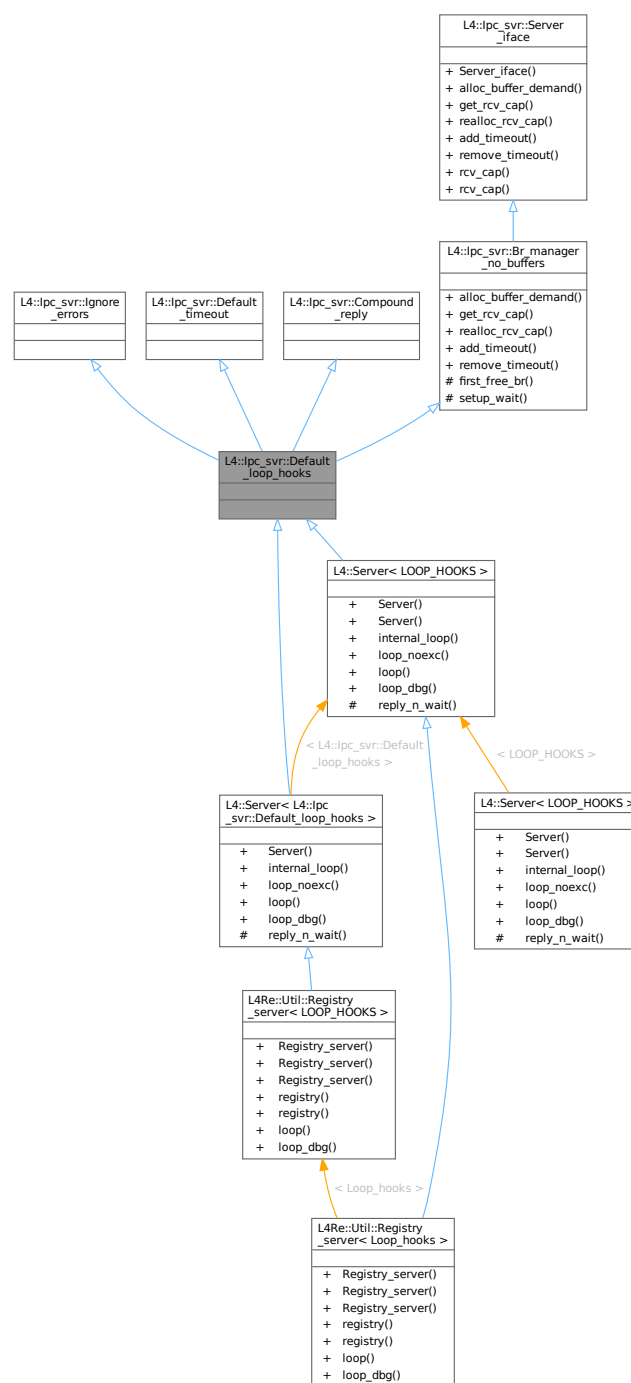
- `l4/sys/cxx/ipc_server_loop`

## 16.162 L4::lpc\_svr::Default\_loop\_hooks Struct Reference

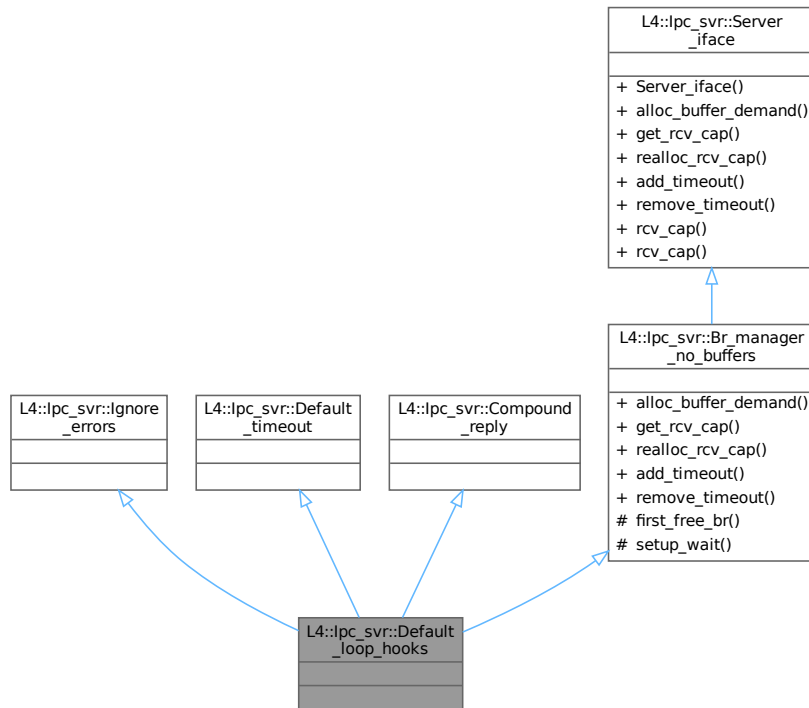
Default LOOP\_HOOKS.

```
#include <ipc_server_loop>
```

Inheritance diagram for L4::lpc\_svr::Default\_loop\_hooks:



Collaboration diagram for L4::lpc\_svr::Default\_loop\_hooks:



### Additional Inherited Members

### Public Types inherited from L4::lpc\_svr::Server\_iface

- typedef L4::Type\_info::Demand Demand  
*Data type expressing server-side demand for receive buffers.*

### Public Member Functions inherited from L4::lpc\_svr::Br\_manager\_no\_buffers

- int **alloc\_buffer\_demand** (Demand const &demand) override  
*Tells the server to allocate buffers for the given demand.*
- L4::Cap< void > **get\_rcv\_cap** (int) const override  
*Returns L4::Cap< void >::Invalid, we have no buffer management.*
- int **realloc\_rcv\_cap** (int) override  
*Returns -L4\_ENOMEM, we have no buffer management.*
- int **add\_timeout** (Timeout \*, l4\_kernel\_clock\_t) override  
*Returns -L4\_ENOSYS, we have no timeout queue.*
- int **remove\_timeout** (Timeout \*) override  
*Returns -L4\_ENOSYS, we have no timeout queue.*

## Public Member Functions inherited from L4::lpc\_svr::Server\_iface

- **Server\_iface** ()  
*Make a server interface.*
- template<typename T>  
L4::Cap< T > **rcv\_cap** (int index) const  
*Get given receive buffer as typed capability.*
- L4::Cap< void > **rcv\_cap** (int index) const  
*Get receive cap with the given index as generic (void) type.*

## Protected Member Functions inherited from L4::lpc\_svr::Br\_manager\_no\_buffers

- unsigned **first\_free\_br** () const  
*Returns 1 as first free buffer.*
- void **setup\_wait** (l4\_utcb\_t \*utcb, L4::lpc\_svr::Reply\_mode)  
*Setup wait function for the server loop (Server<>).*

### 16.162.1 Detailed Description

Default LOOP\_HOOKS.

Combination of [Ignore\\_errors](#), [Default\\_timeout](#), [Compound\\_reply](#), and [Br\\_manager\\_no\\_buffers](#).

Definition at line 285 of file [ipc\\_server\\_loop](#).

The documentation for this struct was generated from the following file:

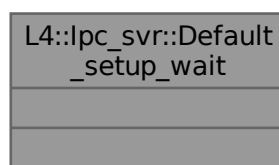
- l4/sys/cxx/ipc\_server\_loop

## 16.163 L4::lpc\_svr::Default\_setup\_wait Struct Reference

Mix in for LOOP\_HOOKS for setup\_wait no op.

```
#include <ipc_server_loop>
```

Collaboration diagram for L4::lpc\_svr::Default\_setup\_wait:



### 16.163.1 Detailed Description

Mix in for LOOP\_HOOKS for setup\_wait no op.

Definition at line 84 of file [ipc\\_server\\_loop](#).

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc\_server\_loop

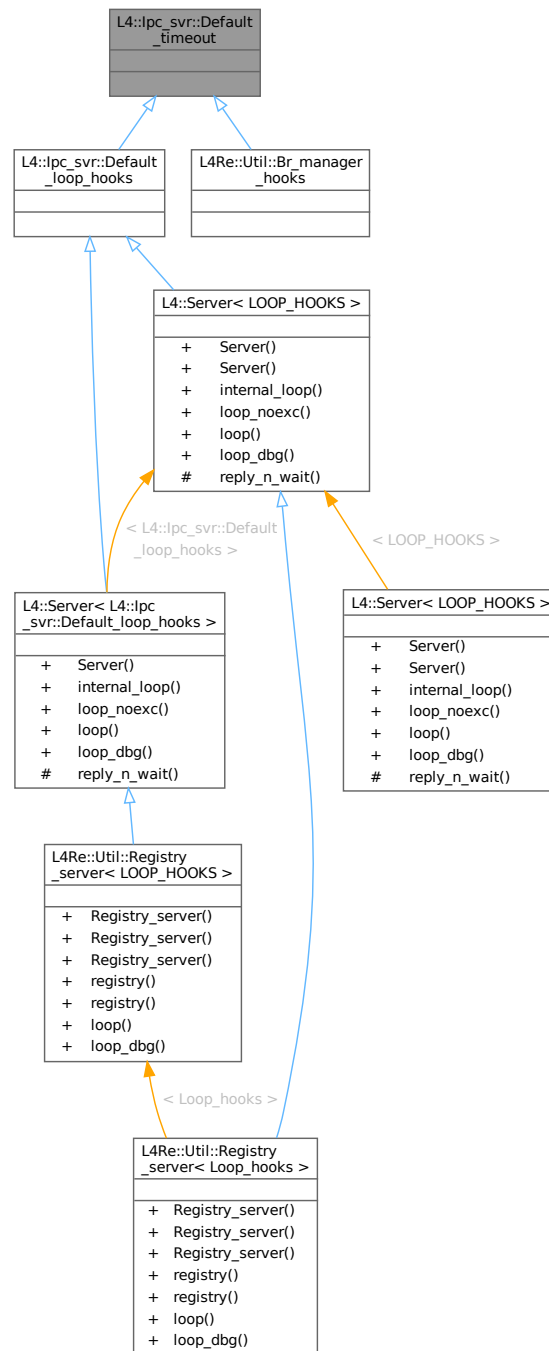
## 16.164 L4::lpc\_svr::Default\_timeout Struct Reference

Mix in for LOOP\_HOOKS to use a 0 send and a infinite receive timeout.

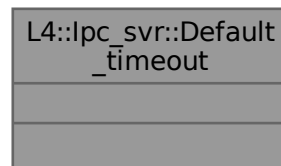
```
#include <ipc_server_loop>
```



Inheritance diagram for L4::lpc\_svr::Default\_timeout:



Collaboration diagram for L4::lpc\_svr::Default\_timeout:



### 16.164.1 Detailed Description

Mix in for LOOP\_HOOKS to use a 0 send and a infinite receive timeout.

Definition at line 65 of file [ipc\\_server\\_loop](#).

The documentation for this struct was generated from the following file:

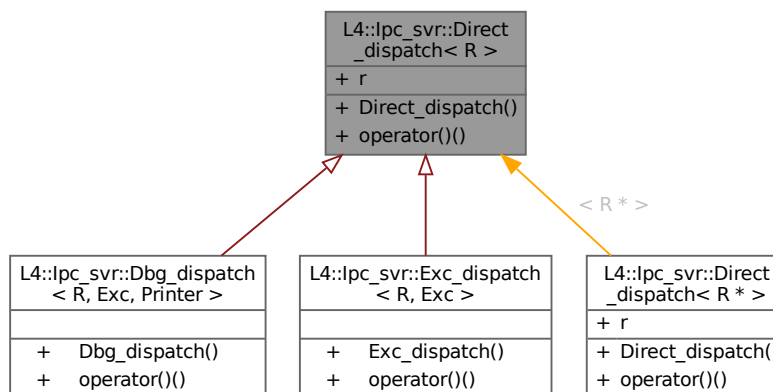
- l4/sys/cxx/ipc\_server\_loop

## 16.165 L4::lpc\_svr::Direct\_dispatch< R > Struct Template Reference

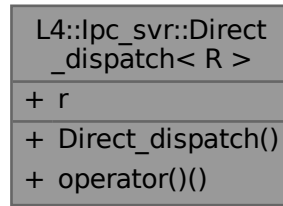
Direct dispatch helper, for forwarding dispatch calls to a registry *R*.

```
#include <ipc_server_loop>
```

Inheritance diagram for L4::lpc\_svr::Direct\_dispatch< R >:



Collaboration diagram for L4::lpc\_svr::Direct\_dispatch< R >:



### Public Member Functions

- **Direct\_dispatch** (R &r)  
*Make a direct dispatcher.*
- **l4\_msgtag\_t operator()** (l4\_msgtag\_t tag, l4\_umword\_t obj, l4\_utcb\_t \*utcb)  
*call operator forwarding to r.dispatch()*

### Data Fields

- **R & r**  
*stores a reference to the registry object*

## 16.165.1 Detailed Description

```
template<typename R>
struct L4::lpc_svr::Direct_dispatch< R >
```

Direct dispatch helper, for forwarding dispatch calls to a registry *R*.

### Template Parameters

<i>R</i>	Data type of the registry that is used for dispatching to different server objects, usually based on the protected IPC label.
----------	-------------------------------------------------------------------------------------------------------------------------------

Definition at line 95 of file [ipc\\_server\\_loop](#).

The documentation for this struct was generated from the following file:

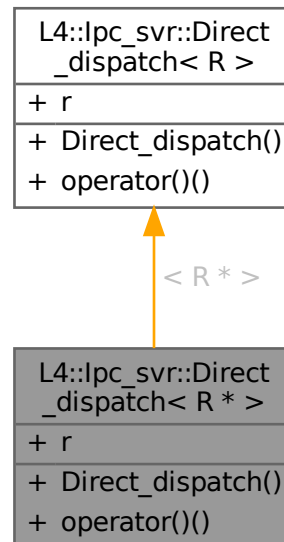
- l4/sys/cxx/ipc\_server\_loop

## 16.166 L4::lpc\_svr::Direct\_dispatch< R \* > Struct Template Reference

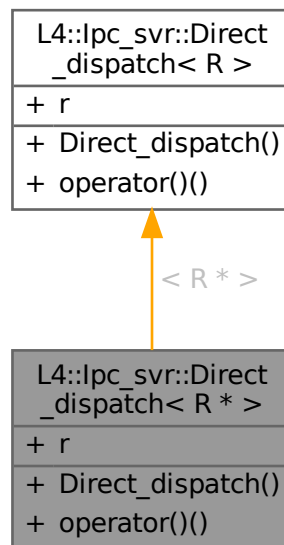
Direct dispatch helper, for forwarding dispatch calls via a pointer to a registry *R*.

```
#include <ipc_server_loop>
```

Inheritance diagram for L4::lpc\_svr::Direct\_dispatch< R \* >:



Collaboration diagram for L4::lpc\_svr::Direct\_dispatch< R \* >:



### Public Member Functions

- **Direct\_dispatch** (R \*r)  
*Make a direct dispatcher.*
- **l4\_msgtag\_t operator()** (l4\_msgtag\_t tag, l4\_umword\_t obj, l4\_utcb\_t \*utcb)  
*call operator forwarding to r->dispatch()*

### Data Fields

- **R \* r**  
*stores a pointer to the registry object*

## 16.166.1 Detailed Description

```
template<typename R>
struct L4::lpc_svr::Direct_dispatch< R * >
```

Direct dispatch helper, for forwarding dispatch calls via a pointer to a registry *R*.

### Template Parameters

<i>R</i>	Data type of the registry that is used for dispatching to different server objects, usually based on the protected IPC label.
----------	-------------------------------------------------------------------------------------------------------------------------------

Definition at line 116 of file [ipc\\_server\\_loop](#).

The documentation for this struct was generated from the following file:

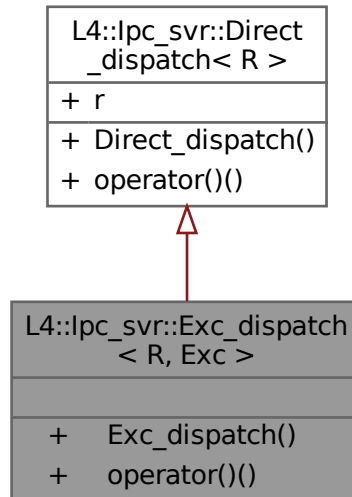
- l4/sys/cxx/ipc\_server\_loop

## 16.167 L4::lpc\_svr::Exc\_dispatch< R, Exc > Struct Template Reference

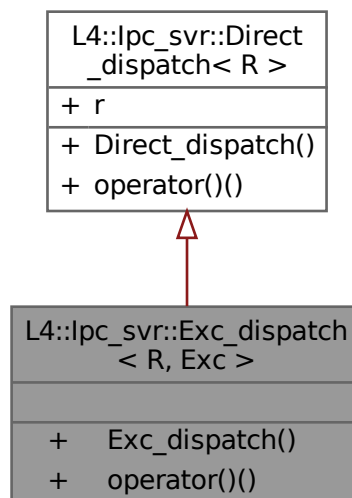
Dispatch helper wrapping try {} catch {} around the dispatch call.

```
#include <ipc_server_loop>
```

Inheritance diagram for L4::lpc\_svr::Exc\_dispatch< R, Exc >:



Collaboration diagram for L4::lpc\_svr::Exc\_dispatch< R, Exc >:



## Public Member Functions

- **Exc\_dispatch** (R r)  
*Make an exception handling dispatcher.*
- **l4\_msgtag\_t operator()** (l4\_msgtag\_t tag, l4\_umword\_t obj, l4\_utcb\_t \*utcb)  
*Dispatch the call via [Direct\\_dispatch<R>\(\)](#) and handle exceptions.*

### 16.167.1 Detailed Description

```
template<typename R, typename Exc>
struct L4::lpc_svr::Exc_dispatch< R, Exc >
```

Dispatch helper wrapping try {} catch {} around the dispatch call.

#### Template Parameters

<i>R</i>	Data type of the registry used for dispatching to objects.
<i>Exc</i>	Data type of the exceptions that shall be caught. This data type must provide a member <code>err_no()</code> that returns the negative integer (int) error code for the exception.

This dispatcher wraps `Direct_dispatch<R>` with a try-catch (Exc).

Definition at line 140 of file [ipc\\_server\\_loop](#).

The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc\_server\_loop

## 16.168 L4::lpc\_svr::Ignore\_errors Struct Reference

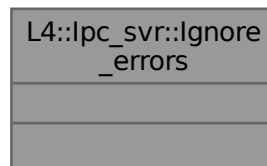
Mix in for LOOP\_HOOKS to ignore IPC errors.

```
#include <ipc_server_loop>
```





Collaboration diagram for L4::lpc\_svr::Ignore\_errors:



### 16.168.1 Detailed Description

Mix in for LOOP\_HOOKS to ignore IPC errors.

Definition at line 57 of file [ipc\\_server\\_loop](#).

The documentation for this struct was generated from the following file:

- I4/sys/cxx/ipc\_server\_loop

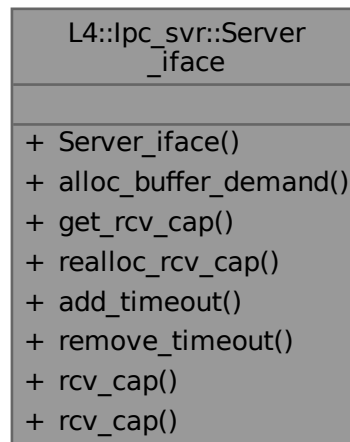
## 16.169 L4::lpc\_svr::Server\_iface Class Reference

Interface for server-loop related functions.

```
#include <ipc_epiface>
```



Collaboration diagram for L4::lpc\_svr::Server\_iface:



## Public Types

- typedef [L4::Type\\_info::Demand](#) **Demand**  
*Data type expressing server-side demand for receive buffers.*

## Public Member Functions

- **Server\_iface ()**  
*Make a server interface.*
- virtual int [alloc\\_buffer\\_demand](#) ([Demand](#) const &demand)=0  
*Tells the server to allocate buffers for the given demand.*
- virtual [L4::Cap](#)< void > [get\\_rcv\\_cap](#) (int index) const =0  
*Get capability slot allocated to the given receive buffer.*
- virtual int [realloc\\_rcv\\_cap](#) (int index)=0  
*Allocate a new capability for the given receive buffer.*
- virtual int [add\\_timeout](#) ([Timeout](#) \*timeout, [l4\\_kernel\\_clock\\_t](#) time)=0  
*Add a timeout to the server internal timeout queue.*
- virtual int [remove\\_timeout](#) ([Timeout](#) \*timeout)=0  
*Remove the given timeout from the timer queue.*
- template<typename T>  
[L4::Cap](#)< T > [rcv\\_cap](#) (int index) const  
*Get given receive buffer as typed capability.*
- [L4::Cap](#)< void > [rcv\\_cap](#) (int index) const  
*Get receive cap with the given index as generic (void) type.*

### 16.169.1 Detailed Description

Interface for server-loop related functions.

This interface provides access to high-level server-loop related functions, such as management of receive buffers and timeouts.

Definition at line 36 of file [ipc\\_epiface](#).

### 16.169.2 Member Function Documentation

#### 16.169.2.1 add\_timeout()

```
virtual int L4::Ipc_svr::Server_iface::add_timeout (
    Timeout * timeout,
    l4_kernel_clock_t time) [pure virtual]
```

Add a timeout to the server internal timeout queue.

##### Parameters

<i>timeout</i>	The timeout object to register.
<i>time</i>	The time (absolute) at which the timeout shall expire.

##### Precondition

timeout must not be in any queue.

##### Returns

0 on success, 1 if timeout is already expired, < 0 on error.

Implemented in [L4::Ipc\\_svr::Br\\_manager\\_no\\_buffers](#), [L4::Ipc\\_svr::Timeout\\_queue\\_hooks< HOOKS, BR\\_MAN >](#), [L4::Ipc\\_svr::Timeout\\_queue\\_hooks< Br\\_manager\\_timeout\\_hooks, Br\\_manager >](#), [L4::Ipc\\_svr::Timeout\\_queue\\_hooks< Loop\\_hook\\_timeout\\_hooks, Loop\\_hook\\_manager >](#) and [L4Re::Util::Br\\_manager](#).

#### 16.169.2.2 alloc\_buffer\_demand()

```
virtual int L4::Ipc_svr::Server_iface::alloc_buffer_demand (
    Demand const & demand) [pure virtual]
```

Tells the server to allocate buffers for the given demand.

##### Parameters

<i>demand</i>	The total server-side demand of receive buffers needed for a given interface, see <a href="#">Demand</a> .
---------------	------------------------------------------------------------------------------------------------------------

This function is not called by user applications directly. Usually the server implementation or the registry implementation calls this function whenever a new object is registered at the server.

Implemented in [L4::Ipc\\_svr::Br\\_manager\\_no\\_buffers](#), and [L4Re::Util::Br\\_manager](#).

### 16.169.2.3 get\_rcv\_cap()

```
virtual L4::Cap< void > L4::Ipc_svr::Server_iface::get_rcv_cap (
    int index) const [pure virtual]
```

Get capability slot allocated to the given receive buffer.

#### Parameters

---

<i>index</i>	The receive buffer index of the expected capability argument ( $0 \leq \text{index} < \text{caps}$ registered with <a href="#">alloc_buffer_demand()</a> ).
--------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------

**Precondition**

$0 \leq \text{index} < \text{caps}$  registered with [alloc\\_buffer\\_demand\(\)](#)

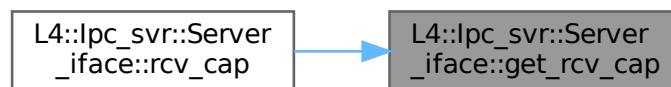
**Returns**

Capability slot currently allocated to the given receive buffer.

Implemented in [L4::lpc\\_svr::Br\\_manager\\_no\\_buffers](#), and [L4Re::Util::Br\\_manager](#).

Referenced by [rcv\\_cap\(\)](#).

Here is the caller graph for this function:

**16.169.2.4 rcv\_cap() [1/2]**

```

L4::Cap< void > L4::lpc_svr::Server_iface::rcv_cap (
    int index) const [inline]
  
```

Get receive cap with the given index as generic (void) type.

**Parameters**

<i>index</i>	The index of the cap receive buffer of the expected capability. ( $0 \leq \text{index} < \text{caps}$ registered with <a href="#">alloc_buffer_demand()</a> .)
--------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------

**Returns**

Capability slot currently allocated to the given capability buffer.

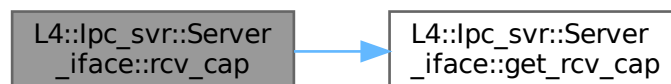
**Note**

This is a convenience wrapper for [get\\_rcv\\_cap\(\)](#).

Definition at line 126 of file [ipc\\_epiface](#).

References [get\\_rcv\\_cap\(\)](#).

Here is the call graph for this function:

**16.169.2.5 rcv\_cap()** [2/2]

```
template<typename T>
L4::Cap< T > L4::lpc_svr::Server_iface::rcv_cap (
    int index) const [inline]
```

Get given receive buffer as typed capability.

**See also**

[get\\_rcv\\_cap\(\)](#)

**Parameters**

<i>index</i>	The receive buffer index of the expected capability argument. ( $0 \leq \text{index} < \text{caps}$ registered with <a href="#">alloc_buffer_demand()</a> .)
--------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------

**Precondition**

$0 \leq \text{index} < \text{caps}$  registered with [alloc\\_buffer\\_demand\(\)](#)

**Returns**

Capability slot currently allocated to the given receive buffer.

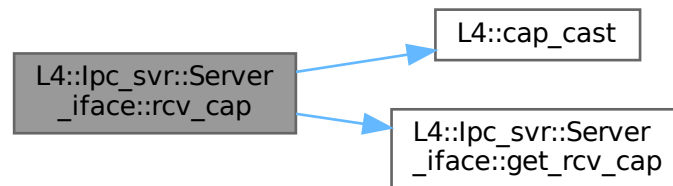
**Note**

This is a convenience wrapper for [get\\_rcv\\_cap\(\)](#) to avoid [L4::cap\\_cast<>\(\)](#).

Definition at line 114 of file [ipc\\_epiface](#).

References [L4::cap\\_cast\(\)](#), and [get\\_rcv\\_cap\(\)](#).

Here is the call graph for this function:

**16.169.2.6 realloc\_rcv\_cap()**

```
virtual int L4::Ipc_svr::Server_iface::realloc_rcv_cap (
    int index) [pure virtual]
```

Allocate a new capability for the given receive buffer.

**Parameters**

<i>index</i>	The receive buffer index of the expected capability argument ( $0 \leq \text{index} < \text{caps}$ registered with <a href="#">alloc_buffer_demand()</a> ).
--------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------

**Precondition**

$0 \leq \text{index} < \text{caps}$  registered with [alloc\\_buffer\\_demand\(\)](#)

**Returns**

0 on success, < 0 on error.

Implemented in [L4::lpc\\_svr::Br\\_manager\\_no\\_buffers](#), and [L4Re::Util::Br\\_manager](#).

**16.169.2.7 remove\_timeout()**

```
virtual int L4::Ipc_svr::Server_iface::remove_timeout (
    Timeout * timeout) [pure virtual]
```

Remove the given timeout from the timer queue.

**Parameters**



<i>timeout</i>	The timeout object to remove.
----------------	-------------------------------

#### Returns

0 on success, < 0 on error.

Implemented in [L4::lpc\\_svr::Br\\_manager\\_no\\_buffers](#), [L4::lpc\\_svr::Timeout\\_queue\\_hooks< HOOKS, BR\\_MAN >](#), [L4::lpc\\_svr::Timeout\\_queue\\_hooks< Br\\_manager\\_timeout\\_hooks, Br\\_manager >](#), [L4::lpc\\_svr::Timeout\\_queue\\_hooks< Loop\\_hooks, Loop\\_manager >](#) and [L4Re::Util::Br\\_manager](#).

The documentation for this class was generated from the following file:

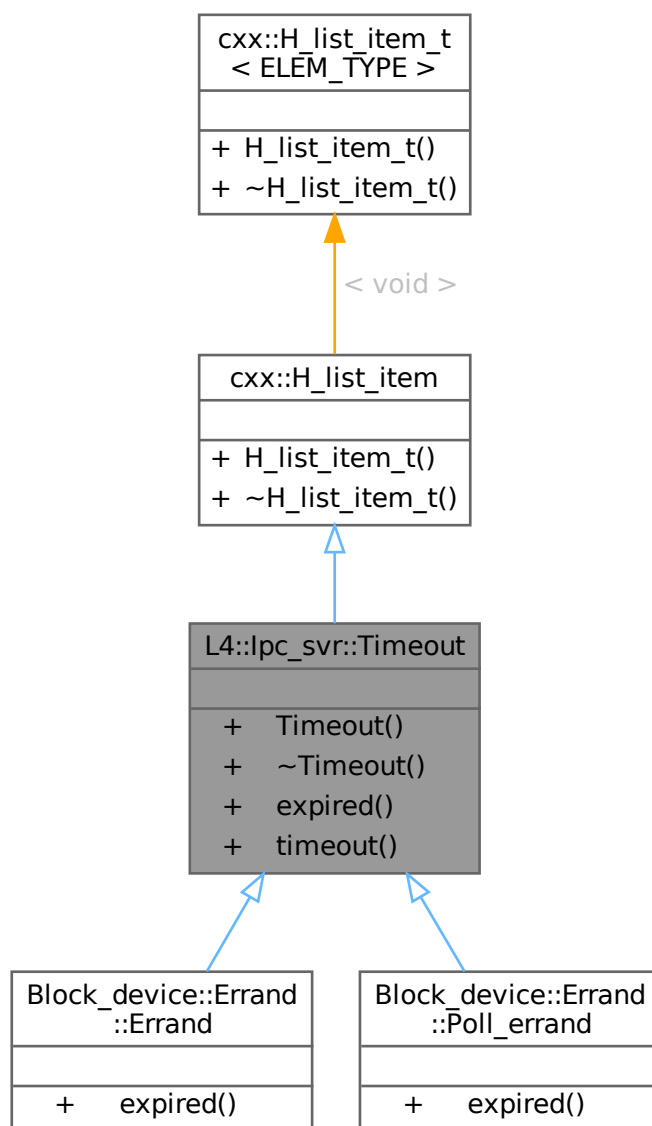
- [l4/sys/cxx/ipc\\_epiface](#)

## 16.170 L4::lpc\_svr::Timeout Class Reference

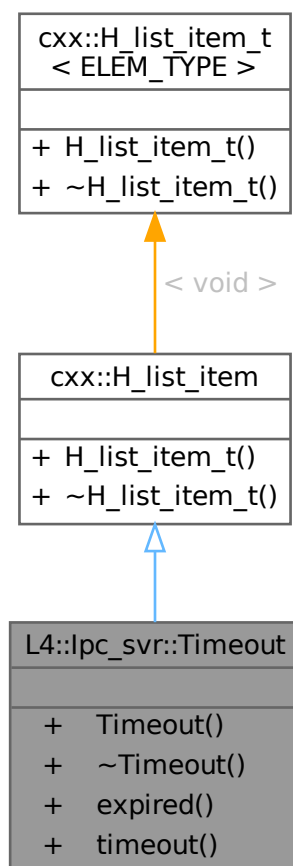
Callback interface for [Timeout\\_queue](#).

```
#include <ipc_timeout_queue>
```

Inheritance diagram for L4::lpc\_svr::Timeout:



Collaboration diagram for L4::lpc\_svr::Timeout:



## Public Member Functions

- **Timeout ()**  
*Make a timeout.*
- virtual **~Timeout ()=0**  
*Destroy a timeout.*
- virtual void **expired ()=0**  
*callback function to be called when timeout happened*
- **l4\_kernel\_clock\_t timeout () const**  
*return absolute timeout of this callback.*

## Public Member Functions inherited from **cxx::H\_list\_item\_t< void >**

- **H\_list\_item\_t ()**  
*Constructor.*
- **~H\_list\_item\_t () noexcept**  
*Destructor.*

### 16.170.1 Detailed Description

Callback interface for [Timeout\\_queue](#).

Definition at line 18 of file [ipc\\_timeout\\_queue](#).

### 16.170.2 Member Function Documentation

#### 16.170.2.1 expired()

```
virtual void L4::Ipc_svr::Timeout::expired () [pure virtual]
```

callback function to be called when timeout happened

##### Note

The timeout object is already dequeued when this function is called, this means the timeout may be safely queued again within the [expired\(\)](#) function.

Implemented in [Block\\_device::Errand::Errand](#), and [Block\\_device::Errand::Poll\\_errand](#).

Referenced by [L4::ipc\\_svr::Timeout\\_queue::handle\\_expired\\_timeouts\(\)](#).

Here is the caller graph for this function:



#### 16.170.2.2 timeout()

```
l4_kernel_clock_t L4::Ipc_svr::Timeout::timeout () const [inline]
```

return absolute timeout of this callback.

##### Returns

absolute timeout for this instance of the timeout.

##### Precondition

The timeout object must have been in a queue before, otherwise the timeout is not set.

Definition at line 42 of file [ipc\\_timeout\\_queue](#).

The documentation for this class was generated from the following file:

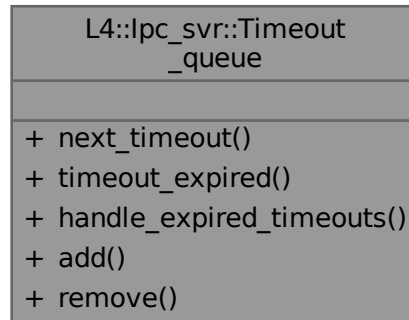
- [l4/cxx/ipc\\_timeout\\_queue](#)

## 16.171 L4::lpc\_svr::Timeout\_queue Class Reference

[Timeout](#) queue to be used in l4re server loop.

```
#include <ipc_timeout_queue>
```

Collaboration diagram for L4::lpc\_svr::Timeout\_queue:



### Public Types

- typedef [L4::lpc\\_svr::Timeout](#) **Timeout**  
Provide a local definition of [Timeout](#) for backward compatibility.

### Public Member Functions

- [l4\\_kernel\\_clock\\_t](#) **next\_timeout** () const  
Get the time for the next timeout.
- bool **timeout\_expired** ([l4\\_kernel\\_clock\\_t](#) now) const  
Determine if a timeout has happened.
- void **handle\_expired\_timeouts** ([l4\\_kernel\\_clock\\_t](#) now)  
run the callbacks of expired timeouts
- void **add** ([Timeout](#) \*timeout, [l4\\_kernel\\_clock\\_t](#) time)  
Add a timeout to the queue.
- void **remove** ([Timeout](#) \*timeout)  
Remove timeout from the queue.

### 16.171.1 Detailed Description

[Timeout](#) queue to be used in l4re server loop.

Definition at line 55 of file [ipc\\_timeout\\_queue](#).

## 16.171.2 Member Function Documentation

### 16.171.2.1 add()

```
void L4::Ipc_svr::Timeout_queue::add (
    Timeout * timeout,
    l4_kernel_clock_t time) [inline]
```

Add a timeout to the queue.

#### Parameters

<i>timeout</i>	timeout object to add
<i>time</i>	the time when the timeout expires

#### Precondition

*timeout* must not be in any queue already

Definition at line 111 of file [ipc\\_timeout\\_queue](#).

### 16.171.2.2 handle\_expired\_timeouts()

```
void L4::Ipc_svr::Timeout_queue::handle_expired_timeouts (
    l4_kernel_clock_t now) [inline]
```

run the callbacks of expired timeouts

#### Parameters

<i>now</i>	the current time.
------------	-------------------

Definition at line 91 of file [ipc\\_timeout\\_queue](#).

References [L4::Ipc\\_svr::Timeout::expired\(\)](#).

Here is the call graph for this function:



**16.171.2.3 next\_timeout()**

```
l4_kernel_clock_t L4::Ipc_svr::Timeout_queue::next_timeout () const [inline]
```

Get the time for the next timeout.

**Returns**

the time for the next timeout or 0 if there is none

Definition at line 65 of file [ipc\\_timeout\\_queue](#).

Referenced by [timeout\\_expired\(\)](#).

Here is the caller graph for this function:

**16.171.2.4 remove()**

```
void L4::Ipc_svr::Timeout_queue::remove (
    Timeout * timeout) [inline]
```

Remove *timeout* from the queue.

**Parameters**

<i>timeout</i>	timeout to remove from timeout queue
----------------	--------------------------------------

**Precondition**

*timeout* must be in this queue

Definition at line 126 of file [ipc\\_timeout\\_queue](#).

**16.171.2.5 timeout\_expired()**

```
bool L4::Ipc_svr::Timeout_queue::timeout_expired (
    l4_kernel_clock_t now) const [inline]
```

Determine if a timeout has happened.

**Parameters**

<i>now</i>	The current time.
------------	-------------------

### Return values

<i>true</i>	There is at least one expired timeout in the queue. <i>false</i> No expired timeout in the queue.
-------------	---------------------------------------------------------------------------------------------------

Definition at line 81 of file [ipc\\_timeout\\_queue](#).

References [next\\_timeout\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- `l4/cxx/ipc_timeout_queue`

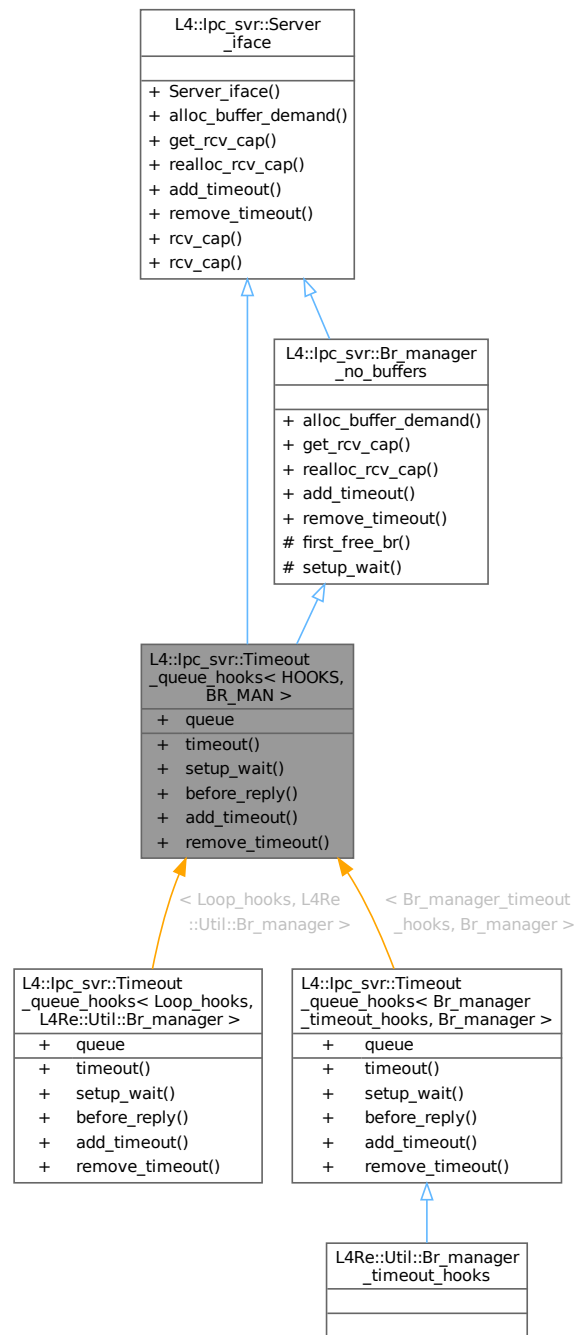
## 16.172 L4::lpc\_svr::Timeout\_queue\_hooks< HOOKS, BR\_MAN > Class Template Reference

Loop hooks mixin for integrating a timeout queue into the server loop.

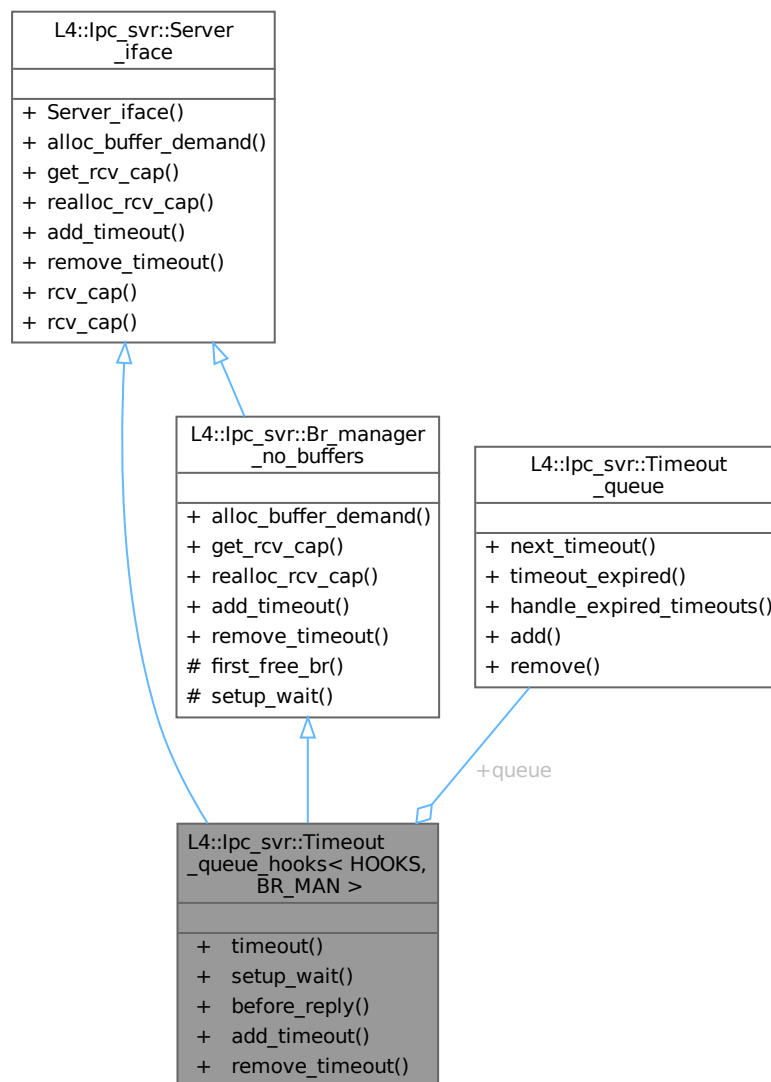
```
#include <ipc_timeout_queue>
```



Inheritance diagram for L4::lpc\_svr::Timeout\_queue\_hooks< HOOKS, BR\_MAN >:



Collaboration diagram for L4::lpc\_svr::Timeout\_queue\_hooks< HOOKS, BR\_MAN >:



## Public Member Functions

- `l4_timeout_t timeout ()`  
*get the time for the next timeout*
- `void setup_wait (l4_utcb_t *utcb, L4::lpc_svr::Reply_mode mode)`  
*setup\_wait() for the server loop*
- `L4::lpc_svr::Reply_mode before_reply (l4_msgtag_t, l4_utcb_t *)`  
*server loop hook*
- `int add_timeout (Timeout *timeout, l4_kernel_clock_t time) override`  
*Add a timeout to the queue for time time.*
- `int remove_timeout (Timeout *timeout) override`  
*Remove timeout from the queue.*

## Public Member Functions inherited from L4::lpc\_svr::Server\_iface

- **Server\_iface** ()  
*Make a server interface.*
- template<typename T>  
L4::Cap< T > **rcv\_cap** (int index) const  
*Get given receive buffer as typed capability.*
- L4::Cap< void > **rcv\_cap** (int index) const  
*Get receive cap with the given index as generic (void) type.*

## Public Member Functions inherited from L4::lpc\_svr::Br\_manager\_no\_buffers

- int **alloc\_buffer\_demand** (Demand const &demand) override  
*Tells the server to allocate buffers for the given demand.*
- L4::Cap< void > **get\_rcv\_cap** (int) const override  
*Returns L4::Cap< void> ::Invalid, we have no buffer management.*
- int **realloc\_rcv\_cap** (int) override  
*Returns -L4\_ENOMEM, we have no buffer management.*

## Data Fields

- **Timeout\_queue** queue  
*Use this timeout queue.*

## Additional Inherited Members

## Public Types inherited from L4::lpc\_svr::Server\_iface

- typedef L4::Type\_info::Demand **Demand**  
*Data type expressing server-side demand for receive buffers.*

## Protected Member Functions inherited from L4::lpc\_svr::Br\_manager\_no\_buffers

- unsigned **first\_free\_br** () const  
*Returns 1 as first free buffer.*
- void **setup\_wait** (l4\_utcb\_t \*utcb, L4::lpc\_svr::Reply\_mode)  
*Setup wait function for the server loop (Server<>).*

### 16.172.1 Detailed Description

```
template<typename HOOKS, typename BR_MAN = Br_manager_no_buffers>
class L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >
```

Loop hooks mixin for integrating a timeout queue into the server loop.

#### Template Parameters

<i>HOOKS</i>	has to inherit from <a href="#">Timeout_queue_hooks&lt;&gt;</a> and provide the functions <code>now()</code> that has to return the current time.
<i>BR_MAN</i>	This used as a base class for and provides the API for selecting the buffer register (BR) that is used to store the timeout value. This is usually <a href="#">L4Re::Util::Br_manager</a> or <a href="#">L4::lpc_svr::Br_manager_no_buffers</a> .

Definition at line 150 of file [ipc\\_timeout\\_queue](#).

## 16.172.2 Member Function Documentation

### 16.172.2.1 `add_timeout()`

```
template<typename HOOKS, typename BR_MAN = Br_manager_no_buffers>
int L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >::add_timeout (
    Timeout * timeout,
    l4_kernel_clock_t time) [inline], [override], [virtual]
```

Add a timeout to the queue for time *time*.

#### Parameters

<i>timeout</i>	The timeout object to add into the queue (must not be in any queue currently).
<i>time</i>	The time when the timeout shall expire.

#### Precondition

timeout must not be in any queue.

#### Note

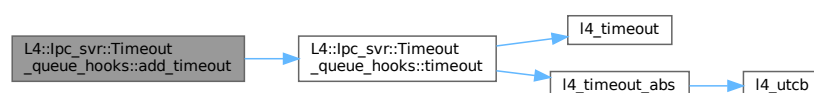
The timeout is automatically dequeued before the [Timeout::expired\(\)](#) function is called

Implements [L4::lpc\\_svr::Server\\_iface](#).

Definition at line 202 of file [ipc\\_timeout\\_queue](#).

References [queue](#), and [timeout\(\)](#).

Here is the call graph for this function:



### 16.172.2.2 remove\_timeout()

```
template<typename HOOKS, typename BR_MAN = Br_manager_no_buffers>
int L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >::remove_timeout (
    Timeout * timeout) [inline], [override], [virtual]
```

Remove timeout from the queue.

#### Parameters

---

<i>timeout</i>	The timeout object to be removed from the queue.
----------------	--------------------------------------------------

**Note**

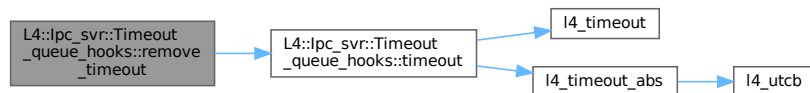
This function may be safely called even if the timeout is not currently enqueued.  
 in [Timeout::expired\(\)](#) the timeout is already dequeued!

Implements [L4::ipc\\_svr::Server\\_iface](#).

Definition at line 215 of file [ipc\\_timeout\\_queue](#).

References [queue](#), and [timeout\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

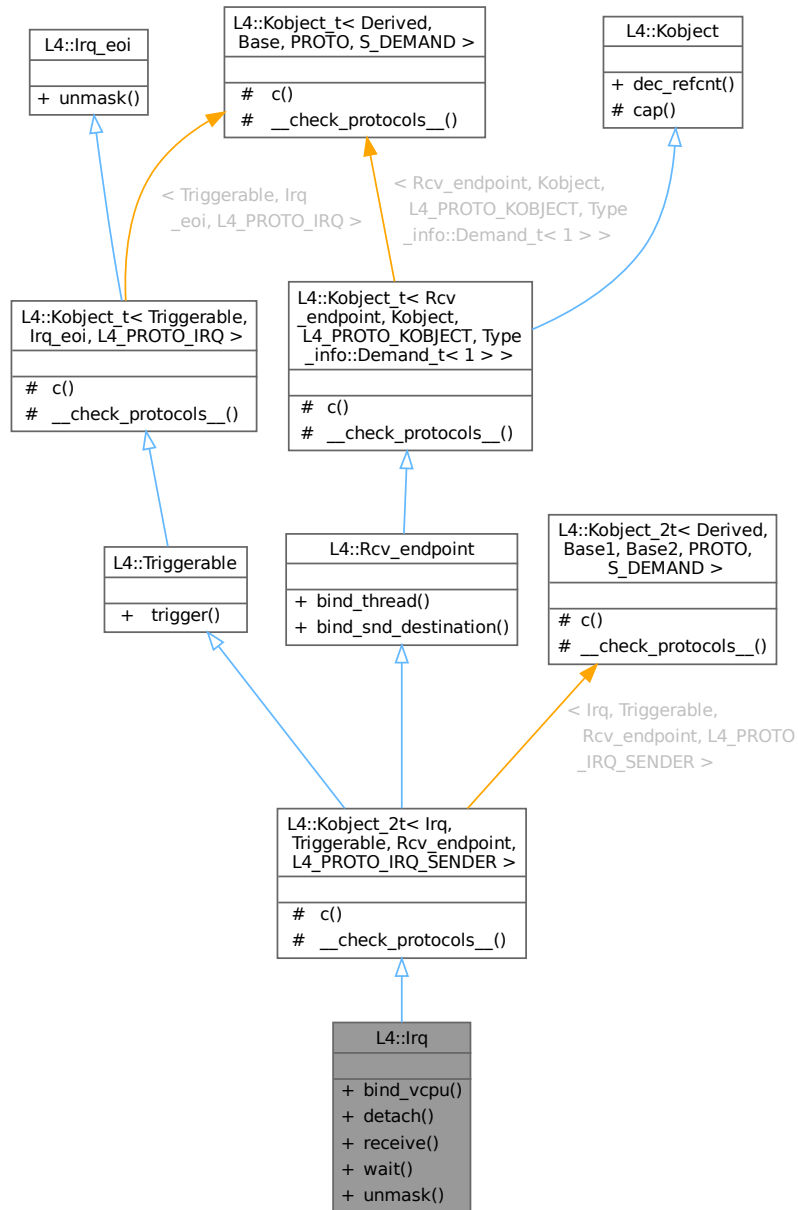
- `I4/cxx/ipc_timeout_queue`

## 16.173 L4::Irq Class Reference

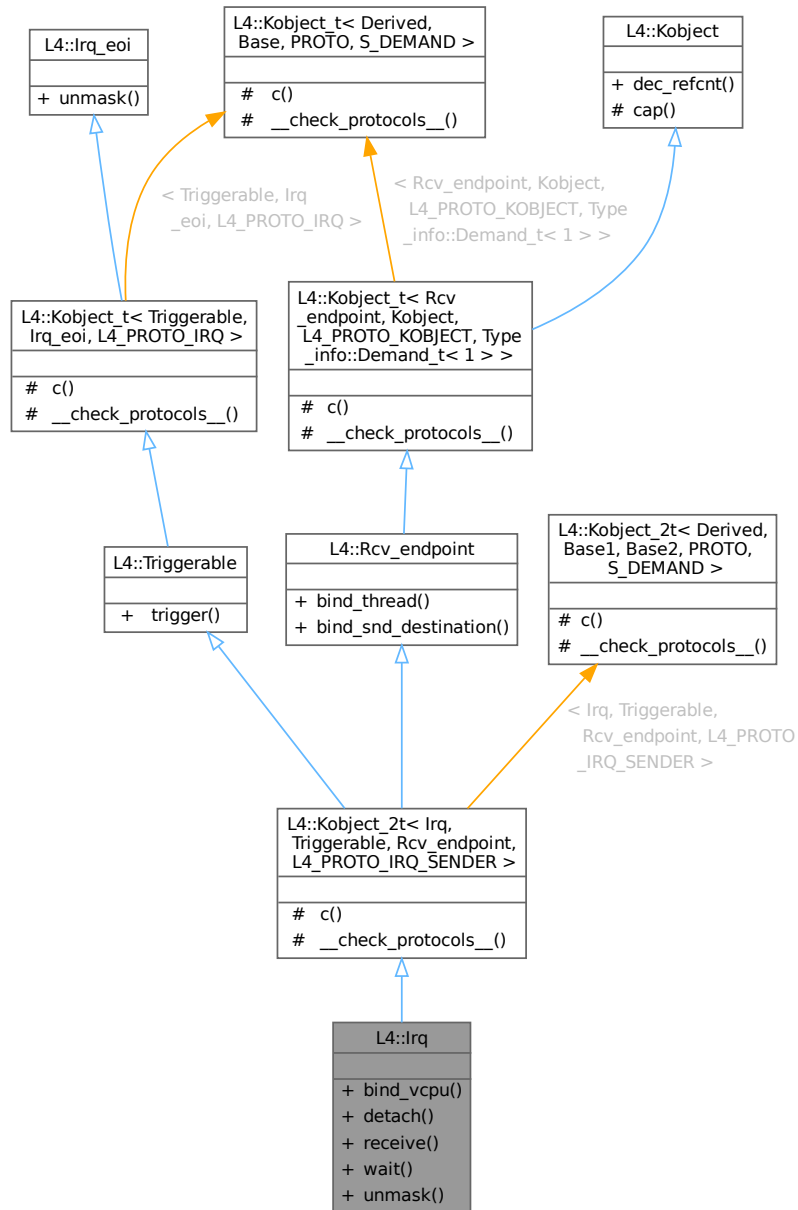
C++ [Irq](#) interface, see [IRQs](#) for the C interface.

```
#include <irq>
```

Inheritance diagram for L4::Irq:



Collaboration diagram for L4::Irq:



## Public Member Functions

- `l4_msgtag_t bind_vcpu (L4::Cap< Thread > const &thread, l4_umword_t cfg, l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Bind a thread to this Irq for vCPU interrupt forwarding.*
- `l4_msgtag_t detach (l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Detach from this interrupt.*
- `l4_msgtag_t receive (l4_timeout_t timeout=L4_IPC_NEVER, l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Unmask and wait for this IRQ.*



- `l4_msgtag_t wait (l4_umword_t *label, l4_timeout_t timeout=L4_IPC_NEVER, l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Unmask IRQ and (open) wait for any message.*
- `l4_msgtag_t unmask (l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Unmask this IRQ.*

### Public Member Functions inherited from [L4::Triggerable](#)

- `l4_msgtag_t trigger (l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Trigger the object.*

### Public Member Functions inherited from [L4::Irq\\_eoi](#)

- `l4_msgtag_t unmask (unsigned irqnum, l4_umword_t *label=0, l4_timeout_t to=L4_IPC_NEVER, l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Unmask the given interrupt line.*

### Public Member Functions inherited from [L4::Rcv\\_endpoint](#)

- `l4_msgtag_t bind_thread (lpc::Cap< Thread > t, l4_umword_t label)`  
*Bind the IPC receive endpoint to a thread.*
- `l4_msgtag_t bind_snd_destination (Cap< Snd_destination > snd_dst, l4_umword_t label)`  
*Bind a send destination (a thread or thread group) to an IPC receive endpoint.*

### Public Member Functions inherited from [L4::Kobject](#)

- `l4_msgtag_t dec_refcnt (l4_mword_t diff, l4_utcb_t *utcb=l4_utcb())`  
*Decrement the in kernel reference counter for the object.*

### Additional Inherited Members

### Protected Types inherited from

#### [L4::Kobject\\_2t< Irq, Triggerable, Rcv\\_endpoint, L4\\_PROTO\\_IRQ\\_SENDER >](#)

- `typedef Irq Class`  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- `typedef Typeid::Iface< PROTO, Irq > __lface`  
*The interface description for the derived class.*
- `typedef Typeid::Merge_list< Typeid::Iface_list< __lface >, Typeid::Merge_list< typename Triggerable::__lface_list, typename Rcv_endpoint::__lface_list > > __lface_list`  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Types inherited from [L4::Kobject\\_t< Triggerable, Irq\\_eoi, L4\\_PROTO\\_IRQ >](#)

- `typedef Triggerable Class`  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- `typedef Typeid::Iface< PROTO, Triggerable > __lface`  
*The interface description for the derived class.*
- `typedef Typeid::Merge_list< Typeid::Iface_list< __lface >, typename Irq_eoi::__lface_list > __lface_list`  
*The list of all RPC interfaces provided directly or through inheritance.*

**Protected Types inherited from****L4::Kobject\_t< Rcv\_endpoint, Kobject, L4\_PROTO\_KOBJECT, Type\_info::Demand\_t< 1 > >**

- typedef **Rcv\_endpoint Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< PROTO, **Rcv\_endpoint** > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename Kobject::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

**Protected Member Functions inherited from****L4::Kobject\_2t< Irq, Triggerable, Rcv\_endpoint, L4\_PROTO\_IRQ\_SENDER >**

- **L4::Cap< Class > c ()** const noexcept  
*Get the capability to ourselves.*

**Protected Member Functions inherited from****L4::Kobject\_t< Triggerable, Irq\_eoi, L4\_PROTO\_IRQ >**

- **L4::Cap< Class > c ()** const noexcept  
*Get the capability to ourselves.*

**Protected Member Functions inherited from****L4::Kobject\_t< Rcv\_endpoint, Kobject, L4\_PROTO\_KOBJECT, Type\_info::Demand\_t< 1 > >**

- **L4::Cap< Class > c ()** const noexcept  
*Get the capability to ourselves.*

**Protected Member Functions inherited from [L4::Kobject](#)**

- **l4\_cap\_idx\_t cap ()** const noexcept  
*Return capability selector.*

**Static Protected Member Functions inherited from****L4::Kobject\_2t< Irq, Triggerable, Rcv\_endpoint, L4\_PROTO\_IRQ\_SENDER >**

- static void **\_\_check\_protocols\_\_ ()** noexcept  
*Helper to check for protocol conflicts.*

**Static Protected Member Functions inherited from****L4::Kobject\_t< Triggerable, Irq\_eoi, L4\_PROTO\_IRQ >**

- static void **\_\_check\_protocols\_\_ ()** noexcept  
*Helper to check for protocol conflicts.*

**Static Protected Member Functions inherited from****L4::Kobject\_t< Rcv\_endpoint, Kobject, L4\_PROTO\_KOBJECT, Type\_info::Demand\_t< 1 > >**

- static void `__check_protocols__()` noexcept  
*Helper to check for protocol conflicts.*

**16.173.1 Detailed Description**

C++ [Irq](#) interface, see [IRQs](#) for the C interface.

**Note**

"IRQ" is short for "interrupt request". This is often used interchangeably for "interrupt"

The [Irq](#) class provides access to abstract interrupts provided by the microkernel. Interrupts may be

- hardware interrupts provided by the platform interrupt controller,
- virtual device interrupts provided by the microkernel's virtual devices (virtual serial or trace buffer) or
- virtual interrupts that can be triggered by user programs (IRQs) via the inherited method [L4::Triggerable::trigger\(\)](#).

For hardware and virtual device interrupts the [Irq](#) object must be bound to an interrupt source, see [L4::lcu](#). To receive interrupts, the [Irq](#) object must be bound to a thread, see [L4::Rcv\\_endpoint](#).

[Irq](#) objects can be created using a factory, see the [L4::Factory](#) API ([L4::Factory::create\(\)](#)).

**Include File**

```
#include <l4/sys/irq>
```

For the C interface refer to the [IRQs](#) API for an overview.

**Examples**

[examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#).

Definition at line 120 of file [irq](#).

**16.173.2 Member Function Documentation****16.173.2.1 bind\_vcpu()**

```
l4_msgtag_t L4::Irq::bind_vcpu (
    L4::Cap< Thread > const & thread,
    l4_umword_t cfg,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Bind a thread to this [Irq](#) for vCPU interrupt forwarding.

If the interrupt is triggered, the kernel will directly inject the interrupt into the guest. This requires that the thread is currently in extended vCPU user mode. Otherwise the interrupt will stay pending and gets injected on the next vCPU user mode transition. Optionally a doorbell [Irq](#) can be registered on the thread (see [Thread::register\\_doorbell\\_irq\(\)](#)) that is triggered in this case.

If a guest has acknowledged the interrupt but has not yet issued an EOI (i.e. the interrupt is in "active" state), it is not possible to bind the [Irq](#) to a new thread object. Either wait for the guest to issue the EOI or [detach\(\)](#) from the current thread. In this case the interrupt will stay active in the guest and it is the responsibility of the VMM to handle the eventual EOI of the guest.

**Parameters**

<i>thread</i>	<a href="#">Thread</a> object this <a href="#">Irq</a> shall be bound to.
<i>cfg</i>	Architecture specific interrupt configuration.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

### Returns

Syscall return tag

### Return values

<code>-L4_EPERM</code>	Insufficient permissions; see precondition.
<code>-L4_EBUSY</code>	Cannot bind to the new thread because interrupt is active on previous thread and guest has to issue end-of-interrupt first.
<code>-L4_ENOSYS</code>	The kernel does not support direct interrupt forwarding.

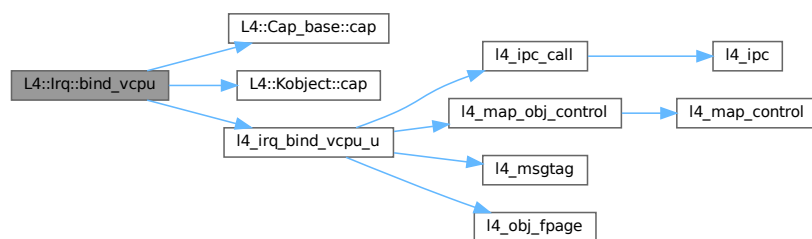
### Precondition

The invoked [Irq](#) capability and the capability `thread` both must have the permission [L4\\_CAP\\_FPAGE\\_S](#).

Definition at line 158 of file [irq](#).

References [L4::Cap\\_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4\\_irq\\_bind\\_vcpu\\_u\(\)](#).

Here is the call graph for this function:



### 16.173.2.2 detach()

```

l4_msgtag_t L4::Irq::detach (
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]

```

Detach from this interrupt.

### Parameters

<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .
-------------	------------------------------------------------------------------------------------------------------------

**Returns**

Syscall return tag

**Return values**

0	Successfully detached, there was no interrupt pending.
1	Successfully detached, there was an interrupt pending.
2	Successfully detached, an active vIRQ was abandoned.
-L4_EPERM	Insufficient permissions; see precondition.

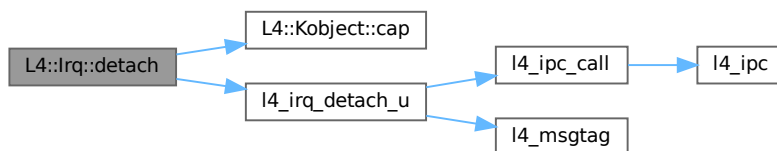
**Precondition**

The invoked [Irq](#) capability must have the permission [L4\\_CAP\\_FPAGE\\_S](#).

Definition at line 176 of file [irq](#).

References [L4::Kobject::cap\(\)](#), and [l4\\_irq\\_detach\\_u\(\)](#).

Here is the call graph for this function:

**16.173.2.3 receive()**

```

l4_msgtag_t L4::Irq::receive (
    l4_timeout_t timeout = L4_IPC_NEVER,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]

```

Unmask and wait for this IRQ.

**Parameters**

<i>timeout</i>	Timeout.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

**Returns**

Syscall return tag

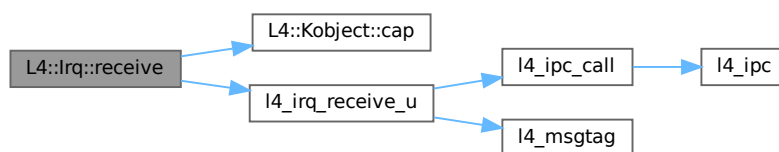
**Note**

If this is the function normally used for your IRQs consider using [L4::Semaphore](#) instead of [L4::Irq](#).

Definition at line 191 of file [irq](#).

References [L4::Kobject::cap\(\)](#), [L4\\_IPC\\_NEVER](#), and [l4\\_irq\\_receive\\_u\(\)](#).

Here is the call graph for this function:

**16.173.2.4 unmask()**

```
l4_msgtag_t L4::Irq::unmask (
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Unmask this IRQ.

**Parameters**

<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .
-------------	------------------------------------------------------------------------------------------------------------

**Returns**

Syscall return tag for a send-only operation, this means there is no return value except [L4\\_MSGTAG\\_ERROR](#) indicating success or failure of the send operation. Use [l4\\_ipc\\_error\(\)](#) to check for errors and **do not** use [l4\\_error\(\)](#).

[Irq::wait\(\)](#) and [Irq::receive\(\)](#) operations already include an [unmask\(\)](#), do not use an extra [unmask\(\)](#) in these cases.

Definition at line 221 of file [irq](#).

References [L4\\_IPC\\_NEVER](#), and [unmask\(\)](#).

Referenced by [unmask\(\)](#), and [wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.173.2.5 wait()

```

l4_msgtag_t L4::Irq::wait (
    l4_umword_t * label,
    l4_timeout_t timeout = L4_IPC_NEVER,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Unmask IRQ and (open) wait for any message.

#### Parameters

<i>label</i>	The <i>protected label</i> shall be received here.
<i>timeout</i>	Timeout.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

#### Returns

Syscall return tag

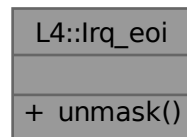
Definition at line 204 of file [irq](#).

References [L4\\_IPC\\_NEVER](#), and [unmask\(\)](#).





Collaboration diagram for L4::Irq\_eoi:



## Public Member Functions

- [l4\\_msgtag\\_t unmask](#) (unsigned irqnum, [l4\\_umword\\_t](#) \*label=0, [l4\\_timeout\\_t](#) to=[L4\\_IPC\\_NEVER](#), [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept

*Unmask the given interrupt line.*

## 16.174.1 Detailed Description

Interface for sending an unmask message to an object.

The object is usually an ICU or an IRQ.

When the kernel receives an IRQ, it masks the interrupt line at the interrupt controller and immediately acknowledges the interrupt. This interface is used to let the kernel know that userspace has dealt with the IRQ. The kernel will unmask the interrupt line and further IRQs can then be delivered.

See also

[L4::lcu](#), [L4::Irq](#)

Definition at line 37 of file [irq](#).

## 16.174.2 Member Function Documentation

### 16.174.2.1 unmask()

```

l4_msgtag_t L4::Irq_eoi::unmask (
    unsigned irqnum,
    l4_umword_t * label = 0,
    l4_timeout_t to = L4_IPC_NEVER,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Unmask the given interrupt line.

When the object is an IRQ, the given interrupt line is ignored and instead the line which the IRQ is bound to (if any) is unmasked.

Its counterpart for explicitly masking an interrupt line is [L4::lcu::mask\(\)](#).

## Parameters

	<i>irqnum</i>	The interrupt line that shall be unmasked. Ignored if the object is an IRQ.
out	<i>label</i>	If NULL, this is a send-only unmask. If not NULL, this operation enters an open wait and the <i>protected label</i> shall be received here.
	<i>to</i>	The timeout-pair (send and receive) that shall be used for this operation. The receive timeout is used with a non-NULL <i>label</i> only.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

### Returns

Syscall return tag. If *label* is NULL, this function performs an IPC send-only operation and there is no return value except [L4\\_MSGTAG\\_ERROR](#) indicating success or failure of the send operation. In this case use [l4\\_ipc\\_error\(\)](#) to check for errors and **do not** use [l4\\_error\(\)](#).

Definition at line 64 of file [irq](#).

References [L4::Kobject::cap\(\)](#), and [L4\\_IPC\\_NEVER](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

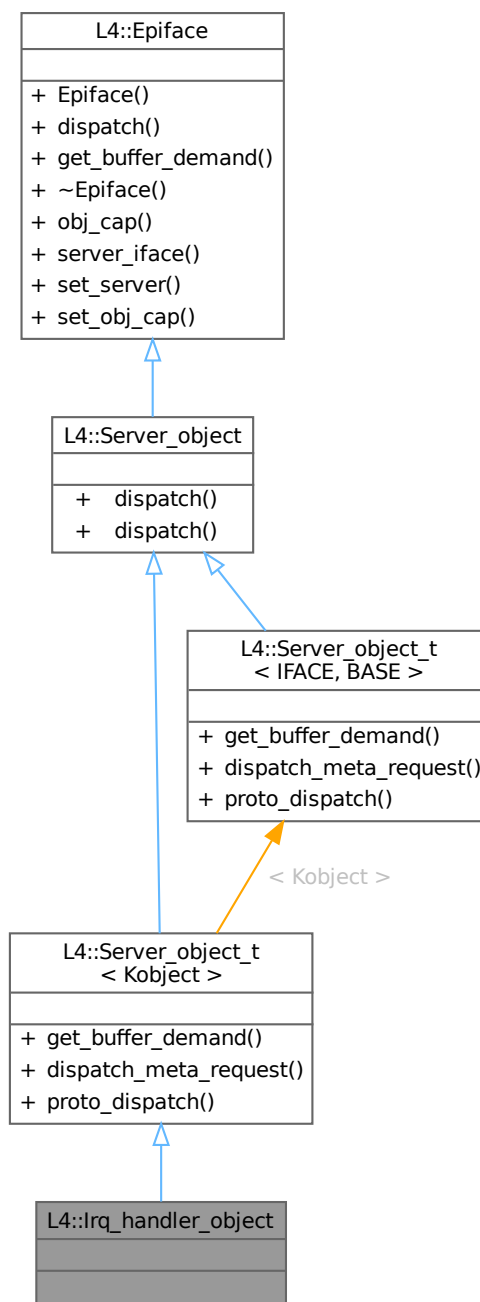
- [l4/sys/irq](#)

## 16.175 L4::Irq\_handler\_object Struct Reference

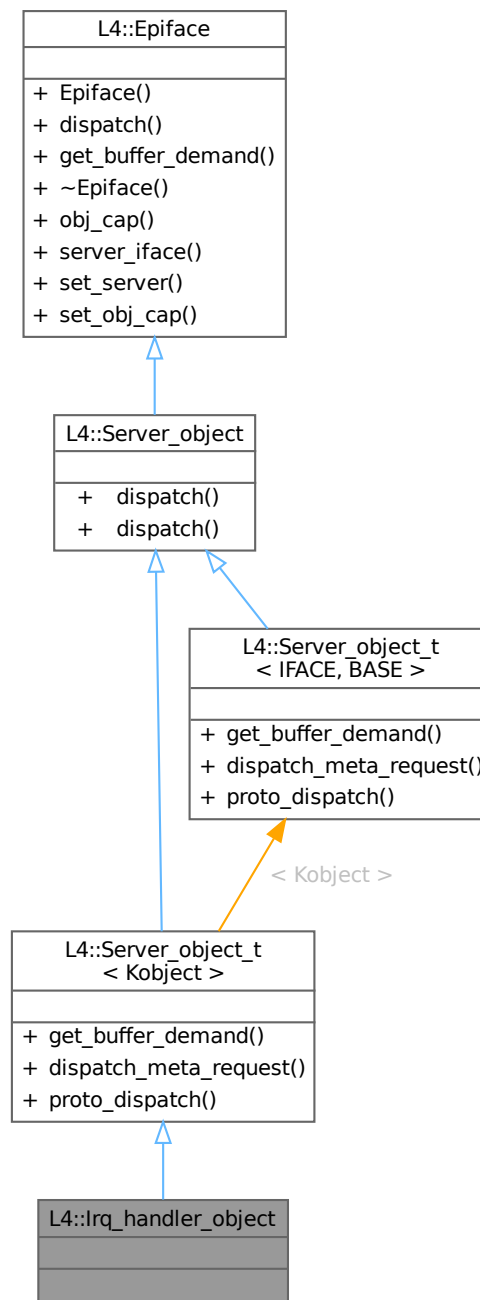
[Server](#) object base class for handling IRQ messages.

```
#include <ipc_server>
```

Inheritance diagram for L4::lrq\_handler\_object:



Collaboration diagram for L4::Irq\_handler\_object:



#### Additional Inherited Members

#### Public Types inherited from **L4::Server\_object\_t< Kobject >**

- typedef **Kobject** Interface

*Data type of the IPC interface definition.*

**Public Types inherited from L4::Epiface**

- typedef [lpc\\_svr::Server\\_iface](#) **Server\_iface**  
*Type for abstract server interface.*
- typedef [lpc\\_svr::Server\\_iface::Demand](#) **Demand**  
*Type for server-side receive buffer demand.*

**Public Member Functions inherited from L4::Server\_object\_t< Kobject >**

- [L4::Server\\_object::Demand](#) **get\_buffer\_demand** () const override
- int [dispatch\\_meta\\_request](#) ([L4::lpc::lostream](#) &ios)  
*Implementation of the meta protocol based on IFACE.*

**Public Member Functions inherited from L4::Server\_object**

- virtual int [dispatch](#) (unsigned long rights, [lpc::lostream](#) &ios)=0  
*The abstract handler for client requests to the object.*
- [l4\\_msgtag\\_t](#) [dispatch](#) ([l4\\_msgtag\\_t](#) tag, unsigned rights, [l4\\_utcb\\_t](#) \*utcb) override  
*The abstract handler for client requests to the object.*

**Public Member Functions inherited from L4::Epiface**

- **Epiface** ()  
*Make a server object.*
- virtual ~**Epiface** ()=0  
*Destroy the object.*
- Stored\_cap [obj\\_cap](#) () const  
*Get the capability to the kernel object belonging to this object.*
- [Server\\_iface](#) \* [server\\_iface](#) () const  
*Get pointer to server interface at which the object is currently registered.*
- int [set\\_server](#) ([Server\\_iface](#) \*srv, [Cap](#)< void > cap, bool managed=false)  
*Set server registration info for the object.*
- void [set\\_obj\\_cap](#) ([Cap](#)< void > const &cap)  
*Deprecated server registration function.*

**Static Public Member Functions inherited from L4::Server\_object\_t< Kobject >**

- static int [proto\\_dispatch](#) (THIS \*self, [l4\\_umword\\_t](#) rights, [L4::lpc::lostream](#) &ios)  
*Implementation of protocol-based dispatch for this server object.*

**16.175.1 Detailed Description**

[Server](#) object base class for handling IRQ messages.

This server object base class implements the empty interface ([L4::Kobject](#)). The implementation of [Server\\_object::dispatch\(\)](#) must return [-L4\\_ENOREPLY](#), because IRQ messages do not handle replies.

**Examples**

[examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#).

Definition at line 161 of file [ipc\\_server](#).

The documentation for this struct was generated from the following file:

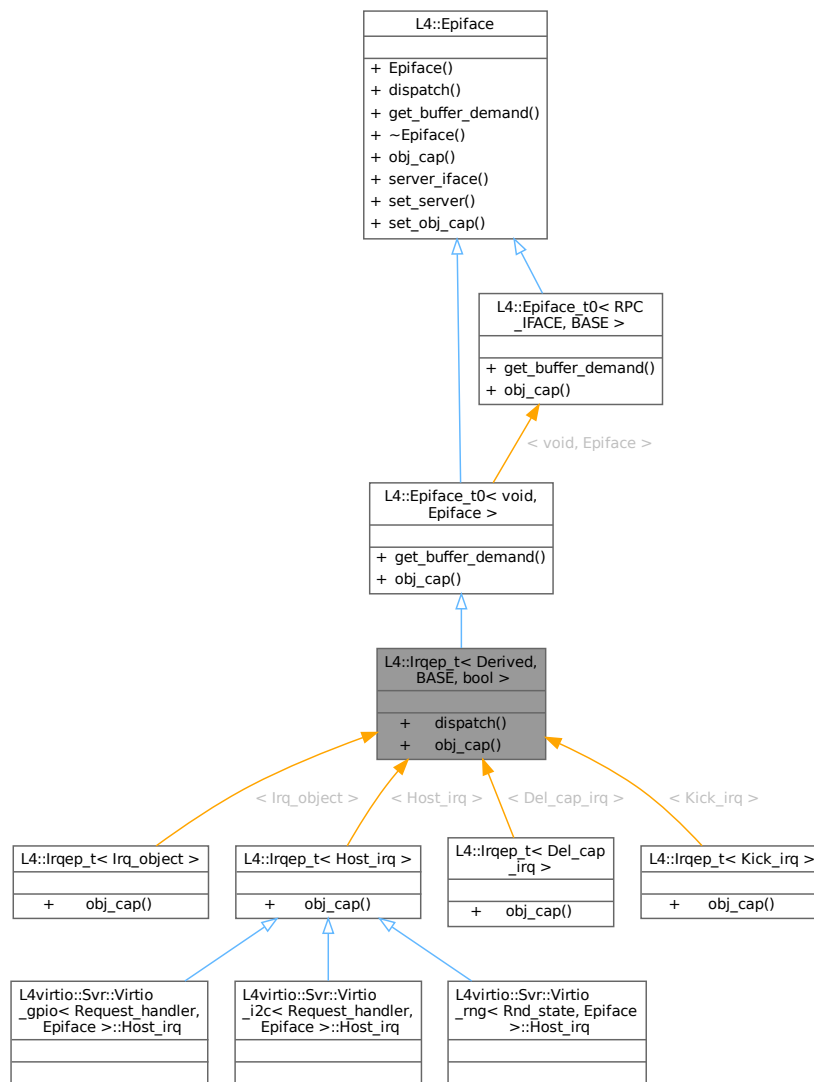
- [l4/cxx/ipc\\_server](#)

## 16.176 L4::lrqep\_t< Derived, BASE, bool > Struct Template Reference

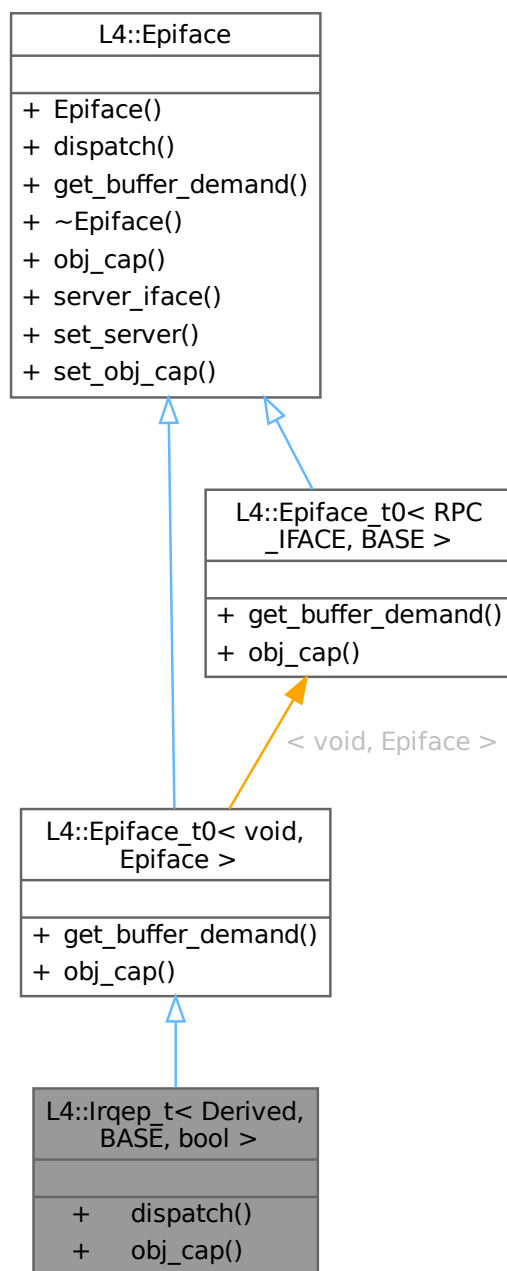
[Epiface](#) implementation for interrupt handlers.

```
#include <ipc_epiface>
```

Inheritance diagram for L4::lrqep\_t< Derived, BASE, bool >:



Collaboration diagram for L4::lrqep\_t< Derived, BASE, bool >:



## Public Member Functions

- `l4_msgtag_t dispatch (l4_msgtag_t, unsigned, l4_utcb_t *)` final  
The abstract handler for client requests to the object.
- `Cap< L4::lrq > obj_cap ()` const  
Get the (typed) capability to this object.

## Public Member Functions inherited from [L4::Epiface\\_t0< void, Epiface >](#)

- [Type\\_info::Demand](#) **get\_buffer\_demand** () const  
*Get the server-side buffer demand based in IFACE.*
- [Cap< void >](#) **obj\_cap** () const  
*Get the (typed) capability to this object.*

## Public Member Functions inherited from [L4::Epiface](#)

- **Epiface** ()  
*Make a server object.*
- virtual **~Epiface** ()=0  
*Destroy the object.*
- Stored\_cap [obj\\_cap](#) () const  
*Get the capability to the kernel object belonging to this object.*
- [Server\\_iface](#) \* **server\_iface** () const  
*Get pointer to server interface at which the object is currently registered.*
- int **set\_server** ([Server\\_iface](#) \*srv, [Cap< void >](#) cap, bool managed=false)  
*Set server registration info for the object.*
- void **set\_obj\_cap** ([Cap< void >](#) const &cap)  
*Deprecated server registration function.*

## Additional Inherited Members

## Public Types inherited from [L4::Epiface\\_t0< void, Epiface >](#)

- typedef void **Interface**  
*Data type of the IPC interface definition.*

## Public Types inherited from [L4::Epiface](#)

- typedef [lpc\\_svr::Server\\_iface](#) **Server\_iface**  
*Type for abstract server interface.*
- typedef [lpc\\_svr::Server\\_iface::Demand](#) **Demand**  
*Type for server-side receive buffer demand.*

### 16.176.1 Detailed Description

```
template<typename Derived, typename BASE = Epiface, bool = cxx::is_polymorphic<BASE>::value>
struct L4::Irqep_t< Derived, BASE, bool >
```

[Epiface](#) implementation for interrupt handlers.

#### Template Parameters

<i>Derived</i>	<a href="#">Irq</a> handler implementation class. The class must provide a single function <code>handle_irq()</code> .
----------------	------------------------------------------------------------------------------------------------------------------------



<i>BASE</i>	Base <a href="#">Epiface</a> class.
-------------	-------------------------------------

Definition at line [282](#) of file [ipc\\_epiface](#).

## 16.176.2 Member Function Documentation

### 16.176.2.1 dispatch()

```
template<typename Derived, typename BASE = Epiface, bool = cxx::is_polymorphic<BASE>::value>
l4_msgtag_t L4::lrqep_t< Derived, BASE, bool >::dispatch (
    l4_msgtag_t tag,
    unsigned rights,
    l4_utcb_t * utcb) [inline], [final], [virtual]
```

The abstract handler for client requests to the object.

#### Parameters

<i>tag</i>	The message tag for this invocation.
<i>rights</i>	The rights bits in the invoked capability.
<i>utcb</i>	The UTCB used for the invocation.

#### Return values

<i>-L4_ENOREPLY</i>	No reply message is send.
<i>&lt;0</i>	Error, reply with error code.
<i>&gt;=0</i>	Success, reply with return value.

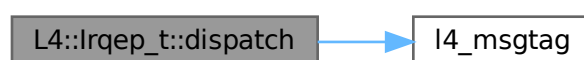
This function must be implemented by application specific server objects.

Implements [L4::Epiface](#).

Definition at line [284](#) of file [ipc\\_epiface](#).

References [L4\\_ENOREPLY](#), and [l4\\_msgtag\(\)](#).

Here is the call graph for this function:



**16.176.2.2 obj\_cap()**

```
template<typename Derived, typename BASE = Epiface, bool = cxx::is_polymorphic<BASE>::value>
Cap< L4::Irq > L4::Irqep_t< Derived, BASE, bool >::obj_cap () const [inline]
```

Get the (typed) capability to this object.

**Returns**

[Irq](#) capability for the kernel object behind the server.

Definition at line 294 of file [ipc\\_epiface](#).

References [L4::cap\\_cast\(\)](#).

Here is the call graph for this function:



The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc\\_epiface](#)

**16.177 L4::Kip::Mem\_desc Class Reference**

Memory descriptors stored in the kernel interface page.

```
#include <kip>
```

Collaboration diagram for L4::Kip::Mem\_desc:

L4::Kip::Mem_desc
<ul style="list-style-type: none"> <li>+ Mem_desc()</li> <li>+ start()</li> <li>+ end()</li> <li>+ size()</li> <li>+ type()</li> <li>+ sub_type()</li> <li>+ is_virtual()</li> <li>+ eager_map()</li> <li>+ set()</li> <li>+ first()</li> <li>+ count()</li> <li>+ count()</li> <li>+ all()</li> <li>+ all()</li> </ul>

## Public Types

- enum [Mem\\_type](#) {  
[Undefined](#) = 0x0 , [Conventional](#) = 0x1 , [Reserved](#) = 0x2 , [Dedicated](#) = 0x3 ,  
[Shared](#) = 0x4 , [Info](#) = 0xd , [Bootloader](#) = 0xe , [Arch](#) = 0xf }  
*Memory types.*
- enum [Info\\_sub\\_type](#) { [Info\\_acpi\\_rsdp](#) = 0 , [Reserved\\_kernel](#) = 0 , [Reserved\\_heap](#) = 1 , [Reserved\\_mmio](#) = 2  
}
  
*Memory sub types for the [Mem\\_type::Info](#) type.*
- enum [Arch\\_sub\\_type\\_common](#) { [Arch\\_acpi\\_tables](#) = 3 , [Arch\\_acpi\\_nvs](#) = 4 }  
*Common sub types across all architectures for the [Mem\\_type::Arch](#) type.*

## Public Member Functions

- [Mem\\_desc](#) (unsigned long [start](#), unsigned long [end](#), [Mem\\_type](#) t, unsigned char st=0, bool virt=false, bool eager=false) noexcept  
*Initialize memory descriptor.*
- unsigned long [start](#) () const noexcept  
*Return start address of memory descriptor.*
- unsigned long [end](#) () const noexcept  
*Return end address of memory descriptor.*
- unsigned long [size](#) () const noexcept  
*Return size of region described by the memory descriptor.*

- `Mem_type type ()` const noexcept  
*Return type of the memory descriptor.*
- unsigned char `sub_type ()` const noexcept  
*Return sub-type of the memory descriptor.*
- unsigned `is_virtual ()` const noexcept  
*Return whether the memory descriptor describes a virtual or physical region.*
- unsigned `eager_map ()` const noexcept  
*Return whether the region shall be eligible for eager mapping in sigma0 or the root task.*
- void `set (unsigned long start, unsigned long end, Mem_type t, unsigned char st=0, bool virt=false, bool eager=false)` noexcept  
*Set values of a memory descriptor.*

### Static Public Member Functions

- static `Mem_desc * first (l4_kernel_info_t *kip)` noexcept  
*Get first memory descriptor.*
- static unsigned long `count (l4_kernel_info_t const *kip)` noexcept  
*Return number of memory descriptors stored in the kernel info page.*
- static void `count (l4_kernel_info_t *kip, unsigned count)` noexcept  
*Set number of memory descriptors.*
- static `cxx::static_vector< Mem_desc const > all (l4_kernel_info_t const *kip)`  
*Return enumerable list of memory descriptors.*
- static `cxx::static_vector< Mem_desc > all (l4_kernel_info_t *kip)`  
*Return enumerable list of memory descriptors.*

## 16.177.1 Detailed Description

Memory descriptors stored in the kernel interface page.

### Include File

```
#include <l4/sys/kip>
```

Definition at line 42 of file `kip`.

## 16.177.2 Member Enumeration Documentation

### 16.177.2.1 Arch\_sub\_type\_common

```
enum L4::Kip::Mem_desc::Arch_sub_type_common
```

Common sub types across all architectures for the `Mem_type::Arch` type.

### Enumerator

Arch_acpi_tables	Firmware ACPI tables.
------------------	-----------------------

Arch_acpi_nvs	Firmware reserved address space.
---------------	----------------------------------

Definition at line 76 of file [kip](#).

### 16.177.2.2 Info\_sub\_type

```
enum L4::Kip::Mem_desc::Info_sub_type
```

Memory sub types for the [Mem\\_type::Info](#) type.

#### Enumerator

Info_acpi_rsdp	Physical address of the ACPI root pointer.
Reserved_kernel	Kernel image.
Reserved_heap	Kernel heap.
Reserved_mmio	MMIO range reserved by kernel.

Definition at line 64 of file [kip](#).

### 16.177.2.3 Mem\_type

```
enum L4::Kip::Mem_desc::Mem_type
```

Memory types.

#### Enumerator

Undefined	Undefined memory.
Conventional	Conventional memory.
Reserved	Reserved region, do not use this memory.
Dedicated	Dedicated.
Shared	Shared.
Info	Info by boot loader.
Bootloader	Memory belongs to the boot loader.
Arch	Architecture specific memory.

Definition at line 48 of file [kip](#).

## 16.177.3 Constructor & Destructor Documentation

### 16.177.3.1 Mem\_desc()

```
L4::Kip::Mem_desc::Mem_desc (  
    unsigned long start,  
    unsigned long end,  
    Mem_type t,  
    unsigned char st = 0,  
    bool virt = false,  
    bool eager = false) [inline], [noexcept]
```

Initialize memory descriptor.

#### Parameters

---

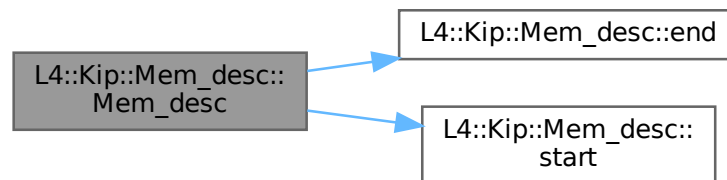
<i>start</i>	Start address
<i>end</i>	End address
<i>t</i>	Memory type
<i>st</i>	Memory subtype, defaults to 0
<i>virt</i>	True for virtual memory, false for physical memory, defaults to physical
<i>eager</i>	The region shall be eligible for eager mapping in sigma0 or the root task. This is just an optimization to prevent on-demand paging.

Definition at line 162 of file [kip](#).

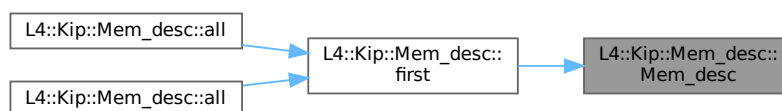
References [end\(\)](#), and [start\(\)](#).

Referenced by [first\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



## 16.177.4 Member Function Documentation

### 16.177.4.1 all() [1/2]

```

cxx::static_vector< Mem_desc > L4::Kip::Mem_desc::all (
    l4_kernel_info_t * kip)  [inline], [static]
  
```

Return enumerable list of memory descriptors.

#### Parameters

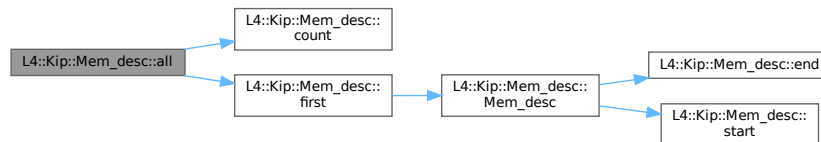
---

<i>kip</i>	Pointer to the kernel info page.
------------	----------------------------------

Definition at line 143 of file [kip](#).

References [count\(\)](#), and [first\(\)](#).

Here is the call graph for this function:



#### 16.177.4.2 [all\(\)](#) [2/2]

```

cxx::static_vector< Mem_desc const > L4::Kip::Mem_desc::all (
    l4_kernel_info_t const * kip) [inline], [static]

```

Return enumerable list of memory descriptors.

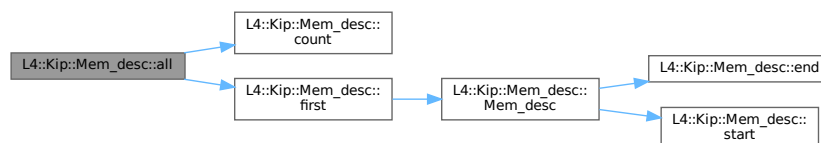
##### Parameters

<i>kip</i>	Pointer to the kernel info page.
------------	----------------------------------

Definition at line 132 of file [kip](#).

References [count\(\)](#), and [first\(\)](#).

Here is the call graph for this function:



#### 16.177.4.3 [count\(\)](#) [1/2]

```

void L4::Kip::Mem_desc::count (
    l4_kernel_info_t * kip,
    unsigned count) [inline], [static], [noexcept]

```

Set number of memory descriptors.

##### Parameters



<i>kip</i>	Pointer to the kernel info page
<i>count</i>	Number of memory descriptors

Definition at line 122 of file [kip](#).

References [count\(\)](#).

Here is the call graph for this function:



#### 16.177.4.4 count() [2/2]

```
unsigned long L4::Kip::Mem_desc::count (  
    l4_kernel_info_t const * kip) [inline], [static], [noexcept]
```

Return number of memory descriptors stored in the kernel info page.

##### Parameters

<i>kip</i>	Pointer to the kernel info page
------------	---------------------------------

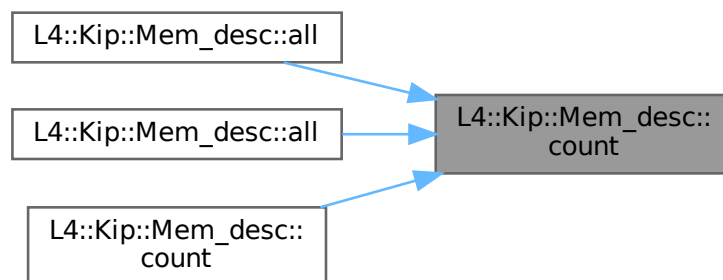
##### Returns

Number of memory descriptors in the kernel info page.

Definition at line 111 of file [kip](#).

Referenced by [all\(\)](#), [all\(\)](#), and [count\(\)](#).

Here is the caller graph for this function:



#### 16.177.4.5 end()

```
unsigned long L4::Kip::Mem_desc::end () const [inline], [noexcept]
```

Return end address of memory descriptor.

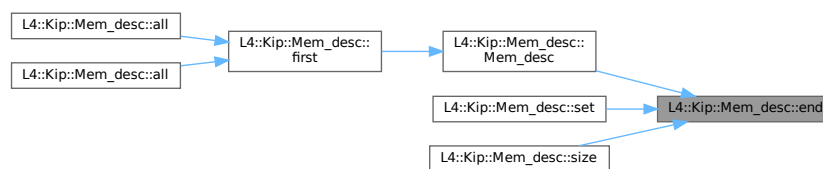
##### Returns

End address of memory descriptor

Definition at line 181 of file [kip](#).

Referenced by [Mem\\_desc\(\)](#), [set\(\)](#), and [size\(\)](#).

Here is the caller graph for this function:



#### 16.177.4.6 first()

```
Mem_desc * L4::Kip::Mem_desc::first (
    l4_kernel_info_t * kip) [inline], [static], [noexcept]
```

Get first memory descriptor.

##### Parameters

<i>kip</i>	Pointer to the kernel info page
------------	---------------------------------

##### Returns

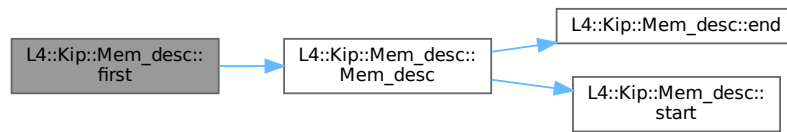
First memory descriptor stored in the kernel info page

Definition at line 93 of file [kip](#).

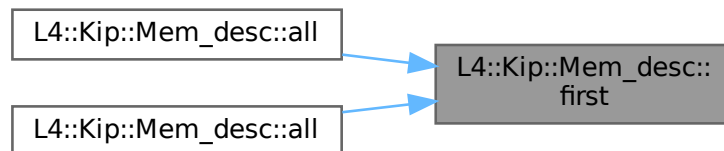
References [Mem\\_desc\(\)](#).

Referenced by [all\(\)](#), and [all\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 16.177.4.7 is\_virtual()

```
unsigned L4::Kip::Mem_desc::is_virtual () const [inline], [noexcept]
```

Return whether the memory descriptor describes a virtual or physical region.

##### Returns

True for virtual region, false for physical region.

Definition at line 213 of file [kip](#).

#### 16.177.4.8 set()

```
void L4::Kip::Mem_desc::set (
    unsigned long start,
    unsigned long end,
    Mem_type t,
    unsigned char st = 0,
    bool virt = false,
    bool eager = false) [inline], [noexcept]
```

Set values of a memory descriptor.

##### Parameters

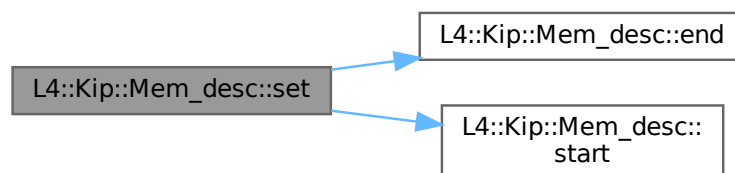
---

<i>start</i>	Start address
<i>end</i>	End address
<i>t</i>	Memory type
<i>st</i>	Sub-type, defaults to 0
<i>virt</i>	Virtual or physical memory region, defaults to physical
<i>eager</i>	The region shall be eligible for eager mapping in sigma0 or the root task. This is just an optimization to prevent on-demand paging.

Definition at line [233](#) of file [kip](#).

References [end\(\)](#), and [start\(\)](#).

Here is the call graph for this function:



#### 16.177.4.9 `size()`

```
unsigned long L4::Kip::Mem_desc::size () const [inline], [noexcept]
```

Return size of region described by the memory descriptor.

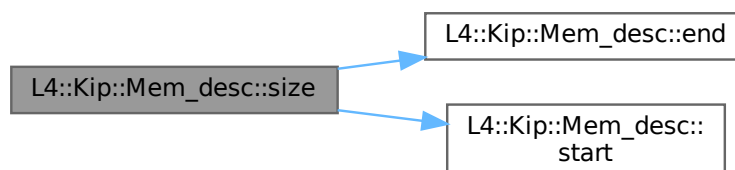
##### Returns

Size of the region described by the memory descriptor

Definition at line [188](#) of file [kip](#).

References [end\(\)](#), and [start\(\)](#).

Here is the call graph for this function:



**16.177.4.10 start()**

```
unsigned long L4::Kip::Mem_desc::start () const [inline], [noexcept]
```

Return start address of memory descriptor.

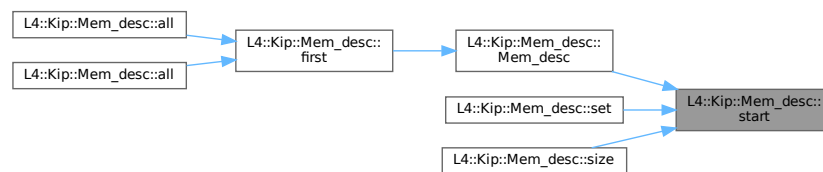
**Returns**

Start address of memory descriptor

Definition at line 174 of file [kip](#).

Referenced by [Mem\\_desc\(\)](#), [set\(\)](#), and [size\(\)](#).

Here is the caller graph for this function:

**16.177.4.11 sub\_type()**

```
unsigned char L4::Kip::Mem_desc::sub_type () const [inline], [noexcept]
```

Return sub-type of the memory descriptor.

**Returns**

Sub-type of the memory descriptor

Definition at line 205 of file [kip](#).

**16.177.4.12 type()**

```
Mem\_type L4::Kip::Mem_desc::type () const [inline], [noexcept]
```

Return type of the memory descriptor.

**Returns**

Type of the memory descriptor

Definition at line 195 of file [kip](#).

The documentation for this class was generated from the following file:

- [l4/sys/kip](#)



## 16.178.2 Member Function Documentation

### 16.178.2.1 cap()

```
l4_cap_idx_t L4::Kobject::cap () const [inline], [protected], [noexcept]
```

Return capability selector.

#### Returns

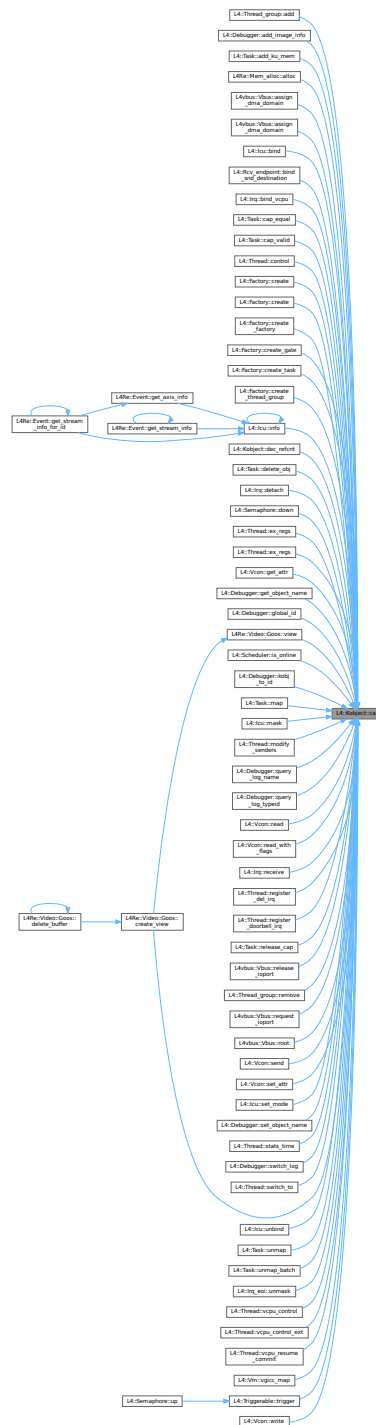
Capability selector.

This method is for derived classes to gain access to the actual capability selector.

Definition at line 69 of file [kobject](#).

Referenced by [L4::Thread\\_group::add\(\)](#), [L4::Debugger::add\\_image\\_info\(\)](#), [L4::Task::add\\_ku\\_mem\(\)](#), [L4Re::Mem\\_alloc::alloc\(\)](#), [L4vbus::Vbus::assign\\_dma\\_domain\(\)](#), [L4vbus::Vbus::assign\\_dma\\_domain\(\)](#), [L4::lcu::bind\(\)](#), [L4::Rcv\\_endpoint::bind\\_snd\\_destination\(\)](#), [L4::lrc::bind\\_vcpu\(\)](#), [L4::Task::cap\\_equal\(\)](#), [L4::Task::cap\\_valid\(\)](#), [L4::Thread::control\(\)](#), [L4::Factory::create\(\)](#), [L4::Factory::create\(\)](#), [L4::Factory::create\\_factory\(\)](#), [L4::Factory::create\\_gate\(\)](#), [L4::Factory::create\\_task\(\)](#), [L4::Factory::create\\_thread\\_group\(\)](#), [L4Re::Video::Goos::create\\_view\(\)](#), [dec\\_refcnt\(\)](#), [L4::Task::delete\\_obj\(\)](#), [L4::lrc::detach\(\)](#), [L4::Semaphore::down\(\)](#), [L4::Thread::ex\\_regs\(\)](#), [L4::Thread::ex\\_regs\(\)](#), [L4::Vcon::get\\_attr\(\)](#), [L4::Debugger::get\\_object\\_name\(\)](#), [L4::Debugger::global\\_id\(\)](#), [L4::lcu::info\(\)](#), [L4::Scheduler::is\\_online\(\)](#), [L4::Debugger::kobj\\_to\\_id\(\)](#), [L4::Task::map\(\)](#), [L4::lcu::mask\(\)](#), [L4::Thread::modify\\_senders\(\)](#), [L4::Debugger::query\\_log\\_name\(\)](#), [L4::Debugger::query\\_log\\_typeid\(\)](#), [L4::Vcon::read\(\)](#), [L4::Vcon::read\\_with\\_flags\(\)](#), [L4::lrc::receive\(\)](#), [L4::Thread::register\\_del\\_irq\(\)](#), [L4::Thread::register\\_doorbell\\_irq\(\)](#), [L4::Task::release\\_cap\(\)](#), [L4vbus::Vbus::release\\_ioport\(\)](#), [L4::Thread\\_group::remove\(\)](#), [L4vbus::Vbus::request\\_ioport\(\)](#), [L4vbus::Vbus::root\(\)](#), [L4::Vcon::send\(\)](#), [L4::Vcon::set\\_attr\(\)](#), [L4::lcu::set\\_mode\(\)](#), [L4::Debugger::set\\_object\\_name\(\)](#), [L4::Thread::stats\\_time\(\)](#), [L4::Debugger::switch\\_log\(\)](#), [L4::Thread::switch\\_to\(\)](#), [L4::Triggerable::trigger\(\)](#), [L4::lcu::unbind\(\)](#), [L4::Task::unmap\(\)](#), [L4::Task::unmap\\_batch\(\)](#), [L4::lrc::unmask\(\)](#), [L4::Thread::vcpu\\_control\(\)](#), [L4::Thread::vcpu\\_control\\_ext\(\)](#), [L4::Thread::vcpu\\_resume\\_commit\(\)](#), [L4::Vm::vgicc\\_map\(\)](#), [L4Re::Video::Goos::view\(\)](#), and [L4::Vcon::write\(\)](#).

Here is the caller graph for this function:



### 16.178.2.2 dec\_refcnt()

```
14_msgtag_t L4::Kobject::dec_refcnt (
    14_mword_t diff,
    14_utcb_t * utcb = 14_utcb()) [inline]
```



Decrement the in kernel reference counter for the object.

**Parameters**

---

<i>diff</i>	The delta that shall be subtracted from the reference count.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

## Returns

Syscall return tag

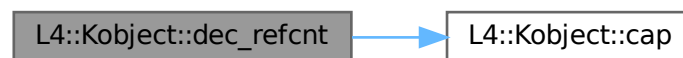
This function is intended for servers to be able to remove the servers own capability from the counted references. This leads to the semantics that the kernel will delete the object even if the capability of the server is valid. The server can detect the deletion by polling its capabilities or by using the IPC-gate deletion IRQs. And to cleanup if the clients dropped the last reference (capability) to the object.

This function only succeeds on a kernel object of type [L4::lpc\\_gate](#) which has the server right ([L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#)). For other kernel objects, -L4\_ENOSYS is returned.

Definition at line 100 of file [kobject](#).

References [cap\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

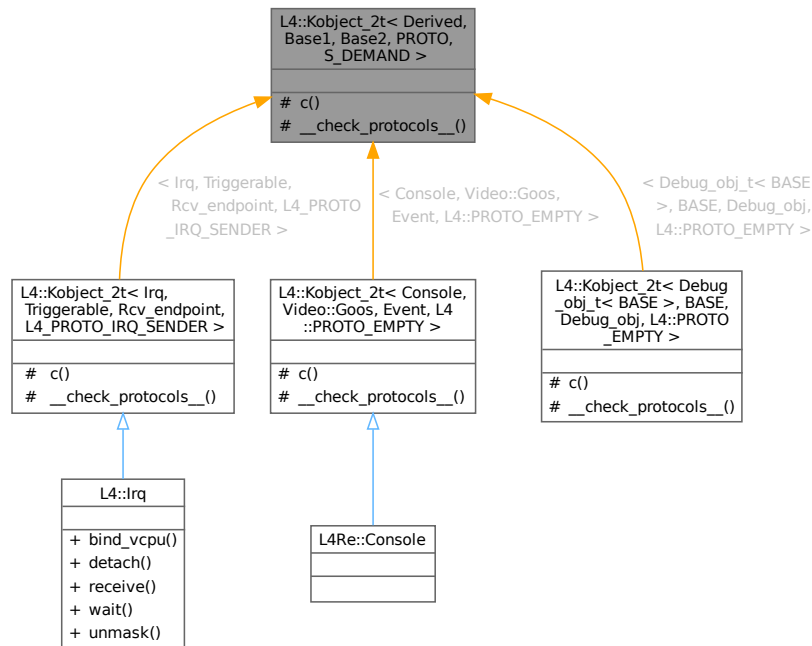
- [l4/sys/kobject](#)

## 16.179 L4::Kobject\_2t< Derived, Base1, Base2, PROTO, S\_DEMAND > Class Template Reference

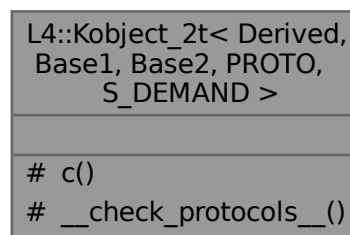
Helper class to create an [L4Re](#) interface class that is derived from two base classes (see [L4::Kobject\\_t](#)).

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Kobject\_2t< Derived, Base1, Base2, PROTO, S\_DEMAND >:



Collaboration diagram for L4::Kobject\_2t< Derived, Base1, Base2, PROTO, S\_DEMAND >:



## Protected Types

- typedef Derived [Class](#)  
The target interface type (inheriting from [Kobject\\_t](#)).
- typedef Typeid::Iface< PROTO, Derived > [\\_\\_iface](#)  
The interface description for the derived class.
- typedef Typeid::Merge\_list< Typeid::Iface\_list< [\\_\\_iface](#) >, Typeid::Merge\_list< typename Base1::\_\_iface\_list, typename Base2::\_\_iface\_list > > [\\_\\_iface\\_list](#)  
The list of all RPC interfaces provided directly or through inheritance.

## Protected Member Functions

- [L4::Cap](#)< [Class](#) > [c](#) () const noexcept  
*Get the capability to ourselves.*

## Static Protected Member Functions

- static void [\\_\\_check\\_protocols\\_\\_](#) () noexcept  
*Helper to check for protocol conflicts.*

### 16.179.1 Detailed Description

```
template<typename Derived, typename Base1, typename Base2, long PROTO = PROTO_ANY, typename
S_DEMAND = Type_info::Demand_t<>>
class L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >
```

Helper class to create an [L4Re](#) interface class that is derived from two base classes (see [L4::Kobject\\_t](#)).

#### Template Parameters

<i>Derived</i>	is the name of the new interface.
<i>Base1</i>	is the name of the interface's first base class.
<i>Base2</i>	is the name of the interface's second base class.
<i>PROTO</i>	may be set to the statically assigned protocol number used to communicate with this interface.
<i>S_DEMAND</i>	type defining the demand of server-side resources for this interface, usually a <a href="#">L4::Type_info::Demand_t</a> . This value must describe the server-side resources needed by the interface itself, the resource demand of the base interfaces (Base1 and Base2) are automatically included.

The typical usage pattern is shown in the following code snippet. The semantics of this example is an interface [My\\_iface](#) that is derived from [L4::Icu](#) and [L4Re::Dataspace](#).

```
class My_iface : public L4::Kobject_2t<My_iface, L4::Icu, L4Re::Dataspace>
{
    ...
};
```

Definition at line [827](#) of file [\\_\\_typeinfo.h](#).

### 16.179.2 Member Typedef Documentation

#### 16.179.2.1 [\\_\\_Iface](#)

```
template<typename Derived, typename Base1, typename Base2, long PROTO = PROTO_ANY, typename
S_DEMAND = Type_info::Demand_t<>>
typedef Typeid::Iface<PROTO, Derived> L4::Kobject\_2t< Derived, Base1, Base2, PROTO, S_DEMAND
>::__Iface [protected]
```

The interface description for the derived class.

Definition at line [833](#) of file [\\_\\_typeinfo.h](#).

### 16.179.2.2 \_\_Iface\_list

```
template<typename Derived, typename Base1, typename Base2, long PROTO = PROTO_ANY, typename
S_DEMAND = Type_info::Demand_t<>>
typedef Typeid::Merge_list< Typeid::Iface_list<__Iface>, Typeid::Merge_list< typename Base1<
::__Iface_list, typename Base2::__Iface_list > > L4::Kobject_2t< Derived, Base1, Base2, PROTO,
S_DEMAND >::__Iface_list [protected]
```

The list of all RPC interfaces provided directly or through inheritance.

Definition at line 841 of file [\\_\\_typeinfo.h](#).

### 16.179.2.3 Class

```
template<typename Derived, typename Base1, typename Base2, long PROTO = PROTO_ANY, typename
S_DEMAND = Type_info::Demand_t<>>
typedef Derived L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >::Class [protected]
```

The target interface type (inheriting from [Kobject\\_t](#)).

Definition at line 831 of file [\\_\\_typeinfo.h](#).

## 16.179.3 Member Function Documentation

### 16.179.3.1 \_\_check\_protocols\_\_()

```
template<typename Derived, typename Base1, typename Base2, long PROTO = PROTO_ANY, typename
S_DEMAND = Type_info::Demand_t<>>
void L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >::__check_protocols__ () [inline],
[static], [protected], [noexcept]
```

Helper to check for protocol conflicts.

Definition at line 844 of file [\\_\\_typeinfo.h](#).

### 16.179.3.2 c()

```
template<typename Derived, typename Base1, typename Base2, long PROTO = PROTO_ANY, typename
S_DEMAND = Type_info::Demand_t<>>
L4::Cap< Class > L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >::c () const [inline],
[protected], [noexcept]
```

Get the capability to ourselves.

Definition at line 863 of file [\\_\\_typeinfo.h](#).

The documentation for this class was generated from the following file:

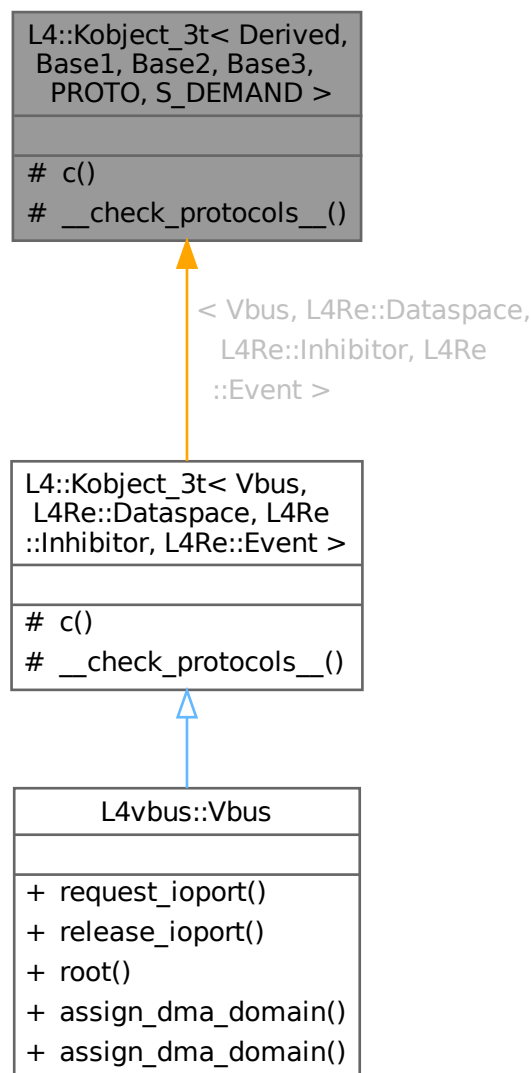
- [l4/sys/\\_\\_typeinfo.h](#)

## 16.180 L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO, S\_DEMAND > Struct Template Reference

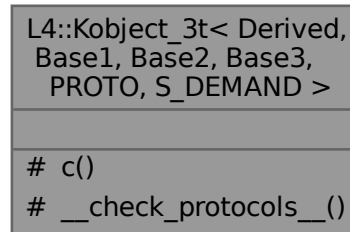
Helper class to create an [L4Re](#) interface class that is derived from three base classes (see [L4::Kobject\\_t](#)).

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO, S\_DEMAND >:



Collaboration diagram for L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO, S\_DEMAND >:



### Protected Types

- typedef Derived [Class](#)  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< PROTO, Derived > [\\_\\_iface](#)  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< [\\_\\_iface](#) >, Typeid::Merge\_list< typename Base1::\_\_iface\_list, Typeid::Merge\_list< typename Base2::\_\_iface\_list, typename Base3::\_\_iface\_list > > > [\\_\\_iface\\_list](#)  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Member Functions

- [L4::Cap](#)< [Class](#) > [c](#) () const noexcept  
*Get the capability to ourselves.*

### Static Protected Member Functions

- static void [\\_\\_check\\_protocols\\_\\_](#) () noexcept  
*Helper to check for protocol conflicts.*

## 16.180.1 Detailed Description

```
template<typename Derived, typename Base1, typename Base2, typename Base3, long PROTO = PROTO_
_ANY, typename S_DEMAND = Type_info::Demand_t<>>
struct L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >
```

Helper class to create an [L4Re](#) interface class that is derived from three base classes (see [L4 : Kobject\\_t](#)).

### Template Parameters

<i>Derived</i>	is the name of the new interface.
----------------	-----------------------------------

<i>Base1</i>	is the name of the interface's first base class.
<i>Base2</i>	is the name of the interface's second base class.
<i>Base3</i>	is the name of the interfaces third base class.
<i>PROTO</i>	may be set to the statically assigned protocol number used to communicate with this interface.
<i>S_DEMAND</i>	type defining the demand on server-side resources for this interface, usually a <a href="#">L4::Type_info::Demand_t</a> . This value must describe the server-side resources needed by the interface itself, the resource demand of the base interfaces (Base1 and Base2) are automatically included.

See also

[L4::Kobject\\_t](#), [L4::Kobject\\_2t](#), [L4::Kobject\\_0t](#), [L4::Kobject\\_x](#)

Definition at line 930 of file [\\_\\_typeinfo.h](#).

## 16.180.2 Member Typedef Documentation

### 16.180.2.1 \_\_Iface

```
template<typename Derived, typename Base1, typename Base2, typename Base3, long PROTO = PROTO↵
_ANY, typename S_DEMAND = Type_info::Demand_t<>>
typedef Typeid::Iface<PROTO, Derived> L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO,
S_DEMAND >::__Iface [protected]
```

The interface description for the derived class.

Definition at line 936 of file [\\_\\_typeinfo.h](#).

### 16.180.2.2 \_\_Iface\_list

```
template<typename Derived, typename Base1, typename Base2, typename Base3, long PROTO = PROTO↵
_ANY, typename S_DEMAND = Type_info::Demand_t<>>
typedef Typeid::Merge_list< Typeid::Iface_list<\_\_Iface>, Typeid::Merge_list< typename Base1↵
::__Iface_list, Typeid::Merge_list< typename Base2::__Iface_list, typename Base3::__Iface↵
_list > > > L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >::__Iface_list
[protected]
```

The list of all RPC interfaces provided directly or through inheritance.

Definition at line 947 of file [\\_\\_typeinfo.h](#).

### 16.180.2.3 Class

```
template<typename Derived, typename Base1, typename Base2, typename Base3, long PROTO = PROTO↵
_ANY, typename S_DEMAND = Type_info::Demand_t<>>
typedef Derived L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >::Class [protected]
```

The target interface type (inheriting from [Kobject\\_t](#)).

Definition at line 934 of file [\\_\\_typeinfo.h](#).



### 16.180.3 Member Function Documentation

#### 16.180.3.1 \_\_check\_protocols\_\_()

```
template<typename Derived, typename Base1, typename Base2, typename Base3, long PROTO = PROTO←
_ANY, typename S_DEMAND = Type_info::Demand_t<>>
void L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >::__check_protocols__ ()
[inline], [static], [protected], [noexcept]
```

Helper to check for protocol conflicts.

Definition at line 950 of file [\\_\\_typeinfo.h](#).

#### 16.180.3.2 c()

```
template<typename Derived, typename Base1, typename Base2, typename Base3, long PROTO = PROTO←
_ANY, typename S_DEMAND = Type_info::Demand_t<>>
L4::Cap< Class > L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >::c () const
[inline], [protected], [noexcept]
```

Get the capability to ourselves.

Definition at line 978 of file [\\_\\_typeinfo.h](#).

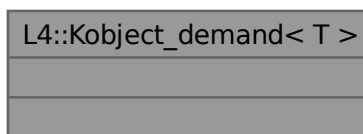
The documentation for this struct was generated from the following file:

- [l4/sys/\\_\\_typeinfo.h](#)

## 16.181 L4::Kobject\_demand< T > Struct Template Reference

Get the combined server-side resource requirements for all type T...

Collaboration diagram for L4::Kobject\_demand< T >:



### 16.181.1 Detailed Description

```
template<typename ... T>
struct L4::Kobject_demand< T >
```

Get the combined server-side resource requirements for all type T...

#### Template Parameters

T	List of IPC interface types for which the combined server-side resource requirements shall be calculated.
---	-----------------------------------------------------------------------------------------------------------

Definition at line 1031 of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

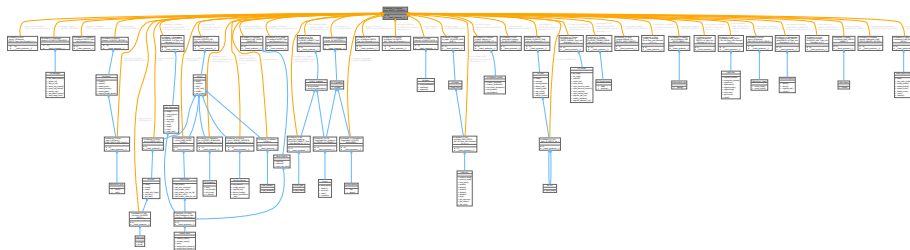
- l4/sys/[\\_\\_typeinfo.h](#)

## 16.182 L4::Kobject\_t< Derived, Base, PROTO, S\_DEMAND > Class Template Reference

Helper class to create an [L4Re](#) interface class that is derived from a single base class.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Kobject\_t< Derived, Base, PROTO, S\_DEMAND >:



Collaboration diagram for L4::Kobject\_t< Derived, Base, PROTO, S\_DEMAND >:

<b>L4::Kobject_t&lt; Derived, Base, PROTO, S_DEMAND &gt;</b>
<pre># c() # __check_protocols__()</pre>

### Protected Types

- typedef Derived **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< PROTO, Derived > **\_\_iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< [\\_\\_iface](#) >, typename Base::\_\_iface\_list > **\_\_iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Member Functions

- [L4::Cap](#)< [Class](#) > [c](#) () const noexcept  
*Get the capability to ourselves.*

## Static Protected Member Functions

- static void [\\_\\_check\\_protocols\\_\\_](#) () noexcept  
*Helper to check for protocol conflicts.*

## 16.182.1 Detailed Description

```
template<typename Derived, typename Base, long PROTO = PROTO_ANY, typename S_DEMAND = Type_↵
_info::Demand_t<>>
class L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >
```

Helper class to create an [L4Re](#) interface class that is derived from a single base class.

### Template Parameters

<i>Derived</i>	is the name of the new interface.
<i>Base</i>	is the name of the interfaces single base class.
<i>PROTO</i>	may be set to the statically assigned protocol number used to communicate with this interface.
<i>S_DEMAND</i>	type defining the demand on server-side resources for this interface, usually a <a href="#">L4::Type_info::Demand_t</a> . This value must describe the server-side resources needed by the interface itself, the resource demand of the base interface <a href="#">Base</a> is automatically included.

The typical usage pattern is shown in the following code snippet. The semantics of this example is an interface [My\\_iface](#) that is derived from [L4::Kobject](#).

```
class My_iface : public L4::Kobject_t<My_iface, L4::Kobject>
{
    ...
};
```

### Examples

[examples/clntsrv/src/shared.h](#).

Definition at line [749](#) of file [\\_\\_typeinfo.h](#).

The documentation for this class was generated from the following file:

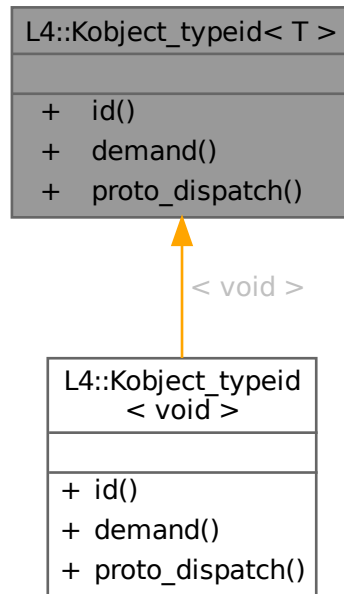
- [l4/sys/\\_\\_typeinfo.h](#)

## 16.183 L4::Kobject\_typeid< T > Struct Template Reference

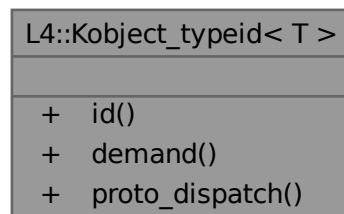
[Meta](#) object for handling access to type information of Kobjects.

```
#include <__typeinfo.h>
```

Inheritance diagram for L4::Kobject\_typeid< T >:



Collaboration diagram for L4::Kobject\_typeid< T >:



### Public Types

- typedef T::\_\_Kobject\_typeid::Demand [Demand](#)

*Data type expressing the static demand of receive buffers in a server.*

## Static Public Member Functions

- static [Type\\_info](#) const \* [id](#) () noexcept  
*Get a pointer to the [Kobject](#) type information of T.*
- static [Type\\_info::Demand](#) [demand](#) () noexcept  
*Get the receive-buffer demand for the server providing the interface T.*
- template<typename THIS, typename A1, typename A2>  
static int [proto\\_dispatch](#) (THIS \*self, long proto, A1 a1, A2 &a2)  
*Protocol based server-side dispatch function.*

### 16.183.1 Detailed Description

template<typename T>  
struct L4::Kobject\_typeid< T >

[Meta](#) object for handling access to type information of Kobjects.

#### Template Parameters

<a href="#">T</a>	The data type derived from <a href="#">Kobject</a> , usually using <a href="#">Kobject_t</a> .
-------------------	------------------------------------------------------------------------------------------------

Definition at line 610 of file [\\_\\_typeinfo.h](#).

### 16.183.2 Member Typedef Documentation

#### 16.183.2.1 Demand

```
template<typename T>
typedef T::__Kobject_typeid::Demand L4::Kobject\_typeid< T >::Demand
```

Data type expressing the static demand of receive buffers in a server.

This information is the combined demand of all base interfaces for T and the buffer demand of T itself. The buffer demand of T is usually specified as the S\_DEMAND argument of the [Kobject\\_t](#) or [Kobject\\_2t](#) inheritance helpers. S\_DEMAND is usually of type [L4::Type\\_info::Demand\\_t](#), or [L4::Type\\_info::Demand\\_union\\_t](#).

Definition at line 622 of file [\\_\\_typeinfo.h](#).

### 16.183.3 Member Function Documentation

#### 16.183.3.1 demand()

```
template<typename T>
Type\_info::Demand L4::Kobject\_typeid< T >::demand () [inline], [static], [noexcept]
```

Get the receive-buffer demand for the server providing the interface T.

#### Returns

A demand value describing the minimum receive buffers needed for handling server side requests for interface T.

Definition at line 639 of file [\\_\\_typeinfo.h](#).

### 16.183.3.2 id()

```
template<typename T>
Type_info const * L4::Kobject_typeid< T >::id () [inline], [static], [noexcept]
```

Get a pointer to the [Kobject](#) type information of T.

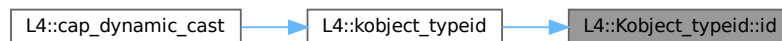
#### Returns

a pointer to the [Kobject](#) typeinfo of T.

Definition at line 630 of file [\\_\\_typeinfo.h](#).

Referenced by [L4::kobject\\_typeid\(\)](#).

Here is the caller graph for this function:



### 16.183.3.3 proto\_dispatch()

```
template<typename T>
template<typename THIS, typename A1, typename A2>
int L4::Kobject_typeid< T >::proto_dispatch (
    THIS * self,
    long proto,
    A1 a1,
    A2 & a2) [inline], [static]
```

Protocol based server-side dispatch function.

#### Template Parameters

<i>THIS</i>	Data type of the server-side object implementing the interface T.
<i>A1</i>	Data type of second argument for p_dispatch()
<i>A2</i>	Data type of third argument for p_dispatch()

#### Parameters

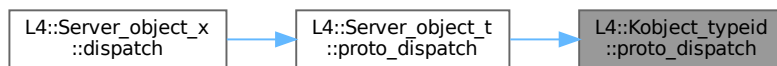
<i>self</i>	The pointer to the server object
<i>proto</i>	The protocol number used by the caller
<i>a1</i>	The second argument passed to self->p_dispatch()
<i>a2</i>	The third argument passed to self->p_dispatch()

This function forwards the call to the overloaded `p_dispatch()` function of `self`. The data type of the first argument for `p_dispatch` is determined by the given protocol number.

Definition at line 660 of file [\\_\\_typeinfo.h](#).

Referenced by [L4::Server\\_object\\_t< IFACE, BASE >::proto\\_dispatch\(\)](#).

Here is the caller graph for this function:



The documentation for this struct was generated from the following file:

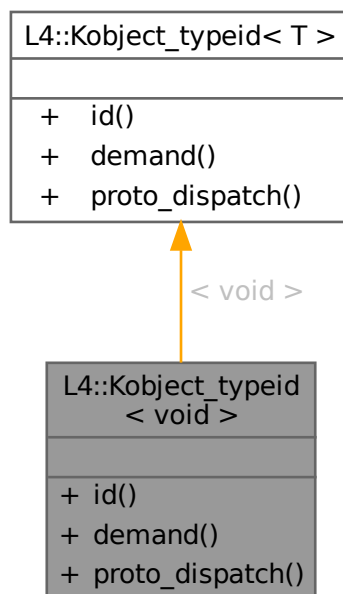
- [l4/sys/\\_\\_typeinfo.h](#)

## 16.184 L4::Kobject\_typeid< void > Struct Reference

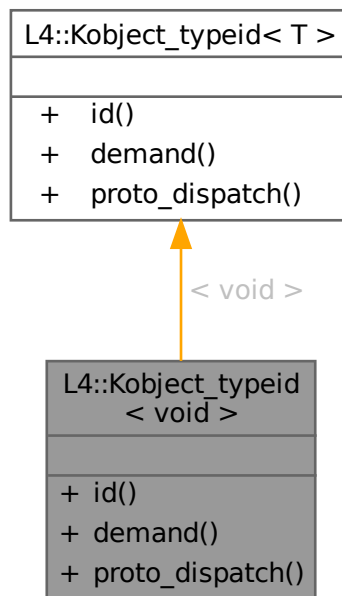
Minimalistic ID for `void` interface.

```
#include <__typeinfo.h>
```

Inheritance diagram for `L4::Kobject_typeid< void >`:



Collaboration diagram for L4::Kobject\_typeid< void >:



### Static Public Member Functions

- static `Type_info` const \* `id` () noexcept  
Get a pointer to teh *Kobject* type information of *T*.
- static `Type_info::Demand` `demand` () noexcept  
Get the receive-buffer demand for the server providing the interface *T*.
- static int `proto_dispatch` (THIS \*self, long proto, A1 a1, A2 &a2)  
Protocol based server-side dispatch function.

## 16.184.1 Detailed Description

Minimalistic ID for `void` interface.

Definition at line 667 of file `__typeinfo.h`.

## 16.184.2 Member Function Documentation

### 16.184.2.1 demand()

`Type_info::Demand L4::Kobject_typeid< void >::demand () [inline], [static], [noexcept]`

Get the receive-buffer demand for the server providing the interface *T*.

#### Returns

A demand value describing the minimum receive buffers needed for handling server side requests for interface *T*.

Definition at line 639 of file `__typeinfo.h`.



**16.184.2.2 id()**

```
Type_info const * L4::Kobject_typeid< void >::id () [inline], [static], [noexcept]
```

Get a pointer to the [Kobject](#) type information of T.

**Returns**

a pointer to the [Kobject](#) typeinfo of T.

Definition at line [630](#) of file [\\_\\_typeinfo.h](#).

**16.184.2.3 proto\_dispatch()**

```
int L4::Kobject_typeid< void >::proto_dispatch (
    THIS * self,
    long proto,
    A1 a1,
    A2 & a2) [inline], [static]
```

Protocol based server-side dispatch function.

**Template Parameters**

<i>THIS</i>	Data type of the server-side object implementing the interface T.
<i>A1</i>	Data type of second argument for p_dispatch()
<i>A2</i>	Data type of third argument for p_dispatch()

**Parameters**

<i>self</i>	The pointer to the server object
<i>proto</i>	The protocol number used by the caller
<i>a1</i>	The second argument passed to self->p_dispatch()
<i>a2</i>	The third argument passed to self->p_dispatch()

This function forwards the call to the overloaded p\_dispatch() function of self. The data type of the first argument for p\_dispatch is determined by the given protocol number.

Definition at line [660](#) of file [\\_\\_typeinfo.h](#).

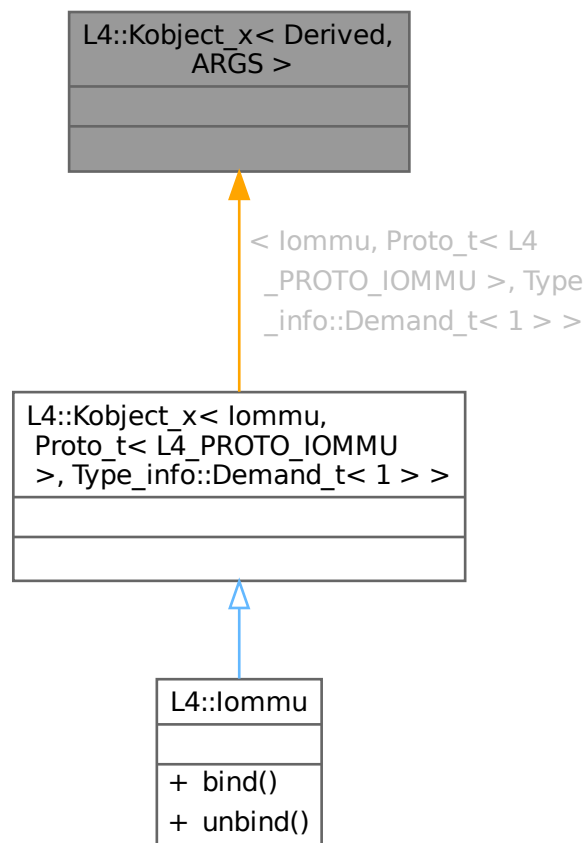
The documentation for this struct was generated from the following file:

- [l4/sys/\\_\\_typeinfo.h](#)

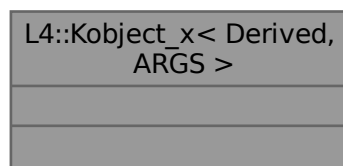
## 16.185 L4::Kobject\_x< Derived, ARGS > Struct Template Reference

Generic [Kobject](#) inheritance template.

Inheritance diagram for L4::Kobject\_x< Derived, ARGS >:



Collaboration diagram for L4::Kobject\_x< Derived, ARGS >:



### 16.185.1 Detailed Description

```
template<typename Derived, typename ... ARGS>
struct L4::Kobject_x< Derived, ARGS >
```

Generic [Kobject](#) inheritance template.

#### Template Parameters

<i>Derived</i>	The class name that derives from <a href="#">Kobject_x</a> .
<i>ARGS</i>	An optional protocol number via <a href="#">L4::Proto_t</a> , followed by an optional server-side requirement passed as <a href="#">L4::Type_info::Demand_t</a> , followed by the list of base classes.

Definition at line 1197 of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

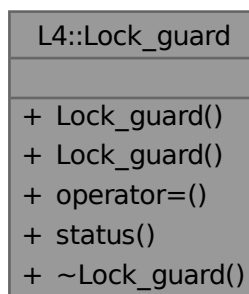
- [l4/sys/\\_\\_typeinfo.h](#)

## 16.186 L4::Lock\_guard Class Reference

Basic lock guard implementation that prevents forgotten unlocks on exit paths from a method or a block of code.

```
#include <lock_guard.h>
```

Collaboration diagram for L4::Lock\_guard:



#### Public Member Functions

- [Lock\\_guard](#) (pthread\_mutex\_t &lock)  
*Construct the lock guard and lock the associated mutex.*
- [Lock\\_guard](#) ([Lock\\_guard](#) &&guard)  
*Move constructor from other lock guard.*
- [Lock\\_guard](#) & [operator=](#) ([Lock\\_guard](#) &&guard)  
*Move assignment from other lock guard.*
- int [status](#) () const  
*Return last lock/unlock operation error status.*
- [~Lock\\_guard](#) ()  
*Lock guard destructor.*

### 16.186.1 Detailed Description

Basic lock guard implementation that prevents forgotten unlocks on exit paths from a method or a block of code.

Targeting `pthread_mutex_t`.

An instance of lock guard cannot be copied, but it can be moved.

The typical usage pattern of the lock guard is:

```
pthread_mutex_t mtx = PTHREAD_MUTEX_INITIALIZER;

{
    auto guard = L4Re::Lock_guard(mtx);

    // Correctness check.
    assert(guard.status() == 0);

    // Critical section protected by mtx.

    // The mtx is automatically unlocked when guard goes out of scope.
}
```

Definition at line 44 of file [lock\\_guard.h](#).

### 16.186.2 Constructor & Destructor Documentation

#### 16.186.2.1 Lock\_guard() [1/2]

```
L4::Lock_guard::Lock_guard (
    pthread_mutex_t & lock) [inline], [explicit]
```

Construct the lock guard and lock the associated mutex.

The error condition of the locking operation can be checked by the [status\(\)](#) method.

#### Parameters

<i>lock</i>	Associated mutex to be locked.
-------------	--------------------------------

Definition at line 59 of file [lock\\_guard.h](#).

#### 16.186.2.2 Lock\_guard() [2/2]

```
L4::Lock_guard::Lock_guard (
    Lock_guard && guard) [inline]
```

Move constructor from other lock guard.

The mutex associated with the other lock guard is kept locked.

#### Parameters

<i>guard</i>	Lock guard to be moved.
--------------	-------------------------

Definition at line 71 of file [lock\\_guard.h](#).

### 16.186.2.3 ~Lock\_guard()

```
L4::Lock_guard::~~Lock_guard () [inline]
```

Lock guard destructor.

The associated mutex (if any) is unlocked.

There is no mechanism for indicating any error conditions. However, if the mutex has been previously locked successfully by this class and if the implementation of the mutex behaves according to the POSIX specification, the construction of this class guarantees that the unlock operation does not fail.

Definition at line 126 of file [lock\\_guard.h](#).

## 16.186.3 Member Function Documentation

### 16.186.3.1 operator=()

```
Lock_guard & L4::Lock_guard::operator= (
    Lock_guard && guard) [inline]
```

Move assignment from other lock guard.

The mutex currently associated with this lock guard is unlocked. The mutex associated with the other lock guard is kept locked.

There is no mechanism for indicating any error conditions of the unlocking operation. However, if the mutex has been previously locked successfully by this class and if the implementation of the mutex behaves according to the POSIX specification, the construction of this class guarantees that the unlock operation does not fail.

#### Parameters

<i>guard</i>	Lock guard to be moved.
--------------	-------------------------

Definition at line 90 of file [lock\\_guard.h](#).

### 16.186.3.2 status()

```
int L4::Lock_guard::status () const [inline]
```

Return last lock/unlock operation error status.

#### Returns

Zero indicating no errors, any other value indicating an error.

Definition at line 110 of file [lock\\_guard.h](#).

The documentation for this class was generated from the following file:

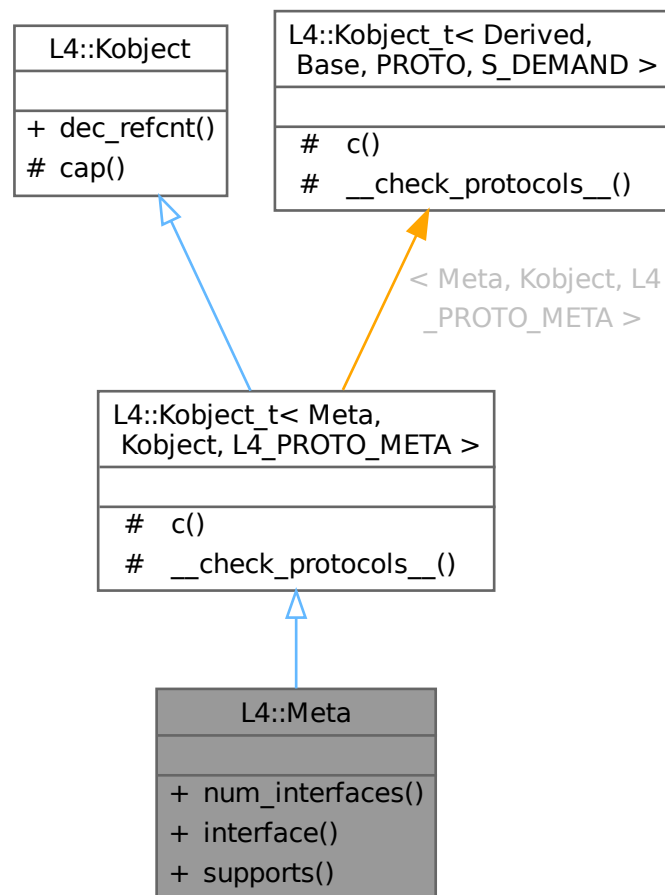
- [l4/cxx/lock\\_guard.h](#)

## 16.187 L4::Meta Class Reference

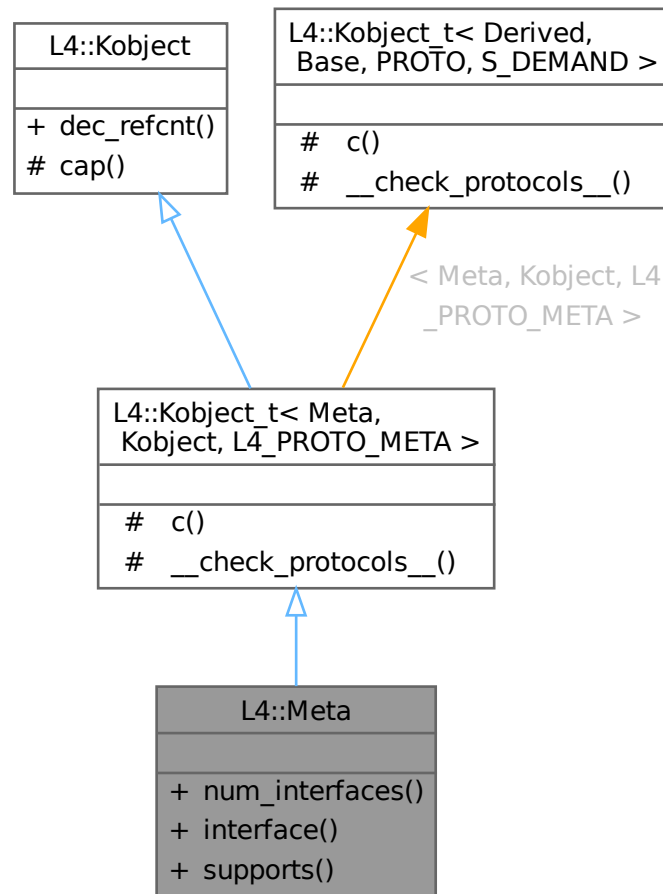
[Meta](#) interface that shall be implemented by each [L4Re](#) object and gives access to the dynamic type information for [L4Re](#) objects.

```
#include <meta>
```

Inheritance diagram for L4::Meta:



Collaboration diagram for L4::Meta:



## Public Member Functions

- [l4\\_msgtag\\_t num\\_interfaces \(\)](#)  
*Get the number of interfaces implemented by this object.*
- [l4\\_msgtag\\_t interface \(l4\\_umword\\_t idx, long \\*proto, L4::lpc::String< char > \\*name\)](#)  
*Get the protocol number that must be used for the interface with the number `idx`.*
- [l4\\_msgtag\\_t supports \(l4\\_mword\\_t protocol\)](#)  
*Figure out if the object supports the given protocol (number).*

## Public Member Functions inherited from [L4::Kobject](#)

- [l4\\_msgtag\\_t dec\\_refcnt \(l4\\_mword\\_t diff, l4\\_utcb\\_t \\*utcb=l4\\_utcb\(\)\)](#)  
*Decrement the in kernel reference counter for the object.*

## Additional Inherited Members

### Protected Types inherited from [L4::Kobject\\_t< Meta, Kobject, L4\\_PROTO\\_META >](#)

- typedef [Meta](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< PROTO, [Meta](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename Kobject::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Member Functions inherited from [L4::Kobject\\_t< Meta, Kobject, L4\\_PROTO\\_META >](#)

- [L4::Cap< Class > c](#) () const noexcept  
*Get the capability to ourselves.*

### Protected Member Functions inherited from [L4::Kobject](#)

- [l4\\_cap\\_idx\\_t cap](#) () const noexcept  
*Return capability selector.*

### Static Protected Member Functions inherited from [L4::Kobject\\_t< Meta, Kobject, L4\\_PROTO\\_META >](#)

- static void **\_\_check\_protocols\_\_** () noexcept  
*Helper to check for protocol conflicts.*

## 16.187.1 Detailed Description

[Meta](#) interface that shall be implemented by each [L4Re](#) object and gives access to the dynamic type information for [L4Re](#) objects.

Definition at line 26 of file [meta](#).

## 16.187.2 Member Function Documentation

### 16.187.2.1 interface()

```
l4_msgtag_t L4::Meta::interface (
    l4_umword_t idx,
    long * proto,
    L4::Ipc::String< char > * name)
```

Get the protocol number that must be used for the interface with the number `idx`.

#### Parameters



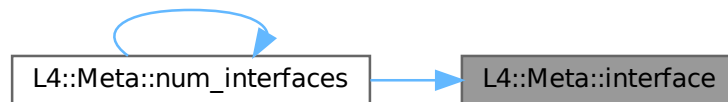
	<i>idx</i>	The index of the interface to get the protocol number for. <i>idx</i> must be $\geq 0$ and $<$ the return value of <a href="#">num_interfaces()</a> .
out	<i>proto</i>	The protocol number for interface <i>idx</i> .
out	<i>name</i>	The protocol name for interface <i>idx</i> .

**Return values**

<i>l4_msgtag_t::label()</i> == 0	Successful; see `proto` and `name`.
<i>l4_msgtag_t::label()</i> < 0	Error code.

Referenced by [num\\_interfaces\(\)](#).

Here is the caller graph for this function:

**16.187.2.2 num\_interfaces()**

```
l4_msgtag_t L4::Meta::num_interfaces ()
```

Get the number of interfaces implemented by this object.

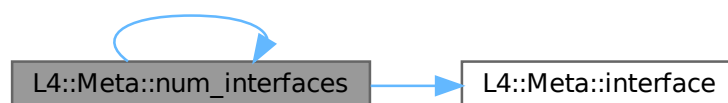
**Return values**

<i>l4_msgtag_t::label()</i> $\geq 0$	The number of supported interfaces.
<i>l4_msgtag_t::label()</i> < 0	Error code of the occurred error.

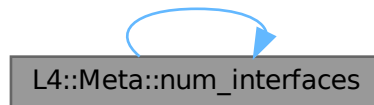
References [interface\(\)](#), and [num\\_interfaces\(\)](#).

Referenced by [num\\_interfaces\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.187.2.3 supports()

```
l4_msgtag_t L4::Meta::supports (
    l4_mword_t protocol)
```

Figure out if the object supports the given protocol (number).

#### Parameters

<i>protocol</i>	The protocol number to check for.
-----------------	-----------------------------------

#### Return values

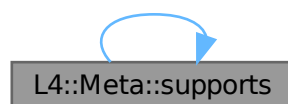
<i>l4_msgtag_t::label()</i> == 1	protocol is supported.
<i>l4_msgtag_t::label()</i> == 0	protocol is not supported.

This method is intended to be used for statically assigned protocol numbers.

References [supports\(\)](#).

Referenced by [supports\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

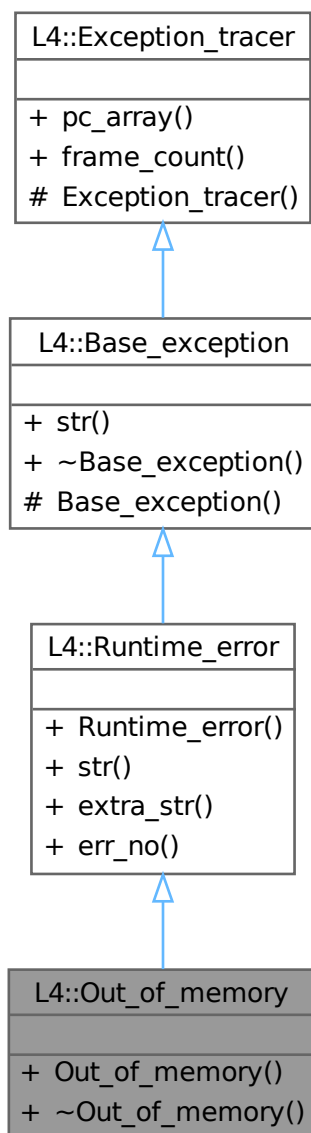
- [l4/sys/meta](#)

## 16.188 L4::Out\_of\_memory Class Reference

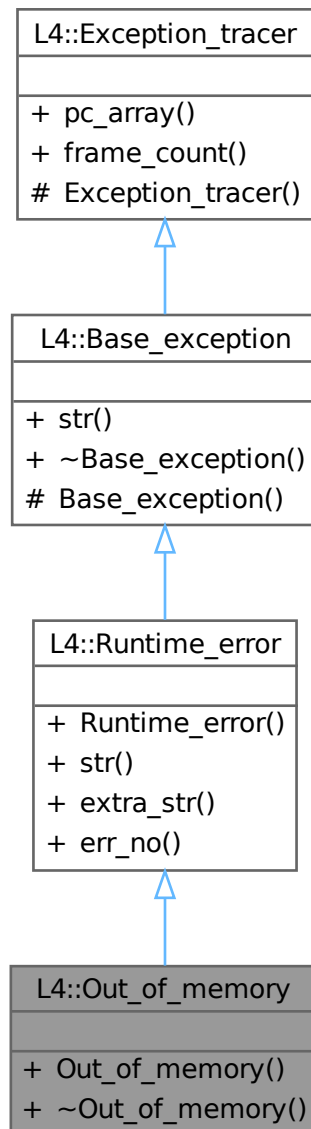
[Exception](#) signalling insufficient memory.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Out\_of\_memory:



Collaboration diagram for L4::Out\_of\_memory:



### Public Member Functions

- **Out\_of\_memory** (char const \*extra="") noexcept  
*Create an out-of-memory exception.*
- **~Out\_of\_memory** () noexcept  
*Destruction.*

### Public Member Functions inherited from [L4::Runtime\\_error](#)

- [Runtime\\_error](#) (long err\_no, char const \*extra=0) noexcept

Create a new [Runtime\\_error](#).

- char const \* **str** () const noexcept override  
Return a human readable string for the exception.
- char const \* **extra\_str** () const  
Get the description text for this runtime error.
- long **err\_no** () const noexcept  
Get the error value for this runtime error.

## Public Member Functions inherited from [L4::Base\\_exception](#)

- virtual ~**Base\_exception** () noexcept  
Destruction.

## Public Member Functions inherited from [L4::Exception\\_tracer](#)

- void const \*const \* **pc\_array** () const noexcept  
Get the array containing the call trace.
- int **frame\_count** () const noexcept  
Get the number of entries that are valid in the call trace.

## Additional Inherited Members

## Protected Member Functions inherited from [L4::Base\\_exception](#)

- **Base\_exception** () noexcept  
Create a base exception.

## Protected Member Functions inherited from [L4::Exception\\_tracer](#)

- **Exception\_tracer** () noexcept  
Create a back trace.

## 16.188.1 Detailed Description

[Exception](#) signalling insufficient memory.

Definition at line 177 of file [exceptions](#).

The documentation for this class was generated from the following file:

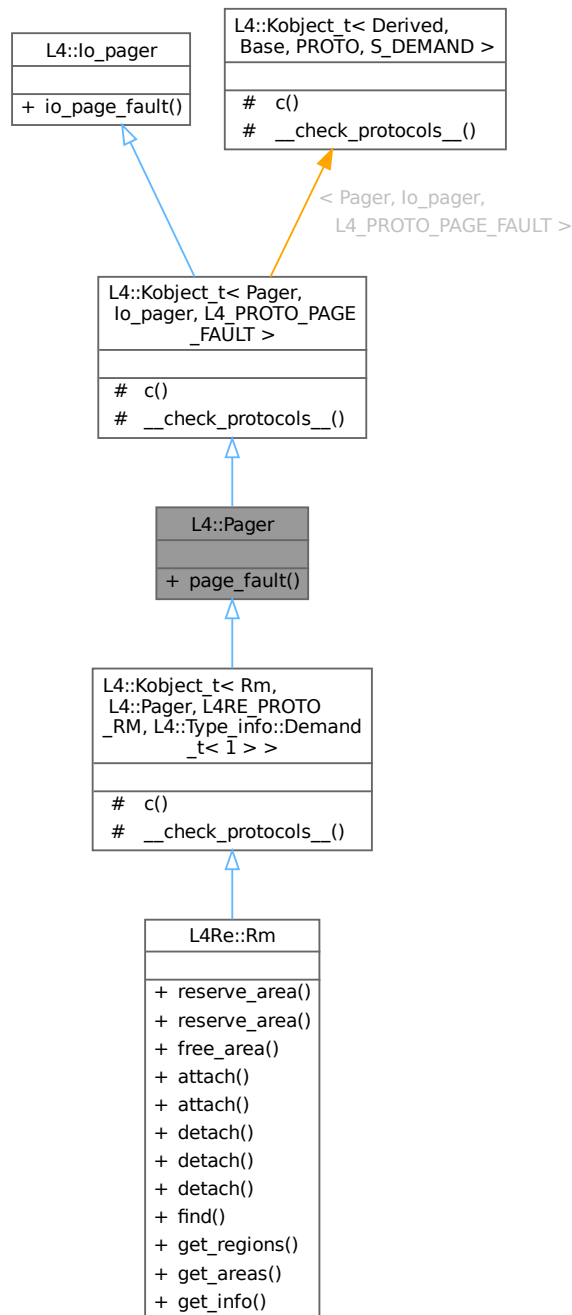
- [l4/cxx/exceptions](#)

## 16.189 L4::Pager Class Reference

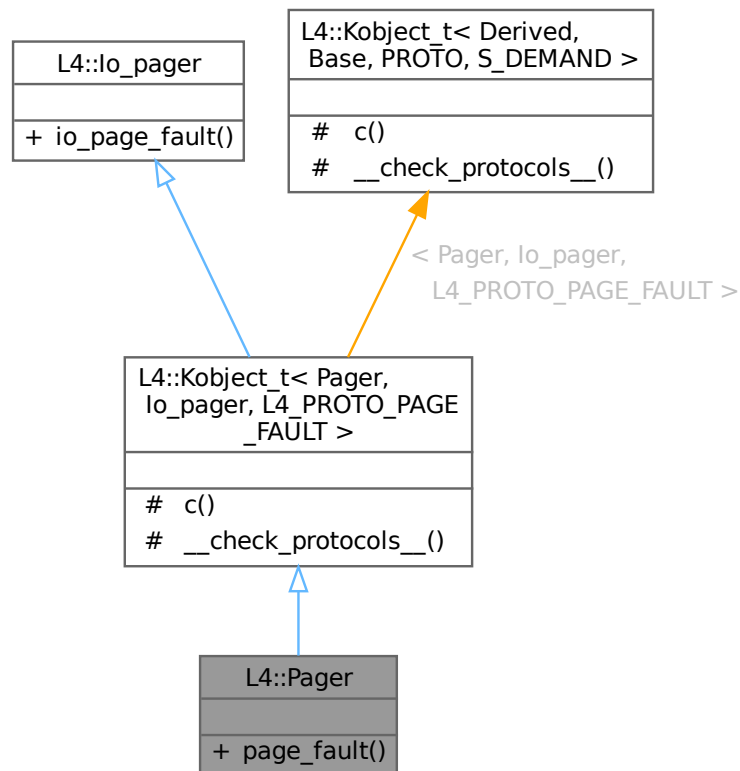
[Pager](#) interface including the [io\\_pager](#) interface.

```
#include <pager>
```

Inheritance diagram for L4::Pager:



Collaboration diagram for L4::Pager:



### Public Member Functions

- [l4\\_msgtag\\_t page\\_fault](#) ([l4\\_umword\\_t](#) pfa, [l4\\_umword\\_t](#) pc, [L4::lpc::Rcv\\_fpage](#) rwin, [L4::lpc::Opt<L4::lpc::Snd\\_fpage & >](#) fp)  
*Page-fault protocol message.*

### Public Member Functions inherited from [L4::lo\\_pager](#)

- [l4\\_msgtag\\_t io\\_page\\_fault](#) ([l4\\_fpage\\_t](#) io\_pfa, [l4\\_umword\\_t](#) pc, [L4::lpc::Rcv\\_fpage](#) rwin, [L4::lpc::Opt<L4::lpc::Snd\\_fpage & >](#) fp)  
*IO page fault protocol message.*

### Additional Inherited Members

### Protected Types inherited from

[L4::Kobject\\_t< Pager, lo\\_pager, L4\\_PROTO\\_PAGE\\_FAULT >](#)

- typedef [Pager](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef [Typeid::Iface< PROTO, Pager >](#) **\_\_Iface**  
*The interface description for the derived class.*
- typedef [Typeid::Merge\\_list< Typeid::Iface\\_list< \[\\\_\\\_Iface\]\(#\) >, typename lo\\_pager::\\_\\_Iface\\_list >](#) **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*



## Protected Member Functions inherited from [L4::Kobject\\_t< Pager, lo\\_pager, L4\\_PROTO\\_PAGE\\_FAULT >](#)

- [L4::Cap< Class > c \(\)](#) const noexcept  
*Get the capability to ourselves.*

## Static Protected Member Functions inherited from [L4::Kobject\\_t< Pager, lo\\_pager, L4\\_PROTO\\_PAGE\\_FAULT >](#)

- static void [\\_\\_check\\_protocols\\_\\_ \(\)](#) noexcept  
*Helper to check for protocol conflicts.*

### 16.189.1 Detailed Description

[Pager](#) interface including the [lo\\_pager](#) interface.

This class defines the interface for handling page fault IPC. If a thread causes a page fault, the microkernel synthesises a page fault IPC message and sends it to the thread's page fault handler (pager). The pager can then handle the message, for example by establishing a suitable page mapping.

The page fault handler is set with the [L4::Thread::control](#) interface.

Definition at line 87 of file [pager](#).

### 16.189.2 Member Function Documentation

#### 16.189.2.1 [page\\_fault\(\)](#)

```
l4_msgtag_t L4::Pager::page_fault (
    l4_umword_t pfa,
    l4_umword_t pc,
    L4::Ipc::Rcv_fpage rwin,
    L4::Ipc::Opt< L4::Ipc::Snd_fpage & > fp)
```

Page-fault protocol message.

#### Parameters

	<i>pfa</i>	Faulting address including failure reason: bits [0:2].
	<i>pc</i>	Faulting program counter.
	<i>rwin</i>	Receive window for a flexpage mapping resolving the page fault.
out	<i>fp</i>	Optional: flexpage descriptor to send to the task raising the page fault.

**Returns**

System call message tag; use [l4\\_error\(\)](#) to check for errors.

Page-fault messages are usually generated by the kernel and need to be handled by an appropriate handler function, potentially filling in `fp` for the reply.

`pfa` encoding is as shown:

[63/31 .. 3]	2	1	0
PFA	X	W	r

- **PFA** Bits 63/31..3 of `pfa` are the page fault address bits 63/31 to 3, bits 2..0 are masked.
- **X** Bit 2 of `pfa` if set, indicates a page fault during instruction fetch. Note, this bit is implementation-defined and might always be clear. Therefore, if this bit is clear it does not imply that the page fault is not due to an instruction fetch.
- **W** Bit 1 of `pfa` is set to 1 for a page fault due to a write operation.
- **r** Bit0: reserved, undefined.

The documentation for this class was generated from the following file:

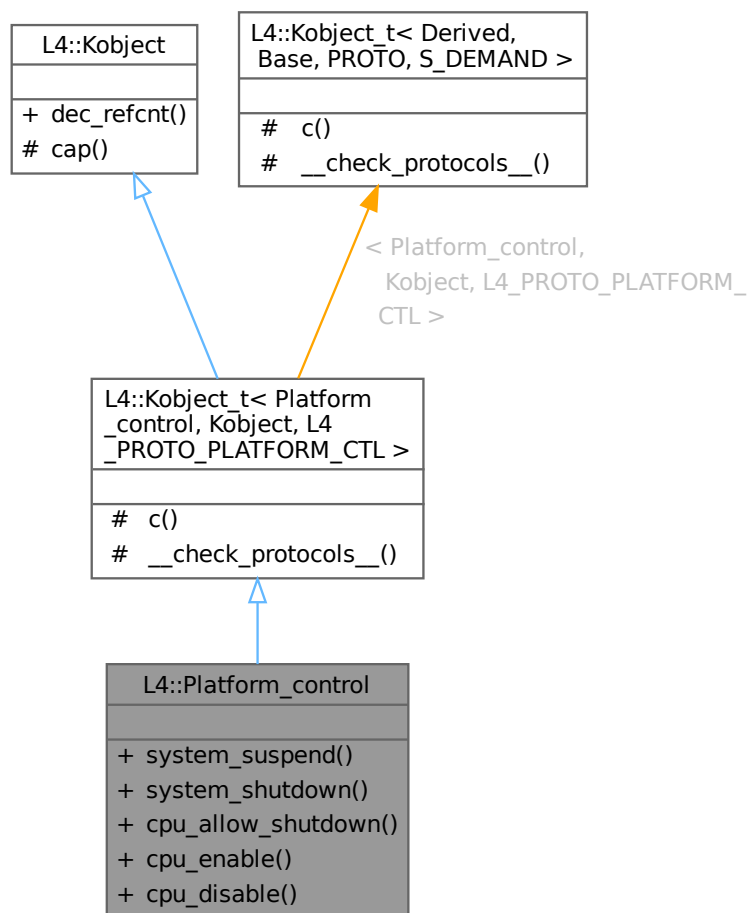
- [l4/sys/pager](#)

## 16.190 L4::Platform\_control Class Reference

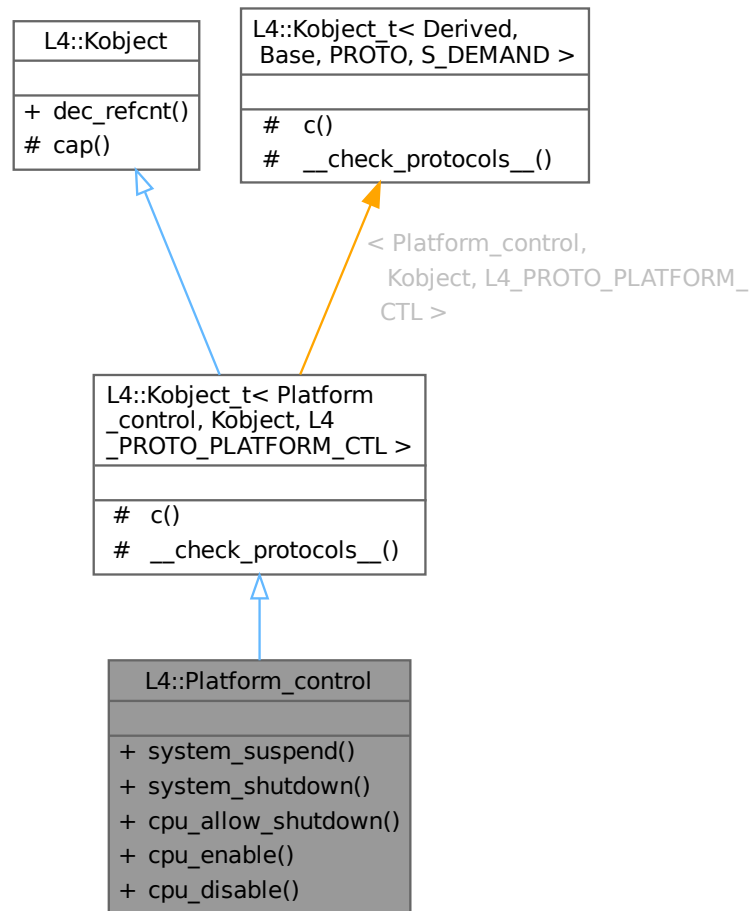
[L4](#) C++ interface for controlling platform-wide properties, see [Platform Control C API](#) for the C interface.

```
#include <platform_control>
```

Inheritance diagram for L4::Platform\_control:



Collaboration diagram for L4::Platform\_control:



## Public Member Functions

- [l4\\_msgtag\\_t system\\_suspend \(l4\\_umword\\_t extras\)](#)  
*Enter suspend to RAM.*
- [l4\\_msgtag\\_t system\\_shutdown \(l4\\_umword\\_t reboot\)](#)  
*Shutdown/Reboot the system.*
- [l4\\_msgtag\\_t cpu\\_allow\\_shutdown \(l4\\_umword\\_t phys\\_id, l4\\_umword\\_t enable\)](#)  
*Allow CPU shutdown.*
- [l4\\_msgtag\\_t cpu\\_enable \(l4\\_umword\\_t phys\\_id\)](#)  
*Enable an offline CPU.*
- [l4\\_msgtag\\_t cpu\\_disable \(l4\\_umword\\_t phys\\_id\)](#)  
*Disable an online CPU.*

## Public Member Functions inherited from L4::Kobject

- [l4\\_msgtag\\_t dec\\_refcnt \(l4\\_mword\\_t diff, l4\\_utcb\\_t \\*utcb=l4\\_utcb\(\)\)](#)  
*Decrement the in kernel reference counter for the object.*

## Additional Inherited Members

### Protected Types inherited from

[L4::Kobject\\_t< Platform\\_control, Kobject, L4\\_PROTO\\_PLATFORM\\_CTL >](#)

- typedef [Platform\\_control](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< [PROTO](#), [Platform\\_control](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< [\\_\\_Iface](#) >, typename Kobject::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Member Functions inherited from

[L4::Kobject\\_t< Platform\\_control, Kobject, L4\\_PROTO\\_PLATFORM\\_CTL >](#)

- [L4::Cap< Class > c](#) () const noexcept  
*Get the capability to ourselves.*

### Protected Member Functions inherited from [L4::Kobject](#)

- [l4\\_cap\\_idx\\_t](#) [cap](#) () const noexcept  
*Return capability selector.*

### Static Protected Member Functions inherited from

[L4::Kobject\\_t< Platform\\_control, Kobject, L4\\_PROTO\\_PLATFORM\\_CTL >](#)

- static void [\\_\\_check\\_protocols\\_\\_](#) () noexcept  
*Helper to check for protocol conflicts.*

## 16.190.1 Detailed Description

[L4](#) C++ interface for controlling platform-wide properties, see [Platform Control C API](#) for the C interface.

Add

```
#include <l4/sys/platform_control>
```

to your code to use the platform control functions. The API allows a client to suspend, reboot or shutdown the system.

For the C interface refer to the [Platform Control C API](#).

Definition at line 36 of file [platform\\_control](#).

## 16.190.2 Member Function Documentation

### 16.190.2.1 [cpu\\_allow\\_shutdown\(\)](#)

```
l4\_msgtag\_t L4::Platform_control::cpu_allow_shutdown (
    l4\_umword\_t phys_id,
    l4\_umword\_t enable)
```

Allow CPU shutdown.

#### Parameters

---

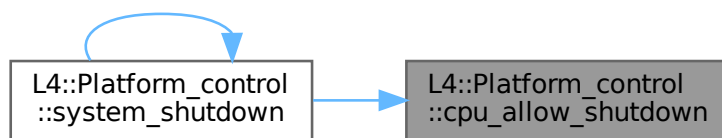
<i>phys↔ _id</i>	Physical CPU id of CPU (e.g. local APIC id) to disable.
<i>enable</i>	Allow shutdown when 1, disallow when 0.

Sets or unsets a hint that a CPU that is not currently used may be powered down.

References [L4\\_PLATFORM\\_CTL\\_CPU\\_ENABLE\\_OP](#).

Referenced by [system\\_shutdown\(\)](#).

Here is the caller graph for this function:



### 16.190.2.2 `cpu_disable()`

```
l4_msgtag_t L4::Platform_control::cpu_disable (
    l4_umword_t phys_id)
```

Disable an online CPU.

#### Parameters

<i>phys↔ _id</i>	Physical CPU id of CPU (e.g. local APIC id) to disable.
----------------------	---------------------------------------------------------

#### Returns

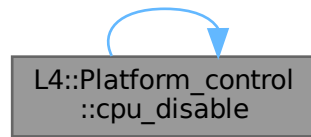
System call message tag

This function is currently only supported on the ARM EXYNOS platform.

References [cpu\\_disable\(\)](#).

Referenced by [cpu\\_disable\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.190.2.3 `cpu_enable()`

```
l4_msgtag_t L4::Platform_control::cpu_enable (  
    l4_umword_t phys_id)
```

Enable an offline CPU.

#### Parameters

<i>phys_id</i>	Physical CPU id of CPU (e.g. local APIC id) to enable.
----------------	--------------------------------------------------------

#### Returns

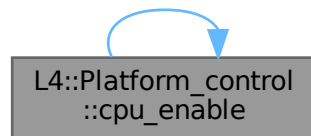
System call message tag

This function is currently only supported on the ARM EXYNOS platform.

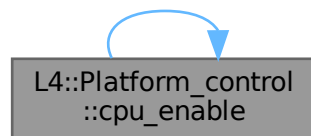
References [cpu\\_enable\(\)](#), and [L4\\_PLATFORM\\_CTL\\_CPU\\_DISABLE\\_OP](#).

Referenced by [cpu\\_enable\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 16.190.2.4 system\_shutdown()

```
l4_msgtag_t L4::Platform_control::system_shutdown (
    l4_umword_t reboot)
```

Shutdown/Reboot the system.

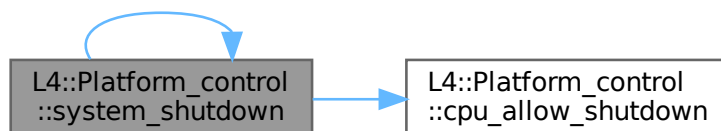
##### Parameters

<i>reboot</i>	1 for reboot, 0 for power off
---------------	-------------------------------

References [cpu\\_allow\\_shutdown\(\)](#), [L4\\_PLATFORM\\_CTL\\_CPU\\_ALLOW\\_SHUTDOWN\\_OP](#), and [system\\_shutdown\(\)](#).

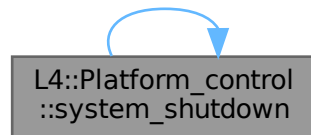
Referenced by [system\\_shutdown\(\)](#).

Here is the call graph for this function:





Here is the caller graph for this function:



### 16.190.2.5 system\_suspend()

```
l4_msgtag_t L4::Platform_control::system_suspend (
    l4_umword_t extras)
```

Enter suspend to RAM.

#### Precondition

Must only be invoked on the boot CPU. Furthermore it must be ensured that the invoking thread is not migrated to a different CPU during the suspend.

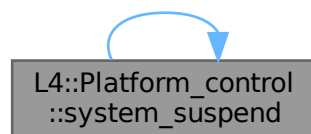
#### Parameters

<i>extras</i>	Some extra platform-specific information needed to enter suspend to RAM. On x86 platforms and when using the <a href="#">Platform_control</a> object provided by Fiasco, the value defines the sleep state. The sleep states are defined in the ACPI table. Other platforms as well as lo's <a href="#">Platform_control</a> object don't make use of this value at the moment.
---------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

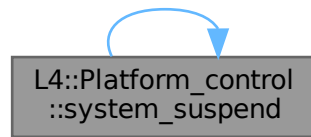
References [L4\\_PLATFORM\\_CTL\\_SYS\\_SHUTDOWN\\_OP](#), and [system\\_suspend\(\)](#).

Referenced by [system\\_suspend\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

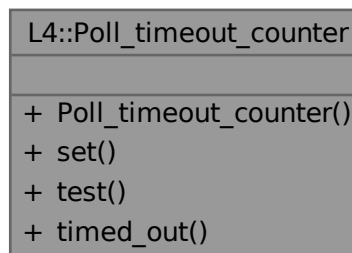
- [l4/sys/platform\\_control](#)

## 16.191 L4::Poll\_timeout\_counter Class Reference

Evaluate an expression for a maximum number of times.

```
#include <poll_timeout_counter.h>
```

Collaboration diagram for L4::Poll\_timeout\_counter:



### Public Member Functions

- [Poll\\_timeout\\_counter](#) (unsigned counter\_val)  
*Constructor.*
- void [set](#) (unsigned counter\_val)  
*Set the counter to a certain value.*
- bool **test** (bool expression=true)  
*Evaluate the expression for a maximum number of times.*
- bool [timed\\_out](#) () const  
*Indicator if the maximum number of tests was required.*

### 16.191.1 Detailed Description

Evaluate an expression for a maximum number of times.

A typical use case is testing for a bit change in a hardware register for a maximum number of times (polling). For example:

```
Mmio_register_block regs;
Poll_timeout_counter i(3000000);
while (i.test(!(regs.read<uint32_t>(0x04) & 1)))
;
```

The following usage is **wrong**:

```
...
Poll_timeout_counter i(3000000);
while (!i.test((regs.read<uint32_t>(0x04) & 1)))
;
```

This loop would never terminate if the hardware register doesn't change!

Definition at line 34 of file [poll\\_timeout\\_counter.h](#).

### 16.191.2 Constructor & Destructor Documentation

#### 16.191.2.1 Poll\_timeout\_counter()

```
L4::Poll_timeout_counter::Poll_timeout_counter (
    unsigned counter_val) [inline]
```

Constructor.

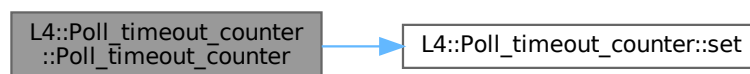
#### Parameters

<i>counter_val</i>	Maximum number of times to repeat the test.
--------------------	---------------------------------------------

Definition at line 42 of file [poll\\_timeout\\_counter.h](#).

References [set\(\)](#).

Here is the call graph for this function:



### 16.191.3 Member Function Documentation

#### 16.191.3.1 set()

```
void L4::Poll_timeout_counter::set (
    unsigned counter_val) [inline]
```

Set the counter to a certain value.

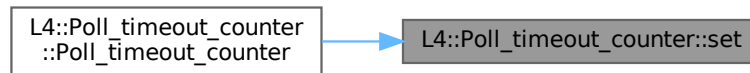
#### Parameters

<i>counter_val</i>	New counter value for maximum number of times to repeat the test.
--------------------	-------------------------------------------------------------------

Definition at line 53 of file [poll\\_timeout\\_counter.h](#).

Referenced by [Poll\\_timeout\\_counter\(\)](#).

Here is the caller graph for this function:



### 16.191.3.2 timed\_out()

```
bool L4::Poll_timeout_counter::timed_out () const [inline]
```

Indicator if the maximum number of tests was required.

#### Return values

<i>true, if</i>	the maximum number of tests was required or if the counter was initialized to zero.
-----------------	-------------------------------------------------------------------------------------

Definition at line 81 of file [poll\\_timeout\\_counter.h](#).

The documentation for this class was generated from the following file:

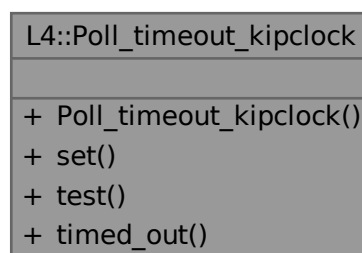
- pkg/drivers-frst/include/poll\_timeout\_counter.h

## 16.192 L4::Poll\_timeout\_kipclock Class Reference

A polling timeout based on the [L4Re](#) clock.

```
#include <poll_timeout_kipclock>
```

Collaboration diagram for L4::Poll\_timeout\_kipclock:



## Public Member Functions

- [Poll\\_timeout\\_kipclock](#) (unsigned poll\_time\_us)  
*Initialise relative timeout in microseconds.*
- void [set](#) (unsigned poll\_time\_us)  
*(Re-)Set relative timeout in microseconds*
- bool [test](#) (bool expression=true)  
*Test whether timeout has expired.*
- bool [timed\\_out](#) () const  
*Query whether timeout has expired.*

### 16.192.1 Detailed Description

A polling timeout based on the [L4Re](#) clock.

This class allows to conveniently add a timeout to a polling loop.

The original

```
while (device.read(State) & Busy)
;
```

is converted to

```
Poll_timeout_kipclock timeout(10000);
while (timeout.test(device.read(State) & Busy))
;
if (timeout.timed_out())
    printf("ERROR: Device does not respond.\n");
```

Definition at line 35 of file [poll\\_timeout\\_kipclock](#).

### 16.192.2 Constructor & Destructor Documentation

#### 16.192.2.1 Poll\_timeout\_kipclock()

```
L4::Poll_timeout_kipclock::Poll_timeout_kipclock (
    unsigned poll_time_us) [inline]
```

Initialise relative timeout in microseconds.

#### Parameters

<i>poll_time_us</i>	Polling timeout in microseconds.
---------------------	----------------------------------

Definition at line 42 of file [poll\\_timeout\\_kipclock](#).

References [set\(\)](#).

Here is the call graph for this function:



### 16.192.3 Member Function Documentation

#### 16.192.3.1 set()

```
void L4::Poll_timeout_kipclock::set (
    unsigned poll_time_us) [inline]
```

(Re-)Set relative timeout in microseconds

##### Parameters

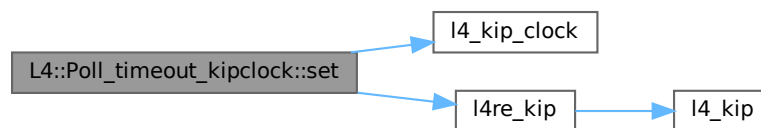
<i>poll_time_us</i>	Polling timeout in microseconds.
---------------------	----------------------------------

Definition at line 51 of file [poll\\_timeout\\_kipclock](#).

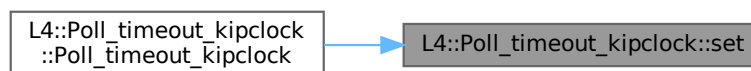
References [l4\\_kip\\_clock\(\)](#), and [l4re\\_kip\(\)](#).

Referenced by [Poll\\_timeout\\_kipclock\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 16.192.3.2 test()

```
bool L4::Poll_timeout_kipclock::test (
    bool expression = true) [inline]
```

Test whether timeout has expired.

##### Parameters

<i>expression</i>	Optional expression.
-------------------	----------------------

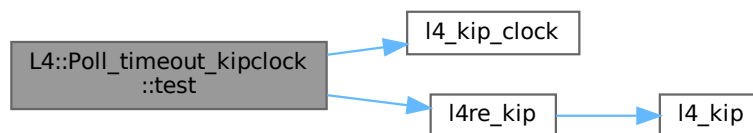
### Return values

<i>false</i>	The timeout has expired or the given expression returned false.
<i>true</i>	The timeout has not expired and the optionally given expression returns true.

Definition at line 65 of file [poll\\_timeout\\_kipclock](#).

References [l4\\_kip\\_clock\(\)](#), and [l4re\\_kip\(\)](#).

Here is the call graph for this function:



### 16.192.3.3 timed\_out()

```
bool L4::Poll_timeout_kipclock::timed_out () const [inline]
```

Query whether timeout has expired.

#### Returns

Expiry state of timeout

Definition at line 77 of file [poll\\_timeout\\_kipclock](#).

The documentation for this class was generated from the following file:

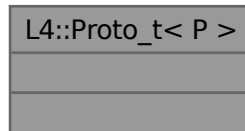
- `l4/re/util/poll_timeout_kipclock`

## 16.193 L4::Proto\_t< P > Struct Template Reference

Data type for defining protocol numbers.

```
#include <__typeinfo.h>
```

Collaboration diagram for L4::Proto\_t< P >:



### 16.193.1 Detailed Description

```
template<long P = PROTO_EMPTY>
struct L4::Proto_t< P >
```

Data type for defining protocol numbers.

#### Template Parameters

<i>P</i>	The protocol number itself
----------	----------------------------

This type must be used when specifying a protocol number with [Kobject\\_x](#).

Definition at line 1181 of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

- [l4/sys/\\_\\_typeinfo.h](#)

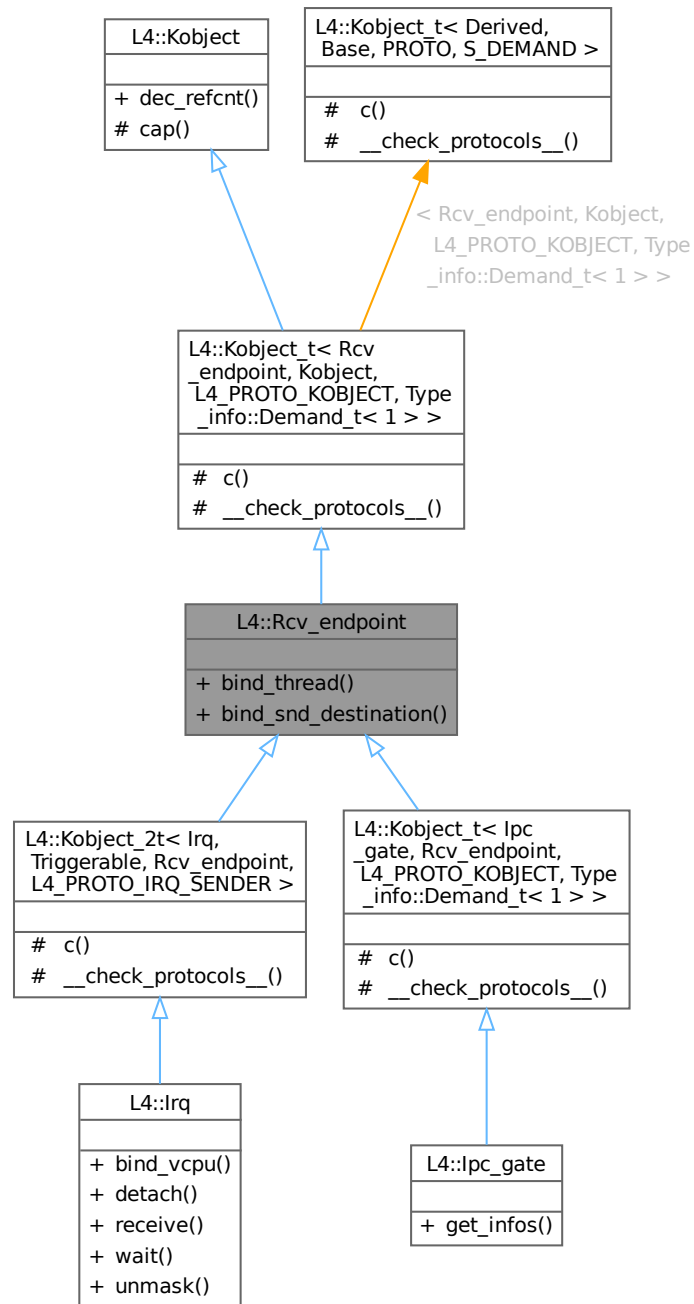
## 16.194 L4::Rcv\_endpoint Class Reference

Interface for kernel objects that allow to receive IPC from them.

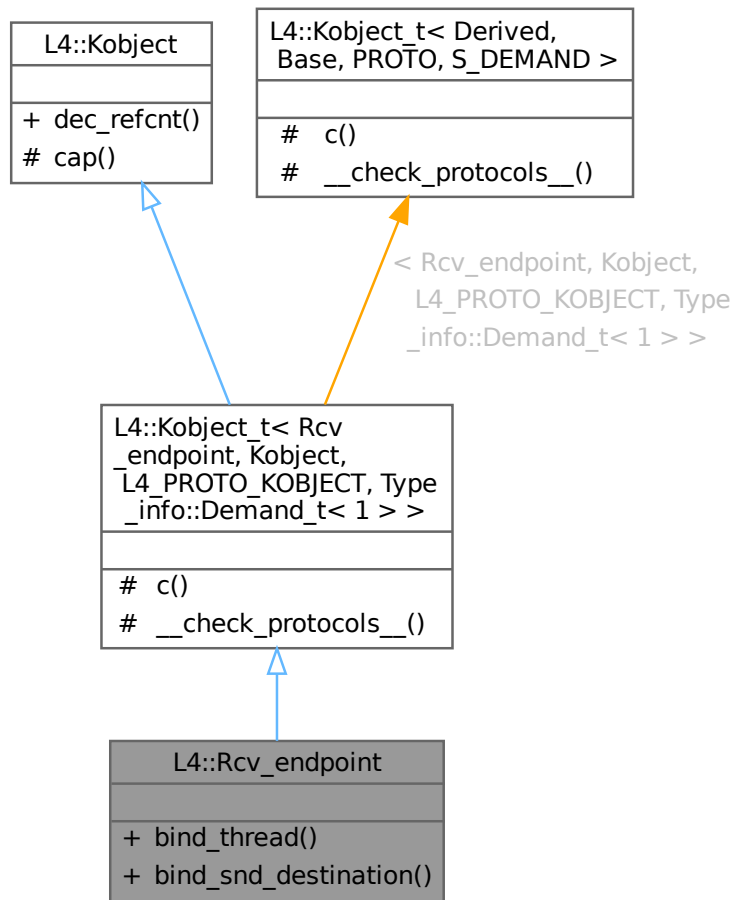
```
#include <rcv_endpoint>
```



Inheritance diagram for L4::Rcv\_endpoint:



Collaboration diagram for L4::Rcv\_endpoint:



## Public Member Functions

- `l4_msgtag_t bind_thread (lpc::Cap< Thread > t, l4_umword_t label)`  
Bind the IPC receive endpoint to a thread.
- `l4_msgtag_t bind_snd_destination (Cap< Snd_destination > snd_dst, l4_umword_t label)`  
Bind a send destination (a thread or thread group) to an IPC receive endpoint.

## Public Member Functions inherited from L4::Kobject

- `l4_msgtag_t dec_refcnt (l4_mword_t diff, l4_utcb_t *utcb=l4_utcb())`  
Decrement the in kernel reference counter for the object.

## Additional Inherited Members

### Protected Types inherited from

[L4::Kobject\\_t< Rcv\\_endpoint, Kobject, L4\\_PROTO\\_KOBJECT, Type\\_info::Demand\\_t< 1 > >](#)

- typedef [Rcv\\_endpoint](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< [PROTO](#), [Rcv\\_endpoint](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename Kobject::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Member Functions inherited from

[L4::Kobject\\_t< Rcv\\_endpoint, Kobject, L4\\_PROTO\\_KOBJECT, Type\\_info::Demand\\_t< 1 > >](#)

- [L4::Cap< Class > c](#) () const noexcept  
*Get the capability to ourselves.*

### Protected Member Functions inherited from [L4::Kobject](#)

- [l4\\_cap\\_idx\\_t cap](#) () const noexcept  
*Return capability selector.*

### Static Protected Member Functions inherited from

[L4::Kobject\\_t< Rcv\\_endpoint, Kobject, L4\\_PROTO\\_KOBJECT, Type\\_info::Demand\\_t< 1 > >](#)

- static void [\\_\\_check\\_protocols\\_\\_](#) () noexcept  
*Helper to check for protocol conflicts.*

## 16.194.1 Detailed Description

Interface for kernel objects that allow to receive IPC from them.

Such an object is for example an [lpc\\_gate](#) (with server rights) or an [lrq](#). Those objects can be bound to a thread that shall receive IPC from these objects via [bind\\_thread\(\)](#) or [bind\\_snd\\_destination\(\)](#).

Definition at line 30 of file [rcv\\_endpoint](#).

## 16.194.2 Member Function Documentation

### 16.194.2.1 [bind\\_snd\\_destination\(\)](#)

```
l4_msgtag_t L4::Rcv_endpoint::bind_snd_destination (
    Cap< Snd_destination > snd_dst,
    l4_umword_t label) [inline]
```

Bind a send destination (a thread or thread group) to an IPC receive endpoint.

#### Parameters

<i>snd_dst</i>	Snd_destination object (a thread or thread group) that shall be bound to this receive endpoint. See <a href="#">bind_thread()</a> and <a href="#">bind_snd_destination()</a> for binding a thread or thread group object.
<i>label</i>	Label to assign to <code>this</code> receive endpoint. For IPC gates, the two least significant bits must be set to zero.

### Returns

Syscall return tag containing one of the following return codes.

### Return values

<i>L4_EOK</i>	Operation successful.
<i>-L4_EINVAL</i>	<code>snd_dst</code> is not a thread or thread group object or other arguments were malformed.
<i>-L4_EPERM</i>	No <a href="#">L4_CAP_FPAGE_S</a> right on <code>snd_dst</code> or the capability used to invoke this operation.

### Precondition

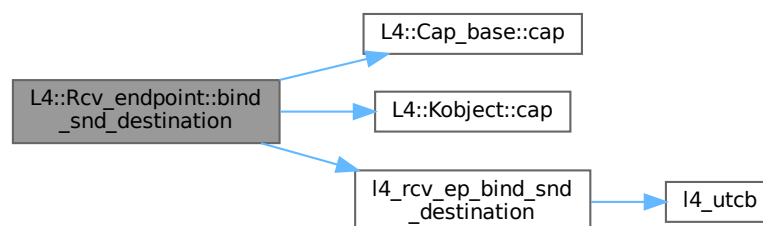
If this operation is invoked using an IPC gate capability without the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) right, the kernel will not perform the operation. Instead, the underlying IPC message will be forwarded to the thread bound to the IPC gate, blocking the caller if no thread or thread group is bound yet.

The specified `label` is passed to the receiver of the incoming IPC. It is possible to re-bind a receive endpoint to the same or a different thread or thread group. In this case, IPC already in flight will be delivered with the old label to the previously bound thread or thread group unless [L4::Thread::modify\\_senders\(\)](#) is used to change these labels.

Definition at line 101 of file [rcv\\_endpoint](#).

References [L4::Cap\\_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4\\_rcv\\_ep\\_bind\\_snd\\_destination\(\)](#).

Here is the call graph for this function:



### 16.194.2.2 bind\_thread()

```

l4_msgtag_t L4::Rcv_endpoint::bind_thread (
    Ipc::Cap< Thread > t,
    l4_umword_t label)

```

Bind the IPC receive endpoint to a thread.

### Parameters

<i>t</i>	<a href="#">Thread</a> object this receive endpoint shall be bound to.
<i>label</i>	Label to assign to <code>this</code> receive endpoint. For IPC gates, the two least significant bits must be set to zero.

### Returns

Syscall return tag containing one of the following return codes.

### Return values

<i>L4_EOK</i>	Operation successful.
<i>-L4_EINVAL</i>	<code>t</code> is not a thread object or other arguments were malformed.
<i>-L4_EPERM</i>	Insufficient permissions; see precondition.

### Precondition

The invoked capability and the capability `t` both must have the permission [L4\\_CAP\\_FPAGE\\_S](#).

**Deprecated** Use [bind\\_snd\\_destination\(\)](#) instead.

### Precondition

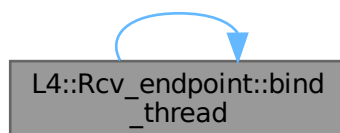
If this operation is invoked using an IPC gate capability without the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) right, the kernel will not perform the operation. Instead, the underlying IPC message will be forwarded to the thread the IPC gate is bound to, blocking the caller if the IPC gate was not bound yet.

The specified `label` is passed to the receiver of the incoming IPC. It is possible to re-bind a receive endpoint to the same or a different thread. In this case, IPC already in flight will be delivered with the old label to the previously bound thread unless [L4::Thread::modify\\_senders\(\)](#) is used to change these labels.

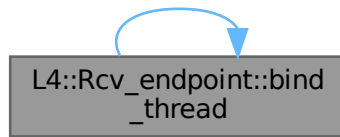
References [bind\\_thread\(\)](#).

Referenced by [bind\\_thread\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

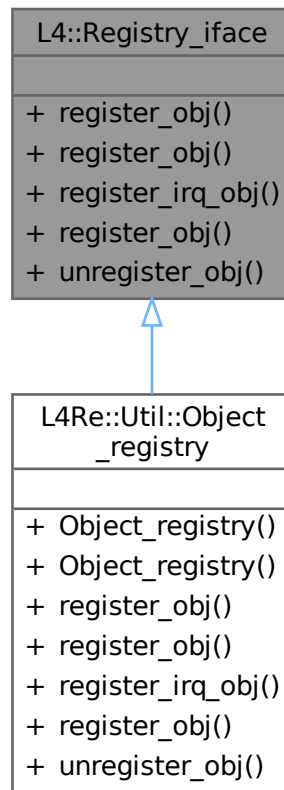
- [l4/sys/rcv\\_endpoint](#)

## 16.195 L4::Registry\_iface Class Reference

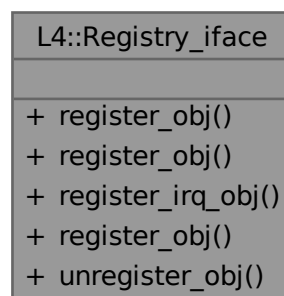
Abstract interface for object registries.

```
#include <ipc_epiface>
```

Inheritance diagram for L4::Registry\_iface:



Collaboration diagram for L4::Registry\_iface:



## Public Member Functions

- virtual [L4::Cap](#)< void > [register\\_obj](#) ([L4::Epiface](#) \*o, char const \*service)=0

- Register an [L4::Epiface](#) for an IPC gate available in the applications environment under the name *service*.
- virtual [L4::Cap](#)< void > [register\\_obj](#) ([L4::Epiface](#) \*o)=0  
Register o as server-side object for synchronous RPC.
- virtual [L4::Cap](#)< [L4::Irq](#) > [register\\_irq\\_obj](#) ([L4::Epiface](#) \*o)=0  
Register o as server-side object for asynchronous IRQs.
- virtual [L4::Cap](#)< [L4::Rcv\\_endpoint](#) > [register\\_obj](#) ([L4::Epiface](#) \*o, [L4::Cap](#)< [L4::Rcv\\_endpoint](#) > ep)=0  
Register o as server-side object for a pre-allocated capability.
- virtual void [unregister\\_obj](#) ([L4::Epiface](#) \*o, bool unmap=true)=0  
Unregister the given object o from the server.

### 16.195.1 Detailed Description

Abstract interface for object registries.

An object registry allows to register [L4::Epiface](#) objects at a server loop either for synchronous RPC messages or for asynchronous IRQ messages.

Definition at line 322 of file [ipc\\_epiface](#).

### 16.195.2 Member Function Documentation

#### 16.195.2.1 [register\\_irq\\_obj\(\)](#)

```
virtual L4::Cap< L4::Irq > L4::Registry\_iface::register\_irq\_obj (  
    L4::Epiface * o) [pure virtual]
```

Register o as server-side object for asynchronous IRQs.

#### Parameters

<i>o</i>	Pointer to an <a href="#">Epiface</a> object that shall be registered as server-side object for IRQs.
----------	-------------------------------------------------------------------------------------------------------

#### Return values

<a href="#">L4::Cap</a> < <a href="#">L4::Irq</a> >	Capability to a new IRQ object on success.
<a href="#">L4::Cap</a> < <a href="#">L4::Irq</a> >:: <a href="#">Invalid</a>	The allocation of the IRQ has failed.

After successful registration `o->obj_cap()` will be the capability of the allocated IRQ object.

The function may allocate a capability slot for the object. In that case [unregister\\_obj\(\)](#) is responsible for freeing the slot as well.

Implemented in [L4Re::Util::Object\\_registry](#).

#### 16.195.2.2 [register\\_obj\(\)](#) [1/3]

```
virtual L4::Cap< void > L4::Registry\_iface::register\_obj (  
    L4::Epiface * o) [pure virtual]
```

Register o as server-side object for synchronous RPC.

#### Parameters



<i>o</i>	Pointer to an <a href="#">Epiface</a> object that shall be registered as server-side object for RPC.
----------	------------------------------------------------------------------------------------------------------

**Return values**

<a href="#">L4::Cap&lt;void&gt;</a>	A valid capability to a new IPC gate.
<a href="#">L4::Cap&lt;void&gt;::Invalid</a>	The allocation of the IPC gate has failed.

After successful registration `o->obj_cap()` will be the capability of the allocated IPC gate.

The function may allocate a capability slot for the object. In that case [unregister\\_obj\(\)](#) is responsible for freeing the slot as well.

Implemented in [L4Re::Util::Object\\_registry](#).

**16.195.2.3 register\_obj() [2/3]**

```
virtual L4::Cap< void > L4::Registry_iface::register_obj (
    L4::Epiface * o,
    char const * service) [pure virtual]
```

Register an [L4::Epiface](#) for an IPC gate available in the applications environment under the name `service`.

**Parameters**

<i>o</i>	Pointer to an <a href="#">Epiface</a> object that shall be registered.
<i>service</i>	Name of the capability that shall be used to connect <code>o</code> to as a server-side object.

**Return values**

<a href="#">L4::Cap&lt;void&gt;</a>	The capability known as <code>service</code> on success.
<a href="#">L4::Cap&lt;void&gt;::Invalid</a>	No capability with the given name found.

After a successful call to this function `o->obj_cap()` is equal to the capability in the environment with the name given by `service`.

Implemented in [L4Re::Util::Object\\_registry](#).

**16.195.2.4 register\_obj() [3/3]**

```
virtual L4::Cap< L4::Rcv\_endpoint > L4::Registry_iface::register_obj (
    L4::Epiface * o,
    L4::Cap< L4::Rcv\_endpoint > ep) [pure virtual]
```

Register `o` as server-side object for a pre-allocated capability.

**Parameters**

<i>o</i>	Pointer to an <a href="#">Epiface</a> object that shall be registered as server-side object.
<i>ep</i>	Capability to an already allocated capability where <i>o</i> shall be attached as server-side handler. The capability may point to an IPC gate or an IRQ.

### Return values

<a href="#">L4::Cap&lt;L4::Rcv_endpoint&gt;</a>	Capability <i>ep</i> on success.
<a href="#">L4::Cap&lt;L4::Rcv_endpoint&gt;::Invalid</a>	The IRQ attach operation has failed.

After successful registration `o->obj_cap()` will be equal to *ep*.

Implemented in [L4Re::Util::Object\\_registry](#).

### 16.195.2.5 unregister\_obj()

```
virtual void L4::Registry_iface::unregister_obj (
    L4::Epiface * o,
    bool unmap = true) [pure virtual]
```

Unregister the given object *o* from the server.

### Parameters

<i>o</i>	Pointer to the <a href="#">Epiface</a> object that shall be unregistered. The object must have been registered with any of the register methods if <a href="#">Registry_iface</a> .
<i>unmap</i>	If true the capability <code>o-&gt;obj_cap()</code> shall be unmapped from the local object space.

The function always unmaps and frees the capability if it was allocated by either [Registry\\_iface::register\\_irq\\_obj\(L4::Epiface \\*\)](#), or by [Registry\\_iface::register\\_obj\(L4::Epiface \\*\)](#).

Implemented in [L4Re::Util::Object\\_registry](#).

The documentation for this class was generated from the following file:

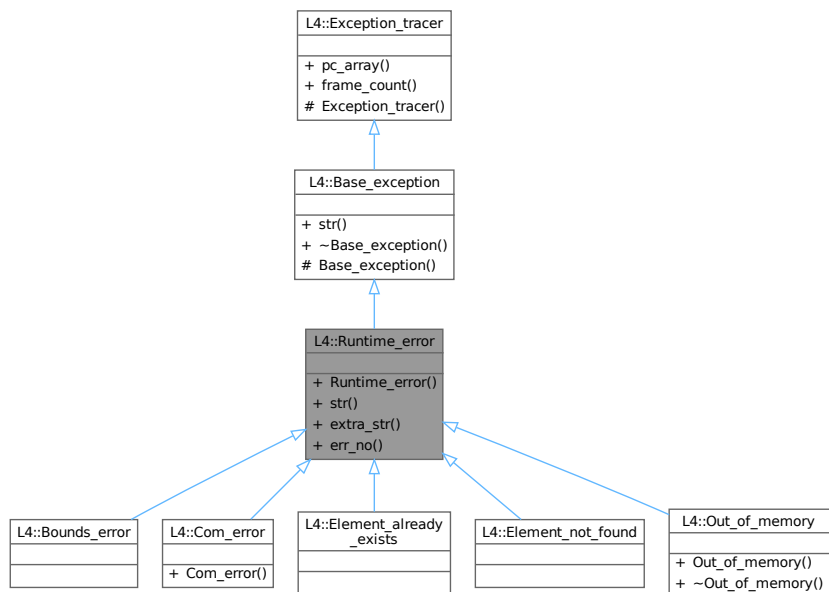
- `l4/sys/cxx/ipc_epiface`

## 16.196 L4::Runtime\_error Class Reference

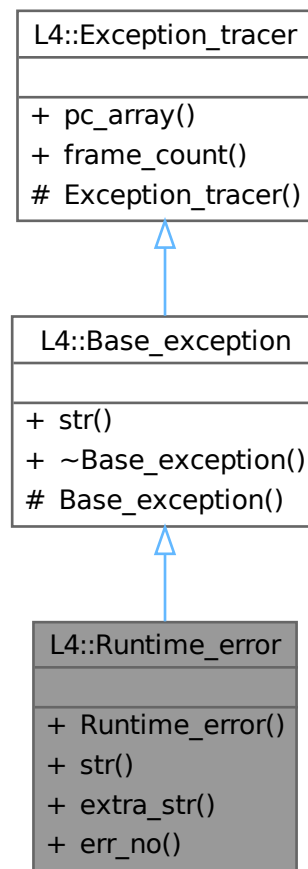
[Exception](#) for an abstract runtime error.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Runtime\_error:



Collaboration diagram for L4::Runtime\_error:



## Public Member Functions

- [Runtime\\_error](#) (long [err\\_no](#), char const \*extra=0) noexcept  
*Create a new [Runtime\\_error](#).*
- char const \* [str](#) () const noexcept override  
*Return a human readable string for the exception.*
- char const \* [extra\\_str](#) () const  
*Get the description text for this runtime error.*
- long [err\\_no](#) () const noexcept  
*Get the error value for this runtime error.*

## Public Member Functions inherited from [L4::Base\\_exception](#)

- virtual `~Base\_exception` () noexcept  
*Destruction.*

## Public Member Functions inherited from [L4::Exception\\_tracer](#)

- `void const *const * pc_array () const noexcept`  
*Get the array containing the call trace.*
- `int frame_count () const noexcept`  
*Get the number of entries that are valid in the call trace.*

## Additional Inherited Members

## Protected Member Functions inherited from [L4::Base\\_exception](#)

- `Base_exception () noexcept`  
*Create a base exception.*

## Protected Member Functions inherited from [L4::Exception\\_tracer](#)

- `Exception_tracer () noexcept`  
*Create a back trace.*

## 16.196.1 Detailed Description

[Exception](#) for an abstract runtime error.

This is the base class for a set of exceptions that cover all errors that have a C error value (see [l4\\_error\\_code\\_t](#)).

Definition at line [128](#) of file [exceptions](#).

## 16.196.2 Constructor & Destructor Documentation

### 16.196.2.1 Runtime\_error()

```
L4::Runtime_error::Runtime_error (
    long err_no,
    char const * extra = 0) [inline], [explicit], [noexcept]
```

Create a new [Runtime\\_error](#).

#### Parameters

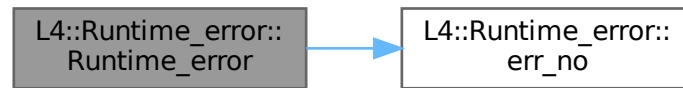
<i>err_no</i>	Error value for this runtime error.
<i>extra</i>	Description of what happened when the error occurred.

Definition at line [141](#) of file [exceptions](#).

References [err\\_no\(\)](#).

Referenced by [L4::Out\\_of\\_memory::Out\\_of\\_memory\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



## 16.196.3 Member Function Documentation

### 16.196.3.1 `err_no()`

```
long L4::Runtime_error::err_no () const [inline], [noexcept]
```

Get the error value for this runtime error.

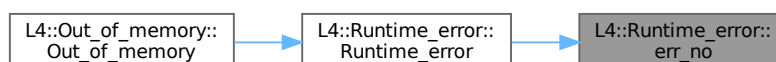
#### Returns

Error value.

Definition at line 170 of file [exceptions](#).

Referenced by [Runtime\\_error\(\)](#).

Here is the caller graph for this function:



### 16.196.3.2 extra\_str()

```
char const * L4::Runtime_error::extra_str () const [inline]
```

Get the description text for this runtime error.

#### Returns

Pointer to the description string.

Definition at line 162 of file [exceptions](#).

The documentation for this class was generated from the following file:

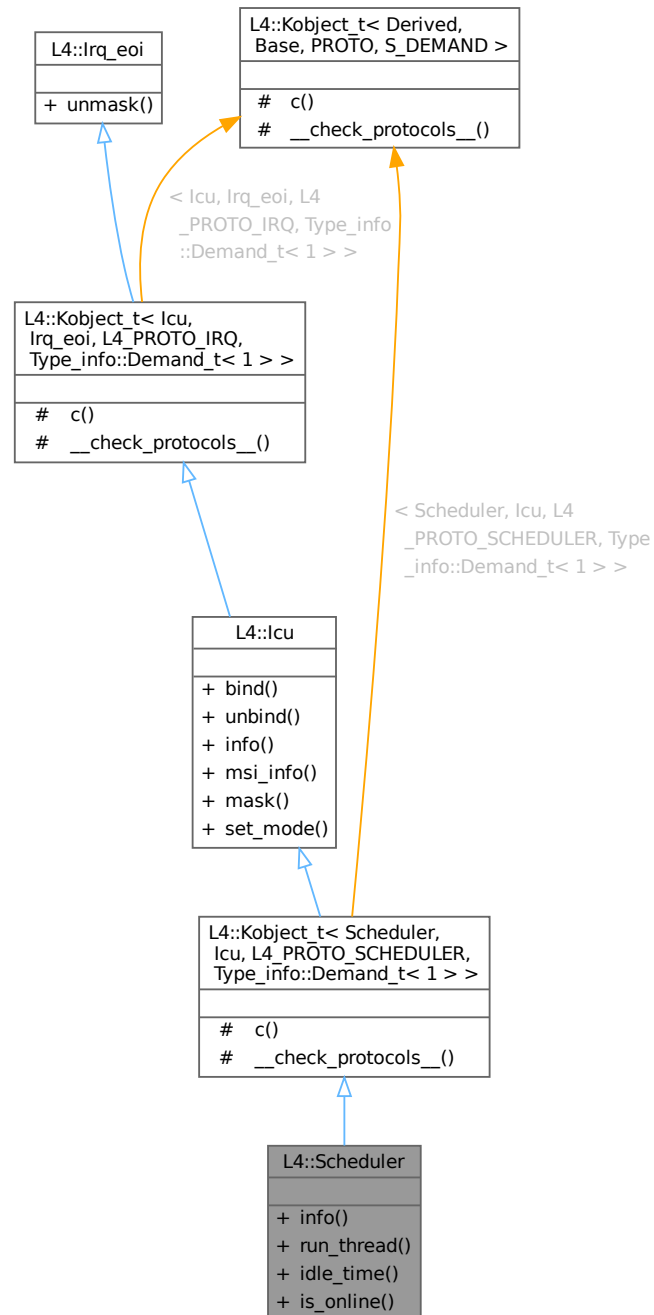
- [l4/cxx/exceptions](#)

## 16.197 L4::Scheduler Class Reference

C++ interface of the [Scheduler](#) kernel object, see [Scheduler](#) for the C interface.

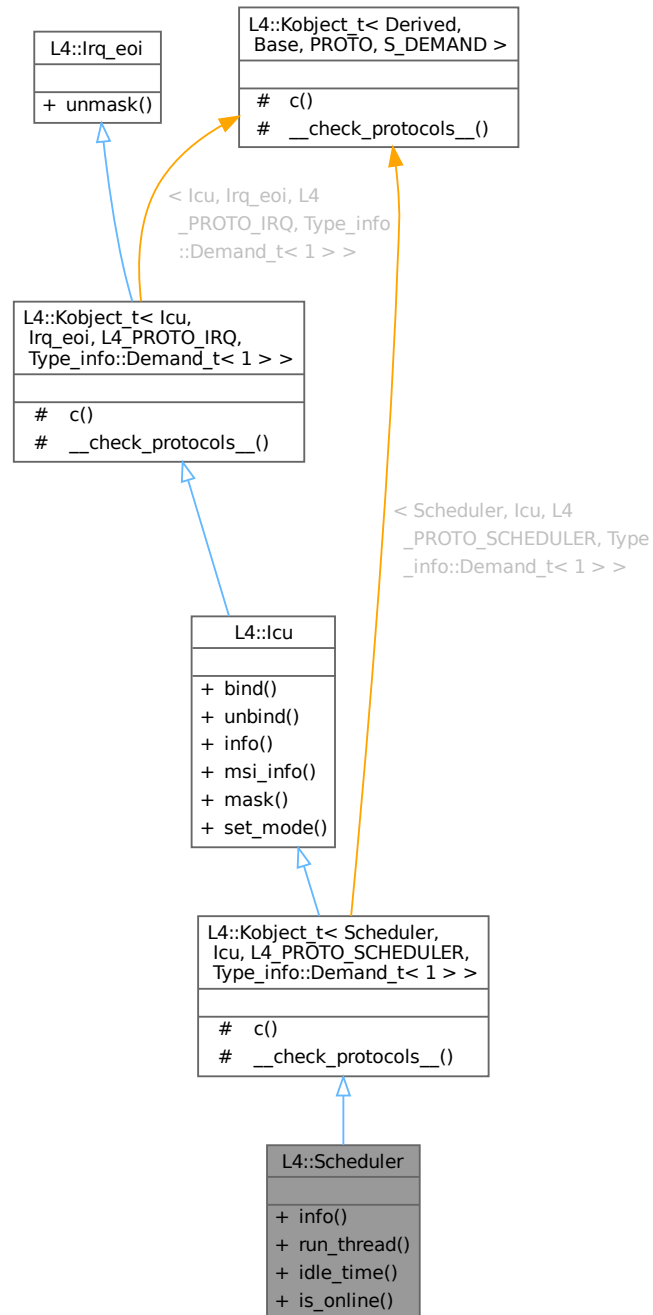
```
#include <scheduler>
```

Inheritance diagram for L4::Scheduler:





Collaboration diagram for L4::Scheduler:



## Public Member Functions

- `l4_msgtag_t info (l4_umword_t *cpu_max, l4_sched_cpu_set_t *cpus, l4_umword_t *sched_classes=nullptr, l4_utcb_t *utcb=l4_utcb()) const noexcept`  
Get scheduler information.
- `l4_msgtag_t run_thread (lpc::Cap< Thread > thread, l4_sched_param_t const &sp)`  
Run a thread on a *Scheduler*.

- [l4\\_msgtag\\_t idle\\_time](#) ([l4\\_sched\\_cpu\\_set\\_t](#) const &cpus, [l4\\_kernel\\_clock\\_t](#) \*us)  
*Query the idle time (in  $\mu$ s) of a CPU.*
- [bool is\\_online](#) ([l4\\_umword\\_t](#) cpu, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) const noexcept  
*Query if a CPU is online.*

## Public Member Functions inherited from [L4::lcu](#)

- [l4\\_msgtag\\_t bind](#) (unsigned irqnum, [L4::Cap](#)< [Triggerable](#) > irq, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Bind an interrupt line of an interrupt controller to an interrupt object.*
- [l4\\_msgtag\\_t unbind](#) (unsigned irqnum, [L4::Cap](#)< [Triggerable](#) > irq, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Remove binding of an interrupt line from the interrupt controller object.*
- [l4\\_msgtag\\_t info](#) ([l4\\_icu\\_info\\_t](#) \*info, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Get information about the ICU features.*
- [l4\\_msgtag\\_t msi\\_info](#) ([l4\\_umword\\_t](#) irqnum, [l4\\_uint64\\_t](#) source, [l4\\_icu\\_msi\\_info\\_t](#) \*msi\_info)  
*Get MSI info about IRQ.*
- [l4\\_msgtag\\_t mask](#) (unsigned irqnum, [l4\\_umword\\_t](#) \*label=0, [l4\\_timeout\\_t](#) to=[L4\\_IPC\\_NEVER](#), [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Mask an IRQ line.*
- [l4\\_msgtag\\_t set\\_mode](#) (unsigned irqnum, [l4\\_umword\\_t](#) mode, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Set interrupt mode.*

## Public Member Functions inherited from [L4::lrq\\_eoi](#)

- [l4\\_msgtag\\_t unmask](#) (unsigned irqnum, [l4\\_umword\\_t](#) \*label=0, [l4\\_timeout\\_t](#) to=[L4\\_IPC\\_NEVER](#), [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Unmask the given interrupt line.*

## Additional Inherited Members

### Protected Types inherited from

[L4::Kobject\\_t](#)< [Scheduler](#), [lcu](#), [L4\\_PROTO\\_SCHEDULER](#), [Type\\_info::Demand\\_t](#)< 1 > >

- typedef [Scheduler](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef [Typeid::Iface](#)< [PROTO](#), [Scheduler](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef [Typeid::Merge\\_list](#)< [Typeid::Iface\\_list](#)< **\_\_Iface** >, typename [lcu](#)::[\\_\\_Iface\\_list](#) > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Types inherited from

[L4::Kobject\\_t](#)< [lcu](#), [lrq\\_eoi](#), [L4\\_PROTO\\_IRQ](#), [Type\\_info::Demand\\_t](#)< 1 > >

- typedef [lcu](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef [Typeid::Iface](#)< [PROTO](#), [lcu](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef [Typeid::Merge\\_list](#)< [Typeid::Iface\\_list](#)< **\_\_Iface** >, typename [lrq\\_eoi](#)::[\\_\\_Iface\\_list](#) > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

**Protected Member Functions inherited from****L4::Kobject\_t< Scheduler, Icu, L4\_PROTO\_SCHEDULER, Type\_info::Demand\_t< 1 > >**

- **L4::Cap< Class > c ()** const noexcept

*Get the capability to ourselves.***Protected Member Functions inherited from****L4::Kobject\_t< Icu, Irq\_eoi, L4\_PROTO\_IRQ, Type\_info::Demand\_t< 1 > >**

- **L4::Cap< Class > c ()** const noexcept

*Get the capability to ourselves.***Static Protected Member Functions inherited from****L4::Kobject\_t< Scheduler, Icu, L4\_PROTO\_SCHEDULER, Type\_info::Demand\_t< 1 > >**

- **static void \_\_check\_protocols\_\_ ()** noexcept

*Helper to check for protocol conflicts.***Static Protected Member Functions inherited from****L4::Kobject\_t< Icu, Irq\_eoi, L4\_PROTO\_IRQ, Type\_info::Demand\_t< 1 > >**

- **static void \_\_check\_protocols\_\_ ()** noexcept

*Helper to check for protocol conflicts.***16.197.1 Detailed Description**

C++ interface of the [Scheduler](#) kernel object, see [Scheduler](#) for the C interface.

The [Scheduler](#) interface allows a client to manage CPU resources. The API provides functions to query scheduler information, check the online state of CPUs, query CPU idle time and to start threads on defined CPU sets.

The scheduler offers IRQ number 0, which triggers when the number of online cores changes, e.g. due to hotplug events.

The [Scheduler](#) interface inherits from [L4::Icu](#) and [L4::Irq\\_eoi](#) for managing the scheduler virtual device IRQ which, in contrast to hardware IRQs, implements a limited functionality:

- Only IRQ line 0 is supported, no MSI vectors.
- The IRQ is edge-triggered and the IRQ mode cannot be changed.
- As the IRQ is edge-triggered, it does not have to be explicitly unmasked.

It depends on the platform, which hotplug events actually trigger the IRQ. Many platforms only support triggering the IRQ when a CPU core different from the boot CPU goes online.

**Include File**

```
#include <l4/sys/scheduler>
```

Definition at line 46 of file [scheduler](#).

## 16.197.2 Member Function Documentation

### 16.197.2.1 idle\_time()

```
l4_msgtag_t L4::Scheduler::idle_time (  
    l4_sched_cpu_set_t const & cpus,  
    l4_kernel_clock_t * us)
```

Query the idle time (in  $\mu$ s) of a CPU.

#### Parameters

---

	<i>cpus</i>	Set of CPUs to query. Only the idle time of the first selected CPU in <code>cpus.map</code> is queried.
out	<i>us</i>	Idle time of queried CPU in $\mu$ s.

### Return values

0	Success.
-L4_EINVAL	Invalid CPU requested in cpu set.

This function retrieves the idle time in  $\mu$ s of the first selected CPU in `cpus.map`. The idle time is the accumulated time a CPU has spent in the idle thread since its last reset. To calculate a load estimate  $l$  one has to retrieve the idle time at the beginning ( $i1$ ) and the end ( $i2$ ) of a known time interval  $t$ . The load is then calculated as  $l = 1 - (i2 - i1)/t$ .

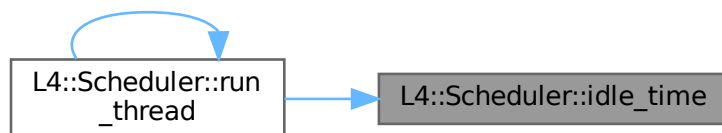
The idle time is only defined for online CPUs. Reading the idle time from offline CPUs is undefined and may result in either getting -L4\_EINVAL or calculating an estimated (incorrect) load of 1.

### Note

The idle time statistics of remote CPUs is updated on context switch events only, hence may not be up-to-date when requested cross-CPU. To get up-to-date idle time you should use a thread running on the same CPU of which the idle time is requested.

Referenced by [run\\_thread\(\)](#).

Here is the caller graph for this function:



### 16.197.2.2 info()

```

l4_msgtag_t L4::Scheduler::info (
    l4_umword_t * cpu_max,
    l4_sched_cpu_set_t * cpus,
    l4_umword_t * sched_classes = nullptr,
    l4_utcb_t * utcb = l4_utcb()) const [inline], [noexcept]
  
```

Get scheduler information.

### Parameters

out	<i>cpu_max</i>	Maximum number of CPUs ever available. Optional, can be nullptr.
in, out	<i>cpus</i>	<i>cpus.offset</i> is first CPU of interest. <i>cpusgranularity</i> (see <a href="#">l4_sched_cpu_set_t</a> ). <i>cpus.map</i> Bitmap of online CPUs. Must not be nullptr.
out	<i>sched_classes</i>	A bitmap of available scheduling classes (see <a href="#">L4_scheduler_classes</a> ). Pass nullptr to omit this information.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

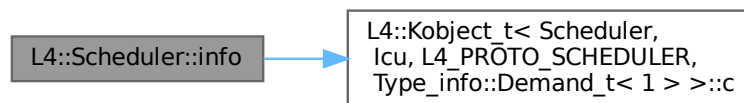
### Return values

0	Success.
-L4_ERANGE	The given CPU offset is larger than the maximum number of CPUs.

Definition at line 74 of file [scheduler](#).

References [L4::Kobject\\_t< Scheduler, lcu, L4\\_PROTO\\_SCHEDULER, Type\\_info::Demand\\_t< 1 > >::c\(\)](#), [l4\\_sched\\_cpu\\_set\\_t::gran\\_offset](#), and [l4\\_sched\\_cpu\\_set\\_t::map](#).

Here is the call graph for this function:



### 16.197.2.3 is\_online()

```

bool L4::Scheduler::is_online (
    l4_umword_t cpu,
    l4_utcb_t * utcb = l4_utcb()) const [inline], [noexcept]
  
```

Query if a CPU is online.

### Parameters

<i>cpu</i>	CPU number whose online status should be queried.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

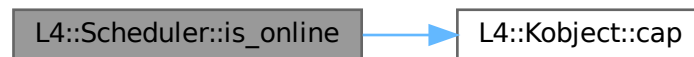
### Return values

<i>true</i>	The CPU is online.
<i>false</i>	The CPU is offline

Definition at line 154 of file [scheduler](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



#### 16.197.2.4 run\_thread()

```

l4_msgtag_t L4::Scheduler::run_thread (
    ipc::Cap< Thread > thread,
    l4_sched_param_t const & sp)
  
```

Run a thread on a [Scheduler](#).

##### Parameters

<i>thread</i>	Capability of the thread to run.
<i>sp</i>	Scheduling parameters.

##### Return values

0	Success.
-L4_EINVAL	Invalid size of the scheduling parameter.

This function launches a thread on a CPU determined by the scheduling parameter `sp.affinity`. A thread can be intentionally stopped by migrating it on an offline or an invalid CPU. The thread is only guaranteed to run if the CPU it is migrated to is currently online.

##### Note

If the target CPU is currently not online, there is no guarantee that the thread will ever run, even if the CPU comes online later on.

A scheduler may impose a policy with regard to selecting CPUs. However the scheduler is required to ensure the following two properties:

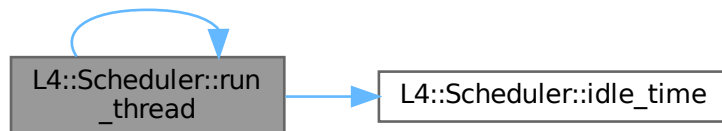
- Two threads with disjoint CPU sets must be scheduled to different CPUs.

- Two threads with identical CPU sets selecting only a single CPU must be scheduled to the same CPU.

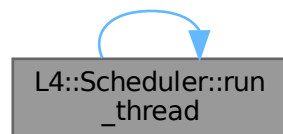
References [idle\\_time\(\)](#), [L4\\_SCHEDULER\\_IDLE\\_TIME\\_OP](#), and [run\\_thread\(\)](#).

Referenced by [run\\_thread\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- [l4/sys/scheduler](#)

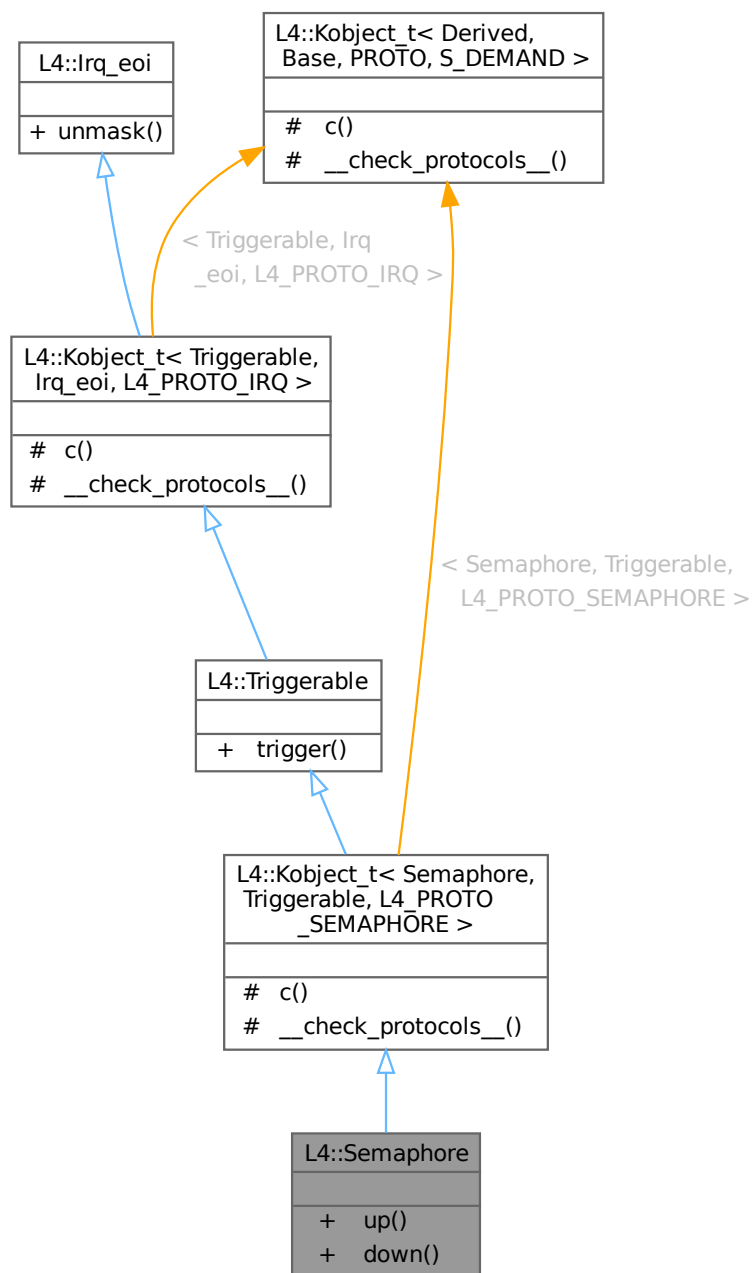
## 16.198 L4::Semaphore Struct Reference

C++ Kernel-provided semaphore interface, see [Kernel-provided semaphore](#) for the C interface.

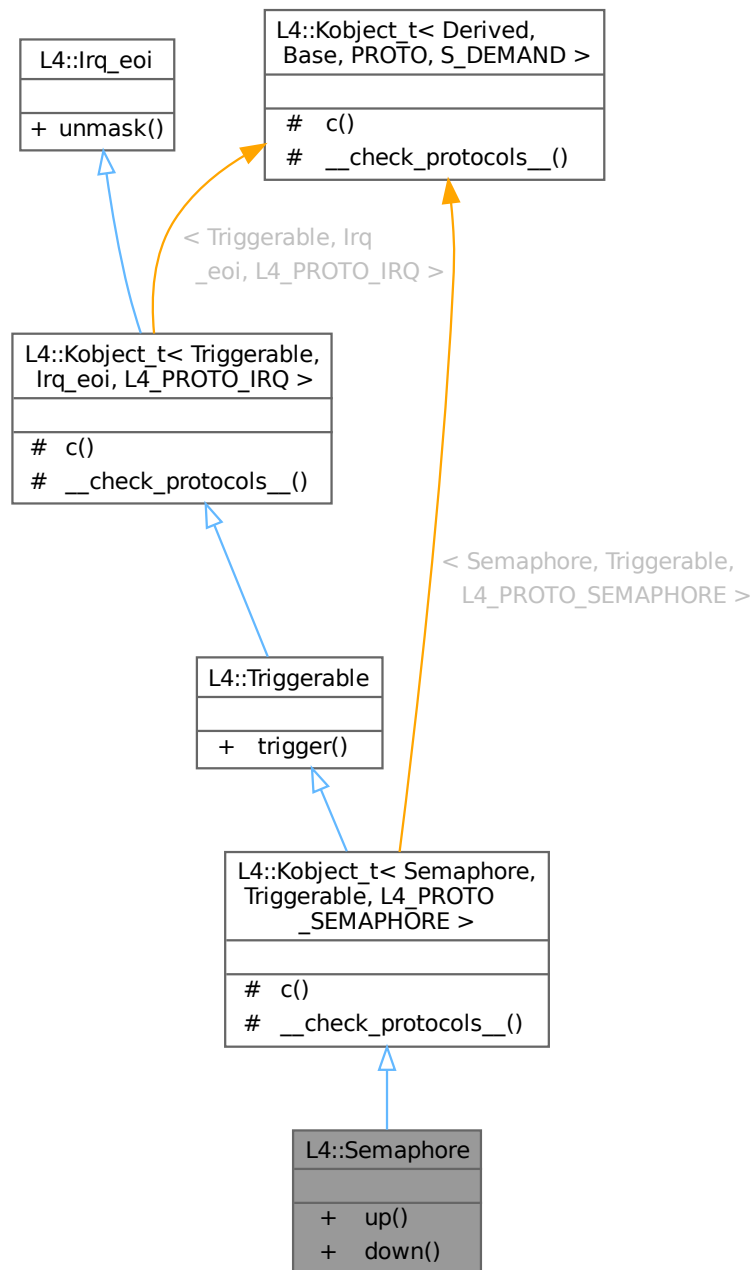
```
#include <semaphore>
```



Inheritance diagram for L4::Semaphore:



Collaboration diagram for L4::Semaphore:



## Public Member Functions

- `l4_msgtag_t up (l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Semaphore up operation (wrapper for `trigger()`).*
- `l4_msgtag_t down (l4_timeout_t timeout=L4_IPC_NEVER, l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Semaphore down operation.*

## Public Member Functions inherited from [L4::Triggerable](#)

- [l4\\_msgtag\\_t trigger](#) ([l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Trigger the object.*

## Public Member Functions inherited from [L4::Irq\\_eoi](#)

- [l4\\_msgtag\\_t unmask](#) (unsigned irqnum, [l4\\_umword\\_t](#) \*label=0, [l4\\_timeout\\_t](#) to=[L4\\_IPC\\_NEVER](#), [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Unmask the given interrupt line.*

## Additional Inherited Members

### Protected Types inherited from [L4::Kobject\\_t](#)< [Semaphore](#), [Triggerable](#), [L4\\_PROTO\\_SEMAPHORE](#) >

- typedef [Semaphore](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< [PROTO](#), [Semaphore](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename [Triggerable](#)::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Types inherited from [L4::Kobject\\_t](#)< [Triggerable](#), [Irq\\_eoi](#), [L4\\_PROTO\\_IRQ](#) >

- typedef [Triggerable](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< [PROTO](#), [Triggerable](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename [Irq\\_eoi](#)::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Member Functions inherited from [L4::Kobject\\_t](#)< [Semaphore](#), [Triggerable](#), [L4\\_PROTO\\_SEMAPHORE](#) >

- [L4::Cap](#)< [Class](#) > **c** () const noexcept  
*Get the capability to ourselves.*

### Protected Member Functions inherited from [L4::Kobject\\_t](#)< [Triggerable](#), [Irq\\_eoi](#), [L4\\_PROTO\\_IRQ](#) >

- [L4::Cap](#)< [Class](#) > **c** () const noexcept  
*Get the capability to ourselves.*

### Static Protected Member Functions inherited from

[L4::Kobject\\_t](#) < [Semaphore](#), [Triggerable](#), [L4\\_PROTO\\_SEMAPHORE](#) >

- static void `__check_protocols__()` noexcept

*Helper to check for protocol conflicts.*

### Static Protected Member Functions inherited from

[L4::Kobject\\_t](#) < [Triggerable](#), [Irq\\_eoi](#), [L4\\_PROTO\\_IRQ](#) >

- static void `__check_protocols__()` noexcept

*Helper to check for protocol conflicts.*

## 16.198.1 Detailed Description

C++ Kernel-provided semaphore interface, see [Kernel-provided semaphore](#) for the C interface.

This is the interface for kernel-provided semaphore objects. The object provides the classical functions `up()` and `down()` for counting the semaphore and blocking. The semaphore is a [Triggerable](#) with respect to the `up()` function, this means that a semaphore can be bound to an interrupt line at an ICU ([L4::lcu](#)) and incoming interrupts increment the semaphore counter.

The `down()` method decrements the semaphore counter and blocks if the counter is already zero. Blocking on a semaphore may—as all blocking operations—either return successfully, or be aborted due to an expired timeout provided to the `down()` operation, or due to an [L4::Thread::ex\\_regs\(\)](#) operation with the [L4\\_THREAD\\_EX\\_REGS\\_CANCEL](#) flag set.

A semaphore object is initialized with counter value 0.

The main reason for using a semaphore instead of an [L4::Irq](#) is to ensure that incoming trigger signals do not interfere with any open-wait operations, as used for example in a server loop.

Note that this is a kernel-level semaphore primitive that shall be used to implement user-level, application-usable synchronization primitives. For example, use `pthread_mutex` functions in applications if possible. When implementing a synchronization primitive, please ensure to only use [L4::Semaphore](#) in the case of contention, and use atomic operations for the non-contended case.

Definition at line 51 of file [semaphore](#).

## 16.198.2 Member Function Documentation

### 16.198.2.1 `down()`

```
l4_msgtag_t L4::Semaphore::down (
    l4_timeout_t timeout = L4_IPC_NEVER,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

[Semaphore](#) down operation.

#### Parameters

<i>timeout</i>	Timeout for blocking the semaphore down operation. Note: The receive timeout of this timeout-pair is significant for blocking, the send part is usually non-blocking.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

#### Returns

Syscall return tag. Use [l4\\_error\(\)](#) to check for errors.

#### Return values

<code>-L4_EPERM</code>	Insufficient permissions; see precondition.
------------------------	---------------------------------------------

#### Precondition

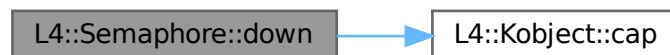
The invoked [Semaphore](#) capability must have the permission [L4\\_CAP\\_FPAGE\\_S](#).

This method decrements the semaphore counter by one, or blocks if the counter is already zero, until either a timeout or cancel condition hits or the counter is increased by an [up\(\)](#) operation.

Definition at line 89 of file [semaphore](#).

References [L4::Kobject::cap\(\)](#), and [L4\\_IPC\\_NEVER](#).

Here is the call graph for this function:



#### 16.198.2.2 up()

```

l4_msgtag_t L4::Semaphore::up (
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

[Semaphore](#) up operation (wrapper for [trigger\(\)](#)).

#### Parameters

<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .
-------------	------------------------------------------------------------------------------------------------------------

**Returns**

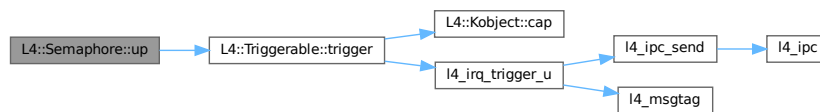
Syscall return tag for a send-only operation, this means there is no return value except [L4\\_MSGTAG\\_ERROR](#) indicating success or failure of the send operation. Use [l4\\_ipc\\_error\(\)](#) to check for errors and **do not** use [l4\\_error\(\)](#).

Increases the semaphore counter by one if it is smaller than an unspecified limit. The unspecified limit is guaranteed to be at least  $2^{31}-1$ .

Definition at line 67 of file [semaphore](#).

References [L4::Triggerable::trigger\(\)](#).

Here is the call graph for this function:



The documentation for this struct was generated from the following file:

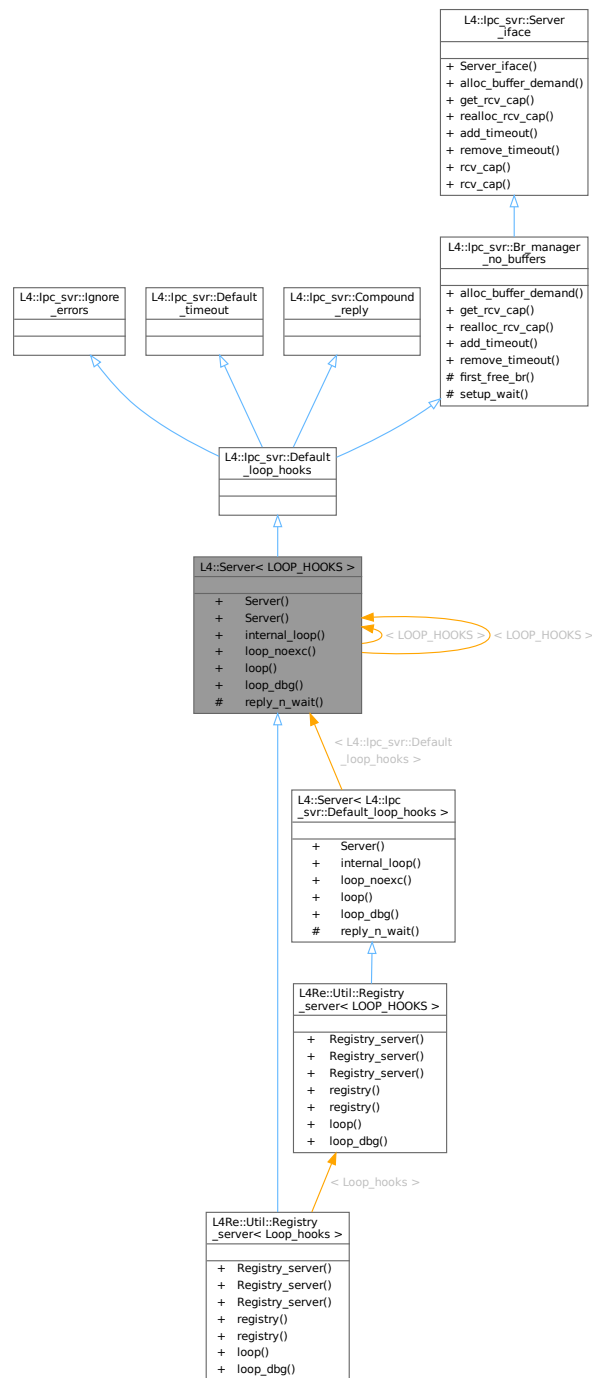
- [l4/sys/semaphore](#)

## 16.199 L4::Server< LOOP\_HOOKS > Class Template Reference

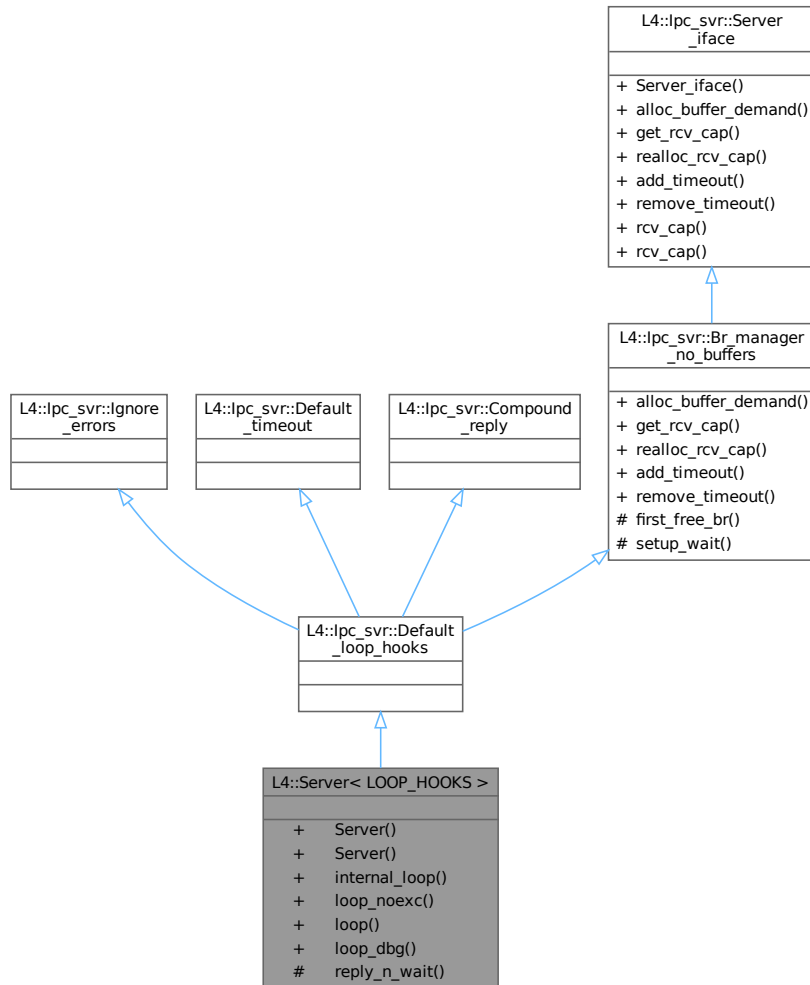
Basic server loop for handling client requests.

```
#include <ipc_server_loop>
```

Inheritance diagram for L4::Server< LOOP\_HOOKS >:



Collaboration diagram for L4::Server< LOOP\_HOOKS >:



## Public Member Functions

- [Server](#) ([l4\\_utcb\\_t](#) \*)  
*Initializes the server loop.*
- [Server](#) ()  
*Initializes the server loop.*
- [template](#)<typename DISPATCH>  
[L4\\_NORETURN](#) void [internal\\_loop](#) (DISPATCH dispatch, [l4\\_utcb\\_t](#) \*)  
*The server loop.*
- [template](#)<typename R>  
[L4\\_NORETURN](#) void [loop\\_noexc](#) (R r, [l4\\_utcb\\_t](#) \*u=[l4\\_utcb](#)())  
*Server loop without exception handling.*
- [template](#)<typename EXC, typename R>  
[L4\\_NORETURN](#) void [loop](#) (R r, [l4\\_utcb\\_t](#) \*u=[l4\\_utcb](#)())  
*Server loop with internal exception handling.*
- [template](#)<typename EXC, typename R, typename Printer>  
[L4\\_NORETURN](#) void [loop\\_dbg](#) (R r, Printer p, [l4\\_utcb\\_t](#) \*u=[l4\\_utcb](#)())  
*Server loop with internal exception handling including message printing.*



## Public Member Functions inherited from L4::lpc\_svr::Br\_manager\_no\_buffers

- int **alloc\_buffer\_demand** ([Demand](#) const &demand) override  
*Tells the server to allocate buffers for the given demand.*
- [L4::Cap](#)< void > **get\_rcv\_cap** (int) const override  
*Returns [L4::Cap](#)<void>::Invalid, we have no buffer management.*
- int **realloc\_rcv\_cap** (int) override  
*Returns -L4\_ENOMEM, we have no buffer management.*
- int **add\_timeout** ([Timeout](#) \*, [l4\\_kernel\\_clock\\_t](#)) override  
*Returns -L4\_ENOSYS, we have no timeout queue.*
- int **remove\_timeout** ([Timeout](#) \*) override  
*Returns -L4\_ENOSYS, we have no timeout queue.*

## Public Member Functions inherited from L4::lpc\_svr::Server\_iface

- **Server\_iface** ()  
*Make a server interface.*
- template<typename T>  
[L4::Cap](#)< T > **rcv\_cap** (int index) const  
*Get given receive buffer as typed capability.*
- [L4::Cap](#)< void > **rcv\_cap** (int index) const  
*Get receive cap with the given index as generic (void) type.*

## Protected Member Functions

- [l4\\_msgtag\\_t](#) **reply\_n\_wait** ([l4\\_msgtag\\_t](#) reply, [l4\\_umword\\_t](#) \*p, [l4\\_utcb\\_t](#) \*)  
*Internal implementation for reply and wait.*

## Protected Member Functions inherited from L4::lpc\_svr::Br\_manager\_no\_buffers

- unsigned **first\_free\_br** () const  
*Returns 1 as first free buffer.*
- void **setup\_wait** ([l4\\_utcb\\_t](#) \*utcb, [L4::lpc\\_svr::Reply\\_mode](#))  
*Setup wait function for the server loop ([Server](#)<>).*

## Additional Inherited Members

## Public Types inherited from L4::lpc\_svr::Server\_iface

- typedef [L4::Type\\_info::Demand](#) **Demand**  
*Data type expressing server-side demand for receive buffers.*

## 16.199.1 Detailed Description

```
template<typename LOOP_HOOKS = lpc_svr::Default_loop_hooks>
class L4::Server< LOOP_HOOKS >
```

Basic server loop for handling client requests.

### Parameters

<code>LOOP_HOOKS</code>	the server inherits from <code>LOOP_HOOKS</code> and calls the hooks defined in <code>LOOP_HOOKS</code> in the server loop. See <a href="#">lpc_svr::Default_loop_hooks</a> , <a href="#">lpc_svr::Ignore_errors</a> , <a href="#">lpc_svr::Default_timeout</a> , <a href="#">lpc_svr::Compound_reply</a> , and <a href="#">lpc_svr::Br_manager_no_buffers</a> .
-------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

This is basically a simple server loop that uses a single message buffer for receiving requests and sending replies. The dispatcher determines how incoming messages are handled.

Definition at line 307 of file [ipc\\_server\\_loop](#).

## 16.199.2 Constructor & Destructor Documentation

### 16.199.2.1 Server()

```
template<typename LOOP_HOOKS = Ipc_svr::Default_loop_hooks>
L4::Server< LOOP_HOOKS >::Server (
    l4_utcb_t * ) [inline], [explicit]
```

Initializes the server loop.

Note that this variant is deprecated. The UTCB can be specified with the server loop function ([l4\\_utcb\(\)](#) is the default). (2021-10)

Definition at line 319 of file [ipc\\_server\\_loop](#).

## 16.199.3 Member Function Documentation

### 16.199.3.1 internal\_loop()

```
template<typename L>
template<typename DISPATCH>
L4_NORETURN void L4::Server< L >::internal_loop (
    DISPATCH dispatch,
    l4_utcb_t * utcb) [inline]
```

The server loop.

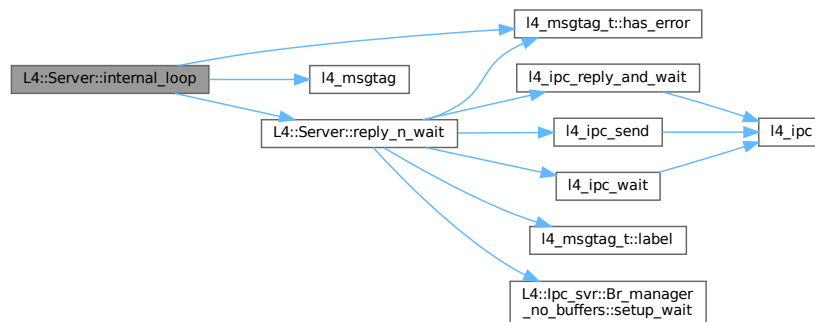
This function usually never returns, it waits for incoming messages calls the dispatcher, sends a reply and waits again.

Definition at line 405 of file [ipc\\_server\\_loop](#).

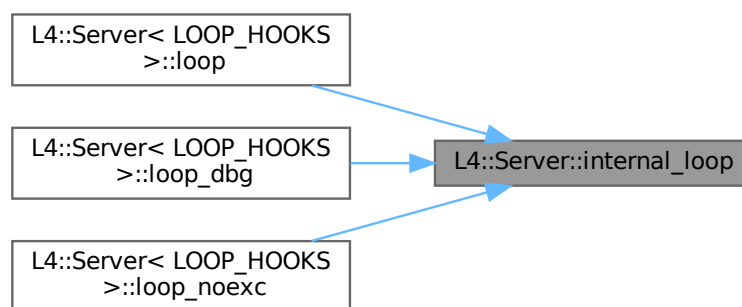
References [l4\\_msgtag\\_t::has\\_error\(\)](#), [L4\\_ENOREPLY](#), [l4\\_msgtag\(\)](#), and [reply\\_n\\_wait\(\)](#).

Referenced by [loop\(\)](#), [loop\\_dbg\(\)](#), and [loop\\_noexc\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.199.3.2 loop()

```

template<typename LOOP_HOOKS = Ipc_svr::Default_loop_hooks>
template<typename EXC, typename R>
L4_NORETURN void L4::Server< LOOP_HOOKS >::loop (
    R r,
    l4_utcb_t * u = l4_utcb()) [inline]
  
```

Server loop with internal exception handling.

This server loop translates [L4::Runtime\\_error](#) exceptions into negative error return codes sent to the caller.

Definition at line 355 of file [ipc\\_server\\_loop](#).

### 16.199.3.3 `loop_dbg()`

```
template<typename LOOP_HOOKS = Ipc_svr::Default_loop_hooks>
template<typename EXC, typename R, typename Printer>
L4_NORETURN void L4::Server< LOOP_HOOKS >::loop_dbg (
    R r,
    Printer p,
    l4_utcb_t * u = l4_utcb()) [inline]
```

[Server](#) loop with internal exception handling including message printing.

Exceptions are translated into error codes, just like in [loop\(\)](#). In addition, error messages are printed using the `Printer` argument.

Definition at line 368 of file [ipc\\_server\\_loop](#).

The documentation for this class was generated from the following file:

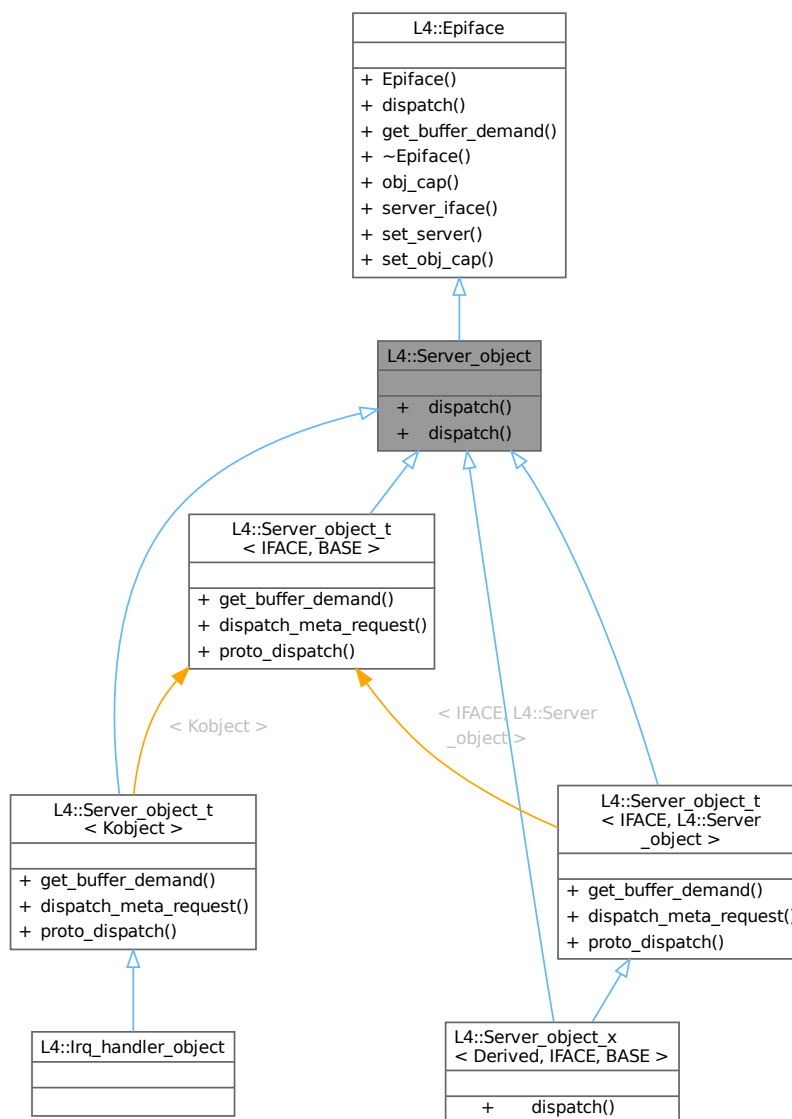
- `l4/sys/cxx/ipc_server_loop`

## 16.200 L4::Server\_object Class Reference

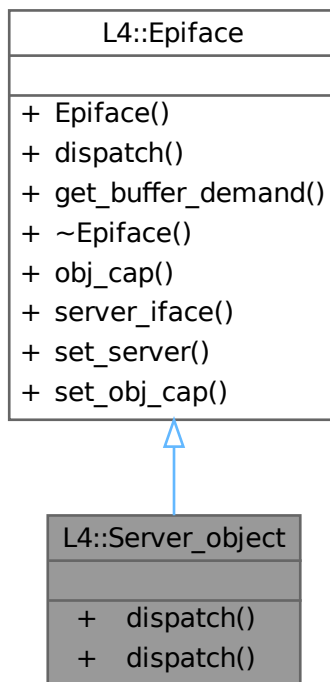
Abstract server object to be used with [L4::Server](#) and [L4::Basic\\_registry](#).

```
#include <ipc_server>
```

Inheritance diagram for L4::Server\_object:



Collaboration diagram for L4::Server\_object:



### Public Member Functions

- virtual int `dispatch` (unsigned long rights, `lpc::lostream` &ios)=0  
*The abstract handler for client requests to the object.*
- `l4_msgtag_t` `dispatch` (`l4_msgtag_t` tag, unsigned rights, `l4_utcb_t` \*utcb) override  
*The abstract handler for client requests to the object.*

### Public Member Functions inherited from **L4::Epiface**

- **Epiface** ()  
*Make a server object.*
- virtual `Demand` `get_buffer_demand` () const =0  
*Get the server-side receive buffer demand for this object.*
- virtual **~Epiface** ()=0  
*Destroy the object.*
- Stored\_cap `obj_cap` () const  
*Get the capability to the kernel object belonging to this object.*
- `Server_iface` \* `server_iface` () const  
*Get pointer to server interface at which the object is currently registered.*
- int `set_server` (`Server_iface` \*srv, `Cap`< void > cap, bool managed=false)  
*Set server registration info for the object.*
- void **set\_obj\_cap** (`Cap`< void > const &cap)  
*Deprecated server registration function.*

## Additional Inherited Members

## Public Types inherited from L4::Epiface

- typedef [lpc\\_svr::Server\\_iface](#) **Server\_iface**  
*Type for abstract server interface.*
- typedef [lpc\\_svr::Server\\_iface::Demand](#) **Demand**  
*Type for server-side receive buffer demand.*

## 16.200.1 Detailed Description

Abstract server object to be used with [L4::Server](#) and [L4::Basic\\_registry](#).

### Note

Usually [L4::Server\\_object\\_t](#) is used as a base class when writing server objects.

This server object provides an abstract interface that is used by the [L4::Registry\\_dispatcher](#) model. You can derive subclasses from this interface and implement application specific server objects.

Definition at line 38 of file [ipc\\_server](#).

## 16.200.2 Member Function Documentation

### 16.200.2.1 dispatch() [1/2]

```
l4_msgtag_t L4::Server_object::dispatch (
    l4_msgtag_t tag,
    unsigned rights,
    l4_utcb_t * utcb) [inline], [override], [virtual]
```

The abstract handler for client requests to the object.

### Parameters

<i>tag</i>	The message tag for this invocation.
<i>rights</i>	The rights bits in the invoked capability.
<i>utcb</i>	The UTCB used for the invocation.

### Return values

<code>-L4_ENOREPLY</code>	No reply message is send.
<code>&lt; 0</code>	Error, reply with error code.
<code>&gt;= 0</code>	Success, reply with return value.

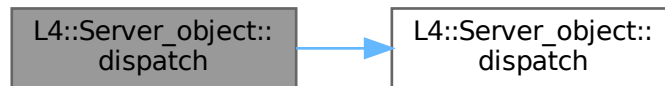
This function must be implemented by application specific server objects.

Implements [L4::Epiface](#).

Definition at line 60 of file [ipc\\_server](#).

References [dispatch\(\)](#).

Here is the call graph for this function:



### 16.200.2.2 `dispatch()` [2/2]

```
virtual int L4::Server_object::dispatch (
    unsigned long rights,
    Ipc::Iostream & ios) [pure virtual]
```

The abstract handler for client requests to the object.

#### Parameters

<i>rights</i>	The rights bits in the invoked capability.
<i>ios</i>	The <a href="#">lpc::lostream</a> for reading the request and writing the reply.

#### Return values

<code>-L4_ENOREPLY</code>	Instructs the server loop to not send a reply.
<code>&lt; 0</code>	Error, reply with error code.
<code>&gt;= 0</code>	Success, reply with return value.

This function must be implemented by application specific server objects. The implementation must unmarshall data from the stream (`ios`) and create a reply by marshalling to the stream (`ios`). For details about the IPC stream see [IPC stream operators](#).

#### Note

You need to extract the complete message from the `ios` stream before inserting any reply data or before doing any function call that may use the UTCB. Otherwise, the incoming message may get lost.

Implemented in [L4::Server\\_object\\_x< Derived, IFACE, BASE >](#).



#### Examples

[examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#), and [examples/libs/l4re/streammap/server.cc](#).

Referenced by [dispatch\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

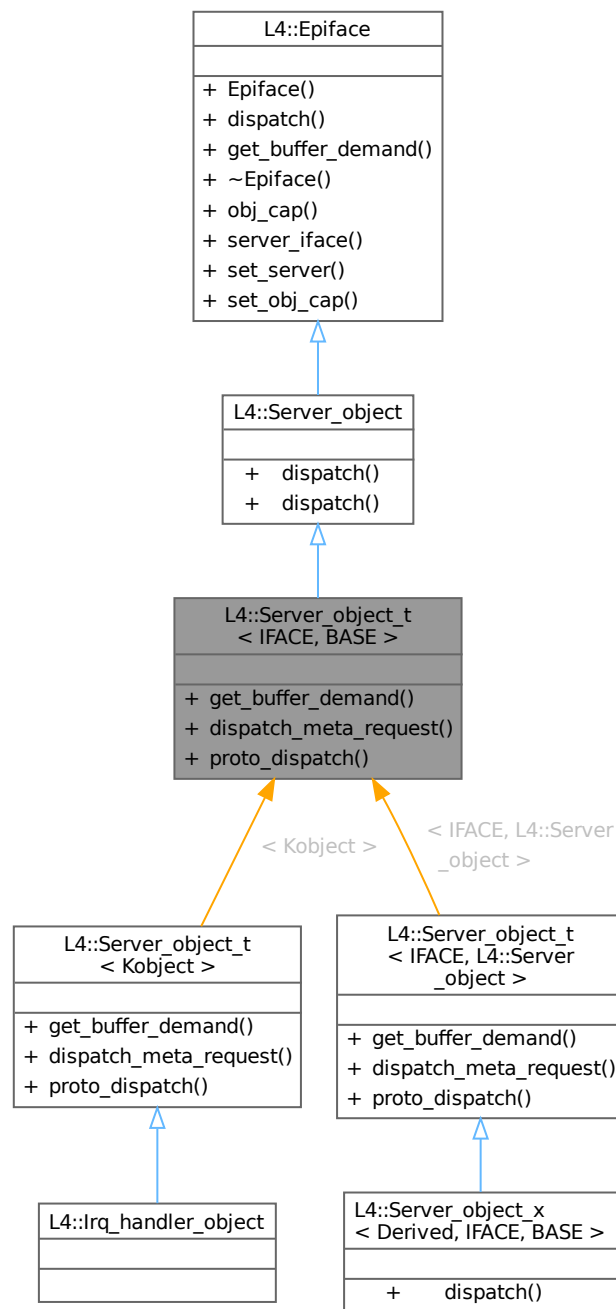
- [l4/cxx/ipc\\_server](#)

## 16.201 L4::Server\_object\_t< IFACE, BASE > Struct Template Reference

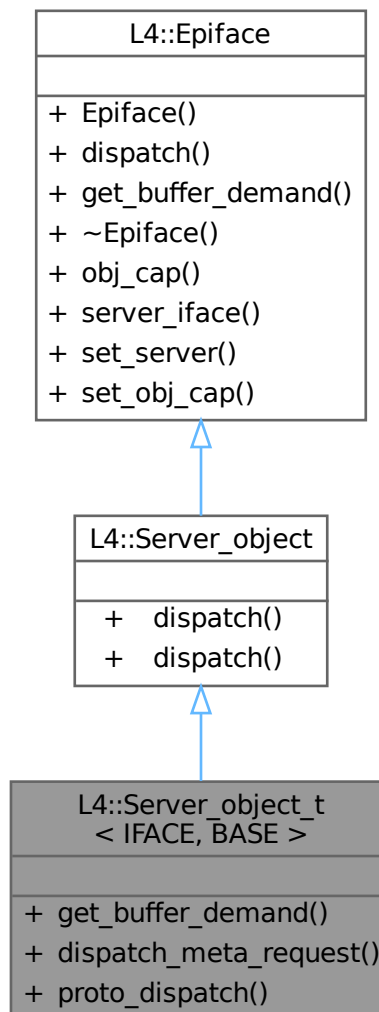
Base class (template) for server implementing server objects.

```
#include <ipc_server>
```

Inheritance diagram for L4::Server\_object\_t< IFACE, BASE >:



Collaboration diagram for L4::Server\_object\_t< IFACE, BASE >:



## Public Types

- typedef IFACE **Interface**  
Data type of the IPC interface definition.

## Public Types inherited from L4::Epiface

- typedef [lpc\\_svr::Server\\_iface](#) **Server\_iface**  
Type for abstract server interface.
- typedef [lpc\\_svr::Server\\_iface::Demand](#) **Demand**  
Type for server-side receive buffer demand.

## Public Member Functions

- `BASE::Demand` `get_buffer_demand` () const override
- `int` `dispatch_meta_request` (`L4::lpc::lostream` &ios)

*Implementation of the meta protocol based on IFACE.*

## Public Member Functions inherited from `L4::Server_object`

- `virtual int` `dispatch` (unsigned long rights, `lpc::lostream` &ios)=0  
*The abstract handler for client requests to the object.*
- `l4_msgtag_t` `dispatch` (`l4_msgtag_t` tag, unsigned rights, `l4_utcb_t` \*utcb) override  
*The abstract handler for client requests to the object.*

## Public Member Functions inherited from `L4::Epiface`

- `Epiface` ()  
*Make a server object.*
- `virtual ~Epiface` ()=0  
*Destroy the object.*
- `Stored_cap` `obj_cap` () const  
*Get the capability to the kernel object belonging to this object.*
- `Server_iface` \* `server_iface` () const  
*Get pointer to server interface at which the object is currently registered.*
- `int` `set_server` (`Server_iface` \*srv, `Cap`< void > cap, bool managed=false)  
*Set server registration info for the object.*
- `void` `set_obj_cap` (`Cap`< void > const &cap)  
*Deprecated server registration function.*

## Static Public Member Functions

- `template<typename THIS>`  
`static int` `proto_dispatch` (`THIS` \*self, `l4_umword_t` rights, `L4::lpc::lostream` &ios)  
*Implementation of protocol-based dispatch for this server object.*

### 16.201.1 Detailed Description

```
template<typename IFACE, typename BASE = L4::Server_object>
struct L4::Server_object_t< IFACE, BASE >
```

Base class (template) for server implementing server objects.

## Template Parameters

<code>IFACE</code>	The IPC interface class that defines the interface that shall be implemented.
<code>BASE</code>	The server object base class (usually <code>L4::Server_object</code> ).

## Examples

`examples/libs/l4re/c++/shared_ds/ds_srv.cc`, and `examples/libs/l4re/streammap/server.cc`.

Definition at line 80 of file `ipc_server`.

## 16.201.2 Member Function Documentation

### 16.201.2.1 dispatch\_meta\_request()

```
template<typename IFACE, typename BASE = L4::Server_object>
int L4::Server_object_t< IFACE, BASE >::dispatch_meta_request (
    L4::Ipc::Iostream & ios) [inline]
```

Implementation of the meta protocol based on *IFACE*.

#### Parameters

<i>ios</i>	The IO stream used for receiving the message.
------------	-----------------------------------------------

This function can be used to handle incoming [L4\\_PROTO\\_META](#) protocol requests. The implementation uses the [L4::Type\\_info](#) of *IFACE* to handle the requests. Call this function in the implementation of [Server\\_object::dispatch\(\)](#) when the received message tag has protocol [L4\\_PROTO\\_META](#) ([L4::Meta::Protocol](#)).

Definition at line 99 of file [ipc\\_server](#).

### 16.201.2.2 get\_buffer\_demand()

```
template<typename IFACE, typename BASE = L4::Server_object>
BASE::Demand L4::Server_object_t< IFACE, BASE >::get_buffer_demand () const [inline], [override],
[virtual]
```

#### Returns

the server-side buffer demand based in *IFACE*.

Implements [L4::Epiface](#).

Definition at line 86 of file [ipc\\_server](#).

### 16.201.2.3 proto\_dispatch()

```
template<typename IFACE, typename BASE = L4::Server_object>
template<typename THIS>
int L4::Server_object_t< IFACE, BASE >::proto_dispatch (
    THIS * self,
    l4_umword_t rights,
    L4::Ipc::Iostream & ios) [inline], [static]
```

Implementation of protocol-based dispatch for this server object.

#### Parameters

<i>self</i>	The this pointer for the object (inherits from <a href="#">Server_object_t</a> ).
-------------	-----------------------------------------------------------------------------------

<i>rights</i>	The rights from the received IPC (forwarded to p_dispatch()).
<i>ios</i>	The message stream for the incoming and the reply message.

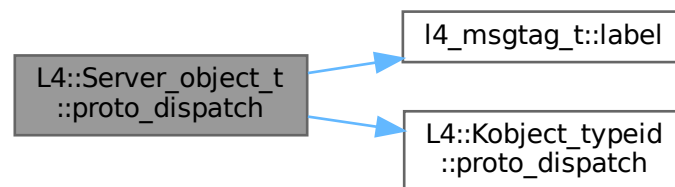
[Server](#) objects may call this function from their [dispatch\(\)](#) function. This function reads the protocol ID from the message tag and uses the p\_dispatch code to dispatch to overloaded p\_dispatch functions of self.

Definition at line 114 of file [ipc\\_server](#).

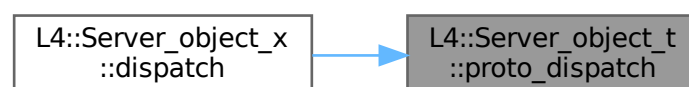
References [l4\\_msgtag\\_t::label\(\)](#), and [L4::Kobject\\_typeid< T >::proto\\_dispatch\(\)](#).

Referenced by [L4::Server\\_object\\_x< Derived, IFACE, BASE >::dispatch\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this struct was generated from the following file:

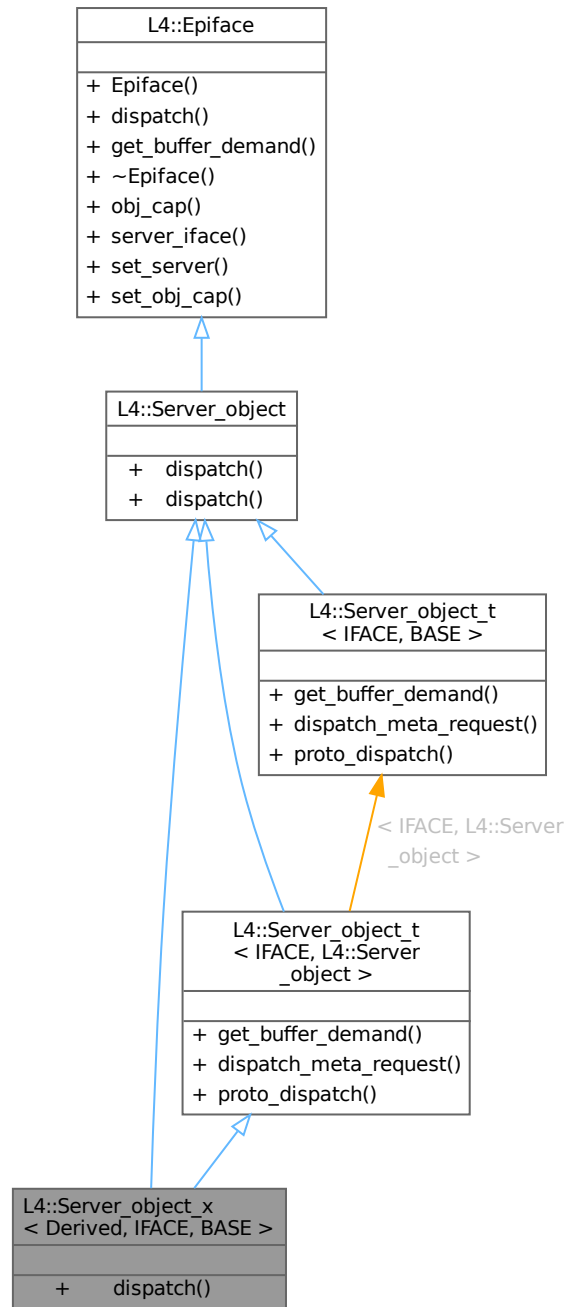
- [l4/cxx/ipc\\_server](#)

## 16.202 L4::Server\_object\_x< Derived, IFACE, BASE > Struct Template Reference

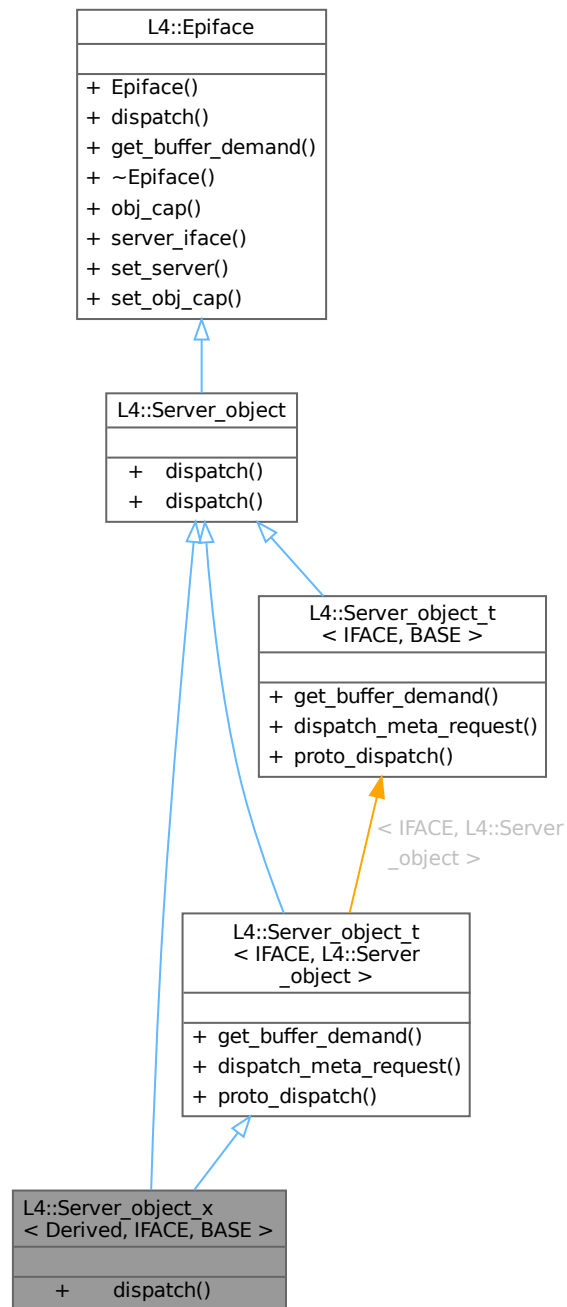
Helper class to implement p\_dispatch based server objects.

```
#include <ipc_server>
```

Inheritance diagram for L4::Server\_object\_x< Derived, IFACE, BASE >:



Collaboration diagram for L4::Server\_object\_x< Derived, IFACE, BASE >:



## Public Member Functions

- `int dispatch (l4_umword_t r, L4::lpc::lostream &ios)`  
Implementation forwarding to `p_dispatch()`.

## Public Member Functions inherited from L4::Server\_object

- `l4_msgtag_t dispatch (l4_msgtag_t tag, unsigned rights, l4_utcb_t *utcb)` override



*The abstract handler for client requests to the object.*

## Public Member Functions inherited from L4::Epiface

- **Epiface** ()  
*Make a server object.*
- virtual **~Epiface** ()=0  
*Destroy the object.*
- Stored\_cap **obj\_cap** () const  
*Get the capability to the kernel object belonging to this object.*
- **Server\_iface** \* **server\_iface** () const  
*Get pointer to server interface at which the object is currently registered.*
- int **set\_server** (**Server\_iface** \*srv, **Cap**< void > cap, bool managed=false)  
*Set server registration info for the object.*
- void **set\_obj\_cap** (**Cap**< void > const &cap)  
*Deprecated server registration function.*

## Public Member Functions inherited from L4::Server\_object\_t< IFACE, L4::Server\_object >

- L4::Server\_object::Demand **get\_buffer\_demand** () const override
- int **dispatch\_meta\_request** (L4::lpc::lostream &ios)  
*Implementation of the meta protocol based on IFACE.*

## Additional Inherited Members

## Public Types inherited from L4::Epiface

- typedef **lpc\_svr::Server\_iface** **Server\_iface**  
*Type for abstract server interface.*
- typedef **lpc\_svr::Server\_iface::Demand** **Demand**  
*Type for server-side receive buffer demand.*

## Public Types inherited from L4::Server\_object\_t< IFACE, L4::Server\_object >

- typedef IFACE **Interface**  
*Data type of the IPC interface definition.*

## Static Public Member Functions inherited from L4::Server\_object\_t< IFACE, L4::Server\_object >

- static int **proto\_dispatch** (THIS \*self, **l4\_umword\_t** rights, L4::lpc::lostream &ios)  
*Implementation of protocol-based dispatch for this server object.*

## 16.202.1 Detailed Description

```
template<typename Derived, typename IFACE, typename BASE = L4::Server_object>
struct L4::Server_object_x< Derived, IFACE, BASE >
```

Helper class to implement p\_dispatch based server objects.

### Template Parameters

<i>Derived</i>	The data type of your server object class.
<i>IFACE</i>	The data type providing the interface definition for the object.
<i>BASE</i>	Optional data-type of the base server object (usually <a href="#">L4::Server_object</a> )

This class implements the standard [dispatch\(\)](#) function of [L4::Server\\_object](#) and forwards incoming messages to a set of overloaded `p_dispatch()` functions. There must be a `p_dispatch()` function in *Derived* for each interface provided by *IFACE* with the signature

```
int p_dispatch(Iface *, unsigned rights, L4::Ipc::Iostream &)
```

that is called for messages with `protocol == Iface::Protocol`.

Example signature for [L4Re::Dataspace](#) is:

```
int p_dispatch(L4Re::Dataspace *, unsigned, L4::Ipc::Iostream &)
```

Definition at line [143](#) of file [ipc\\_server](#).

The documentation for this struct was generated from the following file:

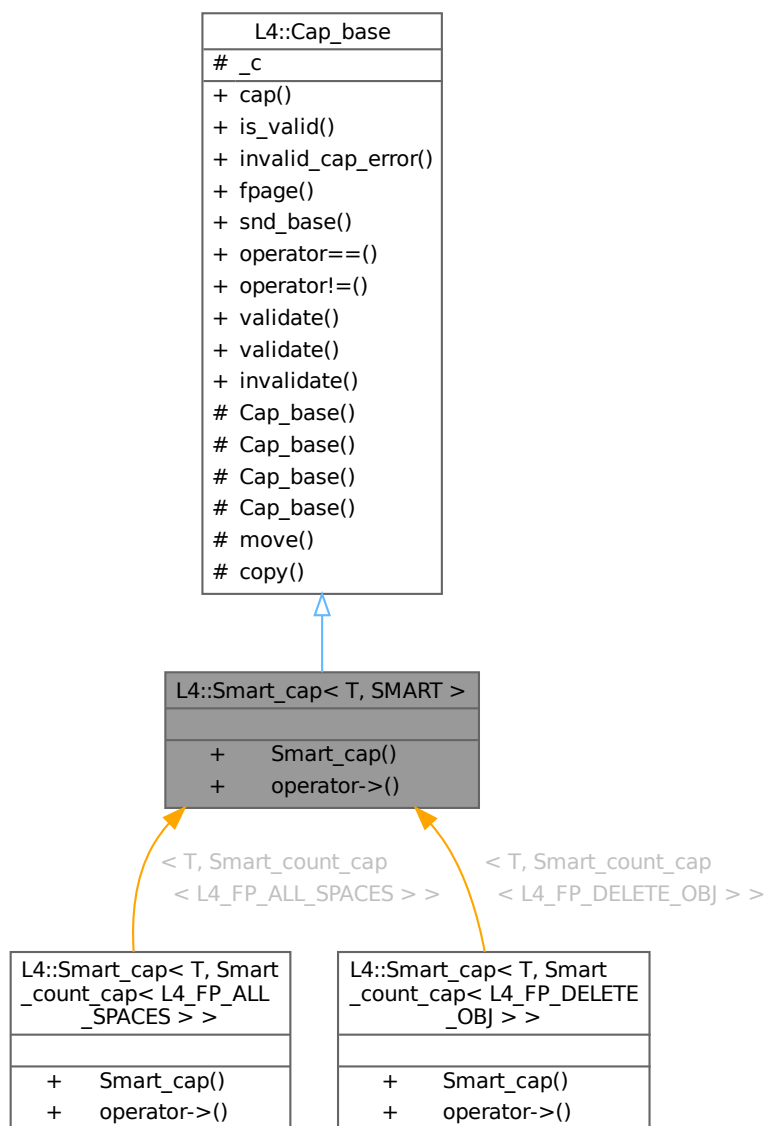
- [l4/cxx/ipc\\_server](#)

## 16.203 L4::Smart\_cap< T, SMART > Class Template Reference

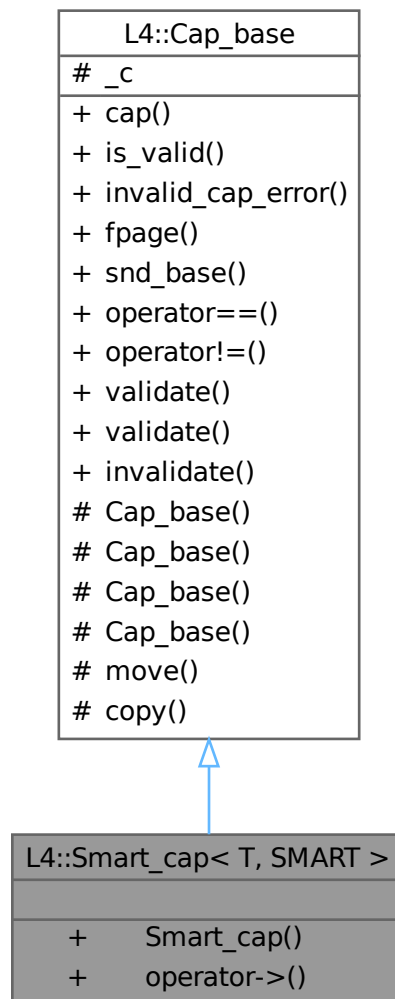
Smart capability class.

```
#include <smart_capability>
```

Inheritance diagram for L4::Smart\_cap< T, SMART >:



Collaboration diagram for L4::Smart\_cap< T, SMART >:



## Public Member Functions

- `template<typename O>`  
`Smart_cap (Cap< O > const &p) noexcept`  
*Internal constructor, use to generate a capability from a `this` pointer.*
- `Cap< T > operator-> () const noexcept`  
*Member access of a `T`.*

## Public Member Functions inherited from L4::Cap\_base

- `l4_cap_idx_t cap () const noexcept`  
*Return capability selector.*
- `bool is_valid () const noexcept`

- Test whether the capability is a valid capability index (i.e., not L4\_INVALID\_CAP).*

  - `int invalid_cap_error ()` `const noexcept`

*Return the transported error code in an invalid capability index.*
- `l4_fpage_t fpage` (unsigned rights=L4\_CAP\_FPAGE\_RWS) `const noexcept`

*Return flexpage for the capability.*
- `l4_umword_t snd_base` (unsigned grant=L4\_MAP\_ITEM\_MAP, `l4_cap_idx_t` base=L4\_INVALID\_CAP) `const noexcept`

*Return send base.*
- `bool operator== (Cap_base const &o)` `const noexcept`

*Test if two capabilities are equal.*
- `bool operator!= (Cap_base const &o)` `const noexcept`

*Test if two capabilities are not equal.*
- `l4_msgtag_t validate (l4_utcb_t *u=l4_utcb())` `const noexcept`

*Check whether a capability is present (refers to an object).*
- `l4_msgtag_t validate (Cap< Task > task, l4_utcb_t *u=l4_utcb())` `const noexcept`

*Check whether a capability is present (refers to an object).*
- `void invalidate ()` `noexcept`

*Set this capability to invalid (L4\_INVALID\_CAP).*

### Additional Inherited Members

### Public Types inherited from L4::Cap\_base

- `enum No_init_type { No_init }`
- Special value for uninitialized capability objects.*
- `enum Cap_type { Invalid = L4_INVALID_CAP }`
- Invalid capability type.*

### Protected Member Functions inherited from L4::Cap\_base

- `Cap_base (l4_cap_idx_t c)` `noexcept`
- Generate a capability from its C representation.*
- `Cap_base (Cap_type cap)` `noexcept`
- Constructor to create an invalid capability.*
- `Cap_base (l4_default_caps_t cap)` `noexcept`
- Initialize capability with one of the default capabilities.*
- `Cap_base ()` `noexcept`
- Create an uninitialized instance.*
- `void move (Cap_base const &src)` `const`
- Replace this capability with the contents of `src`.*
- `void copy (Cap_base const &src)` `const`
- Copy a capability.*

### Protected Attributes inherited from L4::Cap\_base

- `l4_cap_idx_t _c`
- The C representation of a capability selector.*

### 16.203.1 Detailed Description

```
template<typename T, typename SMART>
class L4::Smart_cap< T, SMART >
```

Smart capability class.

Definition at line 25 of file [smart\\_capability](#).

### 16.203.2 Constructor & Destructor Documentation

#### 16.203.2.1 Smart\_cap()

```
template<typename T, typename SMART>
template<typename O>
L4::Smart_cap< T, SMART >::Smart_cap (
    Cap< O > const & p) [inline], [noexcept]
```

Internal constructor, use to generate a capability from a `this` pointer.

#### Attention

This constructor is only useful to generate a capability from the `this` pointer of an objected that is an [L4::Kobject](#). Do *never* use this constructor for something else!

#### Parameters

<i>p</i>	The <code>this</code> pointer of the <a href="#">Kobject</a> or derived object
----------	--------------------------------------------------------------------------------

Definition at line 62 of file [smart\\_capability](#).

The documentation for this class was generated from the following file:

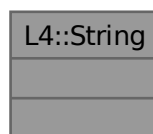
- [l4/sys/smart\\_capability](#)

## 16.204 L4::String Class Reference

A null-terminated string container class.

```
#include <string.h>
```

Collaboration diagram for L4::String:



### 16.204.1 Detailed Description

A null-terminated string container class.

Definition at line 22 of file [string.h](#).

The documentation for this class was generated from the following file:

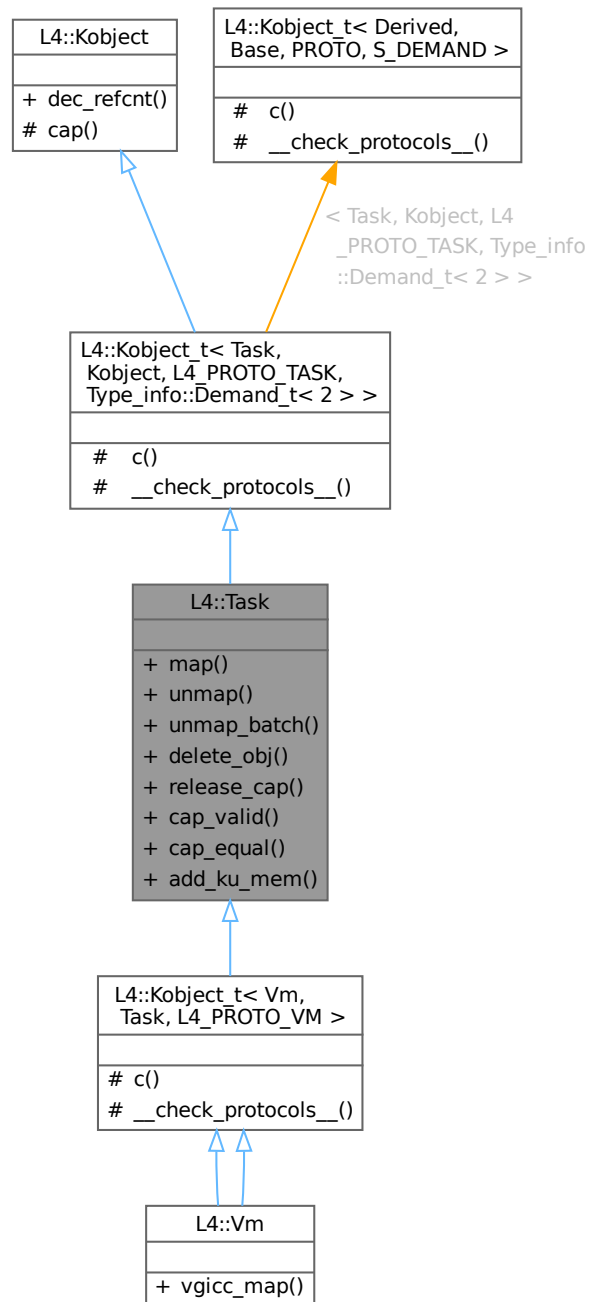
- [l4/cxx/string.h](#)

## 16.205 L4::Task Class Reference

C++ interface of the [Task](#) kernel object, see [Task](#) for the C interface.

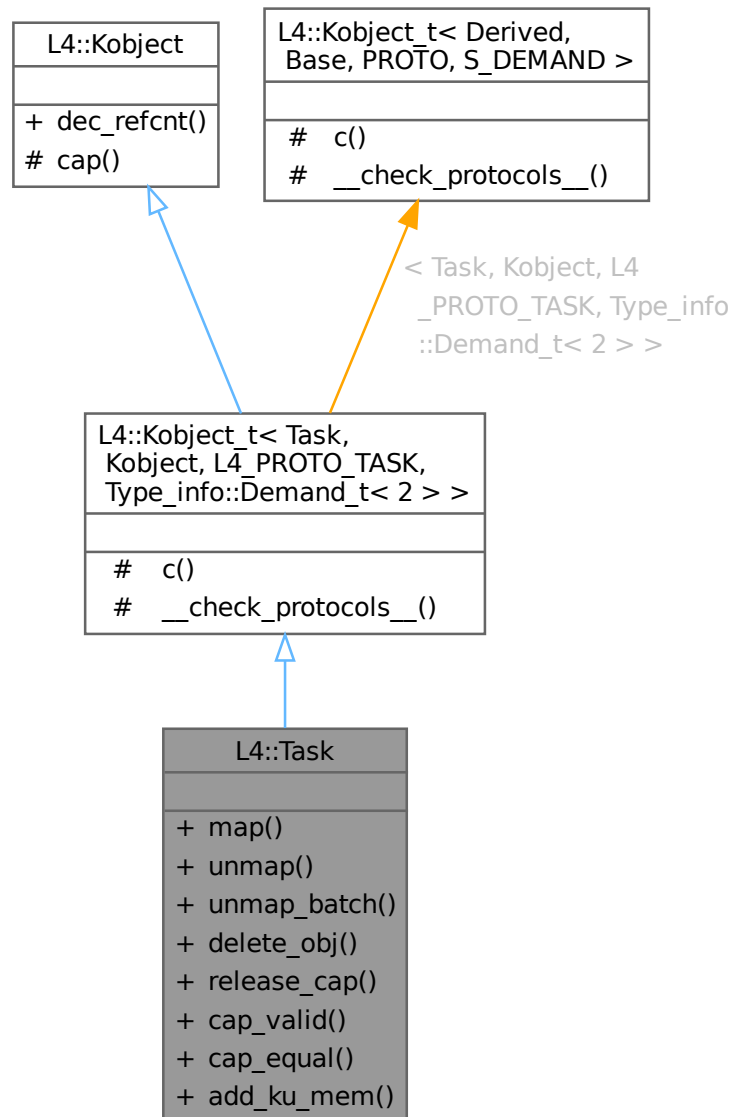
```
#include <task>
```

Inheritance diagram for L4::Task:





Collaboration diagram for L4::Task:



## Public Member Functions

- `l4_msgtag_t map (Cap< Task > const &src_task, l4_fpage_t const &snd_fpage, l4_umword_t snd_base, l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Map resources available in the source task to a destination task.*
- `l4_msgtag_t unmap (l4_fpage_t const &fpage, l4_umword_t map_mask, l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Revoke rights from the task.*
- `l4_msgtag_t unmap_batch (l4_fpage_t const *fpages, unsigned num_fpages, l4_umword_t map_mask, l4_utcb_t *utcb=l4_utcb()) noexcept`  
*Revoke rights from a task.*
- `l4_msgtag_t delete_obj (L4::Cap< void > obj, l4_utcb_t *utcb=l4_utcb()) noexcept`

*Release capability and delete object.*

- `l4_msgtag_t release_cap (L4::Cap< void > cap, l4_utcb_t *utcb=l4_utcb()) noexcept`

*Release object capability.*

- `l4_msgtag_t cap_valid (Cap< void > const &cap, l4_utcb_t *utcb=l4_utcb()) noexcept`

*Check whether a capability is present (refers to an object).*

- `l4_msgtag_t cap_equal (Cap< void > const &cap_a, Cap< void > const &cap_b, l4_utcb_t *utcb=l4_utcb()) noexcept`

*Test whether two capabilities point to the same object with the same permissions (only considering selected permissions).*

- `l4_msgtag_t add_ku_mem (l4_fpage_t *fpage, l4_utcb_t *utcb=l4_utcb()) noexcept`

*Add kernel-user memory.*

## Public Member Functions inherited from `L4::Kobject`

- `l4_msgtag_t dec_refcnt (l4_mword_t diff, l4_utcb_t *utcb=l4_utcb())`

*Decrement the in kernel reference counter for the object.*

## Additional Inherited Members

## Protected Types inherited from

`L4::Kobject_t< Task, Kobject, L4_PROTO_TASK, Type_info::Demand_t< 2 > >`

- typedef `Task` **Class**

*The target interface type (inheriting from `Kobject_t`).*

- typedef `Typeid::Iface< PROTO, Task > __iface`

*The interface description for the derived class.*

- typedef `Typeid::Merge_list< Typeid::Iface_list< __iface >, typename Kobject::__iface_list > __iface_list`

*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Member Functions inherited from

`L4::Kobject_t< Task, Kobject, L4_PROTO_TASK, Type_info::Demand_t< 2 > >`

- `L4::Cap< Class > c () const noexcept`

*Get the capability to ourselves.*

## Protected Member Functions inherited from `L4::Kobject`

- `l4_cap_idx_t cap () const noexcept`

*Return capability selector.*

## Static Protected Member Functions inherited from

`L4::Kobject_t< Task, Kobject, L4_PROTO_TASK, Type_info::Demand_t< 2 > >`

- static void `__check_protocols__ () noexcept`

*Helper to check for protocol conflicts.*

## 16.205.1 Detailed Description

C++ interface of the [Task](#) kernel object, see [Task](#) for the C interface.

The [L4::Task](#) class represents a combination of the address spaces provided by the [L4Re](#) micro kernel. A task consists of at least a memory address space and an object address space. On IA32 there is also an IO-port address space associated with an [L4::Task](#).

[L4::Task](#) objects are created using the [L4::Factory](#) interface.

### Include File

```
#include <l4/sys/task>
```

Definition at line 33 of file [task](#).

## 16.205.2 Member Function Documentation

### 16.205.2.1 add\_ku\_mem()

```
l4_msgtag_t L4::Task::add_ku_mem (
    l4_fpage_t * fpage,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Add kernel-user memory.

#### Parameters

<i>in, out</i>	<i>fpage</i>	Flexpage describing the virtual area the memory goes to. On systems without MMU, the flexpage is adjusted to reflect the actually allocated physical address.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

#### Returns

Syscall return tag

Kernel-user memory (ku\_mem) is memory that is shared between the kernel and user-space. It is needed for the UTCB area of threads (see [L4::Thread::Attr::bind\(\)](#)) and for (extended) vCPU state. Note that existing kernel-user memory cannot be unmapped or mapped somewhere else.

**Note**

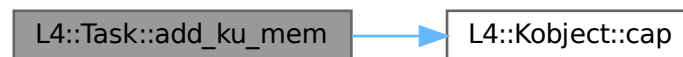
The amount of kernel-user memory that can be allocated at once is limited by the used kernel implementation. The minimum allocatable amount is one page ([L4\\_PAGESIZE](#)). A portable implementation should not depend on allocations greater than 16KiB to succeed.

This function is only guaranteed to work on [L4::Task](#) objects. It might or might not work on [L4::Vm](#) objects or on [L4Re::Dma\\_space](#) objects but there is no practical use for adding kernel-user memory to [L4::Vm](#) objects or to [L4Re::Dma\\_space](#) objects.

Definition at line 281 of file [task](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:

**16.205.2.2 cap\_equal()**

```

l4_msgtag_t L4::Task::cap_equal (
    Cap< void > const & cap_a,
    Cap< void > const & cap_b,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Test whether two capabilities point to the same object with the same permissions (only considering selected permissions).

**Parameters**

<i>cap<sub>a</sub></i>	Capability selector for the first capability to compare.
<i>cap<sub>b</sub></i>	Capability selector for the second capability to compare.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

**Return values**

<i>l4_msgtag_t::label()</i> = 1	The compared capabilities point to the same object with same considered permission.
<i>l4_msgtag_t::label()</i> = 0	The compared capabilities do <b>not</b> point to the same object or differ in the considered permission.

- For [L4::lpc\\_gate](#) objects, only the permissions [L4\\_CAP\\_FPAGE\\_W](#), [L4\\_CAP\\_FPAGE\\_S](#), and [L4\\_FPAGE\\_C\\_OBJ\\_RIGHT1](#) are considered for the comparison. Differences in other permissions are ignored.

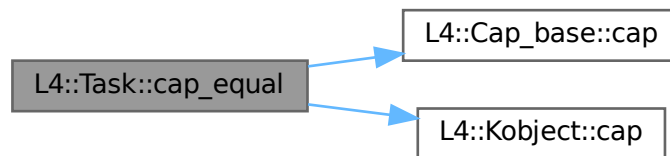
- For other objects, only the permissions [L4\\_CAP\\_FPAGE\\_W](#) and [L4\\_CAP\\_FPAGE\\_S](#) are considered for the comparison. Differences in other permissions are ignored.

Note that having the [L4\\_CAP\\_FPAGE\\_R](#) permission is implicit in possessing the capability.

Definition at line 251 of file [task](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 16.205.2.3 cap\_valid()

```

14_msgtag_t L4::Task::cap_valid (
    Cap< void > const & cap,
    14_utcb_t * utcb = 14_utcb()) [inline], [noexcept]
  
```

Check whether a capability is present (refers to an object).

#### Parameters

<i>cap</i>	Valid capability to check for presence.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">14_utcb</a> .

#### Return values

<i>14_msgtag_t::label()</i> > 0	Capability is present (refers to an object).
<i>14_msgtag_t::label()</i> == 0	No capability present (void object).

A capability is considered present when it refers to an existing kernel object.

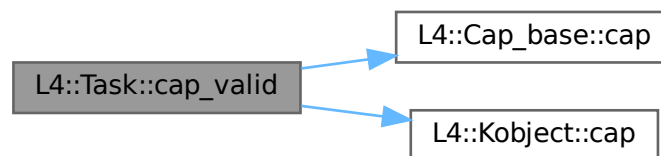
**Precondition**

`cap` must be a valid capability (i.e. `cap.is_valid() == true`). If you are unsure about the validity of your capability use [L4::Cap.validate\(\)](#) instead.

Definition at line 222 of file [task](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:

**16.205.2.4 delete\_obj()**

```

l4_msgtag_t L4::Task::delete_obj (
    L4::Cap< void > obj,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Release capability and delete object.

**Parameters**

<i>obj</i>	Capability index of the object to delete.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

**Returns**

Syscall return tag

If `obj` has the delete permission, initiates the deletion of the object. This implies that all capabilities for that object are gone afterwards. However, kernel-internally, objects are not destroyed until all other kernel objects holding a reference to it drop the reference. Hence, quota used by that object might not be freed immediately.

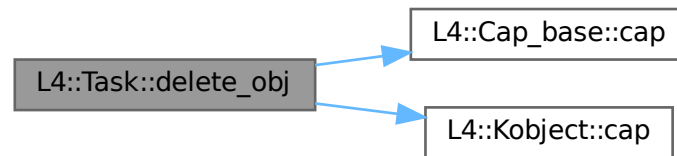
If `obj` does not have the delete permission, no error will be reported and only the capability `obj` is removed. (Note that, depending on the object's reference counter, this might still imply initiation of deletion.)

This operation is equivalent to [unmap\(\)](#) with [L4\\_FP\\_DELETE\\_OBJ](#) flag.

Definition at line 180 of file [task](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 16.205.2.5 map()

```

l4_msgtag_t L4::Task::map (
    Cap< Task > const & src_task,
    l4_fpage_t const & snd_fpage,
    l4_umword_t snd_base,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]

```

Map resources available in the source task to a destination task.

#### Parameters

<i>src_task</i>	Capability selector of the source task.
<i>snd_fpage</i>	Send flexpage that describes an area in the address space or object space of the source task.
<i>snd_base</i>	Send base that describes an offset in the receive window of the destination task. The lower bits contain additional map control flags (see <a href="#">l4_fpage_cacheability_opt_t</a> for memory mappings, <a href="#">L4_obj_fpage_ctl</a> for object mappings, and <a href="#">L4_MAP_ITEM_GRANT</a> ; also see <a href="#">l4_map_control()</a> and <a href="#">l4_map_obj_control()</a> ).
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

#### Returns

Syscall return tag. The function [l4\\_error\(\)](#) shall be used to test if the map operation was successful.

#### Return values

<i>L4_EOK</i>	Operation successful (but see notes below).
<i>-L4_EPERM</i>	Insufficient permissions; see precondition.
<i>-L4_EINVAL</i>	Invalid source task capability.
<i>-L4_IPC_SEMAPFAILED</i>	The map operation failed due to limited quota.

**Precondition**

The invoked [Task](#) capability must have the permission [L4\\_CAP\\_FPAGE\\_W](#).

This method allows for asynchronous transfer of capabilities, memory mappings, and IO-port mappings (on IA32) from one task to another. The destination task is the task referenced by the capability on which the map is invoked, and the receive window is the whole address space of that task. By specifying proper rights in the `snd_fpage` and `snd_base`, it is possible to remove rights during transfer.

**Note**

If the send flexpage is of type [L4\\_FPAGE\\_OBJ](#), the [L4\\_CAP\\_FPAGE\\_S](#) right is removed from the transferred capability unless both the source and destination task capabilities possess the [L4\\_CAP\\_FPAGE\\_S](#) right themselves.

Even with [l4\\_error\(\)](#) returning `L4_EOK` there might be cases where not all pages of the send flexpage were mapped respectively granted to the destination task, for instance, if the corresponding mapping in the destination task does already exist.

For more information on spaces and mappings, see [Spaces and Mappings](#). The flexpage API is described in more detail at [Flexpages](#).

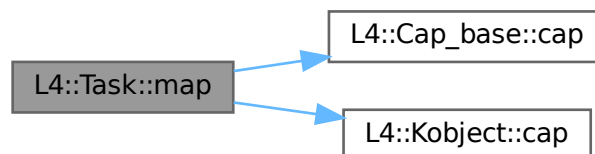
**Note**

For peculiarities when using grant, see [L4\\_MAP\\_ITEM\\_GRANT](#).

Definition at line 85 of file [task](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:

**16.205.2.6 release\_cap()**

```

l4_msgtag_t L4::Task::release_cap (
    L4::Cap< void > cap,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Release object capability.

**Parameters**



<i>cap</i>	Capability selector of the object to release.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

**Returns**

Syscall return tag.

This operation unmaps the capability from `this` task. This operation is equivalent to unmapping a single object capability by specifying all object rights as unmap mask.

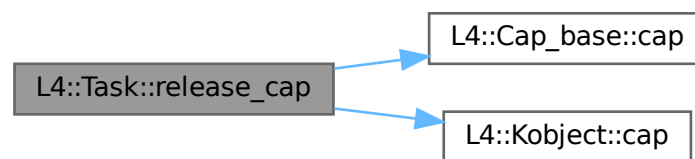
**Note**

If the reference counter of the kernel object referenced by [cap](#) goes down to zero, the deletion of the object is initiated. Objects are not destroyed until all other kernel objects holding a reference to it drop the reference.

Definition at line 201 of file [task](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:

**16.205.2.7 unmap()**

```

14_msgtag_t L4::Task::unmap (
    14_fpage_t const & fpage,
    14_umword_t map_mask,
    14_utcb_t * utcb = 14_utcb()) [inline], [noexcept]

```

Revoke rights from the task.

**Parameters**

<i>fpage</i>	Flexpage that describes an area in one capability space of <code>this</code> task and the rights to revoke.
<i>map_mask</i>	Unmap mask, see <a href="#">l4_unmap_flags_t</a>
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

## Returns

Syscall return tag

This method allows to revoke rights from the destination task. The rights to revoke are specified in the flexpage, see [l4\\_fpage\\_rights\(\)](#). For a flexpage describing IO ports or memory, it also revokes rights from all the tasks that got the rights delegated from the destination task (i.e., this operation does a recursive rights revocation). The capability is unmapped if certain rights are specified, see below for details. It is guaranteed that the rights revocation is completed before this function returns.

Note that this function cannot be used to revoke the reference counting permission (see [L4\\_FPAGE\\_C\\_REF\\_CNT](#)) or the IPC-gate server permission (see [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#)) from object capabilities.

It depends on the platform and the object type which rights need to be specified in the `rights` field of `fpage` to unmap a capability:

- An object capability is unmapped if and only if the [L4\\_CAP\\_FPAGE\\_R](#) right bit is set.
- An IO port is unmapped if and only if any right bit is set.
- Memory is unmapped if and only if the [L4\\_FPAGE\\_RO](#) right bit is set.

## Note

Depending on the page-table features supported by the hardware, revocation of certain rights from a memory capability can be a no-op (i.e., the rights are not revoked). Further, revocation of certain rights may grant other rights which were not present before. For instance, on an architecture without support for NX, revoking X does nothing. For another example, revoking only X from an execute-only page grants read permission (because the mapping remains present in the page table).

If the reference counter of a kernel object referenced in `fpage` goes down to zero (as a result of deleting capabilities), the deletion of the object is initiated. Objects are not destroyed until all other kernel objects holding a reference to it drop the reference.

Definition at line 135 of file [task](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 16.205.2.8 unmap\_batch()

```

l4_msgtag_t L4::Task::unmap_batch (
    l4_fpage_t const * fpages,
    unsigned num_fpages,
    l4_umword_t map_mask,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Revoke rights from a task.

## Parameters

<i>fpages</i>	An array of flexpages. Each item describes an area in one capability space of <code>this</code> task.
<i>num_fpages</i>	Number of fpages in the <code>fpages</code> array.
<i>map_mask</i>	Unmap mask, see <a href="#">l4_unmap_flags_t</a> .
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

Revoke rights for an array of flexpages, see [unmap](#) for details.

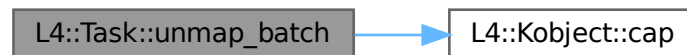
#### Precondition

The caller needs to take care that `num_fpages` is not bigger than `L4_UTCB_GENERIC_DATA_SIZE - 2`.

Definition at line 154 of file [task](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

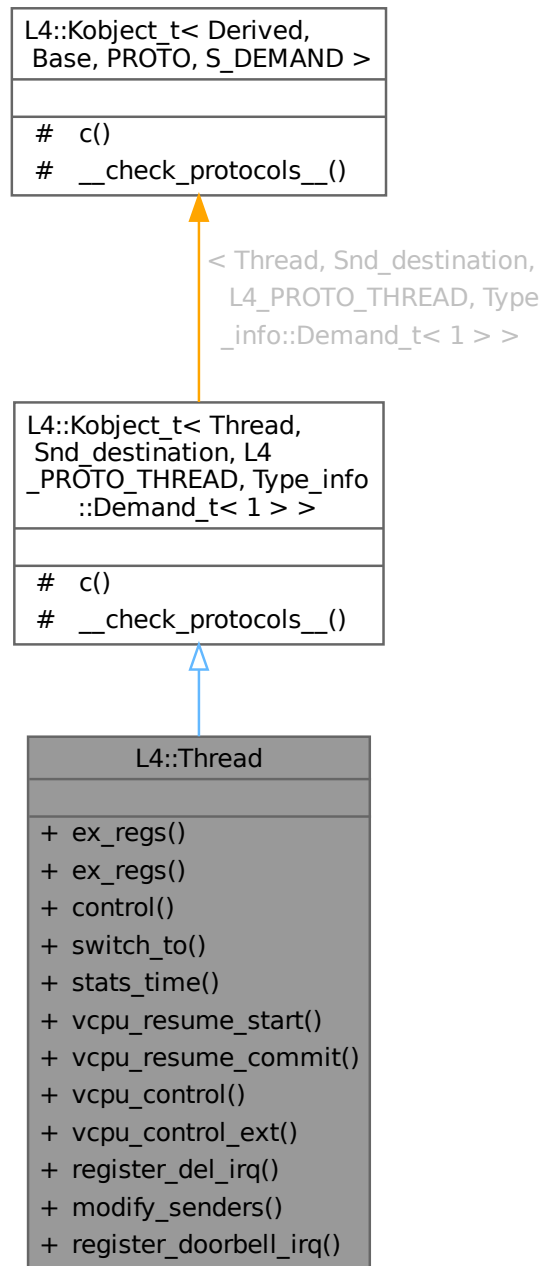
- [l4/sys/task](#)

## 16.206 L4::Thread Class Reference

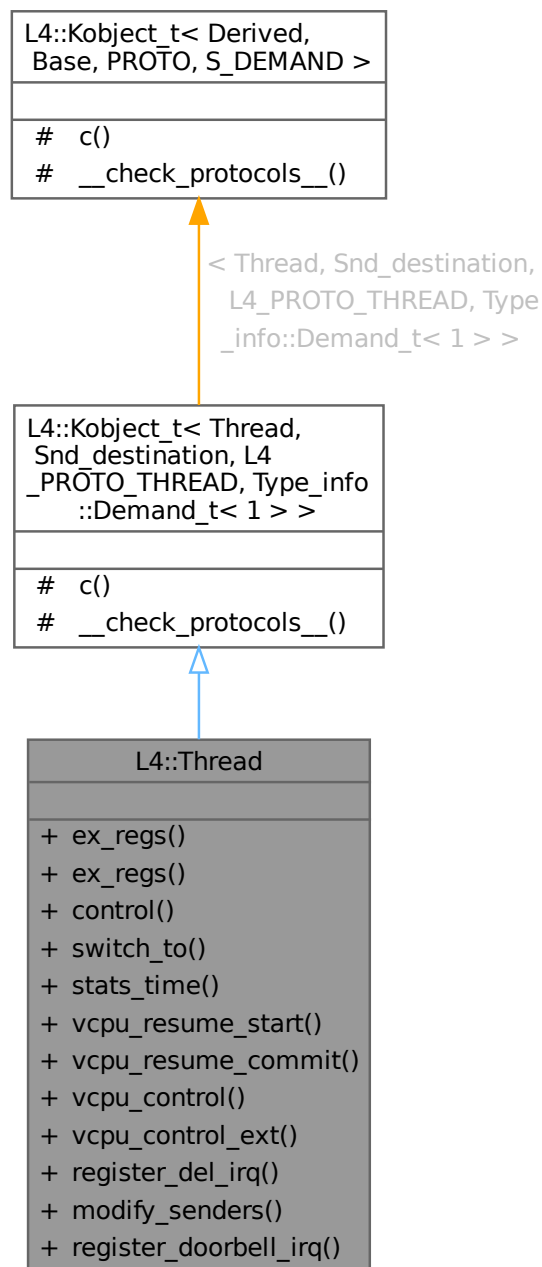
C++ [L4](#) kernel thread interface, see [Thread](#) for the C interface.

```
#include <thread>
```

Inheritance diagram for L4::Thread:



Collaboration diagram for L4::Thread:



## Data Structures

- class [Attr](#)  
*Thread attributes used for `control()`.*
- class [Modify\\_senders](#)  
*Class wrapping a list of rules which modify the sender label of IPC messages inbound to this thread.*

## Public Member Functions

- [l4\\_msgtag\\_t ex\\_regs](#) ([l4\\_addr\\_t ip](#), [l4\\_addr\\_t sp](#), [l4\\_umword\\_t flags](#), [l4\\_utcb\\_t \\*utcb=l4\\_utcb\(\)](#)) noexcept  
*Exchange basic thread registers.*
- [l4\\_msgtag\\_t ex\\_regs](#) ([l4\\_addr\\_t \\*ip](#), [l4\\_addr\\_t \\*sp](#), [l4\\_umword\\_t \\*flags](#), [l4\\_utcb\\_t \\*utcb=l4\\_utcb\(\)](#)) noexcept  
*Exchange basic thread registers and return previous values.*
- [l4\\_msgtag\\_t control](#) ([Attr](#) const &attr) noexcept  
*Commit the given thread-attributes object.*
- [l4\\_msgtag\\_t switch\\_to](#) ([l4\\_utcb\\_t \\*utcb=l4\\_utcb\(\)](#)) noexcept  
*Switch execution to this thread.*
- [l4\\_msgtag\\_t stats\\_time](#) ([l4\\_kernel\\_clock\\_t \\*us](#), [l4\\_utcb\\_t \\*utcb=l4\\_utcb\(\)](#)) noexcept  
*Get consumed time of thread in us.*
- [l4\\_msgtag\\_t vcpu\\_resume\\_start](#) ([l4\\_utcb\\_t \\*utcb=l4\\_utcb\(\)](#)) noexcept  
*Resume from vCPU asynchronous IPC handler, start.*
- [l4\\_msgtag\\_t vcpu\\_resume\\_commit](#) ([l4\\_msgtag\\_t tag](#), [l4\\_utcb\\_t \\*utcb=l4\\_utcb\(\)](#)) noexcept  
*Resume from vCPU asynchronous IPC handler, commit.*
- [l4\\_msgtag\\_t vcpu\\_control](#) ([l4\\_addr\\_t vcpu\\_state](#), [l4\\_utcb\\_t \\*utcb=l4\\_utcb\(\)](#)) noexcept  
*Enable the vCPU feature for the thread.*
- [l4\\_msgtag\\_t vcpu\\_control\\_ext](#) ([l4\\_addr\\_t ext\\_vcpu\\_state](#), [l4\\_utcb\\_t \\*utcb=l4\\_utcb\(\)](#)) noexcept  
*Enable the extended vCPU feature for the thread.*
- [l4\\_msgtag\\_t register\\_del\\_irq](#) ([Cap< Irq > irq](#), [l4\\_utcb\\_t \\*u=l4\\_utcb\(\)](#)) noexcept  
*Register an IRQ that will trigger upon deletion events.*
- [l4\\_msgtag\\_t modify\\_senders](#) ([Modify\\_senders](#) const &todo) noexcept  
*Apply sender modification rules.*
- [l4\\_msgtag\\_t register\\_doorbell\\_irq](#) ([Cap< Irq > irq](#), [l4\\_utcb\\_t \\*u=l4\\_utcb\(\)](#)) noexcept  
*Register an IRQ that will trigger when a forwarded virtual interrupt is pending.*

## Additional Inherited Members

### Protected Types inherited from

[L4::Kobject\\_t< Thread, Snd\\_destination, L4\\_PROTO\\_THREAD, Type\\_info::Demand\\_t< 1 > >](#)

- typedef [Thread](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef [Typeid::Iface< PROTO, Thread >](#) **\_\_lfac**  
*The interface description for the derived class.*
- typedef [Typeid::Merge\\_list< Typeid::Iface\\_list< \\_\\_lfac >, typename Snd\\_destination::\\_\\_lfac\\_list >](#) **\_\_lfac\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Member Functions inherited from

[L4::Kobject\\_t< Thread, Snd\\_destination, L4\\_PROTO\\_THREAD, Type\\_info::Demand\\_t< 1 > >](#)

- [L4::Cap< Class > c](#) () const noexcept  
*Get the capability to ourselves.*

### Static Protected Member Functions inherited from

[L4::Kobject\\_t< Thread, Snd\\_destination, L4\\_PROTO\\_THREAD, Type\\_info::Demand\\_t< 1 > >](#)

- static void [\\_\\_check\\_protocols](#)\_\_ () noexcept  
*Helper to check for protocol conflicts.*

## 16.206.1 Detailed Description

C++ [L4](#) kernel thread interface, see [Thread](#) for the C interface.

The [Thread](#) class defines a thread of execution in the [L4](#) context. Usually user-level and kernel threads are mapped 1:1 to each other. [Thread](#) kernel objects are created using a factory, see the [L4::Factory](#) API ([L4::Factory::create\(\)](#)).

Amongst other things an [L4::Thread](#) encapsulates:

- CPU state
  - General-purpose registers
  - Program counter
  - Stack pointer
- FPU state
- Scheduling parameters, see the [L4::Scheduler](#) API
- Execution state
  - Blocked, Runnable, Running

[Thread](#) objects provide an API for

- [Thread](#) configuration and manipulation
- [Thread](#) switching.

On ARM newly created threads run in EL0 by default and the exception level can be changed there with [ex\\_regs\(\)](#).

### Include File

```
#include <l4/sys/thread>
```

For the C interface see the [Thread](#) API. For more elaborated documentation on the vCPU feature see [vCPU API](#).

Definition at line 52 of file [thread](#).

## 16.206.2 Member Function Documentation

### 16.206.2.1 control()

```
l4_msgtag_t L4::Thread::control (
    Attr const & attr) [inline], [noexcept]
```

Commit the given thread-attributes object.

#### Parameters

<i>attr</i>	the attribute object to commit to the thread.
-------------	-----------------------------------------------

#### Returns

Syscall return tag containing one of the following return codes.

#### Return values

<i>L4_EOK</i>	Operation successful.
<i>-L4_EPERM</i>	Insufficient permissions; see precondition.
<i>-L4_EINVAL</i>	Malformed thread-attributes.

#### Precondition

The invoked [Thread](#) capability must have the permission [L4\\_CAP\\_FPAGE\\_S](#). When using `#Attr::bind()`, also the respective [Task](#) capability must have the permission [L4\\_CAP\\_FPAGE\\_S](#).

Definition at line 243 of file [thread](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



#### 16.206.2.2 `ex_regs()` [1/2]

```

l4_msgtag_t L4::Thread::ex_regs (
    l4_addr_t * ip,
    l4_addr_t * sp,
    l4_umword_t * flags,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Exchange basic thread registers and return previous values.

#### Parameters

in, out	<i>ip</i>	New instruction pointer, use <code>~0UL</code> to leave the instruction pointer unchanged, return previous instruction pointer.
in, out	<i>sp</i>	New stack pointer, use <code>~0UL</code> to leave the stack pointer unchanged, returns previous stack pointer.
in, out	<i>flags</i>	Ex-regs flags, see <a href="#">L4_thread_ex_regs_flags</a> , return previous CPU flags of the thread.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .



## Returns

System call return tag. [out] parameters are only valid if the function returns successfully. Use [l4\\_error\(\)](#) to check.

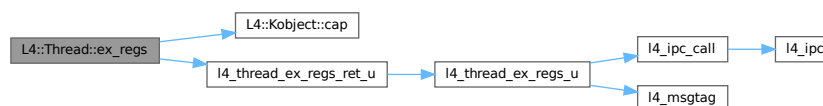
This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see `flags`). If the thread is in an IPC operation or if [L4\\_THREAD\\_EX\\_REGS\\_TRIGGER\\_EXCEPTION](#) forces an IPC then changes in IP and SP take effect directly after returning from this IPC. On ARM this method allows to change the exception level, see [L4\\_thread\\_ex\\_regs\\_flags\\_arm](#) and [L4\\_thread\\_ex\\_regs\\_flags\\_arm64](#).

The thread is started using [L4::Scheduler::run\\_thread\(\)](#). However, if at the time [L4::Scheduler::run\\_thread\(\)](#) is called, the instruction pointer of the thread is invalid, a later call to [ex\\_regs\(\)](#) with a valid instruction pointer might start the thread.

Definition at line 119 of file [thread](#).

References [L4::Kobject::cap\(\)](#), and [l4\\_thread\\_ex\\_regs\\_ret\\_u\(\)](#).

Here is the call graph for this function:



## 16.206.2.3 ex\_regs() [2/2]

```

l4_msgtag_t L4::Thread::ex_regs (
    l4_addr_t ip,
    l4_addr_t sp,
    l4_umword_t flags,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Exchange basic thread registers.

## Parameters

<i>ip</i>	New instruction pointer, use ~0UL to leave the instruction pointer unchanged.
<i>sp</i>	New stack pointer, use ~0UL to leave the stack pointer unchanged.
<i>flags</i>	Ex-regs flags, see <a href="#">L4_thread_ex_regs_flags</a> .
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

**Returns**

System call return tag.

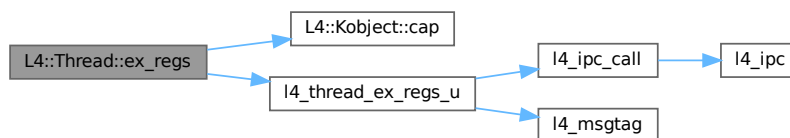
This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see `flags`). If the thread is in an IPC operation or if `L4_THREAD_EX_REGS_TRIGGER_EXCEPTION` forces an IPC then changes in IP and SP take effect directly after returning from this IPC. On ARM this method allows to change the exception level, see `L4_thread_ex_regs_flags_arm` and `L4_thread_ex_regs_flags_arm64`.

The thread is started using `L4::Scheduler::run_thread()`. However, if at the time `L4::Scheduler::run_thread()` is called, the instruction pointer of the thread is invalid, a later call to `ex_regs()` with a valid instruction pointer might start the thread.

Definition at line 84 of file `thread`.

References `L4::Kobject::cap()`, and `l4_thread_ex_regs_u()`.

Here is the call graph for this function:

**16.206.2.4 modify\_senders()**

```
l4_msgtag_t L4::Thread::modify_senders (
    Modify_senders const & todo) [inline], [noexcept]
```

Apply sender modification rules.

**Parameters**

<i>todo</i>	Prepared sender modification rules.
-------------	-------------------------------------

**Returns**

System call return tag.

The modification rules are applied to all IPCs to the thread (whether directly or by IPC gate) that are already in flight, that is that the sender is already blocking on.

See `Modify_senders` for a detailed description when applying sender modification rules is required.

**Note**

Modifying the senders of a thread running on a different CPU core is not supported.

To ensure that no in-flight senders are missed, either the thread itself must execute `modify_senders`, or the thread executing the `modify_senders` must synchronize with the target thread. This synchronization must ensure the following:

1. Before `modify_senders` is executed the target thread must execute at least shortly (so that pending DRQs are handled).
2. The target thread must pause its IPC dispatch, until `modify_senders` is completed. In other words, the target thread must not be receive ready, because otherwise an IPC message with an unmodified label can be transferred to its UTCB or vCPU state.

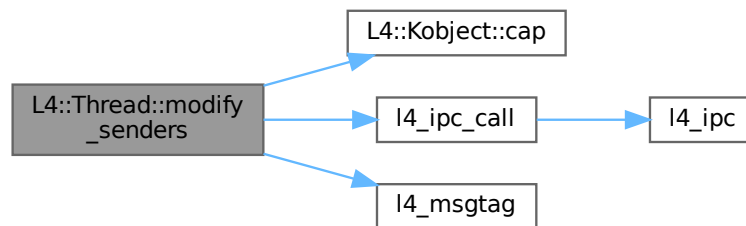
**See also**

[l4\\_thread\\_modify\\_sender\\_commit\(\)](#)

Definition at line 525 of file [thread](#).

References [L4::Kobject::cap\(\)](#), [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), and [L4\\_PROTO\\_THREAD](#).

Here is the call graph for this function:

**16.206.2.5 register\_del\_irq()**

```

l4_msgtag_t L4::Thread::register_del_irq (
    Cap< Irq > irq,
    l4_utcb_t * u = l4_utcb()) [inline], [noexcept]

```

Register an IRQ that will trigger upon deletion events.

**Parameters**

<i>irq</i>	Capability selector for the IRQ object to be triggered.
<i>u</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

**Returns**

System call return tag containing the return code.

**Return values**

-L4_BUSY	A deletion IRQ is already bound to this thread.
-L4_EPERM	Insufficient permissions; see precondition.

#### Precondition

The capability `irq` must have the permission [L4\\_CAP\\_FPAGE\\_W](#).

In case the `irq` is already bound to an interrupt source, it is unbound first. When `irq` is deleted, it will be deregistered first. A registered deletion [irq](#) can only be deregistered by deleting the [irq](#) or the thread.

List of deletion events:

- deletion of one or several IPC gates bound to this thread.

When the deletion event is delivered, there is no indication about which IPC gate was deleted.

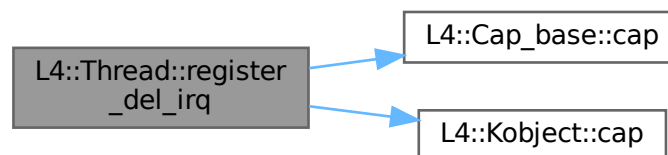
See also

[l4\\_thread\\_register\\_del\\_irq](#)

Definition at line [427](#) of file [thread](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



#### 16.206.2.6 register\_doorbell\_irq()

```

l4_msgtag_t L4::Thread::register_doorbell_irq (
    Cap< Irq > irq,
    l4_utcb_t * u = l4_utcb())  [inline], [noexcept]
  
```

Register an IRQ that will trigger when a forwarded virtual interrupt is pending.

#### Parameters

<i>irq</i>	Capability selector for the IRQ object to be triggered.
<i>u</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

**Returns**

System call return tag containing the return code.

**Return values**

-L4_BUSY	A doorbell IRQ is already bound to this thread.
-L4_EPERM	Insufficient permissions; see precondition.

**Precondition**

The capability `irq` must have the permission [L4\\_CAP\\_FPAGE\\_W](#).

See [Irq::bind\\_vcpu\(\)](#) for more details about how interrupts can be forwarded directly by the kernel to extended vCPU user mode.

In case the `irq` is already bound to an interrupt source, it is unbound first. When `irq` is deleted, it will be deregistered first. A registered doorbell [irq](#) can only be deregistered by deleting the [irq](#) or the thread.

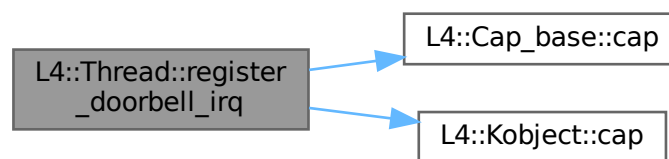
**See also**

[l4\\_thread\\_register\\_doorbell\\_irq](#)

Definition at line 553 of file [thread](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:

**16.206.2.7 stats\_time()**

```

l4_msgtag_t L4::Thread::stats_time (
    l4_kernel_clock_t * us,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Get consumed time of thread in us.

**Parameters**

out	us	Consumed time in $\mu$ s.
	utcb	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

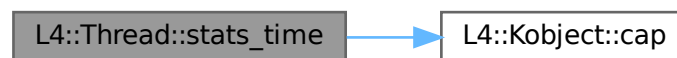
**Returns**

Syscall return tag.

Definition at line 264 of file [thread](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:

**16.206.2.8 switch\_to()**

```

l4_msgtag_t L4::Thread::switch_to (
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Switch execution to this thread.

**Parameters**

utcb	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .
------	--------------------------------------------------------------------------------------------------------------------

**Note**

The current time slice is inherited to this thread.

Definition at line 253 of file [thread](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 16.206.2.9 vcpu\_control()

```
l4_msgtag_t L4::Thread::vcpu_control (
    l4_addr_t vcpu_state,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Enable the vCPU feature for the thread.

#### Parameters

<i>vcpu_state</i>	A virtual address pointing to a <a href="#">l4_vcpu_state_t</a> . It must be a valid kernel-user-memory address (see <a href="#">L4::Task::add_ku_mem()</a> ).
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

#### Returns

Syscall return tag.

This function enables the vCPU feature of `this` thread

The kernel-user memory starting at `vcpu_state` must be at least 128-byte aligned and must cover the size of [l4\\_vcpu\\_state\\_t](#).

The asynchronous IPC handling is described at [vCPU API](#).

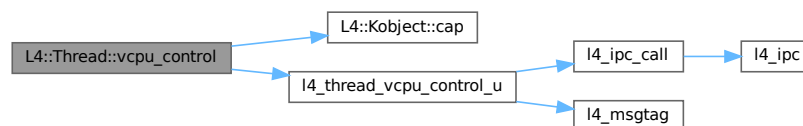
#### Note

Disabling of the vCPU feature is optional and currently not supported.

Definition at line 358 of file [thread](#).

References [L4::Kobject::cap\(\)](#), and [l4\\_thread\\_vcpu\\_control\\_u\(\)](#).

Here is the call graph for this function:



### 16.206.2.10 vcpu\_control\_ext()

```
l4_msgtag_t L4::Thread::vcpu_control_ext (
    l4_addr_t ext_vcpu_state,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Enable the extended vCPU feature for the thread.

#### Parameters

<i>ext_vcpu_state</i>	The virtual address where the kernel shall store the vCPU state in case of vCPU exits. The address must be a valid kernel-user-memory address (see <a href="#">L4::Task::add_ku_mem()</a> ).
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

### Returns

Syscall return tag.

The extended vCPU feature allows the use of hardware-virtualization features such as Intel's VT-x (VMX) or AMD's AMD-V (SVM).

This function enables the extended vCPU feature of `this` thread. Enabling the extended vCPU feature also enables the vCPU feature.

The kernel-user memory area starting at `ext_vcpu_state` must be at least 4 KiB aligned and must cover a size of `L4_PAGESIZE`. It includes the data of [l4\\_vcpu\\_state\\_t](#) at offset 0, the extended vCPU state at offset `L4_VCPU_OFFSET_EXT_STATE`, and, on some platforms, the extended vCPU information at offset `L4_VCPU_OFFSET_EXT_INFOS`.

On Intel's VT-x (VMX), the extended vCPU state is [l4\\_vm\\_vmx\\_vcpu\\_vmcs\\_t](#) and the extended vCPU information is [l4\\_vm\\_vmx\\_vcpu\\_infos\\_t](#). Furthermore, the extended vCPU state needs to be associated with a vCPU context (see [l4\\_vm\\_vmx\\_set\\_hw\\_vmcs\(\)](#)).

On AMD's AMD-V (SVM), the extended vCPU state is [l4\\_vm\\_svm\\_vmcb\\_t](#).

### Note

Enabling the extended vCPU feature for a thread running on a different CPU core is currently not supported.

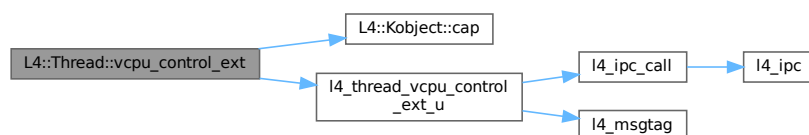
Disabling of the extended vCPU feature is currently not supported.

Upgrading from non-extended vCPU feature to extended vCPU feature is currently not supported.

Definition at line 398 of file [thread](#).

References [L4::Kobject::cap\(\)](#), and [l4\\_thread\\_vcpu\\_control\\_ext\\_u\(\)](#).

Here is the call graph for this function:



### 16.206.2.11 vcpu\_resume\_commit()

```

l4_msgtag_t l4::Thread::vcpu_resume_commit (
    l4_msgtag_t tag,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Resume from vCPU asynchronous IPC handler, commit.

### Parameters



<i>tag</i>	Tag to use, returned by <a href="#">l4_thread_vcpu_resume_start()</a> .
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

### Returns

Syscall return tag containing one of the following return codes.

### Return values

<i>0</i>	Indicates a VM exit, provided that <i>thread</i> is in extended vCPU mode with virtual interrupts cleared.
<i>1</i>	Indicates an incoming IPC message, provided that the <i>thread</i> is in extended vCPU mode with virtual interrupts cleared.
<i>-L4_EPERM</i>	The user task capability set in the vCPU state is missing the <a href="#">L4_CAP_FPAGE_S</a> right. On Intel's VT-x (VMX): The vCPU context capability set in the extended vCPU state is missing the <a href="#">L4_CAP_FPAGE_S</a> right.
<i>-L4_ENOENT</i>	The user task capability set in the vCPU state is invalid.
<i>-L4_EINVAL</i>	<i>thread</i> is not the current running thread, or does not have the vCPU feature enabled. On Intel's VT-x (VMX): No vCPU context associated with the extended vCPU state.
<i>-L4_EBUSY</i>	On Intel's VT-x (VMX): The vCPU context associated with the extended vCPU state is already active on a different CPU.
<i>-L4_ENODEV</i>	On Intel's VT-x (VMX): The vCPU context associated with the extended vCPU state cannot be initialized or activated.
<i>&lt;0</i>	A supplied mapping failed.

All flexpages in the UTCB (added with [l4\\_sndfpage\\_add\(\)](#) after [l4\\_thread\\_vcpu\\_resume\\_start\(\)](#)) are unconditionally mapped into the user task configured in the vCPU state.

To resume into another address space, the capability to the target [Task](#) (or [L4::Vm](#)) must be set in [l4\\_vcpu\\_state\\_t::user\\_task](#) together with [L4\\_VCPU\\_F\\_USER\\_MODE](#). The capability selector must have all lower bits clear (see [L4\\_CAP\\_MASK](#)). The kernel adds the [L4\\_SYSF\\_SEND](#) flag there to indicate that the capability has been referenced in the kernel. Consecutive resumes will not reference the task capability again until all lower bits are cleared again. To release a task use a different task capability or use an invalid capability with the [L4\\_SYSF\\_REPLY](#) flag set.

The asynchronous IPC handling is described at [vCPU API](#).

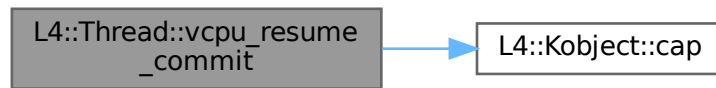
### See also

[l4\\_thread\\_vcpu\\_resume\\_commit](#)

Definition at line 334 of file [thread](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 16.206.2.12 vcpu\_resume\_start()

```
l4_msgtag_t L4::Thread::vcpu_resume_start (
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Resume from vCPU asynchronous IPC handler, start.

#### Parameters

<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .
-------------	--------------------------------------------------------------------------------------------------------------------

#### Returns

Message tag to be used for [l4\\_sndfpage\\_add\(\)](#) and [l4\\_thread\\_vcpu\\_resume\\_commit\(\)](#)

The vCPU resume functionality is split in multiple functions to allow the specification of additional send-flexpages using [l4\\_sndfpage\\_add\(\)](#).

The asynchronous IPC handling is described at [vCPU API](#).

#### See also

[l4\\_thread\\_vcpu\\_resume\\_start](#)

Definition at line 283 of file [thread](#).

The documentation for this class was generated from the following file:

- [l4/sys/thread](#)

## 16.207 L4::Thread::Attr Class Reference

[Thread](#) attributes used for [control\(\)](#).

```
#include <thread>
```

Collaboration diagram for L4::Thread::Attr:

L4::Thread::Attr
<ul style="list-style-type: none"> <li>+ Attr()</li> <li>+ pager()</li> <li>+ pager()</li> <li>+ exc_handler()</li> <li>+ exc_handler()</li> <li>+ bind()</li> <li>+ alien()</li> </ul>

### Public Member Functions

- [Attr](#) ([l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Create a thread-attribute object with the given UTCB.*
- void [pager](#) ([Cap](#)< void > const &pager) noexcept  
*Set the pager capability selector.*
- [Cap](#)< void > [pager](#) () noexcept  
*Get the capability selector used for page-fault messages.*
- void [exc\\_handler](#) ([Cap](#)< void > const &exc\_handler) noexcept  
*Set the exception-handler capability selector.*
- [Cap](#)< void > [exc\\_handler](#) () noexcept  
*Get the capability selector used for exception messages.*
- void [bind](#) ([l4\\_utcb\\_t](#) \*thread\_utcb, [Cap](#)< [Task](#) > const &task) noexcept  
*Bind the thread to a task.*
- void [alien](#) (int on) noexcept  
*Enable alien mode.*

### Friends

- class **L4::Thread**

### 16.207.1 Detailed Description

[Thread](#) attributes used for [control\(\)](#).

This class is responsible for initializing various attributes of a thread in a UTCB for the [control\(\)](#) method.

#### Note

Instantiation of this class starts the preparation of the UTCB. Do not invoke any non-Attr functions between the instantiation and the call to [L4::Thread::control\(\)](#).

#### See also

[Thread control](#) for some more details.

Definition at line 137 of file [thread](#).

### 16.207.2 Constructor & Destructor Documentation

#### 16.207.2.1 Attr()

```
L4::Thread::Attr::Attr (
    l4_utcb_t * utcb = l4_utcb()) [inline], [explicit], [noexcept]
```

Create a thread-attribute object with the given UTCB.

#### Parameters

<i>utcb</i>	The UTCB to use for the later <a href="#">L4::Thread::control()</a> function. Usually this is the UTCB of the calling thread. See <a href="#">l4_utcb()</a> .
-------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------

Definition at line 151 of file [thread](#).

### 16.207.3 Member Function Documentation

#### 16.207.3.1 alien()

```
void L4::Thread::Attr::alien (
    int on) [inline], [noexcept]
```

Enable alien mode.

#### Parameters

<i>on</i>	Boolean value defining the state of the feature.
-----------	--------------------------------------------------

For a thread in alien mode the kernel produces just an exception IPC for each IPC and exception caused by the alien thread instead of handling these events regularly. (Page faults of alien threads and interrupts occurring while the alien thread is running are always handled regularly.) While the alien thread is blocking, the exception handler can inspect and modify the state of the alien thread and potentially also the system call arguments. If the exception handler replies with [L4\\_PROTO\\_ALLOW\\_SYSCALL](#) as message tag, the kernel handles the next IPC or exception of the alien thread in a regular way. If the exception handler leaves certain thread state unchanged (in particular the instruction pointer), this will be the IPC or exception that caused the call of the exception handler. For a regularly processed IPC or exception of the alien thread the kernel also performs an exception IPC on kernel exit.

This feature can be used to attach a debugger to a thread and trace all object invocations and their results. It could also be used to handle other systems that use the same syscall instruction as [L4Re](#).

Definition at line 224 of file [thread](#).

### 16.207.3.2 bind()

```
void L4::Thread::Attr::bind (
    l4_utcb_t * thread_utcb,
    Cap< Task > const & task) [inline], [noexcept]
```

Bind the thread to a task.

#### Parameters

<i>thread_utcb</i>	The thread's UTCB address within the task it shall be bound to. The address must be aligned (architecture dependent; at least word aligned) and it must point to at least L4_UTCB_OFFSET bytes of kernel-user memory.
<i>task</i>	The task the thread shall be bound to.

#### Precondition

The thread must not be bound to a task yet.

The capability `task` must have the permission [L4\\_CAP\\_FPAGE\\_S](#), otherwise the later call to [L4::Thread::control\(\)](#) with this [Attr](#) object will fail with [L4\\_EPERM](#).

A thread may execute code in the context of a task if and only if the thread is bound to the task. To actually start execution, [L4::Thread::ex\\_regs\(\)](#) needs to be used. Execution in the context of the task means that the code has access to all the task's resources (and only those). The executed code itself must be one of those resources. A thread can be bound at most once to a task.

#### Note

The UTCBs of different threads in the same task should not overlap in order to prevent data corruption.

Definition at line [218](#) of file [thread](#).

### 16.207.3.3 exc\_handler() [1/2]

```
Cap< void > L4::Thread::Attr::exc_handler () [inline], [noexcept]
```

Get the capability selector used for exception messages.

#### Returns

The capability selector used to send exception messages. The selector is valid in the task the thread is bound to.

Definition at line [189](#) of file [thread](#).

### 16.207.3.4 exc\_handler() [2/2]

```
void L4::Thread::Attr::exc_handler (
    Cap< void > const & exc_handler) [inline], [noexcept]
```

Set the exception-handler capability selector.

#### Parameters

<i>exc_handler</i>	The capability selector that shall be used for exception messages. This capability selector must be valid within the task the thread is bound to.
--------------------	---------------------------------------------------------------------------------------------------------------------------------------------------

Definition at line 180 of file [thread](#).

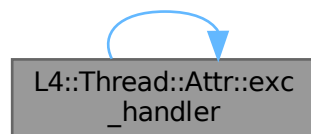
References [exc\\_handler\(\)](#).

Referenced by [exc\\_handler\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.207.3.5 pager() [1/2]

```
Cap< void > L4::Thread::Attr::pager () [inline], [noexcept]
```

Get the capability selector used for page-fault messages.

#### Returns

The capability selector used to send page-fault messages. The selector is valid in the task the thread is bound to.

Definition at line 170 of file [thread](#).

### 16.207.3.6 pager() [2/2]

```
void L4::Thread::Attr::pager (
    Cap< void > const & pager) [inline], [noexcept]
```

Set the pager capability selector.

#### Parameters

<i>pager</i>	The capability selector that shall be used for page-fault messages. This capability selector must be valid within the task the thread is bound to.
--------------	----------------------------------------------------------------------------------------------------------------------------------------------------

Definition at line 161 of file [thread](#).

References [pager\(\)](#).

Referenced by [pager\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

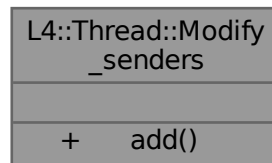
- [l4/sys/thread](#)

## 16.208 L4::Thread::Modify\_senders Class Reference

[Class](#) wrapping a list of rules which modify the sender label of IPC messages inbound to this thread.

```
#include <thread>
```

Collaboration diagram for L4::Thread::Modify\_senders:



### Public Member Functions

- `int add (l4_umword_t match_mask, l4_umword_t match, l4_umword_t del_bits, l4_umword_t add_bits) noexcept`

*Add a rule.*

## 16.208.1 Detailed Description

[Class](#) wrapping a list of rules which modify the sender label of IPC messages inbound to this thread.

Use the [add\(\)](#) function to add modification rules, and use [modify\\_senders\(\)](#) to commit. Do not use the UTCB in between as it is used by [add\(\)](#) and [modify\\_senders\(\)](#).

This mechanism shall be used to change the source object labels of every pending IPC of an IPC gate or an IRQ if the labels in such pending IPC become invalid for the receiving thread, potentially because:

- an IPC gate / IRQ was unbound from a thread, or
- an IPC gate / IRQ was removed, or
- the label of an IPC gate / IRQ bound to a thread was changed.

It is not required to perform the modify\_sender mechanism after an IPC gate or an IRQ was bound to a thread for the first time.

Definition at line [448](#) of file [thread](#).

## 16.208.2 Member Function Documentation

### 16.208.2.1 add()

```

int L4::Thread::Modify_senders::add (
    l4_umword_t match_mask,
    l4_umword_t match,
    l4_umword_t del_bits,
    l4_umword_t add_bits) [inline], [noexcept]
  
```

Add a rule.

#### Parameters

---



<i>match_mask</i>	Bitmask of bits to match the label.
<i>match</i>	Bitmask that must be equal to the label after applying <i>match_mask</i> .
<i>del_bits</i>	Bits to be deleted from the label.
<i>add_bits</i>	Bits to be added to the label.

#### Returns

0 on success, <0 on error

In pseudo code: if ((sender\_label & match\_mask) == match) { sender\_label = (sender\_label & ~del\_bits) | add\_bits; }

Only the first match is applied.

#### See also

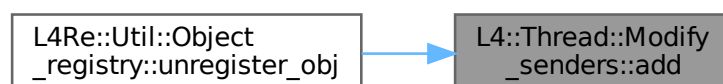
[l4\\_thread\\_modify\\_sender\\_add\(\)](#)

Definition at line [481](#) of file [thread](#).

References [L4\\_ENOMEM](#), and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [L4Re::Util::Object\\_registry::unregister\\_obj\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

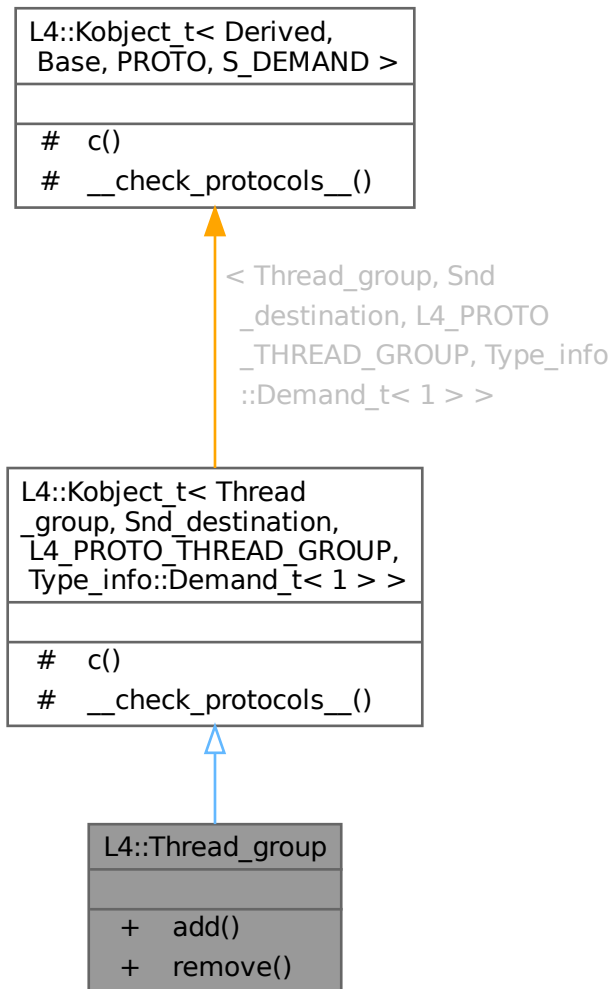
- [l4/sys/thread](#)

## 16.209 L4::Thread\_group Class Reference

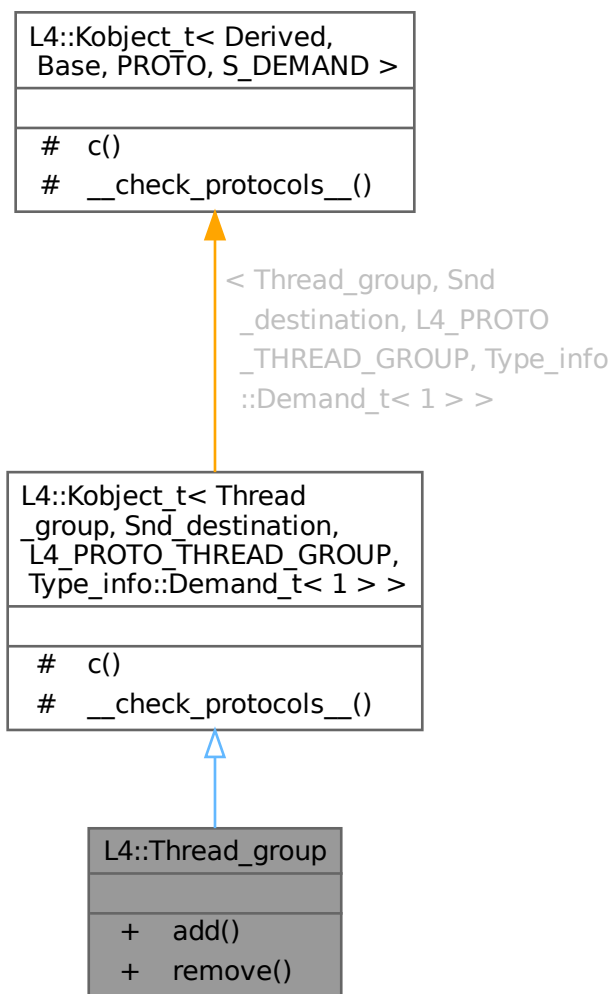
C++ [L4](#) kernel thread group interface, see [Thread groups](#) for the C interface.

```
#include <thread_group>
```

Inheritance diagram for L4::Thread\_group:



Collaboration diagram for L4::Thread\_group:



## Public Member Functions

- `l4_msgtag_t add (Cap< Thread > thread, l4_utcb_t *utcb=l4_utcb()) noexcept`  
Add thread to a thread group.
- `l4_msgtag_t remove (Cap< Thread > thread, l4_utcb_t *utcb=l4_utcb()) noexcept`  
Remove thread from a thread group.

## Additional Inherited Members

## Protected Types inherited from

**L4::Kobject\_t< Thread\_group, Snd\_destination, L4\_PROTO\_THREAD\_GROUP, Type\_info::Demand\_t<**

- typedef `Thread_group` **Class**

The target interface type (inheriting from [Kobject\\_t](#)).

- typedef Typeid::Iface< PROTO, [Thread\\_group](#) > \_\_Iface

The interface description for the derived class.

- typedef Typeid::Merge\_list< Typeid::Iface\_list< \_\_Iface >, typename Snd\_destination::Iface\_list > \_\_Iface\_list

The list of all RPC interfaces provided directly or through inheritance.

### Protected Member Functions inherited from

[L4::Kobject\\_t< Thread\\_group, Snd\\_destination, L4\\_PROTO\\_THREAD\\_GROUP, Type\\_info::Demand\\_t<](#)

- [L4::Cap< Class > c \(\)](#) const noexcept

Get the capability to ourselves.

### Static Protected Member Functions inherited from

[L4::Kobject\\_t< Thread\\_group, Snd\\_destination, L4\\_PROTO\\_THREAD\\_GROUP, Type\\_info::Demand\\_t<](#)

- static void [\\_\\_check\\_protocols\\_\\_ \(\)](#) noexcept

Helper to check for protocol conflicts.

## 16.209.1 Detailed Description

C++ [L4](#) kernel thread group interface, see [Thread groups](#) for the C interface.

An [L4](#) thread group is a collection of threads used as indirection for IPC gate and IRQ objects such that these objects can have multiple receivers, from which the kernel selects one according to a policy.

The primary use case for thread groups are multi-threaded servers and CPU core local IRQ / IPC delivery.

A thread can be bound to at most one thread group. Before binding a thread to a thread group, the thread must be bound to a task. All threads bound to the same thread group must belong to the same task.

Definition at line [34](#) of file [thread\\_group](#).

## 16.209.2 Member Function Documentation

### 16.209.2.1 add()

```
l4\_msgtag\_t L4::Thread_group::add (
    Cap< Thread > thread,
    l4\_utcb\_t * utcb = l4\_utcb\(\)) [inline], [noexcept]
```

Add thread to a thread group.

#### Parameters

<i>thread</i>	<a href="#">Thread</a> to add to the thread group.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

#### Returns

Syscall return tag containing one of the following return codes.

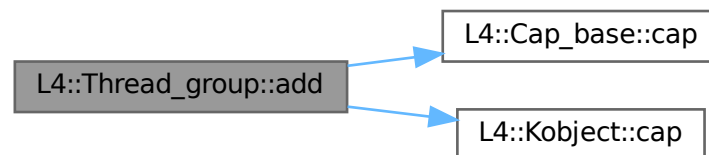
#### Return values

<i>L4_EOK</i>	No error occurred.
<i>-L4_EINVAL</i>	<code>thread</code> is not a thread object.
<i>-L4_EEXIST</i>	<code>thread</code> already bound to this thread group.
<i>-L4_EBUSY</i>	<code>thread</code> already bound to a different thread group.
<i>-L4_ENOENT</i>	<a href="#">Thread</a> group doesn't exist.

Definition at line 53 of file [thread\\_group](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 16.209.2.2 remove()

```

l4_msgtag_t L4::Thread_group::remove (
    Cap< Thread > thread,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Remove thread from a thread group.

#### Parameters

<i>thread</i>	<a href="#">Thread</a> to remove from the thread group.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

#### Returns

Syscall return tag containing one of the following return codes.

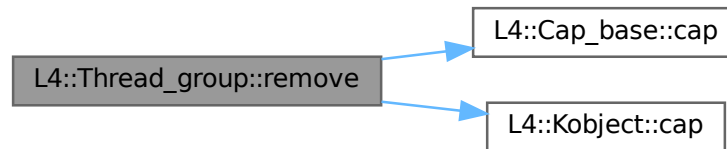
#### Return values

<i>L4_EOK</i>	No error occurred.
<i>-L4_EINVAL</i>	<code>thread</code> is not a thread object.

Definition at line 67 of file [thread\\_group](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

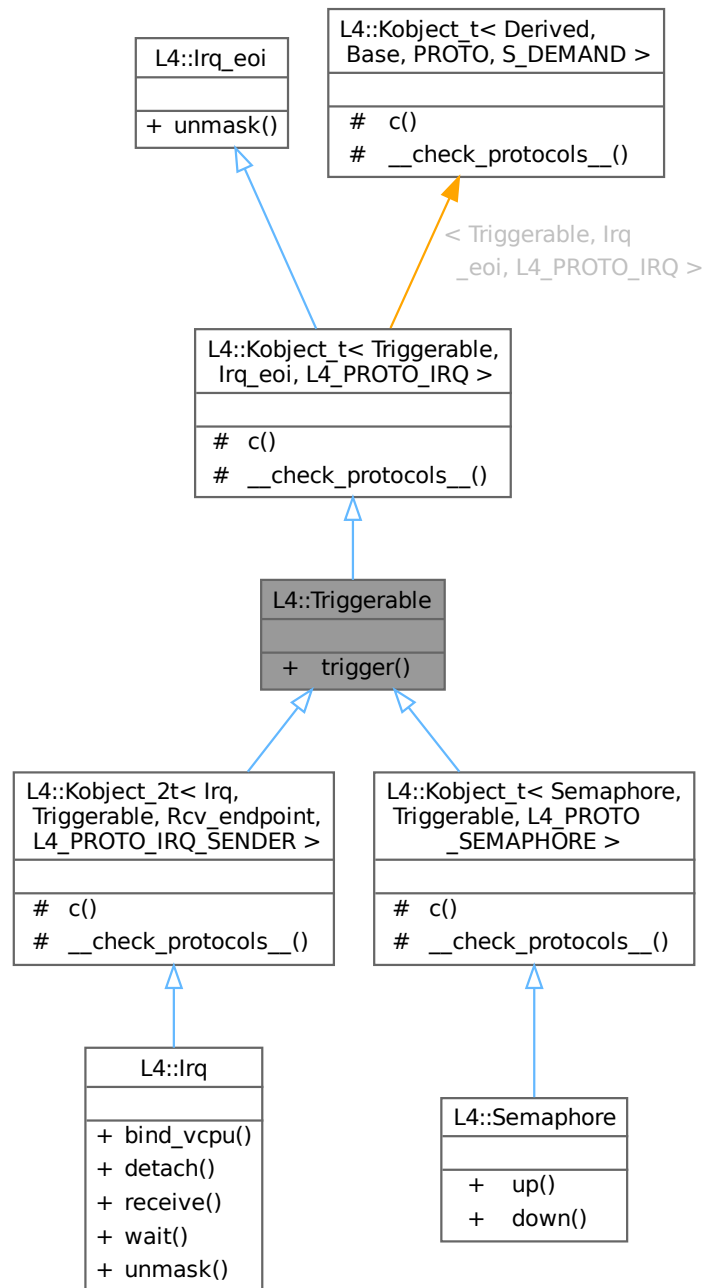
- [l4/sys/thread\\_group](#)

## 16.210 L4::Triggerable Struct Reference

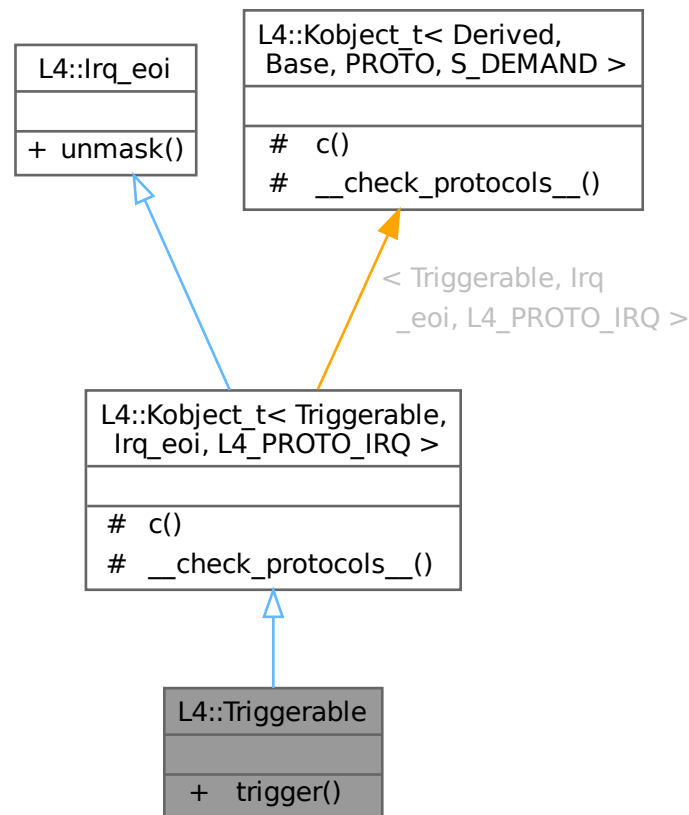
Interface that allows an object to be triggered by some source.

```
#include <irq>
```

Inheritance diagram for L4::Triggerable:



Collaboration diagram for L4::Triggerable:



### Public Member Functions

- [l4\\_msgtag\\_t trigger](#) ([l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Trigger the object.*

### Public Member Functions inherited from [L4::Irq\\_eoi](#)

- [l4\\_msgtag\\_t unmask](#) (unsigned irqnum, [l4\\_umword\\_t](#) \*label=0, [l4\\_timeout\\_t](#) to=[L4\\_IPC\\_NEVER](#), [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Unmask the given interrupt line.*

### Additional Inherited Members

### Protected Types inherited from [L4::Kobject\\_t< Triggerable, Irq\\_eoi, L4\\_PROTO\\_IRQ >](#)

- typedef [Triggerable](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< PROTO, [Triggerable](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< [\\_\\_Iface](#) >, typename Irq\_eoi::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*



## Protected Member Functions inherited from [L4::Kobject\\_t< Triggerable, Irq\\_eoi, L4\\_PROTO\\_IRQ >](#)

- [L4::Cap< Class > c \(\)](#) const noexcept  
*Get the capability to ourselves.*

## Static Protected Member Functions inherited from [L4::Kobject\\_t< Triggerable, Irq\\_eoi, L4\\_PROTO\\_IRQ >](#)

- static void [\\_\\_check\\_protocols\\_\\_ \(\)](#) noexcept  
*Helper to check for protocol conflicts.*

### 16.210.1 Detailed Description

Interface that allows an object to be triggered by some source.

The interface specifies no semantics for the trigger operation, this is defined by derived objects.

This interface is usually used in conjunction with [L4::lcu](#).

Definition at line 79 of file [irq](#).

### 16.210.2 Member Function Documentation

#### 16.210.2.1 trigger()

```
l4_msgtag_t L4::Triggerable::trigger (
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Trigger the object.

#### Parameters

<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .
-------------	------------------------------------------------------------------------------------------------------------

#### Returns

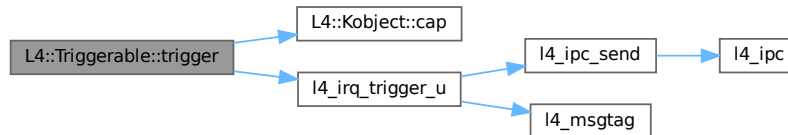
Syscall return tag for a send-only operation, this means there is no return value except [L4\\_MSGTAG\\_ERROR](#) indicating success or failure of the send operation. Use [l4\\_ipc\\_error\(\)](#) to check for errors and **do not** use [l4\\_error\(\)](#).

Definition at line 91 of file [irq](#).

References [L4::Kobject::cap\(\)](#), and [l4\\_irq\\_trigger\\_u\(\)](#).

Referenced by [L4::Semaphore::up\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this struct was generated from the following file:

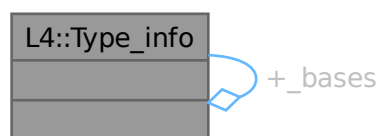
- [l4/sys/irq](#)

## 16.211 L4::Type\_info Struct Reference

Dynamic Type Information for [L4Re](#) Interfaces.

```
#include <l4/sys/capability>
```

Collaboration diagram for `L4::Type_info`:



## Data Structures

- class [Demand](#)  
*Data type for expressing the needed receive buffers at the server-side of an interface.*
- struct [Demand\\_t](#)  
*Template type statically describing demand of receive buffers.*
- struct [Demand\\_union\\_t](#)  
*Template type statically describing the combination of two [Demand](#) object.*

### 16.211.1 Detailed Description

Dynamic Type Information for [L4Re](#) Interfaces.

This class represents the runtime-dynamic type information for [L4Re](#) interfaces, and is not intended to be used directly by applications.

#### Note

The interface of is subject to changes.

The main use for this info is to be used by the implementation of the [L4::cap\\_dynamic\\_cast\(\)](#) function.

Definition at line 499 of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

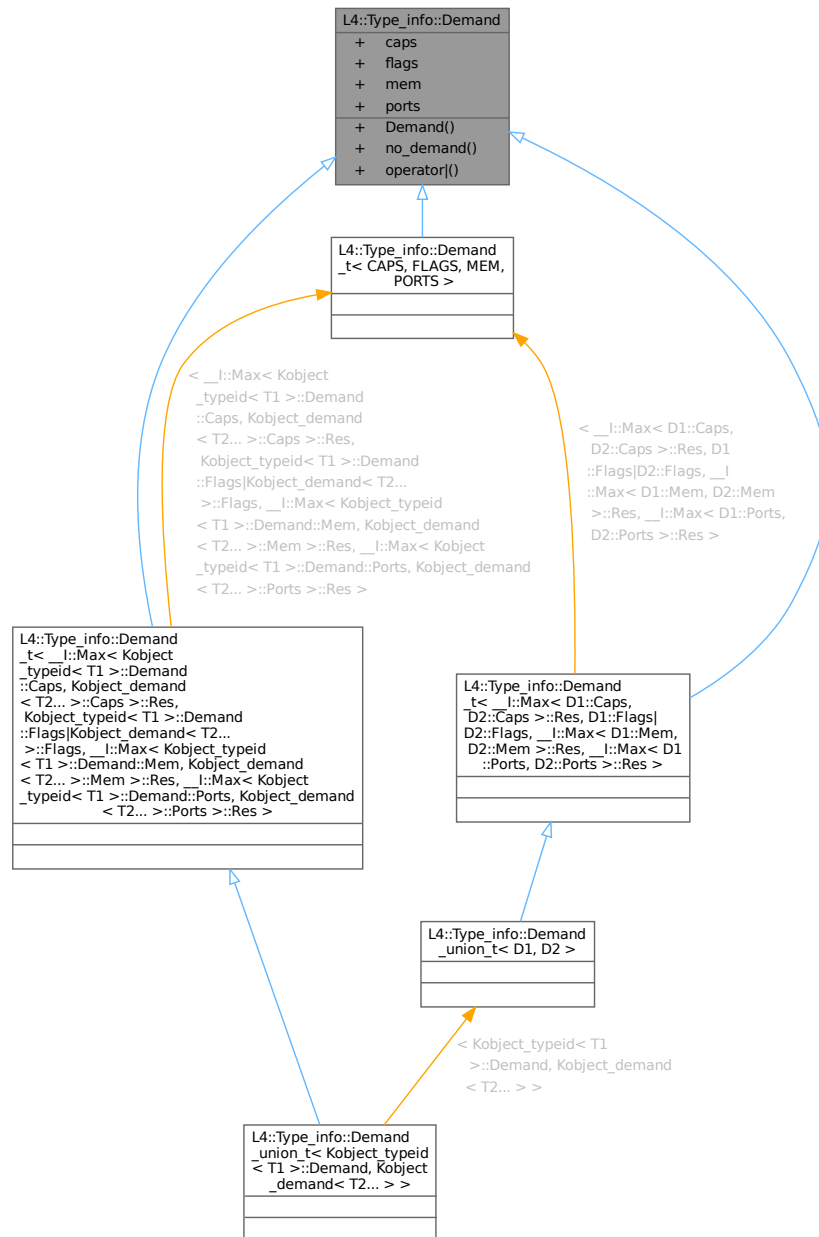
- [l4/sys/\\_\\_typeinfo.h](#)

## 16.212 L4::Type\_info::Demand Class Reference

Data type for expressing the needed receive buffers at the server-side of an interface.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Type\_info::Demand:



Collaboration diagram for L4::Type\_info::Demand:

L4::Type_info::Demand	
+	caps
+	flags
+	mem
+	ports
+	Demand()
+	no_demand()
+	operator ()

### Public Member Functions

- [Demand](#) (unsigned char [caps](#)=0, unsigned char [flags](#)=0, unsigned char [mem](#)=0, unsigned char [ports](#)=0) noexcept  
*Make [Demand](#) object.*
- bool [no\\_demand](#) () const noexcept
- [Demand operator|](#) ([Demand](#) const &rhs) const noexcept  
*get the combined demand of this and rhs*

### Data Fields

- unsigned char **caps**  
*number of capability receive buffers.*
- unsigned char **flags**  
*flags, such as the need for timeouts (TBD).*
- unsigned char **mem**  
*number of memory receive buffers.*
- unsigned char **ports**  
*number of IO-port receive buffers.*

## 16.212.1 Detailed Description

Data type for expressing the needed receive buffers at the server-side of an interface.

Definition at line 506 of file [\\_\\_typeinfo.h](#).

## 16.212.2 Constructor & Destructor Documentation

### 16.212.2.1 Demand()

```
L4::Type_info::Demand::Demand (  
    unsigned char caps = 0,  
    unsigned char flags = 0,  
    unsigned char mem = 0,  
    unsigned char ports = 0) [inline], [explicit], [noexcept]
```

Make [Demand](#) object.

#### Parameters

<i>caps</i>	number of capability receive buffers
<i>flags</i>	flags, such as the need for timeouts (TBD).
<i>mem</i>	number of memory receive windows.
<i>ports</i>	number of IO-port receive windows.

Definition at line [527](#) of file [\\_\\_typeinfo.h](#).

References [caps](#), [flags](#), [mem](#), and [ports](#).

Referenced by [operator|\(\)](#).

Here is the caller graph for this function:



## 16.212.3 Member Function Documentation

### 16.212.3.1 no\_demand()

```
bool L4::Type_info::Demand::no_demand () const [inline], [noexcept]
```

#### Returns

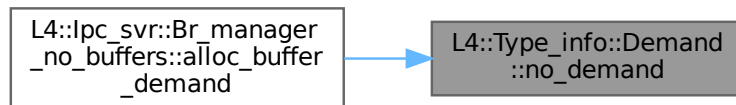
true if there is no demand at all

Definition at line 532 of file [\\_\\_typeinfo.h](#).

References [caps](#), [flags](#), [mem](#), and [ports](#).

Referenced by [L4::lpc\\_svr::Br\\_manager\\_no\\_buffers::alloc\\_buffer\\_demand\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- [l4/sys/\\_\\_typeinfo.h](#)

## 16.213 L4::Type\_info::Demand\_t< CAPS, FLAGS, MEM, PORTS > Struct Template Reference

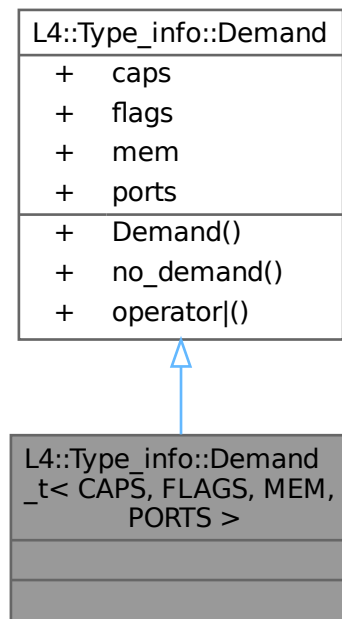
Template type statically describing demand of receive buffers.

```
#include <l4/sys/capability>
```





Collaboration diagram for L4::Type\_info::Demand\_t< CAPS, FLAGS, MEM, PORTS >:



## Public Types

- enum { `Caps` = CAPS , `Flags` = FLAGS , `Mem` = MEM , `Ports` = PORTS }

## Additional Inherited Members

## Public Member Functions inherited from L4::Type\_info::Demand

- `Demand` (unsigned char `caps`=0, unsigned char `flags`=0, unsigned char `mem`=0, unsigned char `ports`=0) noexcept  
*Make `Demand` object.*
- bool `no_demand` () const noexcept
- `Demand operator|` (`Demand` const &rhs) const noexcept  
*get the combined demand of this and rhs*

## Data Fields inherited from L4::Type\_info::Demand

- unsigned char **caps**  
*number of capability receive buffers.*
- unsigned char **flags**  
*flags, such as the need for timeouts (TBD).*
- unsigned char **mem**  
*number of memory receive buffers.*
- unsigned char **ports**  
*number of IO-port receive buffers.*

### 16.213.1 Detailed Description

```
template<unsigned char CAPS = 0, unsigned char FLAGS = 0, unsigned char MEM = 0, unsigned char
PORTS = 0>
struct L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >
```

Template type statically describing demand of receive buffers.

#### Template Parameters

<i>CAPS</i>	number of capability receive buffers needed.
<i>FLAGS</i>	flags, such as the need for timeouts (TBD).
<i>MEM</i>	number of memory receive windows needed.
<i>PORTS</i>	number of IO-port receive windwows needed.

Definition at line 553 of file [\\_\\_typeinfo.h](#).

### 16.213.2 Member Enumeration Documentation

#### 16.213.2.1 anonymous enum

```
template<unsigned char CAPS = 0, unsigned char FLAGS = 0, unsigned char MEM = 0, unsigned char
PORTS = 0>
anonymous enum
```

#### Enumerator

Caps	number of capability receive buffers.
Flags	flags, such as the need for timeouts.
Mem	number of memory receive windows.
Ports	number of IO-port receive windows.

Definition at line 555 of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

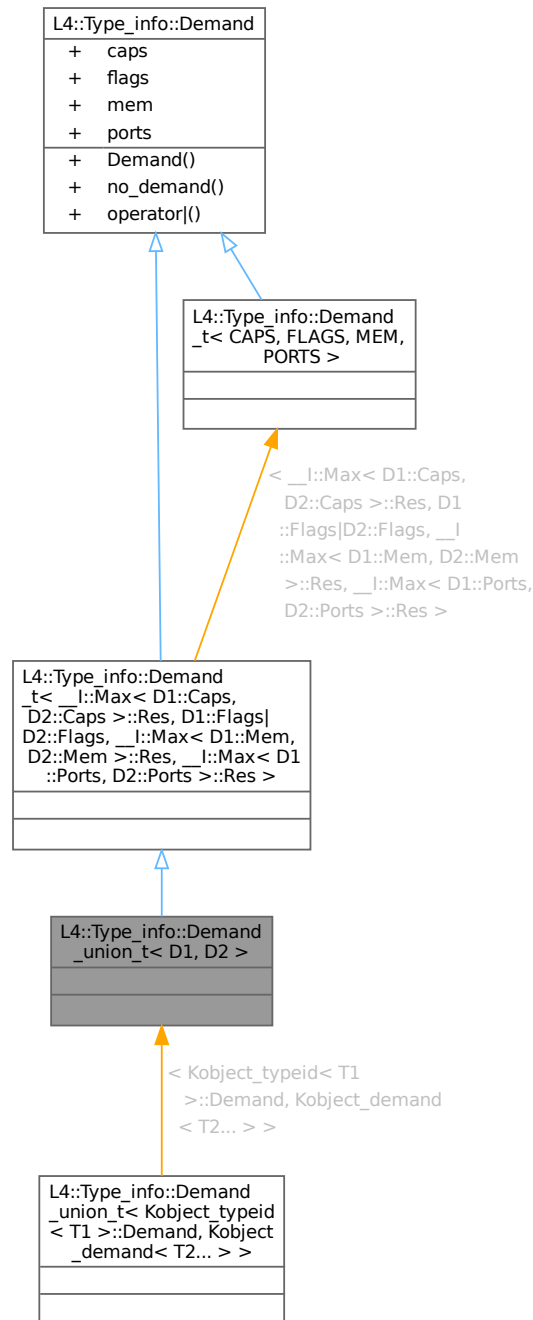
- [l4/sys/\\_\\_typeinfo.h](#)

### 16.214 L4::Type\_info::Demand\_union\_t< D1, D2 > Struct Template Reference

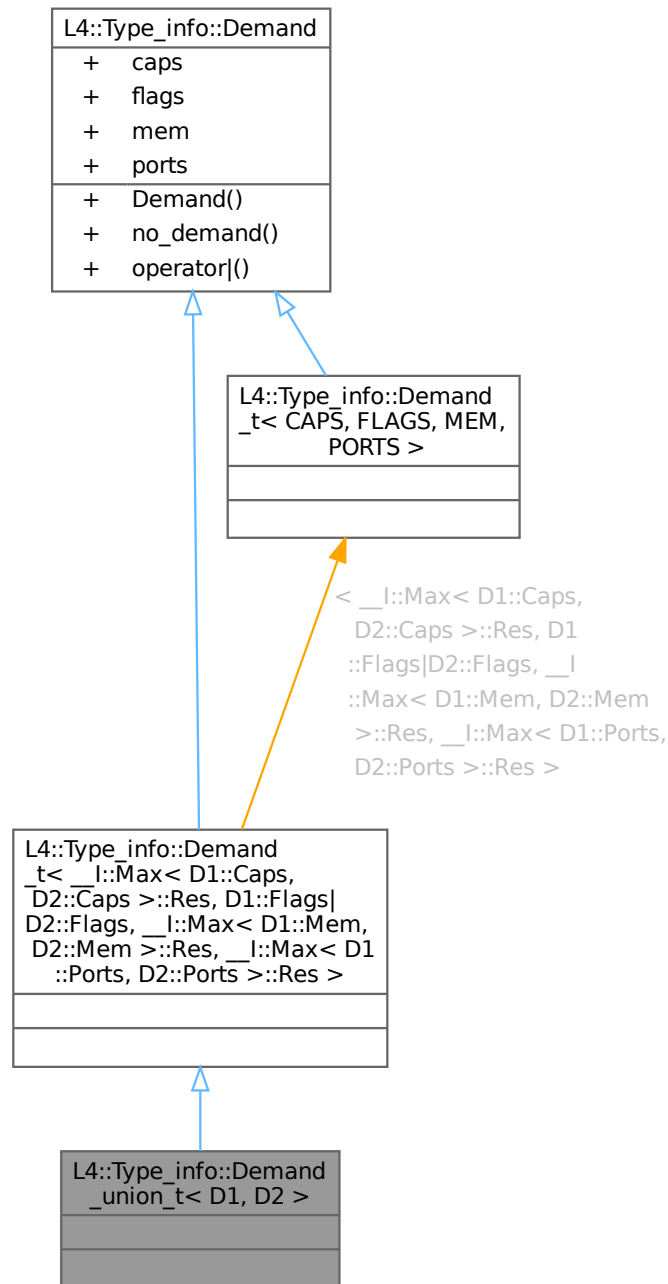
Template type statically describing the combination of two [Demand](#) object.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Type\_info::Demand\_union\_t< D1, D2 >:



Collaboration diagram for L4::Type\_info::Demand\_union\_t< D1, D2 >:



## Additional Inherited Members

### Public Types inherited from

[L4::Type\\_info::Demand\\_t< \\_\\_l::Max< D1::Caps, D2::Caps >::Res, D1::Flags|D2::Flags, \\_\\_l::Max< D1::](#)

### Public Member Functions inherited from [L4::Type\\_info::Demand](#)

- [Demand](#) (unsigned char [caps](#)=0, unsigned char [flags](#)=0, unsigned char [mem](#)=0, unsigned char [ports](#)=0) noexcept  
*Make [Demand](#) object.*
- bool [no\\_demand](#) () const noexcept
- [Demand](#) operator| ([Demand](#) const &rhs) const noexcept  
*get the combined demand of this and rhs*

### Data Fields inherited from [L4::Type\\_info::Demand](#)

- unsigned char **caps**  
*number of capability receive buffers.*
- unsigned char **flags**  
*flags, such as the need for timeouts (TBD).*
- unsigned char **mem**  
*number of memory receive buffers.*
- unsigned char **ports**  
*number of IO-port receive buffers.*

## 16.214.1 Detailed Description

```
template<typename D1, typename D2>
struct L4::Type_info::Demand_union_t< D1, D2 >
```

Template type statically describing the combination of two [Demand](#) object.

### Template Parameters

<i>D1</i>	first demand object.
<i>D2</i>	second demand object.

Definition at line [573](#) of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

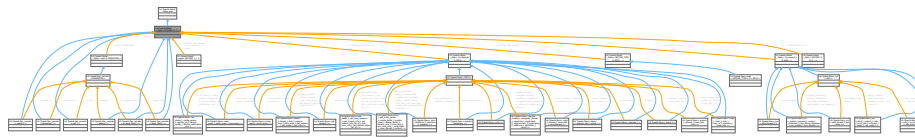
- [l4/sys/\\_\\_typeinfo.h](#)

## 16.215 L4::Typeid::Detail::\_Rpc< OPCODE, O, X > Struct Template Reference

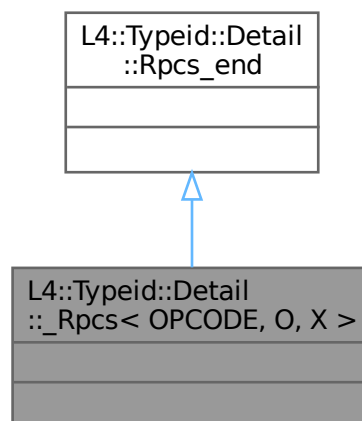
Empty list of RPCs.

```
#include <__typeinfo.h>
```

Inheritance diagram for L4::Typeid::Detail::\_Rpc< OPCODE, O, X >:



Collaboration diagram for L4::Typeid::Detail::\_Rpc< OPCODE, O, X >:



### 16.215.1 Detailed Description

```
template<typename OPCODE, unsigned O, typename ... X>
struct L4::Typeid::Detail::_Rpc< OPCODE, O, X >
```

Empty list of RPCs.

Definition at line 365 of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

- [l4/sys/\\_\\_typeinfo.h](#)

## 16.216 L4::Typeid::Detail::\_Rpc< OPCODE, O, Default\_op< R > >::Rpc< Y > Struct Template Reference

Find the given RPC in the list.

```
#include <__typeinfo.h>
```

Collaboration diagram for L4::Typeid::Detail::\_Rpc< OPCODE, O, Default\_op< R > >::Rpc< Y >:

L4::Typeid::Detail :: _Rpc< OPCODE, O, Default_op< R > >::Rpc< Y >

### 16.216.1 Detailed Description

```
template<typename OPCODE, unsigned O, typename R>
template<typename Y>
struct L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::Rpc< Y >
```

Find the given RPC in the list.

Definition at line 399 of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

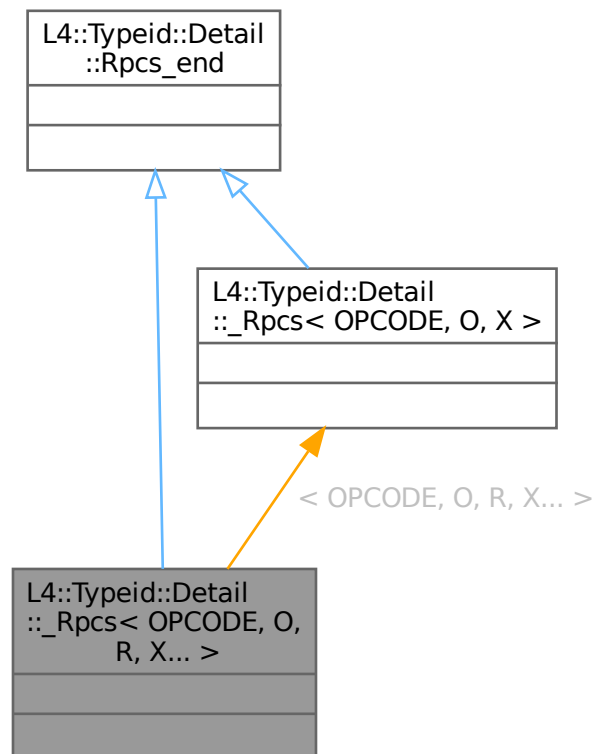
- l4/sys/[\\_\\_typeinfo.h](#)

## 16.217 L4::Typeid::Detail::\_Rpc< OPCODE, O, R, X... > Struct Template Reference

Non-empty list of RPCs.

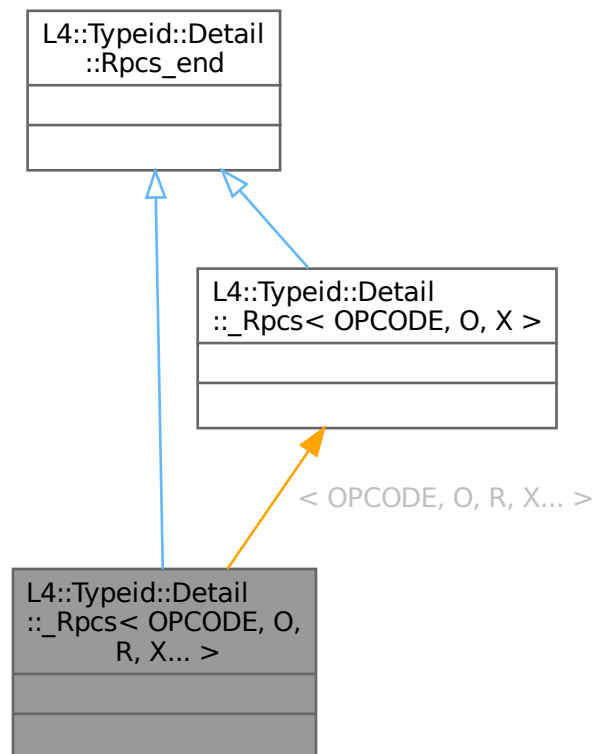
```
#include <__typeinfo.h>
```

Inheritance diagram for L4::Typeid::Detail::\_Rpccs< OPCODE, O, R, X... >:





Collaboration diagram for L4::Typeid::Detail::\_Rpccs< OPCODE, O, R, X... >:



## Data Structures

- struct [Rpc](#)

*Find the given RPC in the list.*

## Public Types

- enum

*The opcode value to use for this RPC, may be bogus if the [opcode\\_type](#) is void.*

- typedef [\\_Rpccs](#) **type**

*The list element itself.*

- typedef OPCODE **opcode\_type**

*The data type for the opcode.*

- typedef R **rpc**

*The RPC type `L4::lpc::Msg::Rpc_call` or `L4::lpc::Msg::Rpc_inline_call`.*

- typedef [\\_Rpccs](#)< OPCODE, `_Get_opcode`< R, O >::value+1, X... >::type **next**

*The next RPC in the list or [Rpccs\\_end](#) if this is the last.*

### 16.217.1 Detailed Description

```
template<typename OPCODE, unsigned O, typename R, typename ... X>
struct L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >
```

Non-empty list of RPCs.

Definition at line 369 of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

- [l4/sys/\\_\\_typeinfo.h](#)

## 16.218 L4::Typeid::Detail::\_Rpc< OPCODE, O, R, X... >::Rpc< Y > Struct Template Reference

Find the given RPC in the list.

```
#include <__typeinfo.h>
```

Collaboration diagram for L4::Typeid::Detail::\_Rpc< OPCODE, O, R, X... >::Rpc< Y >:

L4::Typeid::Detail :: _Rpc< OPCODE, O, R, X... >::Rpc< Y >

### 16.218.1 Detailed Description

```
template<typename OPCODE, unsigned O, typename R, typename ... X>
template<typename Y>
struct L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >::Rpc< Y >
```

Find the given RPC in the list.

Definition at line 382 of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

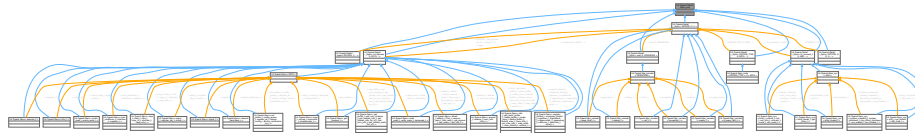
- [l4/sys/\\_\\_typeinfo.h](#)

## 16.219 L4::Typeid::Detail::Rpc\_end Struct Reference

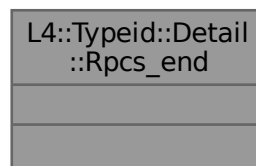
Internal end-of-list marker.

```
#include <__typeinfo.h>
```

Inheritance diagram for L4::Typeid::Detail::Rpc\_end:



Collaboration diagram for L4::Typeid::Detail::Rpc\_end:



### 16.219.1 Detailed Description

Internal end-of-list marker.

Definition at line 317 of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

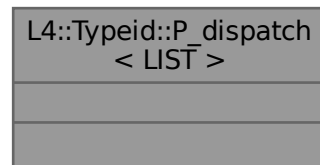
- [l4/sys/\\_\\_typeinfo.h](#)

## 16.220 L4::Typeid::P\_dispatch< LIST > Struct Template Reference

Use for protocol based dispatch stage.

```
#include <__typeinfo.h>
```

Collaboration diagram for L4::Typeid::P\_dispatch< LIST >:



### 16.220.1 Detailed Description

```
template<typename LIST>
struct L4::Typeid::P_dispatch< LIST >
```

Use for protocol based dispatch stage.

Definition at line 308 of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

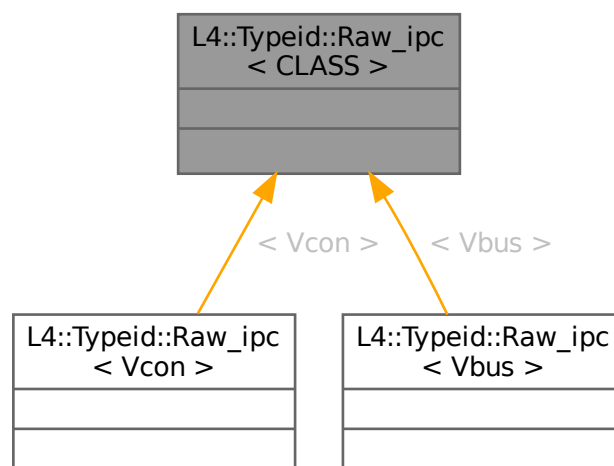
- [l4/sys/\\_\\_typeinfo.h](#)

### 16.221 L4::Typeid::Raw\_ipc< CLASS > Struct Template Reference

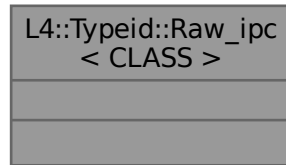
RPCs list for passing raw incoming IPC to the server object.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Typeid::Raw\_ipc< CLASS >:



Collaboration diagram for L4::Typeid::Raw\_ipc< CLASS >:



### 16.221.1 Detailed Description

```
template<typename CLASS>
struct L4::Typeid::Raw_ipc< CLASS >
```

RPCs list for passing raw incoming IPC to the server object.

#### Template Parameters

CLASS	The type of the interface (e.g., <a href="#">L4::lcu</a> )
-------	------------------------------------------------------------

This template allows to have fully handcrafted IPC protocols.

Definition at line [412](#) of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

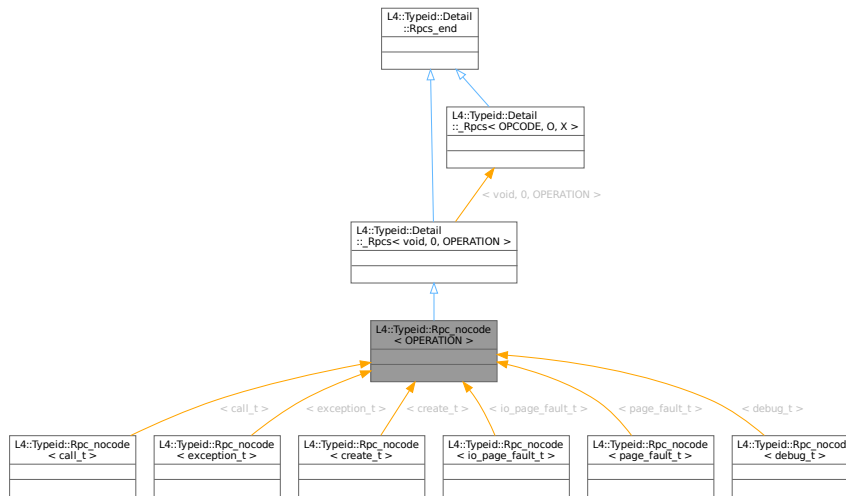
- [l4/sys/\\_\\_typeinfo.h](#)

## 16.222 L4::Typeid::Rpc\_nocode< OPERATION > Struct Template Reference

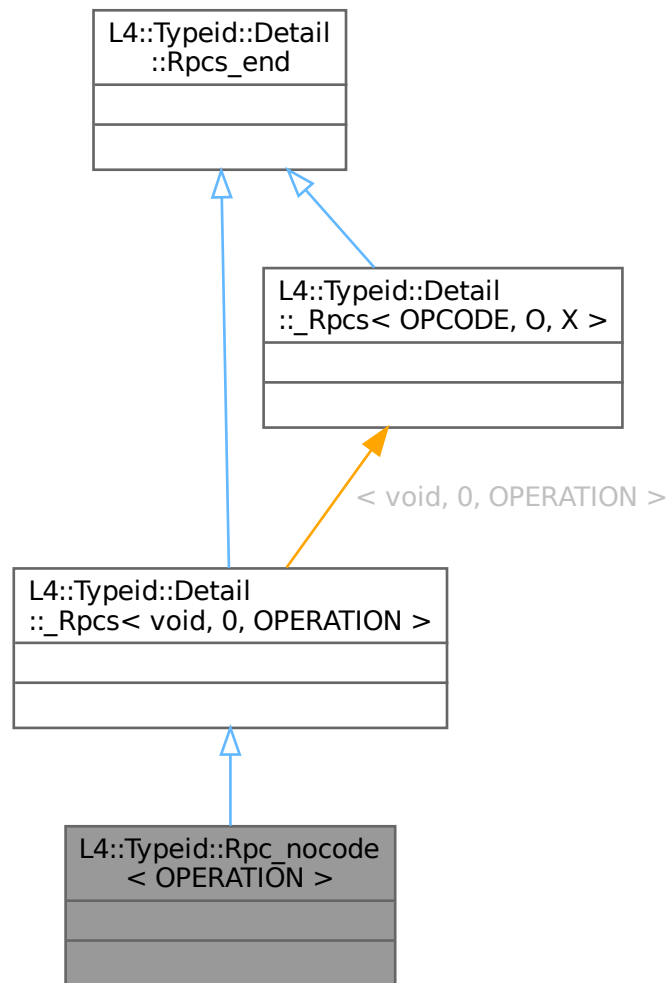
List of RPCs of an interface using a single operation without an opcode.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Typeid::Rpc\_nocode< OPERATION >:



Collaboration diagram for L4::Typeid::Rpc\_nocode< OPERATION >:



### 16.222.1 Detailed Description

```
template<typename OPERATION>
struct L4::Typeid::Rpc_nocode< OPERATION >
```

List of RPCs of an interface using a single operation without an opcode.

#### Template Parameters

<i>OPERATION</i>	The RPC operation as defined by L4_RPC etc.
------------------	---------------------------------------------

Definition at line 454 of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

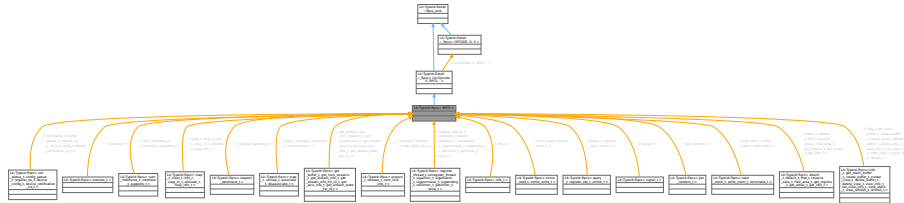
- [l4/sys/\\_\\_typeinfo.h](#)

## 16.223 L4::Typeid::Rpc< RPCS > Struct Template Reference

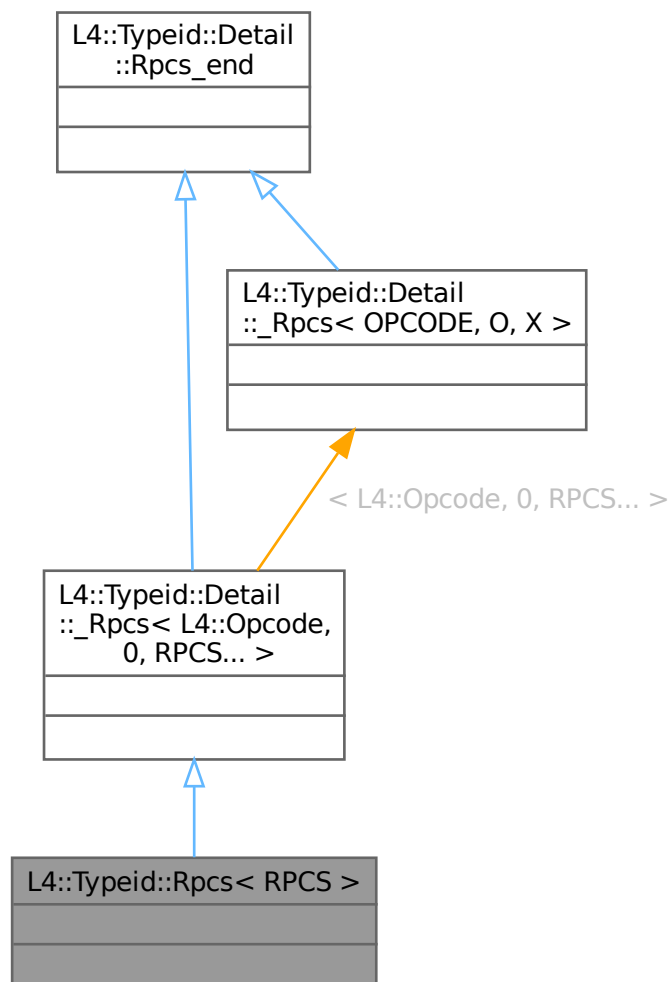
Standard list of RPCs of an interface.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Typeid::Rpc< RPCS >:



Collaboration diagram for L4::Typeid::Rpc< RPCS >:





### 16.223.1 Detailed Description

```
template<typename ... RPCS>
struct L4::Typeid::Rpc< RPCS >
```

Standard list of RPCs of an interface.

#### Template Parameters

<i>RPCS</i>	list of RPC types as defined by L4_RPC etc.
-------------	---------------------------------------------

This is the default list for RPC functions of an interface, it uses [L4::Opcode](#) as opcode type and uses opcodes starting from 0.

#### Examples

[examples/clntsrv/src/shared.h](#).

Definition at line [428](#) of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

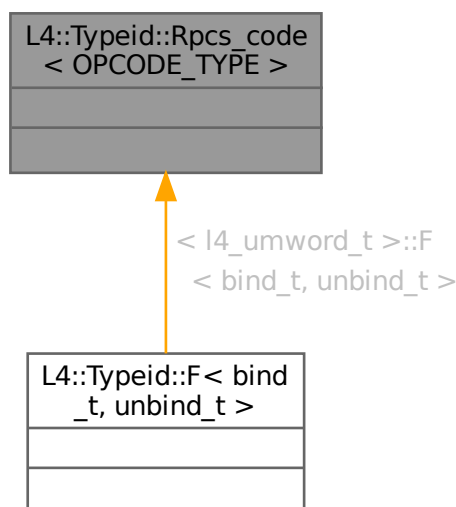
- [l4/sys/\\_\\_typeinfo.h](#)

## 16.224 L4::Typeid::Rpc\_code< OPCODE\_TYPE > Struct Template Reference

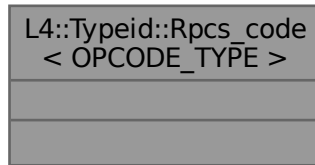
List of RPCs of an interface using a special opcode type.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Typeid::Rpc\_code< OPCODE\_TYPE >:



Collaboration diagram for L4::Typeid::Rpc\_code< OPCODE\_TYPE >:



## Data Structures

- struct [F](#)

### 16.224.1 Detailed Description

```
template<typename OPCODE_TYPE>
struct L4::Typeid::Rpc_code< OPCODE_TYPE >
```

List of RPCs of an interface using a special opcode type.

#### Template Parameters

<i>OPCODE_TYPE</i>	The data type of the opcode.
--------------------	------------------------------

List for RPC functions of an interface, using OPCODE\_TYPE as data type for the opcode, opcodes starting from 0.

Definition at line [439](#) of file [\\_\\_typeinfo.h](#).

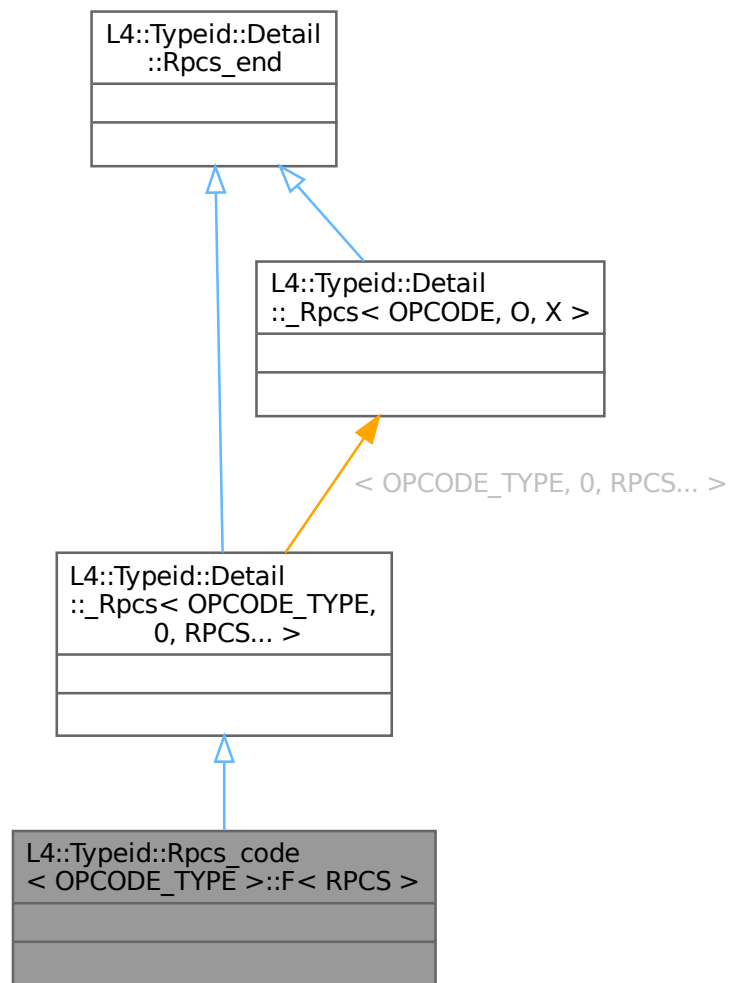
The documentation for this struct was generated from the following file:

- [l4/sys/\\_\\_typeinfo.h](#)

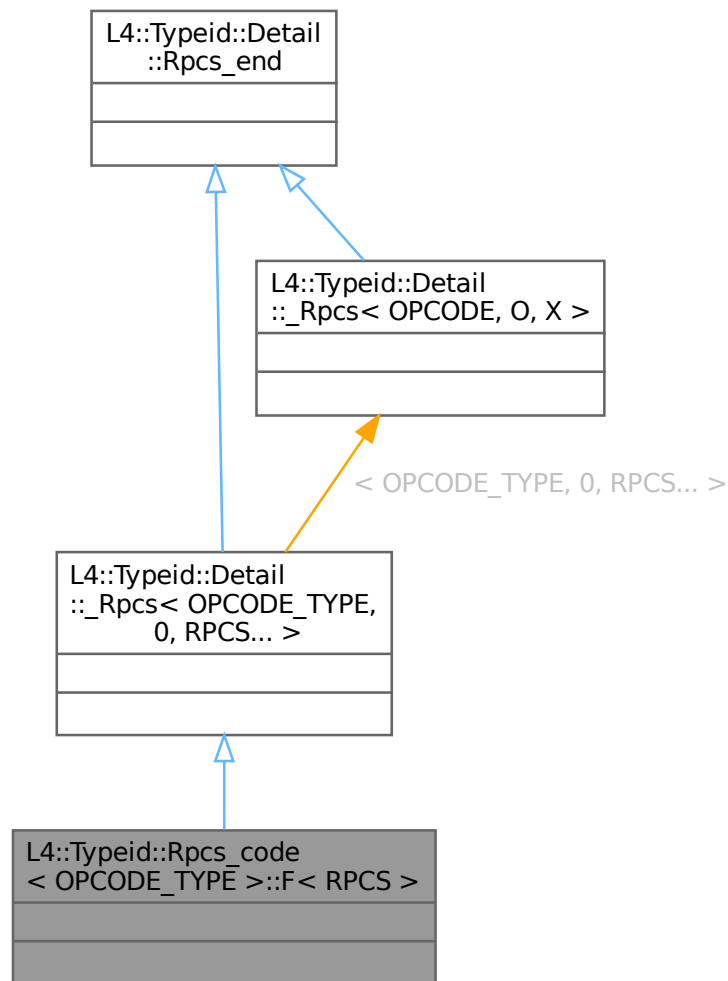
### 16.225 L4::Typeid::Rpc\_code< OPCODE\_TYPE >::F< RPCS > Struct Template Reference

```
#include <__typeinfo.h>
```

Inheritance diagram for L4::Typeid::Rpc\_code< OPCODE\_TYPE >::F< RPCS >:



Collaboration diagram for L4::Typeid::Rpc\_code< OPCODE\_TYPE >::F< RPCS >:



### 16.225.1 Detailed Description

```

template<typename OPCODE_TYPE>
template<typename ... RPCS>
struct L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS >

```

#### Template Parameters

<i>RPCS</i>	list of RPC types as defined by L4_RPC etc.
-------------	---------------------------------------------

Definition at line 445 of file [\\_\\_typeinfo.h](#).

The documentation for this struct was generated from the following file:

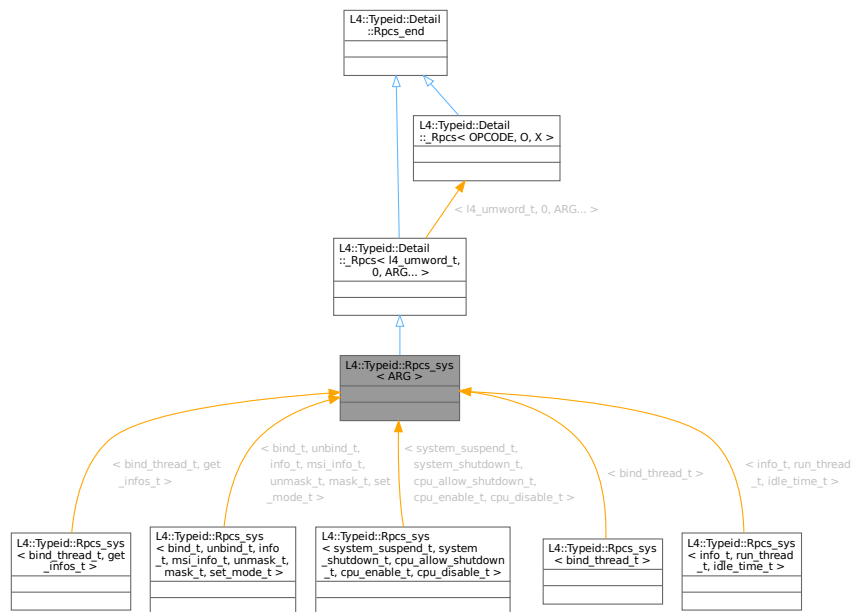
- [l4/sys/\\_\\_typeinfo.h](#)

## 16.226 L4::Typeid::Rpcsys< ARG > Struct Template Reference

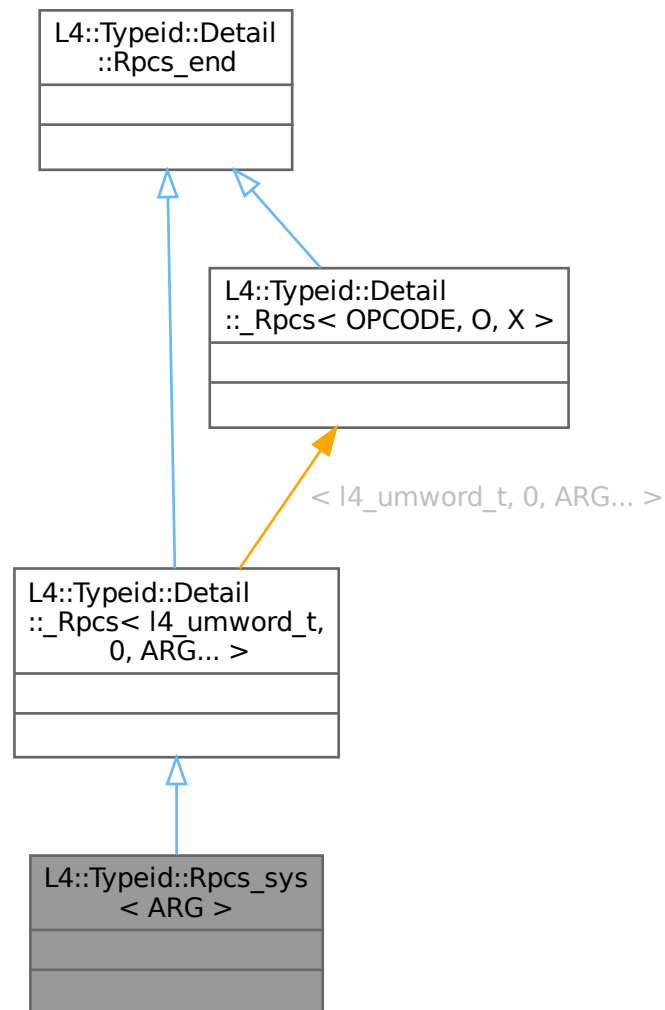
List of RPCs typically used for kernel interfaces.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Typeid::Rpcsys< ARG >:



Collaboration diagram for L4::Typeid::Rpcsys< ARG >:



### 16.226.1 Detailed Description

```
template<typename ... ARG>
struct L4::Typeid::Rpcsys< ARG >
```

List of RPCs typically used for kernel interfaces.

#### Template Parameters

<i>RPCS</i>	list of RPC types as defined by L4_RPC etc.
-------------	---------------------------------------------

This list of RPC functions uses [l4\\_umword\\_t](#) as type for the opcode as most kernel protocol do.



### 16.227.1 Detailed Description

```
template<bool V>  
struct L4::Types::Bool< V >
```

Boolean meta type.

#### Template Parameters

---



V	The boolean value
---	-------------------

Definition at line 288 of file [types](#).

The documentation for this struct was generated from the following file:

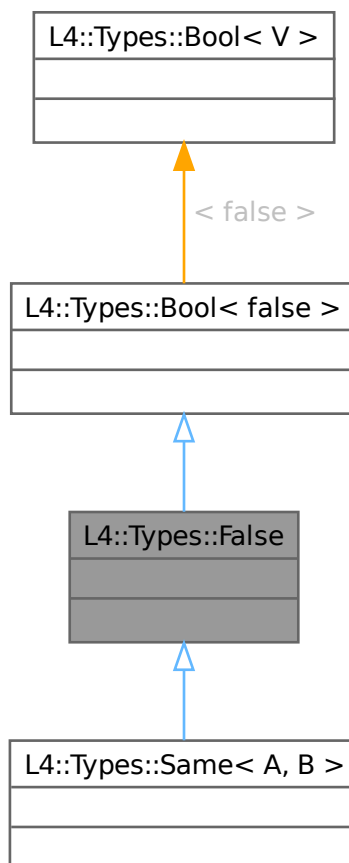
- [l4/sys/cxx/types](#)

## 16.228 L4::Types::False Struct Reference

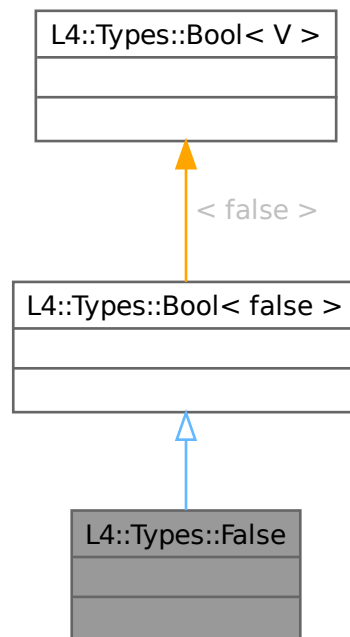
[False](#) meta value.

```
#include <types>
```

Inheritance diagram for L4::Types::False:



Collaboration diagram for L4::Types::False:



#### Additional Inherited Members

#### Public Types inherited from `L4::Types::Bool< false >`

- typedef `Bool< V >` **type**  
*The meta type itself.*

#### 16.228.1 Detailed Description

`False` meta value.

Definition at line 296 of file `types`.

The documentation for this struct was generated from the following file:

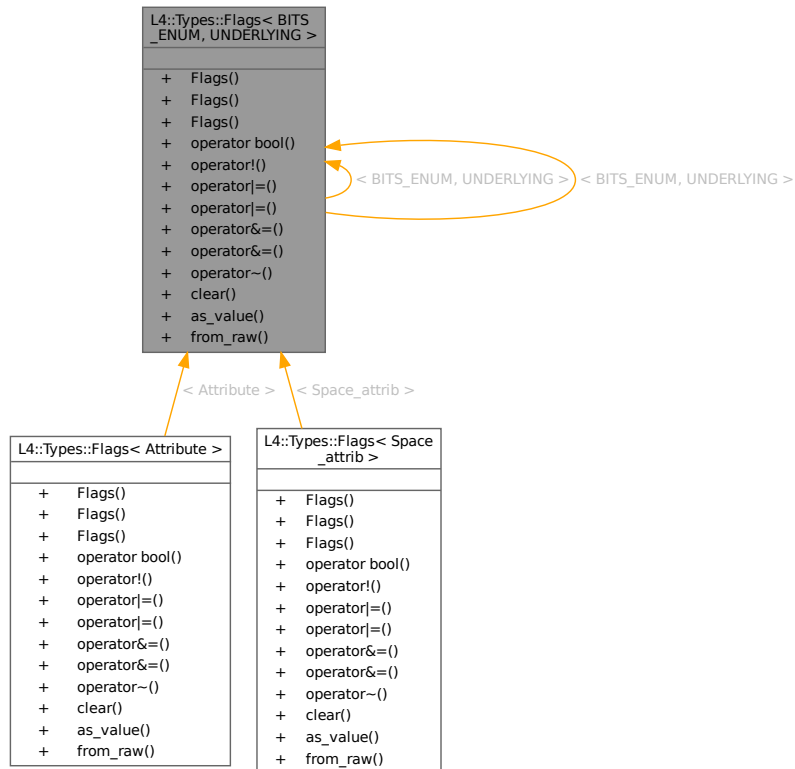
- `l4/sys/cxx/types`

## 16.229 L4::Types::Flags< BITS\_ENUM, UNDERLYING > Class Template Reference

Template for defining typical [Flags](#) bitmaps.

```
#include <types>
```

Inheritance diagram for L4::Types::Flags< BITS\_ENUM, UNDERLYING >:



Collaboration diagram for L4::Types::Flags< BITS\_ENUM, UNDERLYING >:

L4::Types::Flags< BITS_ENUM, UNDERLYING >
<ul style="list-style-type: none"> <li>+ Flags()</li> <li>+ Flags()</li> <li>+ Flags()</li> <li>+ operator bool()</li> <li>+ operator!()</li> <li>+ operator =()</li> <li>+ operator =()</li> <li>+ operator&amp;=()</li> <li>+ operator&amp;=()</li> <li>+ operator~()</li> <li>+ clear()</li> <li>+ as_value()</li> <li>+ from_raw()</li> </ul>

## Public Types

- enum [None\\_type](#) { [None](#) }  
*The none type to get an empty bitmap.*
- typedef UNDERLYING **value\_type**  
*type of the underlying value*
- typedef BITS\_ENUM **bits\_enum\_type**  
*enum type defining a name for each bit*
- typedef [Flags](#)< BITS\_ENUM, UNDERLYING > **type**  
*the Flags<> type itself*

## Public Member Functions

- [Flags](#) ([None\\_type](#))  
*Make an empty bitmap.*
- **Flags** ()  
*Make default [Flags](#).*
- [Flags](#) (BITS\_ENUM e)  
*Make flags from bit name.*
- **operator bool** () const  
*Support for `if (flags)` syntax (test for non-empty flags).*
- **bool operator!** () const  
*Support for `if (!flags)` syntax (test for empty flags).*

- **type & operator|** (type rhs)  
*Support | of two compatible [Flags](#) types.*
- **type & operator|** (bits\_enum\_type rhs)  
*Support | of [Flags](#) type and bit name.*
- **type & operator&=** (type rhs)  
*Support &= of two compatible [Flags](#) types.*
- **type & operator&=** (bits\_enum\_type rhs)  
*Support &= of [Flags](#) type and bit name.*
- **type operator~** () const  
*Support ~ for [Flags](#) types.*
- **type & clear** (bits\_enum\_type flag)  
*Clear the given flag.*
- **value\_type as\_value** () const  
*Get the underlying value.*

### Static Public Member Functions

- static **type from\_raw** (value\_type v)  
*Make flags from a raw value of [value\\_type](#).*

### Friends

- **type operator|** (type lhs, type rhs)  
*Support | of two compatible [Flags](#) types.*
- **type operator|** (type lhs, bits\_enum\_type rhs)  
*Support | of [Flags](#) type and bit name.*
- **type operator&** (type lhs, type rhs)  
*Support & of two compatible [Flags](#) types.*
- **type operator&** (type lhs, bits\_enum\_type rhs)  
*Support & of [Flags](#) type and bit name.*

## 16.229.1 Detailed Description

```
template<typename BITS_ENUM, typename UNDERLYING = unsigned long>
class L4::Types::Flags< BITS_ENUM, UNDERLYING >
```

Template for defining typical [Flags](#) bitmaps.

### Template Parameters

<i>BITS_ENUM</i>	enum type that defines a name for each bit in the bitmap. The values of the enum members must be the number of the bit ( <i>not</i> a mask).
<i>UNDERLYING</i>	The underlying data type used to represent the bitmap.

The resulting data type provides a type-safe version that allows bitwise `and` and `or` operations with the `BITS_↵` ENUM members. As well as, test for `0or !0`.

Example:

```
enum Test_flag
{
    Do_weak_tests,
    Do_strong_tests
};

typedef L4::Types::Flags<Test_flag> Test_flags;

Test_flags x = Do_weak_tests;

if (x & Do_strong_tests) { ... }
x |= Do_strong_tests;
if (x & Do_strong_tests) { ... }
```

Definition at line 52 of file [types](#).

## 16.229.2 Member Enumeration Documentation

### 16.229.2.1 None\_type

```
template<typename BITS_ENUM, typename UNDERLYING = unsigned long>
enum L4::Types::Flags::None_type
```

The none type to get an empty bitmap.

#### Enumerator

None	Use this to get an empty bitmap.
------	----------------------------------

Definition at line 68 of file [types](#).

## 16.229.3 Constructor & Destructor Documentation

### 16.229.3.1 Flags() [1/2]

```
template<typename BITS_ENUM, typename UNDERLYING = unsigned long>
L4::Types::Flags< BITS_ENUM, UNDERLYING >::Flags (
    None_type ) [inline]
```

Make an empty bitmap.

Usually used for implicit conversion from [Flags::None](#).

```
Flags x = Flags::None;
```

Definition at line 78 of file [types](#).

### 16.229.3.2 Flags() [2/2]

```
template<typename BITS_ENUM, typename UNDERLYING = unsigned long>
L4::Types::Flags< BITS_ENUM, UNDERLYING >::Flags (
    BITS_ENUM e) [inline]
```

Make flags from bit name.

Usually used for implicit conversion for a bit name.

```
Test_flags f = Do_strong_tests;
```

Definition at line 91 of file [types](#).

## 16.229.4 Member Function Documentation

### 16.229.4.1 clear()

```
template<typename BITS_ENUM, typename UNDERLYING = unsigned long>
type & L4::Types::Flags< BITS_ENUM, UNDERLYING >::clear (
    bits_enum_type flag) [inline]
```

Clear the given flag.

#### Parameters

<i>flag</i>	The flag that shall be cleared.
-------------	---------------------------------

`flags.clear(The_flag)` is a shortcut for `flags &= ~Flags(The_flag)`.

Definition at line 142 of file [types](#).

### 16.229.4.2 from\_raw()

```
template<typename BITS_ENUM, typename UNDERLYING = unsigned long>
type L4::Types::Flags< BITS_ENUM, UNDERLYING >::from_raw (
    value_type v) [inline], [static]
```

Make flags from a raw value of [value\\_type](#).

This function may be used for example in C wrapper code.

Definition at line 98 of file [types](#).

The documentation for this class was generated from the following file:

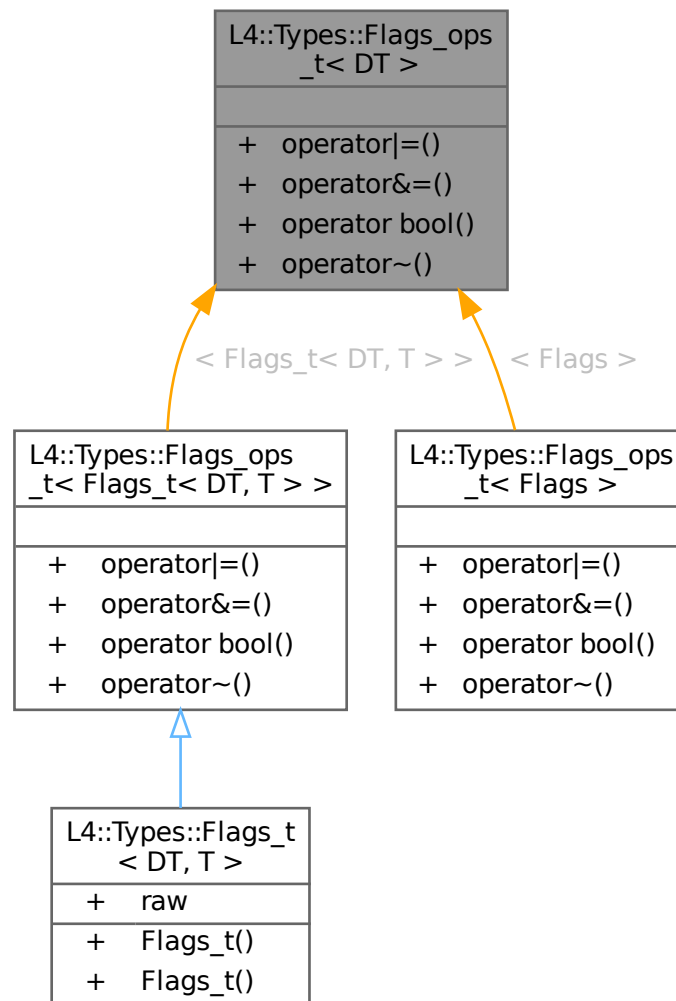
- [l4/sys/cxx/types](#)

## 16.230 L4::Types::Flags\_ops\_t< DT > Struct Template Reference

Mixin class to define a set of friend bitwise operators on `DT`.

```
#include <types>
```

Inheritance diagram for L4::Types::Flags\_ops\_t< DT >:





Collaboration diagram for L4::Types::Flags\_ops\_t< DT >:

L4::Types::Flags_ops_t< DT >
<ul style="list-style-type: none"> <li>+ operator =()</li> <li>+ operator&amp;=()</li> <li>+ operator bool()</li> <li>+ operator~()</li> </ul>

### Public Member Functions

- DT **operator|=** (DT r)  
*bitwise or assignment for DT*
- DT **operator&=** (DT r)  
*bitwise and assignment for DT*
- constexpr **operator bool** () const  
*explicit conversion to bool for tests*
- constexpr DT **operator~** () const  
*bitwise negation for DT*

### Friends

- constexpr DT **operator|** (DT l, DT r)  
*bitwise or for DT*
- constexpr DT **operator&** (DT l, DT r)  
*bitwise and for DT*
- constexpr bool **operator==** (DT l, DT r)  
*equality for DT*
- constexpr bool **operator!=** (DT l, DT r)  
*inequality for DT*

## 16.230.1 Detailed Description

```
template<typename DT>
struct L4::Types::Flags_ops_t< DT >
```

Mixin class to define a set of friend bitwise operators on DT.

### Template Parameters

<i>DT</i>	The type usually inheriting from <a href="#">Flags_ops_t</a> with a member <i>raw</i> of enum or integral type.
-----------	-----------------------------------------------------------------------------------------------------------------

Definition at line 220 of file [types](#).

The documentation for this struct was generated from the following file:

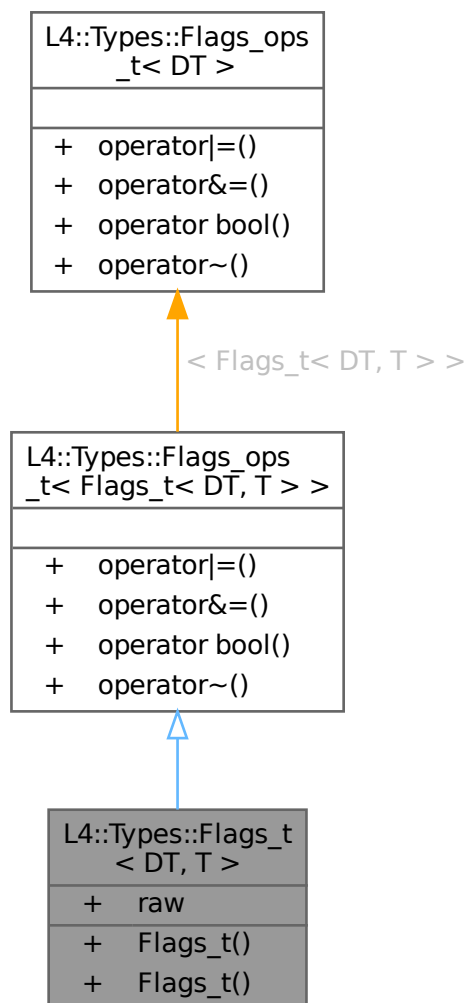
- [l4/sys/cxx/types](#)

## 16.231 L4::Types::Flags\_t< DT, T > Struct Template Reference

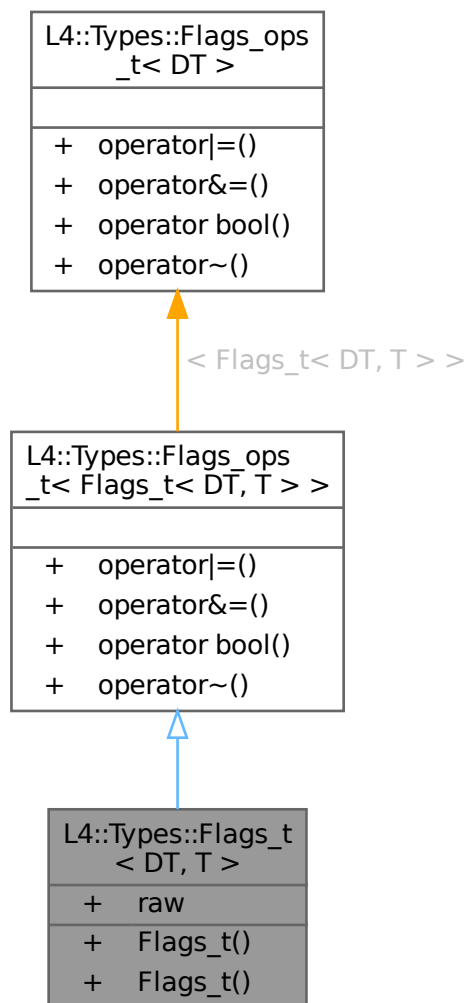
Template type to define a flags type with bitwise operations.

```
#include <types>
```

Inheritance diagram for L4::Types::Flags\_t< DT, T >:



Collaboration diagram for L4::Types::Flags\_t< DT, T >:



### Public Member Functions

- **Flags\_t**( )=default  
*Default (uninitializing) constructor.*
- constexpr **Flags\_t**( T f)  
*Explicit initialization from the underlying type.*

### Public Member Functions inherited from L4::Types::Flags\_ops\_t< Flags\_t< DT, T > >

- DT **operator|**= (DT r)  
*bitwise or assignment for DT*
- DT **operator&**= (DT r)  
*bitwise and assignment for DT*

- constexpr **operator bool** () const  
*explicit conversion to bool for tests*
- constexpr DT **operator~** () const  
*bitwise negation for DT*

### Data Fields

- **T raw**  
*Raw integral value.*

## 16.231.1 Detailed Description

```
template<typename DT, typename T>
struct L4::Types::Flags_t< DT, T >
```

Template type to define a flags type with bitwise operations.

### Template Parameters

<i>DT</i>	determinator type to make the resulting type unique (unused).
<i>T</i>	underlying type used to store the bits, usually an integral type.

Definition at line [272](#) of file [types](#).

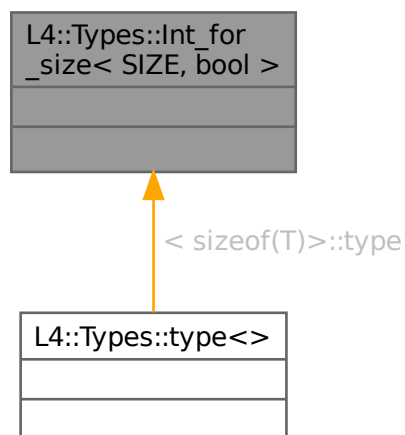
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/types](#)

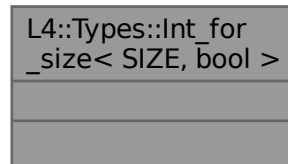
## 16.232 L4::Types::Int\_for\_size< SIZE, bool > Struct Template Reference

Metafunction to get an unsigned integral type for the given size.

Inheritance diagram for L4::Types::Int\_for\_size< SIZE, bool >:



Collaboration diagram for L4::Types::Int\_for\_size< SIZE, bool >:



### 16.232.1 Detailed Description

```
template<unsigned SIZE, bool = true>
struct L4::Types::Int_for_size< SIZE, bool >
```

Metafunction to get an unsigned integral type for the given size.

#### Template Parameters

<i>SIZE</i>	The size of the integer in bytes.
-------------	-----------------------------------

Definition at line 153 of file [types](#).

The documentation for this struct was generated from the following file:

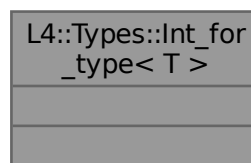
- [l4/sys/cxx/types](#)

## 16.233 L4::Types::Int\_for\_type< T > Struct Template Reference

Metafunction to get an integral type of the same size as T.

```
#include <types>
```

Collaboration diagram for L4::Types::Int\_for\_type< T >:



**Public Types**

- typedef [Int\\_for\\_size](#)< sizeof(T)>::type **type**  
*The resulting unsigned integer type with the size like T.*

**16.233.1 Detailed Description**

```
template<typename T>
struct L4::Types::Int_for_type< T >
```

Metafunction to get an integral type of the same size as T.

**Template Parameters**

<i>T</i>	The type for which an unsigned integral type with the same size is needed.
----------	----------------------------------------------------------------------------

Definition at line [180](#) of file [types](#).

The documentation for this struct was generated from the following file:

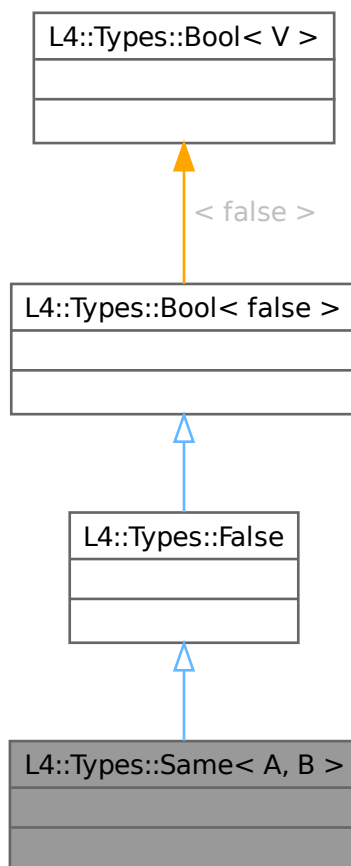
- [l4/sys/cxx/types](#)

**16.234 L4::Types::Same< A, B > Struct Template Reference**

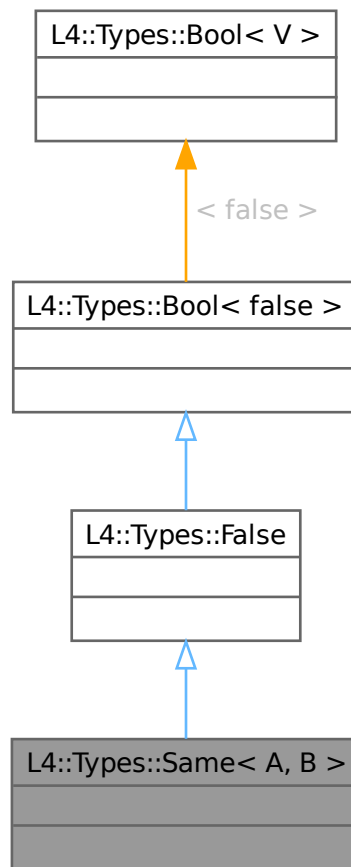
Compare two data types for equality.

```
#include <types>
```

Inheritance diagram for L4::Types::Same< A, B >:



Collaboration diagram for L4::Types::Same< A, B >:



#### Additional Inherited Members

#### Public Types inherited from L4::Types::Bool< false >

- typedef Bool< V > type  
The meta type itself.

#### 16.234.1 Detailed Description

```
template<typename A, typename B>
struct L4::Types::Same< A, B >
```

Compare two data types for equality.

#### Template Parameters

---



<i>A</i>	The first data type
<i>B</i>	The second data type

The result is the boolean [True](#) if A and B are the same types.

Definition at line [312](#) of file [types](#).

The documentation for this struct was generated from the following file:

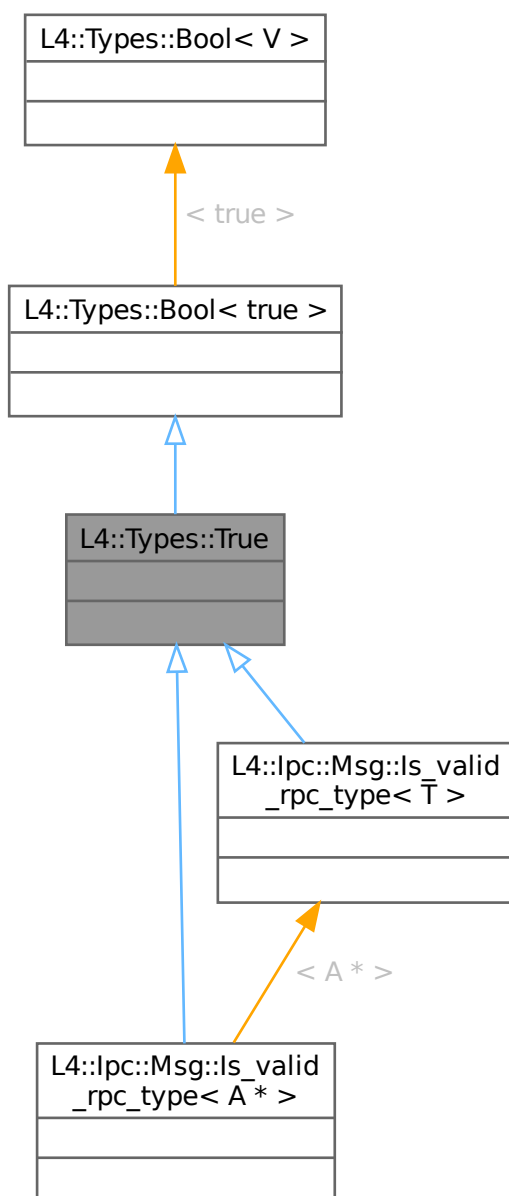
- [l4/sys/cxx/types](#)

## 16.235 L4::Types::True Struct Reference

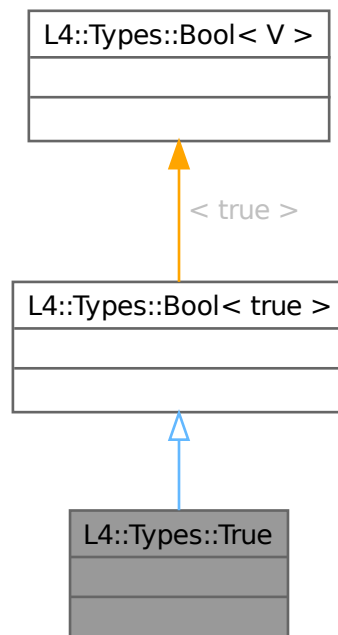
[True](#) meta value.

```
#include <types>
```

Inheritance diagram for L4::Types::True:



Collaboration diagram for L4::Types::True:



#### Additional Inherited Members

#### Public Types inherited from `L4::Types::Bool< true >`

- typedef `Bool< V >` **type**  
*The meta type itself.*

#### 16.235.1 Detailed Description

`True` meta value.

Definition at line 300 of file `types`.

The documentation for this struct was generated from the following file:

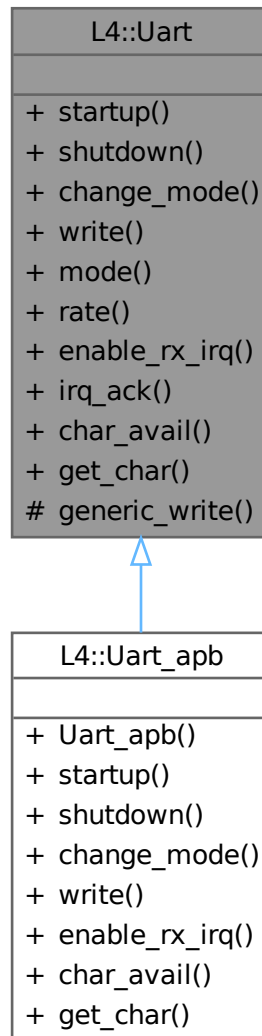
- `l4/sys/cxx/types`

## 16.236 L4::Uart Class Reference

[Uart](#) driver abstraction.

```
#include <uart_base.h>
```

Inheritance diagram for L4::Uart:



Collaboration diagram for L4::Uart:

L4::Uart
<ul style="list-style-type: none"> <li>+ startup()</li> <li>+ shutdown()</li> <li>+ change_mode()</li> <li>+ write()</li> <li>+ mode()</li> <li>+ rate()</li> <li>+ enable_rx_irq()</li> <li>+ irq_ack()</li> <li>+ char_avail()</li> <li>+ get_char()</li> <li># generic_write()</li> </ul>

### Public Member Functions

- virtual bool [startup](#) (Io\_register\_block const \*regs)=0  
*Start the UART driver.*
- virtual void [shutdown](#) ()=0  
*Terminate the UART driver.*
- virtual bool [change\\_mode](#) (Transfer\_mode m, Baud\_rate r)=0  
*Set certain parameters of the UART.*
- virtual int [write](#) (char const \*s, unsigned long count, bool blocking=true) const =0  
*Transmit a number of characters.*
- Transfer\_mode [mode](#) () const  
*Return the transfer mode.*
- Baud\_rate [rate](#) () const  
*Return the baud rate.*
- virtual bool [enable\\_rx\\_irq](#) (bool=true)  
*Enable the receive IRQ.*
- virtual void [irq\\_ack](#) ()  
*Acknowledge a received interrupt.*
- virtual int [char\\_avail](#) () const =0  
*Check if there is at least one character available for reading from the UART.*
- virtual int [get\\_char](#) (bool blocking=true) const =0  
*Read a character from the UART.*

### Protected Member Functions

- template<typename Uart\_driver, bool Timeout\_guard = true>  
int [generic\\_write](#) (char const \*s, unsigned long count, bool blocking=true) const  
*Internal function transmitting each character one-after-another and finally waiting that the transmission did actually finish.*

## 16.236.1 Detailed Description

[Uart](#) driver abstraction.

Definition at line 20 of file [uart\\_base.h](#).

## 16.236.2 Member Function Documentation

### 16.236.2.1 `change_mode()`

```
virtual bool L4::Uart::change_mode (  
    Transfer_mode m,  
    Baud_rate r) [pure virtual]
```

Set certain parameters of the UART.

#### Parameters

<i>m</i>	UART mode. Depends on the hardware.
<i>r</i>	Baud rate.

#### Return values

<i>true</i>	Mode setting succeeded (or was not performed at all).
<i>false</i>	Mode setting failed for some reason.

#### Note

Some drivers don't perform any mode setting at all and just return true.

Implemented in [L4::Uart\\_apb](#).

### 16.236.2.2 `char_avail()`

```
virtual int L4::Uart::char_avail () const [pure virtual]
```

Check if there is at least one character available for reading from the UART.

#### Returns

0 if there is no character available for reading, !=0 otherwise.

Implemented in [L4::Uart\\_apb](#).

### 16.236.2.3 `enable_rx_irq()`

```
virtual bool L4::Uart::enable_rx_irq (  
    bool = true) [inline], [virtual]
```

Enable the receive IRQ.

#### Return values

<i>true</i>	The RX IRQ was successfully enabled / disabled.
<i>false</i>	The RX IRQ couldn't be enabled / disabled. The driver does not support this operation.

Reimplemented in [L4::Uart\\_apb](#).

Definition at line 103 of file [uart\\_base.h](#).

#### 16.236.2.4 generic\_write()

```
template<typename Uart_driver, bool Timeout_guard = true>
int L4::Uart::generic_write (
    char const * s,
    unsigned long count,
    bool blocking = true) const [inline], [protected]
```

Internal function transmitting each character one-after-another and finally waiting that the transmission did actually finish.

##### Parameters

<i>s</i>	<a href="#">Buffer</a> containing the characters.
<i>count</i>	The number of characters to transmit.
<i>blocking</i>	If true, wait until there is space in the transmit buffer and also wait until every character was successful transmitted. Otherwise do not wait.

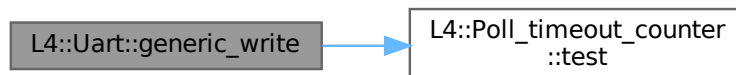
##### Returns

The number of successful written characters.

Definition at line 141 of file [uart\\_base.h](#).

References [L4::Poll\\_timeout\\_counter::test\(\)](#).

Here is the call graph for this function:



#### 16.236.2.5 get\_char()

```
virtual int L4::Uart::get_char (
    bool blocking = true) const [pure virtual]
```

Read a character from the UART.

##### Parameters

<i>blocking</i>	If true, wait until a character is available for reading. Otherwise do not wait and just return -1 if no character is available.
-----------------	----------------------------------------------------------------------------------------------------------------------------------

**Returns**

The actual character read from the UART.

Implemented in [L4::Uart\\_apb](#).

**16.236.2.6 mode()**

```
Transfer_mode L4::Uart::mode () const [inline]
```

Return the transfer mode.

**Returns**

The transfer mode.

Definition at line [86](#) of file [uart\\_base.h](#).

**16.236.2.7 rate()**

```
Baud_rate L4::Uart::rate () const [inline]
```

Return the baud rate.

**Returns**

The baud rate.

Definition at line [93](#) of file [uart\\_base.h](#).

**16.236.2.8 shutdown()**

```
virtual void L4::Uart::shutdown () [pure virtual]
```

Terminate the UART driver.

This includes disabling of interrupts.

Implemented in [L4::Uart\\_apb](#).

**16.236.2.9 startup()**

```
virtual bool L4::Uart::startup (
    Io_register_block const * regs) [pure virtual]
```

Start the UART driver.

**Parameters**



<i>regs</i>	IO register block of the UART.
-------------	--------------------------------

**Return values**

<i>true</i>	Startup succeeded.
<i>false</i>	Startup failed.

Implemented in [L4::Uart\\_apb](#).

**16.236.2.10 write()**

```
virtual int L4::Uart::write (
    char const * s,
    unsigned long count,
    bool blocking = true) const [pure virtual]
```

Transmit a number of characters.

**Parameters**

<i>s</i>	<a href="#">Buffer</a> containing the characters.
<i>count</i>	Number of characters to transmit.
<i>blocking</i>	If true, wait until there is space in the transmit buffer and also wait until every character was successful transmitted. Otherwise do not wait.

**Returns**

The number of successfully written characters.

Implemented in [L4::Uart\\_apb](#).

The documentation for this class was generated from the following file:

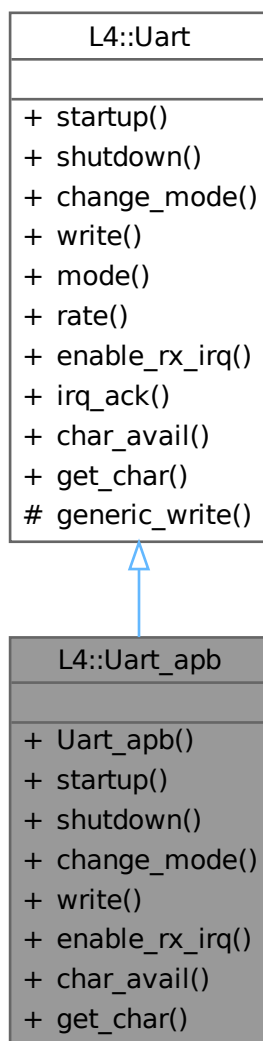
- pkg/drivers-frst/uart/include/uart\_base.h

**16.237 L4::Uart\_apb Class Reference**

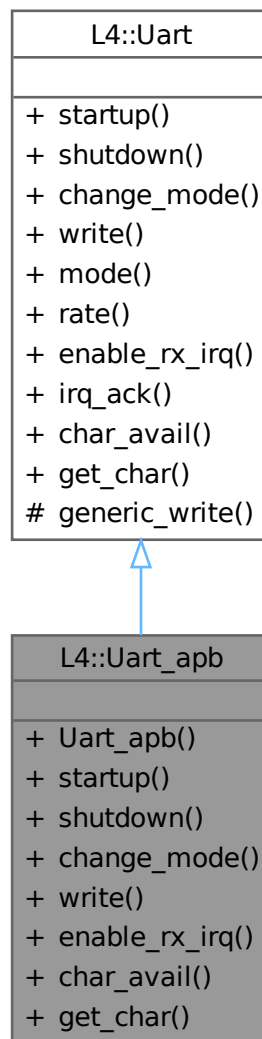
Driver for the Advanced Peripheral Bus (APB) UART from the Cortex-M System Design Kit (CMSDK).

```
#include <uart_apb.h>
```

Inheritance diagram for L4::Uart\_apb:



Collaboration diagram for L4::Uart\_apb:



## Public Member Functions

- **Uart\_apb** (unsigned freq)  
*freq == 0 means unknown and don't change baud rate*
- bool **startup** (lo\_register\_block const \*) override  
*Start the UART driver.*
- void **shutdown** () override  
*Terminate the UART driver.*
- bool **change\_mode** (Transfer\_mode m, Baud\_rate r) override  
*Set certain parameters of the UART.*
- int **write** (char const \*s, unsigned long count, bool blocking=true) const override  
*Transmit a number of characters.*

- bool `enable_rx_irq` (bool enable) override  
*Enable the receive IRQ.*
- int `char_avail` () const override  
*Check if there is at least one character available for reading from the UART.*
- int `get_char` (bool blocking=true) const override  
*Read a character from the UART.*

### Public Member Functions inherited from `L4::Uart`

- Transfer\_mode `mode` () const  
*Return the transfer mode.*
- Baud\_rate `rate` () const  
*Return the baud rate.*
- virtual void `irq_ack` ()  
*Acknowledge a received interrupt.*

### Additional Inherited Members

### Protected Member Functions inherited from `L4::Uart`

- template<typename Uart\_driver, bool Timeout\_guard = true>  
int `generic_write` (char const \*s, unsigned long count, bool blocking=true) const  
*Internal function transmitting each character one-after-another and finally waiting that the transmission did actually finish.*

## 16.237.1 Detailed Description

Driver for the Advanced Peripheral Bus (APB) UART from the Cortex-M System Design Kit (CMSDK).

Definition at line 17 of file `uart_apb.h`.

## 16.237.2 Member Function Documentation

### 16.237.2.1 `change_mode()`

```
bool L4::Uart_apb::change_mode (
    Transfer_mode m,
    Baud_rate r) [override], [virtual]
```

Set certain parameters of the UART.

#### Parameters

<i>m</i>	UART mode. Depends on the hardware.
<i>r</i>	Baud rate.

#### Return values

---

<i>true</i>	Mode setting succeeded (or was not performed at all).
<i>false</i>	Mode setting failed for some reason.

#### Note

Some drivers don't perform any mode setting at all and just return true.

Implements [L4::Uart](#).

#### 16.237.2.2 char\_avail()

```
int L4::Uart_apb::char_avail () const [override], [virtual]
```

Check if there is at least one character available for reading from the UART.

#### Returns

0 if there is no character available for reading, !=0 otherwise.

Implements [L4::Uart](#).

#### 16.237.2.3 enable\_rx\_irq()

```
bool L4::Uart_apb::enable_rx_irq (  
    bool ) [override], [virtual]
```

Enable the receive IRQ.

#### Return values

<i>true</i>	The RX IRQ was successfully enabled / disabled.
<i>false</i>	The RX IRQ couldn't be enabled / disabled. The driver does not support this operation.

Reimplemented from [L4::Uart](#).

#### 16.237.2.4 get\_char()

```
int L4::Uart_apb::get_char (  
    bool blocking = true) const [override], [virtual]
```

Read a character from the UART.

#### Parameters

---

<i>blocking</i>	If true, wait until a character is available for reading. Otherwise do not wait and just return -1 if no character is available.
-----------------	----------------------------------------------------------------------------------------------------------------------------------

#### Returns

The actual character read from the UART.

Implements [L4::Uart](#).

#### 16.237.2.5 shutdown()

```
void L4::Uart_apb::shutdown () [override], [virtual]
```

Terminate the UART driver.

This includes disabling of interrupts.

Implements [L4::Uart](#).

#### 16.237.2.6 startup()

```
bool L4::Uart_apb::startup (
    Io_register_block const * regs) [override], [virtual]
```

Start the UART driver.

#### Parameters

<i>regs</i>	IO register block of the UART.
-------------	--------------------------------

#### Return values

<i>true</i>	Startup succeeded.
<i>false</i>	Startup failed.

Implements [L4::Uart](#).

#### 16.237.2.7 write()

```
int L4::Uart_apb::write (
    char const * s,
    unsigned long count,
    bool blocking = true) const [override], [virtual]
```

Transmit a number of characters.

#### Parameters

<i>s</i>	<a href="#">Buffer</a> containing the characters.
<i>count</i>	Number of characters to transmit.
<i>blocking</i>	If true, wait until there is space in the transmit buffer and also wait until every character was successful transmitted. Otherwise do not wait.

**Returns**

The number of successfully written characters.

Implements [L4::Uart](#).

The documentation for this class was generated from the following file:

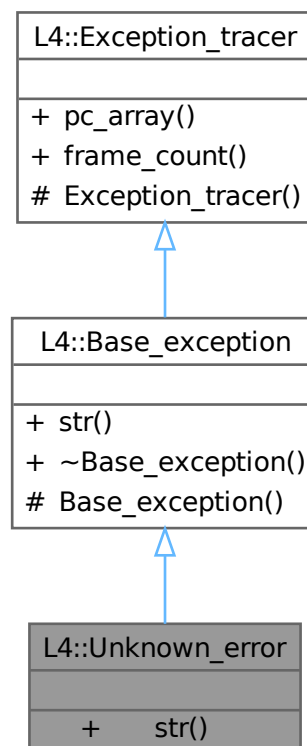
- pkg/drivers-frst/uart/include/uart\_apb.h

## 16.238 L4::Unknown\_error Class Reference

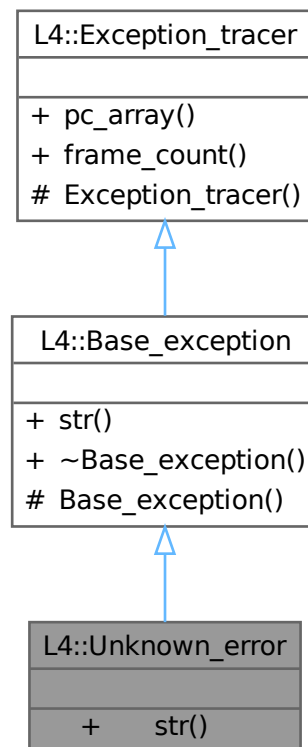
[Exception](#) for an unknown condition.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Unknown\_error:



Collaboration diagram for L4::Unknown\_error:



### Public Member Functions

- `char const * str ()` `const noexcept` override  
*Return a human readable string for the exception.*

### Public Member Functions inherited from [L4::Base\\_exception](#)

- `virtual ~Base_exception ()` `noexcept`  
*Destruction.*

### Public Member Functions inherited from [L4::Exception\\_tracer](#)

- `void const *const * pc_array ()` `const noexcept`  
*Get the array containing the call trace.*
- `int frame_count ()` `const noexcept`  
*Get the number of entries that are valid in the call trace.*



## Additional Inherited Members

### Protected Member Functions inherited from [L4::Base\\_exception](#)

- **Base\_exception** () noexcept  
*Create a base exception.*

### Protected Member Functions inherited from [L4::Exception\\_tracer](#)

- **Exception\_tracer** () noexcept  
*Create a back trace.*

## 16.238.1 Detailed Description

[Exception](#) for an unknown condition.

This error is usually used when a server returns an unknown return state to the client, this may indicate incompatible messages used by the client and the server.

Definition at line 208 of file [exceptions](#).

The documentation for this class was generated from the following file:

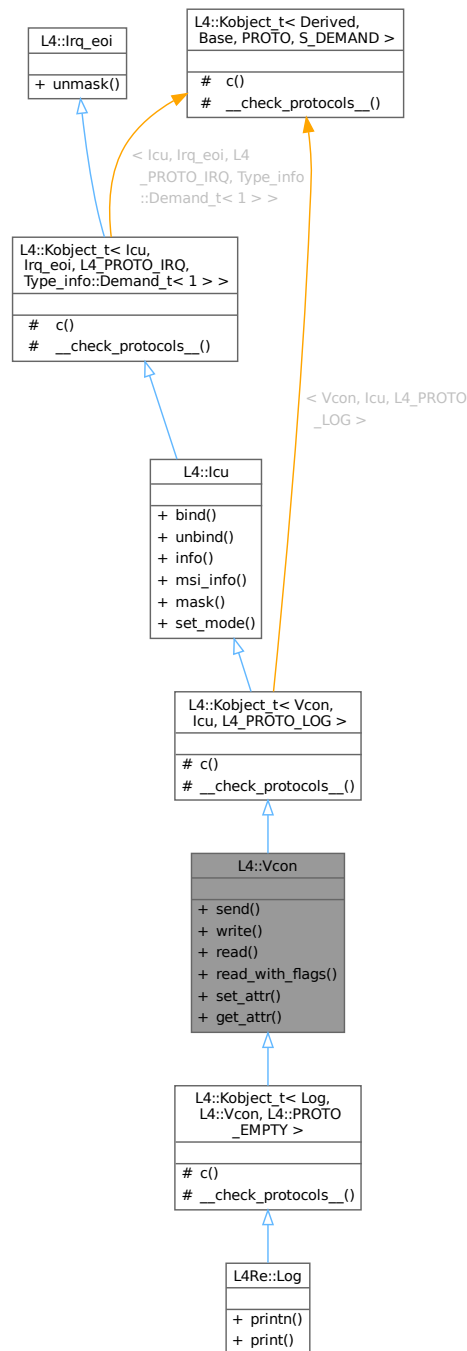
- [l4/cxx/exceptions](#)

## 16.239 L4::Vcon Class Reference

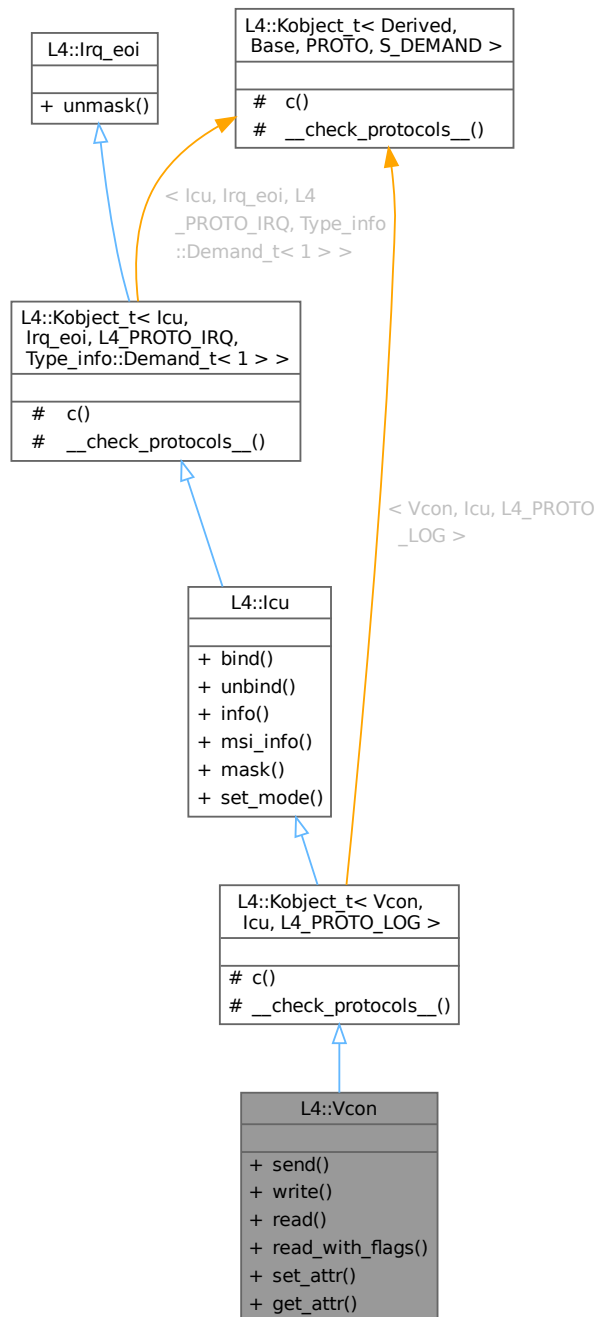
C++ [L4 Vcon](#) interface, see [Virtual Console](#) for the C interface.

```
#include <vcon>
```

Inheritance diagram for L4::Vcon:



Collaboration diagram for L4::Vcon:



## Public Member Functions

- `l4_msgtag_t send` (char const \*buf, unsigned size, `l4_utcb_t *utcb=l4_utcb()`) const noexcept  
Send data to *this* virtual console.
- `long write` (char const \*buf, unsigned size, `l4_utcb_t *utcb=l4_utcb()`) const noexcept  
Write data to *this* virtual console.
- `int read` (char \*buf, unsigned size, `l4_utcb_t *utcb=l4_utcb()`) const noexcept

- *Read data from `this` virtual console.*
- `int read_with_flags (char *buf, unsigned size, l4\_utcb\_t *utcb=l4\_utcb\(\)) const noexcept`  
*Read data from `this` virtual console which also returns flags.*
- `l4\_msgtag\_t set_attr (l4\_vcon\_attr\_t const *attr, l4\_utcb\_t *utcb=l4\_utcb\(\)) const noexcept`  
*Set the attributes of `this` virtual console.*
- `l4\_msgtag\_t get_attr (l4\_vcon\_attr\_t *attr, l4\_utcb\_t *utcb=l4\_utcb\(\)) const noexcept`  
*Get attributes of `this` virtual console.*

## Public Member Functions inherited from [L4::Icu](#)

- `l4\_msgtag\_t bind (unsigned irqnum, L4::Cap< Triggerable > irq, l4\_utcb\_t *utcb=l4\_utcb\(\)) noexcept`  
*Bind an interrupt line of an interrupt controller to an interrupt object.*
- `l4\_msgtag\_t unbind (unsigned irqnum, L4::Cap< Triggerable > irq, l4\_utcb\_t *utcb=l4\_utcb\(\)) noexcept`  
*Remove binding of an interrupt line from the interrupt controller object.*
- `l4\_msgtag\_t info (l4\_icu\_info\_t *info, l4\_utcb\_t *utcb=l4\_utcb\(\)) noexcept`  
*Get information about the ICU features.*
- `l4\_msgtag\_t msi_info (l4\_umword\_t irqnum, l4\_uint64\_t source, l4\_icu\_msi\_info\_t *msi_info)`  
*Get MSI info about IRQ.*
- `l4\_msgtag\_t mask (unsigned irqnum, l4\_umword\_t *label=0, l4\_timeout\_t to=L4\_IPC\_NEVER, l4\_utcb\_t *utcb=l4\_utcb\(\)) noexcept`  
*Mask an IRQ line.*
- `l4\_msgtag\_t set_mode (unsigned irqnum, l4\_umword\_t mode, l4\_utcb\_t *utcb=l4\_utcb\(\)) noexcept`  
*Set interrupt mode.*

## Public Member Functions inherited from [L4::Irq\\_eoi](#)

- `l4\_msgtag\_t unmask (unsigned irqnum, l4\_umword\_t *label=0, l4\_timeout\_t to=L4\_IPC\_NEVER, l4\_utcb\_t *utcb=l4\_utcb\(\)) noexcept`  
*Unmask the given interrupt line.*

## Additional Inherited Members

## Protected Types inherited from [L4::Kobject\\_t](#)< [Vcon](#), [Icu](#), [L4\\_PROTO\\_LOG](#) >

- `typedef Vcon Class`  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- `typedef Typeid::Iface< PROTO, Vcon > __Iface`  
*The interface description for the derived class.*
- `typedef Typeid::Merge_list< Typeid::Iface_list< __Iface >, typename Icu::__Iface_list > __Iface_list`  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Types inherited from

## [L4::Kobject\\_t](#)< [Icu](#), [Irq\\_eoi](#), [L4\\_PROTO\\_IRQ](#), [Type\\_info::Demand\\_t](#)< 1 > >

- `typedef Icu Class`  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- `typedef Typeid::Iface< PROTO, Icu > __Iface`  
*The interface description for the derived class.*
- `typedef Typeid::Merge_list< Typeid::Iface_list< __Iface >, typename Irq\_eoi::__Iface_list > __Iface_list`  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Member Functions inherited from [L4::Kobject\\_t< Vcon, Icu, L4\\_PROTO\\_LOG >](#)

- [L4::Cap< Class > c \(\)](#) const noexcept  
*Get the capability to ourselves.*

### Protected Member Functions inherited from [L4::Kobject\\_t< Icu, Irq\\_eoi, L4\\_PROTO\\_IRQ, Type\\_info::Demand\\_t< 1 > >](#)

- [L4::Cap< Class > c \(\)](#) const noexcept  
*Get the capability to ourselves.*

### Static Protected Member Functions inherited from [L4::Kobject\\_t< Vcon, Icu, L4\\_PROTO\\_LOG >](#)

- static void [\\_\\_check\\_protocols\\_\\_ \(\)](#) noexcept  
*Helper to check for protocol conflicts.*

### Static Protected Member Functions inherited from [L4::Kobject\\_t< Icu, Irq\\_eoi, L4\\_PROTO\\_IRQ, Type\\_info::Demand\\_t< 1 > >](#)

- static void [\\_\\_check\\_protocols\\_\\_ \(\)](#) noexcept  
*Helper to check for protocol conflicts.*

## 16.239.1 Detailed Description

C++ [L4 Vcon](#) interface, see [Virtual Console](#) for the C interface.

[L4::Vcon](#) is a virtual console for simple character-based input and output. The interrupt for read events is provided by the virtual key interrupt.

The [Vcon](#) interface inherits from [L4::Icu](#) and [L4::Irq\\_eoi](#) for managing the virtual key interrupt which, in contrast to hardware IRQs, implements a limited functionality:

- Only IRQ line 0 is supported, no MSI vectors.
- The IRQ is edge-triggered and the IRQ mode cannot be changed.
- As the IRQ is edge-triggered, it does not have to be explicitly unmasked.

A server implementing the virtual console protocol has a queue for input events. When the first input event is added to the empty queue, the virtual key interrupt is triggered. Further events are added to the queue without generating further interrupts. The queue is emptied when a client reads all queued input events.

#### Include File

```
#include <l4/sys/vcon>
```

See the [Virtual Console](#) for the C interface.

Definition at line 45 of file [vcon](#).

## 16.239.2 Member Function Documentation

### 16.239.2.1 get\_attr()

```
l4_msgtag_t L4::Vcon::get_attr (  
    l4_vcon_attr_t * attr,  
    l4_utcb_t * utcb = l4_utcb()) const [inline], [noexcept]
```

Get attributes of this virtual console.

#### Parameters

---

out	<i>attr</i>	Attribute structure. Contains the attributes after a successful call of this function.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

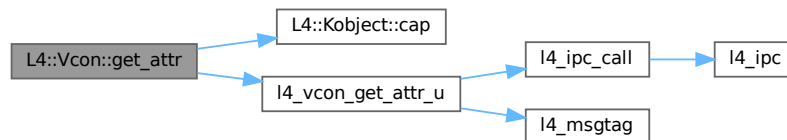
**Returns**

Syscall return tag.

Definition at line 151 of file [vcon](#).

References [L4::Kobject::cap\(\)](#), and [l4\\_vcon\\_get\\_attr\\_u\(\)](#).

Here is the call graph for this function:

**16.239.2.2 read()**

```

int L4::Vcon::read (
    char * buf,
    unsigned size,
    l4_utcb_t * utcb = l4_utcb()) const [inline], [noexcept]

```

Read data from [this](#) virtual console.

**Parameters**

out	<i>buf</i>	Pointer to data buffer.
	<i>size</i>	Size of the data buffer in bytes.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

**Return values**

<code>-L4_EPERM</code>	Insufficient permissions; see precondition.
<code>&gt;size</code>	More bytes to read, <code>size</code> bytes are in the buffer <code>buf</code> .
<code>&lt;=size</code>	Number of bytes read.

**Precondition**

The invoked [Vcon](#) capability must have the permission [L4\\_CAP\\_FPAGE\\_W](#).

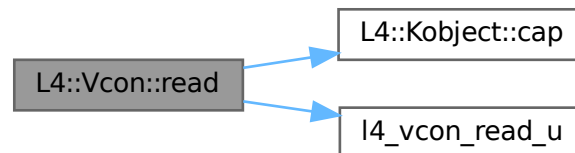
**Note**

Size must not exceed [L4\\_VCON\\_READ\\_SIZE](#).

Definition at line 98 of file [vcon](#).

References [L4::Kobject::cap\(\)](#), and [l4\\_vcon\\_read\\_u\(\)](#).

Here is the call graph for this function:

**16.239.2.3 read\_with\_flags()**

```

int L4::Vcon::read_with_flags (
    char * buf,
    unsigned size,
    l4_utcb_t * utcb = l4_utcb()) const [inline], [noexcept]
  
```

Read data from [this](#) virtual console which also returns flags.

**Parameters**

out	<i>buf</i>	Pointer to data buffer.
	<i>size</i>	Size of the data buffer in bytes.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

**Return values**

<a href="#">-L4_EPERM</a>	Insufficient permissions; see precondition.
<a href="#">&gt;size</a>	More bytes to read, <i>size</i> bytes are in the buffer <i>buf</i> .
<a href="#">&lt;=size</a>	Number of bytes read.



**Precondition**

The invoked [Vcon](#) capability must have the permission [L4\\_CAP\\_FPAGE\\_W](#).

If this function returns a positive value the caller can check the [L4\\_VCON\\_READ\\_STAT\\_BREAK](#) flag bit for a break condition. The bytes read can be obtained by masking the return value with [L4\\_VCON\\_READ\\_SIZE\\_MASK](#).

If a break condition is signaled, it is always the first event in the transmitted content, i.e. all characters supplied by this read call follow the break condition.

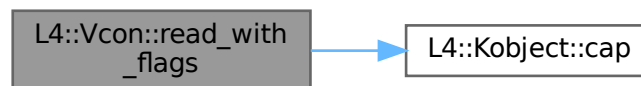
**Note**

Size must not exceed [L4\\_VCON\\_READ\\_SIZE](#).

Definition at line 125 of file [vcon](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:

**16.239.2.4 send()**

```

14_msgtag_t L4::Vcon::send (
    char const * buf,
    unsigned size,
    14_utcb_t * utcb = 14_utcb()) const [inline], [noexcept]
  
```

Send data to `this` virtual console.

**Parameters**

<i>buf</i>	Pointer to the data buffer.
<i>size</i>	Size of the data buffer in bytes.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">14_utcb</a> .

**Returns**

Syscall return tag

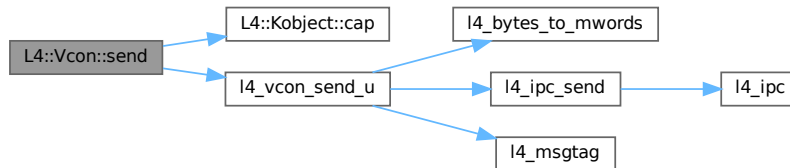
**Note**

Size must not exceed [L4\\_VCON\\_WRITE\\_SIZE](#), a proper value of the `size` parameter is NOT checked. Also, this function is a send only operation, this means there is no return value except for a failed send operation. Use [l4\\_ipc\\_error\(\)](#) to check for send errors, do not use [l4\\_error\(\)](#), as [l4\\_error\(\)](#) will always return an error.

Definition at line 65 of file [vcon](#).

References [L4::Kobject::cap\(\)](#), and [l4\\_vcon\\_send\\_u\(\)](#).

Here is the call graph for this function:

**16.239.2.5 set\_attr()**

```

l4_msgtag_t L4::Vcon::set_attr (
    l4_vcon_attr_t const * attr,
    l4_utcb_t * utcb = l4_utcb()) const [inline], [noexcept]
  
```

Set the attributes of this virtual console.

**Parameters**

<i>attr</i>	Attribute structure with the attributes for the virtual console.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

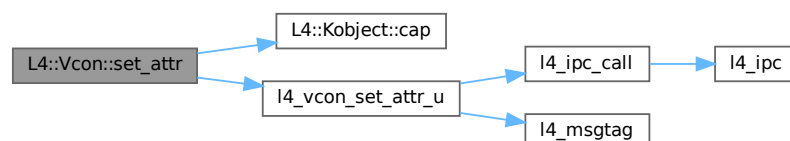
**Returns**

Syscall return tag.

Definition at line 138 of file [vcon](#).

References [L4::Kobject::cap\(\)](#), and [l4\\_vcon\\_set\\_attr\\_u\(\)](#).

Here is the call graph for this function:



## 16.239.2.6 write()

```
long L4::Vcon::write (
    char const * buf,
    unsigned size,
    l4_utcb_t * utcb = l4_utcb()) const [inline], [noexcept]
```

Write data to this virtual console.

## Parameters

<i>buf</i>	Pointer to the data buffer.
<i>size</i>	Size of the data buffer in bytes.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to <a href="#">l4_utcb</a> .

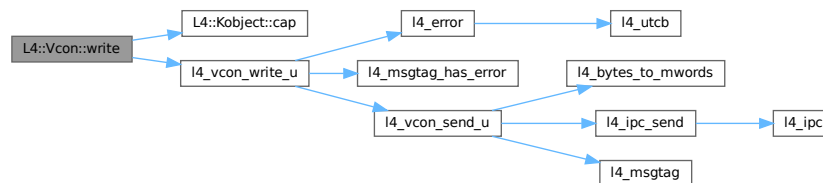
## Return values

< 0	Error.
>= 0	Number of bytes written to the virtual console.

Definition at line 79 of file [vcon](#).

References [L4::Kobject::cap\(\)](#), and [l4\\_vcon\\_write\\_u\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- [l4/sys/vcon](#)

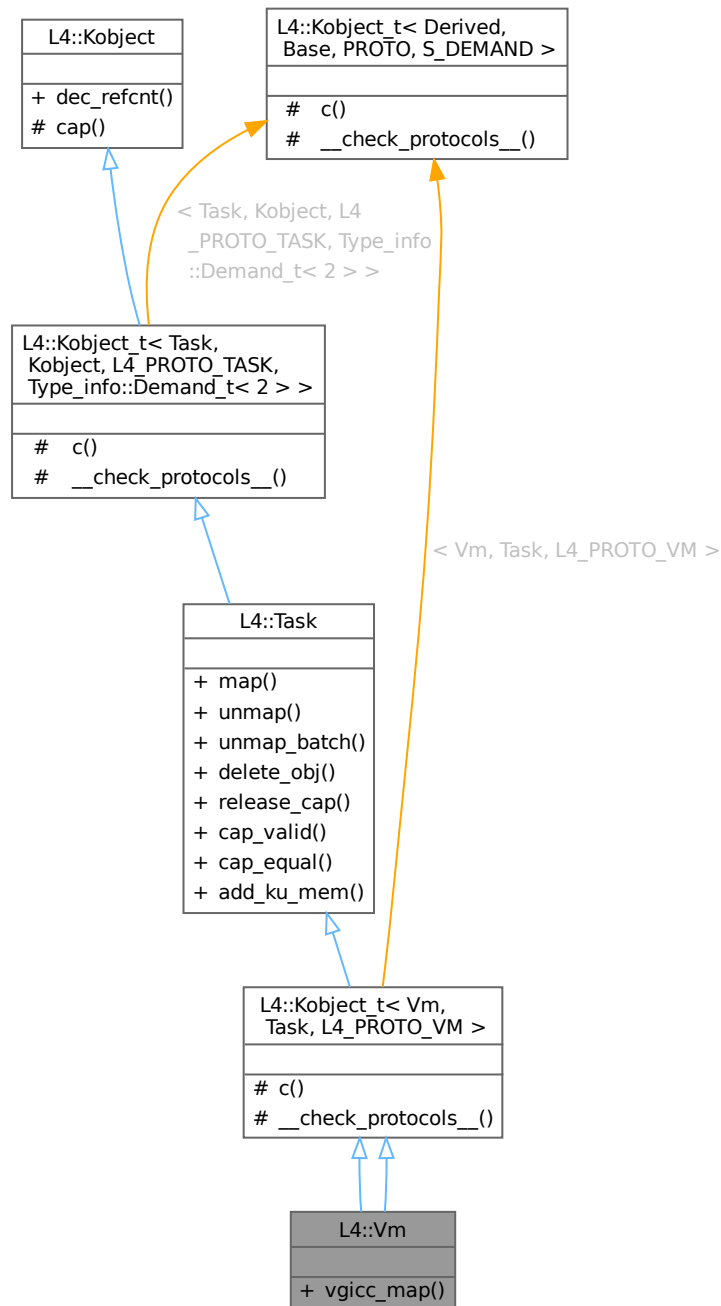
## 16.240 L4::Vm Class Reference

Virtual machine host address space.

```
#include <vm>
```



Collaboration diagram for L4::Vm:



## Public Member Functions

- [l4\\_msgtag\\_t](#) `vgicc_map` ([l4\\_fpage\\_t](#) const `vgicc_fpage`, [l4\\_utcb\\_t](#) \*`utcb`=[l4\\_utcb\(\)](#)) noexcept  
Map the GIC virtual CPU interface page to the task in case Fiasco detected a GIC version 2.

## Public Member Functions inherited from [L4::Task](#)

- [l4\\_msgtag\\_t map](#) ([Cap](#)< [Task](#) > const &src\_task, [l4\\_fpage\\_t](#) const &snd\_fpage, [l4\\_umword\\_t](#) snd\_base, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Map resources available in the source task to a destination task.*
- [l4\\_msgtag\\_t unmap](#) ([l4\\_fpage\\_t](#) const &fpage, [l4\\_umword\\_t](#) map\_mask, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Revoke rights from the task.*
- [l4\\_msgtag\\_t unmap\\_batch](#) ([l4\\_fpage\\_t](#) const \*fpages, unsigned num\_fpages, [l4\\_umword\\_t](#) map\_mask, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Revoke rights from a task.*
- [l4\\_msgtag\\_t delete\\_obj](#) ([L4::Cap](#)< void > obj, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Release capability and delete object.*
- [l4\\_msgtag\\_t release\\_cap](#) ([L4::Cap](#)< void > cap, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Release object capability.*
- [l4\\_msgtag\\_t cap\\_valid](#) ([Cap](#)< void > const &cap, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Check whether a capability is present (refers to an object).*
- [l4\\_msgtag\\_t cap\\_equal](#) ([Cap](#)< void > const &cap\_a, [Cap](#)< void > const &cap\_b, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Test whether two capabilities point to the same object with the same permissions (only considering selected permissions).*
- [l4\\_msgtag\\_t add\\_ku\\_mem](#) ([l4\\_fpage\\_t](#) \*fpage, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Add kernel-user memory.*

## Public Member Functions inherited from [L4::Kobject](#)

- [l4\\_msgtag\\_t dec\\_refcnt](#) ([l4\\_mword\\_t](#) diff, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)())  
*Decrement the in kernel reference counter for the object.*

## Additional Inherited Members

## Protected Types inherited from [L4::Kobject\\_t](#)< [Vm](#), [Task](#), [L4\\_PROTO\\_VM](#) >

- typedef [Vm](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< [PROTO](#), [Vm](#) > **\_\_iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_iface** >, typename [Task](#)::\_\_iface\_list > **\_\_iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Types inherited from [L4::Kobject\\_t](#)< [Task](#), [Kobject](#), [L4\\_PROTO\\_TASK](#), [Type\\_info::Demand\\_t](#)< 2 > >

- typedef [Task](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< [PROTO](#), [Task](#) > **\_\_iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_iface** >, typename [Kobject](#)::\_\_iface\_list > **\_\_iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Member Functions inherited from [L4::Kobject\\_t< Vm, Task, L4\\_PROTO\\_VM >](#)

- [L4::Cap< Class > c\(\)](#) const noexcept  
*Get the capability to ourselves.*

## Protected Member Functions inherited from [L4::Kobject\\_t< Task, Kobject, L4\\_PROTO\\_TASK, Type\\_info::Demand\\_t< 2 > >](#)

- [L4::Cap< Class > c\(\)](#) const noexcept  
*Get the capability to ourselves.*

## Protected Member Functions inherited from [L4::Kobject](#)

- [l4\\_cap\\_idx\\_t cap\(\)](#) const noexcept  
*Return capability selector.*

## Static Protected Member Functions inherited from [L4::Kobject\\_t< Vm, Task, L4\\_PROTO\\_VM >](#)

- static void [\\_\\_check\\_protocols\\_\\_\(\)](#) noexcept  
*Helper to check for protocol conflicts.*

## Static Protected Member Functions inherited from [L4::Kobject\\_t< Task, Kobject, L4\\_PROTO\\_TASK, Type\\_info::Demand\\_t< 2 > >](#)

- static void [\\_\\_check\\_protocols\\_\\_\(\)](#) noexcept  
*Helper to check for protocol conflicts.*

### 16.240.1 Detailed Description

Virtual machine host address space.

[L4::Vm](#) is a specialisation of [L4::Task](#), used for virtual machines. The microkernel employs an appropriate page-table format for hosting VMs, such as ePT on VT-x. On Arm, it offers a call to make the virtual GICC area available to the VM.

Definition at line 17 of file [\\_\\_vm-arm.h](#).

### 16.240.2 Member Function Documentation

#### 16.240.2.1 [vgicc\\_map\(\)](#)

```
l4_msgtag_t L4::Vm::vgicc_map (
    l4_fpage_t const vgicc_fpage,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Map the GIC virtual CPU interface page to the task in case Fiasco detected a GIC version 2.

#### Parameters

---

<i>vgicc_fpage</i>	Flexpage that describes an area in the address space of the destination task to map the vGICC page to.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

#### Returns

Syscall return tag.

Definition at line 30 of file [\\_\\_vm-arm.h](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

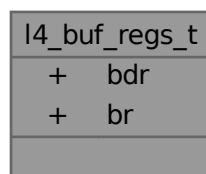
- [l4/sys/\\_\\_vm-arm.h](#)
- [l4/sys/vm](#)

## 16.241 l4\_buf\_regs\_t Struct Reference

Encapsulation of the buffer-registers block in the UTCB.

```
#include <utcb.h>
```

Collaboration diagram for `l4_buf_regs_t`:





## Data Fields

- [l4\\_umword\\_t bdr](#)  
*Buffer descriptor.*
- [l4\\_umword\\_t br](#) [L4\_UTCB\_GENERIC\_BUFFERS\_SIZE]  
*Buffer registers.*

### 16.241.1 Detailed Description

Encapsulation of the buffer-registers block in the UTCB.

Definition at line 82 of file [utcb.h](#).

The documentation for this struct was generated from the following file:

- l4/sys/[utcb.h](#)

## 16.242 l4\_exc\_regs\_t Struct Reference

UTCB structure for exceptions.

```
#include <utcb.h>
```

Collaboration diagram for l4\_exc\_regs\_t:

l4_exc_regs_t
+ pfa
+ err
+ r
+ sp
+ ulr
+ _dummy1
+ pc
+ cpsr
+ tpidruo
+ tpidrurw
and 31 more...

## Data Fields

- [l4\\_umword\\_t pfa](#)  
*page fault address*
- [l4\\_umword\\_t err](#)  
*error code*
- [l4\\_umword\\_t r \[13\]](#)  
*registers*
- [l4\\_umword\\_t sp](#)  
*stack pointer*
- [l4\\_umword\\_t ulr](#)  
*ulr*
- [l4\\_umword\\_t \\_dummy1](#)  
*dummy*
- [l4\\_umword\\_t pc](#)  
*pc*
- [l4\\_umword\\_t cpsr](#)  
*cpsr*
- [l4\\_umword\\_t tpidruro](#)  
*Thread-ID register.*
- [l4\\_umword\\_t tpidrurw](#)  
*Thread-ID register.*
- [l4\\_umword\\_t r15](#)  
*r15*
- [l4\\_umword\\_t r14](#)  
*r14*
- [l4\\_umword\\_t r13](#)  
*r13*
- [l4\\_umword\\_t r12](#)  
*r12*
- [l4\\_umword\\_t r11](#)  
*r11*
- [l4\\_umword\\_t r10](#)  
*r10*
- [l4\\_umword\\_t r9](#)  
*r9*
- [l4\\_umword\\_t r8](#)  
*r8*
- [l4\\_umword\\_t rdi](#)  
*rdi*
- [l4\\_umword\\_t rsi](#)  
*rsi*
- [l4\\_umword\\_t rbp](#)  
*rbp*
- [l4\\_umword\\_t rbx](#)  
*rbx*
- [l4\\_umword\\_t rdx](#)  
*rdx*
- [l4\\_umword\\_t rcx](#)  
*rcx*
- [l4\\_umword\\_t rax](#)

- rax*
- [l4\\_umword\\_t trapno](#)  
*trap number*
- [l4\\_umword\\_t dummy1](#)  
*dummy*
- [l4\\_umword\\_t ss](#)  
*stack segment register*
- [l4\\_umword\\_t es](#)  
*es register*
- [l4\\_umword\\_t ds](#)  
*ds register*
- [l4\\_umword\\_t gs](#)  
*gs register*
- [l4\\_umword\\_t fs](#)  
*fs register*
- [l4\\_umword\\_t edi](#)  
*edi register*
- [l4\\_umword\\_t esi](#)  
*esi register*
- [l4\\_umword\\_t ebp](#)  
*ebp register*
- [l4\\_umword\\_t ebx](#)  
*ebx register*
- [l4\\_umword\\_t edx](#)  
*edx register*
- [l4\\_umword\\_t ecx](#)  
*ecx register*
- [l4\\_umword\\_t eax](#)  
*eax register*

## 16.242.1 Detailed Description

UTCB structure for exceptions.

### Examples

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), and [examples/sys/start-with-exc/main.c](#).

Definition at line 27 of file [utcb.h](#).

## 16.242.2 Field Documentation

### 16.242.2.1 flags

```
l4\_umword\_t l4_exc_regs_t::flags
```

rflags

eflags

Definition at line 37 of file [utcb.h](#).

### 16.242.2.2 ss

`l4_umword_t l4_exc_regs_t::ss`

stack segment register

ss register

Definition at line 71 of file [utcb.h](#).

The documentation for this struct was generated from the following files:

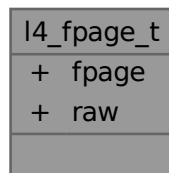
- [arm/l4/sys/utcb.h](#)
- [arm64/l4/sys/utcb.h](#)
- [amd64/l4/sys/utcb.h](#)
- [x86/l4/sys/utcb.h](#)

## 16.243 l4\_fpage\_t Union Reference

[L4](#) flexpage type.

```
#include <__l4_fpage.h>
```

Collaboration diagram for `l4_fpage_t`:



### Data Fields

- [l4\\_umword\\_t](#) **fpage**  
*Raw value.*
- [l4\\_umword\\_t](#) **raw**  
*Raw value.*

### 16.243.1 Detailed Description

[L4](#) flexpage type.

Definition at line 76 of file [\\_\\_l4\\_fpage.h](#).

The documentation for this union was generated from the following file:

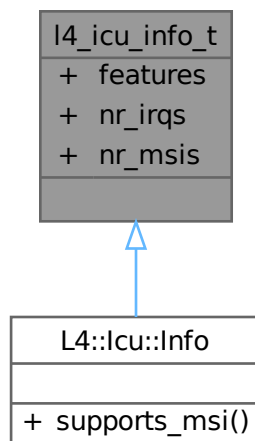
- [l4/sys/\\_\\_l4\\_fpage.h](#)

## 16.244 l4\_icu\_info\_t Struct Reference

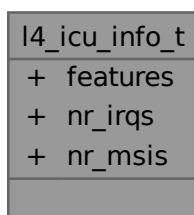
Info structure for an ICU.

```
#include <icu.h>
```

Inheritance diagram for l4\_icu\_info\_t:



Collaboration diagram for l4\_icu\_info\_t:



### Data Fields

- unsigned `features`  
*Feature flags.*
- unsigned `nr_irqs`  
*The number of IRQ lines supported by the ICU,.*
- unsigned `nr_msis`  
*The number of MSI vectors supported by the ICU,.*

### 16.244.1 Detailed Description

Info structure for an ICU.

This structure contains information about the features of an ICU.

See also

[l4\\_icu\\_info\(\)](#).

Definition at line 163 of file [icu.h](#).

### 16.244.2 Field Documentation

#### 16.244.2.1 features

```
unsigned l4_icu_info_t::features
```

Feature flags.

If [L4\\_ICU\\_FLAG\\_MSI](#) is set the ICU supports MSIs.

Definition at line 170 of file [icu.h](#).

Referenced by [L4::Icu::Info::supports\\_msi\(\)](#).

The documentation for this struct was generated from the following file:

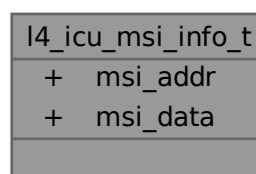
- [l4/sys/icu.h](#)

## 16.245 l4\_icu\_msi\_info\_t Struct Reference

Info to use for a specific MSI.

```
#include <icu.h>
```

Collaboration diagram for [l4\\_icu\\_msi\\_info\\_t](#):



## Data Fields

- [l4\\_uint64\\_t msi\\_addr](#)  
*Value to use as address when sending this MSI.*
- [l4\\_uint32\\_t msi\\_data](#)  
*Value to use as data written to msi\_addr, when sending this MSI.*

### 16.245.1 Detailed Description

Info to use for a specific MSI.

Definition at line 184 of file [icu.h](#).

The documentation for this struct was generated from the following file:

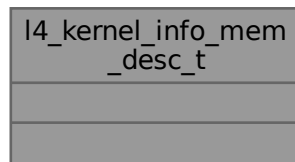
- [l4/sys/icu.h](#)

## 16.246 l4\_kernel\_info\_mem\_desc\_t Struct Reference

Memory descriptor data structure.

```
#include <memdesc.h>
```

Collaboration diagram for l4\_kernel\_info\_mem\_desc\_t:



### 16.246.1 Detailed Description

Memory descriptor data structure.

#### Note

This data type is opaque, and must be accessed by the accessor functions defined in this module.

Definition at line 77 of file [memdesc.h](#).

The documentation for this struct was generated from the following file:

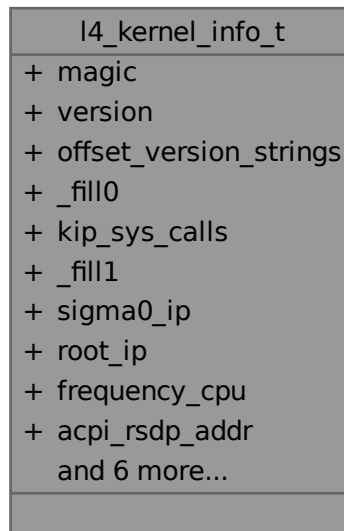
- [l4/sys/memdesc.h](#)

## 16.247 l4\_kernel\_info\_t Struct Reference

[L4 Kernel Interface Page](#).

```
#include <kip.h>
```

Collaboration diagram for l4\_kernel\_info\_t:



### Data Fields

- [l4\\_uint32\\_t](#) **magic**  
*Kernel Info Page identifier ("L4μK").*
- [l4\\_uint32\\_t](#) **version**  
*Kernel version.*
- [l4\\_uint8\\_t](#) **offset\_version\_strings**  
*offset to version string*
- [l4\\_uint8\\_t](#) **\_fill0** [3]  
*reserved*
- [l4\\_uint8\\_t](#) **kip\_sys\_calls**  
*pointer to system calls*
- [l4\\_uint8\\_t](#) **\_fill1** [2]  
*reserved*
- [l4\\_uint64\\_t](#) **sigma0\_ip**  
*Sigma0 instruction pointer.*
- [l4\\_uint64\\_t](#) **root\_ip**  
*Root task instruction pointer.*
- [l4\\_uint64\\_t](#) **frequency\_cpu**  
*CPU frequency in kHz.*



- [l4\\_uint64\\_t acpi\\_rsdp\\_addr](#)  
*ACPI RSDP/XSDP.*
- [l4\\_uint64\\_t dt\\_addr](#)  
*Device Tree.*
- [l4\\_uint64\\_t user\\_ptr](#)  
*user\_ptr*
- [l4\\_uint32\\_t scheduler\\_granularity](#)  
*for rounding time slices*
- [l4\\_uint32\\_t mem\\_descs](#)  
*memory descriptors relative to Kip*
- [l4\\_uint32\\_t mem\\_descs\\_num](#)  
*number of memory descriptors*
- [l4\\_uint64\\_t \\_res2 \[2\]](#)  
*internal - spare space*

### 16.247.1 Detailed Description

[L4 Kernel Interface Page.](#)

32-bit architecture may assume that the upper 32 bits of addresses is 0

Definition at line 36 of file [kip.h](#).

The documentation for this struct was generated from the following file:

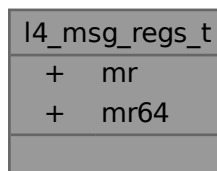
- [l4/sys/kip.h](#)

## 16.248 l4\_msg\_regs\_t Union Reference

Encapsulation of the message-register block in the UTCB.

```
#include <utcb.h>
```

Collaboration diagram for l4\_msg\_regs\_t:



## Data Fields

- [l4\\_umword\\_t](#) **mr** [L4\_UTCB\_GENERIC\_DATA\_SIZE]  
*Message registers.*
- [l4\\_uint64\\_t](#) **mr64** [L4\_UTCB\_GENERIC\_DATA\_SIZE/(sizeof([l4\\_uint64\\_t](#))/sizeof([l4\\_umword\\_t](#)))]  
*Message registers 64bit alias.*

### 16.248.1 Detailed Description

Encapsulation of the message-register block in the UTCB.

#### Examples

[examples/sys/utcb-ipc/main.c](#).

Definition at line 67 of file [utcb.h](#).

The documentation for this union was generated from the following file:

- [l4/sys/utcb.h](#)

## 16.249 l4\_msgtag\_t Struct Reference

Message tag data structure.

```
#include <types.h>
```

Collaboration diagram for [l4\\_msgtag\\_t](#):

<a href="#">l4_msgtag_t</a>
+ raw
+ label()
+ label()
+ words()
+ items()
+ flags()
+ is_page_fault()
+ is_exception()
+ is_sigma0()
+ is_io_page_fault()
+ has_error()

## Public Member Functions

- long **label** () const [L4\\_NOTHROW](#)  
*Get the protocol value.*
- void **label** (long v) [L4\\_NOTHROW](#)  
*Set the protocol value.*
- unsigned **words** () const [L4\\_NOTHROW](#)  
*Get the number of untyped words.*
- unsigned **items** () const [L4\\_NOTHROW](#)  
*Get the number of typed items.*
- unsigned **flags** () const [L4\\_NOTHROW](#)  
*Get the flags value.*
- bool **is\_page\_fault** () const [L4\\_NOTHROW](#)  
*Test if protocol indicates page-fault protocol.*
- bool **is\_exception** () const [L4\\_NOTHROW](#)  
*Test if protocol indicates exception protocol.*
- bool **is\_sigma0** () const [L4\\_NOTHROW](#)  
*Test if protocol indicates sigma0 protocol.*
- bool **is\_io\_page\_fault** () const [L4\\_NOTHROW](#)  
*Test if protocol indicates IO-page-fault protocol.*
- bool **has\_error** () const [L4\\_NOTHROW](#)  
*Test if flags indicate an error.*

## Data Fields

- [l4\\_mword\\_t](#) **raw**  
*raw value*

### 16.249.1 Detailed Description

Message tag data structure.

#### Include File

```
#include <l4/sys/types.h>
```

Describes the details of an IPC operation, in particular which parts of the UTCB have to be transmitted, and also flags to enable real-time and FPU extensions.

The message tag also contains a user-defined label that could be used to specify a protocol ID. Some negative values are reserved for kernel protocols such as page faults and exceptions.

The type must be treated completely opaque.

#### Examples

[examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#), [examples/libs/l4re/streammap/server.cc](#), [examples/sys/aliens/main.c](#), [examples/sys/ipc/ipc\\_example.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 153 of file [types.h](#).

## 16.249.2 Member Function Documentation

### 16.249.2.1 flags()

```
unsigned l4_msgtag_t::flags () const [inline]
```

Get the flags value.

The flags are a combination of the flags defined by [L4\\_msgtag\\_flags](#).

Definition at line 178 of file [types.h](#).

References [L4\\_NOTHROW](#), and [raw](#).

### 16.249.2.2 has\_error()

```
bool l4_msgtag_t::has_error () const [inline]
```

Test if flags indicate an error.

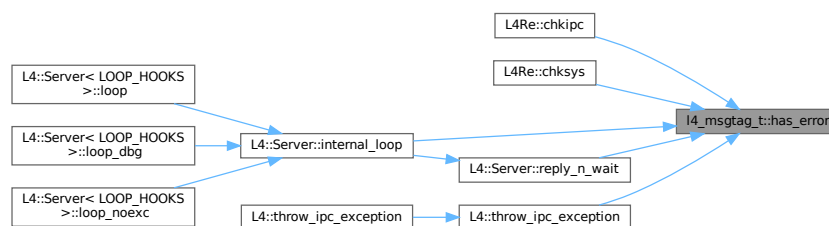
If true, the error code is stored in the UTCB, see [l4\\_utcb\\_tcr\(\)->error](#).

Definition at line 191 of file [types.h](#).

References [L4\\_MSGTAG\\_ERROR](#), [L4\\_NOTHROW](#), and [raw](#).

Referenced by [L4Re::chkipc\(\)](#), [L4Re::chksys\(\)](#), [L4::Server< LOOP\\_HOOKS >::internal\\_loop\(\)](#), [L4::Server< LOOP\\_HOOKS >::reply](#) and [L4::throw\\_ipc\\_exception\(\)](#).

Here is the caller graph for this function:



The documentation for this struct was generated from the following file:

- [l4/sys/types.h](#)

## 16.250 l4\_sched\_cpu\_set\_t Struct Reference

CPU sets.

```
#include <scheduler.h>
```

Collaboration diagram for l4\_sched\_cpu\_set\_t:

l4_sched_cpu_set_t
+ gran_offset
+ map
+ granularity()
+ offset()
+ set()

### Public Member Functions

- unsigned char [granularity](#) () const
- unsigned [offset](#) () const
- void [set](#) (unsigned char [granularity](#), unsigned [offset](#))

*Set offset and granularity.*

### Data Fields

- [l4\\_umword\\_t](#) [gran\\_offset](#)  
*Combination of granularity and offset.*
- [l4\\_umword\\_t](#) [map](#)  
*Bitmap of CPUs.*

### 16.250.1 Detailed Description

CPU sets.

#### Examples

[examples/sys/migrate/thread\\_migrate.cc](#).

Definition at line 58 of file [scheduler.h](#).

## 16.250.2 Member Function Documentation

### 16.250.2.1 granularity()

```
unsigned char l4_sched_cpu_set_t::granularity () const [inline]
```

#### Returns

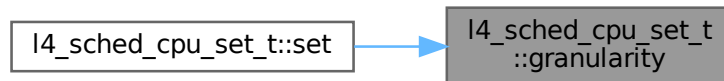
Get granularity value

Definition at line 81 of file [scheduler.h](#).

References [gran\\_offset](#).

Referenced by [set\(\)](#).

Here is the caller graph for this function:



### 16.250.2.2 offset()

```
unsigned l4_sched_cpu_set_t::offset () const [inline]
```

#### Returns

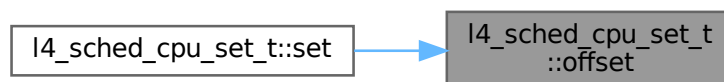
Get offset value

Definition at line 83 of file [scheduler.h](#).

References [gran\\_offset](#).

Referenced by [set\(\)](#).

Here is the caller graph for this function:



### 16.250.2.3 set()

```
void l4_sched_cpu_set_t::set (  
    unsigned char granularity,  
    unsigned offset) [inline]
```

Set offset and granularity.

#### Parameters

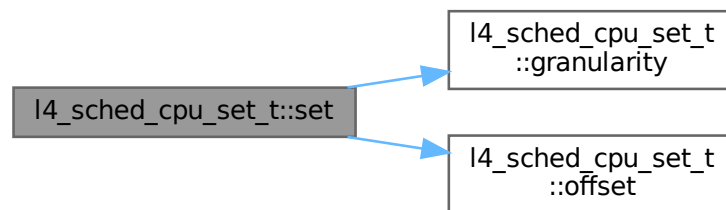
---

<i>granularity</i>	Granularity in log2 notation.
<i>offset</i>	Offset. Must be a multiple of $2^{\text{granularity}}$ .

Definition at line 90 of file [scheduler.h](#).

References [gran\\_offset](#), [granularity\(\)](#), and [offset\(\)](#).

Here is the call graph for this function:



## 16.250.3 Field Documentation

### 16.250.3.1 gran\_offset

`l4_umword_t l4_sched_cpu_set_t::gran_offset`

Combination of granularity and offset.

The granularity defines how many CPUs each bit in map describes. And the offset is the number of the first CPU described by the first bit in the bitmap.

#### Precondition

offset must be a multiple of  $2^{\text{granularity}}$ .

MSB	LSB
8bit granularity	24bit offset ..

Definition at line 72 of file [scheduler.h](#).

Referenced by [granularity\(\)](#), [L4::Scheduler::info\(\)](#), [l4\\_sched\\_cpu\\_set\(\)](#), [offset\(\)](#), and [set\(\)](#).

The documentation for this struct was generated from the following file:

- [l4/sys/scheduler.h](#)

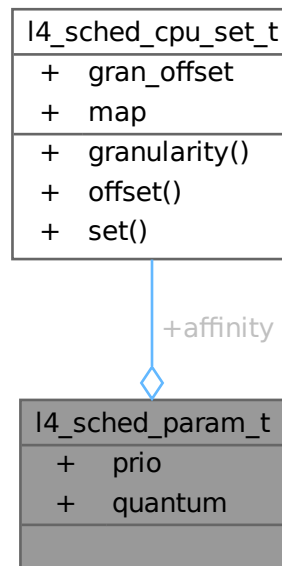


## 16.251 l4\_sched\_param\_t Struct Reference

Scheduler parameter set.

```
#include <scheduler.h>
```

Collaboration diagram for l4\_sched\_param\_t:



### Data Fields

- [l4\\_sched\\_cpu\\_set\\_t affinity](#)  
*CPU affinity.*
- [l4\\_umword\\_t prio](#)  
*Priority for scheduling.*
- [l4\\_umword\\_t quantum](#)  
*Timeslice in micro seconds.*

### 16.251.1 Detailed Description

Scheduler parameter set.

#### Examples

[examples/sys/aliens/main.c](#), [examples/sys/migrate/thread\\_migrate.cc](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 173 of file [scheduler.h](#).

## 16.251.2 Field Documentation

### 16.251.2.1 prio

`l4_umword_t l4_sched_param_t::prio`

Priority for scheduling.

The kernel supports priorities for userland threads in the range of 1..255. Priority 0 is reserved for the kernel.

Definition at line 182 of file [scheduler.h](#).

Referenced by [l4\\_sched\\_param\(\)](#).

The documentation for this struct was generated from the following file:

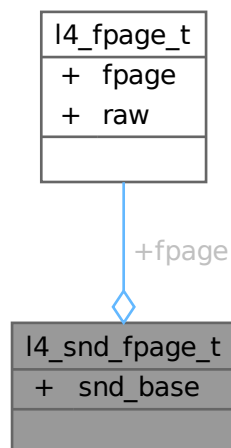
- [l4/sys/scheduler.h](#)

## 16.252 l4\_snd\_fpage\_t Struct Reference

Send-flexpage types.

```
#include <__l4_fpage.h>
```

Collaboration diagram for `l4_snd_fpage_t`:



### Data Fields

- `l4_umword_t snd_base`  
*Offset in receive window (send base).*
- `l4_fpage_t fpage`  
*Source flexpage descriptor.*

### 16.252.1 Detailed Description

Send-flexpage types.

Definition at line 99 of file [\\_\\_l4\\_fpage.h](#).

The documentation for this struct was generated from the following file:

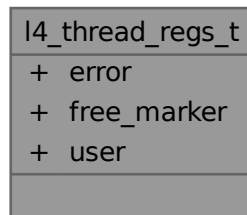
- l4/sys/\_\_l4\_fpage.h

## 16.253 l4\_thread\_regs\_t Struct Reference

Encapsulation of the thread-control-register block of the UTCB.

```
#include <utcb.h>
```

Collaboration diagram for l4\_thread\_regs\_t:



### Data Fields

- [l4\\_umword\\_t error](#)  
*System call error code (see [l4\\_ipc\\_tcr\\_error\\_t](#)).*
- [l4\\_umword\\_t free\\_marker](#)  
*Kernel free marker.*
- [l4\\_umword\\_t user](#) [3]  
*User values (ignored and preserved by the kernel).*

### 16.253.1 Detailed Description

Encapsulation of the thread-control-register block of the UTCB.

Definition at line 99 of file [utcb.h](#).

## 16.253.2 Field Documentation

### 16.253.2.1 error

```
l4_umword_t l4_thread_regs_t::error
```

System call error code (see [l4\\_ipc\\_tcr\\_error\\_t](#)).

If the kernel indicates an error in the message tag (see [l4\\_msgtag\\_has\\_error\(\)](#) and [l4\\_msgtag\\_t::has\\_error\(\)](#)), the kernel writes the error code to this field.

Definition at line [106](#) of file [utcb.h](#).

### 16.253.2.2 free\_marker

```
l4_umword_t l4_thread_regs_t::free_marker
```

Kernel free marker.

The kernel sets this field to zero as soon as it is guaranteed that the kernel does not use the UTCB anymore for the bound thread. This usually happens while a thread is deleted. However, it is not defined when exactly the kernel sets the field. In particular, the point in time is not necessarily related to any IPC.

Userland may use this field for determining if a UTCB can be re-used for another thread. Note that, in order to make use of that feature, userland has to set this field to a non-zero value when a thread is bound with this UTCB.

Definition at line [120](#) of file [utcb.h](#).

The documentation for this struct was generated from the following file:

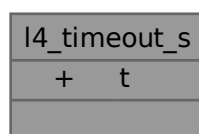
- [l4/sys/utcb.h](#)

## 16.254 l4\_timeout\_s Struct Reference

Basic timeout specification.

```
#include <__timeout.h>
```

Collaboration diagram for `l4_timeout_s`:



**Data Fields**

- [l4\\_uint16\\_t t](#)  
*timeout value*

**16.254.1 Detailed Description**

Basic timeout specification.

If bit 15 == 0, basically a floating point number with 10 bits mantissa and 5 bits exponent ( $t = m \cdot 2^e$ ).

If the mantissa is zero, the exponent encodes special values, see [L4\\_IPC\\_TIMEOUT\\_0](#) and [L4\\_IPC\\_TIMEOUT\\_NEVER](#).

If bit 15 == 1 the timeout is absolute and the lower 6 bits encode the index of the UTCB buffer register(s) holding the absolute 64-bit timeout value. On 32-bit systems, two consecutive UTCB buffer registers are used.

Definition at line 40 of file [\\_\\_timeout.h](#).

The documentation for this struct was generated from the following file:

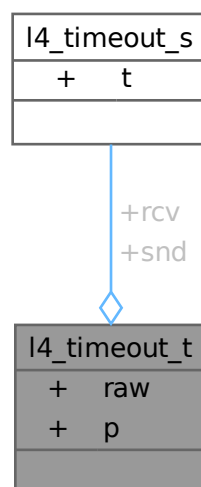
- [l4/sys/\\_\\_timeout.h](#)

**16.255 l4\_timeout\_t Union Reference**

Timeout pair.

```
#include <__timeout.h>
```

Collaboration diagram for l4\_timeout\_t:



## Data Fields

- `l4_uint32_t raw`  
*raw value*
- struct {
  - `l4_timeout_s rcv`  
*receive timeout*
  - `l4_timeout_s snd`  
*send timeout*
- `p`  
*combined timeout*

### 16.255.1 Detailed Description

Timeout pair.

For IPC there are usually a send and a receive timeout. So this structure contains a pair of timeouts.

Definition at line 52 of file `__timeout.h`.

The documentation for this union was generated from the following file:

- `l4/sys/__timeout.h`

### 16.256 l4\_vcon\_attr\_t Struct Reference

Vcon attribute structure.

```
#include <vcon.h>
```

Collaboration diagram for `l4_vcon_attr_t`:

<code>l4_vcon_attr_t</code>
+ <code>i_flags</code>
+ <code>o_flags</code>
+ <code>l_flags</code>
+ <code>set_raw()</code>

## Public Member Functions

- void `set_raw()`  
*Set terminal attributes to disable all special processing.*

## Data Fields

- [l4\\_umword\\_t i\\_flags](#)  
*input flags*
- [l4\\_umword\\_t o\\_flags](#)  
*output flags*
- [l4\\_umword\\_t l\\_flags](#)  
*local flags*

## 16.256.1 Detailed Description

Vcon attribute structure.

The flags members can be a combination of their respective enums.

See also

[L4\\_vcon\\_i\\_flags](#)  
[L4\\_vcon\\_o\\_flags](#)  
[L4\\_vcon\\_l\\_flags](#)

Examples

[examples/sys/isr/main.c](#).

Definition at line 187 of file [vcon.h](#).

## 16.256.2 Member Function Documentation

### 16.256.2.1 set\_raw()

```
void l4_vcon_attr_t::set_raw () [inline]
```

Set terminal attributes to disable all special processing.

Removes all flags that would mangle the read or written characters. Also disables echoing and any special processing of characters.

Definition at line 450 of file [vcon.h](#).

References [l4\\_vcon\\_set\\_attr\\_raw\(\)](#).

Here is the call graph for this function:



The documentation for this struct was generated from the following file:

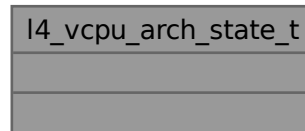
- [l4/sys/vcon.h](#)

## 16.257 l4\_vcpu\_arch\_state\_t Struct Reference

Architecture-specific vCPU state.

```
#include <__vcpu-arch.h>
```

Collaboration diagram for l4\_vcpu\_arch\_state\_t:



### 16.257.1 Detailed Description

Architecture-specific vCPU state.

Definition at line 74 of file [\\_\\_vcpu-arch.h](#).

The documentation for this struct was generated from the following files:

- [arm/l4/sys/\\_\\_vcpu-arch.h](#)
- [arm64/l4/sys/\\_\\_vcpu-arch.h](#)
- [amd64/l4/sys/\\_\\_vcpu-arch.h](#)

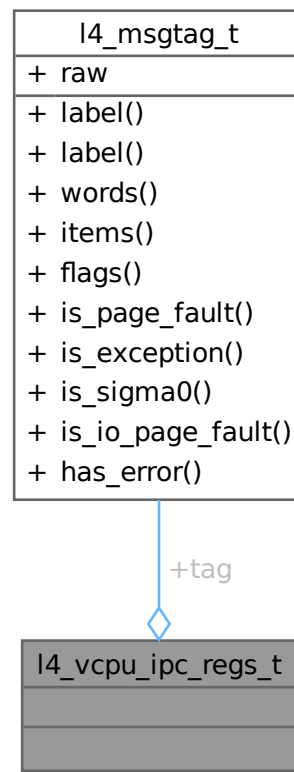
## 16.258 l4\_vcpu\_ipc\_regs\_t Struct Reference

vCPU message registers.

```
#include <__vcpu-arch.h>
```



Collaboration diagram for l4\_vcpu\_ipc\_regs\_t:



### 16.258.1 Detailed Description

vCPU message registers.

Definition at line 83 of file [\\_\\_vcpu-arch.h](#).

The documentation for this struct was generated from the following files:

- [arm/l4/sys/\\_\\_vcpu-arch.h](#)
- [arm64/l4/sys/\\_\\_vcpu-arch.h](#)
- [amd64/l4/sys/\\_\\_vcpu-arch.h](#)
- [x86/l4/sys/\\_\\_vcpu-arch.h](#)

## 16.259 l4\_vcpu\_regs\_t Struct Reference

vCPU registers.

```
#include <__vcpu-arch.h>
```

Collaboration diagram for `l4_vcpu_regs_t`:



## Data Fields

- [l4\\_umword\\_t](#) **pfa**  
*page fault address*
- [l4\\_umword\\_t](#) **err**  
*error code*
- [l4\\_umword\\_t](#) **sp**  
*stack pointer*
- [l4\\_umword\\_t](#) **ip**  
*instruction pointer*
- [l4\\_umword\\_t](#) **flags**  
*eflags*
- [l4\\_umword\\_t](#) **tpidrur0**  
*Thread-ID register.*
- [l4\\_umword\\_t](#) **tpidrurw**  
*Thread-ID register.*
- [l4\\_umword\\_t](#) **r15**  
*r15 register*
- [l4\\_umword\\_t](#) **r14**  
*r14 register*
- [l4\\_umword\\_t](#) **r13**  
*r13 register*
- [l4\\_umword\\_t](#) **r12**  
*r12 register*
- [l4\\_umword\\_t](#) **r11**  
*r11 register*

- [l4\\_umword\\_t r10](#)  
*r10 register*
- [l4\\_umword\\_t r9](#)  
*r9 register*
- [l4\\_umword\\_t r8](#)  
*r8 register*
- [l4\\_umword\\_t di](#)  
*rdi register*
- [l4\\_umword\\_t si](#)  
*rsi register*
- [l4\\_umword\\_t bp](#)  
*rbp register*
- [l4\\_umword\\_t bx](#)  
*rbx register*
- [l4\\_umword\\_t dx](#)  
*rdx register*
- [l4\\_umword\\_t cx](#)  
*rcx register*
- [l4\\_umword\\_t ax](#)  
*rax register*
- [l4\\_umword\\_t trapno](#)  
*trap number*
- [l4\\_umword\\_t cs](#)  
*dummy*
- [l4\\_umword\\_t ss](#)  
*ss register*
- [l4\\_umword\\_t es](#)  
*es register*
- [l4\\_umword\\_t ds](#)  
*ds register*
- [l4\\_umword\\_t gs](#)  
*gs register*
- [l4\\_umword\\_t fs](#)  
*fs register*
- [l4\\_umword\\_t dummy1](#)  
*dummy*

### 16.259.1 Detailed Description

vCPU registers.

Definition at line 55 of file [\\_\\_vcpu-arch.h](#).

### 16.259.2 Field Documentation

#### 16.259.2.1 ax

[l4\\_umword\\_t](#) `l4_vcpu_regs_t::ax`

rax register

eax register

Definition at line 77 of file [\\_\\_vcpu-arch.h](#).

### 16.259.2.2 bp

`l4_umword_t l4_vcpu_regs_t::bp`

rbp register

ebp register

Definition at line 72 of file [\\_\\_vcpu-arch.h](#).

### 16.259.2.3 bx

`l4_umword_t l4_vcpu_regs_t::bx`

rbx register

ebx register

Definition at line 74 of file [\\_\\_vcpu-arch.h](#).

### 16.259.2.4 cx

`l4_umword_t l4_vcpu_regs_t::cx`

rcx register

ecx register

Definition at line 76 of file [\\_\\_vcpu-arch.h](#).

### 16.259.2.5 di

`l4_umword_t l4_vcpu_regs_t::di`

rdi register

edi register

Definition at line 70 of file [\\_\\_vcpu-arch.h](#).

### 16.259.2.6 dx

`l4_umword_t l4_vcpu_regs_t::dx`

rdx register

edx register

Definition at line 75 of file [\\_\\_vcpu-arch.h](#).

### 16.259.2.7 si

`l4_umword_t l4_vcpu_regs_t::si`

rsi register

esi register

Definition at line 71 of file `__vcpu-arch.h`.

The documentation for this struct was generated from the following files:

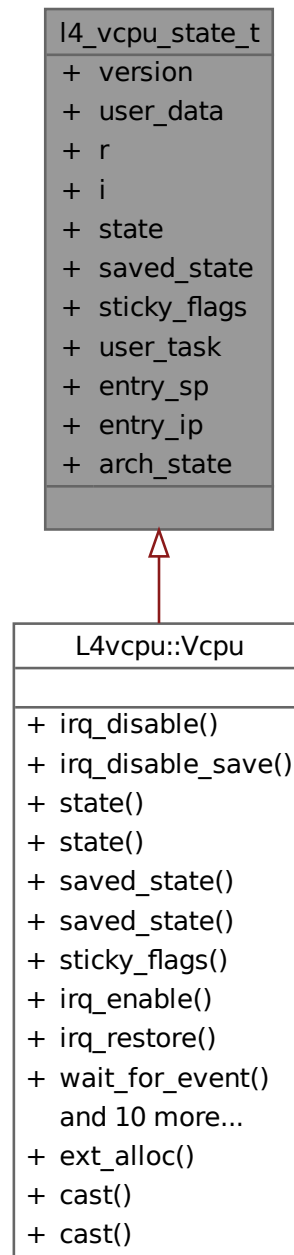
- `arm/l4/sys/__vcpu-arch.h`
- `amd64/l4/sys/__vcpu-arch.h`
- `x86/l4/sys/__vcpu-arch.h`

## 16.260 l4\_vcpu\_state\_t Struct Reference

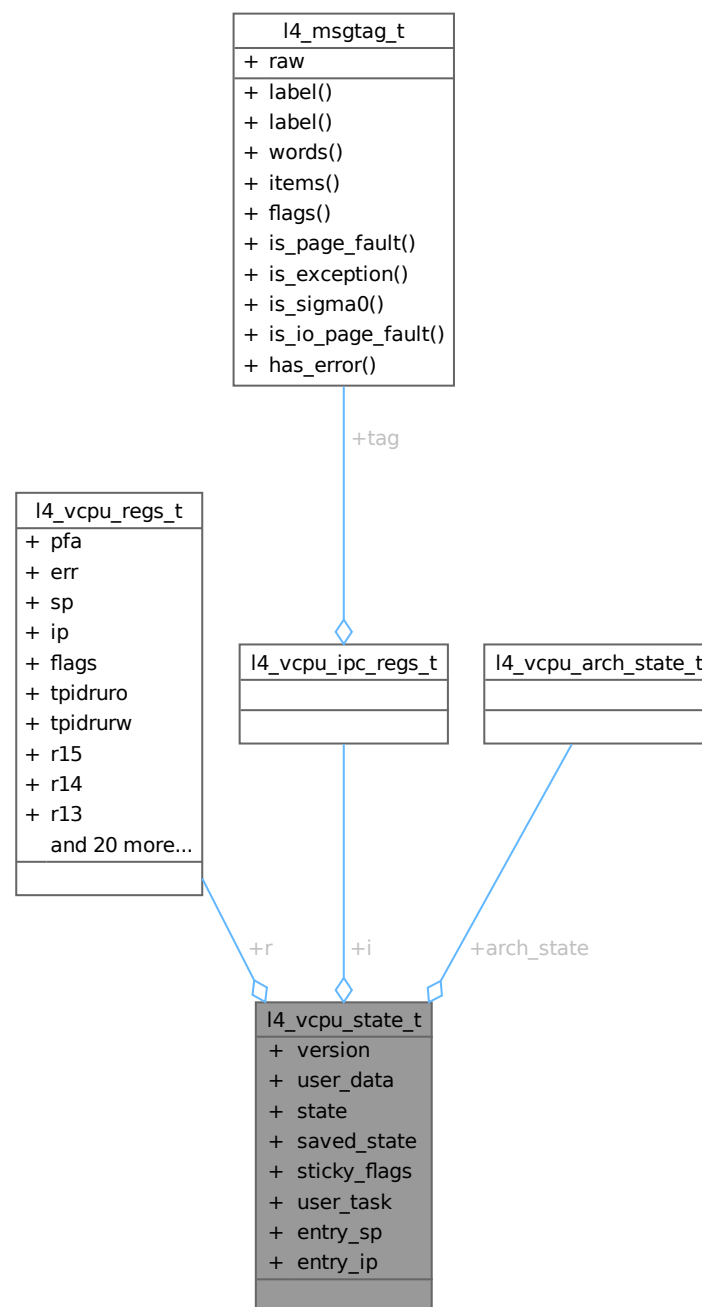
State of a vCPU.

```
#include <vcpu.h>
```

Inheritance diagram for `l4_vcpu_state_t`:



Collaboration diagram for l4\_vcpu\_state\_t:



## Data Fields

- [l4\\_umword\\_t version](#)  
*vCPU ABI version.*
- [l4\\_umword\\_t user\\_data](#) [7]  
*User-specific data.*
- [l4\\_vcpu\\_regs\\_t r](#)

- Register state.*
  - [l4\\_vcpu\\_ipc\\_regs\\_t](#) **i**
- IPC state.*
  - [l4\\_uint16\\_t](#) **state**
- Current vCPU state. See [L4\\_vcpu\\_state\\_flags](#).*
  - [l4\\_uint16\\_t](#) **saved\_state**
- Saved vCPU state. See [L4\\_vcpu\\_state\\_flags](#).*
  - [l4\\_uint16\\_t](#) **sticky\_flags**
- Pending flags. See [L4\\_vcpu\\_sticky\\_flags](#).*
  - [l4\\_cap\\_idx\\_t](#) **user\_task**
- User task to use.*
  - [l4\\_umword\\_t](#) **entry\_sp**
- Stack pointer for entry (when coming from user task).*
  - [l4\\_umword\\_t](#) **entry\_ip**
- IP for entry.*
  - [l4\\_vcpu\\_arch\\_state\\_t](#) **arch\_state**
- Architecture-specific state.*

### 16.260.1 Detailed Description

State of a vCPU.

Definition at line 75 of file [vcpu.h](#).

### 16.260.2 Field Documentation

#### 16.260.2.1 version

```
l4\_umword\_t l4_vcpu_state_t::version
```

vCPU ABI version.

Set by the kernel and must be checked by the user for equality with [L4\\_VCPU\\_STATE\\_VERSION](#).

Definition at line 77 of file [vcpu.h](#).

The documentation for this struct was generated from the following file:

- [l4/sys/vcpu.h](#)

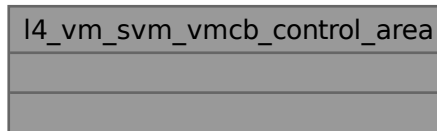


## 16.261 l4\_vm\_svm\_vmcb\_control\_area Struct Reference

VMCB structure for SVM VMs.

```
#include <__vm-svm.h>
```

Collaboration diagram for l4\_vm\_svm\_vmcb\_control\_area:



### 16.261.1 Detailed Description

VMCB structure for SVM VMs.

Definition at line 28 of file [\\_\\_vm-svm.h](#).

The documentation for this struct was generated from the following file:

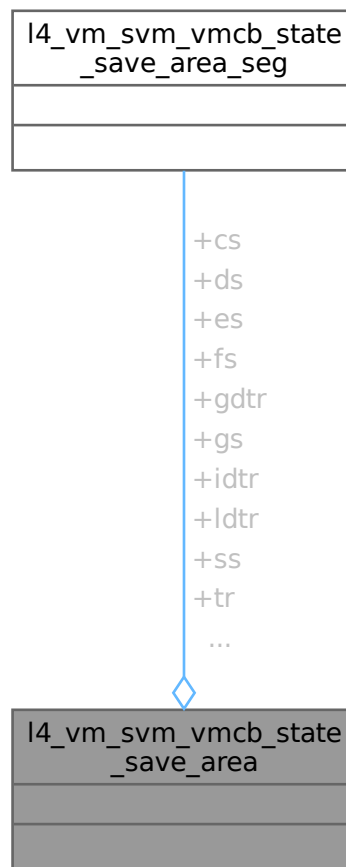
- l4/sys/\_\_vm-svm.h

## 16.262 l4\_vm\_svm\_vmcb\_state\_save\_area Struct Reference

State save area structure for SVM VMs.

```
#include <__vm-svm.h>
```

Collaboration diagram for `l4_vm_svm_vmcb_state_save_area`:



### 16.262.1 Detailed Description

State save area structure for SVM VMs.

Definition at line 85 of file [\\_\\_vm-svm.h](#).

The documentation for this struct was generated from the following file:

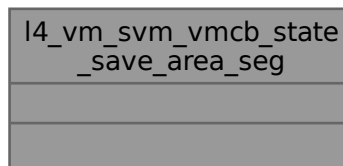
- `l4/sys/__vm-svm.h`

## 16.263 `l4_vm_svm_vmcb_state_save_area_seg` Struct Reference

State save area segment selector struct.

```
#include <__vm-svm.h>
```

Collaboration diagram for l4\_vm\_svm\_vmcb\_state\_save\_area\_seg:



### 16.263.1 Detailed Description

State save area segment selector struct.

Definition at line 73 of file [\\_\\_vm-svm.h](#).

The documentation for this struct was generated from the following file:

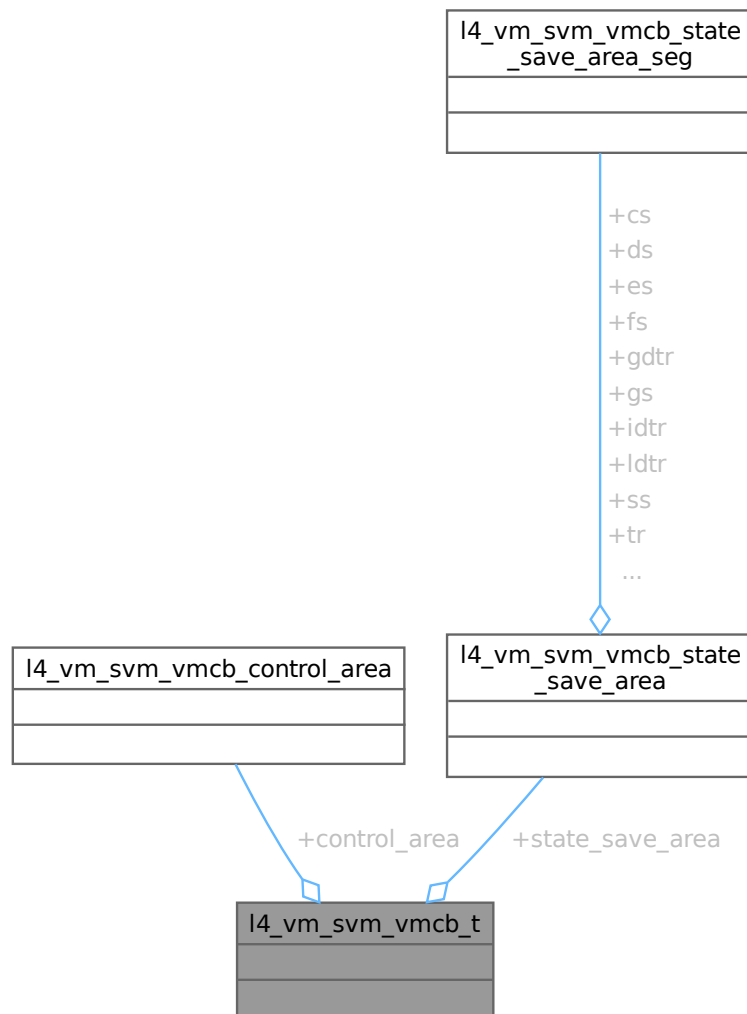
- l4/sys/\_\_vm-svm.h

## 16.264 l4\_vm\_svm\_vmcb\_t Struct Reference

Control structure for SVM VMs.

```
#include <__vm-svm.h>
```

Collaboration diagram for `l4_vm_svm_vmcb_t`:



### 16.264.1 Detailed Description

Control structure for SVM VMs.

Definition at line 154 of file [\\_\\_vm-svm.h](#).

The documentation for this struct was generated from the following file:

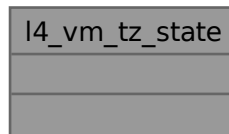
- `l4/sys/__vm-svm.h`

## 16.265 l4\_vm\_tz\_state Struct Reference

state structure for TrustZone VMs

```
#include <vm.h>
```

Collaboration diagram for l4\_vm\_tz\_state:



### 16.265.1 Detailed Description

state structure for TrustZone VMs

Definition at line 41 of file [vm.h](#).

The documentation for this struct was generated from the following file:

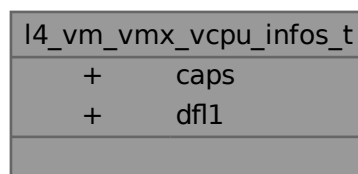
- [arm/l4/sys/vm.h](#)

## 16.266 l4\_vm\_vmx\_vcpu\_infos\_t Struct Reference

VMX information members.

```
#include <__vm-vmx.h>
```

Collaboration diagram for l4\_vm\_vmx\_vcpu\_infos\_t:



## Data Fields

- [l4\\_uint64\\_t caps](#) [[L4\\_VM\\_VMX\\_NUM\\_CAPS\\_REGS](#)]  
*Exported VMX capability registers. See [L4\\_vm\\_vmx\\_caps\\_regs](#).*
- [l4\\_uint32\\_t dfl1](#) [[L4\\_VM\\_VMX\\_NUM\\_DFL1\\_REGS](#)]  
*Exported VMX capability registers (default to 1 bits).*

### 16.266.1 Detailed Description

VMX information members.

Definition at line [239](#) of file [\\_\\_vm-vmx.h](#).

### 16.266.2 Field Documentation

#### 16.266.2.1 dfl1

```
l4\_uint32\_t l4_vm_vmx_vcpu_infos_t::dfl1 [L4\_VM\_VMX\_NUM\_DFL1\_REGS]
```

Exported VMX capability registers (default to 1 bits).

See [L4\\_vm\\_vmx\\_dfl1\\_regs](#).

Definition at line [246](#) of file [\\_\\_vm-vmx.h](#).

The documentation for this struct was generated from the following file:

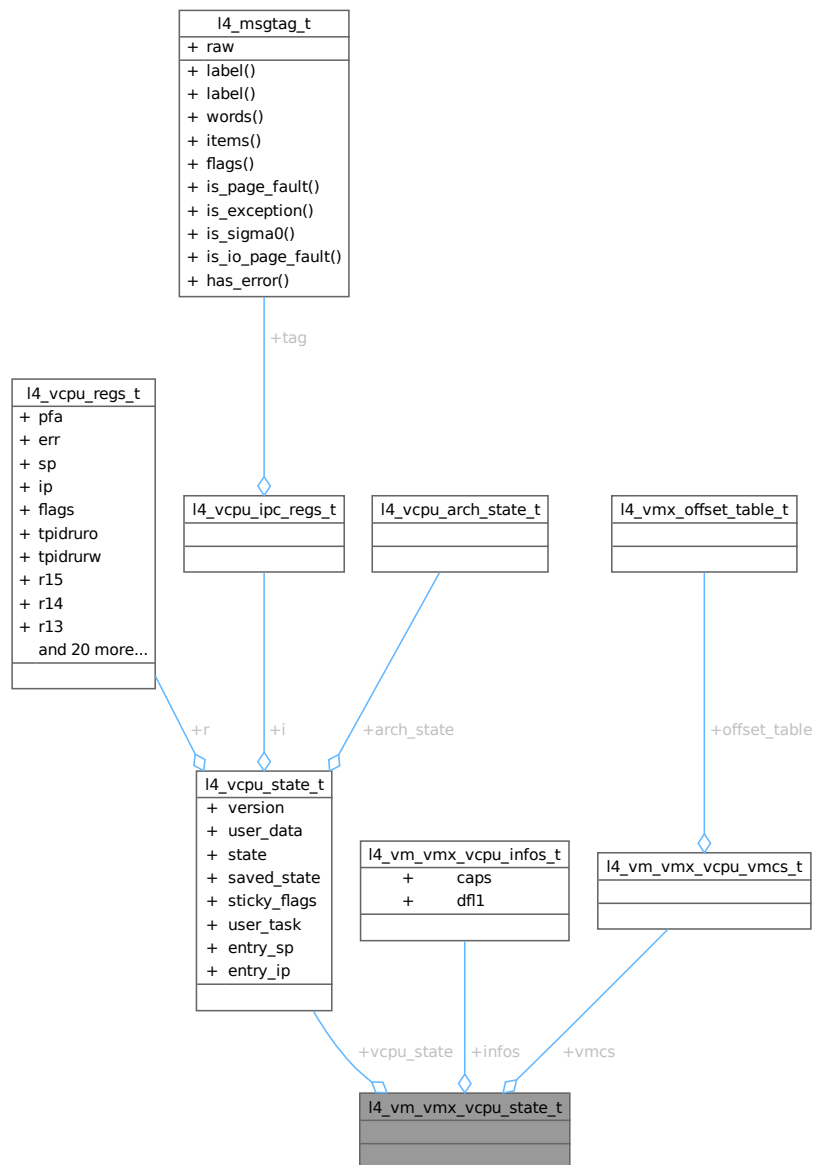
- [l4/sys/\\_\\_vm-vmx.h](#)

## 16.267 l4\_vm\_vmx\_vcpu\_state\_t Struct Reference

VMX vCPU state.

```
#include <\_\_vm-vmx.h>
```

Collaboration diagram for l4\_vm\_vmx\_vcpu\_state\_t:



### 16.267.1 Detailed Description

VMX vCPU state.

This is a specialization of the generic vCPU state for VMX. This data structure represents the following memory layout:

- 0x000 - 0x1ff: Standard vCPU state (with padding). See [l4\\_vcpu\\_state\\_t](#).
- 0x200 - 0x3ff: VMX information members (with padding). See [l4\\_vm\\_vmx\\_vcpu\\_infos\\_t](#).
- 0x400 - 0xffff: VMX software VMCS. See [l4\\_vm\\_vmx\\_vcpu\\_vmcs\\_t](#).

**Note**

The memory layout is documented here for reference purposes. However, the users are strongly discouraged from accessing the data structure directly. The API functions defined in this file are the preferred way of achieving the functionality.

Definition at line 267 of file [\\_\\_vm-vmx.h](#).

The documentation for this struct was generated from the following file:

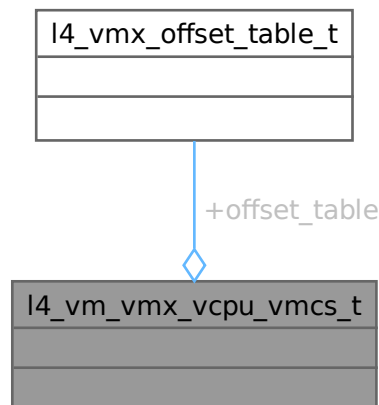
- l4/sys/\_\_vm-vmx.h

## 16.268 l4\_vm\_vmx\_vcpu\_vmcs\_t Struct Reference

VMX software VMCS.

```
#include <__vm-vmx.h>
```

Collaboration diagram for l4\_vm\_vmx\_vcpu\_vmcs\_t:



### 16.268.1 Detailed Description

VMX software VMCS.

This data structure represents the following memory layout:

- 0x000 - 0x007: Reserved (ignored by the kernel). In the hardware VMCS, the revision identifier and the abort indicator are stored in this area. Hereby we simply ignore these two entries.
- 0x008 - 0x00f: User space data (ignored by the kernel). This currently stores the pointer to a different software VMCS whose content has been loaded to this software VMCS.



- 0x010 - 0x013: VMCS field index of the software-defined CR2 field in the software VMCS.
- 0x014 - 0x017: Reserved.
- 0x018 - 0x01f: Capability of the vCPU context, i.e. the hardware VMCS object (with padding).
- 0x020 - 0x047: Software VMCS field offset table. See [l4\\_vmx\\_offset\\_table\\_t](#).
- 0x048 - 0x0bf: Reserved.
- 0x0c0 - 0xabf: Software VMCS fields (with padding).
- 0xac0 - 0xbff: Software VMCS fields dirty bitmap (with padding).

#### Note

The memory layout is documented here for reference purposes. However, the users are strongly discouraged from accessing the data structure directly. The API functions defined in this file are the preferred way of achieving the functionality.

Definition at line 205 of file [\\_\\_vm-vmx.h](#).

The documentation for this struct was generated from the following file:

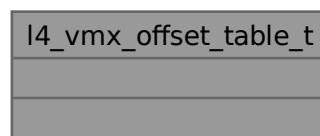
- `l4/sys/__vm-vmx.h`

## 16.269 `l4_vmx_offset_table_t` Struct Reference

Software VMCS field offset table.

```
#include <__vm-vmx.h>
```

Collaboration diagram for `l4_vmx_offset_table_t`:



### 16.269.1 Detailed Description

Software VMCS field offset table.

This data structure represents the following memory layout:

- 0x00 - 0x02: 3 offsets for 16-bit fields.
- 0x03: Reserved.
- 0x04 - 0x06: 3 offsets for 64-bit fields.
- 0x07: Reserved.
- 0x08 - 0x0a: 3 offsets for 32-bit fields.
- 0x0b: Reserved.
- 0x0c - 0x0e: 3 offsets for natural-width fields.
- 0x0f: Reserved.
- 0x10 - 0x12: 3 limits for 16-bit fields.
- 0x13: Reserved.
- 0x14 - 0x16: 3 limits for 64-bit fields.
- 0x17: Reserved.
- 0x18 - 0x1a: 3 limits for 32-bit fields.
- 0x1b: Reserved.
- 0x1c - 0x1e: 3 limits for natural-width fields.
- 0x1f: Reserved.
- 0x20 - 0x23: 4 index shifts.
- 0x24: Offset of the first software VMCS field.
- 0x25: Size of the software VMCS fields.
- 0x26 - 0x27: Reserved.

The offsets/limits in each size category are in the following order:

- Control fields.
- Read-only fields.
- Guest fields.

The index shifts are in the following order:

- 16-bit.
- 64-bit.
- 32-bit.
- Natural-width.

All offsets/limits/sizes are represented in a 64-byte granule.

The offsets (after being multiplied by 64) are indexes in the values array in [l4\\_vm\\_vmx\\_vcpu\\_vmcs\\_t](#) and bit indexes in the dirty\_bitmap array in [l4\\_vm\\_vmx\\_vcpu\\_vmcs\\_t](#).

The limits (after being multiplied by 64) represent the range of the available indexes.

**Note**

The memory layout is documented here for reference purposes. However, the users are strongly discouraged from accessing the data structure directly. The API functions defined in this file are the preferred way of achieving the functionality.

Definition at line 155 of file [\\_\\_vm-vmx.h](#).

The documentation for this struct was generated from the following file:

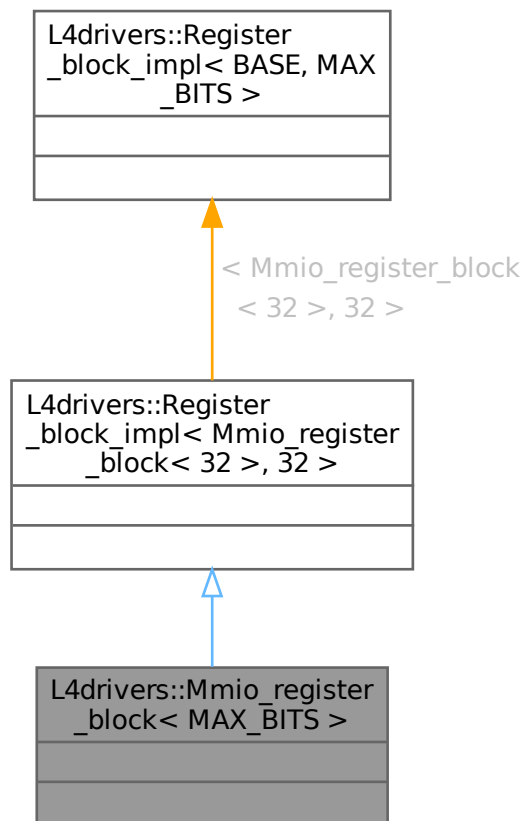
- l4/sys/\_\_vm-vmx.h

## 16.270 L4drivers::Mmio\_register\_block< MAX\_BITS > Struct Template Reference

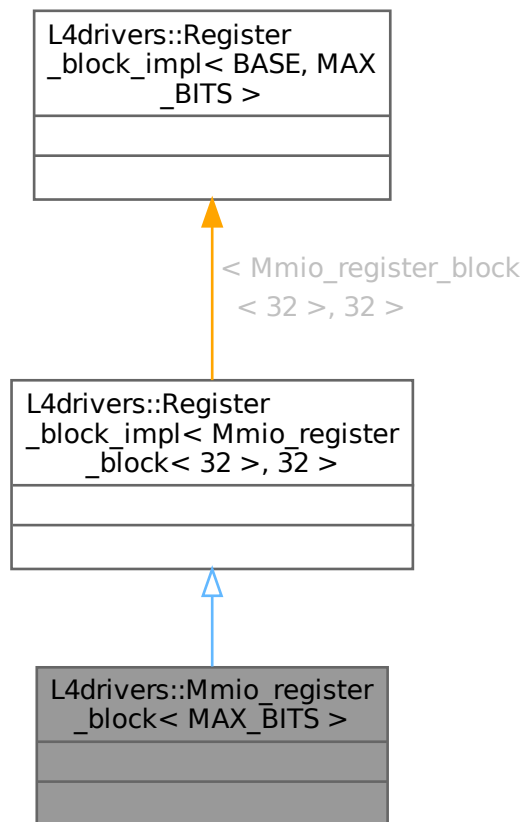
An MMIO block with up to 64-bit register access (32-bit default) and little endian byte order.

```
#include <hw_mmio_register_block>
```

Inheritance diagram for L4drivers::Mmio\_register\_block< MAX\_BITS >:



Collaboration diagram for L4drivers::Mmio\_register\_block< MAX\_BITS >:



### 16.270.1 Detailed Description

```
template<unsigned MAX_BITS = 32>
struct L4drivers::Mmio_register_block< MAX_BITS >
```

An MMIO block with up to 64-bit register access (32-bit default) and little endian byte order.

Definition at line 43 of file [hw\\_mmio\\_register\\_block](#).

The documentation for this struct was generated from the following file:

- `pkg/drivers-frst/include/hw_mmio_register_block`

## 16.271 L4drivers::Register\_block< MAX\_BITS, BLOCK > Class Template Reference

Handles a reference to a register block of the given maximum access width.

```
#include <hw_register_block>
```

Collaboration diagram for L4drivers::Register\_block< MAX\_BITS, BLOCK >:

L4drivers::Register_block< MAX_BITS, BLOCK >	
+	r()
+	operator[]()
+	r()
+	operator[]()

### Public Member Functions

- `template<unsigned BITS>`  
`Ro_register_tmpl< BITS, Block > r (unsigned offset) const`  
*Read only access to register at offset offset.*
- `Ro_register_operator[] (unsigned offset) const`  
*Read only access to register at offset offset.*
- `template<unsigned BITS>`  
`Register_tmpl< BITS, Block > r (unsigned offset)`  
*Read/write access to register at offset offset.*
- `Register_operator[] (unsigned offset)`  
*Read/write access to register at offset offset.*

### 16.271.1 Detailed Description

```
template<unsigned MAX_BITS, typename BLOCK = Register_block_tmpl< Register_block_base<MAX_BITS> >>
class L4drivers::Register_block< MAX_BITS, BLOCK >
```

Handles a reference to a register block of the given maximum access width.

Register block.

### Template Parameters

<b>MAX_BITS</b>	Maximum access width for the registers in this block.
<b>BLOCK</b>	Type implementing the register accesses ( <code>read&lt;&gt;()</code> , <code>write&lt;&gt;()</code> , <code>modify&lt;&gt;()</code> , <code>set&lt;&gt;()</code> , and <code>clear&lt;&gt;()</code> ).

Provides access to registers in this block via `r<WIDTH>()` and `operator[]()`.

Example usage:

```
void test()
{
    // create a register block reference for max. 16bit accesses, using a
    // MMIO register block implementation (at address 0x1000).
    Hw::Register_block<16> regs = new Hw::Mmio_register_block<16>(0x1000);

    // Alternatively it is allowed to use an implementation that allows
    // wider access than actually needed.
    Hw::Register_block<16> regs = new Hw::Mmio_register_block<32>(0x1000);

    // read a 16bit register at offset 8byte
    unsigned short x = regs.r<16>(8);
    unsigned short x1 = regs[8]; // alternative

    // read an 8bit register at offset 0byte
    unsigned v = regs.r<8>(0);

    // do a 16bit write to register at offset 2byte (four variants)
    regs[2] = 22;
    regs.r<16>(2) = 22;
    regs[2].write(22);
    regs.r<16>().write(22);

    // do an 8bit write (two variants)
    regs.r<8>(0) = 9;
    regs.r<8>(0).write(9);

    // do 16bit read-modify-write (two variants)
    regs[4].modify(0xf, 3); // clear 4 lowest bits and set them to 3
    regs.r<16>(4).modify(0xf, 3);

    // do 8bit read-modify-write
    regs.r<8>(0).modify(0xf, 3);

    // fails to compile, because of too wide access
    // (32 bit access but regs is Hw::Register_block<16>)
    unsigned long v = regs.r<32>(4)
}
```

Definition at line 330 of file `hw_register_block`.

## 16.271.2 Member Function Documentation

### 16.271.2.1 `operator[]()` [1/2]

```
template<unsigned MAX_BITS, typename BLOCK = Register_block_tmpl< Register_block_base<MAX_↵
BITS> >>
Register L4drivers::Register_block< MAX_BITS, BLOCK >::operator[] (
    unsigned offset) [inline]
```

Read/write access to register at offset *offset*.

#### Parameters

<i>offset</i>	The offset of the register within the register file.
---------------	------------------------------------------------------

**Returns**

register object allowing read and write access with width *MAX\_BITS*.

Definition at line 385 of file [hw\\_register\\_block](#).

References [r\(\)](#).

Here is the call graph for this function:

**16.271.2.2 operator[]() [2/2]**

```

template<unsigned MAX_BITS, typename BLOCK = Register_block_tmpl< Register_block_base<MAX_BITS> >>
Ro_register L4drivers::Register_block< MAX_BITS, BLOCK >::operator[] (
    unsigned offset) const [inline]
  
```

Read only access to register at offset *offset*.

**Parameters**

<i>offset</i>	The offset of the register within the register file.
---------------	------------------------------------------------------

**Returns**

register object allowing read only access with width *MAX\_BITS*.

Definition at line 365 of file [hw\\_register\\_block](#).

References [r\(\)](#).

Here is the call graph for this function:



**16.271.2.3 r()** [1/2]

```
template<unsigned MAX_BITS, typename BLOCK = Register_block_tmpl< Register_block_base<MAX_BITS> >>
template<unsigned BITS>
Register_tmpl< BITS, Block > L4drivers::Register_block< MAX_BITS, BLOCK >::r (
    unsigned offset) [inline]
```

Read/write access to register at offset *offset*.

**Template Parameters**

<i>BITS</i>	the access width in bits for the register.
-------------	--------------------------------------------

**Parameters**

<i>offset</i>	The offset of the register within the register file.
---------------	------------------------------------------------------

**Returns**

register object allowing read and write access with width *BITS*.

Definition at line 376 of file [hw\\_register\\_block](#).

**16.271.2.4 r()** [2/2]

```
template<unsigned MAX_BITS, typename BLOCK = Register_block_tmpl< Register_block_base<MAX_BITS> >>
template<unsigned BITS>
Ro_register_tmpl< BITS, Block > L4drivers::Register_block< MAX_BITS, BLOCK >::r (
    unsigned offset) const [inline]
```

Read only access to register at offset *offset*.

**Template Parameters**

<i>BITS</i>	the access width in bits for the register.
-------------	--------------------------------------------

**Parameters**

<i>offset</i>	The offset of the register within the register file.
---------------	------------------------------------------------------



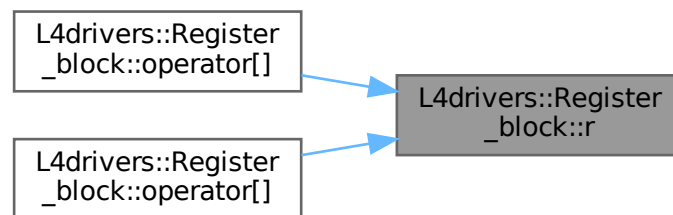
**Returns**

register object allowing read only access with width *BITS*.

Definition at line 357 of file [hw\\_register\\_block](#).

Referenced by [operator\[\]\(\)](#), and [operator\[\]\(\)](#).

Here is the caller graph for this function:



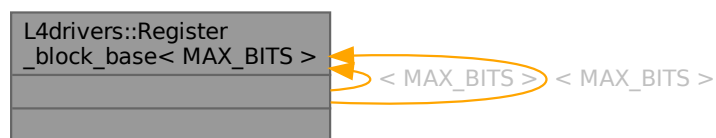
The documentation for this class was generated from the following file:

- `pkg/drivers-frst/include/hw_register_block`

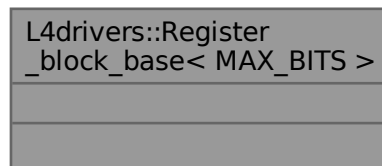
## 16.272 L4drivers::Register\_block\_base< MAX\_BITS > Struct Template Reference

Abstract register block interface.

Inheritance diagram for `L4drivers::Register_block_base< MAX_BITS >`:



Collaboration diagram for L4drivers::Register\_block\_base< MAX\_BITS >:



### 16.272.1 Detailed Description

```
template<unsigned MAX_BITS = 32>
struct L4drivers::Register_block_base< MAX_BITS >
```

Abstract register block interface.

#### Template Parameters

<i>MAX_BITS</i>	The maximum access width for the registers.
-----------------	---------------------------------------------

This interfaces is based on virtual `do_read_<xx>` and `do_write_<xx>` methods that have to be implemented up to the maximum access width.

Definition at line 72 of file [hw\\_register\\_block](#).

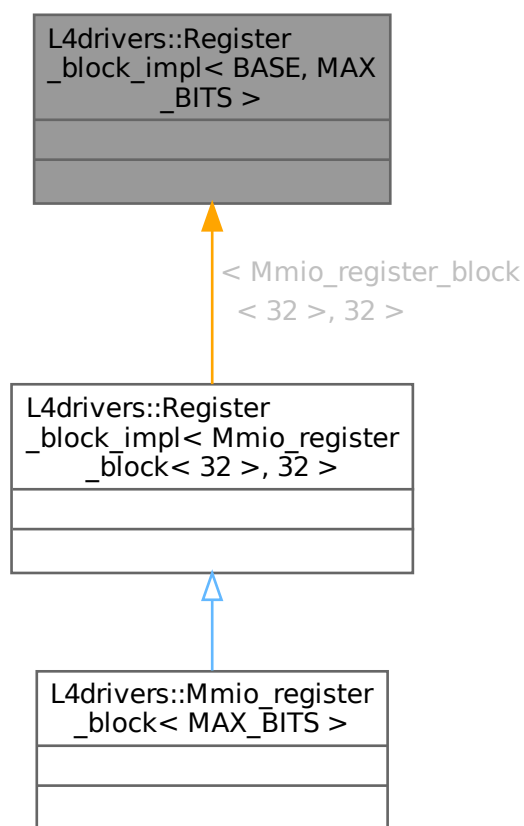
The documentation for this struct was generated from the following file:

- `pkg/drivers-frst/include/hw_register_block`

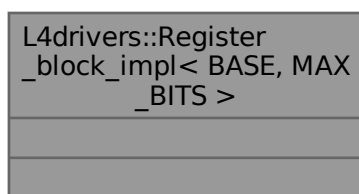
## 16.273 L4drivers::Register\_block\_impl< BASE, MAX\_BITS > Struct Template Reference

Implementation helper for register blocks.

Inheritance diagram for L4drivers::Register\_block\_impl< BASE, MAX\_BITS >:



Collaboration diagram for L4drivers::Register\_block\_impl< BASE, MAX\_BITS >:



### 16.273.1 Detailed Description

```
template<typename BASE, unsigned MAX_BITS = 32>
struct L4drivers::Register_block_impl< BASE, MAX_BITS >
```

Implementation helper for register blocks.

#### Parameters

<i>BASE</i>	The class implementing read<> and write<> template functions for accessing the registers. This class must inherit from <a href="#">Register_block_impl</a> .
<i>MAX_BITS</i>	The maximum access width for the register file. Supported values are 8, 16, 32, or 64.

This template allows easy implementation of register files by providing read<> and write<> template functions, see [Mmio\\_register\\_block](#) as an example.

Definition at line 455 of file [hw\\_register\\_block](#).

The documentation for this struct was generated from the following file:

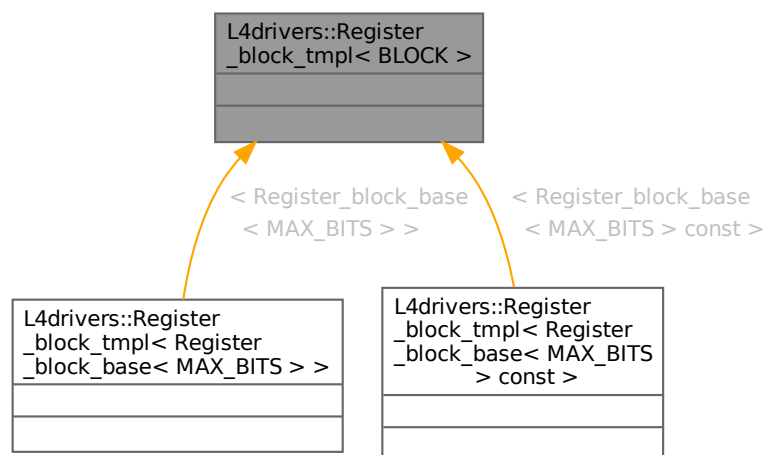
- [pkg/drivers-frst/include/hw\\_register\\_block](#)

## 16.274 L4drivers::Register\_block\_tmpl< BLOCK > Class Template Reference

Helper template that translates to the [Register\\_block\\_base](#) interface.

```
#include <hw_register_block>
```

Inheritance diagram for L4drivers::Register\_block\_tmpl< BLOCK >:



Collaboration diagram for L4drivers::Register\_block\_tmpl< BLOCK >:



### 16.274.1 Detailed Description

```
template<typename BLOCK>
class L4drivers::Register_block_tmpl< BLOCK >
```

Helper template that translates to the [Register\\_block\\_base](#) interface.

#### Template Parameters

<i>BLOCK</i>	The type of the <a href="#">Register_block_base</a> interface to use.
--------------	-----------------------------------------------------------------------

This helper translates `read<T>()`, `write<T>()`, `set<T>()`, `clear<T>()`, and `modify<T>()` calls to `BLOCK::do_read_<xx>` and `BLOCK::do_write_<xx>`.

Definition at line 156 of file [hw\\_register\\_block](#).

The documentation for this class was generated from the following file:

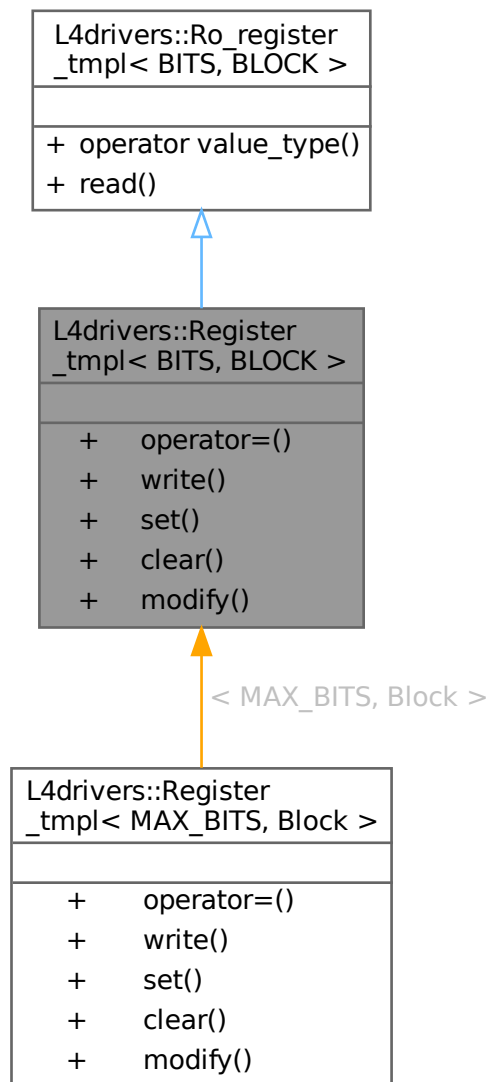
- `pkg/drivers-frst/include/hw_register_block`

## 16.275 L4drivers::Register\_tmpl< BITS, BLOCK > Class Template Reference

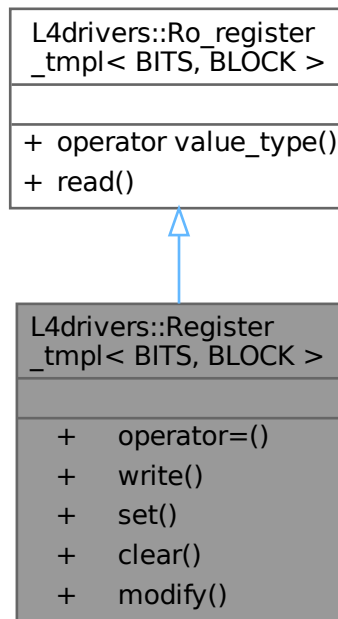
Single hardware register inside a [Register\\_block\\_base](#) interface.

```
#include <hw_register_block>
```

Inheritance diagram for L4drivers::Register\_tmpl< BITS, BLOCK >:



Collaboration diagram for L4drivers::Register\_tmpl< BITS, BLOCK >:



### Public Member Functions

- [Register\\_tmpl](#) & `operator=` (value\_type val)  
*write val into the hardware register.*
- void [write](#) (value\_type val)  
*write val into the hardware register.*
- value\_type [set](#) (value\_type set\_bits)  
*set bits in set\_bits in the hardware register.*
- value\_type [clear](#) (value\_type clear\_bits)  
*clears bits in clear\_bits in the hardware register.*
- value\_type [modify](#) (value\_type clear\_bits, value\_type set\_bits)  
*clears bits in clear\_bits and sets bits in set\_bits in the hardware register.*

### Public Member Functions inherited from [L4drivers::Ro\\_register\\_tmpl< BITS, BLOCK >](#)

- `operator value_type` () const  
*read the value from the hardware register.*
- value\_type [read](#) () const  
*read the value from the hardware register.*

### 16.275.1 Detailed Description

```
template<unsigned BITS, typename BLOCK>  
class L4drivers::Register_tmpl< BITS, BLOCK >
```

Single hardware register inside a [Register\\_block\\_base](#) interface.

#### Template Parameters

---



<i>BITS</i>	The access width for the register in bits.
<i>BLOCK</i>	the type of the <a href="#">Register_block_base</a> interface.

**Note**

Objects of this type must be used only in temporary contexts not in global, class, or object scope.

Definition at line 237 of file [hw\\_register\\_block](#).

**16.275.2 Member Function Documentation****16.275.2.1 clear()**

```
template<unsigned BITS, typename BLOCK>
value_type L4drivers::Register_tmpl< BITS, BLOCK >::clear (
    value_type clear_bits) [inline]
```

clears bits in *clear\_bits* in the hardware register.

**Parameters**

<i>clear_bits</i>	bits to be cleared within the hardware register.
-------------------	--------------------------------------------------

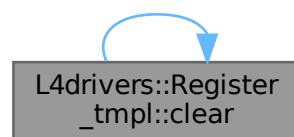
This is a read-modify-write function that does a logical and of the old value from the register with the negated value of *clear\_bits*.

```
unsigned old_value = read();
write(old_value & ~clear_bits);
```

Definition at line 290 of file [hw\\_register\\_block](#).

Referenced by [L4drivers::Register\\_tmpl< MAX\\_BITS, Block >::clear\(\)](#).

Here is the caller graph for this function:

**16.275.2.2 modify()**

```
template<unsigned BITS, typename BLOCK>
value_type L4drivers::Register_tmpl< BITS, BLOCK >::modify (
    value_type clear_bits,
    value_type set_bits) [inline]
```

clears bits in *clear\_bits* and sets bits in *set\_bits* in the hardware register.

**Parameters**

<i>clear_bits</i>	bits to be cleared within the hardware register.
<i>set_bits</i>	bits to set in the hardware register.

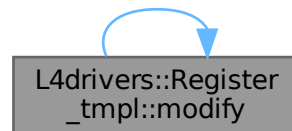
This is a read-modify-write function that first does a logical and of the old value from the register with the negated value of *clear\_bits* and then does a logical or with *set\_bits*.

```
unsigned old_value = read();
write((old_value & ~clear_bits) | set_bits);
```

Definition at line 308 of file [hw\\_register\\_block](#).

Referenced by [L4drivers::Register\\_tmpl< MAX\\_BITS, Block >::modify\(\)](#).

Here is the caller graph for this function:



### 16.275.2.3 operator=()

```
template<unsigned BITS, typename BLOCK>
Register_tmpl & L4drivers::Register_tmpl< BITS, BLOCK >::operator= (
    value_type val) [inline]
```

write *val* into the hardware register.

#### Parameters

<i>val</i>	the value to write into the hardware register.
------------	------------------------------------------------

Definition at line 252 of file [hw\\_register\\_block](#).

### 16.275.2.4 set()

```
template<unsigned BITS, typename BLOCK>
value_type L4drivers::Register_tmpl< BITS, BLOCK >::set (
    value_type set_bits) [inline]
```

set bits in *set\_bits* in the hardware register.

#### Parameters

<i>set_bits</i>	bits to be set within the hardware register.
-----------------	----------------------------------------------

This is a read-modify-write function that does a logical or of the old value from the register with *set\_bits*.

```
unsigned old_value = read();
write(old_value | set_bits);
```

Definition at line 274 of file [hw\\_register\\_block](#).

Referenced by [L4drivers::Register\\_tmpl< MAX\\_BITS, Block >::set\(\)](#).

Here is the caller graph for this function:



### 16.275.2.5 write()

```
template<unsigned BITS, typename BLOCK>
void L4drivers::Register_tmpl< BITS, BLOCK >::write (
    value_type val) [inline]
```

write *val* into the hardware register.

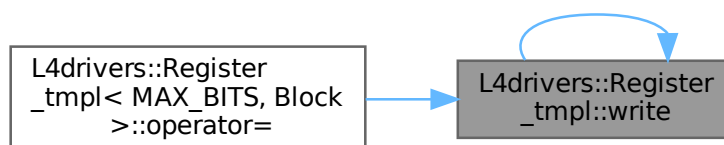
#### Parameters

<i>val</i>	the value to write into the hardware register.
------------	------------------------------------------------

Definition at line 259 of file [hw\\_register\\_block](#).

Referenced by [L4drivers::Register\\_tmpl< MAX\\_BITS, Block >::operator=\(\)](#), and [L4drivers::Register\\_tmpl< MAX\\_BITS, Block >::write\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

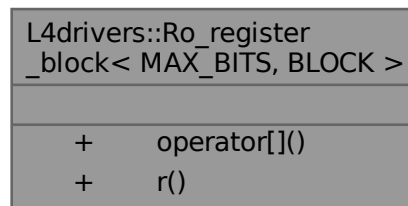
- [pkg/drivers-frst/include/hw\\_register\\_block](#)

## 16.276 L4drivers::Ro\_register\_block< MAX\_BITS, BLOCK > Class Template Reference

Handles a reference to a read only register block of the given maximum access width.

```
#include <hw_register_block>
```

Collaboration diagram for L4drivers::Ro\_register\_block< MAX\_BITS, BLOCK >:



### Public Member Functions

- [Ro\\_register operator\[\]](#) (unsigned offset) const  
*Read only access to register at offset offset.*
- [template<unsigned BITS>](#)  
[Ro\\_register\\_tmpl](#)< BITS, Block > [r](#) (unsigned offset) const  
*Read only access to register at offset offset.*

### 16.276.1 Detailed Description

```
template<unsigned MAX_BITS, typename BLOCK = Register_block_tmpl< Register_block_base<MAX_BITS>
BITS> const >>
class L4drivers::Ro_register_block< MAX_BITS, BLOCK >
```

Handles a reference to a read only register block of the given maximum access width.

#### Template Parameters

<i>MAX_BITS</i>	Maximum access width for the registers in this block.
<i>BLOCK</i>	Type implementing the register accesses (read<>()),

Provides read only access to registers in this block via [r<WIDTH>\(\)](#) and [operator\[\]\(\)](#).

Definition at line 404 of file [hw\\_register\\_block](#).

## 16.276.2 Member Function Documentation

### 16.276.2.1 operator[]()

```
template<unsigned MAX_BITS, typename BLOCK = Register_block_tmpl< Register_block_base<MAX_BITS> const >>
Ro_register L4drivers::Ro_register_block< MAX_BITS, BLOCK >::operator[] (
    unsigned offset) const [inline]
```

Read only access to register at offset *offset*.

#### Parameters

<i>offset</i>	The offset of the register within the register file.
---------------	------------------------------------------------------

#### Returns

register object allowing read only access with width *MAX\_BITS*.

Definition at line 426 of file [hw\\_register\\_block](#).

### 16.276.2.2 r()

```
template<unsigned MAX_BITS, typename BLOCK = Register_block_tmpl< Register_block_base<MAX_BITS> const >>
template<unsigned BITS>
Ro_register_tmpl< BITS, Block > L4drivers::Ro_register_block< MAX_BITS, BLOCK >::r (
    unsigned offset) const [inline]
```

Read only access to register at offset *offset*.

#### Template Parameters

<i>BITS</i>	the access width in bits for the register.
-------------	--------------------------------------------

#### Parameters

<i>offset</i>	The offset of the register within the register file.
---------------	------------------------------------------------------

#### Returns

register object allowing read only access with width *BITS*.

Definition at line 436 of file [hw\\_register\\_block](#).

The documentation for this class was generated from the following file:

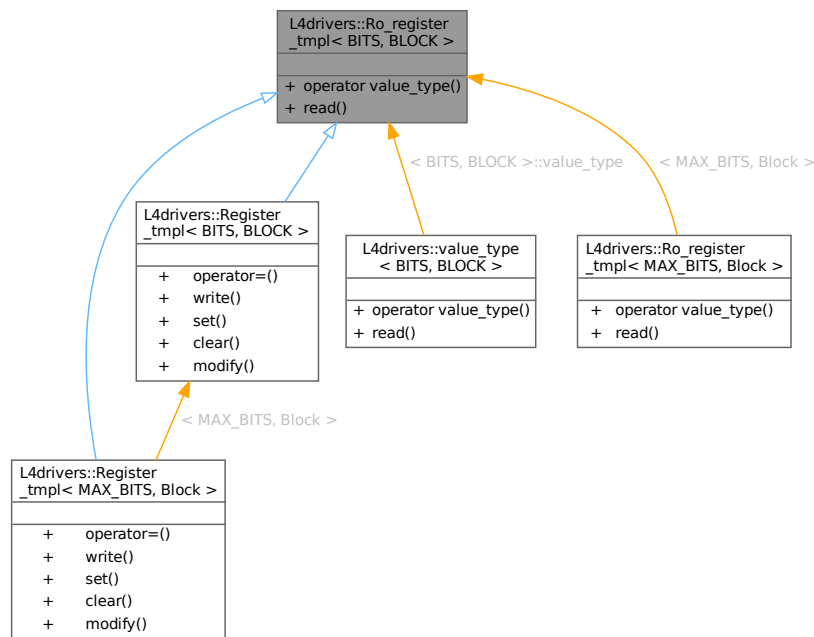
- pkg/drivers-frst/include/hw\_register\_block

## 16.277 L4drivers::Ro\_register\_tmpl< BITS, BLOCK > Class Template Reference

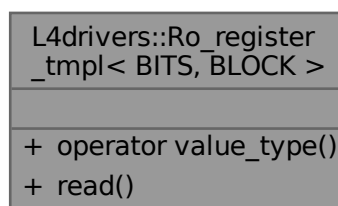
Single read only register inside a [Register\\_block\\_base](#) interface.

```
#include <hw_register_block>
```

Inheritance diagram for L4drivers::Ro\_register\_tmpl< BITS, BLOCK >:



Collaboration diagram for L4drivers::Ro\_register\_tmpl< BITS, BLOCK >:



### Public Member Functions

- `operator value_type()` const  
read the value from the hardware register.
- `value_type read()` const  
read the value from the hardware register.

## 16.277.1 Detailed Description

```
template<unsigned BITS, typename BLOCK>
class L4drivers::Ro_register_tmpl< BITS, BLOCK >
```

Single read only register inside a [Register\\_block\\_base](#) interface.

### Template Parameters

<i>BITS</i>	The access with of the register in bits.
<i>BLOCK</i>	The type for the <a href="#">Register_block_base</a> interface.

### Note

Objects of this type must be used only in temporary contexts not in global, class, or object scope.

Allows simple read only access to a hardware register.

Definition at line 201 of file [hw\\_register\\_block](#).

## 16.277.2 Member Function Documentation

### 16.277.2.1 operator value\_type()

```
template<unsigned BITS, typename BLOCK>
L4drivers::Ro_register_tmpl< BITS, BLOCK >::operator value_type () const [inline]
```

read the value from the hardware register.

### Returns

value read from the hardware register.

Definition at line 217 of file [hw\\_register\\_block](#).

### 16.277.2.2 read()

```
template<unsigned BITS, typename BLOCK>
value_type L4drivers::Ro_register_tmpl< BITS, BLOCK >::read () const [inline]
```

read the value from the hardware register.

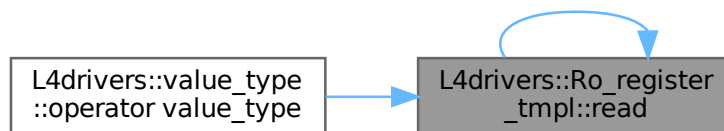
**Returns**

value from the hardware register.

Definition at line 224 of file [hw\\_register\\_block](#).

Referenced by [L4drivers::value\\_type< BITS, BLOCK >::operator value\\_type\(\)](#), and [L4drivers::value\\_type< BITS, BLOCK >::read\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

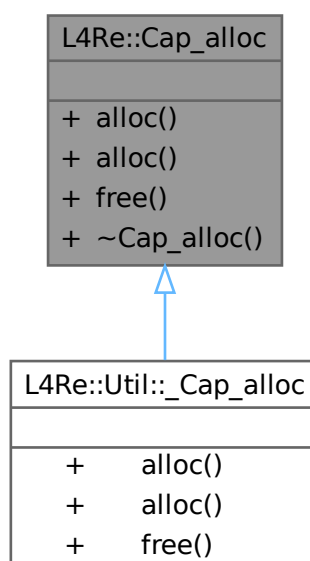
- `pkg/drivers-frst/include/hw_register_block`

## 16.278 L4Re::Cap\_alloc Class Reference

Capability allocator interface.

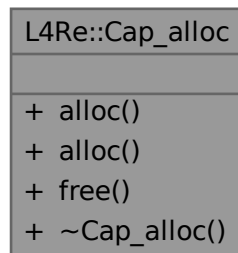
```
#include <cap_alloc>
```

Inheritance diagram for L4Re::Cap\_alloc:





Collaboration diagram for L4Re::Cap\_alloc:



## Public Member Functions

- virtual [L4::Cap](#)< void > [alloc](#) () noexcept=0  
*Allocate a capability.*
- template<typename T>  
[L4::Cap](#)< T > [alloc](#) () noexcept  
*Allocate a capability.*
- virtual void [free](#) ([L4::Cap](#)< void > cap, [l4\\_cap\\_idx\\_t](#) task=[L4\\_INVALID\\_CAP](#), unsigned unmap\_↔  
flags=[L4\\_FP\\_ALL\\_SPACES](#)) noexcept=0  
*Free a capability.*
- virtual [~Cap\\_alloc](#) ()=0  
*Destructor.*

## 16.278.1 Detailed Description

Capability allocator interface.

Definition at line 30 of file [cap\\_alloc](#).

## 16.278.2 Member Function Documentation

### 16.278.2.1 [alloc\(\)](#) [1/2]

```
template<typename T>
L4::Cap< T > L4Re::Cap_alloc::alloc () [inline], [noexcept]
```

Allocate a capability.

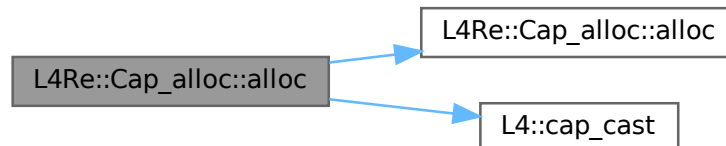
**Returns**

Capability of type T

Definition at line 53 of file [cap\\_alloc](#).

References [alloc\(\)](#), and [L4::cap\\_cast\(\)](#).

Here is the call graph for this function:

**16.278.2.2 alloc() [2/2]**

```
virtual L4::Cap< void > L4Re::Cap_alloc::alloc () [pure virtual], [noexcept]
```

Allocate a capability.

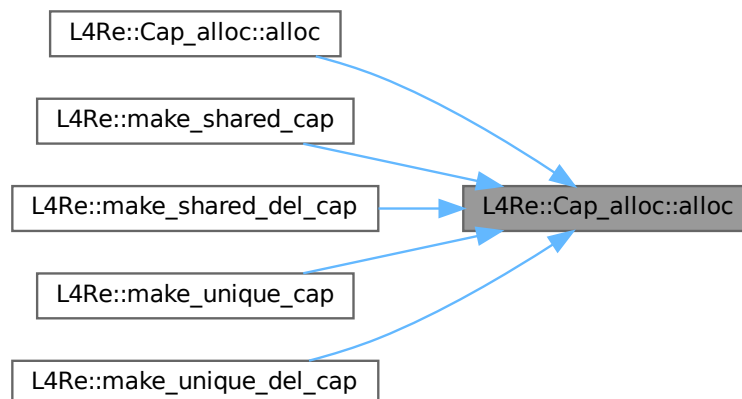
**Returns**

Capability of type void

Implemented in [L4Re::Util::\\_Cap\\_alloc](#), and [L4Re::Util::\\_Cap\\_alloc](#).

Referenced by [alloc\(\)](#), [L4Re::make\\_shared\\_cap\(\)](#), [L4Re::make\\_shared\\_del\\_cap\(\)](#), [L4Re::make\\_unique\\_cap\(\)](#), and [L4Re::make\\_unique\\_del\\_cap\(\)](#).

Here is the caller graph for this function:



### 16.278.2.3 free()

```
virtual void L4Re::Cap_alloc::free (  
    L4::Cap< void > cap,  
    l4\_cap\_idx\_t task = L4\_INVALID\_CAP,  
    unsigned unmap_flags = L4\_FP\_ALL\_SPACES) [pure virtual], [noexcept]
```

Free a capability.

#### Parameters

<i>cap</i>	Capability to free.
<i>task</i>	If set, task to unmap the capability from.
<i>unmap_flags</i>	Flags for unmap, see <a href="#">l4_unmap_flags_t</a> .

Implemented in [L4Re::Util::\\_Cap\\_alloc](#).

References [L4\\_FP\\_ALL\\_SPACES](#), and [L4\\_INVALID\\_CAP](#).

The documentation for this class was generated from the following file:

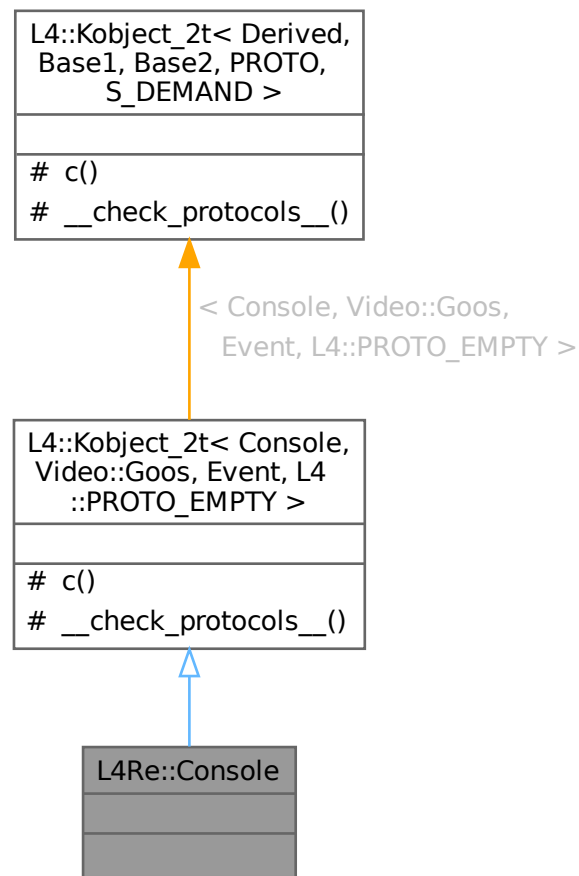
- [l4/re/cap\\_alloc](#)

## 16.279 L4Re::Console Class Reference

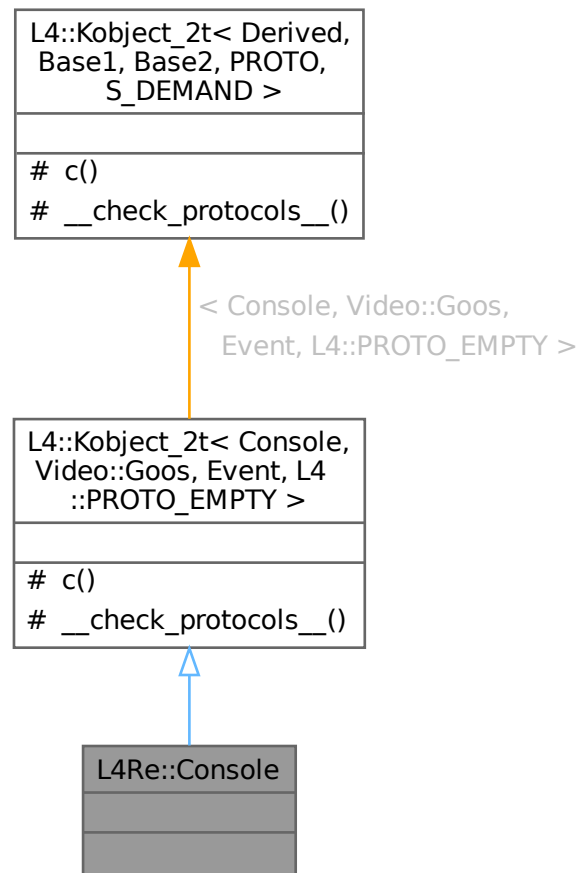
[Console](#) class.

```
#include <console>
```

Inheritance diagram for L4Re::Console:



Collaboration diagram for L4Re::Console:



### Additional Inherited Members

#### Protected Types inherited from

**L4::Kobject\_2t< Console, Video::Goos, Event, L4::PROTO\_EMPTY >**

- typedef Console [Class](#)  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< PROTO, Console > [\\_\\_iface](#)  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< [\\_\\_iface](#) >, Typeid::Merge\_list< typename Video::Goos::\_↔  
\_iface\_list, typename Event::\_iface\_list > > [\\_\\_iface\\_list](#)  
*The list of all RPC interfaces provided directly or through inheritance.*

#### Protected Member Functions inherited from

**L4::Kobject\_2t< Console, Video::Goos, Event, L4::PROTO\_EMPTY >**

- [L4::Cap< Class > c \(\)](#) const noexcept  
*Get the capability to ourselves.*

## Static Protected Member Functions inherited from L4::Kobject\_2t< Console, Video::Goos, Event, L4::PROTO\_EMPTY >

- static void `__check_protocols__()` noexcept  
*Helper to check for protocol conflicts.*

### 16.279.1 Detailed Description

`Console` class.

Definition at line 28 of file `console`.

The documentation for this class was generated from the following file:

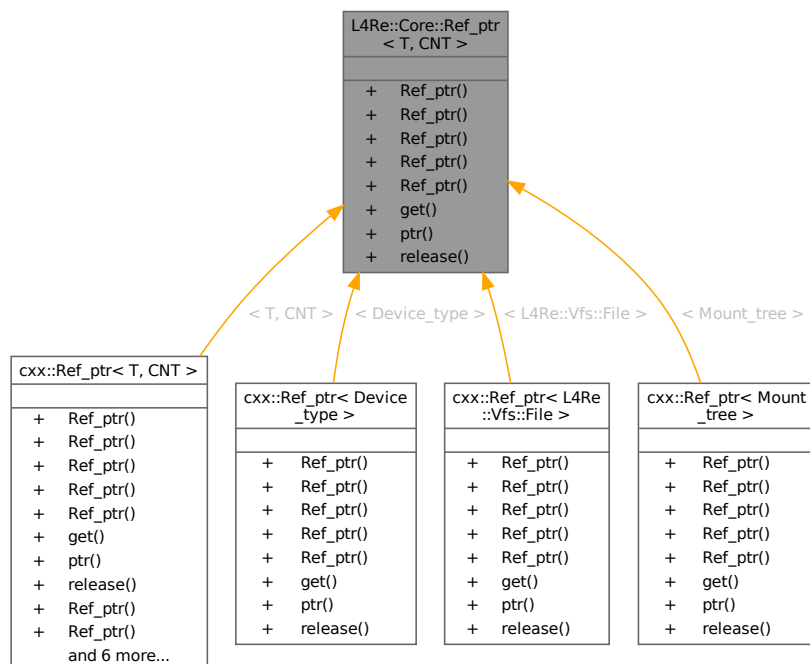
- `l4/re/console`

## 16.280 L4Re::Core::Ref\_ptr< T, CNT > Class Template Reference

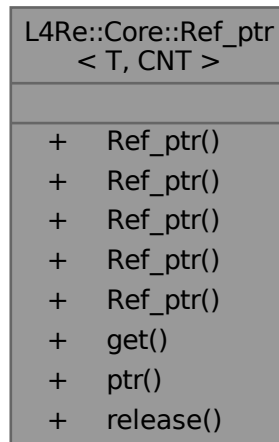
A reference-counting pointer with automatic cleanup.

```
#include <ref_ptr>
```

Inheritance diagram for L4Re::Core::Ref\_ptr< T, CNT >:



Collaboration diagram for L4Re::Core::Ref\_ptr< T, CNT >:



## Public Member Functions

- **Ref\_ptr** () noexcept  
*Default constructor creates a pointer with no managed object.*
- **Ref\_ptr** (Wp const &o) noexcept  
*Create a shared pointer from a weak pointer.*
- **Ref\_ptr** (decltype(nullptr) n) noexcept  
*allow creation from `nullptr`*
- template<typename X>  
**Ref\_ptr** (X \*o) noexcept  
*Create a shared pointer from a raw pointer.*
- **Ref\_ptr** (T \*o, bool d) noexcept  
*Create a shared pointer from a raw pointer without creating a new reference.*
- T \* **get** () const noexcept  
*Return a raw pointer to the object this shared pointer points to.*
- T \* **ptr** () const noexcept  
*Return a raw pointer to the object this shared pointer points to.*
- T \* **release** () noexcept  
*Release the shared pointer without removing the reference.*

## 16.280.1 Detailed Description

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
class L4Re::Core::Ref_ptr< T, CNT >
```

A reference-counting pointer with automatic cleanup.

### Template Parameters

<i>T</i>	Type of object the pointer points to.
<i>CNT</i>	Type of management class that manages the life time of the object.

This pointer is similar to the standard C++-11 `shared_ptr` but it does the reference counting directly in the object being pointed to, so that no additional management structures need to be allocated from the heap.

Classes that use this pointer type must implement two functions:

```
int remove_ref()
```

is called when a reference is removed and must return 0 when there are no further references to the object.

```
void add_ref()
```

is called when another `ref_ptr` to the object is created.

`Ref_obj` provides a simple implementation of this interface from which classes may inherit.

Definition at line 70 of file [ref\\_ptr](#).

## 16.280.2 Constructor & Destructor Documentation

### 16.280.2.1 `Ref_ptr()` [1/3]

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
cxx::Ref_ptr< T, CNT >::Ref_ptr (
    Wp const & o) [inline], [noexcept]
```

Create a shared pointer from a weak pointer.

Increases references.

Definition at line 88 of file [ref\\_ptr](#).

### 16.280.2.2 `Ref_ptr()` [2/3]

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
template<typename X>
cxx::Ref_ptr< T, CNT >::Ref_ptr (
    X * o) [inline], [explicit], [noexcept]
```

Create a shared pointer from a raw pointer.

In contrast to C++11 `shared_ptr` it is safe to use this constructor multiple times and have the same reference counter.

Definition at line 101 of file [ref\\_ptr](#).

### 16.280.2.3 `Ref_ptr()` [3/3]

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
cxx::Ref_ptr< T, CNT >::Ref_ptr (
    T * o,
    bool d) [inline], [noexcept]
```

Create a shared pointer from a raw pointer without creating a new reference.

#### Parameters



<i>o</i>	Pointer to the object.
<i>d</i>	Dummy parameter to select this constructor at compile time. The value may be true or false.

This is the counterpart to [release\(\)](#).

Definition at line 114 of file [ref\\_ptr](#).

## 16.280.3 Member Function Documentation

### 16.280.3.1 get()

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
T * cxx::Ref_ptr< T, CNT >::get () const [inline], [noexcept]
```

Return a raw pointer to the object this shared pointer points to.

This does not release the pointer or decrease the reference count.

Definition at line 121 of file [ref\\_ptr](#).

### 16.280.3.2 ptr()

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
T * cxx::Ref_ptr< T, CNT >::ptr () const [inline], [noexcept]
```

Return a raw pointer to the object this shared pointer points to.

This does not release the pointer or decrease the reference count.

Definition at line 127 of file [ref\\_ptr](#).

### 16.280.3.3 release()

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
T * cxx::Ref_ptr< T, CNT >::release () [inline], [noexcept]
```

Release the shared pointer without removing the reference.

#### Returns

A raw pointer to the managed object.

Definition at line 138 of file [ref\\_ptr](#).

The documentation for this class was generated from the following file:

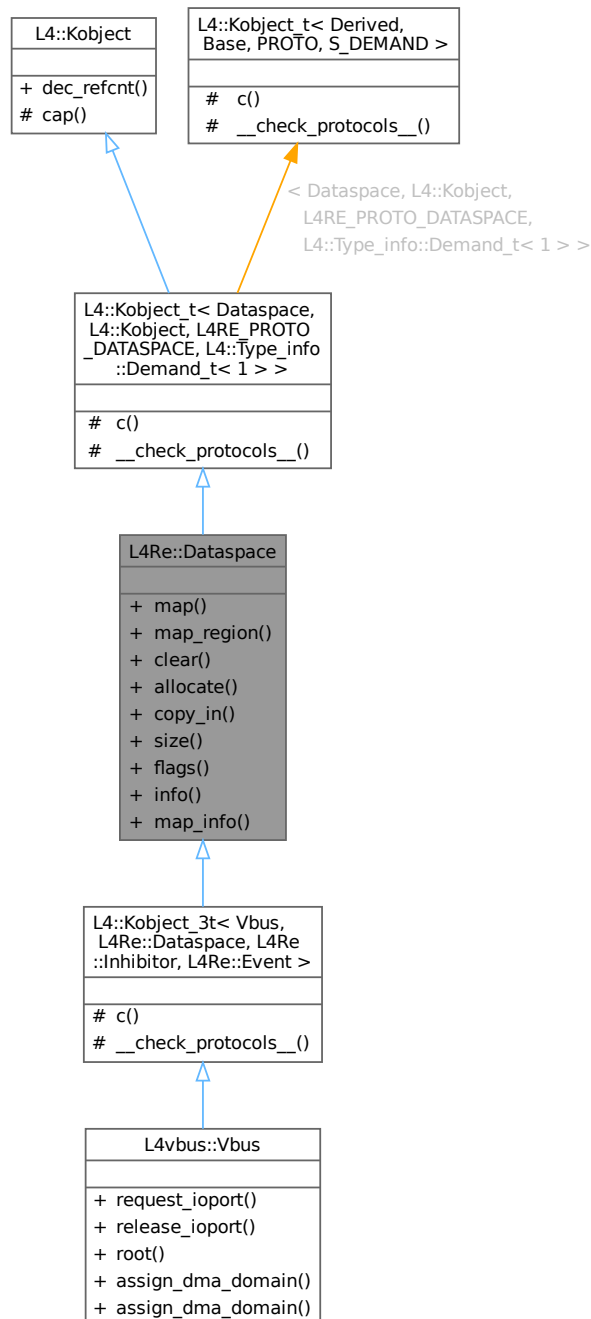
- [l4/cxx/ref\\_ptr](#)

## 16.281 L4Re::Dataspace Class Reference

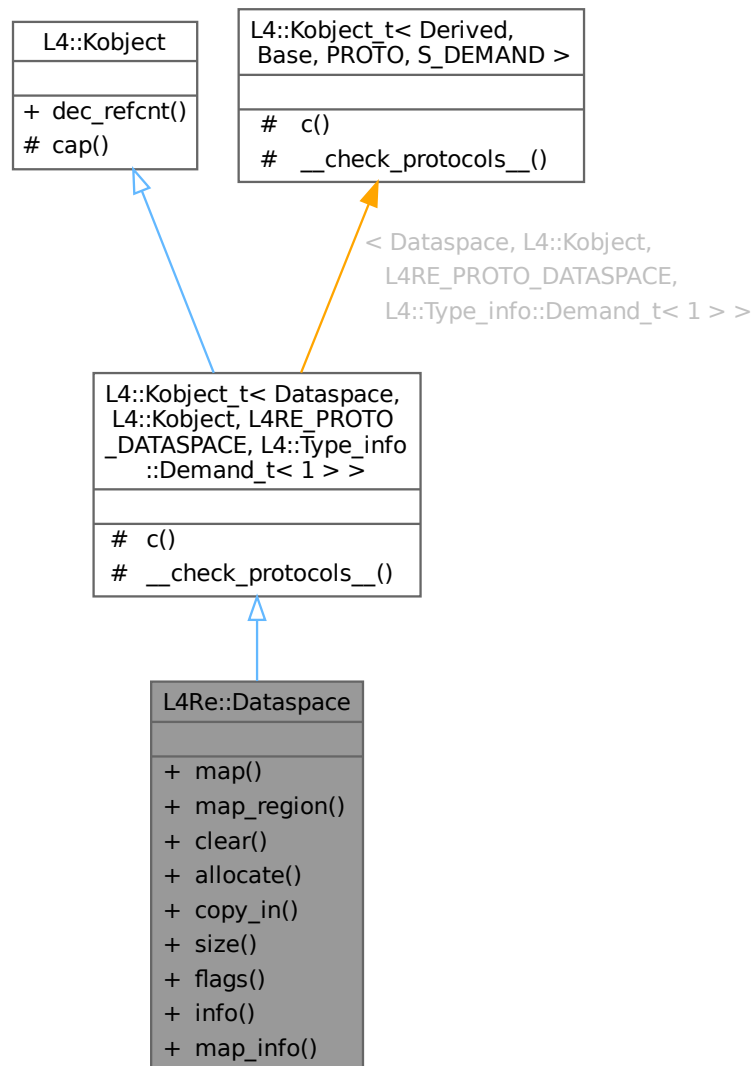
Interface for memory-like objects.

```
#include <dataspace>
```

Inheritance diagram for L4Re::Dataspace:



Collaboration diagram for L4Re::Dataspace:



## Data Structures

- struct [F](#)  
*Dataspace flags definitions.*
- struct [Stats](#)  
*Information about the dataspace.*

## Public Member Functions

- long [map](#) (Offset offset, Flags [flags](#), Map\_addr local\_addr, Map\_addr min\_addr, Map\_addr max\_addr, [L4::Cap](#)< [L4::Task](#) > dst=[L4::Cap](#)< [L4::Task](#) >::Invalid) const noexcept  
*Request a flexpage mapping from the dataspace.*

- long [map\\_region](#) (Offset offset, Flags [flags](#), Map\_addr min\_addr, Map\_addr max\_addr, [L4::Cap](#)< [L4::Task](#) > dst=[L4::Cap](#)< [L4::Task](#) >::Invalid) const noexcept  
*Map a part of a dataspace into a local memory area.*
- long [clear](#) (Offset offset, Size [size](#))  
*Clear parts of a dataspace.*
- long [allocate](#) (Offset offset, Size [size](#))  
*Allocate a range in the dataspace.*
- long [copy\\_in](#) (Offset dst\_offs, [L4::lpc::Cap](#)< [Dataspace](#) > src, Offset src\_offs, Size [size](#))  
*Copy contents from another dataspace.*
- Size [size](#) () const noexcept  
*Get size of a dataspace.*
- Flags [flags](#) () const noexcept  
*Get flags of the dataspace.*
- long [info](#) ([Stats](#) \*stats)  
*Get information on the dataspace.*
- long [map\\_info](#) ([l4\\_addr\\_t](#) \*start\_addr, [l4\\_addr\\_t](#) \*end\_addr)  
*Get mapping range of dataspace.*

## Public Member Functions inherited from [L4::Kobject](#)

- [l4\\_msgtag\\_t dec\\_refcnt](#) ([l4\\_mword\\_t](#) diff, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)())  
*Decrement the in kernel reference counter for the object.*

## Additional Inherited Members

## Protected Types inherited from

[L4::Kobject\\_t](#)< [Dataspace](#), [L4::Kobject](#), [L4RE\\_PROTO\\_DATASPACE](#), [L4::Type\\_info::Demand\\_t](#)< 1 > >

- typedef [Dataspace](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef [Typeid::Iface](#)< [PROTO](#), [Dataspace](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef [Typeid::Merge\\_list](#)< [Typeid::Iface\\_list](#)< [\\_\\_Iface](#) >, typename [L4::Kobject::\\_\\_Iface\\_list](#) > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Member Functions inherited from

[L4::Kobject\\_t](#)< [Dataspace](#), [L4::Kobject](#), [L4RE\\_PROTO\\_DATASPACE](#), [L4::Type\\_info::Demand\\_t](#)< 1 > >

- [L4::Cap](#)< [Class](#) > **c** () const noexcept  
*Get the capability to ourselves.*

## Protected Member Functions inherited from [L4::Kobject](#)

- [l4\\_cap\\_idx\\_t cap](#) () const noexcept  
*Return capability selector.*

**Static Protected Member Functions inherited from****[L4::Kobject\\_t< Dataspace, L4::Kobject, L4RE\\_PROTO\\_DATASPACE, L4::Type\\_info::Demand\\_t< 1 > >](#)**

- static void **\_\_check\_protocols\_\_** () noexcept  
*Helper to check for protocol conflicts.*

**16.281.1 Detailed Description**

Interface for memory-like objects.

Dataspaces are a central abstraction provided by [L4Re](#). A dataspace is an abstraction for any thing that is available via usual memory access instructions. A dataspace can be a file, as well as the memory-mapped registers of a device, or anonymous memory, such as a heap.

The dataspace interface defines a set of methods that allow any kind of dataspace to be attached (mapped) to the virtual address space of an [L4](#) task and then be accessed via memory-access instructions. The [L4Re::Rm](#) interface can be used to attach a dataspace to a virtual address space of a task paged by a certain instance of a region map.

**Include File**

```
#include <l4/re/dataspace>
```

**Examples**

[examples/libs/l4re/c++/mem\\_alloc/ma+rm.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#), and [examples/libs/l4re/c++/shared](#)

Definition at line 50 of file [dataspace](#).

**16.281.2 Member Function Documentation****16.281.2.1 allocate()**

```
long L4Re::Dataspace::allocate (
    Offset offset,
    Size size)
```

Allocate a range in the dataspace.

**Parameters**

<i>offset</i>	Offset in the dataspace, in bytes.
<i>size</i>	Size of the range, in bytes.

**Return values**

<i>L4_EOK</i>	Success
---------------	---------

<code>-L4_ERANGE</code>	Given range is outside the dataspace. (A dataspace provider may also silently ignore areas outside the dataspace.)
<code>-L4_ENOMEM</code>	Not enough memory available.
<code>&lt; 0</code>	IPC errors

On success, at least the given range is guaranteed to be allocated. The dataspace manager may also allocate more memory due to page granularity.

The memory is allocated with the same rights as the dataspace capability.

References [allocate\(\)](#), [copy\\_in\(\)](#), and [size\(\)](#).

Referenced by [allocate\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.281.2.2 clear()

```

long L4Re::Dataspace::clear (
    Offset offset,
    Size size)

```

Clear parts of a dataspace.

#### Parameters

<i>offset</i>	Offset within dataspace (in bytes).
<i>size</i>	Size of region to clear (in bytes).

#### Return values

$\geq 0$	Success.
<code>-L4_ERANGE</code>	Given range is outside the dataspace. (A dataspace provider may also silently ignore areas outside the dataspace.)
<code>-L4_EACCESS</code>	No <code>L4_CAP_FPAGE_W</code> right on dataspace capability.
$< 0$	IPC errors

Zeros out the memory. Depending on the type of memory the memory could also be deallocated and replaced by a shared zero-page.

References [clear\(\)](#), and [size\(\)](#).

Referenced by [clear\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.281.2.3 copy\_in()

```

long L4Re::Dataspace::copy_in (
    Offset dst_offs,
    L4::Ipc::Cap< Dataspace > src,
    Offset src_offs,
    Size size)

```

Copy contents from another dataspace.

#### Parameters

<code>dst_offs</code>	Offset in destination dataspace.
-----------------------	----------------------------------

<i>src</i>	Source dataspace to copy from.
<i>src_offs</i>	Offset in the source dataspace.
<i>size</i>	Size to copy (in bytes).

### Return values

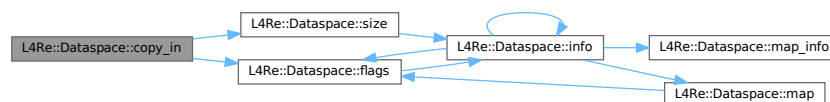
<i>L4_EOK</i>	Success
<i>-L4_EACCESS</i>	No <a href="#">L4_CAP_FPAGE_W</a> right on the destination dataspace.
<i>-L4_EINVAL</i>	Invalid parameter supplied.
<i>&lt; 0</i>	IPC errors

The copy operation may use copy-on-write mechanisms. The operation may also fail if both dataspace managers are not from the same dataspace manager or the dataspace managers do not cooperate.

References [flags\(\)](#), and [size\(\)](#).

Referenced by [allocate\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 16.281.2.4 flags()

```
Dataspace::Flags Dataspace::flags () const [noexcept]
```

Get flags of the dataspace.

### Return values



$\geq 0$	Flags of the dataspace
$< 0$	IPC errors

See also

[L4Re::Dataspace::F::Flags](#)

Definition at line 111 of file [dataspace\\_impl.h](#).

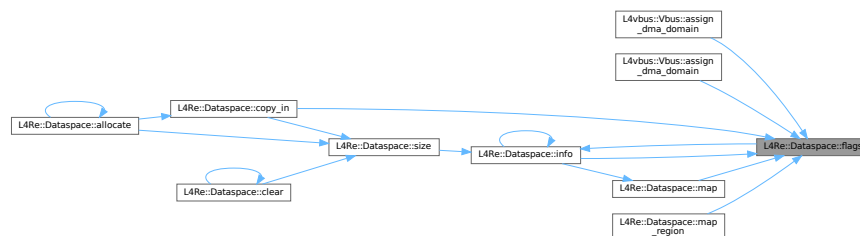
References [L4Re::Dataspace::Stats::flags](#), and [info\(\)](#).

Referenced by [L4vbus::Vbus::assign\\_dma\\_domain\(\)](#), [L4vbus::Vbus::assign\\_dma\\_domain\(\)](#), [copy\\_in\(\)](#), [info\(\)](#), [map\(\)](#), and [map\\_region\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.281.2.5 info()

```
long L4Re::Dataspace::info (
    Stats * stats)
```

Get information on the dataspace.

#### Parameters

out	stats	Dataspace information
-----	-------	-----------------------

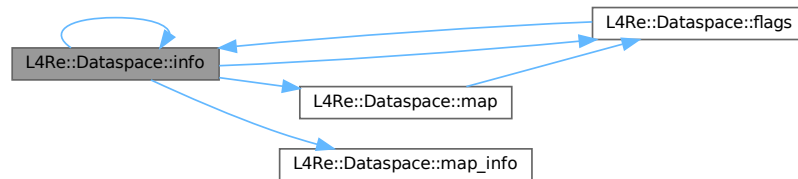
#### Return values

0	Success
<0	IPC errors

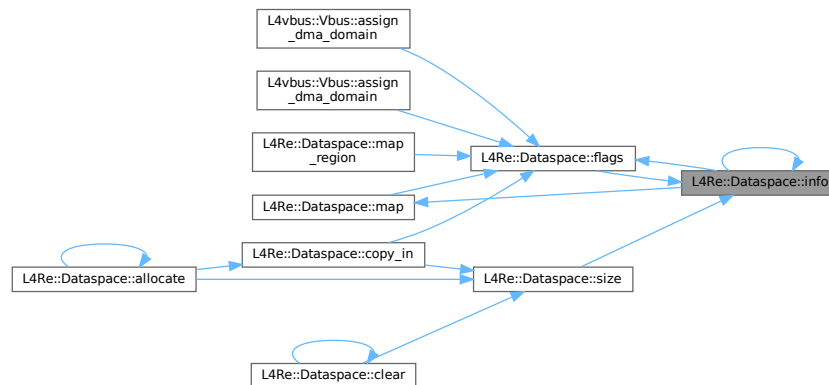
References [flags\(\)](#), [info\(\)](#), [L4\\_RPC](#), [L4\\_RPC\\_NF](#), [map\(\)](#), and [map\\_info\(\)](#).

Referenced by [flags\(\)](#), [info\(\)](#), and [size\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.281.2.6 map()

```

long Dataspace::map (
    Dataspace::Offset offset,
    Dataspace::Flags flags,
    Dataspace::Map_addr local_addr,
    Dataspace::Map_addr min_addr,
    Dataspace::Map_addr max_addr,
    L4::Cap< L4::Task > dst = L4::Cap<L4::Task>::Invalid) const [noexcept]

```

Request a flexpage mapping from the dataspace.

#### Parameters

<i>offset</i>	Offset to start within dataspace
<i>flags</i>	Dataspace flags, see <a href="#">L4Re::Dataspace::F::Flags</a> .
<i>local_addr</i>	Local address to map to.
<i>min_addr</i>	Defines start of receive window. (Rounded down to page size.)
<i>max_addr</i>	Defines end of receive window. (Rounded up to page size.)
<i>dst</i>	Optional destination task of the mapping. If invalid, the callers task is implicitly the destination.

### Return values

<i>L4_EOK</i>	Success
<i>-L4_ERANGE</i>	Invalid offset.
<i>-L4_EPERM</i>	Insufficient permission to map with requested rights.
<i>&lt;0</i>	IPC errors

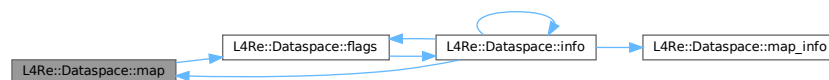
The map call will attempt to map the largest possible flexpage that covers the given local address and still fits into the region defined by `min_addr` and `max_addr`. If the given region is invalid or does not overlap the local address, the smallest valid page size is used.

Definition at line 85 of file [dataspace\\_impl.h](#).

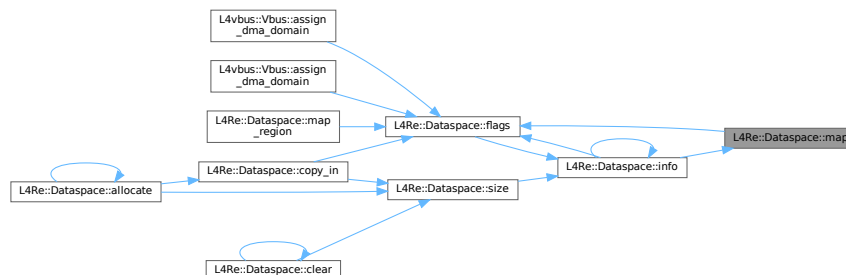
References [flags\(\)](#), and [L4\\_LOG2\\_PAGESIZE](#).

Referenced by [info\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.281.2.7 map\_info()

```
long L4Re::Dataspace::map_info (
    l4_addr_t * start_addr,
    l4_addr_t * end_addr) [inline]
```

Get mapping range of dataspace.

In case of a MMU-less system, the dataspace must be mapped at the correct address in the task because virtual and physical address must match. This method returns the start and end address of the physically contiguous buffer backing the dataspace.

On MMU-enabled system any page aligned address is permissible. On such systems the method is just a stub.

#### Parameters

out	<i>start_addr</i>	Start address of dataspace.
out	<i>end_addr</i>	End address (inclusive) of dataspace.

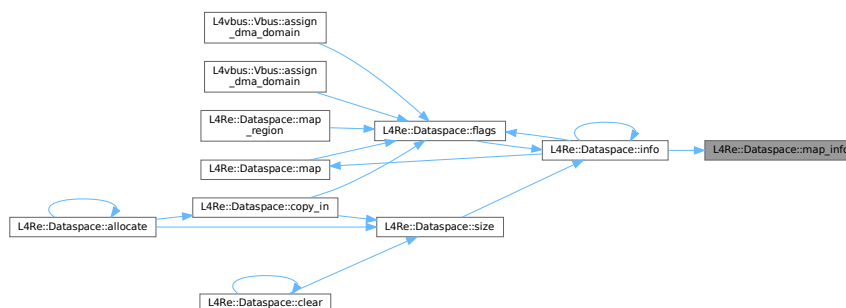
#### Return values

>0	Start/end address have been set and need to be obeyed.
0	No constraint of mapping address.
-L4_EPERM	Cannot infer mapping address. <a href="#">Dataspace</a> not mappable.
<0	IPC errors.

Definition at line 305 of file [dataspace](#).

Referenced by [info\(\)](#).

Here is the caller graph for this function:



### 16.281.2.8 map\_region()

```
long Dataspace::map_region (
    Dataspace::Offset offset,
    Dataspace::Flags flags,
    Dataspace::Map_addr min_addr,
    Dataspace::Map_addr max_addr,
    L4::Cap< L4::Task > dst = L4::Cap<L4::Task>::Invalid) const [noexcept]
```

Map a part of a dataspace into a local memory area.

#### Parameters

---

<i>offset</i>	Offset to start within dataspace.
<i>flags</i>	<a href="#">Dataspace</a> flags, see <a href="#">L4Re::Dataspace::F::Flags</a> .
<i>min_addr</i>	(Inclusive) start of the receive area.
<i>max_addr</i>	(Exclusive) end of receive area.
<i>dst</i>	Optional destination task of the mapping. If invalid, the callers task is implicitly the destination.

### Return values

<i>L4_EOK</i>	Success
<i>-L4_ERANGE</i>	Invalid offset or receive area larger than the dataspace.
<i>-L4_EPERM</i>	Insufficient permission to map with requested rights.
<i>&lt;0</i>	IPC errors

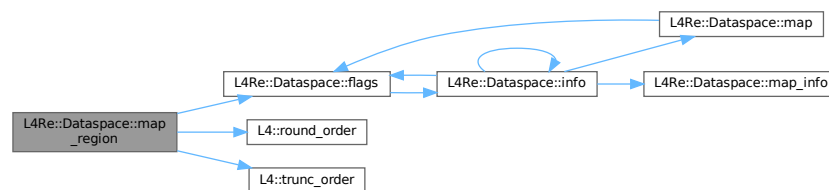
This is a convenience function which maps flexpages consecutively into the given memory area in the local task. The area is expected to be filled completely. If the dataspace is not large enough to provide the mappings for the entire size of the area, then an error is returned. Mappings may or may not have been already established at that point.

*offset* and *min\_addr* are rounded down to the next [L4\\_PAGESIZE](#) boundary when necessary. *max\_addr* is rounded up to the page boundary. If the resulting maximum address is less or equal than the minimum address, then the function is a noop.

Definition at line 45 of file [dataspace\\_impl.h](#).

References [flags\(\)](#), [L4\\_LOG2\\_PAGESIZE](#), [L4\\_UNLIKELY](#), [L4::round\\_order\(\)](#), and [L4::trunc\\_order\(\)](#).

Here is the call graph for this function:



### 16.281.2.9 size()

```
Dataspace::Size Dataspace::size () const [noexcept]
```

Get size of a dataspace.

**Returns**

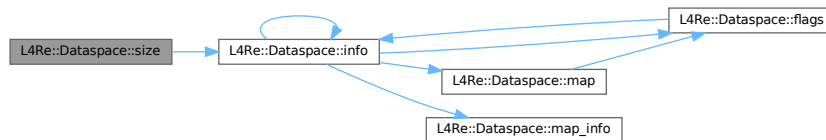
Size of the dataspace in bytes.

Definition at line 101 of file [dataspace\\_impl.h](#).

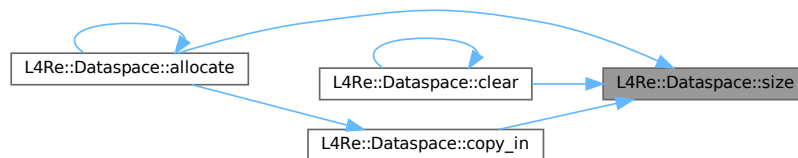
References [info\(\)](#), and [L4Re::Dataspace::Stats::size](#).

Referenced by [allocate\(\)](#), [clear\(\)](#), and [copy\\_in\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

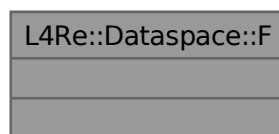
- [l4/re/dataspace](#)
- [l4/re/impl/dataspace\\_impl.h](#)

## 16.282 L4Re::Dataspace::F Struct Reference

[Dataspace](#) flags definitions.

```
#include <dataspace>
```

Collaboration diagram for L4Re::Dataspace::F:



## Public Types

- enum { [Caching\\_shift](#) = 4 }
- enum [Flags](#) {  
[R](#) = L4\_FPAGE\_RO , [Ro](#) = L4\_FPAGE\_RO , [RW](#) = L4\_FPAGE\_RW , [W](#) = L4\_FPAGE\_W ,  
[X](#) = L4\_FPAGE\_X , [RX](#) = L4\_FPAGE\_RX , [RWX](#) = L4\_FPAGE\_RWX , [Rights\\_mask](#) = 0x0f ,  
[Normal](#) = 0x00 , [Cacheable](#) = Normal , [Bufferable](#) = 0x10 , [Uncacheable](#) = 0x20 ,  
[Caching\\_mask](#) = 0x30 }  
*[Flags](#) for map operations.*

### 16.282.1 Detailed Description

[Dataspace](#) flags definitions.

Definition at line 57 of file [dataspace](#).

### 16.282.2 Member Enumeration Documentation

#### 16.282.2.1 anonymous enum

anonymous enum

#### Enumerator

<a href="#">Caching_shift</a>	shift value for caching flags
-------------------------------	-------------------------------

Definition at line 59 of file [dataspace](#).

#### 16.282.2.2 Flags

enum [L4Re::Dataspace::F::Flags](#)

[Flags](#) for map operations.

A dataspace implementation must check the requested flags during the map and other operations against the dataspace rights.

#### Enumerator

<a href="#">R</a>	Request read-only mapping.
<a href="#">Ro</a>	Request read-only mapping.
<a href="#">RW</a>	Request read-write mapping.
<a href="#">W</a>	Request write-only mapping.
<a href="#">X</a>	Request execute-only mapping.
<a href="#">RX</a>	Request read-execute mapping.
<a href="#">RWX</a>	Request read-write-execute mapping.



Rights_mask	All rights bits available for mappings.
Normal	Request normal (cached) memory mapping. This is the default if no other cache-related flag was specified.
Cacheable	Request normal memory mapping.
Bufferable	Request bufferable (write buffered) mappings.
Uncacheable	Request uncacheable memory mappings.
Caching_mask	Mask for caching flags.

Definition at line 70 of file [dataspace](#).

The documentation for this struct was generated from the following file:

- [l4/re/dataspace](#)

## 16.283 L4Re::Dataspace::Stats Struct Reference

Information about the dataspace.

```
#include <dataspace>
```

Collaboration diagram for L4Re::Dataspace::Stats:

L4Re::Dataspace::Stats	
+	size
+	flags

### Data Fields

- Size **size**  
*size*
- Flags **flags**  
*flags*

### 16.283.1 Detailed Description

Information about the dataspace.

Definition at line 126 of file [dataspace](#).

The documentation for this struct was generated from the following file:

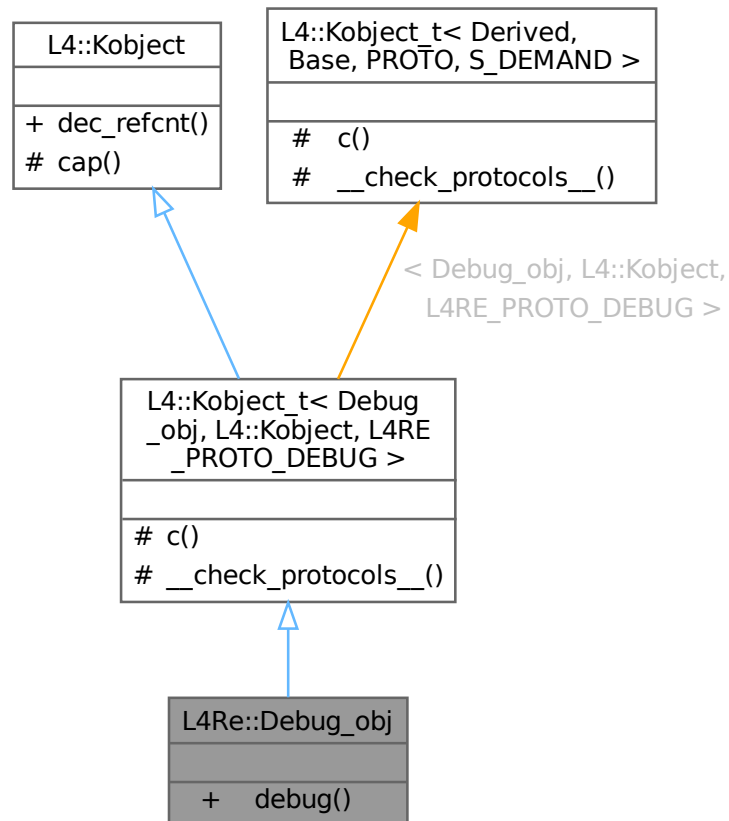
- [l4/re/dataspace](#)

## 16.284 L4Re::Debug\_obj Class Reference

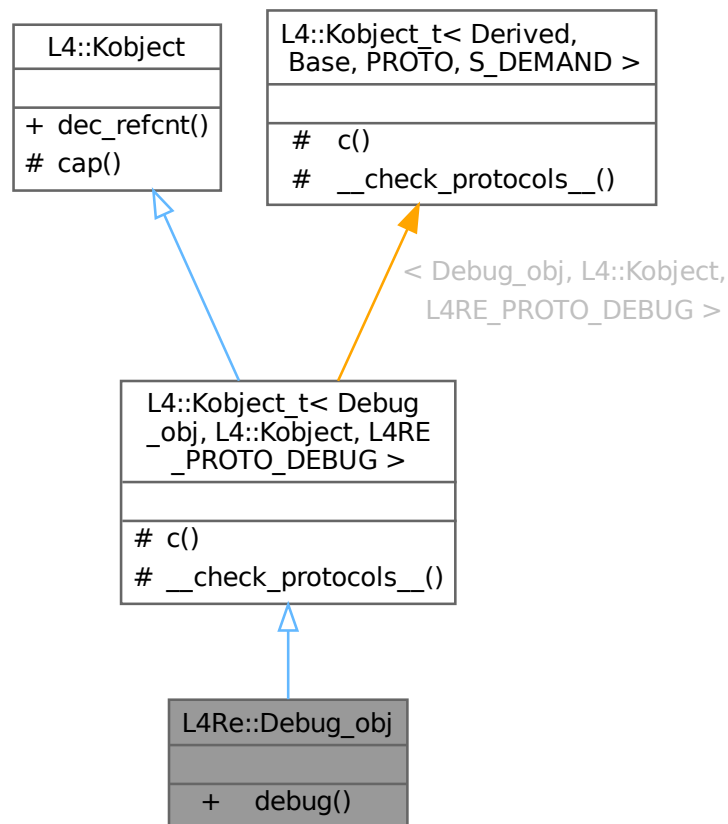
Debug interface.

```
#include <debug>
```

Inheritance diagram for L4Re::Debug\_obj:



Collaboration diagram for L4Re::Debug\_obj:



### Public Member Functions

- long [debug](#) (unsigned long function)  
*Debug call.*

### Public Member Functions inherited from [L4::Kobject](#)

- [l4\\_msgtag\\_t dec\\_refcnt](#) ([l4\\_mword\\_t diff](#), [l4\\_utcb\\_t \\*utcb=l4\\_utcb\(\)](#))  
*Decrement the in kernel reference counter for the object.*

### Additional Inherited Members

### Protected Types inherited from

[L4::Kobject\\_t< Debug\\_obj, L4::Kobject, L4RE\\_PROTO\\_DEBUG >](#)

- typedef [Debug\\_obj](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef [Typeid::Iface< PROTO, Debug\\_obj >](#) **\_\_Iface**  
*The interface description for the derived class.*
- typedef [Typeid::Merge\\_list< Typeid::Iface\\_list< \[\\\_\\\_Iface\]\(#\) >, typename L4::Kobject::\\_\\_Iface\\_list >](#) **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Member Functions inherited from [L4::Kobject\\_t< Debug\\_obj, L4::Kobject, L4RE\\_PROTO\\_DEBUG >](#)

- [L4::Cap< Class > c \(\)](#) const noexcept  
*Get the capability to ourselves.*

### Protected Member Functions inherited from [L4::Kobject](#)

- [l4\\_cap\\_idx\\_t cap \(\)](#) const noexcept  
*Return capability selector.*

### Static Protected Member Functions inherited from [L4::Kobject\\_t< Debug\\_obj, L4::Kobject, L4RE\\_PROTO\\_DEBUG >](#)

- static void [\\_\\_check\\_protocols\\_\\_ \(\)](#) noexcept  
*Helper to check for protocol conflicts.*

## 16.284.1 Detailed Description

Debug interface.

See also

[Debugging API](#) .

Definition at line 40 of file [debug](#).

## 16.284.2 Member Function Documentation

### 16.284.2.1 debug()

```
long L4Re::Debug_obj::debug (
    unsigned long function)
```

Debug call.

#### Parameters

<i>function</i>	Function to call.
-----------------	-------------------

**Returns**

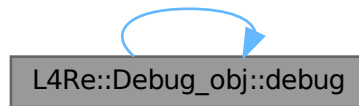
- L4\_EOK
- IPC errors

An object can provide a number of debug functions, each identified by some integer. This method is used to fan out to these functions from a common entry point.

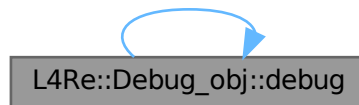
References [debug\(\)](#).

Referenced by [debug\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- [l4/re/debug](#)

## 16.285 L4Re::Default\_event\_payload Struct Reference

Default event stream payload.

```
#include <event>
```

Collaboration diagram for L4Re::Default\_event\_payload:

L4Re::Default_event_payload	
+	type
+	code
+	value
+	stream_id

### Data Fields

- unsigned short **type**  
*Type of event.*
- unsigned short **code**  
*Code of event.*
- int **value**  
*Value of event.*
- [l4\\_umword\\_t](#) **stream\_id**  
*Stream ID.*

## 16.285.1 Detailed Description

Default event stream payload.

Definition at line [232](#) of file [event](#).

The documentation for this struct was generated from the following file:

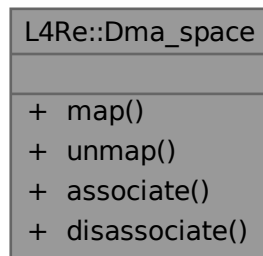
- [l4/re/event](#)

## 16.286 L4Re::Dma\_space Class Reference

Managed DMA Address Space.

```
#include <dma_space>
```

Collaboration diagram for L4Re::Dma\_space:



## Public Types

- enum [Direction](#) { [Bidirectional](#) , [To\\_device](#) , [From\\_device](#) , [None](#) }  
*Direction of the DMA transfers.*
- enum [Attribute](#) { [No\\_sync](#) }  
*Attributes used for the memory region during the transfer.*
- enum [Space\\_attrib](#) { [Coherent](#) , [Phys\\_space](#) }  
*Attributes assigned to the DMA space when associated with a specific device.*
- typedef [l4\\_uint64\\_t](#) [Dma\\_addr](#)  
*Data type for DMA addresses.*
- typedef [L4::Types::Flags](#)< [Attribute](#) > [Attributes](#)  
*Attributes for DMA mappings.*
- typedef [L4::Types::Flags](#)< [Space\\_attrib](#) > [Space\\_attribs](#)  
*Attributes used when configuring the DMA space.*

## Public Member Functions

- long [map](#) ([L4::lpc::Cap](#)< [L4Re::Dataspace](#) > src, [L4Re::Dataspace::Offset](#) offset, [L4::lpc::ln\\_out](#)< [l4\\_size\\_t](#) \* > size, [Attributes](#) attrs, [Direction](#) dir, [Dma\\_addr](#) \*dma\_addr)  
*Map the given part of this data space into the DMA address space.*
- long [unmap](#) ([Dma\\_addr](#) dma\_addr, [l4\\_size\\_t](#) size, [Attributes](#) attrs, [Direction](#) dir)  
*Unmap the given part of this data space from the DMA address space.*
- long [associate](#) ([L4::lpc::Opt](#)< [L4::lpc::Cap](#)< [L4::Task](#) > > dma\_task, [Space\\_attribs](#) attr)  
*Associate a (kernel) DMA space for a device to this DMA\_space.*
- long [disassociate](#) ()  
*Disassociate the (kernel) DMA space from this DMA\_space.*

### 16.286.1 Detailed Description

Managed DMA Address Space.

A managed [Dma\\_space](#) represents the [L4Re](#) abstraction of an DMA address space of one or several devices. Devices are assigned to a managed [Dma\\_space](#) by binding the [Dma\\_space](#) to the respective DMA domain (see [L4vbus::Vbus::assign\\_dma\\_domain\(\)](#)), which might link the [Dma\\_space](#) with a kernel [DMA space](#). Note that several DMA domains can be bound to the same [Dma\\_space](#). Whenever a device needs direct access to parts of an [L4Re::Dataspace](#), that part of the data space must be mapped to the managed [Dma\\_space](#) that is assigned to that device. Binding to DMA domains must happen before mapping. After the DMA accesses to the memory are finished the memory must be unmapped from the device's DMA address space.

Mapping to a managed DMA address space, using [map\(\)](#), makes the given parts of the data space visible to the associated device at the returned DMA address. As long as the memory is mapped into a DMA space it is 'pinned' and cannot be subject to dynamic memory management such as swapping. Additionally, [map\(\)](#) is responsible for the necessary syncing operations before the DMA.

[unmap\(\)](#) is the reverse operation to [map\(\)](#) and unmaps the given data-space part for the DMA address space. [unmap\(\)](#) is responsible for the necessary sync operations after the DMA.

Definition at line 52 of file [dma\\_space](#).

### 16.286.2 Member Typedef Documentation

#### 16.286.2.1 Attributes

```
typedef L4::Types::Flags<Attribute> L4Re::Dma_space::Attributes
```

[Attributes](#) for DMA mappings.

See also

[Attribute](#)

Definition at line 97 of file [dma\\_space](#).

### 16.286.3 Member Enumeration Documentation

#### 16.286.3.1 Attribute

```
enum L4Re::Dma_space::Attribute
```

[Attributes](#) used for the memory region during the transfer.

See also

[Attributes](#)

**Enumerator**



No_sync	Do not sync the memory hierarchy. When this flag is <i>not set</i> (default) the memory region shall be made coherent to the point-of-coherency of the device associated with this <a href="#">Dma_space</a> . When using this attribute the client is responsible for syncing the memory hierarchy for DMA. This can either be done using the cache API or by another <a href="#">map()</a> or <a href="#">unmap()</a> operation of the same part of the data space (without the <a href="#">No_sync</a> attribute).
---------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Definition at line 76 of file [dma\\_space](#).

### 16.286.3.2 Direction

enum [L4Re::Dma\\_space::Direction](#)

[Direction](#) of the DMA transfers.

#### Enumerator

Bidirectional	device reads and writes to the memory
To_device	device reads the memory
From_device	device writes to the memory
None	device is coherently connected to the memory

Definition at line 64 of file [dma\\_space](#).

### 16.286.3.3 Space\_attrib

enum [L4Re::Dma\\_space::Space\\_attrib](#)

[Attributes](#) assigned to the DMA space when associated with a specific device.

See also

[Space\\_attribs](#)

#### Enumerator

Coherent	The device is connected coherently with the cache. This means that the <a href="#">map()</a> and <a href="#">unmap()</a> do not need to sync CPU caches before and after DMA.
Phys_space	The DMA space has no DMA task assigned and uses the CPUs physical memory.

Definition at line 104 of file [dma\\_space](#).

## 16.286.4 Member Function Documentation

### 16.286.4.1 associate()

```
long L4Re::Dma_space::associate (
    L4::Ipc::Opt< L4::Ipc::Cap< L4::Task > > dma_task,
    Space_attribs attr)
```

Associate a (kernel) [DMA space](#) for a device to this [Dma\\_space](#).

#### Parameters

in	<i>dma_task</i>	The (kernel) <a href="#">DMA space</a> used for the device that shall be associated with this DMA space. In case no IOMMU is present or configured, the <i>dma_task</i> might be an invalid capability when <a href="#">L4Re::Dma_space::Phys_space</a> is set in <i>attr</i> , in this case the CPUs physical memory is used as DMA address space.
in	<i>attr</i>	<a href="#">Attributes</a> for this DMA space. See <a href="#">L4Re::Dma_space::Space_attr</a> .

### Return values

<i>L4_EOK</i>	Operation successful.
<i>-L4_EPERM</i>	Insufficient permissions; see precondition.
<i>-L4_EINVAL</i>	
<i>-L4_ENOENT</i>	

### Precondition

The invoked [Dma\\_space](#) capability must have the permission [L4\\_CAP\\_FPAGE\\_W](#).

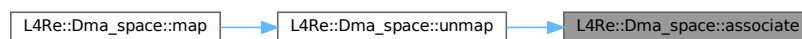
References [disassociate\(\)](#).

Referenced by [unmap\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.286.4.2 disassociate()

```
long L4Re::Dma_space::disassociate ()
```

Disassociate the (kernel) [DMA space](#) from this [Dma\\_space](#).

### Return values

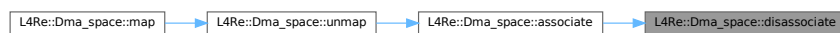
<code>L4_EOK</code>	Operation successful.
<code>-L4_EPERM</code>	Insufficient permissions; see precondition.
<code>-L4_ENOENT</code>	

#### Precondition

The invoked [Dma\\_space](#) capability must have the permission [L4\\_CAP\\_FPAGE\\_W](#).

Referenced by [associate\(\)](#).

Here is the caller graph for this function:



### 16.286.4.3 map()

```

long L4Re::Dma_space::map (
    L4::Ipc::Cap< L4Re::Dataspace > src,
    L4Re::Dataspace::Offset offset,
    L4::Ipc::In_out< l4_size_t * > size,
    Attributes attrs,
    Direction dir,
    Dma_addr * dma_addr)

```

Map the given part of this data space into the DMA address space.

#### Parameters

in	<i>src</i>	Source data space (that describes the memory). Caller needs write right to the data space.
in	<i>offset</i>	The offset (bytes) within <i>src</i> .
in, out	<i>size</i>	The size (bytes) of the region to be mapped for DMA, after successful mapping the size returned is the size mapped for DMA as a single block. This size might be smaller than the original input size, in this case the caller might call <a href="#">map()</a> again with a new offset and the remaining size.
in	<i>attrs</i>	The attributes used for this DMA mapping (a combination of <a href="#">Dma_space::Attribute</a> values).
in	<i>dir</i>	The direction of the DMA transfer issued with this mapping. The same value must later be passed to <a href="#">unmap()</a> .
out	<i>dma_addr</i>	The DMA address to use for DMA with the associated device.

#### Return values

<i>L4_EOK</i>	Operation successful.
<i>-L4_EPERM</i>	Insufficient permissions; see precondition.
<i>-L4_EINVAL</i>	The capability <i>src</i> is invalid or does not refer to a valid dataspace.
<i>-L4_EEXIST</i>	The specified region overlaps an existing mapping.
<i>-L4_ENOMEM</i>	Not enough memory to allocate internal datastructures.
<i>-L4_ERANGE</i>	<i>offset</i> is larger than the size of the dataspace.

#### Precondition

The capability *src* must have the permission [L4\\_CAP\\_FPAGE\\_W](#).

#### Note

[associate\(\)](#) must be called prior to mapping memory. Usually this is done implicitly when binding the managed [Dma\\_space](#) to a DMA domain (see [L4vbus::Vbus::assign\\_dma\\_domain\(\)](#)).

References [unmap\(\)](#).

Here is the call graph for this function:



#### 16.286.4.4 unmap()

```

long L4Re::Dma_space::unmap (
    Dma_addr dma_addr,
    l4_size_t size,
    Attributes attrs,
    Direction dir)
  
```

Unmap the given part of this data space from the DMA address space.

#### Parameters

<i>dma_addr</i>	The DMA address (returned by <a href="#">Dma_space::map()</a> ).
<i>size</i>	The size (bytes) of the memory region to unmap.
<i>attrs</i>	The attributes for the unmap (currently none).
<i>dir</i>	The direction of the finished DMA operation.

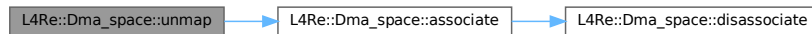
### Returns

0 in the case of success, a negative error code otherwise.

References [associate\(\)](#).

Referenced by [map\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

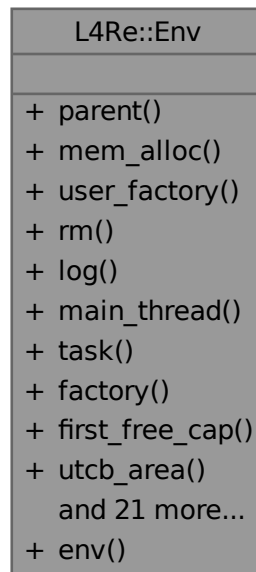
- [l4/re/dma\\_space](#)

## 16.287 L4Re::Env Class Reference

C++ interface of the initial environment that is provided to an [L4](#) task.

```
#include <env>
```

Collaboration diagram for L4Re::Env:



## Public Types

- typedef [l4re\\_env\\_cap\\_entry\\_t](#) **Cap\_entry**  
C++ type for an entry in the initial objects array.

## Public Member Functions

- [L4::Cap](#)< [Parent](#) > [parent](#) () const noexcept  
*Object-capability to the parent.*
- [L4::Cap](#)< [Mem\\_alloc](#) > [mem\\_alloc](#) () const noexcept  
*Object-capability to the memory allocator.*
- [L4::Cap](#)< [L4::Factory](#) > **user\_factory** () const noexcept  
*Object-capability to the user-level object factory.*
- [L4::Cap](#)< [Rm](#) > [rm](#) () const noexcept  
*Object-capability to the region map.*
- [L4::Cap](#)< [Log](#) > [log](#) () const noexcept  
*Object-capability to the logging service.*
- [L4::Cap](#)< [L4::Thread](#) > [main\\_thread](#) () const noexcept  
*Object-capability of the first user thread.*
- [L4::Cap](#)< [L4::Task](#) > [task](#) () const noexcept  
*Object-capability of the user task.*
- [L4::Cap](#)< [L4::Factory](#) > [factory](#) () const noexcept  
*Object-capability to the factory object available to the task.*
- [l4\\_cap\\_idx\\_t](#) [first\\_free\\_cap](#) () const noexcept

- First available capability selector.*

  - [l4\\_fpage\\_t utcb\\_area](#) () const noexcept

*UTCB area of the task.*
- [l4\\_addr\\_t first\\_free\\_utcb](#) () const noexcept

*First free UTCB.*
- [Cap\\_entry](#) const \* [initial\\_caps](#) () const noexcept

*Get a pointer to the first entry in the initial objects array.*
- [Cap\\_entry](#) const \* [get](#) (char const \*name, unsigned l) const noexcept

*Get the [Cap\\_entry](#) for the object named name.*
- template<typename T>
  - [L4::Cap](#)< T > [get\\_cap](#) (char const \*name, unsigned l) const noexcept

*Get the capability selector for the object named name.*
- template<typename T>
  - [L4::Cap](#)< T > [get\\_cap](#) (char const \*name) const noexcept

*Get the capability selector for the object named name.*
- void [parent](#) ([L4::Cap](#)< [Parent](#) > const &c) noexcept

*Set parent object-capability.*
- void [mem\\_alloc](#) ([L4::Cap](#)< [Mem\\_alloc](#) > const &c) noexcept

*Set memory allocator object-capability.*
- void [rm](#) ([L4::Cap](#)< [Rm](#) > const &c) noexcept

*Set region map object-capability.*
- void [log](#) ([L4::Cap](#)< [Log](#) > const &c) noexcept

*Set log object-capability.*
- void [main\\_thread](#) ([L4::Cap](#)< [L4::Thread](#) > const &c) noexcept

*Set object-capability of first user thread.*
- void [factory](#) ([L4::Cap](#)< [L4::Factory](#) > const &c) noexcept

*Set factory object-capability.*
- void [first\\_free\\_cap](#) ([l4\\_cap\\_idx\\_t](#) c) noexcept

*Set first available capability selector.*
- void [utcb\\_area](#) ([l4\\_fpage\\_t](#) utcb) noexcept

*Set UTCB area of the task.*
- void [first\\_free\\_utcb](#) ([l4\\_addr\\_t](#) u) noexcept

*Set first free UTCB.*
- [L4::Cap](#)< [L4::Scheduler](#) > [scheduler](#) () const noexcept

*Get the scheduler capability for the task.*
- void [scheduler](#) ([L4::Cap](#)< [L4::Scheduler](#) > const &c) noexcept

*Set the scheduler capability.*
- [L4::Cap](#)< [Itas](#) > [itas](#) () const noexcept

*Object-capability to the ITAS services.*
- void [itas](#) ([L4::Cap](#)< [Itas](#) > const &c) noexcept

*Set the ITAS capability.*
- [L4::Cap](#)< [Dbg\\_events](#) > [dbg\\_events](#) () const noexcept

*Object-capability to a debugger events service.*
- void [dbg\\_events](#) ([L4::Cap](#)< [Dbg\\_events](#) > const &dbg\_events) noexcept

*Set the dbg\_events capability.*
- void [initial\\_caps](#) ([Cap\\_entry](#) \*first) noexcept

*Set the pointer to the first [Cap\\_entry](#) in the initial objects array.*

## Static Public Member Functions

- static [Env](#) const \* [env](#) () noexcept

*Returns the initial environment for the current task.*

### 16.287.1 Detailed Description

C++ interface of the initial environment that is provided to an [L4](#) task.

The initial environment is provided to each [L4](#) task that is started by an [L4Re](#) conform loader, such as the Moe root task. The initial environment provides access to a set of initial capabilities and some additional information about the available resources, such as free UTCBs (see [Virtual Registers](#) ) and available entries in capability table (provided by the micro kernel).

Each of the initial capabilities is stored at a fixed index in the task's capability table and the [L4](#) runtime environment provides convenience functions to retrieve the capabilities. See the table below for an comprehensive overview.

Name	Object Type	Convenience Function
parent	<a href="#">L4Re::Parent</a>	<a href="#">L4Re::Env::parent()</a>
user_factory	<a href="#">L4::Factory</a>	<a href="#">L4Re::Env::user_factory()</a>
log	<a href="#">L4Re::Log</a>	<a href="#">L4Re::Env::log()</a>
main_thread	<a href="#">L4::Thread</a>	<a href="#">L4Re::Env::main_thread()</a>
rm	<a href="#">L4Re::Rm</a>	<a href="#">L4Re::Env::rm()</a>
factory	<a href="#">L4::Factory</a>	<a href="#">L4Re::Env::factory()</a>
task	<a href="#">L4::Task</a>	<a href="#">L4Re::Env::task()</a>
scheduler	<a href="#">L4::Scheduler</a>	<a href="#">L4Re::Env::scheduler()</a>
itas	<a href="#">L4Re::Itas</a>	<a href="#">L4Re::Env::itas()</a>

Additional information found in the initial environment is:

- First free entry in capability table
- The [UTCB](#) area (as flexpage)
- First free UTCB (address in the UTCB area)

#### Include File

```
#include <l4/re/env>
```

For an explanation of the default task capabilities see [l4\\_default\\_caps\\_t](#).

For the C interface refer to [Initial Environment](#).

Definition at line [78](#) of file [env](#).

### 16.287.2 Member Function Documentation

#### 16.287.2.1 dbg\_events() [1/2]

```
L4::Cap< Dbg_events > L4Re::Env::dbg\_events () const [inline], [noexcept]
```

Object-capability to a debugger events service.



**Returns**

Dbg\_events object-capability

This capability can be invalid.

Definition at line 310 of file [env](#).

Referenced by [dbg\\_events\(\)](#).

Here is the caller graph for this function:

**16.287.2.2 dbg\_events() [2/2]**

```
void L4Re::Env::dbg_events (
    L4::Cap< Dbg_events > const & dbg_events) [inline], [noexcept]
```

Set the dbg\_events capability.

**Parameters**

<i>dbg_events</i>	is the capability to be set for the debug events service.
-------------------	-----------------------------------------------------------

Note that the capability can be invalid.

Definition at line 320 of file [env](#).

References [dbg\\_events\(\)](#).

Here is the call graph for this function:



### 16.287.2.3 env()

```
Env const * L4Re::Env::env () [inline], [static], [noexcept]
```

Returns the initial environment for the current task.

#### Returns

Pointer to the initial environment class.

A typical use of this function is `L4Re::Env::env()-><member>()`

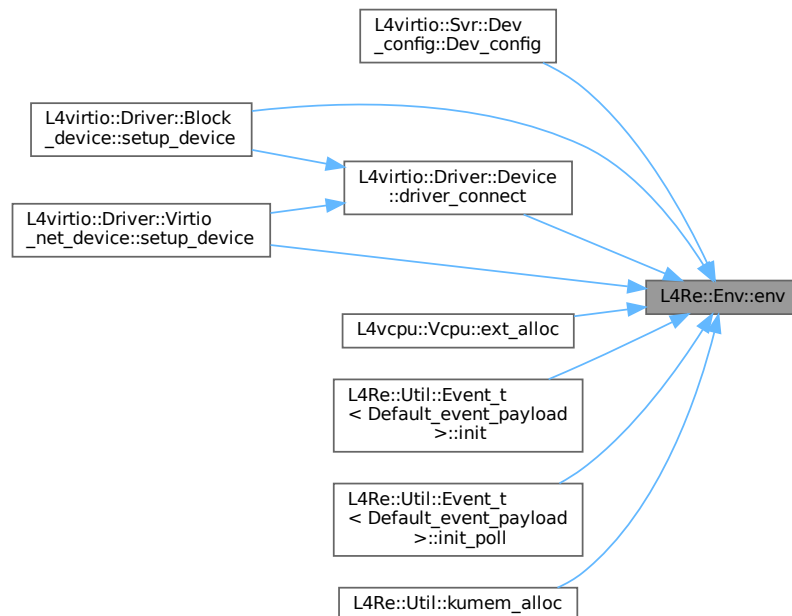
#### Examples

`examples/clntsrv/src/client.cc`, `examples/libs/l4re/c++/mem_alloc/ma+rm.cc`, `examples/libs/l4re/c++/shared_ds/ds_clnt.cc`, `examples/libs/l4re/c++/shared_ds/ds_srv.cc`, `examples/libs/l4re/streammap/client.cc`, and `examples/sys/migrate/thread_migrate.cc`

Definition at line 96 of file `env`.

Referenced by `L4virtio::Svr::Dev_config::Dev_config()`, `L4virtio::Driver::Device::driver_connect()`, `L4vcpu::Vcpu::ext_alloc()`, `L4Re::Util::Event_t< Default_event_payload >::init()`, `L4Re::Util::Event_t< Default_event_payload >::init_poll()`, `L4Re::Util::kumem_alloc()`, `L4virtio::Driver::Block_device::setup_device()`, and `L4virtio::Driver::Virtio_net_device::setup_device()`.

Here is the caller graph for this function:



**16.287.2.4 factory()** [1/2]

```
L4::Cap< L4::Factory > L4Re::Env::factory () const [inline], [noexcept]
```

Object-capability to the factory object available to the task.

**Returns**

Factory object-capability

Definition at line 144 of file [env](#).

**16.287.2.5 factory()** [2/2]

```
void L4Re::Env::factory (
    L4::Cap< L4::Factory > const & c) [inline], [noexcept]
```

Set factory object-capability.

**Parameters**

<i>c</i>	Factory object-capability
----------	---------------------------

Definition at line 249 of file [env](#).

**16.287.2.6 first\_free\_cap()** [1/2]

```
l4_cap_idx_t L4Re::Env::first_free_cap () const [inline], [noexcept]
```

First available capability selector.

**Returns**

First capability selector.

First capability selector available for use for in the application.

Definition at line 152 of file [env](#).

**16.287.2.7 first\_free\_cap()** [2/2]

```
void L4Re::Env::first_free_cap (
    l4_cap_idx_t c) [inline], [noexcept]
```

Set first available capability selector.

**Parameters**

<i>c</i>	First capability selector available to the application.
----------	---------------------------------------------------------

Definition at line 255 of file [env](#).

#### 16.287.2.8 first\_free\_utcb() [1/2]

```
l4_addr_t L4Re::Env::first_free_utcb () const [inline], [noexcept]
```

First free UTCB.

##### Returns

object-capability

First free UTCB within the UTCB area available for the application to use.

Definition at line 167 of file [env](#).

#### 16.287.2.9 first\_free\_utcb() [2/2]

```
void L4Re::Env::first_free_utcb (
    l4_addr_t u) [inline], [noexcept]
```

Set first free UTCB.

##### Parameters

<i>u</i>	First UTCB available for the application to use.
----------	--------------------------------------------------

Definition at line 267 of file [env](#).

#### 16.287.2.10 get()

```
Cap_entry const * L4Re::Env::get (
    char const * name,
    unsigned l) const [inline], [noexcept]
```

Get the [Cap\\_entry](#) for the object named *name*.

##### Parameters

<i>name</i>	is the name of the object.
<i>l</i>	is the length of the name, thus <i>name</i> might not be zero terminated.

**Returns**

A pointer to the [Cap\\_entry](#) for the object named *name*, or NULL if no such object was found.

Definition at line 185 of file [env](#).

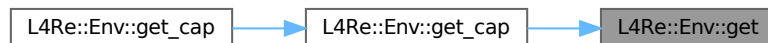
References [l4re\\_env\\_get\\_cap\\_l\(\)](#).

Referenced by [get\\_cap\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**16.287.2.11 get\_cap() [1/2]**

```

template<typename T>
L4::Cap< T > L4Re::Env::get_cap (
    char const * name) const [inline], [noexcept]
  
```

Get the capability selector for the object named *name*.

**Parameters**

<i>name</i>	is the name of the object (zero terminated).
-------------	----------------------------------------------

**Returns**

A capability selector for the object named *name*, or an invalid capability selector if no such object was found.

Definition at line 212 of file [env](#).

References [get\\_cap\(\)](#).

Here is the call graph for this function:



**16.287.2.12 get\_cap() [2/2]**

```
template<typename T>
L4::Cap< T > L4Re::Env::get_cap (
    char const * name,
    unsigned l) const [inline], [noexcept]
```

Get the capability selector for the object named *name*.

**Parameters**

<i>name</i>	is the name of the object.
<i>l</i>	is the length of the name, thus <i>name</i> might not be zero terminated.

**Returns**

A capability selector for the object named *name*, or an invalid capability selector if no such object was found.

**Examples**

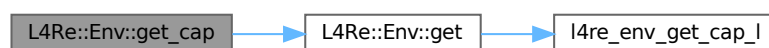
[examples/clntsrv/src/client.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#), and [examples/libs/l4re/streammap/client.cc](#).

Definition at line 197 of file [env](#).

References [get\(\)](#), and [L4\\_ENOENT](#).

Referenced by [get\\_cap\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



**16.287.2.13 initial\_caps()** [1/2]

```
Cap_entry const * L4Re::Env::initial_caps () const [inline], [noexcept]
```

Get a pointer to the first entry in the initial objects array.

**Returns**

A pointer to the first entry in the initial objects array.

Definition at line 174 of file [env](#).

**16.287.2.14 initial\_caps()** [2/2]

```
void L4Re::Env::initial_caps (
    Cap_entry * first) [inline], [noexcept]
```

Set the pointer to the first [Cap\\_entry](#) in the initial objects array.

**Parameters**

<i>first</i>	is the first element in the array.
--------------	------------------------------------

Definition at line 327 of file [env](#).

**16.287.2.15 itas()** [1/2]

```
L4::Cap< Itas > L4Re::Env::itas () const [inline], [noexcept]
```

Object-capability to the ITAS services.

**Returns**

ITAS object-capability

Attention: this capability might be invalid, depending on the system configuration. Regular applications must not use it directly as it is an implementation detail of the [L4Re](#) libc that is subject to change without notice!

Definition at line 294 of file [env](#).

**16.287.2.16 itas()** [2/2]

```
void L4Re::Env::itas (
    L4::Cap< Itas > const & c) [inline], [noexcept]
```

Set the ITAS capability.

**Parameters**

<code>c</code>	is the capability to be set as ITAS.
----------------	--------------------------------------

Definition at line 301 of file [env](#).

### 16.287.2.17 `log()` [1/2]

```
L4::Cap< Log > L4Re::Env::log () const [inline], [noexcept]
```

Object-capability to the logging service.

#### Returns

[Log](#) object-capability

Definition at line 126 of file [env](#).

### 16.287.2.18 `log()` [2/2]

```
void L4Re::Env::log (
    L4::Cap< Log > const & c) [inline], [noexcept]
```

Set log object-capability.

#### Parameters

<code>c</code>	<a href="#">Log</a> object-capability
----------------	---------------------------------------

Definition at line 237 of file [env](#).

### 16.287.2.19 `main_thread()` [1/2]

```
L4::Cap< L4::Thread > L4Re::Env::main_thread () const [inline], [noexcept]
```

Object-capability of the first user thread.

#### Returns

Object-capability of the first user thread.

Definition at line 132 of file [env](#).

### 16.287.2.20 `main_thread()` [2/2]

```
void L4Re::Env::main_thread (
    L4::Cap< L4::Thread > const & c) [inline], [noexcept]
```

Set object-capability of first user thread.

#### Parameters



<code>c</code>	First thread's object-capability
----------------	----------------------------------

Definition at line 243 of file [env](#).

#### 16.287.2.21 `mem_alloc()` [1/2]

```
L4::Cap< Mem_alloc > L4Re::Env::mem_alloc () const [inline], [noexcept]
```

Object-capability to the memory allocator.

##### Returns

Memory allocator object-capability

##### Examples

[examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#).

Definition at line 109 of file [env](#).

#### 16.287.2.22 `mem_alloc()` [2/2]

```
void L4Re::Env::mem_alloc (
    L4::Cap< Mem_alloc > const & c) [inline], [noexcept]
```

Set memory allocator object-capability.

##### Parameters

<code>c</code>	Memory allocator object-capability
----------------	------------------------------------

Definition at line 225 of file [env](#).

#### 16.287.2.23 `parent()` [1/2]

```
L4::Cap< Parent > L4Re::Env::parent () const [inline], [noexcept]
```

Object-capability to the parent.

##### Returns

[Parent](#) object-capability

Definition at line 103 of file [env](#).

#### 16.287.2.24 `parent()` [2/2]

```
void L4Re::Env::parent (
    L4::Cap< Parent > const & c) [inline], [noexcept]
```

Set parent object-capability.

##### Parameters

c	Parent object-capability
---	--------------------------

Definition at line 219 of file [env](#).

#### 16.287.2.25 `rm()` [1/2]

```
L4::Cap< Rm > L4Re::Env::rm () const [inline], [noexcept]
```

Object-capability to the region map.

##### Returns

Region map object-capability

##### Examples

[examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#), and [examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#).

Definition at line 120 of file [env](#).

#### 16.287.2.26 `rm()` [2/2]

```
void L4Re::Env::rm (
    L4::Cap< Rm > const & c) [inline], [noexcept]
```

Set region map object-capability.

##### Parameters

c	Region map object-capability
---	------------------------------

Definition at line 231 of file [env](#).

#### 16.287.2.27 `scheduler()` [1/2]

```
L4::Cap< L4::Scheduler > L4Re::Env::scheduler () const [inline], [noexcept]
```

Get the scheduler capability for the task.

##### Returns

The capability selector for the default scheduler used for this task.

Definition at line 275 of file [env](#).

#### 16.287.2.28 `scheduler()` [2/2]

```
void L4Re::Env::scheduler (
    L4::Cap< L4::Scheduler > const & c) [inline], [noexcept]
```

Set the scheduler capability.

##### Parameters

<code>c</code>	is the capability to be set as scheduler.
----------------	-------------------------------------------

Definition at line 282 of file [env](#).

### 16.287.2.29 task()

```
L4::Cap< L4::Task > L4Re::Env::task () const [inline], [noexcept]
```

Object-capability of the user task.

#### Returns

Object-capability of the user task.

Definition at line 138 of file [env](#).

### 16.287.2.30 utcb\_area() [1/2]

```
l4_fpage_t L4Re::Env::utcb_area () const [inline], [noexcept]
```

UTCB area of the task.

#### Returns

UTCB area

Definition at line 158 of file [env](#).

### 16.287.2.31 utcb\_area() [2/2]

```
void L4Re::Env::utcb_area (
    l4_fpage_t utcbs) [inline], [noexcept]
```

Set UTCB area of the task.

#### Parameters

<code>utcbs</code>	UTCB area
--------------------	-----------

Definition at line 261 of file [env](#).

The documentation for this class was generated from the following file:

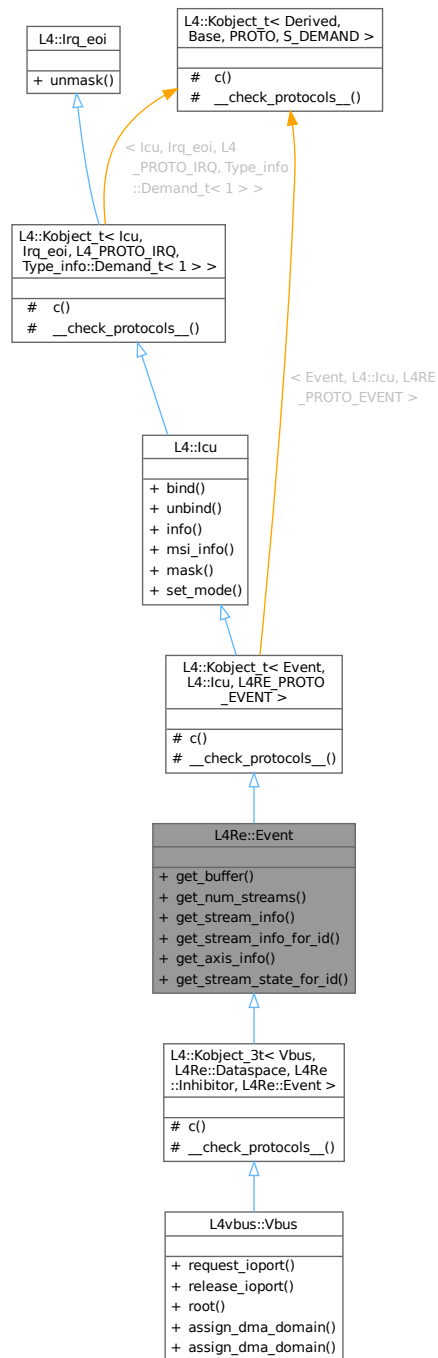
- [l4/re/env](#)

## 16.288 L4Re::Event Class Reference

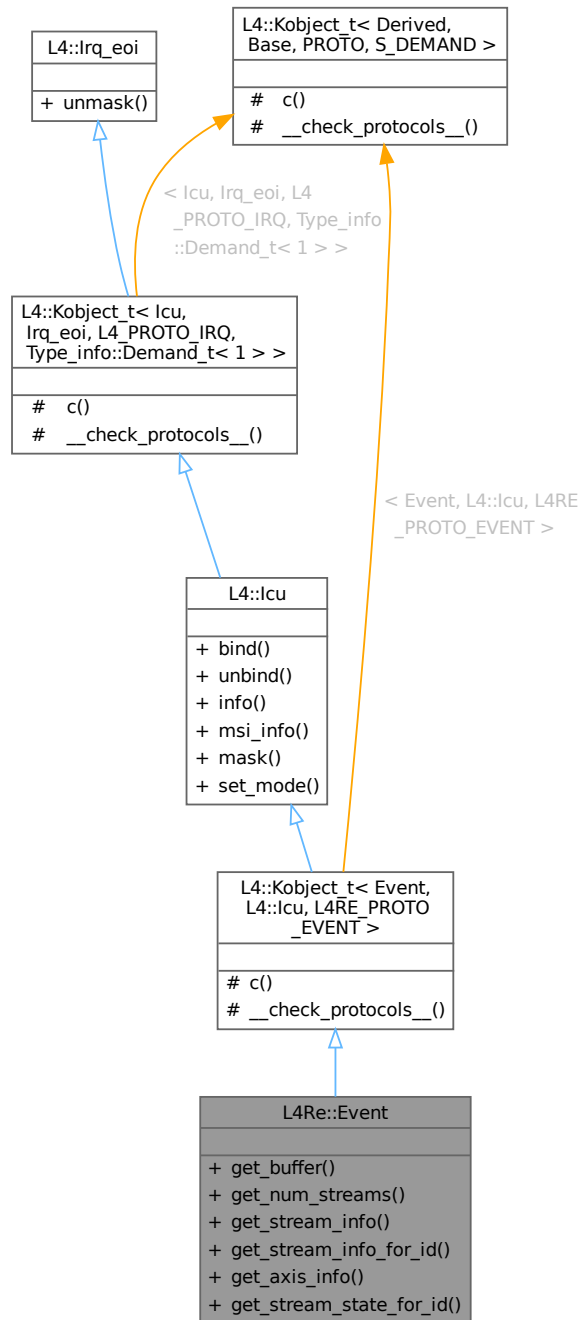
Event class.

```
#include <event>
```

Inheritance diagram for L4Re::Event:



Collaboration diagram for L4Re::Event:



## Public Member Functions

- long [get\\_buffer](#) (L4::lpc::Out< L4::Cap< Dataspace > > ds)  
Get event signal buffer.
- long [get\\_num\\_streams](#) ()  
Get number of event streams.
- long [get\\_stream\\_info](#) (int idx, Event\_stream\_info \*info)

*Get event stream infos.*

- long [get\\_stream\\_info\\_for\\_id](#) ([l4\\_umword\\_t](#) stream\_id, [Event\\_stream\\_info](#) \*info)

*Get event stream infos.*

- long [get\\_axis\\_info](#) ([l4\\_umword\\_t](#) stream\_id, unsigned naxes, unsigned const \*axis, [Event\\_absinfo](#) \*info) const noexcept

*Get event stream axis infos.*

- long [get\\_stream\\_state\\_for\\_id](#) ([l4\\_umword\\_t](#) stream\_id, [Event\\_stream\\_state](#) \*state)

*Get event stream state.*

## Public Member Functions inherited from [L4::Icu](#)

- [l4\\_msgtag\\_t](#) bind (unsigned irqnum, [L4::Cap](#)< [Triggerable](#) > irq, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept

*Bind an interrupt line of an interrupt controller to an interrupt object.*

- [l4\\_msgtag\\_t](#) unbind (unsigned irqnum, [L4::Cap](#)< [Triggerable](#) > irq, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept

*Remove binding of an interrupt line from the interrupt controller object.*

- [l4\\_msgtag\\_t](#) info ([l4\\_icu\\_info\\_t](#) \*info, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept

*Get information about the ICU features.*

- [l4\\_msgtag\\_t](#) msi\_info ([l4\\_umword\\_t](#) irqnum, [l4\\_uint64\\_t](#) source, [l4\\_icu\\_msi\\_info\\_t](#) \*msi\_info)

*Get MSI info about IRQ.*

- [l4\\_msgtag\\_t](#) mask (unsigned irqnum, [l4\\_umword\\_t](#) \*label=0, [l4\\_timeout\\_t](#) to=[L4\\_IPC\\_NEVER](#), [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept

*Mask an IRQ line.*

- [l4\\_msgtag\\_t](#) set\_mode (unsigned irqnum, [l4\\_umword\\_t](#) mode, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept

*Set interrupt mode.*

## Public Member Functions inherited from [L4::Irq\\_eoi](#)

- [l4\\_msgtag\\_t](#) unmask (unsigned irqnum, [l4\\_umword\\_t](#) \*label=0, [l4\\_timeout\\_t](#) to=[L4\\_IPC\\_NEVER](#), [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept

*Unmask the given interrupt line.*

## Additional Inherited Members

## Protected Types inherited from [L4::Kobject\\_t](#)< [Event](#), [L4::Icu](#), [L4RE\\_PROTO\\_EVENT](#) >

- typedef [Event](#) **Class**

*The target interface type (inheriting from [Kobject\\_t](#)).*

- typedef [Typeid::Iface](#)< [PROTO](#), [Event](#) > **\_\_Iface**

*The interface description for the derived class.*

- typedef [Typeid::Merge\\_list](#)< [Typeid::Iface\\_list](#)< **\_\_Iface** >, typename [L4::Icu](#)::[\\_\\_Iface\\_list](#) > **\_\_Iface\_list**

*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Types inherited from

## [L4::Kobject\\_t](#)< [Icu](#), [Irq\\_eoi](#), [L4\\_PROTO\\_IRQ](#), [Type\\_info::Demand\\_t](#)< 1 > >

- typedef [Icu](#) **Class**

*The target interface type (inheriting from [Kobject\\_t](#)).*

- typedef [Typeid::Iface](#)< [PROTO](#), [Icu](#) > **\_\_Iface**

*The interface description for the derived class.*

- typedef [Typeid::Merge\\_list](#)< [Typeid::Iface\\_list](#)< **\_\_Iface** >, typename [Irq\\_eoi](#)::[\\_\\_Iface\\_list](#) > **\_\_Iface\_list**

*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Member Functions inherited from [L4::Kobject\\_t< Event, L4::lcu, L4RE\\_PROTO\\_EVENT >](#)

- [L4::Cap< Class > c \(\)](#) const noexcept  
*Get the capability to ourselves.*

### Protected Member Functions inherited from [L4::Kobject\\_t< lcu, Irq\\_eoi, L4\\_PROTO\\_IRQ, Type\\_info::Demand\\_t< 1 > >](#)

- [L4::Cap< Class > c \(\)](#) const noexcept  
*Get the capability to ourselves.*

### Static Protected Member Functions inherited from [L4::Kobject\\_t< Event, L4::lcu, L4RE\\_PROTO\\_EVENT >](#)

- static void [\\_\\_check\\_protocols\\_\\_ \(\)](#) noexcept  
*Helper to check for protocol conflicts.*

### Static Protected Member Functions inherited from [L4::Kobject\\_t< lcu, Irq\\_eoi, L4\\_PROTO\\_IRQ, Type\\_info::Demand\\_t< 1 > >](#)

- static void [\\_\\_check\\_protocols\\_\\_ \(\)](#) noexcept  
*Helper to check for protocol conflicts.*

## 16.288.1 Detailed Description

[Event](#) class.

See also

[L4Re Event API](#)

Definition at line 137 of file [event](#).

## 16.288.2 Member Function Documentation

### 16.288.2.1 [get\\_axis\\_info\(\)](#)

```
long L4Re::Event::get_axis_info (
    l4_umword_t stream_id,
    unsigned naxes,
    unsigned const * axis,
    Event_absinfo * info) const [inline], [noexcept]
```

Get event stream axis infos.

#### Parameters

---

	<i>stream</i> ↔ <i>_id</i>	ID of the event stream.
in	<i>naxes</i>	Number of axis IDs and axis infos.
in	<i>axis</i>	Array of axis IDs.
out	<i>info</i>	Array of axis infos.

### Return values

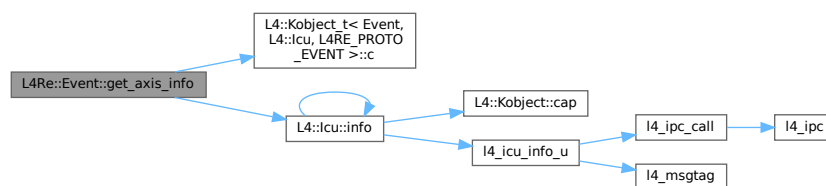
$\geq 0$	Number of returned axes infos.
$< 0$	Error code.

Definition at line 198 of file [event](#).

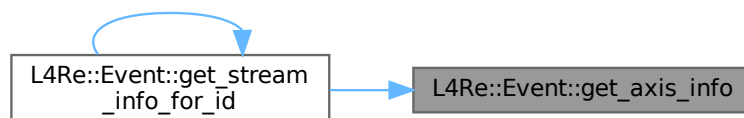
References [L4::Kobject\\_t< Event, L4::lcu, L4RE\\_PROTO\\_EVENT >::c\(\)](#), and [L4::lcu::info\(\)](#).

Referenced by [get\\_stream\\_info\\_for\\_id\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.288.2.2 get\_buffer()

```

long L4Re::Event::get_buffer (
    L4::Ipc::Out< L4::Cap< Dataspace > > ds)

```

Get event signal buffer.

### Parameters



out	ds	Event buffer.
-----	----	---------------

**Return values**

0	Success
<0	Error

References [get\\_buffer\(\)](#).

Referenced by [get\\_buffer\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**16.288.2.3 get\_num\_streams()**

```
long L4Re::Event::get_num_streams ()
```

Get number of event streams.

**Return values**

$\geq 0$	Number of streams.
<0	Error code.

References [get\\_num\\_streams\(\)](#).

Referenced by [get\\_num\\_streams\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 16.288.2.4 `get_stream_info()`

```
long L4Re::Event::get_stream_info (  
    int idx,  
    Event_stream_info * info)
```

Get event stream infos.

Deprecated. Use [get\\_stream\\_info\\_for\\_id\(\)](#).

##### Parameters

	<i>idx</i>	ID of the event stream.
out	<i>info</i>	<a href="#">Event</a> stream info.

##### Return values

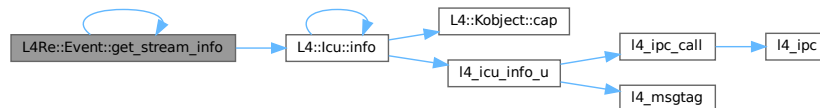
---

0	Success
<0	Error

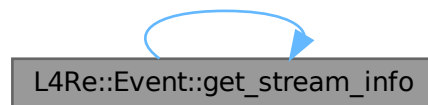
References [get\\_stream\\_info\(\)](#), and [L4::Icu::info\(\)](#).

Referenced by [get\\_stream\\_info\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.288.2.5 get\_stream\_info\_for\_id()

```

long L4Re::Event::get_stream_info_for_id (
    l4_umword_t stream_id,
    Event_stream_info * info)

```

Get event stream infos.

#### Parameters

	<i>stream_id</i>	ID of the event stream.
out	<i>info</i>	<a href="#">Event</a> stream info.

#### Return values

0	Success
---	---------



<0	Error
----	-------

The documentation for this class was generated from the following file:

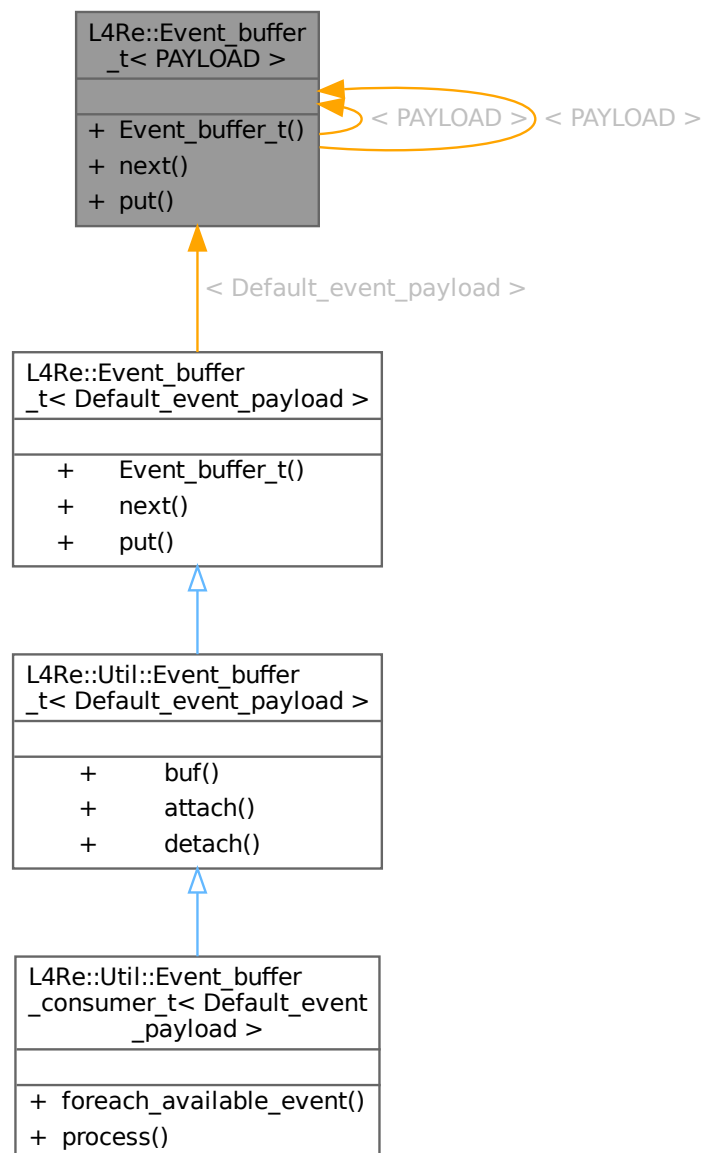
- l4/re/event

## 16.289 L4Re::Event\_buffer\_t< PAYLOAD > Class Template Reference

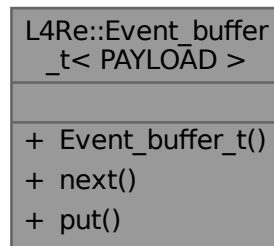
[Event](#) buffer class.

```
#include <event>
```

Inheritance diagram for L4Re::Event\_buffer\_t< PAYLOAD >:



Collaboration diagram for L4Re::Event\_buffer\_t< PAYLOAD >:



## Data Structures

- struct [Event](#)  
*[Event](#) structure used in buffer.*

## Public Member Functions

- [Event\\_buffer\\_t](#) (void \*buffer, [l4\\_addr\\_t](#) size)  
*Initialize event buffer.*
- [Event](#) \* [next](#) () noexcept  
*Next event in buffer.*
- bool [put](#) ([Event](#) const &ev) noexcept  
*Put event into buffer at current position.*

## 16.289.1 Detailed Description

```
template<typename PAYLOAD = Default_event_payload>
class L4Re::Event_buffer_t< PAYLOAD >
```

[Event](#) buffer class.

Definition at line [246](#) of file [event](#).

## 16.289.2 Constructor & Destructor Documentation

### 16.289.2.1 Event\_buffer\_t()

```
template<typename PAYLOAD = Default_event_payload>
L4Re::Event_buffer_t< PAYLOAD >::Event_buffer_t (
    void * buffer,
    l4_addr_t size) [inline]
```

Initialize event buffer.

## Parameters

---

<i>buffer</i>	Pointer to buffer.
<i>size</i>	Size of buffer in bytes.

Definition at line 293 of file [event](#).

## 16.289.3 Member Function Documentation

### 16.289.3.1 next()

```
template<typename PAYLOAD = Default_event_payload>
Event * L4Re::Event_buffer_t< PAYLOAD >::next () [inline], [noexcept]
```

Next event in buffer.

#### Returns

0 if no event available, event otherwise.

Definition at line 303 of file [event](#).

### 16.289.3.2 put()

```
template<typename PAYLOAD = Default_event_payload>
bool L4Re::Event_buffer_t< PAYLOAD >::put (
    Event const & ev) [inline], [noexcept]
```

Put event into buffer at current position.

#### Parameters

<i>ev</i>	<a href="#">Event</a> to put into the buffer.
-----------	-----------------------------------------------

#### Returns

false if buffer is full and entry could not be added.

Definition at line 320 of file [event](#).

The documentation for this class was generated from the following file:

- [l4/re/event](#)

## 16.290 L4Re::Event\_buffer\_t< PAYLOAD >::Event Struct Reference

[Event](#) structure used in buffer.

```
#include <event>
```

Collaboration diagram for L4Re::Event\_buffer\_t< PAYLOAD >::Event:

L4Re::Event_buffer _t< PAYLOAD >::Event	
+	time
+	free()

### Public Member Functions

- void **free** () noexcept  
*Free the entry.*

### Data Fields

- long long **time**  
*[Event](#) time stamp.*

### 16.290.1 Detailed Description

```
template<typename PAYLOAD = Default_event_payload>  
struct L4Re::Event_buffer_t< PAYLOAD >::Event
```

[Event](#) structure used in buffer.

Definition at line 253 of file [event](#).

The documentation for this struct was generated from the following file:

- l4/re/event

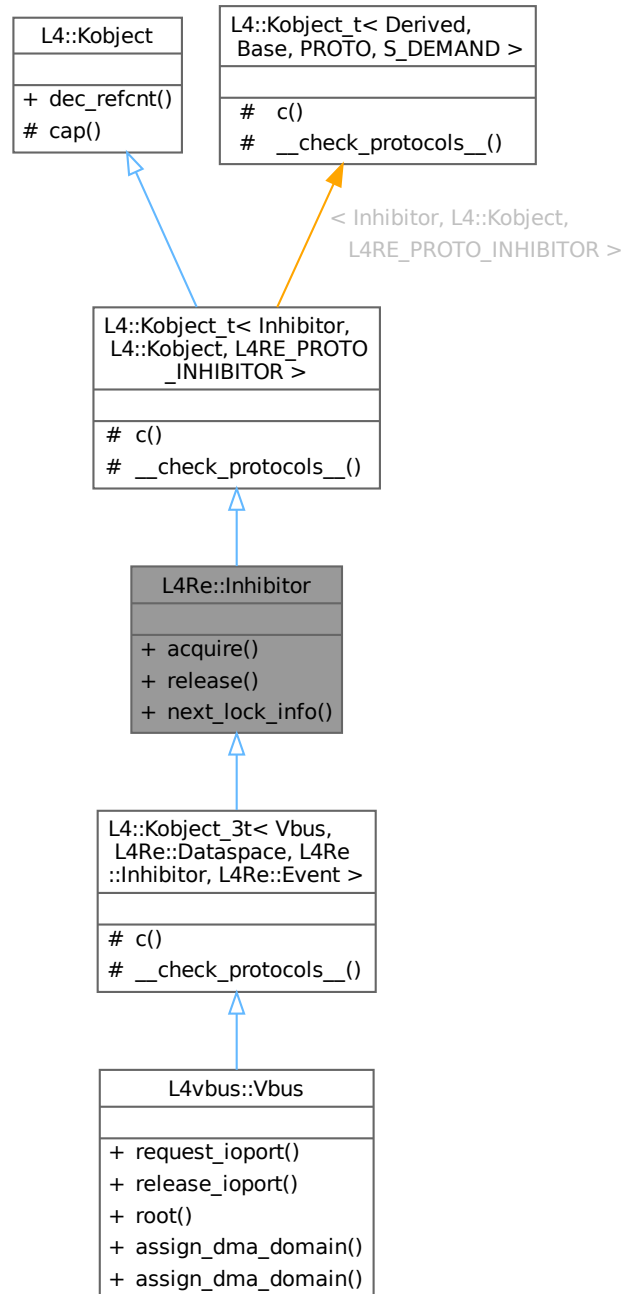


## 16.291 L4Re::Inhibitor Class Reference

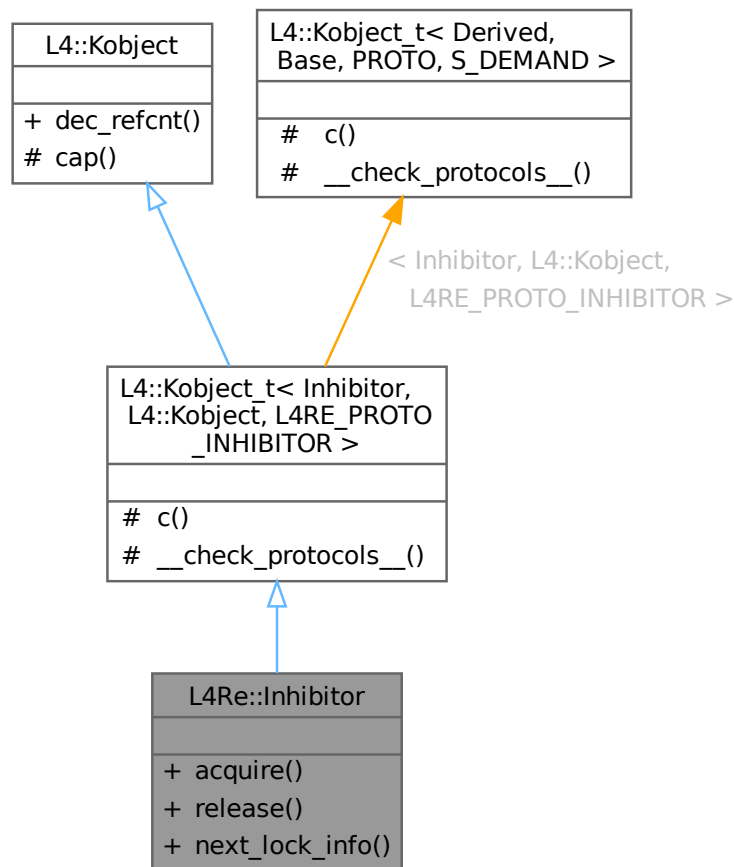
Set of inhibitor locks, which inhibit specific actions when held.

```
#include <inhibitor>
```

Inheritance diagram for L4Re::Inhibitor:



Collaboration diagram for L4Re::Inhibitor:



## Public Types

- enum { `Name_max` = 20 }

## Public Member Functions

- long `acquire` (`l4_umword_t` id, `L4::lpc::String<>` reason)  
*Acquire a specific inhibitor lock.*
- long `release` (`l4_umword_t` id)  
*Release a specific inhibitor lock.*
- long `next_lock_info` (char \*name, unsigned len, `l4_mword_t` current\_id=-1, `l4_utcb_t` \*utcb=`l4_utcb()`)  
*Get information for the next available inhibitor lock.*

## Public Member Functions inherited from L4::Kobject

- `l4_msgtag_t` `dec_refcnt` (`l4_mword_t` diff, `l4_utcb_t` \*utcb=`l4_utcb()`)  
*Decrement the in kernel reference counter for the object.*

## Additional Inherited Members

### Protected Types inherited from

[L4::Kobject\\_t< Inhibitor, L4::Kobject, L4RE\\_PROTO\\_INHIBITOR >](#)

- typedef Inhibitor **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< PROTO, Inhibitor > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename L4::Kobject::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Member Functions inherited from

[L4::Kobject\\_t< Inhibitor, L4::Kobject, L4RE\\_PROTO\\_INHIBITOR >](#)

- [L4::Cap< Class > c](#) () const noexcept  
*Get the capability to ourselves.*

### Protected Member Functions inherited from [L4::Kobject](#)

- [l4\\_cap\\_idx\\_t cap](#) () const noexcept  
*Return capability selector.*

### Static Protected Member Functions inherited from

[L4::Kobject\\_t< Inhibitor, L4::Kobject, L4RE\\_PROTO\\_INHIBITOR >](#)

- static void **\_\_check\_protocols** () noexcept  
*Helper to check for protocol conflicts.*

## 16.291.1 Detailed Description

Set of inhibitor locks, which inhibit specific actions when held.

This interface provides access to a set of inhibitor locks, each determined by an ID that is specific to the [Inhibitor](#) object. Each individual lock shall prevent, a specific (implementation defined) action to be executed, as long as the lock is held.

For example there can be an inhibitor lock to prevent a transition to suspend-to-RAM state and a different one to prevent shutdown.

A client shall take an inhibitor lock if it needs to execute code before the action is taken. For example a lock-screen application shall grab an inhibitor lock for the suspend action to be able to lock the screen before the system goes to sleep.

[Inhibitor](#) locks are usually closely related to specific events. Usually a server automatically subscribes a client holding a lock to the corresponding event. The server shall send the event to inform the client that an action is pending. Upon reception of the event, the client is supposed to release the corresponding inhibitor lock.

Definition at line 38 of file [inhibitor](#).

## 16.291.2 Member Enumeration Documentation

### 16.291.2.1 anonymous enum

anonymous enum

#### Enumerator

---

Name_max	The maximum length of a lock's name.
----------	--------------------------------------

Definition at line 42 of file [inhibitor](#).

## 16.291.3 Member Function Documentation

### 16.291.3.1 acquire()

```
long L4Re::Inhibitor::acquire (
    l4_umword_t id,
    L4::Ipc::String<> reason)
```

Acquire a specific inhibitor lock.

#### Parameters

<i>id</i>	ID of the inhibitor lock that the client intends to acquire
<i>reason</i>	The reason why you need the lock. Used for informing the user or debugging.

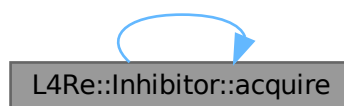
#### Return values

0	Success
-L4_ENODEV	The specified <i>id</i> does not exist.

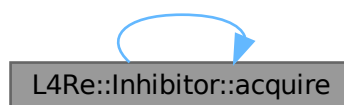
References [acquire\(\)](#).

Referenced by [acquire\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.291.3.2 next\_lock\_info()

```
long L4Re::Inhibitor::next_lock_info (
    char * name,
    unsigned len,
    l4_mword_t current_id = -1,
    l4_utcb_t * utcb = l4_utcb()) [inline]
```

Get information for the next available inhibitor lock.

#### Parameters

<i>name</i>	A pointer to a buffer for the name of the lock.
<i>len</i>	The length of the available buffer (usually <a href="#">Name_max</a> is used).
<i>current_id</i>	The ID of the last available lock, use -1 to get the first lock.
<i>utcb</i>	The UTCB to use for the message.

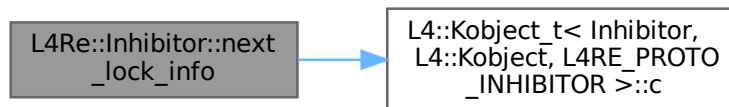
#### Return values

<i>&gt;0</i>	The ID of the next available lock if there is one (in this case <i>name</i> shall contain the name of the inhibitor lock).
<i>-L4_ENODEV</i>	There are no more locks.

Definition at line 84 of file [inhibitor](#).

References [L4::Kobject\\_t< Inhibitor, L4::Kobject, L4RE\\_PROTO\\_INHIBITOR >::c\(\)](#).

Here is the call graph for this function:



### 16.291.3.3 release()

```
long L4Re::Inhibitor::release (
    l4_umword_t id)
```

Release a specific inhibitor lock.

#### Parameters

<i>id</i>	The ID of the inhibitor lock to release.
-----------	------------------------------------------

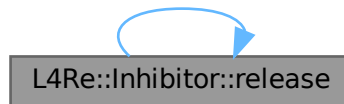
**Return values**

0	Success
-L4_ENODEV	Lock with the given <i>id</i> does not exist.

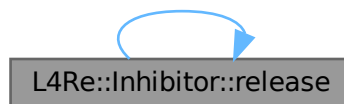
References [release\(\)](#).

Referenced by [release\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

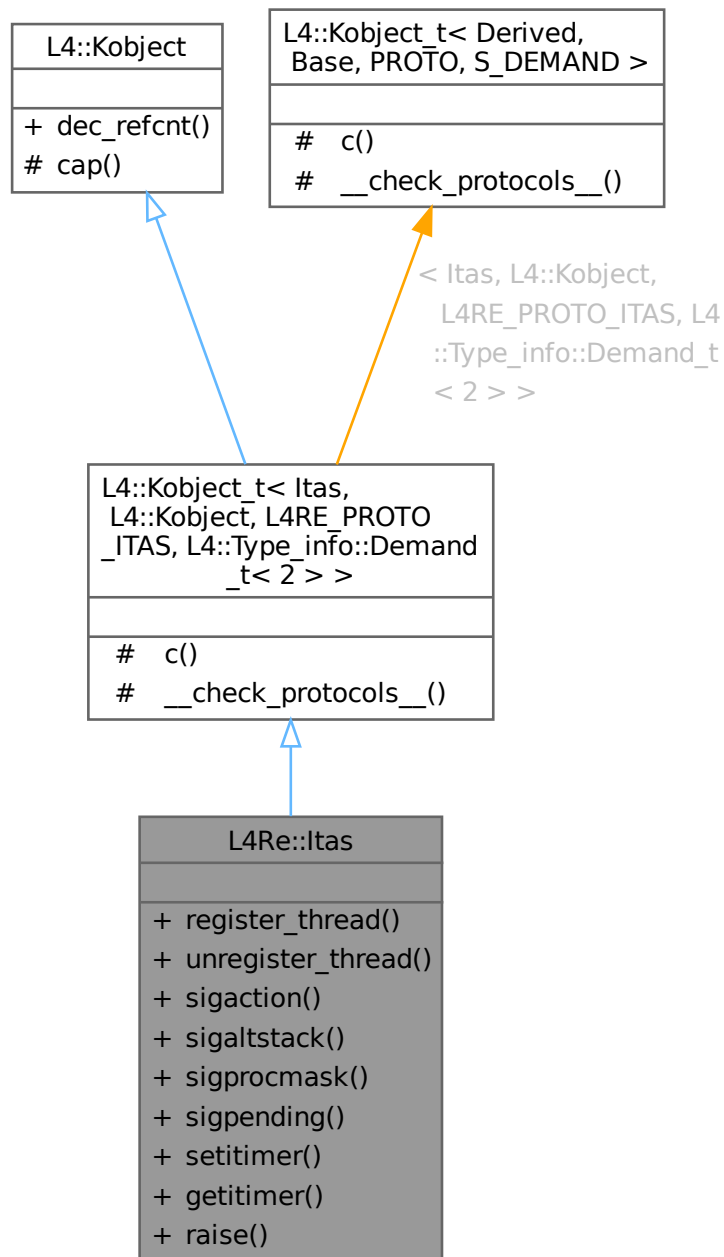
- `I4/re/inhibitor`

## 16.292 L4Re::Itas Class Reference

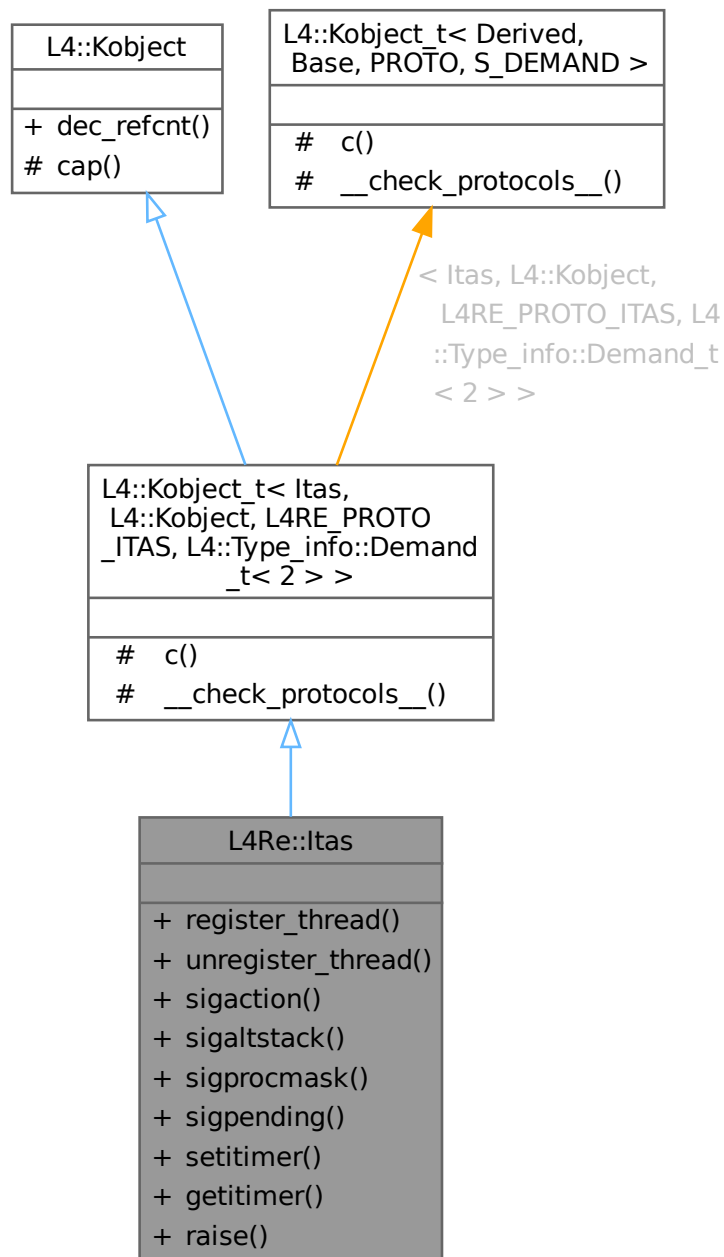
Interface to the ITAS.

```
#include <itas>
```

Inheritance diagram for L4Re::Itas:



Collaboration diagram for L4Re::Itas:



## Public Types

- enum : unsigned { `ignore_sigaction` = ~0U }

## Public Member Functions

- int `register_thread` (L4::lpc::Cap< L4::Thread > parent, L4::lpc::Cap< L4::Thread > thread\_cap, l4\_addr\_t thread\_utcb)



- *Register new thread.*
- `int unregister_thread (L4::lpc::Cap< L4::Thread > thread)`
- *Unregister a thread.*
- `int sigaction (int signum, const struct sigaction *act, struct sigaction *oldact)`
- *Examine and change a POSIX signal action.*
- `int sigaltstack (L4::lpc::Cap< L4::Thread > thread, const struct sigaltstack *ss, struct sigaltstack *oss)`
- *Examine or set alternate POSIX signal stack.*
- `int sigprocmask (L4::lpc::Cap< L4::Thread > thread, int how, sigset_t const *set, sigset_t *oldset)`
- *Examine or set process signal mask.*
- `int sigpending (L4::lpc::Cap< L4::Thread > thread, sigset_t *set)`
- *Query pending signals.*
- `int setitimer (int which, const struct itimerval *new_value, struct itimerval *old_value)`
- *Set process interval timer.*
- `int getitimer (int which, struct itimerval *curr_value)`
- *Get process interval timer.*
- `int raise (L4::lpc::Cap< L4::Thread > thread, int sig)`
- *Send a signal to the calling thread.*

## Public Member Functions inherited from L4::Kobject

- `l4_msgtag_t dec_refcnt (l4_mword_t diff, l4_utcb_t *utcb=l4_utcb())`
- *Decrement the in kernel reference counter for the object.*

## Additional Inherited Members

## Protected Types inherited from

**L4::Kobject\_t< Itas, L4::Kobject, L4RE\_PROTO\_ITAS, L4::Type\_info::Demand\_t< 2 > >**

- `typedef Itas Class`
- *The target interface type (inheriting from [Kobject\\_t](#)).*
- `typedef Typeid::Iface< PROTO, Itas > __iface`
- *The interface description for the derived class.*
- `typedef Typeid::Merge_list< Typeid::Iface_list< __iface >, typename L4::Kobject::__iface_list > __iface_list`
- *The list of all RPC interfaces provided directly or through inheritance.*

## Protected Member Functions inherited from

**L4::Kobject\_t< Itas, L4::Kobject, L4RE\_PROTO\_ITAS, L4::Type\_info::Demand\_t< 2 > >**

- `L4::Cap< Class > c () const noexcept`
- *Get the capability to ourselves.*

## Protected Member Functions inherited from L4::Kobject

- `l4_cap_idx_t cap () const noexcept`
- *Return capability selector.*

## Static Protected Member Functions inherited from

[L4::Kobject\\_t< Itas, L4::Kobject, L4RE\\_PROTO\\_ITAS, L4::Type\\_info::Demand\\_t< 2 > >](#)

- static void `__check_protocols__()` noexcept  
*Helper to check for protocol conflicts.*

### 16.292.1 Detailed Description

Interface to the ITAS.

This is an internal interface between libc and the `l4re_itas`. Do not use it. It is subject to change.

Definition at line 30 of file [itas](#).

### 16.292.2 Member Enumeration Documentation

#### 16.292.2.1 anonymous enum

anonymous enum : unsigned

#### Enumerator

Ignore_sigaction	Ignore new action of <a href="#">sigaction()</a> call.
------------------	--------------------------------------------------------

Definition at line 63 of file [itas](#).

### 16.292.3 Member Function Documentation

#### 16.292.3.1 getitimer()

```
int L4Re::Itas::getitimer (
    int which,
    struct itimerval * curr_value)
```

Get process interval timer.

See IEEE Std 1003.1-2017 [getitimer\(\)](#) for details.

#### Parameters

in	<i>which</i>	Timer type (ITIMER_REAL).
out	<i>curr_value</i>	Old timer value.

References [getitimer\(\)](#).

Referenced by [getitimer\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.292.3.2 raise()

```
int L4Re::Itas::raise (
    L4::Ipc::Cap< L4::Thread > thread,
    int sig)
```

Send a signal to the calling thread.

#### Parameters

in	<i>thread</i>	Thread cap of the calling thread.
in	<i>sig</i>	Signal that shall be raised.

References [raise\(\)](#).

Referenced by [raise\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.292.3.3 register\_thread()

```
int L4Re::Itas::register_thread (
    L4::Ipc::Cap< L4::Thread > parent,
    L4::Ipc::Cap< L4::Thread > thread_cap,
    l4_addr_t thread_utcb)
```

Register new thread.

Makes the newly created thread known to ITAS. The ITAS will do the `thread_cap->control()` to bind the thread to the task and attach the gates for pager and exception handler.

#### Parameters

<i>parent</i>	The capability of the thread that created the new thread.
<i>thread_cap</i>	The capability of the new thread.
<i>thread_utcb</i>	The address of the allocated UTCB of the new thread.

### 16.292.3.4 setitimer()

```
int L4Re::Itas::setitimer (
    int which,
    const struct itimerval * new_value,
    struct itimerval * old_value)
```

Set process interval timer.

See IEEE Std 1003.1-2017 [setitimer\(\)](#) for details.

#### Parameters

in	<i>which</i>	Timer type (ITIMER_REAL).
in	<i>new_value</i>	New timer value.
out	<i>old_value</i>	Old timer value.

Referenced by [sigpending\(\)](#).

Here is the caller graph for this function:



### 16.292.3.5 sigaction()

```
int L4Re::Itas::sigaction (
    int signum,
    const struct sigaction * act,
    struct sigaction * oldact)
```

Examine and change a POSIX signal action.

See IEEE Std 1003.1-2024 [sigaction\(\)](#) for the behaviour of the method.

If `act->sa_flags` is [Ignore\\_sigaction](#), the new action is ignored.

#### Parameters

in	<i>signum</i>	Signal number to be examined and/or modified.
in	<i>act</i>	New signal action.
out	<i>oldact</i>	Old signal action.

References [sigaction\(\)](#), and [sigaltstack\(\)](#).

Referenced by [sigaction\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.292.3.6 sigaltstack()

```
int L4Re::Itas::sigaltstack (
    L4::Ipc::Cap< L4::Thread > thread,
    const struct sigaltstack * ss,
    struct sigaltstack * oss)
```

Examine or set alternate POSIX signal stack.

See IEEE Std 1003.1-2024 [sigaltstack\(\)](#) for the behaviour of the method.

If `ss->ss_flags` is -1, the new sigaltstack will be ignored.

#### Parameters

in	<i>thread</i>	Thread cap of the thread whose sigaltstack is examined and/or modified.
in	<i>ss</i>	The new sigaltstack.
out	<i>oss</i>	The old sigaltstack.

References [sigaltstack\(\)](#), and [sigprocmask\(\)](#).

Referenced by [sigaction\(\)](#), and [sigaltstack\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.292.3.7 sigpending()

```
int L4Re::Itas::sigpending (
    L4::Ipc::Cap< L4::Thread > thread,
    sigset_t * set)
```

Query pending signals.

See IEEE Std 1003.1-2024 [sigpending\(\)](#) for details.

#### Parameters

in	<i>thread</i>	Thread cap of the thread whose pending signal are examined.
out	<i>set</i>	Pending signals of thread.

References [setitimer\(\)](#).

Referenced by [sigprocmask\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.292.3.8 sigprocmask()

```

int L4Re::Itas::sigprocmask (
    L4::Ipc::Cap< L4::Thread > thread,
    int how,
    sigset_t const * set,
    sigset_t * oldset)

```

Examine or set process signal mask.

See IEEE Std 1003.1-2024 [sigprocmask\(\)](#) for the behaviour or the method.

If `how` is `-1`, the signal mask is left unchanged.

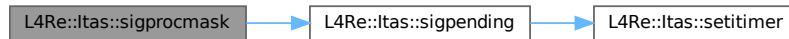
#### Parameters

in	<i>thread</i>	Thread cap of the thread whose signal mask is examined and/or modified.
in	<i>how</i>	Operation ( <code>SIG_BLOCK</code> , <code>SIG_UNBLOCK</code> , <code>SIG_SETMASK</code> or <code>-1</code> ).
in	<i>set</i>	The new signal mask.
out	<i>oldset</i>	The old signal mask.

References [sigpending\(\)](#).

Referenced by [sigaltstack\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.292.3.9 unregister\_thread()

```
int L4Re::Itas::unregister_thread (
    L4::Ipc::Cap< L4::Thread > thread)
```

Unregister a thread.

The gates for the thread's pager and exception handler will be destroyed. Thus, the thread must be destroyed after the call.

#### Parameters

<i>thread</i>	The destroyed thread.
---------------	-----------------------

References [unregister\\_thread\(\)](#).

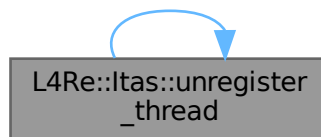
Referenced by [unregister\\_thread\(\)](#).

Here is the call graph for this function:





Here is the caller graph for this function:



The documentation for this class was generated from the following file:

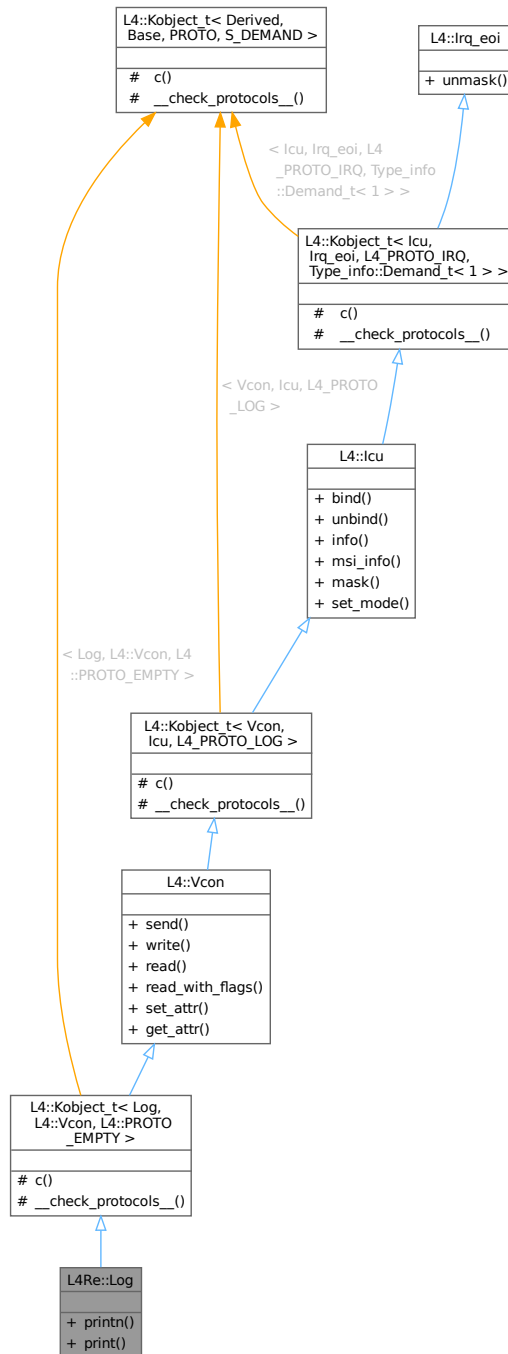
- l4/re/itas

## 16.293 L4Re::Log Class Reference

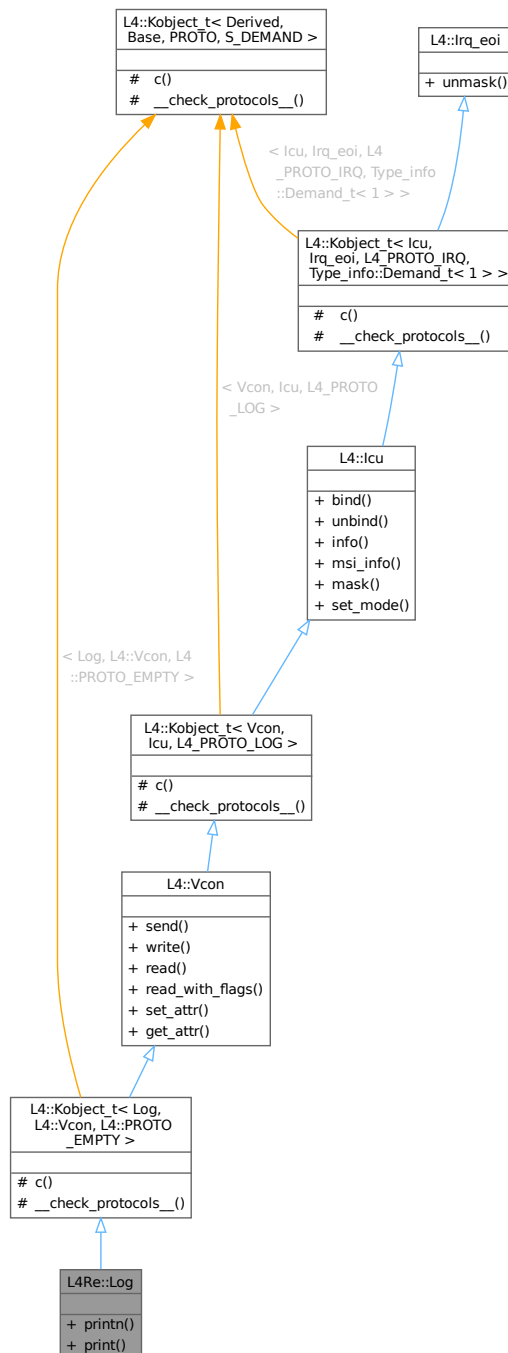
[Log](#) interface class.

```
#include <log>
```

Inheritance diagram for L4Re::Log:



Collaboration diagram for L4Re::Log:



## Public Member Functions

- void `printn` (char const \*string, int len) const noexcept  
Print string with length len, NULL characters don't matter.
- void `print` (char const \*string) const noexcept  
Print NULL-terminated string.

## Public Member Functions inherited from [L4::Vcon](#)

- [l4\\_msgtag\\_t send](#) (char const \*buf, unsigned size, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) const noexcept  
*Send data to `this` virtual console.*
- long [write](#) (char const \*buf, unsigned size, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) const noexcept  
*Write data to `this` virtual console.*
- int [read](#) (char \*buf, unsigned size, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) const noexcept  
*Read data from `this` virtual console.*
- int [read\\_with\\_flags](#) (char \*buf, unsigned size, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) const noexcept  
*Read data from `this` virtual console which also returns flags.*
- [l4\\_msgtag\\_t set\\_attr](#) ([l4\\_vcon\\_attr\\_t](#) const \*attr, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) const noexcept  
*Set the attributes of `this` virtual console.*
- [l4\\_msgtag\\_t get\\_attr](#) ([l4\\_vcon\\_attr\\_t](#) \*attr, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) const noexcept  
*Get attributes of `this` virtual console.*

## Public Member Functions inherited from [L4::Icu](#)

- [l4\\_msgtag\\_t bind](#) (unsigned irqnum, [L4::Cap< Triggerable >](#) irq, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Bind an interrupt line of an interrupt controller to an interrupt object.*
- [l4\\_msgtag\\_t unbind](#) (unsigned irqnum, [L4::Cap< Triggerable >](#) irq, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Remove binding of an interrupt line from the interrupt controller object.*
- [l4\\_msgtag\\_t info](#) ([l4\\_icu\\_info\\_t](#) \*info, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Get information about the ICU features.*
- [l4\\_msgtag\\_t msi\\_info](#) ([l4\\_umword\\_t](#) irqnum, [l4\\_uint64\\_t](#) source, [l4\\_icu\\_msi\\_info\\_t](#) \*msi\_info)  
*Get MSI info about IRQ.*
- [l4\\_msgtag\\_t mask](#) (unsigned irqnum, [l4\\_umword\\_t](#) \*label=0, [l4\\_timeout\\_t](#) to=[L4\\_IPC\\_NEVER](#), [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Mask an IRQ line.*
- [l4\\_msgtag\\_t set\\_mode](#) (unsigned irqnum, [l4\\_umword\\_t](#) mode, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Set interrupt mode.*

## Public Member Functions inherited from [L4::Irq\\_eoi](#)

- [l4\\_msgtag\\_t unmask](#) (unsigned irqnum, [l4\\_umword\\_t](#) \*label=0, [l4\\_timeout\\_t](#) to=[L4\\_IPC\\_NEVER](#), [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept  
*Unmask the given interrupt line.*

## Additional Inherited Members

## Protected Types inherited from [L4::Kobject\\_t< Log, L4::Vcon, L4::PROTO\\_EMPTY >](#)

- typedef Log **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< PROTO, Log > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename L4::Vcon::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Types inherited from [L4::Kobject\\_t< Vcon, Icu, L4\\_PROTO\\_LOG >](#)

- typedef [Vcon](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< PROTO, [Vcon](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< [\\_\\_Iface](#) >, typename Icu::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Types inherited from [L4::Kobject\\_t< Icu, Irq\\_eoi, L4\\_PROTO\\_IRQ, Type\\_info::Demand\\_t< 1 > >](#)

- typedef [Icu](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< PROTO, [Icu](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< [\\_\\_Iface](#) >, typename Irq\_eoi::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Member Functions inherited from [L4::Kobject\\_t< Log, L4::Vcon, L4::PROTO\\_EMPTY >](#)

- [L4::Cap< Class > c](#) () const noexcept  
*Get the capability to ourselves.*

## Protected Member Functions inherited from [L4::Kobject\\_t< Vcon, Icu, L4\\_PROTO\\_LOG >](#)

- [L4::Cap< Class > c](#) () const noexcept  
*Get the capability to ourselves.*

## Protected Member Functions inherited from [L4::Kobject\\_t< Icu, Irq\\_eoi, L4\\_PROTO\\_IRQ, Type\\_info::Demand\\_t< 1 > >](#)

- [L4::Cap< Class > c](#) () const noexcept  
*Get the capability to ourselves.*

## Static Protected Member Functions inherited from [L4::Kobject\\_t< Log, L4::Vcon, L4::PROTO\\_EMPTY >](#)

- static void [\\_\\_check\\_protocols](#) () noexcept  
*Helper to check for protocol conflicts.*

## Static Protected Member Functions inherited from [L4::Kobject\\_t< Vcon, Icu, L4\\_PROTO\\_LOG >](#)

- static void [\\_\\_check\\_protocols](#) () noexcept  
*Helper to check for protocol conflicts.*

## Static Protected Member Functions inherited from

[L4::Kobject\\_t< lcu, Irq\\_eoi, L4\\_PROTO\\_IRQ, Type\\_info::Demand\\_t< 1 > >](#)

- static void `__check_protocols__()` noexcept  
*Helper to check for protocol conflicts.*

### 16.293.1 Detailed Description

[Log](#) interface class.

Definition at line 33 of file [log](#).

### 16.293.2 Member Function Documentation

#### 16.293.2.1 `print()`

```
void L4Re::Log::print (
    char const * string) const    [noexcept]
```

Print NULL-terminated string.

##### Parameters

<i>string</i>	string to print
---------------	-----------------

#### 16.293.2.2 `println()`

```
void L4Re::Log::println (
    char const * string,
    int len) const    [noexcept]
```

Print string with length *len*, NULL characters don't matter.

##### Parameters

<i>string</i>	string to print
<i>len</i>	length of string

The documentation for this class was generated from the following file:

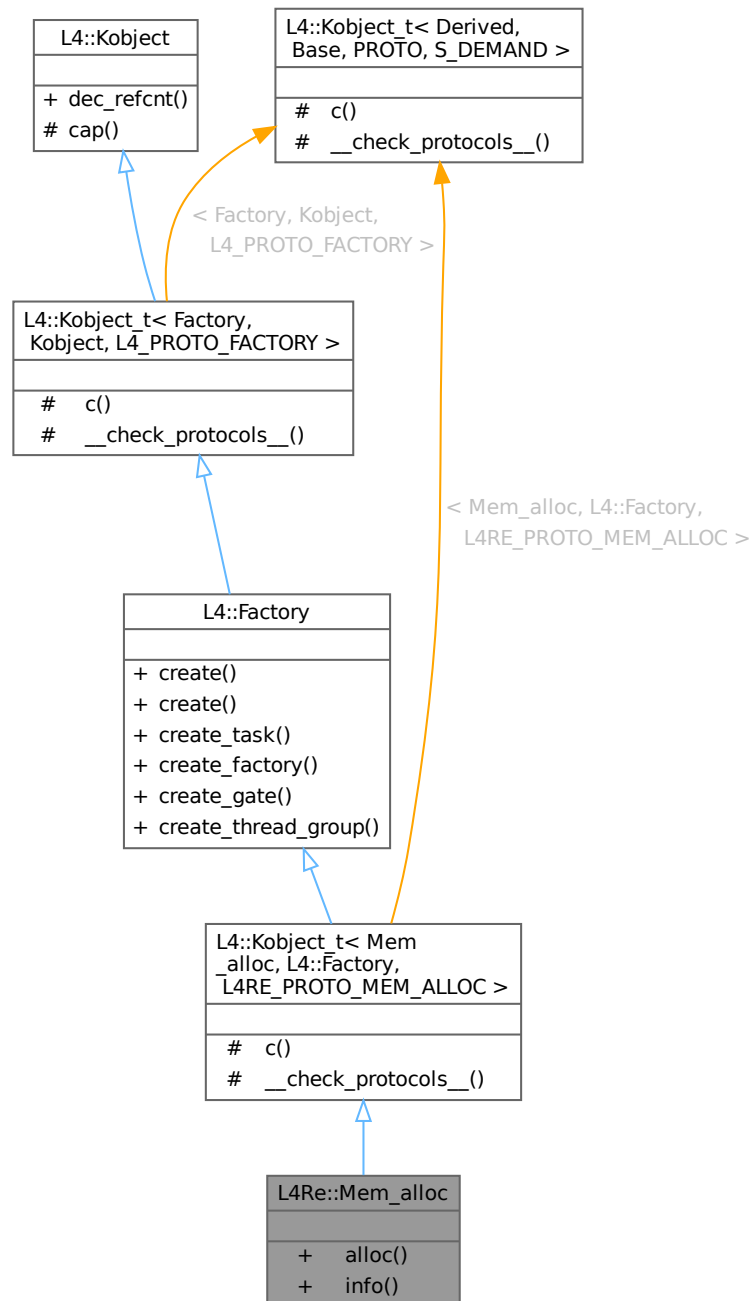
- [l4/re/log](#)

## 16.294 L4Re::Mem\_alloc Class Reference

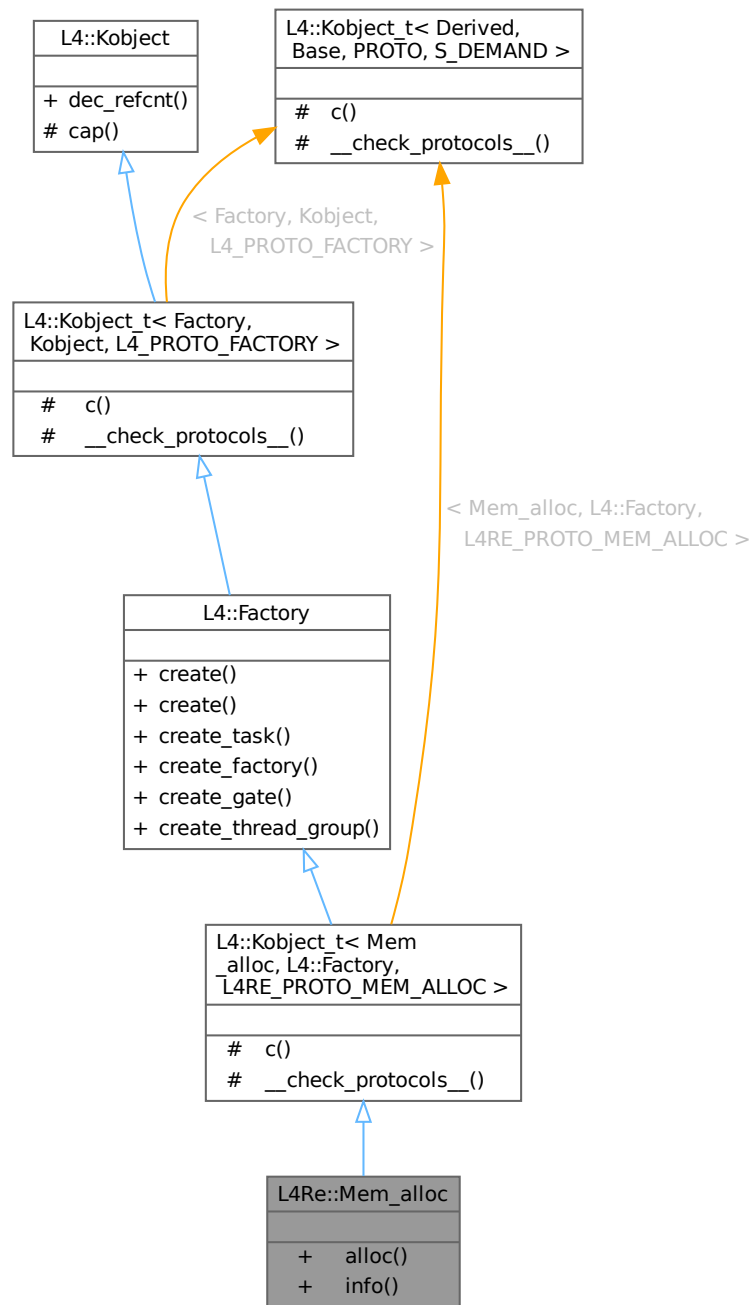
Memory allocation interface.

```
#include <mem_alloc>
```

Inheritance diagram for L4Re::Mem\_alloc:



Collaboration diagram for L4Re::Mem\_alloc:



## Data Structures

- struct [Stats](#)  
*Statistics about memory-allocator.*

## Public Types

- enum [Mem\\_alloc\\_flags](#) { [Continuous](#) = 0x01 , [Pinned](#) = 0x02 , [Super\\_pages](#) = 0x04 , [Fixed\\_paddr](#) = 0x08 }



*Flags for the allocator.*

## Public Member Functions

- long [alloc](#) (long size, [L4::Cap](#)< [Dataspace](#) > mem, unsigned long flags=0, unsigned long align=0, [l4\\_addr\\_t](#) paddr=0) const noexcept  
*Allocate anonymous memory.*
- long [info](#) ([Stats](#) &stats)  
*Get allocator information.*

## Public Member Functions inherited from [L4::Factory](#)

- [S](#) [create](#) ([Cap](#)< void > target, long obj, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Generic create call to the factory.*
- template<typename OBJ>  
[S](#) [create](#) ([Cap](#)< OBJ > target, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Create call for typed capabilities.*
- [l4\\_msgtag\\_t](#) [create\\_task](#) ([Cap](#)< [Task](#) > const &target\_cap, [l4\\_fpage\\_t](#) \*utcb\_area, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Create a new task.*
- [l4\\_msgtag\\_t](#) [create\\_factory](#) ([Cap](#)< [Factory](#) > const &target\_cap, unsigned long limit, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Create a new factory.*
- [l4\\_msgtag\\_t](#) [create\\_gate](#) ([Cap](#)< void > const &target\_cap, [Cap](#)< [Snd\\_destination](#) > const &snd\_dst\_cap, [l4\\_umword\\_t](#) label, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Create a new IPC gate, optionally bound to a send destination (a thread or thread group).*
- [l4\\_msgtag\\_t](#) [create\\_thread\\_group](#) ([Cap](#)< [Thread\\_group](#) > const &target\_cap, unsigned policy, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) noexcept  
*Create a new thread group.*

## Public Member Functions inherited from [L4::Kobject](#)

- [l4\\_msgtag\\_t](#) [dec\\_refcnt](#) ([l4\\_mword\\_t](#) diff, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)())  
*Decrement the in kernel reference counter for the object.*

## Additional Inherited Members

## Protected Types inherited from

[L4::Kobject\\_t](#)< [Mem\\_alloc](#), [L4::Factory](#), [L4RE\\_PROTO\\_MEM\\_ALLOC](#) >

- typedef [Mem\\_alloc](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef [Typeid::Iface](#)< [PROTO](#), [Mem\\_alloc](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef [Typeid::Merge\\_list](#)< [Typeid::Iface\\_list](#)< **\_\_Iface** >, typename [L4::Factory::\\_\\_Iface\\_list](#) > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Types inherited from [L4::Kobject\\_t< Factory, Kobject, L4\\_PROTO\\_FACTORY >](#)

- typedef [Factory](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< PROTO, [Factory](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename Kobject::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Member Functions inherited from [L4::Kobject\\_t< Mem\\_alloc, L4::Factory, L4RE\\_PROTO\\_MEM\\_ALLOC >](#)

- [L4::Cap< Class > c](#) () const noexcept  
*Get the capability to ourselves.*

### Protected Member Functions inherited from [L4::Kobject\\_t< Factory, Kobject, L4\\_PROTO\\_FACTORY >](#)

- [L4::Cap< Class > c](#) () const noexcept  
*Get the capability to ourselves.*

### Protected Member Functions inherited from [L4::Kobject](#)

- [l4\\_cap\\_idx\\_t cap](#) () const noexcept  
*Return capability selector.*

### Static Protected Member Functions inherited from [L4::Kobject\\_t< Mem\\_alloc, L4::Factory, L4RE\\_PROTO\\_MEM\\_ALLOC >](#)

- static void **\_\_check\_protocols** () noexcept  
*Helper to check for protocol conflicts.*

### Static Protected Member Functions inherited from [L4::Kobject\\_t< Factory, Kobject, L4\\_PROTO\\_FACTORY >](#)

- static void **\_\_check\_protocols** () noexcept  
*Helper to check for protocol conflicts.*

## 16.294.1 Detailed Description

Memory allocation interface.

The memory-allocator API is the basic API to allocate memory from the [L4Re](#) subsystem. The memory is allocated in terms of dataspace (see [L4Re::Dataspace](#)). The provided dataspace have at least the property that data written to such a dataspace is available as long as the dataspace is not freed or the data is not overwritten. In particular, the memory backing a dataspace from an allocator need not be allocated instantly, but may be allocated lazily on demand.

A memory allocator can provide dataspace with additional properties, such as physically contiguous memory, pre-allocated memory, or pinned memory. To request memory with an additional property the [L4Re::Mem\\_alloc::alloc\(\)](#) method provides a flags parameter. If the concrete implementation of a memory allocator does not support or allow allocation of memory with a certain property, the allocation may be refused.

Definition at line 52 of file [mem\\_alloc](#).

## 16.294.2 Member Enumeration Documentation

### 16.294.2.1 Mem\_alloc\_flags

enum [L4Re::Mem\\_alloc::Mem\\_alloc\\_flags](#)

Flags for the allocator.

They describe requested properties of the allocated memory. Support of these properties by the dataspace provider is optional.

#### Enumerator

Continuous	Allocate physically contiguous memory.
Pinned	Deprecated, use <a href="#">L4Re::Dma_space</a> instead.
Super_pages	Allocate super pages.
Fixed_paddr	Allocate at fixed physical address. Only honored on no-MMU systems. Will fail on MMU systems.

Definition at line 62 of file [mem\\_alloc](#).

## 16.294.3 Member Function Documentation

### 16.294.3.1 alloc()

```
long L4Re::Mem_alloc::alloc (
    long size,
    L4::Cap< Dataspace > mem,
    unsigned long flags = 0,
    unsigned long align = 0,
    l4_addr_t paddr = 0) const [noexcept]
```

Allocate anonymous memory.

#### Parameters

	<i>size</i>	Size in bytes to be requested. Allocation granularity is (super)pages, however, the allocator will store the byte-granular given size as the size of the dataspace and consecutively will use this byte-granular size for servicing the dataspace. Allocators may optionally also implement a maximum allocation strategy: if <i>size</i> is a negative value and <i>flags</i> set the <a href="#">Mem_alloc_flags::Continuous</a> bit, the allocator tries to allocate as much memory as possible leaving an amount of at least <i>-size</i> bytes within the associated quota.
out	<i>mem</i>	Capability slot where the capability to the dataspace is received.
	<i>flags</i>	Special dataspace properties, see <a href="#">Mem_alloc_flags</a>
	<i>align</i>	Log2 alignment of dataspace if supported by allocator, will be at least L4_PAGESHIFT, with Super_pages flag set at least L4_SUPERPAGESHIFT
	<i>paddr</i>	The physical address where the dataspace should be allocated if <a href="#">Mem_alloc_flags::Fixed</a> flag is set.

#### Return values

<i>0</i>	Success
<i>-L4_ERANGE</i>	Given size not supported.
<i>-L4_ENOMEM</i>	Not enough memory available.
<i>&lt;0</i>	IPC error

Definition at line 24 of file [mem\\_alloc\\_impl.h](#).

References [L4::Kobject::cap\(\)](#), [Fixed\\_paddr](#), and [l4\\_error\(\)](#).

Here is the call graph for this function:



### 16.294.3.2 info()

```
long L4Re::Mem_alloc::info (
    Stats & stats)
```

Get allocator information.

#### Parameters

out	<i>stats</i>	Allocator information
-----	--------------	-----------------------

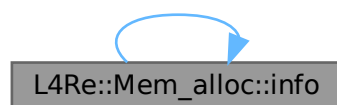
#### Return values

<i>0</i>	Success
<i>&lt;0</i>	IPC error

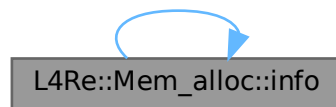
References [info\(\)](#), and [L4\\_INLINE\\_RPC](#).

Referenced by [info\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

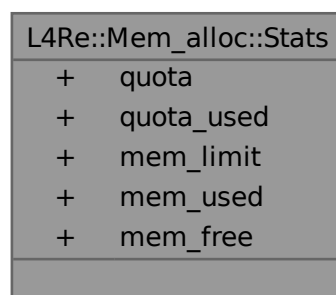
- [l4/re/mem\\_alloc](#)
- [l4/re/impl/mem\\_alloc\\_impl.h](#)

## 16.295 L4Re::Mem\_alloc::Stats Struct Reference

Statistics about memory-allocator.

```
#include <mem_alloc>
```

Collaboration diagram for L4Re::Mem\_alloc::Stats:



### Data Fields

- [l4\\_size\\_t quota](#)  
*Memory quota of this allocator.*
- [l4\\_size\\_t quota\\_used](#)  
*Amount of currently used quota of this allocator.*
- [l4\\_size\\_t mem\\_limit](#)  
*Maximum amount of memory that can be allocated by this allocator.*
- [l4\\_size\\_t mem\\_used](#)  
*Amount of currently allocated memory.*
- [l4\\_size\\_t mem\\_free](#)  
*Amount of memory that is still available for allocation.*

### 16.295.1 Detailed Description

Statistics about memory-allocator.

Definition at line 74 of file [mem\\_alloc](#).

### 16.295.2 Field Documentation

#### 16.295.2.1 `mem_free`

```
l4_size_t L4Re::Mem_alloc::Stats::mem_free
```

Amount of memory that is still available for allocation.

This field can be lower than `mem_limit` - `mem_used`. In this case the system may be over-committed and there is globally not enough memory left. Also, if the quota is already used up for sub-factories (see `quota_used`), there may be not enough quota left.

Definition at line 126 of file [mem\\_alloc](#).

#### 16.295.2.2 `mem_limit`

```
l4_size_t L4Re::Mem_alloc::Stats::mem_limit
```

Maximum amount of memory that can be allocated by this allocator.

Will never exceed the `quota` but may be smaller if the system has less memory installed.

Definition at line 102 of file [mem\\_alloc](#).

#### 16.295.2.3 `mem_used`

```
l4_size_t L4Re::Mem_alloc::Stats::mem_used
```

Amount of currently allocated memory.

This field represents the amount of memory that is in use by this allocator. It recursively includes the memory used by sub-factories, if any.

Will never exceed `mem_limit` or `quota_used`.

#### Note

Dataspaces may allocate memory lazily! As such, the field will increase only after pages have been allocated to a dataspace.

Definition at line 116 of file [mem\\_alloc](#).

#### 16.295.2.4 quota

```
l4_size_t L4Re::Mem_alloc::Stats::quota
```

Memory quota of this allocator.

Strictly limits the amount of memory that can be allocated. This may be larger than there is actual physical memory available. In particular, the root factory has an artificial quota and returns -1 in this field.

Definition at line 83 of file [mem\\_alloc](#).

#### 16.295.2.5 quota\_used

```
l4_size_t L4Re::Mem_alloc::Stats::quota_used
```

Amount of currently used quota of this allocator.

The amount of used quota is not necessarily linked to the current memory usage. See [mem\\_used](#) for this information. The quota of a factory is immediately and fully accounted to the parent factory quota.

This value may even exceed [mem\\_limit](#) if the system is over-committed.

Definition at line 94 of file [mem\\_alloc](#).

The documentation for this struct was generated from the following file:

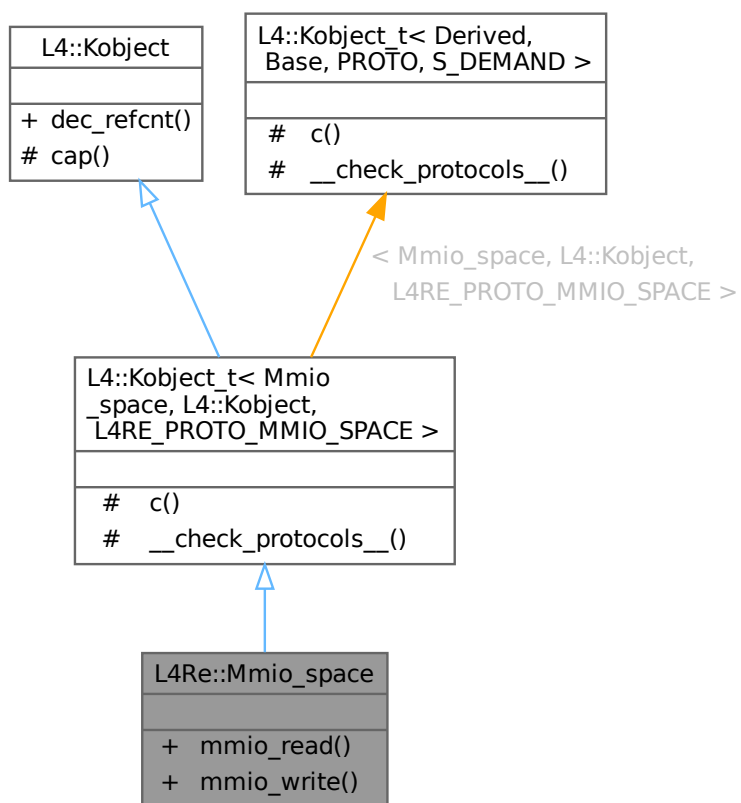
- [l4/re/mem\\_alloc](#)

## 16.296 L4Re::Mmio\_space Struct Reference

Interface for memory-like address space accessible via IPC.

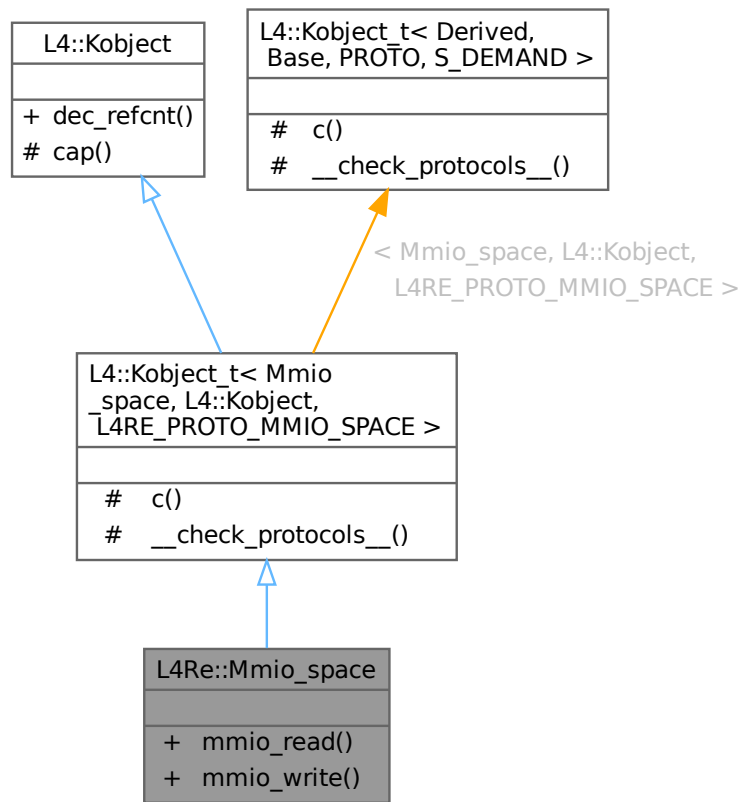
```
#include <mmio_space>
```

Inheritance diagram for L4Re::Mmio\_space:





Collaboration diagram for L4Re::Mmio\_space:



## Public Types

- enum `Access_width` { `Wd_8bit` = 0 , `Wd_16bit` = 1 , `Wd_32bit` = 2 , `Wd_64bit` = 3 }  
*Actual size of the value to read or write.*
- typedef `l4_uint64_t` `Addr`  
*Device address.*

## Public Member Functions

- long `mmio_read` (`Addr` addr, char width, `l4_uint64_t` \*value)  
*Read a value from the given address.*
- long `mmio_write` (`Addr` addr, char width, `l4_uint64_t` value)  
*Write a value to the given address.*

## Public Member Functions inherited from L4::Kobject

- `l4_msgtag_t` `dec_refcnt` (`l4_mword_t` diff, `l4_utcb_t` \*utcb=`l4_utcb()`)  
*Decrement the in kernel reference counter for the object.*

## Additional Inherited Members

### Protected Types inherited from

[L4::Kobject\\_t](#)< [Mmio\\_space](#), [L4::Kobject](#), [L4RE\\_PROTO\\_MMIO\\_SPACE](#) >

- typedef [Mmio\\_space](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< [PROTO](#), [Mmio\\_space](#) > **\_\_iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< [\\_\\_iface](#) >, typename [L4::Kobject::\\_\\_iface\\_list](#) > **\_\_iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Member Functions inherited from

[L4::Kobject\\_t](#)< [Mmio\\_space](#), [L4::Kobject](#), [L4RE\\_PROTO\\_MMIO\\_SPACE](#) >

- [L4::Cap](#)< [Class](#) > **c** () const noexcept  
*Get the capability to ourselves.*

### Protected Member Functions inherited from [L4::Kobject](#)

- [l4\\_cap\\_idx\\_t](#) **cap** () const noexcept  
*Return capability selector.*

### Static Protected Member Functions inherited from

[L4::Kobject\\_t](#)< [Mmio\\_space](#), [L4::Kobject](#), [L4RE\\_PROTO\\_MMIO\\_SPACE](#) >

- static void **\_\_check\_protocols\_\_** () noexcept  
*Helper to check for protocol conflicts.*

## 16.296.1 Detailed Description

Interface for memory-like address space accessible via IPC.

This interface defines methods for indirect access to MMIO regions.

Memory mapped IO (MMIO) is used by device drivers to control hardware devices. Access to MMIO regions is assigned to user-level device drivers via mappings of memory pages.

However, there are hardware platforms where MMIO regions for different devices share the same memory page. With respect to security and safety, it is often not allowed to map a memory page to multiple device drivers because the driver of one device could then influence operation of another device, which violates security boundaries.

A solution to that problem is to implement a third (trusted) component that gets exclusive access to the shared memory page, and that drivers can access via IPC with the [Mmio\\_space](#) protocol. This proxy-component can then enforce an access policy.

#### Include File

```
#include <l4/re/mmio_space>
```

Definition at line 45 of file [mmio\\_space](#).

## 16.296.2 Member Enumeration Documentation

### 16.296.2.1 Access\_width

enum [L4Re::Mmio\\_space::Access\\_width](#)

Actual size of the value to read or write.

#### Enumerator

---

Wd_8bit	Value is a byte.
Wd_16bit	Value is a 2-byte word.
Wd_32bit	Value is a 4-byte word.
Wd_64bit	Value is a 8-byte word.

Definition at line 49 of file [mmio\\_space](#).

## 16.296.3 Member Function Documentation

### 16.296.3.1 mmio\_read()

```
long L4Re::Mmio_space::mmio_read (
    Addr addr,
    char width,
    l4_uint64_t * value)
```

Read a value from the given address.

#### Parameters

	<i>addr</i>	Device virtual address to read from. The address must be aligned relative to the access width.
	<i>width</i>	Access width of value to be read, see <a href="#">Access_width</a> .
out	<i>value</i>	Return value. If width is smaller than 64 bit, the upper bits are guaranteed to be 0.

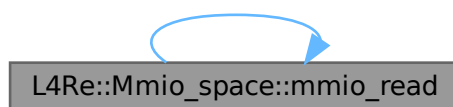
#### Return values

<a href="#">L4_EOK</a>	Success.
-L4_EPERM	Insufficient read rights.
-L4_EINVAL	Address does not exist or cannot be accessed with the given width.

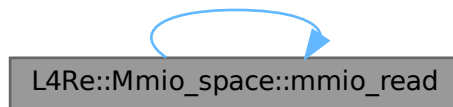
References [mmio\\_read\(\)](#).

Referenced by [mmio\\_read\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.296.3.2 mmio\_write()

```
long L4Re::Mmio_space::mmio_write (  
    Addr addr,  
    char width,  
    l4_uint64_t value)
```

Write a value to the given address.

#### Parameters

<i>addr</i>	Device virtual address to write to. The address must be aligned relative to the access width.
<i>width</i>	Access width of value to write, see <a href="#">Access_width</a> .
<i>value</i>	Value to write. If width is smaller than 64 bit, the upper bits are ignored.

#### Return values

<a href="#">L4_EOK</a>	Success.
<code>-L4_EPERM</code>	Insufficient write rights.
<code>-L4_EINVAL</code>	Address does not exist or cannot be accessed with the given width.

References [mmio\\_write\(\)](#).

Referenced by [mmio\\_write\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this struct was generated from the following file:

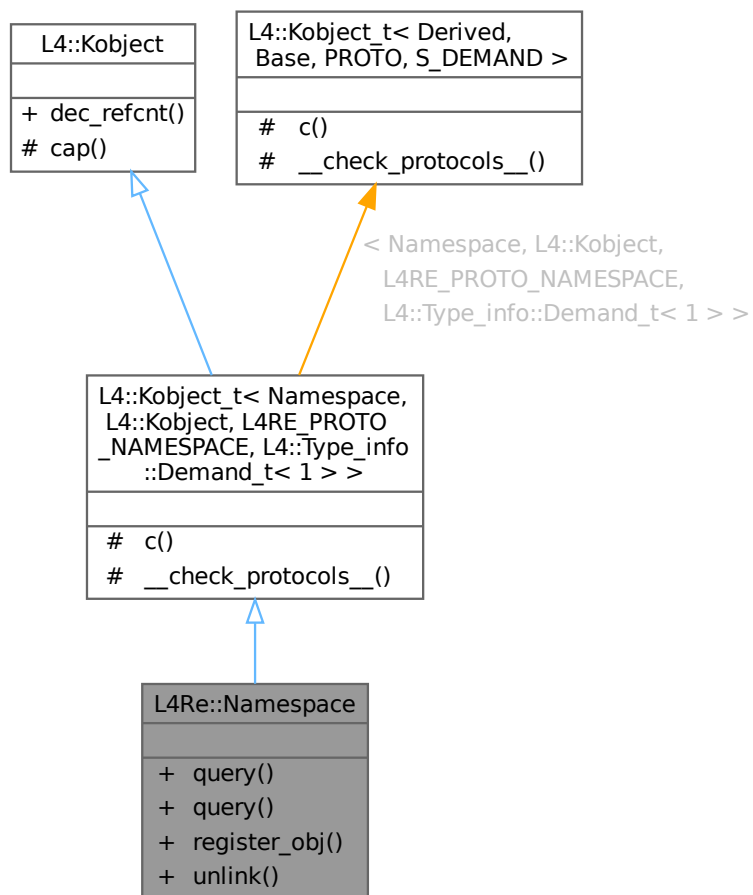
- [l4/re/mmio\\_space](#)

## 16.297 L4Re::Namespace Class Reference

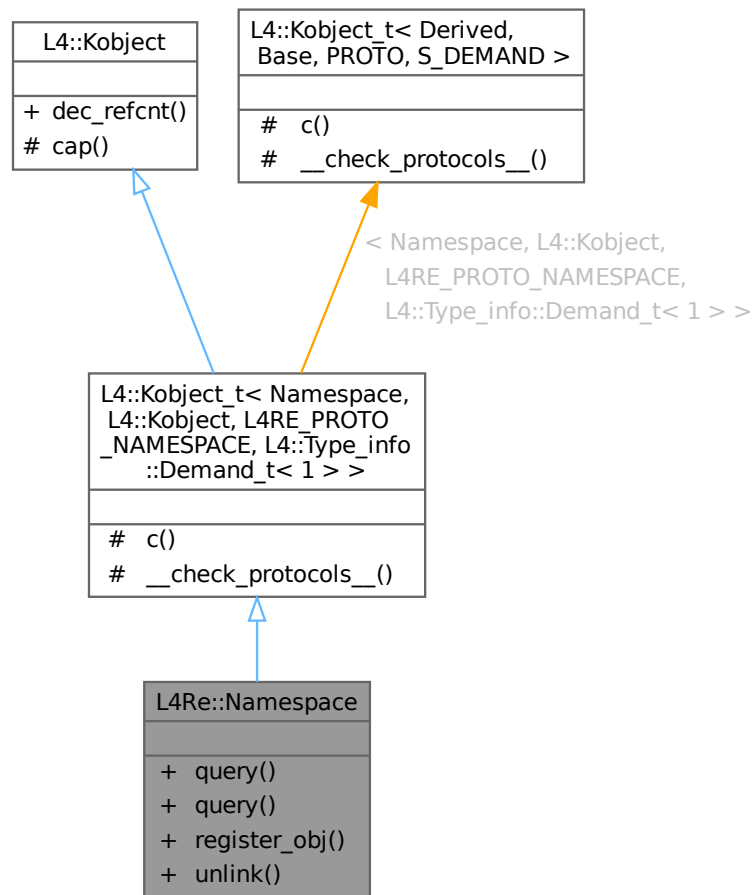
Name-space interface.

```
#include <namespace>
```

Inheritance diagram for L4Re::Namespace:



Collaboration diagram for L4Re::Namespace:



## Public Types

- enum **Register\_flags** {  
**Ro** = L4\_CAP\_FPAGE\_RO , **Rw** = L4\_CAP\_FPAGE\_RW , **Rs** = L4\_CAP\_FPAGE\_RS , **Rws** = L4\_CAP\_FPAGE\_RWS ,  
**Strong** = L4\_CAP\_FPAGE\_S , **Trusted** = 0x008 , **Cap\_flags** = Ro | Rw | Strong | Trusted , **Link** = 0x100 ,  
**Overwrite** = 0x200 }  
*Flags for registering name spaces.*
- enum **Query\_result\_flags** { **Partly\_resolved** = 0x020 }  
*Flags returned by query IPC, only used internally.*
- enum **Query\_timeout** { **To\_default** = 3600000 , **To\_non\_blocking** = 0 }  
*Timeout values for query operation.*

## Public Member Functions

- long **query** (char const \*name, L4::Cap< void > const &cap, int timeout=To\_default, l4\_umword\_t \*local\_id=0, bool iterate=true) const noexcept



*Query the name space for a named object.*

- long [query](#) (char const \*name, unsigned len, [L4::Cap](#)< void > const &cap, int timeout=[To\\_default](#), [l4\\_umword\\_t](#) \*local\_id=0, bool iterate=true) const noexcept

*Query the name space for a named object.*

- long [register\\_obj](#) (char const \*name, [L4::lpc::Cap](#)< void > obj, unsigned flags=[Rw](#)) const noexcept

*Register an object with a name.*

- long [unlink](#) (char const \*name)

*Remove an entry from the name space.*

## Public Member Functions inherited from [L4::Kobject](#)

- [l4\\_msgtag\\_t dec\\_refcnt](#) ([l4\\_mword\\_t](#) diff, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)())

*Decrement the in kernel reference counter for the object.*

## Additional Inherited Members

## Protected Types inherited from

[L4::Kobject\\_t](#)< [Namespace](#), [L4::Kobject](#), [L4RE\\_PROTO\\_NAMESPACE](#), [L4::Type\\_info::Demand\\_t](#)< 1 >

- typedef [Namespace](#) **Class**

*The target interface type (inheriting from [Kobject\\_t](#)).*

- typedef [Typeid::Iface](#)< [PROTO](#), [Namespace](#) > **\_\_iface**

*The interface description for the derived class.*

- typedef [Typeid::Merge\\_list](#)< [Typeid::Iface\\_list](#)< **\_\_iface** >, typename [L4::Kobject::\\_\\_iface\\_list](#) > **\_\_iface\_list**

*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Member Functions inherited from

[L4::Kobject\\_t](#)< [Namespace](#), [L4::Kobject](#), [L4RE\\_PROTO\\_NAMESPACE](#), [L4::Type\\_info::Demand\\_t](#)< 1 >

- [L4::Cap](#)< [Class](#) > **c** () const noexcept

*Get the capability to ourselves.*

## Protected Member Functions inherited from [L4::Kobject](#)

- [l4\\_cap\\_idx\\_t cap](#) () const noexcept

*Return capability selector.*

## Static Protected Member Functions inherited from

[L4::Kobject\\_t](#)< [Namespace](#), [L4::Kobject](#), [L4RE\\_PROTO\\_NAMESPACE](#), [L4::Type\\_info::Demand\\_t](#)< 1 >

- static void **\_\_check\_protocols** () noexcept

*Helper to check for protocol conflicts.*

### 16.297.1 Detailed Description

Name-space interface.

All name space objects must provide this interface. However, it is not mandatory that a name space object allows to register new capabilities.

The name lookup is done iteratively, this means the hierarchical names are resolved component wise by the client itself.

Definition at line 49 of file [namespace](#).

### 16.297.2 Member Enumeration Documentation

#### 16.297.2.1 Query\_result\_flags

```
enum L4Re::Namespace::Query_result_flags
```

Flags returned by query IPC, only used internally.

##### Enumerator

Partly_resolved	Name was only partly resolved.
-----------------	--------------------------------

Definition at line 77 of file [namespace](#).

#### 16.297.2.2 Query\_timeout

```
enum L4Re::Namespace::Query_timeout
```

Timeout values for query operation.

##### Enumerator

To_default	Default timeout.
To_non_blocking	Expect callee to answer immediately.

Definition at line 83 of file [namespace](#).

#### 16.297.2.3 Register\_flags

```
enum L4Re::Namespace::Register_flags
```

Flags for registering name spaces.

##### Enumerator

---

Ro	Read-only.
Rw	Read-write.
Rs	Read-only + strong.
Rws	Read-write + strong.
Strong	Strong.
Trusted	Obsolete, do not use.
Link	Obsolete, do not use.
Overwrite	If entry already exists, overwrite it.

Definition at line 57 of file [namespace](#).

## 16.297.3 Member Function Documentation

### 16.297.3.1 query() [1/2]

```
long L4Re::Namespace::query (
    char const * name,
    L4::Cap< void > const & cap,
    int timeout = To_default,
    l4_umword_t * local_id = 0,
    bool iterate = true) const [noexcept]
```

Query the name space for a named object.

#### Parameters

in	<i>name</i>	String to query (without any leading slashes).
out	<i>cap</i>	Capability slot where the received capability will be put.
in	<i>timeout</i>	Timeout of query in milliseconds. The client will only wait if a name has already been registered with the server but no object has yet been attached.
out	<i>local_id</i>	If given, <a href="#">L4_RCV_ITEM_LOCAL_ID</a> will be set for the IPC from the name space, so that if the capability that was received is a local item, the capability ID will be returned with this parameter.
in	<i>iterate</i>	If true, the client will try to resolve names by iteratively calling the name spaces until the name is fully resolved.

#### Return values

0	Name could be fully resolved.
>0	Name could only be partly resolved. The number of remaining characters is returned.
-L4_ENOENT	Entry could not be found.
-L4_EAGAIN	Entry exists but no object is yet attached. Try again later.
<0	IPC errors, see <a href="#">l4_error_code_t</a> .

Definition at line 114 of file [namespace\\_impl.h](#).

References [query\(\)](#).

Referenced by [query\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.297.3.2 `query()` [2/2]

```

long L4Re::Namespace::query (
    char const * name,
    unsigned len,
    L4::Cap< void > const & cap,
    int timeout = To_default,
    l4_umword_t * local_id = 0,
    bool iterate = true) const [noexcept]
  
```

Query the name space for a named object.

The query string does not necessarily need to be null-terminated.

#### Parameters

in	<i>len</i>	Length of the string to query without any terminating null characters.
in	<i>name</i>	String to query (without any leading slashes).
out	<i>cap</i>	Capability slot where the received capability will be put.

in	<i>timeout</i>	Timeout of query in milliseconds. The client will only wait if a name has already been registered with the server but no object has yet been attached.
out	<i>local_id</i>	If given, <a href="#">L4_RCV_ITEM_LOCAL_ID</a> will be set for the IPC from the name space, so that if the capability that was received is a local item, the capability ID will be returned with this parameter.
in	<i>iterate</i>	If true, the client will try to resolve names by iteratively calling the name spaces until the name is fully resolved.

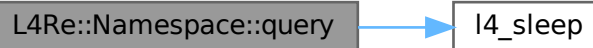
### Return values

<i>0</i>	Name could be fully resolved.
<i>&gt;0</i>	Name could only be partly resolved. The number of remaining characters is returned.
<i>-L4_ENOENT</i>	Entry could not be found.
<i>-L4_EAGAIN</i>	Entry exists but no object is yet attached. Try again later.
<i>&lt;0</i>	IPC errors, see <a href="#">l4_error_code_t</a> .

Definition at line 66 of file [namespace\\_impl.h](#).

References [L4\\_EAGAIN](#), [L4\\_EINVAL](#), [l4\\_sleep\(\)](#), and [L4\\_UNLIKELY](#).

Here is the call graph for this function:



### 16.297.3.3 register\_obj()

```

long L4Re::Namespace::register_obj (
    char const * name,
    L4::Ipc::Cap< void > obj,
    unsigned flags = Rw) const [inline], [noexcept]
  
```

Register an object with a name.

### Parameters

<i>name</i>	Name under which the object should be registered.
<i>obj</i>	Capability to object to register. An invalid capability may be given to only reserve the name for later use.

<i>flags</i>	Flags to assign to the entry, see <a href="#">L4Re::Namespace::Register_flags</a> . Note that the rights that are assigned to a capability are not only determined by the rights given in these flags but also by the rights with which the <code>obj</code> capability was mapped to the name space.
--------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Return values

<i>0</i>	Object was successfully registered with <i>name</i> .
<i>-L4_EEXIST</i>	Name already registered.
<i>-L4_EPERM</i>	Insufficient permissions; see precondition.
<i>-L4_ENOMEM</i>	Server has insufficient resources.
<i>-L4_EINVAL</i>	Invalid parameter.
<i>&lt;0</i>	IPC errors, see <a href="#">l4_error_code_t</a> .

### Precondition

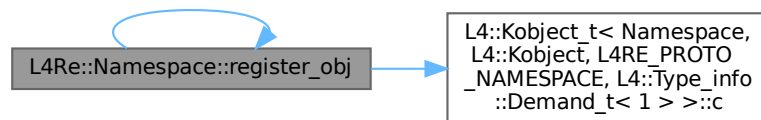
The invoked [Namespace](#) capability must have the permission [L4\\_CAP\\_FPAGE\\_W](#).

Definition at line 165 of file [namespace](#).

References [L4::Kobject\\_t< Namespace, L4::Kobject, L4RE\\_PROTO\\_NAMESPACE, L4::Type\\_info::Demand\\_t< 1 > >::c\(\)](#), [register\\_obj\(\)](#), and [Rw](#).

Referenced by [register\\_obj\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.297.3.4 unlink()

```
long L4Re::Namespace::unlink (
    char const * name) [inline]
```

Remove an entry from the name space.

#### Parameters

<i>name</i>	Name of the entry to remove.
-------------	------------------------------

#### Return values

0	Entry successfully removed.
-L4_ENOENT	Given name does not exist.
-L4_EPERM	Insufficient permissions; see precondition.
-L4_EACCESS	Name cannot be removed.
<0	IPC errors, see <a href="#">l4_error_code_t</a> .

#### Precondition

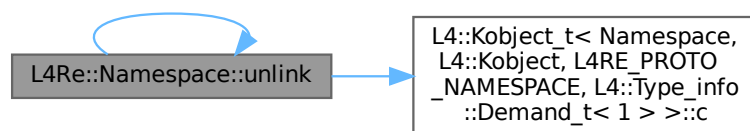
The invoked [Namespace](#) capability must have the permission [L4\\_CAP\\_FPAGE\\_W](#).

Definition at line 191 of file [namespace](#).

References [L4::Kobject\\_t< Namespace, L4::Kobject, L4RE\\_PROTO\\_NAMESPACE, L4::Type\\_info::Demand\\_t< 1 > >::c\(\)](#), and [unlink\(\)](#).

Referenced by [unlink\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

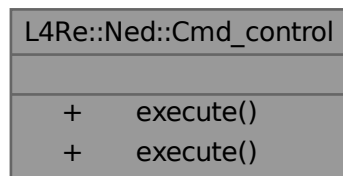
- [l4/re/namespace](#)
- [l4/re/impl/namespace\\_impl.h](#)

## 16.298 L4Re::Ned::Cmd\_control Class Reference

Direct control interface for Ned.

```
#include <cmd_control>
```

Collaboration diagram for L4Re::Ned::Cmd\_control:



### Public Member Functions

- long [execute](#) (L4::lpc::String<> cmd) noexcept  
*Execute the given Lua code.*
- long [execute](#) (L4::lpc::String<> cmd, L4::lpc::String< char > \*result) noexcept  
*Execute the given Lua code.*

### 16.298.1 Detailed Description

Direct control interface for Ned.

Definition at line 19 of file [cmd\\_control](#).

### 16.298.2 Member Function Documentation

#### 16.298.2.1 execute() [1/2]

```
long L4Re::Ned::Cmd_control::execute (
    L4::lpc::String<> cmd) [inline], [noexcept]
```

Execute the given Lua code.

#### Parameters

---



in	<i>cmd</i>	String with Lua code to execute.
----	------------	----------------------------------

**Return values**

<i>L4_EOK</i>	Code was successfully executed.
<i>-L4_EINVAL</i>	Code could not be parsed.
<i>-L4_EIO</i>	Error during code execution.

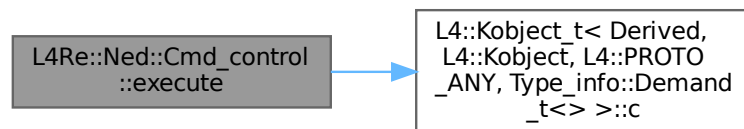
The code is executed using the global Lua state of ned which is retained between successive calls to execute. Thus you may define data in one call to execute and use it in a subsequent call.

This function does not return any results from the execution of the Lua code itself.

Definition at line 42 of file [cmd\\_control](#).

References [L4::Kobject\\_t< Derived, L4::Kobject, L4::PROTO\\_ANY, Type\\_info::Demand\\_t<> >::c\(\)](#).

Here is the call graph for this function:

**16.298.2.2 execute() [2/2]**

```

long L4Re::Ned::Cmd_control::execute (
    L4::Ipc::String<> cmd,
    L4::Ipc::String< char > * result) [inline], [noexcept]
  
```

Execute the given Lua code.

**Parameters**

in	<i>cmd</i>	String with Lua code to execute.
out	<i>result</i>	The first return value of the Lua code block as string.

**Return values**

<i>L4_EOK</i>	Code was successfully executed.
---------------	---------------------------------

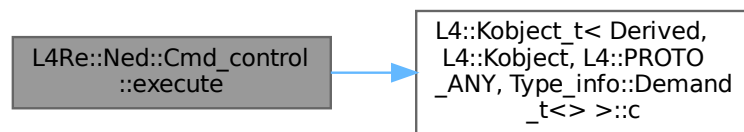
-L4_EINVAL	Code could not be parsed.
-L4_EIO	Error during code execution.

The code is executed using the global Lua state of ned which is retained between successive calls to execute. Thus you may define data in one call to execute and use it in a subsequent call.

Definition at line 64 of file [cmd\\_control](#).

References [L4::Kobject\\_t< Derived, L4::Kobject, L4::PROTO\\_ANY, Type\\_info::Demand\\_t<> >::c\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

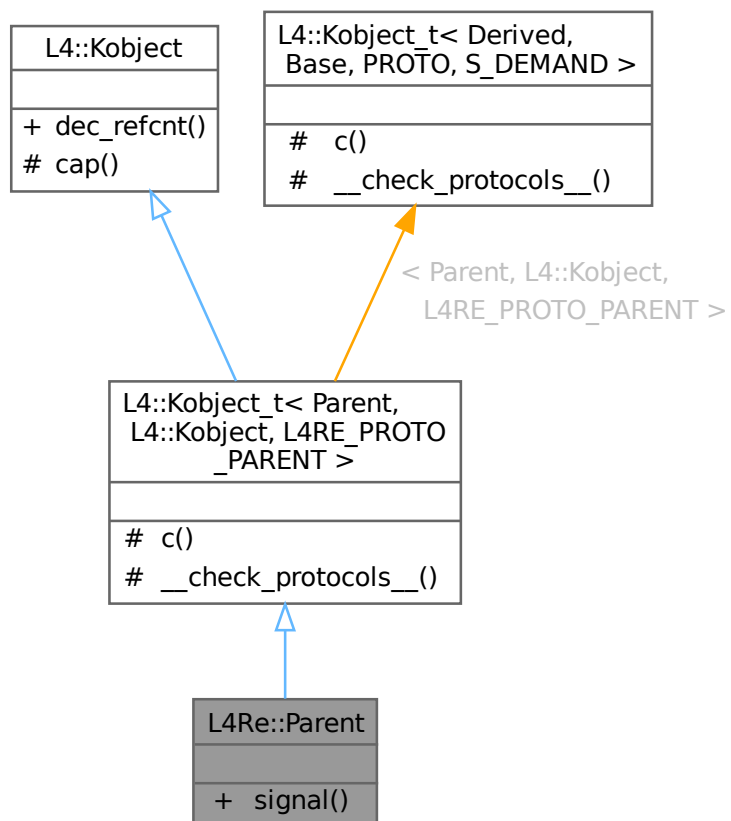
- `pkg/l4re-core/ned/lib/include/cmd_control`

## 16.299 L4Re::Parent Class Reference

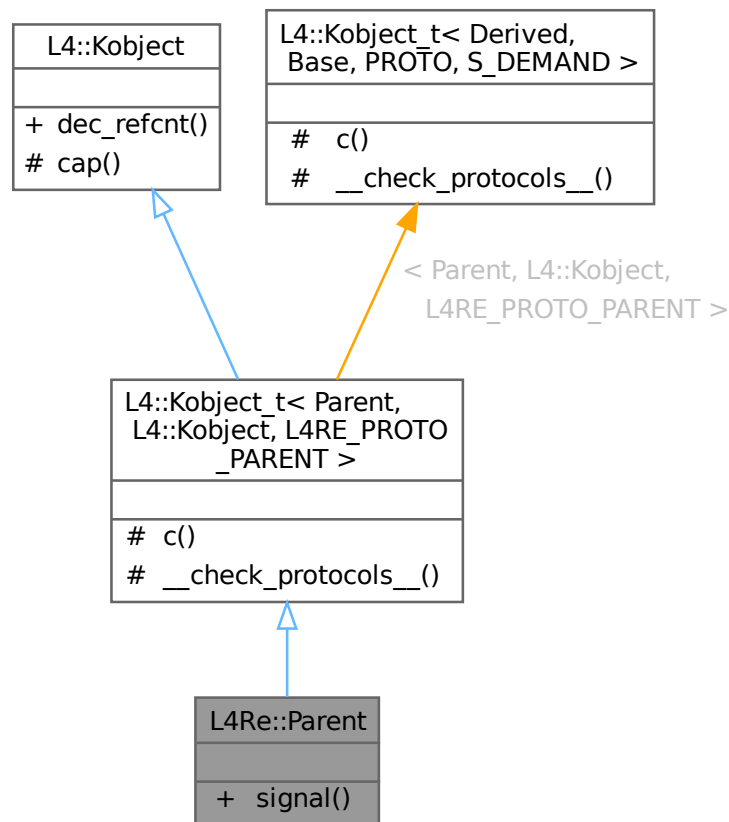
[Parent](#) interface.

```
#include <parent>
```

Inheritance diagram for L4Re::Parent:



Collaboration diagram for L4Re::Parent:



### Public Member Functions

- long [signal](#) (unsigned long sig, unsigned long val)  
*Send a signal to the parent.*

### Public Member Functions inherited from [L4::Kobject](#)

- [l4\\_msgtag\\_t dec\\_refcnt](#) ([l4\\_mword\\_t](#) diff, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#))  
*Decrement the in kernel reference counter for the object.*

### Additional Inherited Members

### Protected Types inherited from

[L4::Kobject\\_t < Parent, L4::Kobject, L4RE\\_PROTO\\_PARENT >](#)

- typedef Parent **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< PROTO, Parent > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< [\\_\\_Iface](#) >, typename L4::Kobject::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Member Functions inherited from [L4::Kobject\\_t< Parent, L4::Kobject, L4RE\\_PROTO\\_PARENT >](#)

- [L4::Cap< Class > c \(\)](#) const noexcept  
*Get the capability to ourselves.*

## Protected Member Functions inherited from [L4::Kobject](#)

- [l4\\_cap\\_idx\\_t cap \(\)](#) const noexcept  
*Return capability selector.*

## Static Protected Member Functions inherited from [L4::Kobject\\_t< Parent, L4::Kobject, L4RE\\_PROTO\\_PARENT >](#)

- static void [\\_\\_check\\_protocols\\_\\_ \(\)](#) noexcept  
*Helper to check for protocol conflicts.*

### 16.299.1 Detailed Description

[Parent](#) interface.

See also

[Parent API](#) for more details about the purpose.

Definition at line 42 of file [parent](#).

### 16.299.2 Member Function Documentation

#### 16.299.2.1 [signal\(\)](#)

```
long L4Re::Parent::signal (
    unsigned long sig,
    unsigned long val)
```

Send a signal to the parent.

#### Parameters

<i>sig</i>	Signal to send
<i>val</i>	Value of the signal

#### Return values

---

0	Success
<0	IPC error

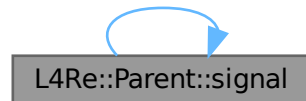
#### Note

The implementations of this interface in Moe and Ned only recognize the signal 0, in which case they will terminate the application from which the interface was invoked and not return. In this case, `val` is treated as the application's return code. For any other value of `sig`, the method just returns successfully.

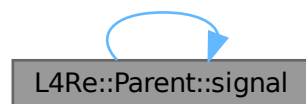
References [signal\(\)](#).

Referenced by [signal\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

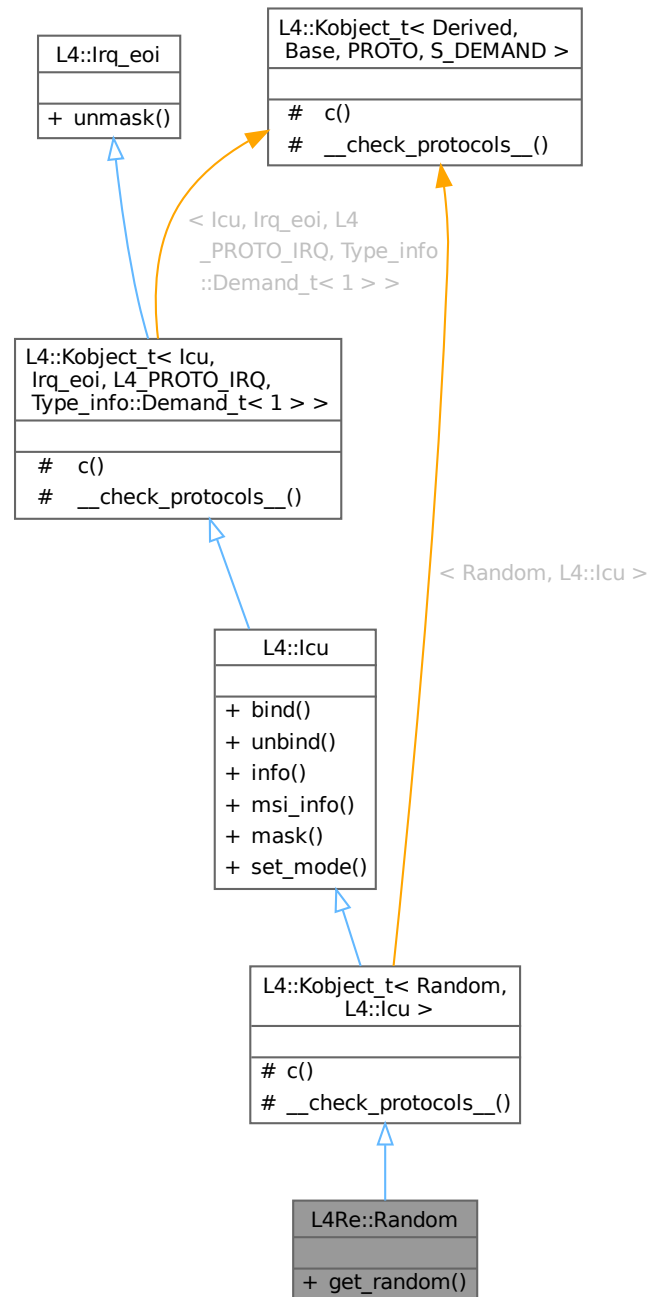
- [l4/re/parent](#)

## 16.300 L4Re::Random Struct Reference

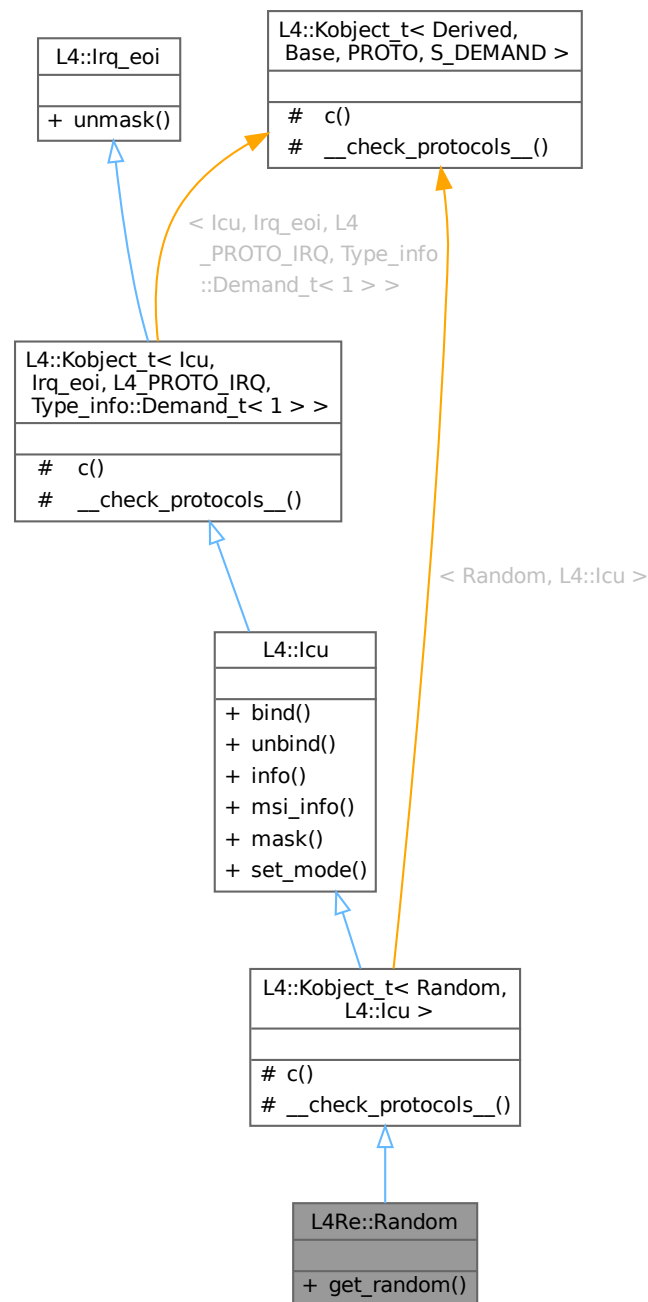
Low-bandwidth interface for random number generators.

```
#include <random>
```

Inheritance diagram for L4Re::Random:



Collaboration diagram for L4Re::Random:



### Public Member Functions

- long `get_random` (`l4_size_t` size, `L4::lpc::Array< char, unsigned long > *buffer`)  
*Get a random number.*

### Public Member Functions inherited from L4::lcu

- `l4_msgtag_t bind` (unsigned irqnum, `L4::Cap< Triggerable > irq`, `l4_utcb_t *utcb=l4_utcb()`) noexcept



*Bind an interrupt line of an interrupt controller to an interrupt object.*

- [l4\\_msgtag\\_t unbind](#) (unsigned irqnum, [L4::Cap](#)< [Triggerable](#) > irq, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept

*Remove binding of an interrupt line from the interrupt controller object.*

- [l4\\_msgtag\\_t info](#) ([l4\\_icu\\_info\\_t](#) \*info, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept

*Get information about the ICU features.*

- [l4\\_msgtag\\_t msi\\_info](#) ([l4\\_umword\\_t](#) irqnum, [l4\\_uint64\\_t](#) source, [l4\\_icu\\_msi\\_info\\_t](#) \*msi\_info)

*Get MSI info about IRQ.*

- [l4\\_msgtag\\_t mask](#) (unsigned irqnum, [l4\\_umword\\_t](#) \*label=0, [l4\\_timeout\\_t](#) to=[L4\\_IPC\\_NEVER](#), [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept

*Mask an IRQ line.*

- [l4\\_msgtag\\_t set\\_mode](#) (unsigned irqnum, [l4\\_umword\\_t](#) mode, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept

*Set interrupt mode.*

## Public Member Functions inherited from [L4::lrq\\_eoi](#)

- [l4\\_msgtag\\_t unmask](#) (unsigned irqnum, [l4\\_umword\\_t](#) \*label=0, [l4\\_timeout\\_t](#) to=[L4\\_IPC\\_NEVER](#), [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) noexcept

*Unmask the given interrupt line.*

## Additional Inherited Members

## Protected Types inherited from [L4::Kobject\\_t](#)< [Random](#), [L4::lcu](#) >

- typedef [Random](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< [PROTO\\_ANY](#), [Random](#) > **\_\_iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_iface** >, typename [L4::lcu](#)::\_\_iface\_list > **\_\_iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Types inherited from [L4::Kobject\\_t](#)< [lcu](#), [lrq\\_eoi](#), [L4\\_PROTO\\_IRQ](#), [Type\\_info::Demand\\_t](#)< 1 > >

- typedef [lcu](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< [PROTO](#), [lcu](#) > **\_\_iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_iface** >, typename [lrq\\_eoi](#)::\_\_iface\_list > **\_\_iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Member Functions inherited from [L4::Kobject\\_t](#)< [Random](#), [L4::lcu](#) >

- [L4::Cap](#)< [Class](#) > **c** () const noexcept  
*Get the capability to ourselves.*

### Protected Member Functions inherited from

[L4::Kobject\\_t< lcu, Irq\\_eoi, L4\\_PROTO\\_IRQ, Type\\_info::Demand\\_t< 1 > >](#)

- [L4::Cap< Class > c \(\)](#) const noexcept

*Get the capability to ourselves.*

### Static Protected Member Functions inherited from [L4::Kobject\\_t< Random, L4::lcu >](#)

- static void [\\_\\_check\\_protocols\\_\\_ \(\)](#) noexcept

*Helper to check for protocol conflicts.*

### Static Protected Member Functions inherited from

[L4::Kobject\\_t< lcu, Irq\\_eoi, L4\\_PROTO\\_IRQ, Type\\_info::Demand\\_t< 1 > >](#)

- static void [\\_\\_check\\_protocols\\_\\_ \(\)](#) noexcept

*Helper to check for protocol conflicts.*

## 16.300.1 Detailed Description

Low-bandwidth interface for random number generators.

The interface offers an ICU interface where a client can register an interrupt to get notified when entropy is available. Support for notifications is optional. If a service does not implement notification, it must return 0 for the number of interrupts in the [info\(\)](#) call. The notification interrupt must have index 0.

#### Include File

```
#include <l4/re/random>
```

Definition at line 33 of file [random](#).

## 16.300.2 Member Function Documentation

### 16.300.2.1 [get\\_random\(\)](#)

```
long L4Re::Random::get_random (
    l4_size_t size,
    L4::Ipc::Array< char, unsigned long > * buffer)
```

Get a random number.

#### Parameters

	<i>size</i>	Number of bytes of entropy requested.
out	<i>buffer</i>	<a href="#">Buffer</a> containing the random number. Each byte in the buffer contains 8 bits of randomness.

#### Return values

$\geq 0$	Actual size of the returned random number in bytes. This may be less than the requested size. The return value may also be 0 if temporarily no entropy is available.
$-L4\_EIO$	Source of randomness permanently unavailable.
$< 0$	IPC error.

This function should never block. It should immediately return as much entropy as is available. If the call returns less than the requested bytes and a notification interrupt was installed, then the service triggers an interrupt as soon as the remaining entropy is available. That means that when an interrupt is triggered, the service must guarantee that the next call to [get\\_random\(\)](#) returns at least the number of missing bytes for the call that initially triggered the notification.

If [get\\_random\(\)](#) is called while a notification is pending, then the behaviour is implementation-defined.

The documentation for this struct was generated from the following file:

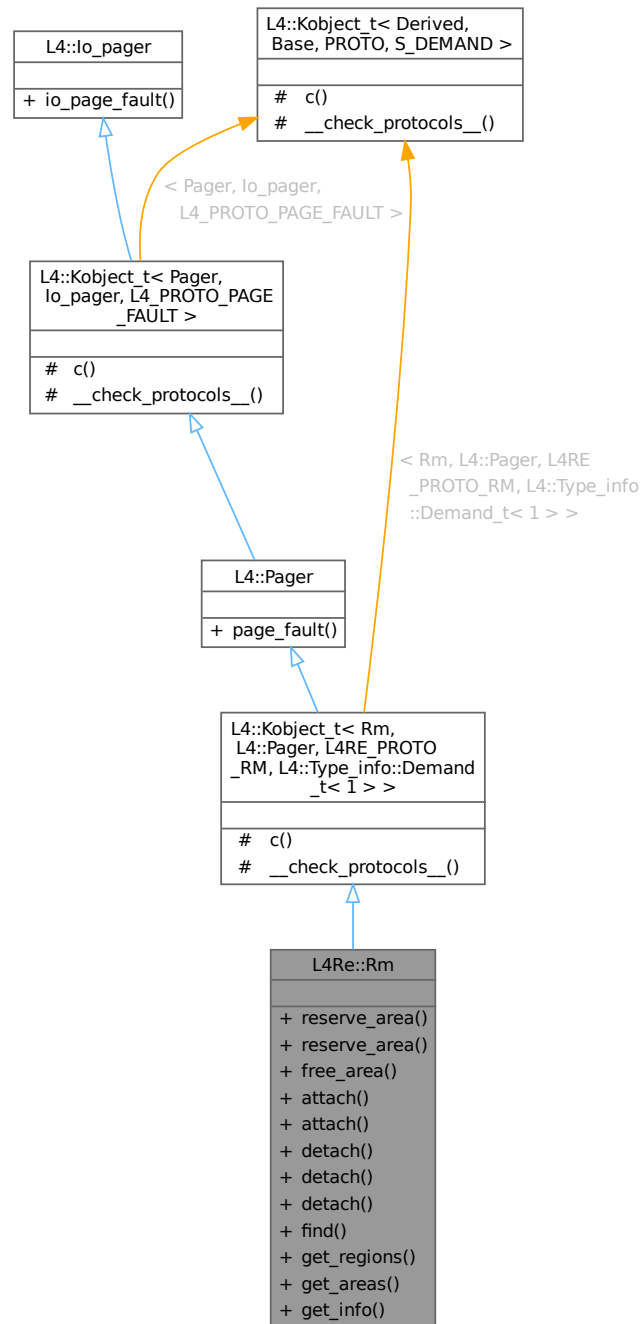
- [l4/re/random](#)

## 16.301 L4Re::Rm Class Reference

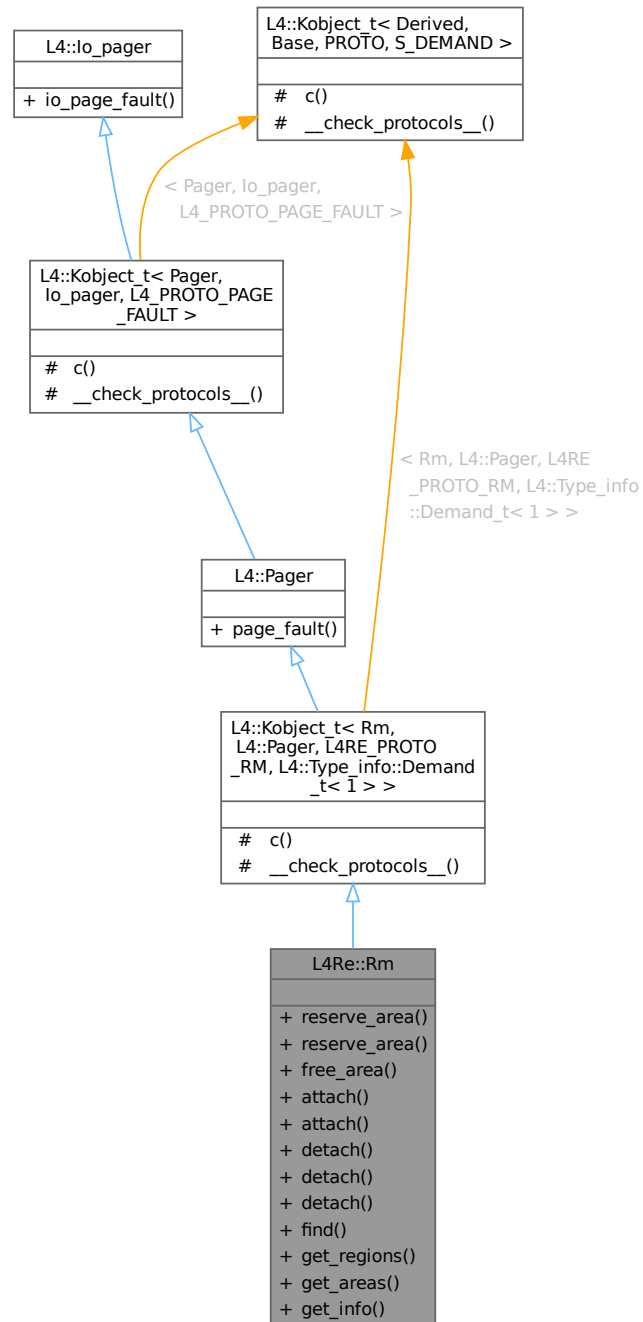
[Region](#) map.

```
#include <l4/re/rm>
```

Inheritance diagram for L4Re::Rm:



Collaboration diagram for L4Re::Rm:



## Data Structures

- struct [F](#)  
*Rm flags definitions.*
- class [Unique\\_region](#)  
*Unique region.*
- struct [Region](#)

*A region is a range of virtual addresses which is backed by content.*

- struct [Area](#)

*An area is a range of virtual addresses which is reserved, see [L4Re::Rm::reserve\\_area\(\)](#).*

## Public Types

- enum [Detach\\_result](#) { [Detached\\_ds](#) = 0 , [Kept\\_ds](#) = 1 , [Split\\_ds](#) = 2 , [Detach\\_result\\_mask](#) = 3 , [Detach\\_again](#) = 4 }
- Result values for detach operation.*
- enum [Region\\_flag\\_shifts](#) { [Caching\\_shift](#) = Dataspace::F::Caching\_shift }
- Region flag shifts.*
- enum [Detach\\_flags](#) { [Detach\\_exact](#) = 1 , [Detach\\_overlap](#) = 2 , [Detach\\_keep](#) = 4 }
- Flags for detach operation.*

## Public Member Functions

- long [reserve\\_area](#) ([l4\\_addr\\_t](#) \*start, unsigned long size, Flags flags=[Flags\(0\)](#), unsigned char align=[L4\\_PAGESHIFT](#)) const noexcept
- Reserve the given area in the region map.*
- template<typename T>  
long [reserve\\_area](#) (T \*\*start, unsigned long size, Flags flags=[Flags\(0\)](#), unsigned char align=[L4\\_PAGESHIFT](#)) const noexcept
- Reserve the given area in the region map.*
- long [free\\_area](#) ([l4\\_addr\\_t](#) addr)
- Free an area from the region map.*
- long [attach](#) ([l4\\_addr\\_t](#) \*start, unsigned long size, Flags flags, [L4::lpc::Cap](#)< [Dataspace](#) > mem, Offset offs=0, unsigned char align=[L4\\_PAGESHIFT](#), [L4::Cap](#)< [L4::Task](#) > const task=[L4::Cap](#)< [L4::Task](#) >::Invalid, char const \*name=nullptr, Offset backing\_offset=0) const noexcept
- Attach a data space to a region.*
- template<typename T>  
long [attach](#) (T \*\*start, unsigned long size, Flags flags, [L4::lpc::Cap](#)< [Dataspace](#) > mem, Offset offs=0, unsigned char align=[L4\\_PAGESHIFT](#), [L4::Cap](#)< [L4::Task](#) > const task=[L4::Cap](#)< [L4::Task](#) >::Invalid, char const \*name=nullptr, Offset backing\_offset=0) const noexcept
- Attach a data space to a region.*
- int [detach](#) ([l4\\_addr\\_t](#) addr, [L4::Cap](#)< [Dataspace](#) > \*mem, [L4::Cap](#)< [L4::Task](#) > const &task=[This\\_task](#)) const noexcept
- Detach and unmap a region from the address space.*
- int [detach](#) (void \*addr, [L4::Cap](#)< [Dataspace](#) > \*mem, [L4::Cap](#)< [L4::Task](#) > const &task=[This\\_task](#)) const noexcept
- Detach and unmap a region from the address space.*
- int [detach](#) ([l4\\_addr\\_t](#) start, unsigned long size, [L4::Cap](#)< [Dataspace](#) > \*mem, [L4::Cap](#)< [L4::Task](#) > const &task) const noexcept
- Detach and unmap all parts of the regions within the specified interval.*
- long [find](#) ([l4\\_addr\\_t](#) \*addr, unsigned long \*size, Offset \*offset, [L4Re::Rm::Flags](#) \*flags, [L4::Cap](#)< [Dataspace](#) > \*m) noexcept
- Find a region given an address and size.*
- long [get\\_regions](#) ([l4\\_addr\\_t](#) start, [L4::lpc::Ret\\_array](#)< [Region](#) > regions)
- Return the list of regions whose starting addresses are higher or equal to *start* in the address space managed by this region map.*
- long [get\\_areas](#) ([l4\\_addr\\_t](#) start, [L4::lpc::Ret\\_array](#)< [Area](#) > areas)
- Return the list of areas whose starting addresses are higher or equal to *start* in the address space managed by this region map.*
- long [get\\_info](#) ([l4\\_addr\\_t](#) addr, [L4::lpc::String](#)< char > &name, Offset &backing\_offset)
- Return auxiliary information of a region.*

## Public Member Functions inherited from [L4::Pager](#)

- [l4\\_msgtag\\_t page\\_fault](#) ([l4\\_umword\\_t](#) pfa, [l4\\_umword\\_t](#) pc, [L4::lpc::Rcv\\_fpage](#) rwin, [L4::lpc::Opt<L4::lpc::Snd\\_fpage & > fp](#))

*Page-fault protocol message.*

## Public Member Functions inherited from [L4::io\\_pager](#)

- [l4\\_msgtag\\_t io\\_page\\_fault](#) ([l4\\_fpage\\_t](#) io\_pfa, [l4\\_umword\\_t](#) pc, [L4::lpc::Rcv\\_fpage](#) rwin, [L4::lpc::Opt<L4::lpc::Snd\\_fpage & > fp](#))

*IO page fault protocol message.*

## Additional Inherited Members

### Protected Types inherited from

[L4::Kobject\\_t< Rm, L4::Pager, L4RE\\_PROTO\\_RM, L4::Type\\_info::Demand\\_t< 1 > >](#)

- typedef [Rm](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< PROTO, [Rm](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename [L4::Pager::\\_\\_Iface\\_list](#) > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Types inherited from

[L4::Kobject\\_t< Pager, io\\_pager, L4\\_PROTO\\_PAGE\\_FAULT >](#)

- typedef [Pager](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< PROTO, [Pager](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename [io\\_pager::\\_\\_Iface\\_list](#) > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Member Functions inherited from

[L4::Kobject\\_t< Rm, L4::Pager, L4RE\\_PROTO\\_RM, L4::Type\\_info::Demand\\_t< 1 > >](#)

- [L4::Cap< Class > c](#) () const noexcept  
*Get the capability to ourselves.*

### Protected Member Functions inherited from

[L4::Kobject\\_t< Pager, io\\_pager, L4\\_PROTO\\_PAGE\\_FAULT >](#)

- [L4::Cap< Class > c](#) () const noexcept  
*Get the capability to ourselves.*

**Static Protected Member Functions inherited from****[L4::Kobject\\_t< Rm, L4::Pager, L4RE\\_PROTO\\_RM, L4::Type\\_info::Demand\\_t< 1 > >](#)**

- static void **\_\_check\_protocols\_\_** () noexcept

*Helper to check for protocol conflicts.***Static Protected Member Functions inherited from****[L4::Kobject\\_t< Pager, lo\\_pager, L4\\_PROTO\\_PAGE\\_FAULT >](#)**

- static void **\_\_check\_protocols\_\_** () noexcept

*Helper to check for protocol conflicts.***16.301.1 Detailed Description**[Region](#) map.

See also

[Region map API](#) .Definition at line [81](#) of file [rm](#).**16.301.2 Member Enumeration Documentation****16.301.2.1 Detach\_flags**enum [L4Re::Rm::Detach\\_flags](#)

Flags for detach operation.

**Enumerator**

<a href="#">Detach_exact</a>	Do an unmap of the exact region given.
<a href="#">Detach_overlap</a>	Do an unmap of all overlapping regions.
<a href="#">Detach_keep</a>	Do not free the detached data space, ignore the <a href="#">F::Detach_free</a> .

Definition at line [221](#) of file [rm](#).**16.301.2.2 Detach\_result**enum [L4Re::Rm::Detach\\_result](#)

Result values for detach operation.

**Enumerator**



Detached_ds	Detached data sapce.
Kept_ds	Kept data space.
Split_ds	Splitted data space, and done.
Detach_again	Detached data space, more to do.

Definition at line 89 of file [rm](#).

### 16.301.2.3 Region\_flag\_shifts

```
enum L4Re::Rm::Region_flag_shifts
```

[Region](#) flag shifts.

#### Enumerator

Caching_shift	Start of <a href="#">Rm</a> cache bits.
---------------	-----------------------------------------

Definition at line 101 of file [rm](#).

## 16.301.3 Member Function Documentation

### 16.301.3.1 attach() [1/2]

```
long L4Re::Rm::attach (
    l4_addr_t * start,
    unsigned long size,
    Rm::Flags flags,
    L4::Ipc::Cap< Dataspace > mem,
    Rm::Offset offs = 0,
    unsigned char align = L4_PAGESHIFT,
    L4::Cap< L4::Task > const task = L4::Cap<L4::Task>::Invalid,
    char const * name = nullptr,
    Rm::Offset backing_offset = 0) const [noexcept]
```

Attach a data space to a region.

#### Parameters

in, out	<i>start</i>	Virtual start address where the region manager shall attach the data space. Will be rounded down to the nearest start of a page. If <a href="#">L4Re::Rm::F::Search_addr</a> is given this value is used as the start address to search for a free virtual memory region and the resulting address is returned here. If <a href="#">L4Re::Rm::F::In_area</a> is given the value is used as a selector for the area (see <a href="#">L4Re::Rm::reserve_area</a> ) to attach the data space to.
	<i>size</i>	Size of the data space to attach (in bytes). Will be rounded up to the nearest multiple of the page size.

	<i>flags</i>	The flags control how and with which rights the dataspace is attached to the region. See <a href="#">L4Re::Rm::F::Attach_flags</a> and <a href="#">L4Re::Rm::F::Region_flags</a> . The caller must specify the desired rights of the attached region explicitly. The default set of rights is empty. If the <a href="#">F::Eager_map</a> flag is set this function may also return <a href="#">L4Re::Dataspace::map</a> error codes if the mapping fails.
	<i>mem</i>	Data space.
	<i>offs</i>	Offset into the data space to use.
	<i>align</i>	Alignment of the virtual region, log2-size, default: a page ( <a href="#">L4_PAGESHIFT</a> ). This is only meaningful if the <a href="#">L4Re::Rm::F::Search_addr</a> flag is used.
	<i>task</i>	Optional destination task of mapping if <a href="#">F::Eager_map</a> flag was passed. If invalid, the mapping is established in the current task. This parameter is only useful if the region manager is for a foreign task.
	<i>name</i>	Optional name of the region.
	<i>backing_offset</i>	Optional value describing an offset into the backing store of this region.

### Return values

0	Success
-L4_ENOENT	No area could be found (see <a href="#">L4Re::Rm::F::In_area</a> )
-L4_EPERM	Operation not allowed.
-L4_EINVAL	
-L4_EADDRNOTAVAIL	The given address is not available.
<0	IPC errors

Makes the whole or parts of a data space visible in the virtual memory of the corresponding task. The corresponding region in the virtual address space is backed with the contents of the dataspace.

### Note

When searching for a free place in the virtual address space, the space between *start* and the end of the virtual address space is searched.

There is no region object created, instead the region is defined by a virtual address within this range (see [L4Re::Rm::find](#)).

Definition at line 34 of file [rm\\_impl.h](#).

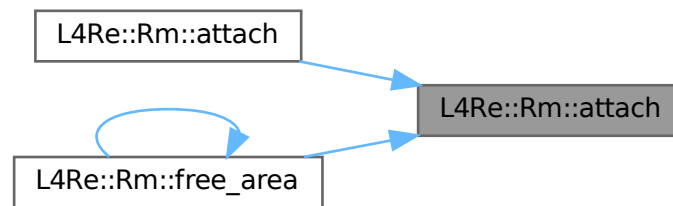
References [L4::Kobject\\_t< Rm, L4::Pager, L4RE\\_PROTO\\_RM, L4::Type\\_info::Demand\\_t< 1 > >::c\(\)](#), [L4Re::Rm::F::Eager\\_map](#), [L4Re::Rm::F::Kernel](#), [L4Re::Rm::F::No\\_eager\\_map](#), [L4Re::Rm::F::Reserved](#), and [L4Re::Rm::F::Rights\\_mask](#).

Referenced by [attach\(\)](#), and [free\\_area\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.301.3.2 attach() [2/2]

```

template<typename T>
long L4Re::Rm::attach (
    T ** start,
    unsigned long size,
    Flags flags,
    L4::Ipc::Cap< Dataspace > mem,
    Offset offs = 0,
    unsigned char align = L4_PAGESHIFT,
    L4::Cap< L4::Task > const task = L4::Cap<L4::Task>::Invalid,
    char const * name = nullptr,
    Offset backing_offset = 0) const [inline], [noexcept]
  
```

Attach a data space to a region.

#### Parameters

<i>in, out</i>	<i>start</i>	Virtual start address where the region manager shall attach the data space. Will be rounded down to the nearest start of a page. If <a href="#">L4Re::Rm::F::Search_addr</a> is given this value is used as the start address to search for a free virtual memory region and the resulting address is returned here. If <a href="#">L4Re::Rm::F::In_area</a> is given the value is used as a selector for the area (see <a href="#">L4Re::Rm::reserve_area</a> ) to attach the data space to.
	<i>size</i>	Size of the data space to attach (in bytes). Will be rounded up to the nearest multiple of the page size.
	<i>flags</i>	The flags control how and with which rights the dataspace is attached to the region. See <a href="#">L4Re::Rm::F::Attach_flags</a> and <a href="#">L4Re::Rm::F::Region_flags</a> . The caller must specify the desired rights of the attached region explicitly. The default set of rights is empty. If the <a href="#">F::Eager_map</a> flag is set this function may also return <a href="#">L4Re::Dataspace::map</a> error codes if the mapping fails.
	<i>mem</i>	Data space.
	<i>offs</i>	Offset into the data space to use.
	<i>align</i>	Alignment of the virtual region, log2-size, default: a page ( <a href="#">L4_PAGESHIFT</a> ). This is only meaningful if the <a href="#">L4Re::Rm::F::Search_addr</a> flag is used.

	<i>task</i>	Optional destination task of mapping if <code>F::Eager_map</code> flag was passed. If invalid, the mapping is established in the current task. This parameter is only useful if the region manager is for a foreign task.
	<i>name</i>	Optional name of the region.
	<i>backing_offset</i>	Optional value describing an offset into the backing store of this region.

### Return values

0	Success
-L4_ENOENT	No area could be found (see <a href="#">L4Re::Rm::F::In_area</a> )
-L4_EPERM	Operation not allowed.
-L4_EINVAL	
-L4_EADDRNOTAVAIL	The given address is not available.
<0	IPC errors

Makes the whole or parts of a data space visible in the virtual memory of the corresponding task. The corresponding region in the virtual address space is backed with the contents of the dataspace.

### Note

When searching for a free place in the virtual address space, the space between *start* and the end of the virtual address space is searched.

There is no region object created, instead the region is defined by a virtual address within this range (see [L4Re::Rm::find](#)).

Definition at line 409 of file [rm](#).

References [attach\(\)](#), and [L4\\_PAGESHIFT](#).

Here is the call graph for this function:



### 16.301.3.3 detach() [1/3]

```

int L4Re::Rm::detach (
    l4_addr_t addr,
    L4::Cap< Dataspace > * mem,
    L4::Cap< L4::Task > const & task = This_task) const [inline], [noexcept]

```

Detach and unmap a region from the address space.

### Parameters

	<i>addr</i>	Virtual address of region, any address within the region is valid.
out	<i>mem</i>	<a href="#">Dataspace</a> that is affected. Give 0 if not interested.
	<i>task</i>	This argument specifies the task where the pages are unmapped. Provide <a href="#">L4::Cap&lt;L4::Task&gt;::Invalid</a> for none. The default is the current task.

### Return values

<a href="#">L4Re::Rm::Detach_result</a>	On success.
<a href="#">-L4_ENOENT</a>	No region found.
<a href="#">&lt;0</a>	IPC errors

Frees a region in the virtual address space given by *addr* (address type). The corresponding part of the address space is now available again.

Definition at line 765 of file [rm](#).

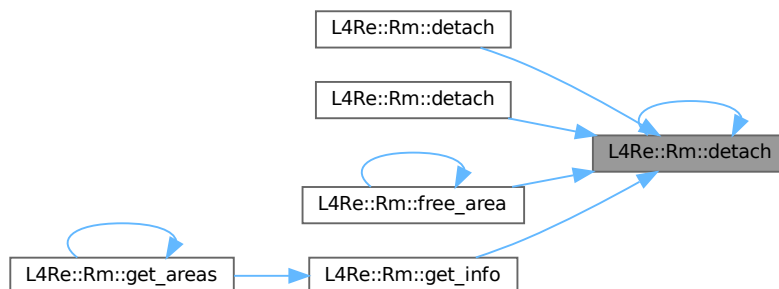
References [detach\(\)](#), and [Detach\\_overlap](#).

Referenced by [detach\(\)](#), [detach\(\)](#), [detach\(\)](#), [free\\_area\(\)](#), and [get\\_info\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



**16.301.3.4 detach()** [2/3]

```
int L4Re::Rm::detach (
    l4_addr_t start,
    unsigned long size,
    L4::Cap< Dataspace > * mem,
    L4::Cap< L4::Task > const & task) const [inline], [noexcept]
```

Detach and unmap all parts of the regions within the specified interval.

**Parameters**

	<i>start</i>	Start of area to detach, must be within region.
	<i>size</i>	Size of of area to detach (in bytes).
out	<i>mem</i>	<a href="#">Dataspace</a> that is affected. Give 0 if not interested.
	<i>task</i>	This argument specifies the task where the pages are unmapped. Provide <a href="#">L4::Cap&lt;L4::Task&gt;::Invalid</a> for none. The default is the current task.

**Return values**

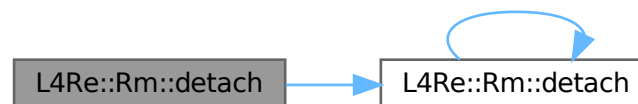
<a href="#">L4Re::Rm::Detach_result</a>	On success.
<a href="#">-L4_ENOENT</a>	No region found.
<a href="#">&lt;0</a>	IPC errors

Frees all regions within the interval given by start and size. If a region overlaps the start or the end of the interval this region is only detached partly. If the interval is within one region the original region is split up into two separate regions.

Definition at line [778](#) of file [rm](#).

References [detach\(\)](#), and [Detach\\_exact](#).

Here is the call graph for this function:



### 16.301.3.5 detach() [3/3]

```
int L4Re::Rm::detach (
    void * addr,
    L4::Cap< Dataspace > * mem,
    L4::Cap< L4::Task > const & task = This_task) const [inline], [noexcept]
```

Detach and unmap a region from the address space.

#### Parameters

---

	<i>addr</i>	Virtual address of region, any address within the region is valid.
out	<i>mem</i>	<a href="#">Dataspace</a> that is affected. Give 0 if not interested.
	<i>task</i>	This argument specifies the task where the pages are unmapped. Provide <a href="#">L4::Cap&lt;L4::Task&gt;::Invalid</a> for none. The default is the current task.

### Return values

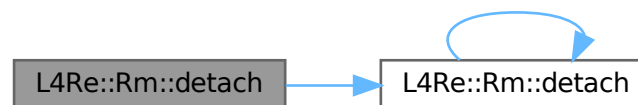
<a href="#">L4Re::Rm::Detach_result</a>	On success.
<a href="#">-L4_ENOENT</a>	No region found.
<a href="#">&lt;0</a>	IPC errors

Frees a region in the virtual address space given by *addr* (address type). The corresponding part of the address space is now available again.

Definition at line 770 of file [rm](#).

References [detach\(\)](#), and [Detach\\_overlap](#).

Here is the call graph for this function:



### 16.301.3.6 find()

```

long L4Re::Rm::find (
    l4\_addr\_t * addr,
    unsigned long * size,
    Offset * offset,
    L4Re::Rm::Flags * flags,
    L4::Cap< Dataspace > \* m) [inline], [noexcept]

```

Find a region given an address and size.

### Parameters

in, out	<i>addr</i>	Address to look for. Returns the start address of the found region.
in, out	<i>size</i>	Size of the area to look for (in bytes). Returns the size of the found region (in bytes).
out	<i>offset</i>	Offset at the beginning of the region within the associated dataspace.

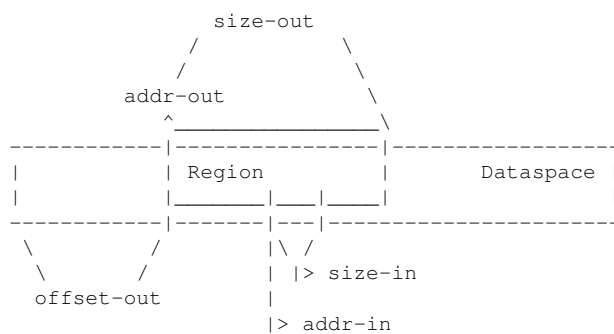


out	<i>flags</i>	Region flags, see <a href="#">F::Region_flags</a> (and <a href="#">F::In_area</a> ).
out	<i>m</i>	Associated dataspace or paging service.

### Return values

0	Success
-L4_EPERM	Operation not allowed.
-L4_ENOENT	No region found.
<0	IPC errors

This function returns the properties of the region that contains the area described by the `addr` and `size` parameter. If no such region is found but a reserved area, the area is returned and [F::In\\_area](#) is set in `flags`. Note, in the case of an area the `offset` and `m` return values are invalid.



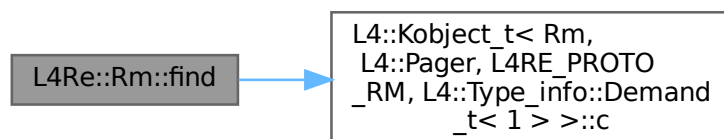
### Note

The value of the `size` input parameter should be 1 to assure that a region can be determined unambiguously.

Definition at line 672 of file [rm](#).

References [L4::Kobject\\_t< Rm, L4::Pager, L4RE\\_PROTO\\_RM, L4::Type\\_info::Demand\\_t< 1 > >::c\(\)](#).

Here is the call graph for this function:



### 16.301.3.7 free\_area()

```
long L4Re::Rm::free_area (  
    l4_addr_t addr)
```

Free an area from the region map.

#### Parameters

---

<i>addr</i>	An address within the area to free.
-------------	-------------------------------------

**Return values**

0	Success
-L4_ENOENT	No area found.
<0	IPC errors

**Note**

The data spaces that are attached to that area are not detached by this operation.

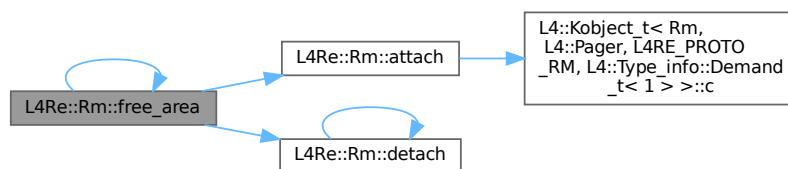
**See also**

[reserve\\_area\(\)](#) for more information about areas.

References [attach\(\)](#), [detach\(\)](#), [free\\_area\(\)](#), [L4::Cap\\_base::Invalid](#), and [L4\\_PAGESHIFT](#).

Referenced by [free\\_area\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



**16.301.3.8 get\_areas()**

```
long L4Re::Rm::get_areas (
    l4_addr_t start,
    L4::Ipc::Ret_array< Area > areas)
```

Return the list of areas whose starting addresses are higher or equal to `start` in the address space managed by this region map.

**Parameters**

---

	<i>start</i>	Virtual address from where to start searching.
out	<i>areas</i>	List of areas found in this region map.

### Return values

$\geq 0$	Number of returned areas in the <i>areas</i> array.
$< 0$	IPC errors

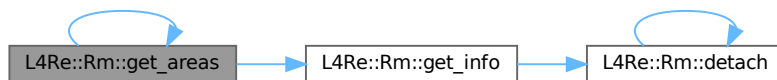
### Note

The returned list of areas might not be complete and the caller shall use the function repeatedly with a start address one larger than the end address of the last area from the previous call.

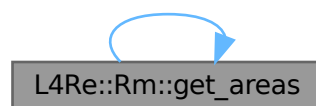
References [get\\_areas\(\)](#), and [get\\_info\(\)](#).

Referenced by [get\\_areas\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.301.3.9 get\_info()

```

long L4Re::Rm::get_info (
    l4_addr_t addr,
    L4::Ipc::String< char > & name,
    Offset & backing_offset)

```

Return auxiliary information of a region.

This is a debugging feature and might not be available.

### Parameters

	<i>addr</i>	Virtual address of the region.
out	<i>name</i>	Name of the region.
out	<i>backing_offset</i>	Backing offset information.

### Return values

0	Success
-L4_ENOENT	<a href="#">Region</a> not found.
-L4_ENOSYS	Function not available.
<0	IPC errors

References [detach\(\)](#).

Referenced by [get\\_areas\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.301.3.10 get\_regions()

```

long L4Re::Rm::get_regions (
    l4_addr_t start,
    L4::Ipc::Ret_array< Region > regions)
  
```

Return the list of regions whose starting addresses are higher or equal to `start` in the address space managed by this region map.

### Parameters

	<i>start</i>	Virtual address from where to start searching.
out	<i>regions</i>	List of regions found in this region map.

### Return values

$\geq 0$	Number of returned regions in the <code>regions</code> array.
$< 0$	IPC errors

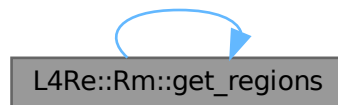
### Note

The returned list of regions might not be complete and the caller shall use the function repeatedly with a start address one larger than the end address of the last region from the previous call.

References [get\\_regions\(\)](#).

Referenced by [get\\_regions\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 16.301.3.11 `reserve_area()` [1/2]

```
long L4Re::Rm::reserve_area (  
    14_addr_t * start,  
    unsigned long size,
```

```
Flags flags = Flags(0),  
unsigned char align = L4_PAGESHIFT) const [inline], [noexcept]
```

Reserve the given area in the region map.

#### Parameters

---



<i>in, out</i>	<i>start</i>	The virtual start address of the area to reserve. Returns the start address of the area.
	<i>size</i>	The size of the area to reserve (in bytes).
	<i>flags</i>	Flags for the reserved area (see <a href="#">L4Re::Rm::F::Region_flags</a> and <a href="#">L4Re::Rm::F::Attach_flags</a> ).
	<i>align</i>	Alignment of area if searched as bits (log2 value).

### Return values

<i>0</i>	Success
<i>-L4_EADDRNOTAVAIL</i>	The given area cannot be reserved.
<i>&lt;0</i>	IPC errors

This function reserves an area within the virtual address space managed by the region map. There are two kinds of areas available:

- Reserved areas (*flags* = [L4Re::Rm::F::Reserved](#)), where no data spaces can be attached
- Special purpose areas (*flags* = 0), where data spaces can be attached to the area via the [L4Re::Rm::F::In\\_area](#) flag and a start address within the area itself.

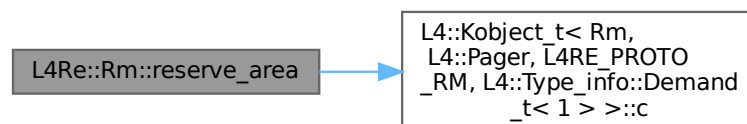
### Note

When searching for a free place in the virtual address space (with *flags* = [L4Re::Rm::F::Search\\_addr](#)), the space between *start* and the end of the virtual address space is searched.

Definition at line 280 of file [rm](#).

References [L4::Kobject\\_t< Rm, L4::Pager, L4RE\\_PROTO\\_RM, L4::Type\\_info::Demand\\_t< 1 > >::c\(\)](#), and [L4\\_PAGESHIFT](#).

Here is the call graph for this function:



**16.301.3.12 reserve\_area()** [2/2]

```
template<typename T>
long L4Re::Rm::reserve_area (
    T ** start,
    unsigned long size,
    Flags flags = Flags(0),
    unsigned char align = L4_PAGESHIFT) const [inline], [noexcept]
```

Reserve the given area in the region map.

**Parameters**

---

<i>in, out</i>	<i>start</i>	The virtual start address of the area to reserve. Returns the start address of the area.
	<i>size</i>	The size of the area to reserve (in bytes).
	<i>flags</i>	Flags for the reserved area (see <a href="#">F::Region_flags</a> and <a href="#">F::Attach_flags</a> ).
	<i>align</i>	Alignment of area if searched as bits (log2 value).

### Return values

0	Success
-L4_EADDRNOTAVAIL	The given area cannot be reserved.
<0	IPC errors

For more information, please refer to the analogous function

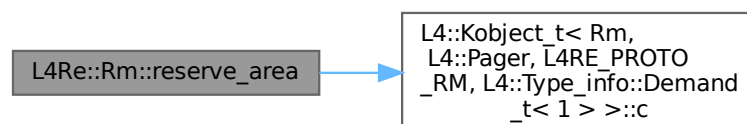
See also

[L4Re::Rm::reserve\\_area](#).

Definition at line 306 of file [rm](#).

References [L4::Kobject\\_t< Rm, L4::Pager, L4RE\\_PROTO\\_RM, L4::Type\\_info::Demand\\_t< 1 > >::c\(\)](#), [L4\\_PAGESHIFT](#), and

Here is the call graph for this function:



The documentation for this class was generated from the following files:

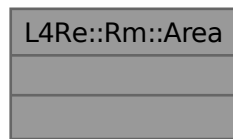
- [l4/re/rm](#)
- [l4/re/impl/rm\\_impl.h](#)

## 16.302 L4Re::Rm::Area Struct Reference

An area is a range of virtual addresses which is reserved, see [L4Re::Rm::reserve\\_area\(\)](#).

```
#include <rm>
```

Collaboration diagram for L4Re::Rm::Area:



### 16.302.1 Detailed Description

An area is a range of virtual addresses which is reserved, see [L4Re::Rm::reserve\\_area\(\)](#).

See also

[Region map API](#)

Definition at line [699](#) of file [rm](#).

The documentation for this struct was generated from the following file:

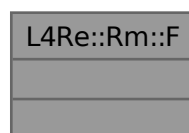
- [l4/re/rm](#)

## 16.303 L4Re::Rm::F Struct Reference

[Rm](#) flags definitions.

```
#include <rm>
```

Collaboration diagram for L4Re::Rm::F:



## Public Types

- enum [Attach\\_flags](#) : `l4_uint32_t` {  
[Search\\_addr](#) = 0x20000 , [In\\_area](#) = 0x40000 , [Eager\\_map](#) = 0x80000 , [No\\_eager\\_map](#) = 0x100000 ,  
[Attach\\_mask](#) = 0x1f0000 }  
*Flags for attach operation.*
- enum [Region\\_flags](#) : `l4_uint16_t` {  
[Rights\\_mask](#) = 0x0f , [R](#) = `Datspace::F::R` , [W](#) = `Datspace::F::W` , [X](#) = `Datspace::F::X` ,  
[RW](#) = `Datspace::F::RW` , [RX](#) = `Datspace::F::RX` , [RWX](#) = `Datspace::F::RWX` , [Kernel](#) = 0x100 ,  
[Detach\\_free](#) = 0x200 , [Pager](#) = 0x400 , [Reserved](#) = 0x800 , [Caching\\_mask](#) = `Datspace::F::Caching_mask` ,  
[Cache\\_normal](#) = `Datspace::F::Normal` , [Cache\\_buffered](#) = `Datspace::F::Bufferable` , [Cache\\_uncached](#) =  
`Datspace::F::Uncacheable` , [Ds\\_map\\_mask](#) = 0xff ,  
[Region\\_flags\\_mask](#) = 0xffff }  
*Region flags (permissions, cacheability, special).*

### 16.303.1 Detailed Description

[Rm](#) flags definitions.

Definition at line 108 of file [rm](#).

### 16.303.2 Member Enumeration Documentation

#### 16.303.2.1 [Attach\\_flags](#)

```
enum L4Re::Rm::F::Attach_flags : l4_uint32_t
```

Flags for attach operation.

#### Enumerator

<a href="#">Search_addr</a>	Search for a suitable address range.
<a href="#">In_area</a>	Search only in area, or map into area.
<a href="#">Eager_map</a>	Eagerly map the attached data space in.
<a href="#">No_eager_map</a>	Prevent eager mapping of the attached data space.
<a href="#">Attach_mask</a>	Mask of all attach flags.

Definition at line 111 of file [rm](#).

#### 16.303.2.2 [Region\\_flags](#)

```
enum L4Re::Rm::F::Region_flags : l4_uint16_t
```

[Region](#) flags (permissions, cacheability, special).

#### Enumerator

Rights_mask	<a href="#">Region</a> rights.
R	Readable region.
W	Writable region.
X	Executable region.
RW	Readable and writable region.
RX	Readable and executable region.
RWX	Readable, writable and executable region.
Kernel	Kernel-provided memory (KUMEM).
Detach_free	Free the portion of the data space after detach.
Pager	<a href="#">Region</a> has a pager.
Reserved	<a href="#">Region</a> is reserved (blocked).
Caching_mask	Mask of all <a href="#">Rm</a> cache bits.
Cache_normal	Cache bits for normal cacheable memory. This is the default if no other cache-related flag was specified.
Cache_buffered	Cache bits for buffered (write combining) memory.
Cache_uncached	Cache bits for uncached memory.
Ds_map_mask	Mask for all bits for cache options and rights.
Region_flags_mask	Mask of all region flags.

Definition at line 128 of file [rm](#).

The documentation for this struct was generated from the following file:

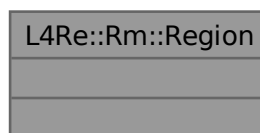
- [l4/re/rm](#)

## 16.304 L4Re::Rm::Region Struct Reference

A region is a range of virtual addresses which is backed by content.

```
#include <rm>
```

Collaboration diagram for L4Re::Rm::Region:



### 16.304.1 Detailed Description

A region is a range of virtual addresses which is backed by content.

See also

[Region map API](#)

Definition at line 686 of file [rm](#).

The documentation for this struct was generated from the following file:

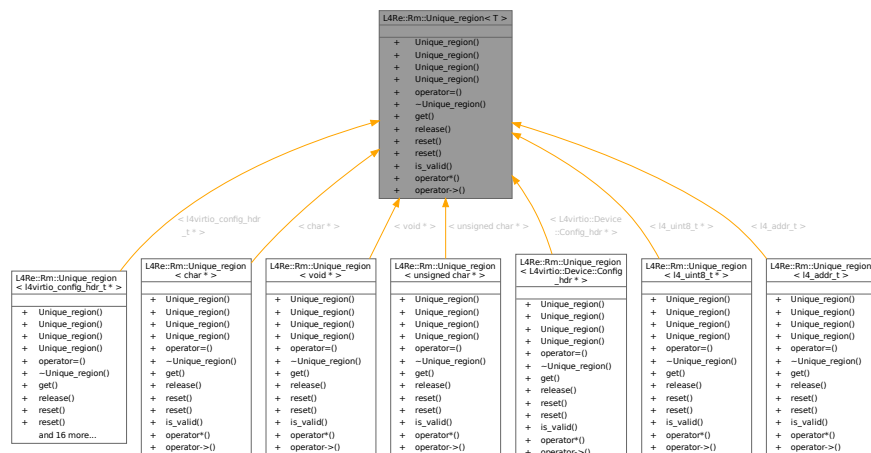
- [l4/re/rm](#)

## 16.305 L4Re::Rm::Unique\_region< T > Class Template Reference

Unique region.

```
#include <rm>
```

Inheritance diagram for L4Re::Rm::Unique\_region< T >:



Collaboration diagram for L4Re::Rm::Unique\_region< T >:

L4Re::Rm::Unique_region< T >	
+	Unique_region()
+	Unique_region()
+	Unique_region()
+	Unique_region()
+	operator=()
+	~Unique_region()
+	get()
+	release()
+	reset()
+	reset()
+	is_valid()
+	operator*()
+	operator->()

## Public Member Functions

- **Unique\_region** () noexcept  
*Construct an invalid [Unique\\_region](#).*
- **Unique\_region** (T addr) noexcept  
*Construct a [Unique\\_region](#) from an address.*
- **Unique\_region** (T addr, L4::Cap< Rm > const &rm) noexcept  
*Construct a valid [Unique\\_region](#) from an address and a region manager.*
- **Unique\_region** (Unique\_region &&o) noexcept  
*Move-Construct a [Unique\\_region](#).*
- **Unique\_region** & operator= (Unique\_region &&o) noexcept  
*Move-assign a [Unique\\_region](#).*
- **~Unique\_region** () noexcept  
*Destructor.*
- T **get** () const noexcept  
*Return the address.*
- T **release** () noexcept  
*Return the address and invalidate the [Unique\\_region](#).*
- void **reset** (T addr, L4::Cap< Rm > const &rm) noexcept  
*Set new address and region manager.*
- void **reset** () noexcept  
*Make the [Unique\\_region](#) invalid.*
- bool **is\_valid** () const noexcept  
*Check if the [Unique\\_region](#) is valid.*



- **T operator\*** () const noexcept  
*Dereference the address.*
- **T operator->** () const noexcept  
*Member access for the address.*

### 16.305.1 Detailed Description

```
template<typename T>
class L4Re::Rm::Unique_region< T >
```

Unique region.

Capture a single region with automatic detach on destruction and unique ownership. Stores the start address and the region-mapper capability internally. A unique region is valid precisely if the internal region-mapper capability is valid. The features for unique ownership and automatic detach are only active for valid unique regions.

Definition at line 435 of file [rm](#).

### 16.305.2 Constructor & Destructor Documentation

#### 16.305.2.1 Unique\_region() [1/3]

```
template<typename T>
L4Re::Rm::Unique_region< T >::Unique_region (
    T addr) [inline], [explicit], [noexcept]
```

Construct a [Unique\\_region](#) from an address.

No region manager is set.

##### Parameters

<i>addr</i>	The new address
-------------	-----------------

Definition at line 456 of file [rm](#).

#### 16.305.2.2 Unique\_region() [2/3]

```
template<typename T>
L4Re::Rm::Unique_region< T >::Unique_region (
    T addr,
    L4::Cap< Rm > const & rm) [inline], [noexcept]
```

Construct a valid [Unique\\_region](#) from an address and a region manager.

##### Parameters

<i>addr</i>	The address
<i>rm</i>	The region manager

Definition at line 465 of file [rm](#).

### 16.305.2.3 Unique\_region() [3/3]

```
template<typename T>
L4Re::Rm::Unique_region< T >::Unique_region (
    Unique_region< T > && o) [inline], [noexcept]
```

Move-Construct a [Unique\\_region](#).

#### Parameters

<i>o</i>	L-value reference to other region.
----------	------------------------------------

Definition at line 473 of file [rm](#).

### 16.305.2.4 ~Unique\_region()

```
template<typename T>
L4Re::Rm::Unique_region< T >::~~Unique_region () [inline], [noexcept]
```

Destructor.

If the region is valid, call [detach](#).

Definition at line 498 of file [rm](#).

## 16.305.3 Member Function Documentation

### 16.305.3.1 get()

```
template<typename T>
T L4Re::Rm::Unique_region< T >::get () const [inline], [noexcept]
```

Return the address.

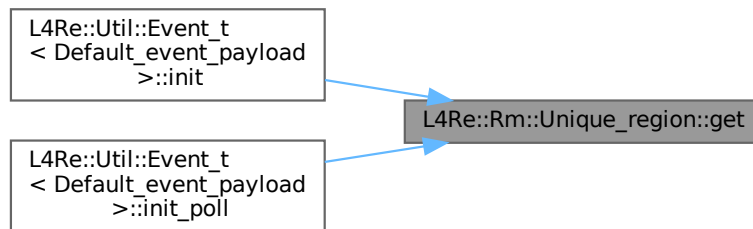
**Returns**

the address

Definition at line 509 of file [rm](#).

Referenced by [L4Re::Util::Event\\_t< Default\\_event\\_payload >::init\(\)](#), and [L4Re::Util::Event\\_t< Default\\_event\\_payload >::init\\_poll\(\)](#).

Here is the caller graph for this function:

**16.305.3.2 is\_valid()**

```
template<typename T>
bool L4Re::Rm::Unique_region< T >::is_valid () const [inline], [noexcept]
```

Check if the [Unique\\_region](#) is valid.

**Returns**

true iff the [Unique\\_region](#) is valid

Definition at line 549 of file [rm](#).

**16.305.3.3 operator=()**

```
template<typename T>
Unique_region & L4Re::Rm::Unique_region< T >::operator= (
    Unique_region< T > && o) [inline], [noexcept]
```

Move-assign a [Unique\\_region](#).

**Parameters**

<i>o</i>	L-value reference to region to assign from
----------	--------------------------------------------

Definition at line 481 of file [rm](#).

**16.305.3.4 release()**

```
template<typename T>
T L4Re::Rm::Unique_region< T >::release () [inline], [noexcept]
```

Return the address and invalidate the [Unique\\_region](#).

**Returns**

the address

Definition at line [517](#) of file [rm](#).

**16.305.3.5 reset()**

```
template<typename T>
void L4Re::Rm::Unique_region< T >::reset (
    T addr,
    L4::Cap< Rm > const & rm) [inline], [noexcept]
```

Set new address and region manager.

**Parameters**

<i>addr</i>	The new address
<i>rm</i>	The new region manager

Definition at line [529](#) of file [rm](#).

The documentation for this class was generated from the following file:

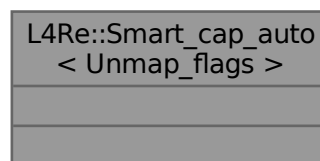
- [l4/re/rm](#)

## 16.306 L4Re::Smart\_cap\_auto< Unmap\_flags > Class Template Reference

Helper for [Unique\\_cap](#) and [Unique\\_del\\_cap](#).

```
#include <cap_alloc>
```

Collaboration diagram for L4Re::Smart\_cap\_auto< Unmap\_flags >:



### 16.306.1 Detailed Description

```
template<unsigned long Unmap_flags = L4_FP_ALL_SPACES>
class L4Re::Smart_cap_auto< Unmap_flags >
```

Helper for [Unique\\_cap](#) and [Unique\\_del\\_cap](#).

Definition at line 110 of file [cap\\_alloc](#).

The documentation for this class was generated from the following file:

- [l4/re/cap\\_alloc](#)

## 16.307 L4Re::Smart\_count\_cap< Unmap\_flags > Class Template Reference

Helper for [Ref\\_cap](#) and [Ref\\_del\\_cap](#).

```
#include <cap_alloc>
```

Collaboration diagram for L4Re::Smart\_count\_cap< Unmap\_flags >:

L4Re::Smart_count_cap < Unmap_flags >	
+	<a href="#">free()</a>
+	<a href="#">copy()</a>
+	<a href="#">invalidate()</a>

### Public Member Functions

- void **free** ([L4::Cap\\_base](#) &c) noexcept  
*Free operation for [L4::Smart\\_cap](#) (decrement ref count and delete if 0).*
- [L4::Cap\\_base](#) **copy** ([L4::Cap\\_base](#) const &src)  
*Copy operation for [L4::Smart\\_cap](#) (increment ref count).*

### Static Public Member Functions

- static void **invalidate** ([L4::Cap\\_base](#) &c) noexcept  
*Invalidate operation for [L4::Smart\\_cap](#).*

### 16.307.1 Detailed Description

```
template<unsigned long Unmap_flags = L4_FP_ALL_SPACES>
class L4Re::Smart_count_cap< Unmap_flags >
```

Helper for Ref\_cap and Ref\_del\_cap.

Definition at line 139 of file [cap\\_alloc](#).

The documentation for this class was generated from the following file:

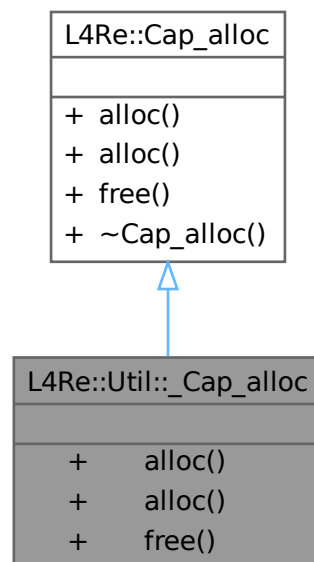
- [l4/re/cap\\_alloc](#)

## 16.308 L4Re::Util::\_Cap\_alloc Class Reference

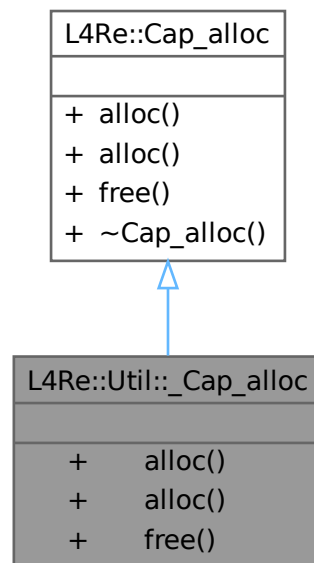
Adapter to expose the cap allocator implementation as [L4Re::Cap\\_alloc](#) compatible class.

```
#include <cap_alloc_impl.h>
```

Inheritance diagram for L4Re::Util::\_Cap\_alloc:



Collaboration diagram for L4Re::Util::\_Cap\_alloc:



### Public Member Functions

- [L4::Cap](#)< void > [alloc](#) () noexcept override  
*Allocate a capability.*
- template<typename T>  
[L4::Cap](#)< T > [alloc](#) () noexcept  
*Allocate a capability.*
- void [free](#) (L4::Cap< void > cap, [l4\\_cap\\_idx\\_t](#) task=[L4\\_INVALID\\_CAP](#), unsigned unmap\_flags=[L4\\_FP\\_ALL\\_SPACES](#)) noexcept override  
*Free a capability.*

### Public Member Functions inherited from [L4Re::Cap\\_alloc](#)

- template<typename T>  
[L4::Cap](#)< T > [alloc](#) () noexcept  
*Allocate a capability.*
- virtual [~Cap\\_alloc](#) ()=0  
*Destructor.*

## 16.308.1 Detailed Description

Adapter to expose the cap allocator implementation as [L4Re::Cap\\_alloc](#) compatible class.

Not intended to be used in application code.

Definition at line 66 of file [cap\\_alloc\\_impl.h](#).

## 16.308.2 Member Function Documentation

### 16.308.2.1 `alloc()` [1/2]

```
template<typename T>
L4::Cap< T > L4Re::Util::_Cap_alloc::alloc () [inline], [virtual], [noexcept]
```

Allocate a capability.

#### Returns

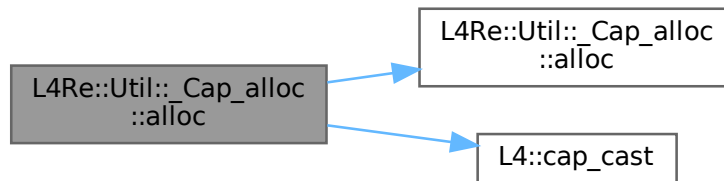
Capability of type void

Implements [L4Re::Cap\\_alloc](#).

Definition at line 79 of file [cap\\_alloc\\_impl.h](#).

References [alloc\(\)](#), and [L4::cap\\_cast\(\)](#).

Here is the call graph for this function:



### 16.308.2.2 `alloc()` [2/2]

```
L4::Cap< void > L4Re::Util::_Cap_alloc::alloc () [inline], [override], [virtual], [noexcept]
```

Allocate a capability.

#### Returns

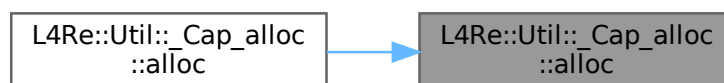
Capability of type void

Implements [L4Re::Cap\\_alloc](#).

Definition at line 75 of file [cap\\_alloc\\_impl.h](#).

Referenced by [alloc\(\)](#).

Here is the caller graph for this function:





### 16.308.2.3 free()

```
void L4Re::Util::_Cap_alloc::free (
    L4::Cap< void > cap,
    l4_cap_idx_t task = L4_INVALID_CAP,
    unsigned unmap_flags = L4_FP_ALL_SPACES) [inline], [override], [virtual], [noexcept]
```

Free a capability.

#### Parameters

<i>cap</i>	Capability to free.
<i>task</i>	If set, task to unmap the capability from.
<i>unmap_flags</i>	Flags for unmap, see <a href="#">l4_unmap_flags_t</a> .

Implements [L4Re::Cap\\_alloc](#).

Definition at line 85 of file [cap\\_alloc\\_impl.h](#).

References [L4\\_FP\\_ALL\\_SPACES](#), and [L4\\_INVALID\\_CAP](#).

The documentation for this class was generated from the following file:

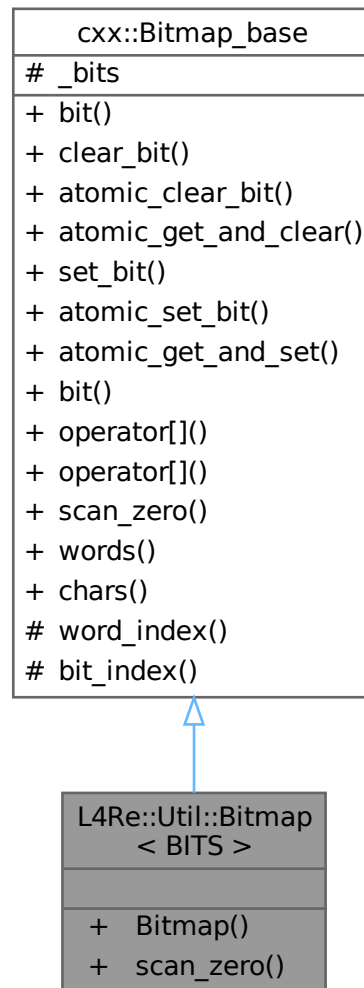
- [l4/re/util/cap\\_alloc\\_impl.h](#)

## 16.309 L4Re::Util::Bitmap< BITS > Class Template Reference

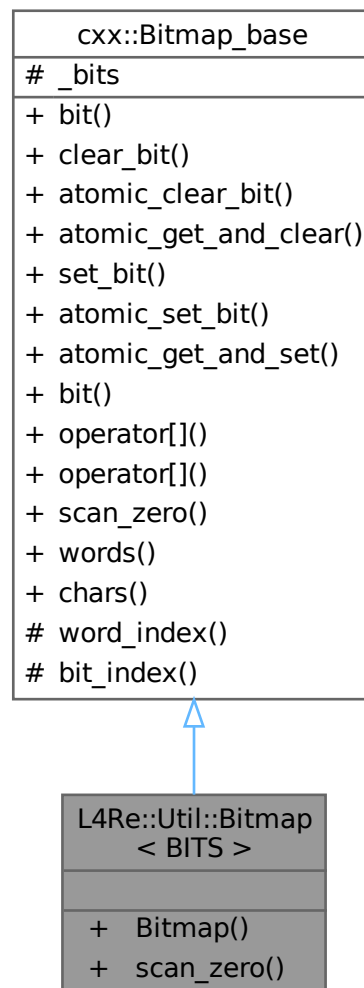
A static bitmap.

```
#include <bitmap>
```

Inheritance diagram for L4Re::Util::Bitmap< BITS >:



Collaboration diagram for L4Re::Util::Bitmap< BITS >:



## Public Member Functions

- **Bitmap** () noexcept  
*Create a bitmap with `BITS` bits.*
- long `scan_zero` (long start\_bit=0) const noexcept  
*Scan for the first zero bit.*

## Public Member Functions inherited from `cx::Bitmap_base`

- void `bit` (long bit, bool on) noexcept  
*Set the value of bit `bit` to on.*
- void `clear_bit` (long bit) noexcept  
*Clear bit `bit`.*

- void `atomic_clear_bit` (long `bit`) noexcept  
*Clear bit `bit` atomically.*
- `word_type` `atomic_get_and_clear` (long `bit`) noexcept  
*Clear bit `bit` atomically and return old state.*
- void `set_bit` (long `bit`) noexcept  
*Set bit `bit`.*
- void `atomic_set_bit` (long `bit`) noexcept  
*Set bit `bit` atomically.*
- `word_type` `atomic_get_and_set` (long `bit`) noexcept  
*Set bit `bit` atomically and return old state.*
- `word_type` `bit` (long `bit`) const noexcept  
*Get the truth value of a bit.*
- `word_type` `operator[]` (long `bit`) const noexcept  
*Get the bit at index `bit`.*
- `Bit operator[]` (long `bit`) noexcept  
*Get the lvalue for the bit at index `bit`.*
- long `scan_zero` (long `max_bit`, long `start_bit`=0) const noexcept  
*Scan for the first zero bit.*

### Additional Inherited Members

### Static Public Member Functions inherited from `cxx::Bitmap_base`

- static long `words` (long bits) noexcept  
*Get the number of `Words` that are used for the bitmap.*
- static long `chars` (long bits) noexcept  
*Get the number of chars that are used for the bitmap.*

### Protected Types inherited from `cxx::Bitmap_base`

- enum { `W_bits` = sizeof(word\_type) \* 8 , `C_bits` = 8 }
- typedef unsigned long `word_type`  
*Data type for each element of the bit buffer.*

### Static Protected Member Functions inherited from `cxx::Bitmap_base`

- static unsigned `word_index` (unsigned `bit`)  
*Get the word index for the given bit.*
- static unsigned `bit_index` (unsigned `bit`)  
*Get the bit index within `word_type` for the given bit.*

### Protected Attributes inherited from `cxx::Bitmap_base`

- `word_type` \* `_bits`  
*Pointer to the buffer storing the bits.*

## 16.309.1 Detailed Description

```
template<int BITS>
class L4Re::Util::Bitmap< BITS >
```

A static bitmap.

### Template Parameters

<i>BITS</i>	The number of bits that shall be in the bitmap.
-------------	-------------------------------------------------

Definition at line 220 of file [bitmap](#).

## 16.309.2 Member Function Documentation

### 16.309.2.1 scan\_zero()

```
template<int BITS>
long cxx::Bitmap< BITS >::scan_zero (
    long start_bit = 0) const [inline], [noexcept]
```

Scan for the first zero bit.

#### Parameters

<i>start_bit</i>	Hint at the number of the first bit to look at. Zero bits below <i>start_bit</i> may or may not be taken into account by the implementation.
------------------	----------------------------------------------------------------------------------------------------------------------------------------------

#### Return values

<i>&gt;=</i>	0 Number of first zero bit found.
<i>-1</i>	All bits at <i>start_bit</i> or higher are set.

Compared to [Bitmap\\_base::scan\\_zero\(\)](#), the upper bound is set to BITS.

Definition at line 365 of file [bitmap](#).

The documentation for this class was generated from the following file:

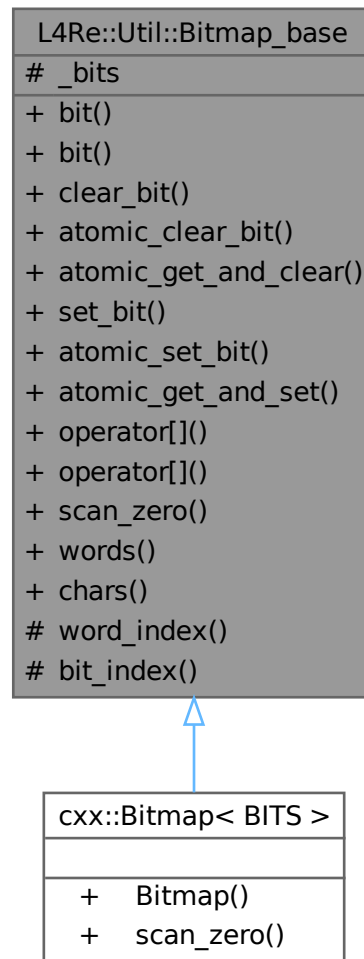
- [l4/cxx/bitmap](#)

## 16.310 L4Re::Util::Bitmap\_base Class Reference

Basic bitmap abstraction.

```
#include <bitmap>
```

Inheritance diagram for L4Re::Util::Bitmap\_base:



Collaboration diagram for L4Re::Util::Bitmap\_base:

L4Re::Util::Bitmap_base
# _bits
+ bit()
+ bit()
+ clear_bit()
+ atomic_clear_bit()
+ atomic_get_and_clear()
+ set_bit()
+ atomic_set_bit()
+ atomic_get_and_set()
+ operator[]()
+ operator[]()
+ scan_zero()
+ words()
+ chars()
# word_index()
# bit_index()

## Data Structures

- class [Bit](#)  
*A writable bit in a bitmap.*
- class [Word](#)  
*Helper abstraction for a word contained in the bitmap.*
- class [Char](#)  
*Helper abstraction for a byte contained in the bitmap.*

## Public Member Functions

- void [bit](#) (long bit, bool on) noexcept  
*Set the value of bit [bit](#) to on.*
- [word\\_type](#) [bit](#) (long bit) const noexcept  
*Get the truth value of a bit.*
- void [clear\\_bit](#) (long [bit](#)) noexcept  
*Clear bit [bit](#).*
- void [atomic\\_clear\\_bit](#) (long [bit](#)) noexcept  
*Clear bit [bit](#) atomically.*
- [word\\_type](#) [atomic\\_get\\_and\\_clear](#) (long [bit](#)) noexcept  
*Clear bit [bit](#) atomically and return old state.*

- void `set_bit` (long `bit`) noexcept  
*Set bit `bit`.*
- void `atomic_set_bit` (long `bit`) noexcept  
*Set bit `bit` atomically.*
- `word_type atomic_get_and_set` (long `bit`) noexcept  
*Set bit `bit` atomically and return old state.*
- `word_type operator[]` (long `bit`) const noexcept  
*Get the bit at index `bit`.*
- `Bit operator[]` (long `bit`) noexcept  
*Get the lvalue for the bit at index `bit`.*
- long `scan_zero` (long `max_bit`, long `start_bit=0`) const noexcept  
*Scan for the first zero bit.*

### Static Public Member Functions

- static long `words` (long bits) noexcept  
*Get the number of `Words` that are used for the bitmap.*
- static long `chars` (long bits) noexcept  
*Get the number of chars that are used for the bitmap.*

### Protected Types

- enum { `W_bits` = sizeof(word\_type) \* 8 , `C_bits` = 8 }
- typedef unsigned long `word_type`  
*Data type for each element of the bit buffer.*

### Static Protected Member Functions

- static unsigned `word_index` (unsigned `bit`)  
*Get the word index for the given bit.*
- static unsigned `bit_index` (unsigned `bit`)  
*Get the bit index within `word_type` for the given bit.*

### Protected Attributes

- `word_type * _bits`  
*Pointer to the buffer storing the bits.*

## 16.310.1 Detailed Description

Basic bitmap abstraction.

This abstraction keeps a pointer to a memory area that is used as bitmap.

Definition at line 18 of file `bitmap`.

## 16.310.2 Member Enumeration Documentation

### 16.310.2.1 anonymous enum

```
anonymous enum [protected]
```

#### Enumerator

---



W_bits	number of bits in <a href="#">word_type</a>
C_bits	number of bits in char

Definition at line 26 of file [bitmap](#).

## 16.310.3 Member Function Documentation

### 16.310.3.1 atomic\_clear\_bit()

```
void cxx::Bitmap_base::atomic_clear_bit (
    long bit) [inline], [noexcept]
```

Clear bit [bit](#) atomically.

Use this function for multi-threaded access to the bitmap.

#### Parameters

<i>bit</i>	The number of the bit to clear.
------------	---------------------------------

Definition at line 269 of file [bitmap](#).

### 16.310.3.2 atomic\_get\_and\_clear()

```
Bitmap_base::word_type cxx::Bitmap_base::atomic_get_and_clear (
    long bit) [inline], [noexcept]
```

Clear bit [bit](#) atomically and return old state.

Use this function for multi-threaded access to the bitmap.

#### Parameters

<i>bit</i>	The number of the bit to clear.
------------	---------------------------------

Definition at line 279 of file [bitmap](#).

### 16.310.3.3 atomic\_get\_and\_set()

```
Bitmap_base::word_type cxx::Bitmap_base::atomic_get_and_set (
    long bit) [inline], [noexcept]
```

Set bit [bit](#) atomically and return old state.

Use this function for multi-threaded access to the bitmap.

#### Parameters

<i>bit</i>	The number of the bit to set.
------------	-------------------------------

Definition at line 308 of file [bitmap](#).

#### 16.310.3.4 `atomic_set_bit()`

```
void cxx::Bitmap_base::atomic_set_bit (
    long bit) [inline], [noexcept]
```

Set bit `bit` atomically.

Use this function for multi-threaded access to the bitmap.

##### Parameters

<i>bit</i>	The number of the bit to set.
------------	-------------------------------

Definition at line 298 of file [bitmap](#).

#### 16.310.3.5 `bit()` [1/2]

```
Bitmap_base::word_type cxx::Bitmap_base::bit (
    long bit) const [inline], [noexcept]
```

Get the truth value of a bit.

##### Parameters

<i>bit</i>	The number of the bit to read.
------------	--------------------------------

##### Return values

0	Bit is not set.
<code>!= 0</code>	Bit is set.

Definition at line 318 of file [bitmap](#).

#### 16.310.3.6 `bit()` [2/2]

```
void cxx::Bitmap_base::bit (
    long bit,
    bool on) [inline], [noexcept]
```

Set the value of bit `bit` to on.

##### Parameters

<i>bit</i>	The number of the bit.
<i>on</i>	The boolean value that shall be assigned to the bit.

Definition at line 251 of file [bitmap](#).

### 16.310.3.7 bit\_index()

```
unsigned cxx::Bitmap_base::bit_index (
    unsigned bit) [inline], [static], [protected]
```

Get the bit index within [word\\_type](#) for the given bit.

#### Parameters

<i>bit</i>	The bit index in the bitmap.
------------	------------------------------

#### Returns

the bit index within [word\\_type](#) (bit % W\_bits).

Definition at line 53 of file [bitmap](#).

### 16.310.3.8 clear\_bit()

```
void cxx::Bitmap_base::clear_bit (
    long bit) [inline], [noexcept]
```

Clear bit [bit](#).

#### Parameters

<i>bit</i>	The number of the bit to clear.
------------	---------------------------------

Definition at line 260 of file [bitmap](#).

### 16.310.3.9 operator[]() [1/2]

```
word\_type cxx::Bitmap_base::operator[] (
    long bit) const [inline], [noexcept]
```

Get the bit at index [bit](#).

#### Parameters

---

<i>bit</i>	The number of the bit to read.
------------	--------------------------------

#### Return values

0	Bit is not set.
!= 0	Bit is set.

Definition at line 181 of file [bitmap](#).

#### 16.310.3.10 operator[]() [2/2]

```
Bit cxx::Bitmap_base::operator[] (
    long bit) [inline], [noexcept]
```

Get the lvalue for the bit at index [bit](#).

#### Parameters

<i>bit</i>	The number.
------------	-------------

#### Returns

lvalue for [bit](#)

Definition at line 191 of file [bitmap](#).

#### 16.310.3.11 scan\_zero()

```
long cxx::Bitmap_base::scan_zero (
    long max_bit,
    long start_bit = 0) const [inline], [noexcept]
```

Scan for the first zero bit.

#### Parameters

<i>max_bit</i>	Upper bound (exclusive) for the scanning operation.
<i>start_bit</i>	Hint at the number of the first bit to look at. Zero bits below <i>start_bit</i> may or may not be taken into account by the implementation.

#### Return values

>=	0 Number of first zero bit found.
----	-----------------------------------

-1	All bits between <code>start_bit</code> and <code>max_bit</code> are set.
----	---------------------------------------------------------------------------

Definition at line 339 of file [bitmap](#).

### 16.310.3.12 `set_bit()`

```
void cxx::Bitmap_base::set_bit (  
    long bit) [inline], [noexcept]
```

Set bit [bit](#).

#### Parameters

<i>bit</i>	The number of the bit to set.
------------	-------------------------------

Definition at line 289 of file [bitmap](#).

### 16.310.3.13 `word_index()`

```
unsigned cxx::Bitmap_base::word_index (  
    unsigned bit) [inline], [static], [protected]
```

Get the word index for the given bit.

#### Parameters

<i>bit</i>	The index of the bit in question.
------------	-----------------------------------

#### Returns

the index in [Bitmap\\_base::\\_bits](#) for the given bit (bit / W\_bits).

Definition at line 44 of file [bitmap](#).

The documentation for this class was generated from the following file:

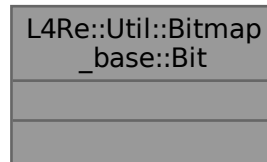
- I4/cxx/bitmap

## 16.311 L4Re::Util::Bitmap\_base::Bit Class Reference

A writable bit in a bitmap.

```
#include <bitmap>
```

Collaboration diagram for L4Re::Util::Bitmap\_base::Bit:



### 16.311.1 Detailed Description

A writable bit in a bitmap.

Definition at line 58 of file [bitmap](#).

The documentation for this class was generated from the following file:

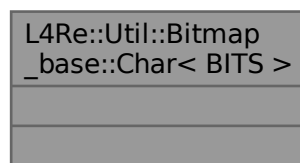
- l4/cxx/bitmap

## 16.312 L4Re::Util::Bitmap\_base::Char< BITS > Class Template Reference

Helper abstraction for a byte contained in the bitmap.

```
#include <bitmap>
```

Collaboration diagram for L4Re::Util::Bitmap\_base::Char< BITS >:



### 16.312.1 Detailed Description

```
template<long BITS>
class L4Re::Util::Bitmap_base::Char< BITS >
```

Helper abstraction for a byte contained in the bitmap.

Definition at line 95 of file [bitmap](#).

The documentation for this class was generated from the following file:

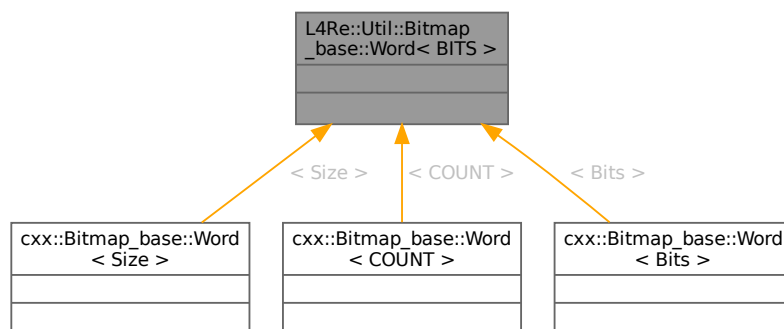
- I4/cxx/bitmap

## 16.313 L4Re::Util::Bitmap\_base::Word< BITS > Class Template Reference

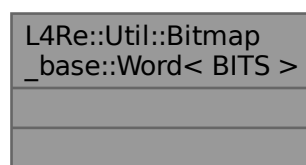
Helper abstraction for a word contained in the bitmap.

```
#include <bitmap>
```

Inheritance diagram for L4Re::Util::Bitmap\_base::Word< BITS >:



Collaboration diagram for L4Re::Util::Bitmap\_base::Word< BITS >:



### 16.313.1 Detailed Description

```
template<long BITS>
class L4Re::Util::Bitmap_base::Word< BITS >
```

Helper abstraction for a word contained in the bitmap.

Definition at line 79 of file [bitmap](#).

The documentation for this class was generated from the following file:

- I4/cxx/bitmap

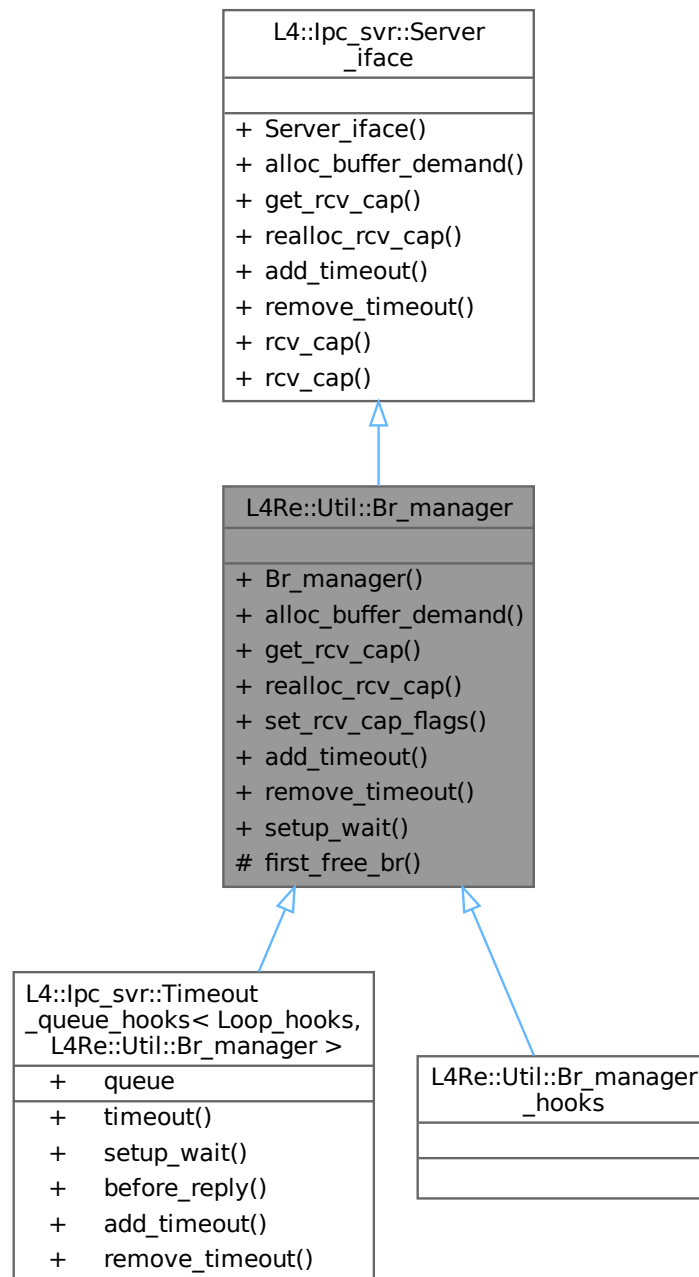
## 16.314 L4Re::Util::Br\_manager Class Reference

Buffer-register (BR) manager for [L4::Server](#).

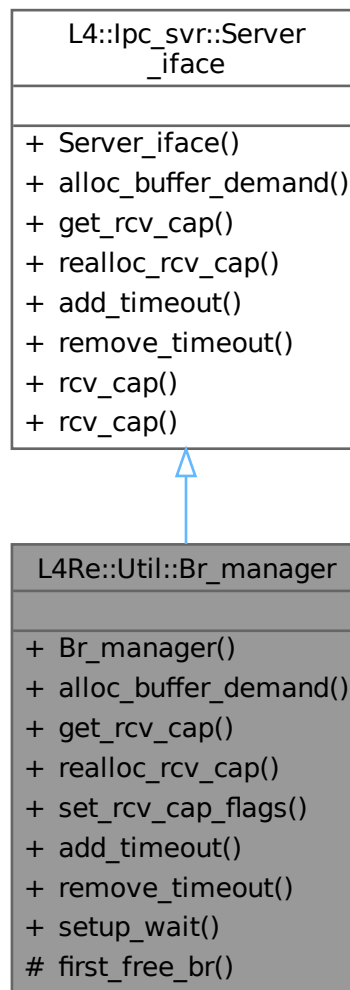
```
#include <br_manager>
```



Inheritance diagram for L4Re::Util::Br\_manager:



Collaboration diagram for L4Re::Util::Br\_manager:



## Public Member Functions

- **Br\_manager ()**  
*Make a buffer-register (BR) manager.*
- int **alloc\_buffer\_demand** (**Demand** const &d) override  
*Tells the server to allocate buffers for the given demand.*
- **L4::Cap**< void > **get\_rcv\_cap** (int i) const override  
*Get capability slot allocated to the given receive buffer.*
- int **realloc\_rcv\_cap** (int i) override  
*Allocate a new capability for the given receive buffer.*
- void **set\_rcv\_cap\_flags** (unsigned long flags)  
*Set the receive flags for the buffers.*
- int **add\_timeout** (**L4::lpc\_svr::Timeout** \*, **l4\_kernel\_clock\_t**) override  
*No timeouts handled by us.*

- int **remove\_timeout** ([L4::lpc\\_svr::Timeout](#) \*) override  
*No timeouts handled by us.*
- void **setup\_wait** ([l4\\_utcb\\_t](#) \*utcb, [L4::lpc\\_svr::Reply\\_mode](#))  
*setup\_wait() used the server loop ([L4::Server](#))*

## Public Member Functions inherited from [L4::lpc\\_svr::Server\\_iface](#)

- **Server\_iface** ()  
*Make a server interface.*
- template<typename T>  
[L4::Cap](#)< T > **rcv\_cap** (int index) const  
*Get given receive buffer as typed capability.*
- [L4::Cap](#)< void > **rcv\_cap** (int index) const  
*Get receive cap with the given index as generic (void) type.*

## Protected Member Functions

- unsigned **first\_free\_br** () const  
*Used for assigning BRs for a timeout.*

## Additional Inherited Members

## Public Types inherited from [L4::lpc\\_svr::Server\\_iface](#)

- typedef [L4::Type\\_info::Demand](#) **Demand**  
*Data type expressing server-side demand for receive buffers.*

### 16.314.1 Detailed Description

Buffer-register (BR) manager for [L4::Server](#).

Implementation of the [L4::lpc\\_svr::Server\\_iface](#) API for managing the server-side receive buffers needed for a set of server objects running within a server.

Definition at line 25 of file [br\\_manager](#).

### 16.314.2 Member Function Documentation

#### 16.314.2.1 **alloc\_buffer\_demand()**

```
int L4Re::Util::Br_manager::alloc_buffer_demand (
    Demand const & demand) [inline], [override], [virtual]
```

Tells the server to allocate buffers for the given demand.

#### Parameters

---

<i>demand</i>	The total server-side demand of receive buffers needed for a given interface, see <a href="#">Demand</a> .
---------------	------------------------------------------------------------------------------------------------------------

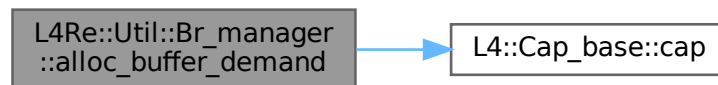
This function is not called by user applications directly. Usually the server implementation or the registry implementation calls this function whenever a new object is registered at the server.

Implements [L4::lpc\\_svr::Server\\_iface](#).

Definition at line 54 of file [br\\_manager](#).

References [L4::Cap\\_base::cap\(\)](#), [L4Re::Util::cap\\_alloc](#), [L4::Type\\_info::Demand::caps](#), [L4\\_EINVAL](#), [L4\\_ENOMEM](#), [L4\\_EOK](#), [L4\\_ERANGE](#), [L4::Type\\_info::Demand::mem](#), and [L4::Type\\_info::Demand::ports](#).

Here is the call graph for this function:



### 16.314.2.2 get\_rcv\_cap()

```
L4::Cap< void > L4Re::Util::Br_manager::get_rcv_cap (
    int index) const [inline], [override], [virtual]
```

Get capability slot allocated to the given receive buffer.

#### Parameters

<i>index</i>	The receive buffer index of the expected capability argument ( $0 \leq \text{index} < \text{caps}$ registered with <a href="#">alloc_buffer_demand()</a> ).
--------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------

#### Precondition

$0 \leq \text{index} < \text{caps}$  registered with [alloc\\_buffer\\_demand\(\)](#)

#### Returns

Capability slot currently allocated to the given receive buffer.

Implements [L4::lpc\\_svr::Server\\_iface](#).

Definition at line 86 of file [br\\_manager](#).

References [L4::Cap\\_base::Invalid](#), and [L4\\_CAP\\_MASK](#).

### 16.314.2.3 realloc\_rcv\_cap()

```
int L4Re::Util::Br_manager::realloc_rcv_cap (
    int index) [inline], [override], [virtual]
```

Allocate a new capability for the given receive buffer.

#### Parameters

<i>index</i>	The receive buffer index of the expected capability argument ( $0 \leq \text{index} < \text{caps}$ registered with <a href="#">alloc_buffer_demand()</a> ).
--------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------

**Precondition**

$0 \leq \text{index} < \text{caps}$  registered with [alloc\\_buffer\\_demand\(\)](#)

**Returns**

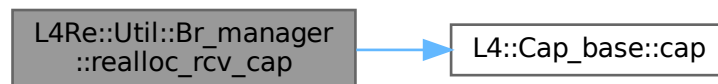
0 on success, < 0 on error.

Implements [L4::lpc\\_svr::Server\\_iface](#).

Definition at line 94 of file [br\\_manager](#).

References [L4::Cap\\_base::cap\(\)](#), [L4Re::Util::cap\\_alloc](#), [L4\\_EINVAL](#), [L4\\_ENOMEM](#), and [L4\\_EOK](#).

Here is the call graph for this function:

**16.314.2.4 set\_rcv\_cap\_flags()**

```
void L4Re::Util::Br_manager::set_rcv_cap_flags (
    unsigned long flags) [inline]
```

Set the receive flags for the buffers.

**Precondition**

Must be called before any handlers are registered.

**Parameters**

<i>flags</i>	New receive capability flags, see <a href="#">l4_msg_item_consts_t</a> .
--------------	--------------------------------------------------------------------------

Definition at line 118 of file [br\\_manager](#).

References [l4\\_assert](#).

The documentation for this class was generated from the following file:

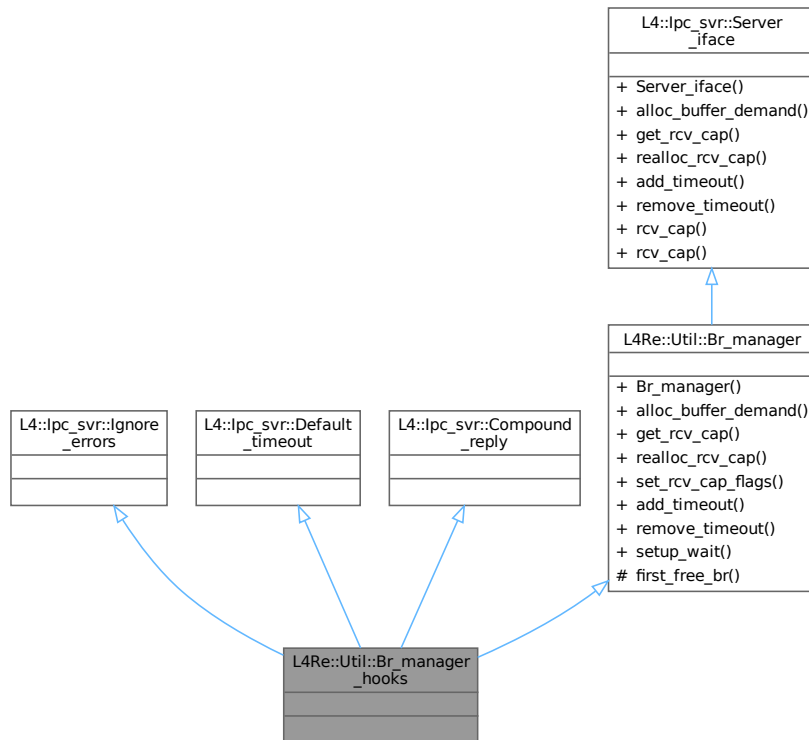
- `l4/re/util/br_manager`

## 16.315 L4Re::Util::Br\_manager\_hooks Struct Reference

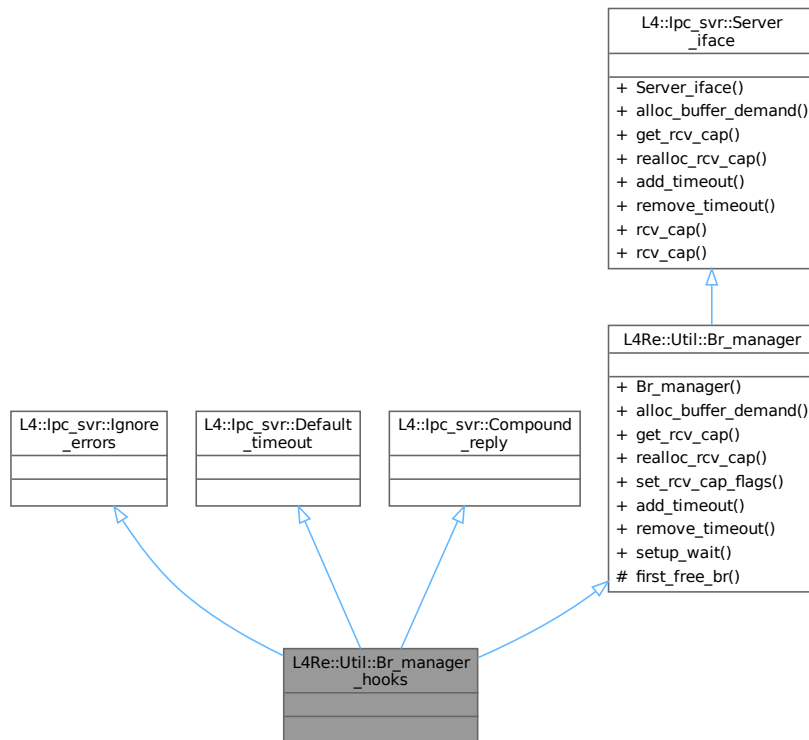
Predefined server-loop hooks for a server loop using the [Br\\_manager](#).

```
#include <br_manager>
```

Inheritance diagram for L4Re::Util::Br\_manager\_hooks:



Collaboration diagram for L4Re::Util::Br\_manager\_hooks:



### Additional Inherited Members

### Public Types inherited from L4::lpc\_svr::Server\_iface

- typedef L4::Type\_info::Demand Demand  
*Data type expressing server-side demand for receive buffers.*

### Public Member Functions inherited from L4Re::Util::Br\_manager

- **Br\_manager** ()  
*Make a buffer-register (BR) manager.*
- int **alloc\_buffer\_demand** (Demand const &d) override  
*Tells the server to allocate buffers for the given demand.*
- L4::Cap< void > **get\_rcv\_cap** (int i) const override  
*Get capability slot allocated to the given receive buffer.*
- int **realloc\_rcv\_cap** (int i) override  
*Allocate a new capability for the given receive buffer.*
- void **set\_rcv\_cap\_flags** (unsigned long flags)  
*Set the receive flags for the buffers.*
- int **add\_timeout** (L4::lpc\_svr::Timeout \*, l4\_kernel\_clock\_t) override  
*No timeouts handled by us.*
- int **remove\_timeout** (L4::lpc\_svr::Timeout \*) override  
*No timeouts handled by us.*
- void **setup\_wait** (l4\_utcb\_t \*utcb, L4::lpc\_svr::Reply\_mode)  
*setup\_wait() used the server loop (L4::Server)*

## Public Member Functions inherited from [L4::lpc\\_svr::Server\\_iface](#)

- **Server\_iface** ()  
*Make a server interface.*
- `template<typename T>`  
[L4::Cap](#)< T > [rcv\\_cap](#) (int index) const  
*Get given receive buffer as typed capability.*
- [L4::Cap](#)< void > [rcv\\_cap](#) (int index) const  
*Get receive cap with the given index as generic (void) type.*

## Protected Member Functions inherited from [L4Re::Util::Br\\_manager](#)

- `unsigned first_free_br` () const  
*Used for assigning BRs for a timeout.*

### 16.315.1 Detailed Description

Predefined server-loop hooks for a server loop using the [Br\\_manager](#).

This class can be used whenever a server loop including full management of receive buffer resources is needed.

Definition at line 165 of file [br\\_manager](#).

The documentation for this struct was generated from the following file:

- `l4/re/util/br_manager`

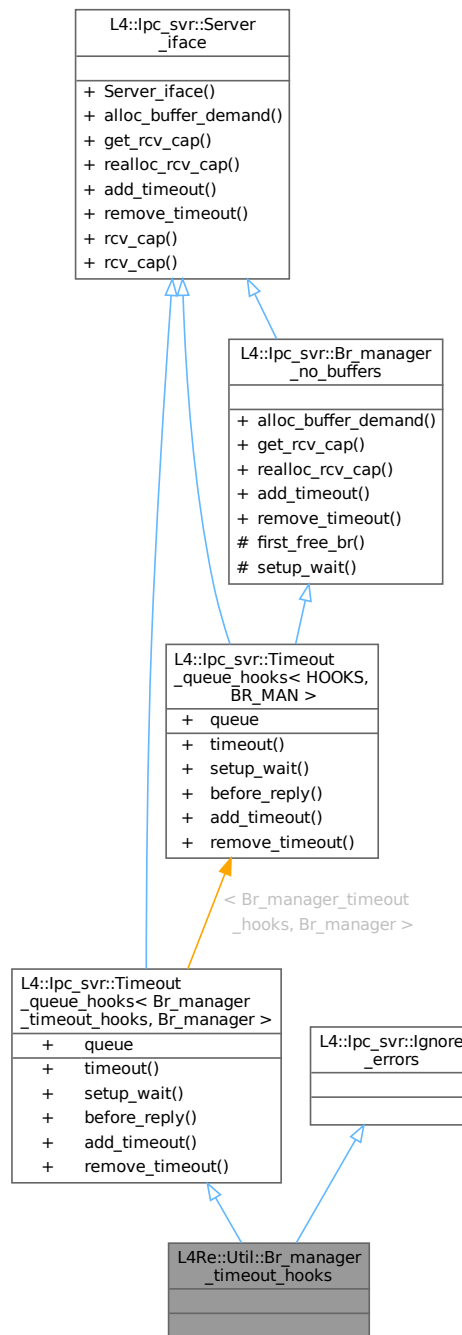
### 16.316 [L4Re::Util::Br\\_manager\\_timeout\\_hooks](#) Struct Reference

Predefined server-loop hooks for a server with using the [Br\\_manager](#) and a timeout queue.

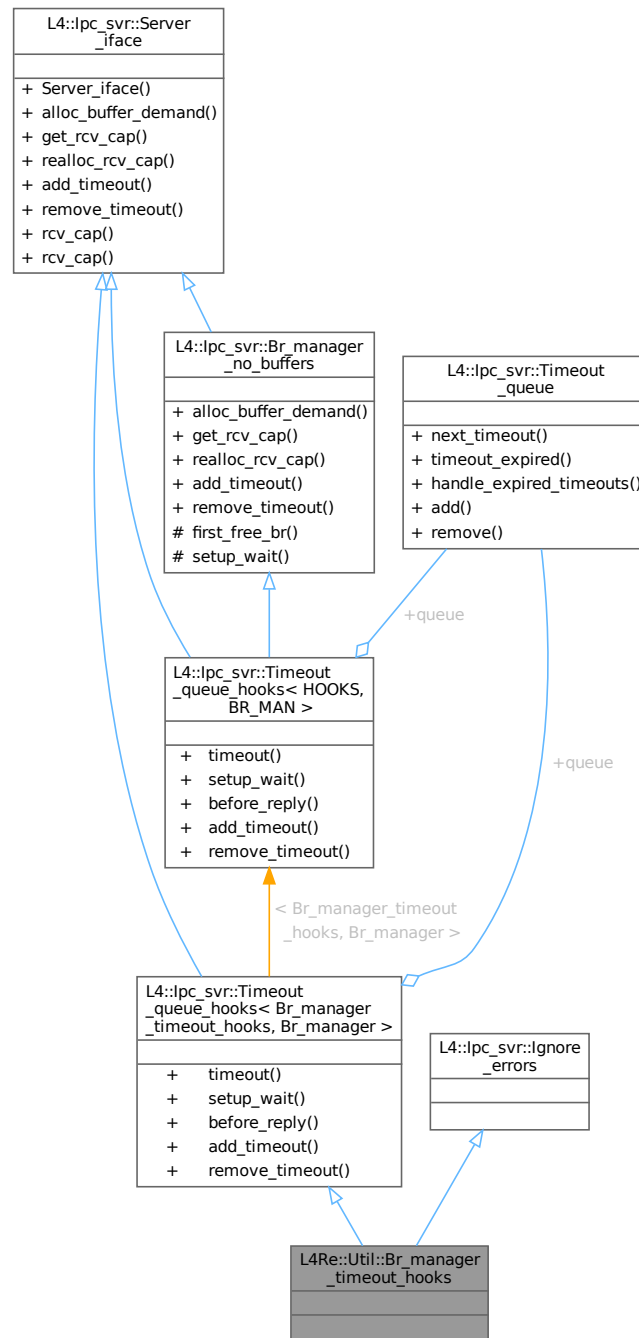
```
#include <br_manager>
```



Inheritance diagram for L4Re::Util::Br\_manager\_timeout\_hooks:



Collaboration diagram for L4Re::Util::Br\_manager\_timeout\_hooks:



### Additional Inherited Members

### Public Types inherited from L4::lpc\_svr::Server\_iface

- typedef L4::Type\_info::Demand Demand  
Data type expressing server-side demand for receive buffers.

**Public Member Functions inherited from****L4::lpc\_svr::Timeout\_queue\_hooks< Br\_manager\_timeout\_hooks, Br\_manager >**

- **l4\_timeout\_t** **timeout** ()  
*get the time for the next timeout*
- void **setup\_wait** (l4\_utcb\_t \*utcb, L4::lpc\_svr::Reply\_mode mode)  
*setup\_wait() for the server loop*
- L4::lpc\_svr::Reply\_mode **before\_reply** (l4\_msgtag\_t, l4\_utcb\_t \*)  
*server loop hook*
- int **add\_timeout** (Timeout \*timeout, l4\_kernel\_clock\_t time) override  
*Add a timeout to the queue for time time.*
- int **remove\_timeout** (Timeout \*timeout) override  
*Remove timeout from the queue.*

**Public Member Functions inherited from L4::lpc\_svr::Server\_iface**

- **Server\_iface** ()  
*Make a server interface.*
- virtual int **alloc\_buffer\_demand** (Demand const &demand)=0  
*Tells the server to allocate buffers for the given demand.*
- virtual L4::Cap< void > **get\_rcv\_cap** (int index) const =0  
*Get capability slot allocated to the given receive buffer.*
- virtual int **realloc\_rcv\_cap** (int index)=0  
*Allocate a new capability for the given receive buffer.*
- template<typename T>  
L4::Cap< T > **rcv\_cap** (int index) const  
*Get given receive buffer as typed capability.*
- L4::Cap< void > **rcv\_cap** (int index) const  
*Get receive cap with the given index as generic (void) type.*

**Data Fields inherited from****L4::lpc\_svr::Timeout\_queue\_hooks< Br\_manager\_timeout\_hooks, Br\_manager >**

- **Timeout\_queue** **queue**  
*Use this timeout queue.*

**16.316.1 Detailed Description**

Predefined server-loop hooks for a server with using the [Br\\_manager](#) and a timeout queue.

This class can be used for server loops that need the full package of buffer-register management and a timeout queue.

Definition at line 179 of file [br\\_manager](#).

The documentation for this struct was generated from the following file:

- l4/re/util/br\_manager

## 16.317 L4Re::Util::Cap\_alloc\_base Class Reference

Capability allocator.

```
#include <bitmap_cap_alloc>
```

Collaboration diagram for L4Re::Util::Cap\_alloc\_base:

L4Re::Util::Cap_alloc_base	
+	alloc()
+	free()

### Public Member Functions

- template<typename T>  
[L4::Cap](#)< T > **alloc** () noexcept  
*Allocate a capability slot.*
- template<typename T>  
void **free** ([L4::Cap](#)< T > const &cap, [l4\\_cap\\_idx\\_t](#) task=[L4\\_INVALID\\_CAP](#), [l4\\_umword\\_t](#) unmap\_↔  
flags=[L4\\_FP\\_ALL\\_SPACES](#)) noexcept  
*Free a capability slot.*

### 16.317.1 Detailed Description

Capability allocator.

Definition at line 28 of file [bitmap\\_cap\\_alloc](#).

The documentation for this class was generated from the following file:

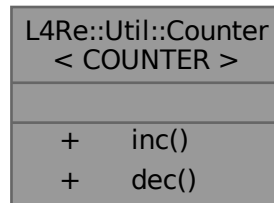
- [l4/re/util/bitmap\\_cap\\_alloc](#)

## 16.318 L4Re::Util::Counter< COUNTER > Struct Template Reference

[Counter](#) for [Counting\\_cap\\_alloc](#) with variable data width.

```
#include <counting_cap_alloc>
```

Collaboration diagram for L4Re::Util::Counter< COUNTER >:



### Public Member Functions

- bool [inc](#) ()  
*Increment counter if not yet saturated.*
- Type [dec](#) ()  
*Decrement counter if not saturated.*

### 16.318.1 Detailed Description

```
template<typename COUNTER = unsigned char>
struct L4Re::Util::Counter< COUNTER >
```

[Counter](#) for [Counting\\_cap\\_alloc](#) with variable data width.

This version is not thread safe.

Definition at line 27 of file [counting\\_cap\\_alloc](#).

### 16.318.2 Member Function Documentation

#### 16.318.2.1 dec()

```
template<typename COUNTER = unsigned char>
Type L4Re::Util::Counter< COUNTER >::dec () [inline]
```

Decrement counter if not saturated.

Once the counter reached the saturated state, the counter value isn't changed.

Definition at line 67 of file [counting\\_cap\\_alloc](#).

### 16.318.2.2 inc()

```
template<typename COUNTER = unsigned char>  
bool L4Re::Util::Counter< COUNTER >::inc () [inline]
```

Increment counter if not yet saturated.

Once the counter reached the saturated state, the counter value isn't changed.

#### Return values

---

<i>false</i>	The counter just went saturated after it was increased.
<i>true</i>	Either the counter was already saturated – in that case the counter value was not changed, or the counter was not saturated – in that case the counter was increased.

Definition at line 50 of file [counting\\_cap\\_alloc](#).

The documentation for this struct was generated from the following file:

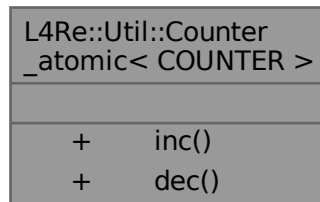
- [l4/re/util/counting\\_cap\\_alloc](#)

## 16.319 L4Re::Util::Counter\_atomic< COUNTER > Struct Template Reference

Thread safe version of counter for [Counting\\_cap\\_alloc](#).

```
#include <counting_cap_alloc>
```

Collaboration diagram for L4Re::Util::Counter\_atomic< COUNTER >:



### Public Member Functions

- bool [inc](#) ()  
*Increment counter if not yet saturated.*
- Type [dec](#) ()  
*Decrement counter if not saturated.*

### 16.319.1 Detailed Description

```
template<typename COUNTER = unsigned char>
struct L4Re::Util::Counter_atomic< COUNTER >
```

Thread safe version of counter for [Counting\\_cap\\_alloc](#).

Despite using atomic instructions, this version has to make sure that capability slots are not reused too early. If the last reference is gone, the capability slot has to be unmapped. The slot must only be allocated again when the unmap has completed. This is accomplished by starting with an initial count of 2. The last reference will decrease the counter to 1. Only then, after the slot was unmapped, will the counter be set to 0. This will allow other threads to reallocate the slot.

Definition at line 98 of file [counting\\_cap\\_alloc](#).

## 16.319.2 Member Function Documentation

### 16.319.2.1 dec()

```
template<typename COUNTER = unsigned char>
Type L4Re::Util::Counter_atomic< COUNTER >::dec () [inline]
```

Decrement counter if not saturated.

Once the counter reached the saturated state, the counter value isn't changed.

Definition at line 142 of file [counting\\_cap\\_alloc](#).

### 16.319.2.2 inc()

```
template<typename COUNTER = unsigned char>
bool L4Re::Util::Counter_atomic< COUNTER >::inc () [inline]
```

Increment counter if not yet saturated.

Once the counter reached the saturated state, the counter value isn't changed.

#### Return values

<i>false</i>	The counter just went saturated after it was increased.
<i>true</i>	Either the counter was already saturated – in that case the counter value was not changed, or the counter was not saturated – in that case the counter was increased.

Definition at line 121 of file [counting\\_cap\\_alloc](#).

The documentation for this struct was generated from the following file:

- [l4/re/util/counting\\_cap\\_alloc](#)

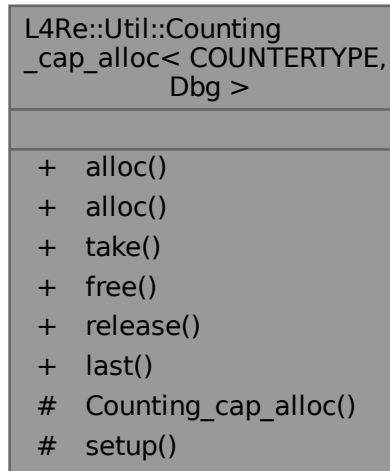
## 16.320 L4Re::Util::Counting\_cap\_alloc< COUNTERTYPE, Dbg > Class Template Reference

Internal reference-counting cap allocator.

```
#include <counting_cap_alloc>
```



Collaboration diagram for L4Re::Util::Counting\_cap\_alloc< COUNTERTYPE, Dbg >:



## Public Member Functions

- `L4::Cap< void > alloc () noexcept`  
*Allocate a new capability slot.*
- `template<typename T>`  
`L4::Cap< T > alloc () noexcept`  
*Allocate a new capability slot.*
- `void take (L4::Cap< void > cap) noexcept`  
*Increase the reference counter for the capability.*
- `bool free (L4::Cap< void > cap, l4_cap_idx_t task=L4_INVALID_CAP, unsigned unmap_flags=L4_FP_ALL_SPACES) noexcept`  
*Free the capability.*
- `bool release (L4::Cap< void > cap, l4_cap_idx_t task=L4_INVALID_CAP, unsigned unmap_flags=L4_FP_ALL_SPACES) noexcept`  
*Decrease the reference counter for a capability.*
- `long last () noexcept`  
*Return highest capability id managed by this allocator.*

## Protected Member Functions

- `Counting_cap_alloc () noexcept`  
*Create a new, empty allocator.*
- `void setup (void *m, long capacity, long bias, Dbg *dbg) noexcept`  
*Set up the backing memory for the allocator and the area of managed capability slots.*

### 16.320.1 Detailed Description

```
template<typename COUNTERTYPE, typename Dbg>
class L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >
```

Internal reference-counting cap allocator.

This is intended for internal use only. [L4Re](#) applications should use [L4Re::Util::cap\\_alloc\(\)](#).

Allocator for capability slots that automatically frees the slot and optionally unmaps the capability when the reference count goes down to zero. Reference counting must be done manually via [take\(\)](#) and [release\(\)](#). The backing store for the reference counters must be provided in the [setup\(\)](#) method. The allocator can recognize capability slots that are not managed by itself and does nothing on such slots.

#### Note

The user must ensure that the backing store is zero-initialized.

The user must ensure that the capability slots managed by this allocator are not used by a different allocator, see [setup\(\)](#).

The operations in this class are not thread-safe.

Definition at line [191](#) of file [counting\\_cap\\_alloc](#).

### 16.320.2 Constructor & Destructor Documentation

#### 16.320.2.1 Counting\_cap\_alloc()

```
template<typename COUNTERTYPE, typename Dbg>
L4Re::Util::Counting\_cap\_alloc< COUNTERTYPE, Dbg >::Counting_cap_alloc () [inline], [protected],
[noexcept]
```

Create a new, empty allocator.

Needs to be initialized with [setup\(\)](#) before it can be used.

Definition at line [224](#) of file [counting\\_cap\\_alloc](#).

### 16.320.3 Member Function Documentation

#### 16.320.3.1 alloc() [1/2]

```
template<typename COUNTERTYPE, typename Dbg>
template<typename T>
L4::Cap< T > L4Re::Util::Counting\_cap\_alloc< COUNTERTYPE, Dbg >::alloc () [inline], [noexcept]
```

Allocate a new capability slot.

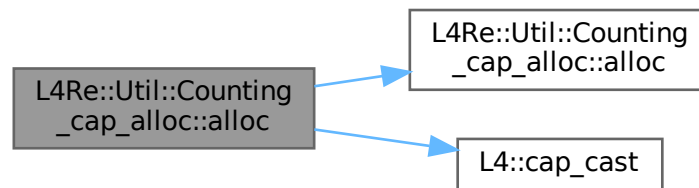
**Returns**

The newly allocated capability slot, invalid if the allocator was exhausted.

Definition at line 282 of file [counting\\_cap\\_alloc](#).

References [alloc\(\)](#), and [L4::cap\\_cast\(\)](#).

Here is the call graph for this function:

**16.320.3.2 alloc() [2/2]**

```
template<typename COUNTERTYPE, typename Dbg>
L4::Cap< void > L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >::alloc () [inline], [noexcept]
```

Allocate a new capability slot.

**Returns**

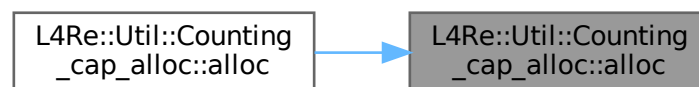
The newly allocated capability slot, invalid if the allocator was exhausted.

Definition at line 257 of file [counting\\_cap\\_alloc](#).

References [L4::Cap\\_base::Invalid](#), and [L4\\_CAP\\_SHIFT](#).

Referenced by [alloc\(\)](#).

Here is the caller graph for this function:



### 16.320.3.3 free()

```
template<typename COUNTERTYPE, typename Dbg>
bool L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >::free (
    L4::Cap< void > cap,
    l4_cap_idx_t task = L4_INVALID_CAP,
    unsigned unmap_flags = L4_FP_ALL_SPACES) [inline], [noexcept]
```

Free the capability.

#### Parameters

<i>cap</i>	Capability to free.
<i>task</i>	If set, task to unmap the capability from.
<i>unmap_flags</i>	Flags for unmap, see <a href="#">l4_unmap_flags_t</a> .

#### Precondition

The capability has been allocated. Calling free twice on a capability managed by this allocator results in undefined behaviour.

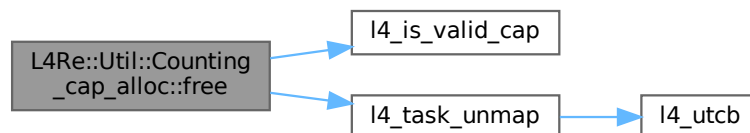
#### Returns

True, if the capability was managed by this allocator.

Definition at line 321 of file [counting\\_cap\\_alloc](#).

References [l4\\_assert](#), [L4\\_FP\\_ALL\\_SPACES](#), [L4\\_INVALID\\_CAP](#), [l4\\_is\\_valid\\_cap\(\)](#), and [l4\\_task\\_unmap\(\)](#).

Here is the call graph for this function:



### 16.320.3.4 release()

```
template<typename COUNTERTYPE, typename Dbg>
bool L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >::release (
    L4::Cap< void > cap,
    l4_cap_idx_t task = L4_INVALID_CAP,
    unsigned unmap_flags = L4_FP_ALL_SPACES) [inline], [noexcept]
```

Decrease the reference counter for a capability.

#### Parameters

<i>cap</i>	Capability to release.
<i>task</i>	If set, task to unmap the capability from.
<i>unmap_flags</i>	Flags for unmap, see <a href="#">l4_unmap_flags_t</a> .

**Precondition**

The capability has been allocated. Calling release on a free capability results in undefined behaviour.

**Returns**

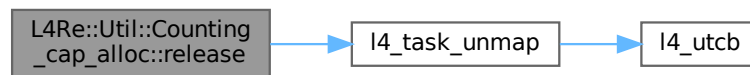
True, if the capability was freed as a result of this operation. If false is returned the capability is either still in use or is not managed by this allocator.

Does nothing apart from returning false if the capability is not managed by this allocator.

Definition at line 359 of file [counting\\_cap\\_alloc](#).

References [l4\\_assert](#), [L4\\_FP\\_ALL\\_SPACES](#), [L4\\_INVALID\\_CAP](#), and [l4\\_task\\_unmap\(\)](#).

Here is the call graph for this function:

**16.320.3.5 setup()**

```

template<typename COUNTERTYPE, typename Dbg>
void L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >::setup (
    void * m,
    long capacity,
    long bias,
    Dbg * dbg) [inline], [protected], [noexcept]
  
```

Set up the backing memory for the allocator and the area of managed capability slots.

**Parameters**

<i>m</i>	Pointer to backing memory.
<i>capacity</i>	Number of capabilities that can be stored.
<i>bias</i>	First capability id to use by this allocator.
<i>dbg</i>	Logger for warnings if counter got saturated.

The allocator will manage the capability slots between `bias` and `bias + capacity - 1` (inclusive). It is the responsibility of the user to ensure that these slots are not used otherwise.

Definition at line 242 of file [counting\\_cap\\_alloc](#).

### 16.320.3.6 take()

```
template<typename COUNTERTYPE, typename Dbg>
void L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >::take (
    L4::Cap< void > cap) [inline], [noexcept]
```

Increase the reference counter for the capability.

#### Parameters

<i>cap</i>	Capability, whose reference counter should be increased.
------------	----------------------------------------------------------

If the capability was still free, it will be automatically allocated. Silently does nothing if the capability is not managed by this allocator.

Definition at line 296 of file [counting\\_cap\\_alloc](#).

References [L4\\_CAP\\_SHIFT](#), and [L4\\_UNLIKELY](#).

The documentation for this class was generated from the following file:

- [l4/re/util/counting\\_cap\\_alloc](#)

## 16.321 L4Re::Util::Dataspace\_svr Class Reference

[Dataspace](#) server class.

```
#include <dataspace_svr>
```

Collaboration diagram for L4Re::Util::Dataspace\_svr:

L4Re::Util::Dataspace_svr	
+	map()
+	map_hook()
+	take()
+	release()
+	copy()
+	clear()
+	allocate()
+	page_shift()
+	is_static()
+	map_info()

## Public Member Functions

- int [map](#) (Dataspace::Offset offset, Dataspace::Map\_addr local\_addr, Dataspace::Flags flags, Dataspace::↔ Map\_addr min\_addr, Dataspace::Map\_addr max\_addr, [L4::lpc::Snd\\_fpage](#) &memory)  
*Map a region of the dataspace.*
- virtual int [map\\_hook](#) (Dataspace::Offset offs, unsigned order, Dataspace::Flags flags, Dataspace::Map\_addr \*base, unsigned \*send\_order)  
*A hook that is called for acquiring the data to be mapped.*
- virtual void [take](#) () noexcept  
*Take a reference to this dataspace.*
- virtual unsigned long [release](#) () noexcept  
*Release a reference to this dataspace.*
- virtual long [copy](#) ([l4\\_addr\\_t](#) dst\_offs, [l4\\_umword\\_t](#) src\_id, [l4\\_addr\\_t](#) src\_offs, unsigned long size) noexcept  
*Copy from src dataspace to this destination dataspace.*
- virtual long [clear](#) (unsigned long offs, unsigned long size) const noexcept  
*Clear a region in the dataspace.*
- virtual long [allocate](#) ([l4\\_addr\\_t](#) offset, [l4\\_size\\_t](#) size, unsigned access) noexcept  
*Allocate a region within a dataspace.*
- virtual unsigned long [page\\_shift](#) () const noexcept  
*Define the size of the flexpage to map.*
- virtual bool [is\\_static](#) () const noexcept  
*Return whether the dataspace is static.*
- virtual long [map\\_info](#) ([l4\\_addr\\_t](#) &start\_addr, [l4\\_addr\\_t](#) &end\_addr) noexcept  
*Return mapping information for no-MMU systems.*

### 16.321.1 Detailed Description

[Dataspace](#) server class.

The default implementation of the interface provides a continuous dataspace with contiguous pages.

Definition at line 29 of file [dataspace\\_svr](#).

### 16.321.2 Member Function Documentation

#### 16.321.2.1 [allocate\(\)](#)

```
virtual long L4Re::Util::Dataspace_svr::allocate (
    l4\_addr\_t offset,
    l4\_size\_t size,
    unsigned access) [inline], [virtual], [noexcept]
```

Allocate a region within a dataspace.

#### Parameters

<i>offset</i>	Offset in the dataspace, in bytes.
<i>size</i>	Size of the range, in bytes.

<i>access</i>	Access mode with which the memory backing the dataspace region should be allocated.
---------------	-------------------------------------------------------------------------------------

### Return values

0	Success
<0	Error

Definition at line 224 of file [dataspace\\_svr](#).

References [L4\\_ENODEV](#).

### 16.321.2.2 clear()

```
virtual long L4Re::Util::Dataspace_svr::clear (  
    unsigned long offs,  
    unsigned long size) const [inline], [virtual], [noexcept]
```

Clear a region in the dataspace.

### Parameters

<i>offs</i>	Start of the region
<i>size</i>	Size of the region

### Return values

0	Success
<0	Error

Definition at line 192 of file [dataspace\\_svr](#).

References [L4\\_ERANGE](#).

### 16.321.2.3 copy()

```
virtual long L4Re::Util::Dataspace_svr::copy (  
    l4_addr_t dst_offs,  
    l4_umword_t src_id,  
    l4_addr_t src_offs,  
    unsigned long size) [inline], [virtual], [noexcept]
```

Copy from src dataspace to this destination dataspace.

### Parameters

---



<i>dst_offs</i>	Offset into the destination dataspace
<i>src_id</i>	Local id of the source dataspace
<i>src_offs</i>	Offset into the source dataspace
<i>size</i>	Number of bytes to copy

**Return values**

$\geq 0$	Number of bytes copied
$< 0$	An error occurred. The error code may depend on the implementation.

Definition at line 175 of file [dataspace\\_svr](#).

References [L4\\_ENODEV](#).

**16.321.2.4 is\_static()**

```
virtual bool L4Re::Util::Dataspace_svr::is_static () const [inline], [virtual], [noexcept]
```

Return whether the dataspace is static.

**Returns**

True if dataspace is static

Definition at line 244 of file [dataspace\\_svr](#).

**16.321.2.5 map()**

```
int L4Re::Util::Dataspace_svr::map (
    Dataspace::Offset offset,
    Dataspace::Map_addr local_addr,
    Dataspace::Flags flags,
    Dataspace::Map_addr min_addr,
    Dataspace::Map_addr max_addr,
    L4::Ipc::Snd_fpage & memory) [inline]
```

Map a region of the dataspace.

**Parameters**

	<i>offset</i>	Offset to start within data space
	<i>local_addr</i>	Local address to map to.
	<i>flags</i>	<a href="#">Dataspace</a> flags, see <a href="#">L4Re::Dataspace::F::Flags</a> .
	<i>min_addr</i>	Defines start of receive window.

	<i>max_addr</i>	Defines end of receive window.
out	<i>memory</i>	Send fpage to map

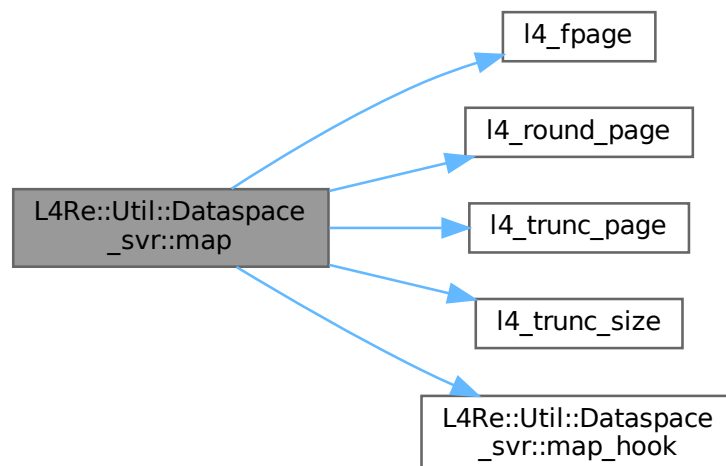
### Return values

0	Success
<0	Error

Definition at line 57 of file [dataspace\\_svr](#).

References [L4\\_EOK](#), [L4\\_ERANGE](#), [l4\\_fpage\(\)](#), [L4\\_PAGESHIFT](#), [l4\\_round\\_page\(\)](#), [l4\\_trunc\\_page\(\)](#), [l4\\_trunc\\_size\(\)](#), and [map\\_hook\(\)](#).

Here is the call graph for this function:



### 16.321.2.6 map\_hook()

```

virtual int L4Re::Util::Dataspace_svr::map_hook (
    Dataspace::Offset offs,
    unsigned order,
    Dataspace::Flags flags,
    Dataspace::Map_addr * base,
    unsigned * send_order) [inline], [virtual]

```

A hook that is called for acquiring the data to be mapped.

### Parameters

<i>offs</i>	Offset in dataspace to supply
<i>order</i>	Log2-size of data to supply
<i>flags</i>	Flags for the mapping
<i>base</i>	Start address of the flexpage to be mapped
<i>send_order</i>	Order (log2 of size) of the flexpage to be mapped

**Return values**

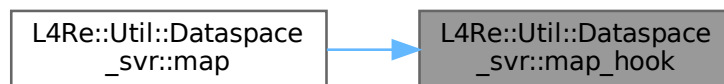
$< 0$	Error and the map request will be aborted with that error.
$\geq 0$	Success

**See also**[map](#)

Definition at line 136 of file [dataspace\\_svr](#).

Referenced by [map\(\)](#).

Here is the caller graph for this function:

**16.321.2.7 map\_info()**

```
virtual long L4Re::Util::Dataspace_svr::map_info (
    l4_addr_t & start_addr,
    l4_addr_t & end_addr) [inline], [virtual], [noexcept]
```

Return mapping information for no-MMU systems.

The method is only called on no-MMU systems. It should return the address of the underlying backing buffer so that the caller might map the dataspace.

The default implementation always returns an error because the derived class must provide the required information.

**See also**[L4Re::Dataspace::map\\_info\(\)](#)

Definition at line 259 of file [dataspace\\_svr](#).

References [L4\\_EPERM](#).

#### 16.321.2.8 `page_shift()`

```
virtual unsigned long L4Re::Util::Dataspace_svr::page_shift () const [inline], [virtual],  
[noexcept]
```

Define the size of the flexpage to map.

##### Returns

flexpage size

Definition at line 236 of file [dataspace\\_svr](#).

References [L4\\_LOG2\\_PAGESIZE](#).

#### 16.321.2.9 `release()`

```
virtual unsigned long L4Re::Util::Dataspace_svr::release () [inline], [virtual], [noexcept]
```

Release a reference to this dataspace.

##### Returns

Number of references to the dataspace

Default does nothing and returns always zero.

Definition at line 160 of file [dataspace\\_svr](#).

#### 16.321.2.10 `take()`

```
virtual void L4Re::Util::Dataspace_svr::take () [inline], [virtual], [noexcept]
```

Take a reference to this dataspace.

Default does nothing.

Definition at line 150 of file [dataspace\\_svr](#).

The documentation for this class was generated from the following file:

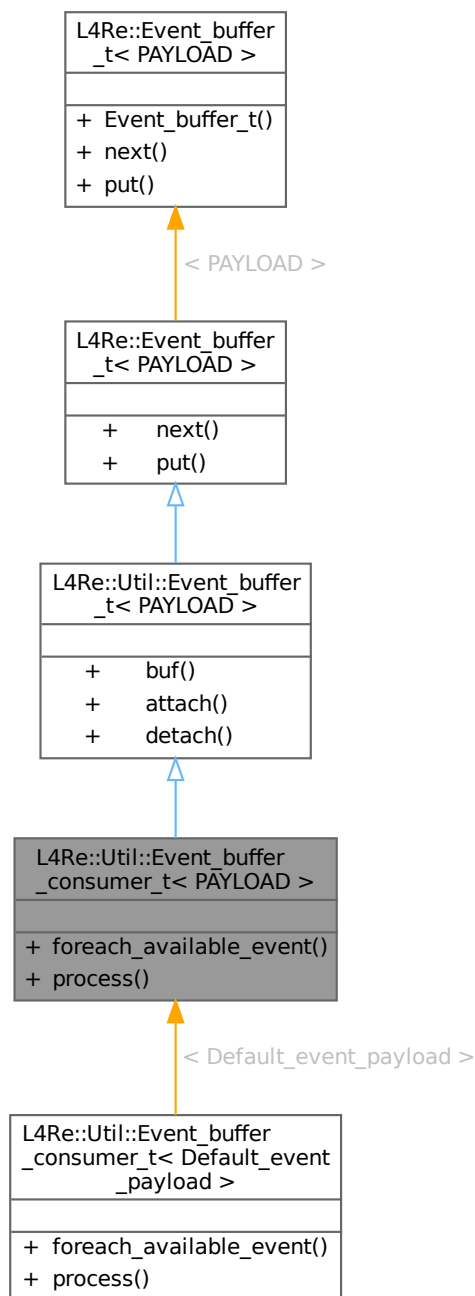
- `l4/re/util/dataspace_svr`

## 16.322 L4Re::Util::Event\_buffer\_consumer\_t< PAYLOAD > Class Template Reference

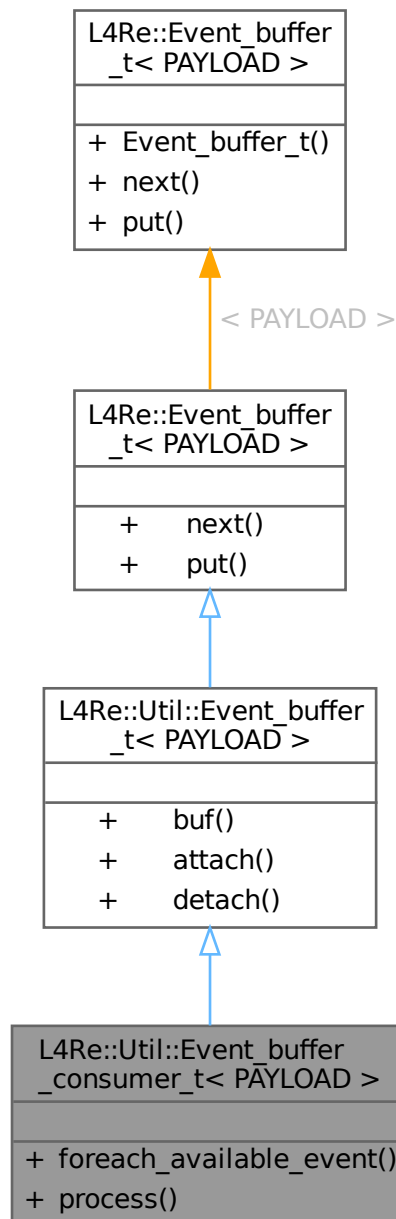
An event buffer consumer.

```
#include <event_buffer>
```

Inheritance diagram for L4Re::Util::Event\_buffer\_consumer\_t< PAYLOAD >:



Collaboration diagram for L4Re::Util::Event\_buffer\_consumer\_t< PAYLOAD >:



## Public Member Functions

- `template<typename CB, typename D>`  
`void foreach_available_event (CB const &cb, D data=D())`  
*Call function on every available event.*
- `template<typename CB, typename D>`  
`void process (L4::Cap< L4::Irq > irq, L4::Cap< L4::Thread > thread, CB const &cb, D data=D())`  
*Continuously wait for events and process them.*

**Public Member Functions inherited from [L4Re::Util::Event\\_buffer\\_t< PAYLOAD >](#)**

- void \* [buf](#) () const noexcept  
*Return the buffer.*
- long [attach](#) ([L4::Cap](#)< [L4Re::Dataspace](#) > ds, [L4::Cap](#)< [L4Re::Rm](#) > rm) noexcept  
*Attach event buffer from address space.*
- long [detach](#) ([L4::Cap](#)< [L4Re::Rm](#) > rm) noexcept  
*Detach event buffer from address space.*

**Public Member Functions inherited from [L4Re::Event\\_buffer\\_t< PAYLOAD >](#)**

- Event \* [next](#) () noexcept  
*Next event in buffer.*
- bool [put](#) (Event const &ev) noexcept  
*Put event into buffer at current position.*

**16.322.1 Detailed Description**

template<typename PAYLOAD>  
class L4Re::Util::Event\_buffer\_consumer\_t< PAYLOAD >

An event buffer consumer.

Definition at line 83 of file [event\\_buffer](#).

**16.322.2 Member Function Documentation****16.322.2.1 [foreach\\_available\\_event\(\)](#)**

```
template<typename PAYLOAD>
template<typename CB, typename D>
void L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::foreach_available_event (
    CB const & cb,
    D data = D()) [inline]
```

Call function on every available event.

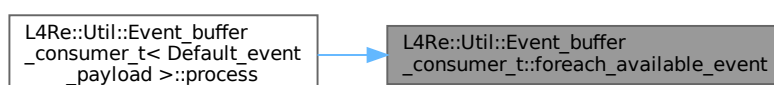
**Parameters**

<i>cb</i>	Function callback.
<i>data</i>	Data to pass as an argument to the callback.

Definition at line 94 of file [event\\_buffer](#).

Referenced by [L4Re::Util::Event\\_buffer\\_consumer\\_t< Default\\_event\\_payload >::process\(\)](#).

Here is the caller graph for this function:



### 16.322.2.2 process()

```
template<typename PAYLOAD>
template<typename CB, typename D>
void L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::process (
    L4::Cap< L4::Irq > irq,
    L4::Cap< L4::Thread > thread,
    CB const & cb,
    D data = D()) [inline]
```

Continuously wait for events and process them.

#### Parameters

<i>irq</i>	<a href="#">Event</a> signal to wait for.
<i>thread</i>	Thread capability of the thread calling this function.
<i>cb</i>	Callback function that is called for each received event.
<i>data</i>	Data to pass as an argument to the processing callback.

#### Note

This function never returns.

Definition at line 115 of file [event\\_buffer](#).

The documentation for this class was generated from the following file:

- l4/re/util/event\_buffer

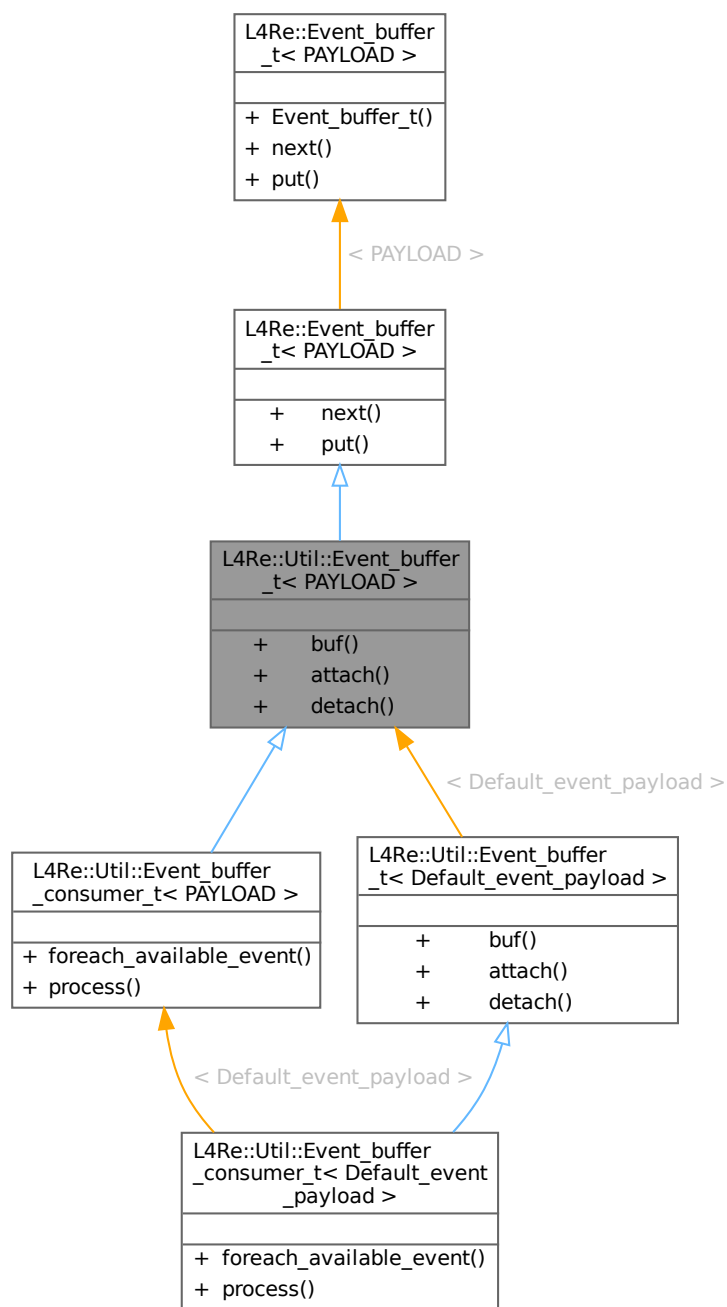
## 16.323 L4Re::Util::Event\_buffer\_t< PAYLOAD > Class Template Reference

Event\_buffer utility class.

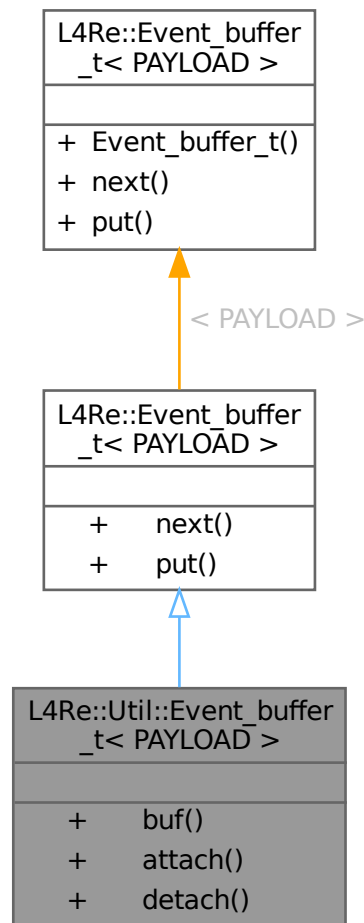
```
#include <event_buffer>
```



Inheritance diagram for L4Re::Util::Event\_buffer\_t< PAYLOAD >:



Collaboration diagram for L4Re::Util::Event\_buffer\_t< PAYLOAD >:



### Public Member Functions

- void \* `buf()` const noexcept  
*Return the buffer.*
- long `attach` (L4::Cap< L4Re::Dataspace > ds, L4::Cap< L4Re::Rm > rm) noexcept  
*Attach event buffer from address space.*
- long `detach` (L4::Cap< L4Re::Rm > rm) noexcept  
*Detach event buffer from address space.*

### Public Member Functions inherited from L4Re::Event\_buffer\_t< PAYLOAD >

- Event \* `next()` noexcept  
*Next event in buffer.*
- bool `put` (Event const &ev) noexcept  
*Put event into buffer at current position.*

### 16.323.1 Detailed Description

```
template<typename PAYLOAD>
class L4Re::Util::Event_buffer_t< PAYLOAD >
```

Event\_buffer utility class.

Definition at line 25 of file [event\\_buffer](#).

### 16.323.2 Member Function Documentation

#### 16.323.2.1 attach()

```
template<typename PAYLOAD>
long L4Re::Util::Event_buffer_t< PAYLOAD >::attach (
    L4::Cap< L4Re::Dataspace > ds,
    L4::Cap< L4Re::Rm > rm) [inline], [noexcept]
```

Attach event buffer from address space.

##### Parameters

<i>ds</i>	<a href="#">Dataspace</a> of the event buffer.
<i>rm</i>	Region manager to attach buffer to.

##### Returns

0 on success, negative error code otherwise.

Definition at line 45 of file [event\\_buffer](#).

#### 16.323.2.2 buf()

```
template<typename PAYLOAD>
void * L4Re::Util::Event_buffer_t< PAYLOAD >::buf () const [inline], [noexcept]
```

Return the buffer.

##### Returns

Pointer to the event buffer.

Definition at line 35 of file [event\\_buffer](#).

#### 16.323.2.3 detach()

```
template<typename PAYLOAD>
long L4Re::Util::Event_buffer_t< PAYLOAD >::detach (
    L4::Cap< L4Re::Rm > rm) [inline], [noexcept]
```

Detach event buffer from address space.

##### Parameters

<i>rm</i>	Region manager to detach buffer from.
-----------	---------------------------------------

#### Returns

0 on success, negative error code otherwise.

Definition at line 68 of file [event\\_buffer](#).

The documentation for this class was generated from the following file:

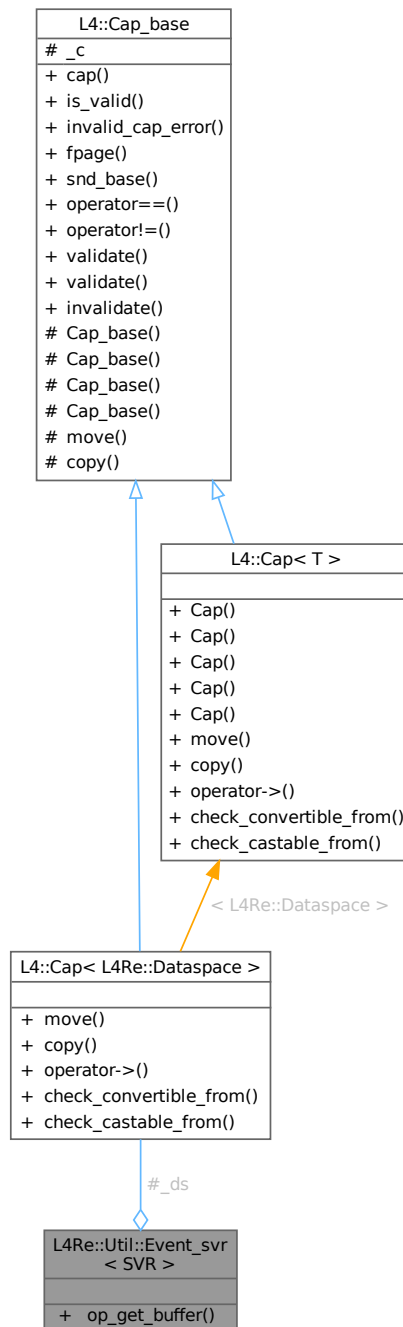
- l4/re/util/event\_buffer

## 16.324 L4Re::Util::Event\_svr< SVR > Class Template Reference

Convenience wrapper for implementing an event server.

```
#include <event_svr>
```

Collaboration diagram for L4Re::Util::Event\_svr< SVR >:



## Public Member Functions

- long **op\_get\_buffer** (L4Re::Event::Rights, [L4::lpc::Cap< L4Re::Dataspace >](#) &ds)  
Handle [L4Re::Event](#) protocol.

### 16.324.1 Detailed Description

```
template<typename SVR>
class L4Re::Util::Event_svr< SVR >
```

Convenience wrapper for implementing an event server.

See also

[L4Re::Event](#), [L4Re::Util::Event\\_t](#)

Definition at line 28 of file [event\\_svr](#).

The documentation for this class was generated from the following file:

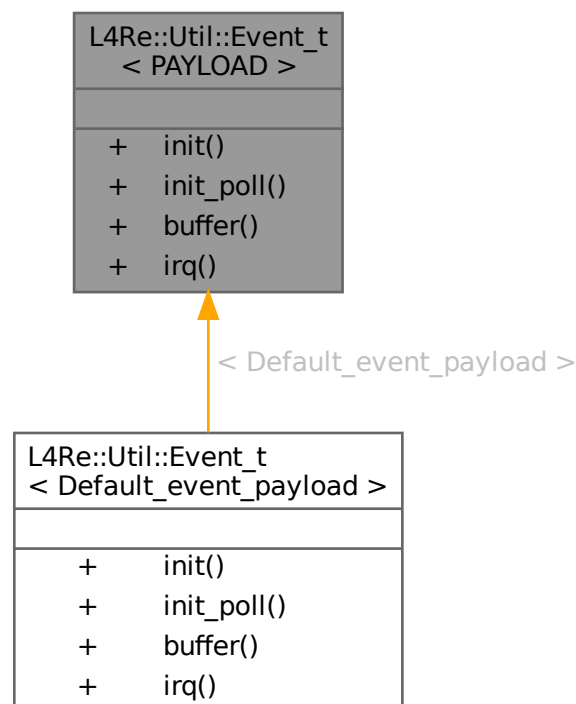
- [l4/re/util/event\\_svr](#)

## 16.325 L4Re::Util::Event\_t< PAYLOAD > Class Template Reference

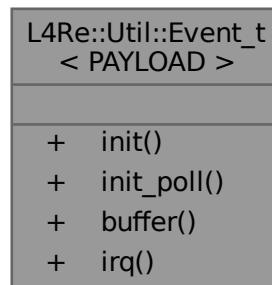
Convenience wrapper for getting access to an event object.

```
#include <event>
```

Inheritance diagram for L4Re::Util::Event\_t< PAYLOAD >:



Collaboration diagram for L4Re::Util::Event\_t< PAYLOAD >:



### Public Types

- enum [Mode](#) { [Mode\\_irq](#) , [Mode\\_polling](#) }  
*Modes of operation.*

### Public Member Functions

- template<typename IRQ\_TYPE>  
int [init](#) (L4::Cap< [L4Re::Event](#) > event, [L4Re::Env](#) const \*env=[L4Re::Env::env\(\)](#), [L4Re::Cap\\_alloc](#) \*ca=&[L4Re::Util::cap\\_alloc](#))  
*Initialise an event object.*
- int [init\\_poll](#) (L4::Cap< [L4Re::Event](#) > event, [L4Re::Env](#) const \*env=[L4Re::Env::env\(\)](#), [L4Re::Cap\\_alloc](#) \*ca=&[L4Re::Util::cap\\_alloc](#))  
*Initialise an event object in polling mode.*
- [L4Re::Event\\_buffer\\_t](#)< PAYLOAD > & [buffer](#) ()  
*Get event buffer.*
- [L4::Cap](#)< [L4::Triggerable](#) > [irq](#) () const  
*Get event IRQ.*

## 16.325.1 Detailed Description

```
template<typename PAYLOAD>
class L4Re::Util::Event_t< PAYLOAD >
```

Convenience wrapper for getting access to an event object.

After calling [init\(\)](#) the class supplies the event-buffer and the associated IRQ object.

Definition at line [32](#) of file [event](#).

## 16.325.2 Member Enumeration Documentation

### 16.325.2.1 Mode

```
template<typename PAYLOAD>  
enum L4Re::Util::Event_t::Mode
```

Modes of operation.

#### Enumerator

---



Mode_irq	Create an IRQ and attach, to get notifications.
Mode_polling	Do not use an IRQ.

Definition at line 38 of file [event](#).

## 16.325.3 Member Function Documentation

### 16.325.3.1 buffer()

```
template<typename PAYLOAD>
L4Re::Event_buffer_t< PAYLOAD > & L4Re::Util::Event_t< PAYLOAD >::buffer () [inline]
```

Get event buffer.

#### Returns

[Event](#) buffer object.

Definition at line 148 of file [event](#).

### 16.325.3.2 init()

```
template<typename PAYLOAD>
template<typename IRQ_TYPE>
int L4Re::Util::Event_t< PAYLOAD >::init (
    L4::Cap< L4Re::Event > event,
    L4Re::Env const * env = L4Re::Env::env(),
    L4Re::Cap_alloc * ca = &L4Re::Util::cap_alloc) [inline]
```

Initialise an event object.

#### Template Parameters

<i>IRQ_TYPE</i>	Type used for handling notifications from the event provider. This must be derived from <a href="#">L4::Triggerable</a> .
-----------------	---------------------------------------------------------------------------------------------------------------------------

#### Parameters

<i>event</i>	Capability to event.
<i>env</i>	Pointer to L4Re-Environment
<i>ca</i>	Pointer to capability allocator.

#### Return values

---

<i>0</i>	Success
<i>-L4_ENOMEM</i>	No memory to allocate required capabilities.
<i>&lt;0</i>	Other IPC errors.

Definition at line 59 of file [event](#).

### 16.325.3.3 `init_poll()`

```
template<typename PAYLOAD>
int L4Re::Util::Event_t< PAYLOAD >::init_poll (
    L4::Cap< L4Re::Event > event,
    L4Re::Env const * env = L4Re::Env::env(),
    L4Re::Cap_alloc * ca = &L4Re::Util::cap_alloc) [inline]
```

Initialise an event object in polling mode.

#### Parameters

<i>event</i>	Capability to event.
<i>env</i>	Pointer to L4Re-Environment
<i>ca</i>	Pointer to capability allocator.

#### Return values

<i>0</i>	Success
<i>-L4_ENOMEM</i>	No memory to allocate required capabilities.
<i>&lt;0</i>	Other IPC errors.

Definition at line 112 of file [event](#).

### 16.325.3.4 `irq()`

```
template<typename PAYLOAD>
L4::Cap< L4::Triggerable > L4Re::Util::Event_t< PAYLOAD >::irq () const [inline]
```

Get event IRQ.

#### Returns

[Event](#) IRQ.

Definition at line 155 of file [event](#).

The documentation for this class was generated from the following file:

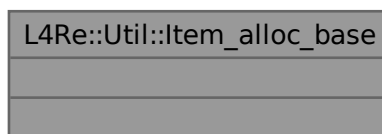
- [l4/re/util/event](#)

## 16.326 L4Re::Util::Item\_alloc\_base Class Reference

Item allocator.

```
#include <item_alloc>
```

Collaboration diagram for L4Re::Util::Item\_alloc\_base:



### 16.326.1 Detailed Description

Item allocator.

Definition at line 27 of file [item\\_alloc](#).

The documentation for this class was generated from the following file:

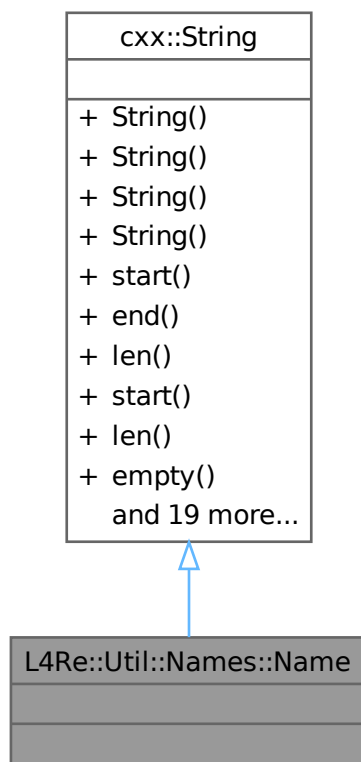
- [l4/re/util/item\\_alloc](#)

## 16.327 L4Re::Util::Names::Name Class Reference

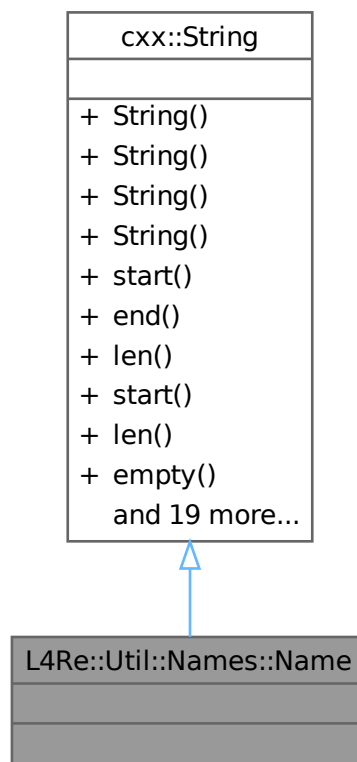
[Name](#) class.

```
#include <name_space_svr>
```

Inheritance diagram for L4Re::Util::Names::Name:



Collaboration diagram for L4Re::Util::Names::Name:



### Additional Inherited Members

### Public Types inherited from `cxx::String`

- `typedef char const * Index`  
*Character index type.*

### Public Member Functions inherited from `cxx::String`

- **`String`** (`char const *s`) `noexcept`  
*Initialize from a zero-terminated string.*
- **`String`** (`char const *s`, unsigned long `len`) `noexcept`  
*Initialize from a pointer to first character and a length.*
- **`String`** (`char const *s`, `char const *e`) `noexcept`  
*Initialize with start and end pointer.*
- **`String`** ()  
*Zero-initialize. Create an invalid string.*
- **`Index start`** () `const`  
*Pointer to first character.*

- **Index end** () const  
*Pointer to first byte behind the string.*
- int **len** () const  
*Length.*
- void **start** (char const \*s)  
*Set start.*
- void **len** (unsigned long len)  
*Set length.*
- bool **empty** () const  
*Check if the string has length zero.*
- **String head** (**Index end**) const  
*Return prefix up to index.*
- **String head** (unsigned long **end**) const  
*Prefix of length **end**.*
- **String substr** (unsigned long idx, unsigned long **len**=~0UL) const  
*Substring of length **len** starting at **idx**.*
- **String substr** (char const \***start**, unsigned long **len**=0) const  
*Substring of length **len** starting at **start**.*
- template<typename F>  
char const \* **find\_match** (F &&match) const  
*Find matching character. **match** should be a function such as **isspace**.*
- char const \* **find** (char const \*c) const  
*Find character. Return **end()** if not found.*
- char const \* **find** (int c) const  
*Find character. Return **end()** if not found.*
- char const \* **rfind** (char const \*c) const  
*Find right-most character. Return **end()** if not found.*
- **Index starts\_with** (cxx::String const &c) const  
*Check if **c** is a prefix of string.*
- char const \* **find** (int c, char const \*s) const  
*Find character **c** starting at position **s**. Return **end()** if not found.*
- char const \* **find** (char const \*c, char const \*s) const  
*Find character set at position.*
- char const & **operator[]** (unsigned long idx) const  
*Get character at **idx**.*
- char const & **operator[]** (int idx) const  
*Get character at **idx**.*
- char const & **operator[]** (**Index** idx) const  
*Get character at **idx**.*
- bool **eof** (char const \*s) const  
*Check if pointer **s** points behind string.*
- template<typename INT>  
int **from\_dec** (INT \*v) const  
*Convert decimal string to integer.*
- template<typename INT>  
int **from\_hex** (INT \*v) const  
*Convert hex string to integer.*
- bool **operator==** (**String** const &o) const  
*Equality.*
- bool **operator!=** (**String** const &o) const  
*Inequality.*

### 16.327.1 Detailed Description

[Name](#) class.

Definition at line 28 of file [name\\_space\\_svr](#).

The documentation for this class was generated from the following file:

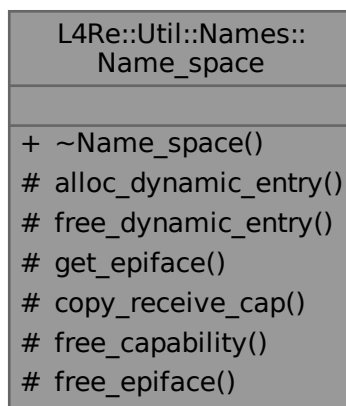
- [l4/re/util/name\\_space\\_svr](#)

## 16.328 L4Re::Util::Names::Name\_space Class Reference

Abstract server-side implementation of the L4::Namespace interface.

```
#include <name_space_svr>
```

Collaboration diagram for L4Re::Util::Names::Name\_space:



### Public Member Functions

- virtual **~Name\_space** ()  
*The destructor of the derived class is responsible for freeing resources.*

### Protected Member Functions

- virtual Entry \* **alloc\_dynamic\_entry** (Name const &n, unsigned flags)=0  
*Allocate a new entry for the given name.*
- virtual void **free\_dynamic\_entry** (Entry \*e)=0  
*Free an entry previously allocated with [alloc\\_dynamic\\_entry\(\)](#).*
- virtual int **get\_epiface** (l4\_umword\_t data, bool is\_local, L4::Epiface \*\*lo)=0  
*Return a pointer to the epiface assigned to a given label.*
- virtual int **copy\_receive\_cap** (L4::Cap< void > \*cap)=0  
*Return the receive capability for permanent use.*
- virtual void **free\_capability** (L4::Cap< void > cap)=0  
*Free a capability previously acquired with [copy\\_receive\\_cap\(\)](#).*
- virtual void **free\_epiface** (L4::Epiface \*epiface)=0  
*Free epiface previously acquired with [get\\_epiface\(\)](#).*

### 16.328.1 Detailed Description

Abstract server-side implementation of the L4::Namespace interface.

#### Note

The derived class is responsible for resource management through the provided interfaces. This includes freeing all resources on destruction!

Definition at line 180 of file [name\\_space\\_svr](#).

### 16.328.2 Member Function Documentation

#### 16.328.2.1 alloc\_dynamic\_entry()

```
virtual Entry * L4Re::Util::Names::Name_space::alloc_dynamic_entry (
    Name const & n,
    unsigned flags) [protected], [pure virtual]
```

Allocate a new entry for the given name.

#### Parameters

<i>n</i>	<a href="#">Name</a> of the entry, must be copied.
<i>flags</i>	Entry flags, see Obj::Flags.

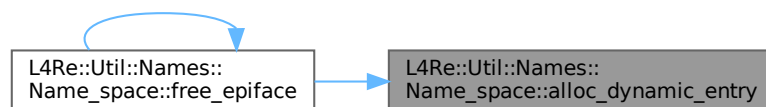
#### Returns

A pointer to the newly allocated entry or NULL on error.

This method is called when a new entry was received. It must allocate memory, copy *n* out of the receive buffer and wrap everything in an Entry.

Referenced by [free\\_epiface\(\)](#).

Here is the caller graph for this function:



#### 16.328.2.2 copy\_receive\_cap()

```
virtual int L4Re::Util::Names::Name_space::copy_receive_cap (
    L4::Cap< void > * cap) [protected], [pure virtual]
```

Return the receive capability for permanent use.

#### Parameters



out	cap	Capability slot with the received capability. Must be permanently available until <a href="#">free_capability()</a> is called.
-----	-----	--------------------------------------------------------------------------------------------------------------------------------

This method is called when a new entry is registered together with a capability mapping. It must decide whether and where to store the capability and return the final capability slot. Typical implementations return the capability slot in the receive window and allocate a new receive window.

### 16.328.2.3 free\_capability()

```
virtual void L4Re::Util::Names::Name_space::free_capability (
    L4::Cap< void > cap) [protected], [pure virtual]
```

Free a capability previously acquired with [copy\\_receive\\_cap\(\)](#).

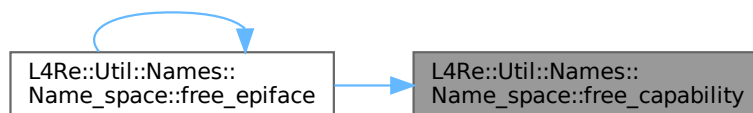
#### Parameters

in	cap	Capability to free.
----	-----	---------------------

Counterpart of [copy\\_receive\\_cap](#). Free the capability slot when the entry is deleted or changed.

Referenced by [free\\_epiface\(\)](#).

Here is the caller graph for this function:



### 16.328.2.4 free\_dynamic\_entry()

```
virtual void L4Re::Util::Names::Name_space::free_dynamic_entry (
    Entry * e) [protected], [pure virtual]
```

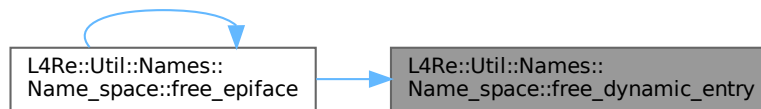
Free an entry previously allocated with [alloc\\_dynamic\\_entry\(\)](#).

#### Parameters

e	Entry to free.
---	----------------

Referenced by [free\\_epiface\(\)](#).

Here is the caller graph for this function:



### 16.328.2.5 free\_epiface()

```
virtual void L4Re::Util::Names::Name_space::free_epiface (
    L4::Epiface * epiface) [protected], [pure virtual]
```

Free epiface previously acquired with [get\\_epiface\(\)](#).

#### Parameters

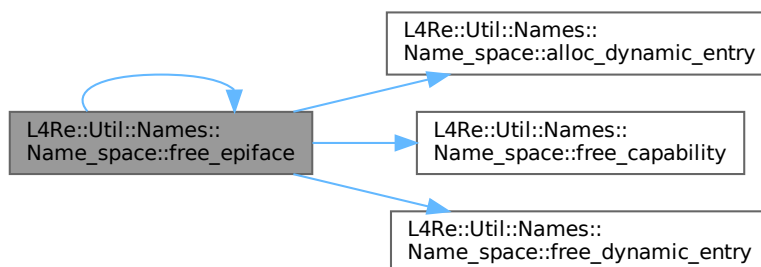
in	<i>epiface</i>	Epiface to free.
----	----------------	------------------

Called when an entry that points to an epiface is deleted allowing implementations that hold resources to free them.

References [alloc\\_dynamic\\_entry\(\)](#), [free\\_capability\(\)](#), [free\\_dynamic\\_entry\(\)](#), [free\\_epiface\(\)](#), [L4\\_BASE\\_TASK\\_CAP](#), [L4\\_EEXIST](#), [L4\\_ENOMEM](#), and [L4Re::Namespace::Overwrite](#).

Referenced by [free\\_epiface\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.328.2.6 get\_epiface()

```
virtual int L4Re::Util::Names::Name_space::get_epiface (
    l4_umword_t data,
    bool is_local,
    L4::Epiface ** lo) [protected], [pure virtual]
```

Return a pointer to the epiface assigned to a given label.

#### Parameters

in	<i>data</i>	Label or in the local case the capability slot of the receiving capability.
in	<i>is_local</i>	If true, a local capability slot was supplied, if false the label of a locally bound IPC gate.
out	<i>lo</i>	Pointer to epiface responsible for the capability.

#### Returns

[L4\\_EOK](#) if a valid interface could be found or an error message otherwise.

This method is called when a new entry is registered and some local ID was received for the capability. In this case, the generic implementation needs to get the epiface in order to get the capability.

The callee must make sure that the epiface remains valid until `free_epiface` is called. In particular, the capability slot must not be reallocated as long as the namespace server holds a reference to the epiface.

The documentation for this class was generated from the following file:

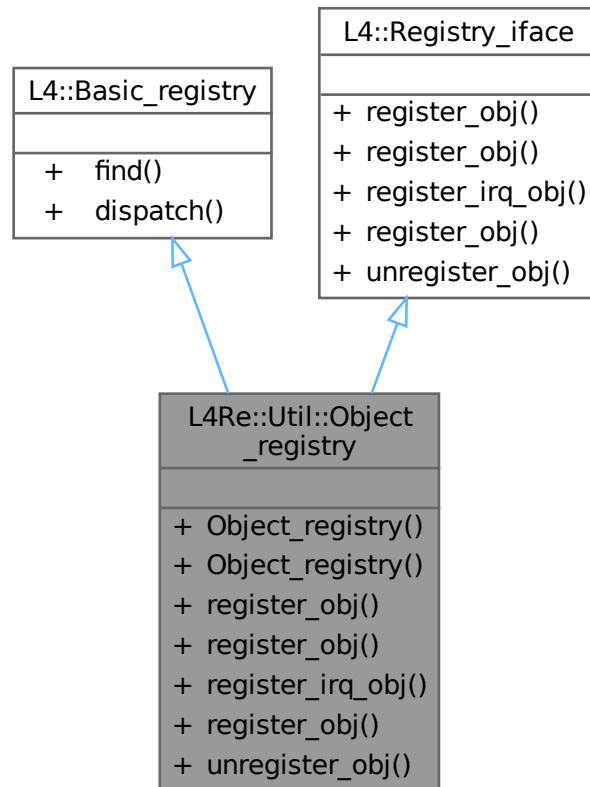
- `l4/re/util/name_space_svr`

## 16.329 L4Re::Util::Object\_registry Class Reference

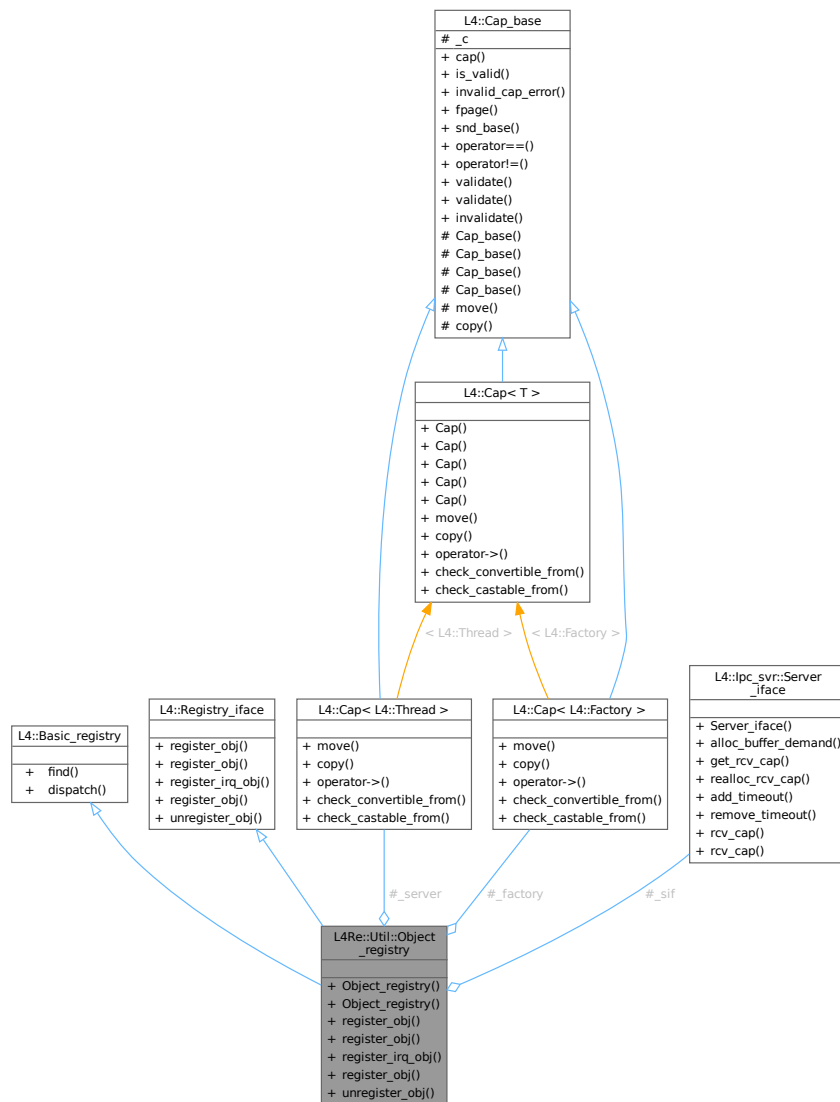
A registry that manages server objects and their attached IPC gates for a single server loop for a specific thread.

```
#include <object_registry>
```

Inheritance diagram for L4Re::Util::Object\_registry:



Collaboration diagram for L4Re::Util::Object\_registry:



## Public Member Functions

- [Object\\_registry](#) ([L4::lpc\\_svr::Server\\_iface](#) \*sif)  
*Create a registry for the main thread of the task using the default factory.*
- [Object\\_registry](#) ([L4::lpc\\_svr::Server\\_iface](#) \*sif, [L4::Cap< L4::Thread >](#) server, [L4::Cap< L4::Factory >](#) factory)  
*Create a registry for arbitrary threads.*
- [L4::Cap< void > register\\_obj](#) ([L4::Epiface](#) \*o, char const \*service) override  
*Register a new server object to a pre-allocated receive endpoint.*
- [L4::Cap< void > register\\_obj](#) ([L4::Epiface](#) \*o) override  
*Register a new server object on a newly allocated capability.*
- [L4::Cap< L4::Irq > register\\_irq\\_obj](#) ([L4::Epiface](#) \*o) override  
*Register a handler for an interrupt.*
- [L4::Cap< L4::Rcv\\_endpoint > register\\_obj](#) ([L4::Epiface](#) \*o, [L4::Cap< L4::Rcv\\_endpoint >](#) ep) override

*Register a handler for an already existing interrupt.*

- void [unregister\\_obj](#) ([L4::Epiface](#) \*o, bool unmap=true) override

*Remove a server object from the handler list.*

## Additional Inherited Members

## Static Public Member Functions inherited from [L4::Basic\\_registry](#)

- static [Value](#) \* [find](#) ([l4\\_umword\\_t](#) label)

*Get the server object for an [lpc\\_gate](#) label.*

- static [l4\\_msgtag\\_t](#) [dispatch](#) ([l4\\_msgtag\\_t](#) tag, [l4\\_umword\\_t](#) label, [l4\\_utcb\\_t](#) \*utcb)

*The dispatch function called by the server loop.*

### 16.329.1 Detailed Description

A registry that manages server objects and their attached IPC gates for a single server loop for a specific thread.

This class manages most of the setup of a server object. If necessary, an IPC gate is created, the specified thread is bound to the IPC gate. Incoming IPC is dispatched to the server object based on the label of the IPC gate.

The object registry is also able to manage IRQ endpoints. They require a different method for the object creation. Otherwise they are handled in the same way as IPC gates: a server object is responsible to process the incoming interrupts.

Definition at line 41 of file [object\\_registry](#).

### 16.329.2 Constructor & Destructor Documentation

#### 16.329.2.1 [Object\\_registry\(\)](#) [1/2]

```
L4Re::Util::Object_registry::Object_registry (
    L4::Ipc\_svr::Server\_iface * sif) [inline], [explicit]
```

Create a registry for the main thread of the task using the default factory.

#### Parameters

<i>sif</i>	Server loop interface.
------------	------------------------

Definition at line 67 of file [object\\_registry](#).

#### 16.329.2.2 [Object\\_registry\(\)](#) [2/2]

```
L4Re::Util::Object_registry::Object_registry (
    L4::Ipc\_svr::Server\_iface * sif,
    L4::Cap< L4::Thread > server,
    L4::Cap< L4::Factory > factory) [inline]
```

Create a registry for arbitrary threads.

#### Parameters

<i>sif</i>	Server loop interface.
<i>server</i>	Capability to the thread that executes the server objects.
<i>factory</i>	Capability to a factory object capable of creating new IPC gates.

Definition at line 81 of file [object\\_registry](#).

## 16.329.3 Member Function Documentation

### 16.329.3.1 register\_irq\_obj()

```
L4::Cap< L4::Irq > L4Re::Util::Object_registry::register_irq_obj (
    L4::Epiface * o) [inline], [override], [virtual]
```

Register a handler for an interrupt.

#### Parameters

<i>o</i>	Server object that handles IRQs.
----------	----------------------------------

#### Return values

<a href="#">L4::Cap&lt;L4::Irq&gt;</a>	Capability to a new IRQ object on success.
<a href="#">L4::Cap&lt;L4::Irq&gt;::Invalid</a>	The allocation of the IRQ has failed.

The IRQ will be newly allocated using the registry's factory object. The caller must call [unregister\\_obj\(\)](#) to free all resources.

Implements [L4::Registry\\_iface](#).

Definition at line 227 of file [object\\_registry](#).

### 16.329.3.2 register\_obj() [1/3]

```
L4::Cap< void > L4Re::Util::Object_registry::register_obj (
    L4::Epiface * o) [inline], [override], [virtual]
```

Register a new server object on a newly allocated capability.

#### Parameters

<i>o</i>	Server object that handles IPC requests.
----------	------------------------------------------

#### Return values

<a href="#"><i>L4::Cap&lt;void&gt;</i></a>	A valid capability to a new IPC gate.
<a href="#"><i>L4::Cap&lt;void&gt;::Invalid</i></a>	The allocation of the IPC gate has failed.

The IPC gate will be allocated using the registry's factory. The caller must call [unregister\\_obj\(\)](#) to free all resources.

Implements [L4::Registry\\_iface](#).

Definition at line 211 of file [object\\_registry](#).

### 16.329.3.3 register\_obj() [2/3]

```
L4::Cap< void > L4Re::Util::Object_registry::register_obj (
    L4::Epiface * o,
    char const * service) [inline], [override], [virtual]
```

Register a new server object to a pre-allocated receive endpoint.

#### Parameters

<i>o</i>	Server object that handles IPC requests.
<i>service</i>	Name of a pre-allocated receive endpoint.

#### Return values

<a href="#"><i>L4::Cap&lt;void&gt;</i></a>	The capability known as <i>service</i> on success.
<a href="#"><i>L4::Cap&lt;void&gt;::Invalid</i></a>	No capability with the given name found.

The interface must be freed with [unregister\\_obj\(\)](#) by the caller to unbind the thread from the capability.

Implements [L4::Registry\\_iface](#).

Definition at line 194 of file [object\\_registry](#).

### 16.329.3.4 register\_obj() [3/3]

```
L4::Cap< L4::Rcv_endpoint > L4Re::Util::Object_registry::register_obj (
    L4::Epiface * o,
    L4::Cap< L4::Rcv_endpoint > ep) [inline], [override], [virtual]
```

Register a handler for an already existing interrupt.

#### Parameters

<i>o</i>	Server object that handles the IPC.
<i>ep</i>	Capability to a receive endpoint, may be a hardware or software interrupt or an IPC gate.

#### Return values



<a href="#">L4::Cap&lt;L4::Rcv_endpoint&gt;</a>	Capability ep on success.
<a href="#">L4::Cap&lt;L4::Rcv_endpoint&gt;::Invalid</a>	The IRQ attach operation has failed.

The interface must be freed with [unregister\\_obj\(\)](#) by the caller to unbind the thread from the capability.

Implements [L4::Registry\\_iface](#).

Definition at line 246 of file [object\\_registry](#).

### 16.329.3.5 unregister\_obj()

```
void L4Re::Util::Object_registry::unregister_obj (
    L4::Epiface * o,
    bool unmap = true) [inline], [override], [virtual]
```

Remove a server object from the handler list.

#### Parameters

<i>o</i>	Server object to unbind.
<i>unmap</i>	Specifies if the object capability shall be unmapped (true) or not. The default (true) is to unmap the capability.

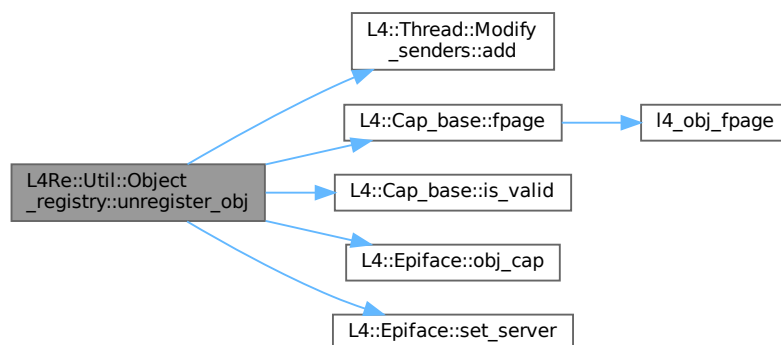
The capability used by the server object will be unmapped if `unmap` is true.

Implements [L4::Registry\\_iface](#).

Definition at line 262 of file [object\\_registry](#).

References [L4::Thread::Modify\\_senders::add\(\)](#), [L4Re::Util::cap\\_alloc](#), [L4::Cap\\_base::fpage\(\)](#), [L4::Cap\\_base::Invalid](#), [L4::Cap\\_base::is\\_valid\(\)](#), [L4\\_FP\\_ALL\\_SPACES](#), [L4::Epiface::obj\\_cap\(\)](#), and [L4::Epiface::set\\_server\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

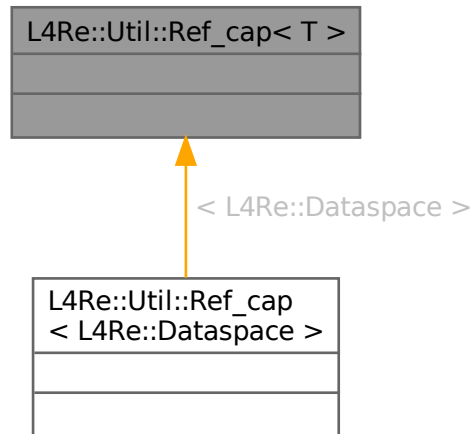
- `I4/re/util/object_registry`

## 16.330 L4Re::Util::Ref\_cap< T > Struct Template Reference

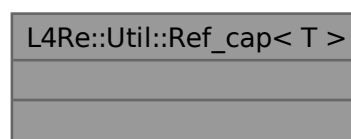
Automatic capability that implements automatic free and unmap of the capability selector.

```
#include <cap_alloc>
```

Inheritance diagram for L4Re::Util::Ref\_cap< T >:



Collaboration diagram for L4Re::Util::Ref\_cap< T >:



### 16.330.1 Detailed Description

```
template<typename T>
struct L4Re::Util::Ref_cap< T >
```

Automatic capability that implements automatic free and unmap of the capability selector.

#### Template Parameters

---

<i>T</i>	Type of the object that is referred by the capability.
----------	--------------------------------------------------------

This kind of automatic capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero.

#### Usage:

```
L4Re::Util::Ref_cap<L4Re::Dataspace>::Cap global_ds_cap;

{
    L4Re::Util::Ref_cap<L4Re::Dataspace>::Cap
        ds_cap(L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>());
    // reference count for the allocated cap selector is now 1

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

    global_ds_cap = ds_cap;
    // reference count is now 2
    ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
```

Definition at line 142 of file [cap\\_alloc](#).

The documentation for this struct was generated from the following file:

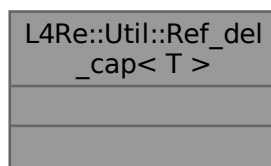
- [l4/re/util/cap\\_alloc](#)

## 16.331 L4Re::Util::Ref\_del\_cap< T > Struct Template Reference

Automatic capability that implements automatic free and unmap+delete of the capability selector.

```
#include <cap_alloc>
```

Collaboration diagram for L4Re::Util::Ref\_del\_cap< T >:



### 16.331.1 Detailed Description

```
template<typename T>
struct L4Re::Util::Ref_del_cap< T >
```

Automatic capability that implements automatic free and unmap+delete of the capability selector.

#### Template Parameters

<i>T</i>	Type of the object that is referred by the capability.
----------	--------------------------------------------------------

This kind of automatic capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero. The main difference to [Ref\\_cap](#) is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

#### Usage:

```
L4Re::Util::Ref_del_cap<L4Re::Dataspace>::Cap global_ds_cap;

{
    L4Re::Util::Ref_del_cap<L4Re::Dataspace>::Cap
        ds_cap(L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>());
    // reference count for the allocated cap selector is now 1

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

    global_ds_cap = ds_cap;
    // reference count is now 2
    ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
...
global_ds_cap = L4_INVALID_CAP;
// reference count dropped to 0 (data space shall be deleted).
```

Definition at line 183 of file [cap\\_alloc](#).

The documentation for this struct was generated from the following file:

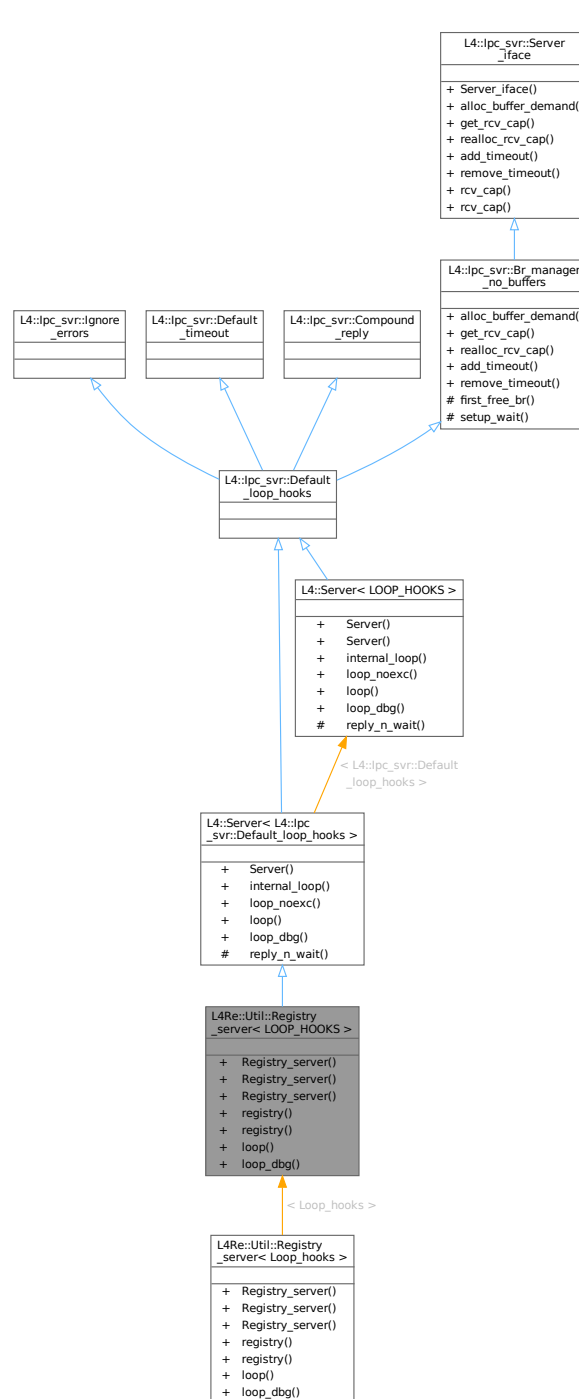
- [l4/re/util/cap\\_alloc](#)

## 16.332 L4Re::Util::Registry\_server< LOOP\_HOOKS > Class Template Reference

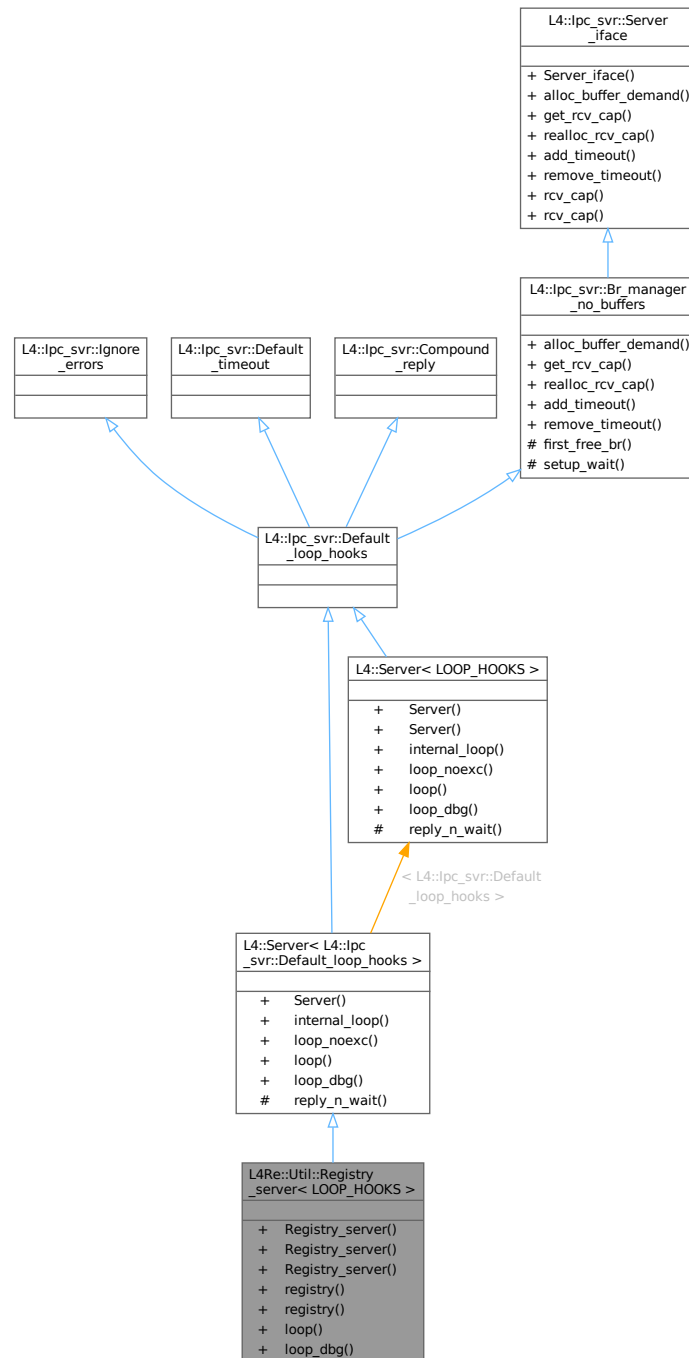
A server loop object which has a [Object\\_registry](#) included.

```
#include <object_registry>
```

Inheritance diagram for L4Re::Util::Registry\_server< LOOP\_HOOKS >:



Collaboration diagram for L4Re::Util::Registry\_server< LOOP\_HOOKS >:



## Public Member Functions

- `Registry_server ()`  
*Create a new server loop object for the main thread of the task.*
- `Registry_server (l4_utcb_t *, L4::Cap< L4::Thread > server, L4::Cap< L4::Factory > factory)`  
*Create a new server loop object for an arbitrary thread and factory.*
- `Registry_server (L4::Cap< L4::Thread > server, L4::Cap< L4::Factory > factory)`

- *Create a new server loop object for an arbitrary thread and factory.*
- `Object_registry` const \* **registry** () const  
*Return registry of this server loop.*
- `Object_registry` \* **registry** ()  
*Return registry of this server loop.*
- void `L4_NORETURN` **loop** (`l4_utcb_t` \*utcb=`l4_utcb`())  
*Start the server loop.*
- template<typename Printer>  
void `L4_NORETURN` **loop\_dbg** (Printer printer, `l4_utcb_t` \*utcb=`l4_utcb`())  
*Start the server loop with error printing.*

## Public Member Functions inherited from `L4::Server< L4::lpc_svr::Default_loop_hooks >`

- `Server` (`l4_utcb_t` \*)  
*Initializes the server loop.*
- `L4_NORETURN` void **internal\_loop** (DISPATCH dispatch, `l4_utcb_t` \*)  
*The server loop.*
- `L4_NORETURN` void **loop\_noexc** (R r, `l4_utcb_t` \*u=`l4_utcb`())  
*Server loop without exception handling.*
- `L4_NORETURN` void **loop** (R r, `l4_utcb_t` \*u=`l4_utcb`())  
*Server loop with internal exception handling.*
- `L4_NORETURN` void **loop\_dbg** (R r, Printer p, `l4_utcb_t` \*u=`l4_utcb`())  
*Server loop with internal exception handling including message printing.*

## Public Member Functions inherited from `L4::lpc_svr::Br_manager_no_buffers`

- int **alloc\_buffer\_demand** (Demand const &demand) override  
*Tells the server to allocate buffers for the given demand.*
- `L4::Cap< void >` **get\_rcv\_cap** (int) const override  
*Returns `L4::Cap<void>::Invalid`, we have no buffer management.*
- int **realloc\_rcv\_cap** (int) override  
*Returns `-L4_ENOMEM`, we have no buffer management.*
- int **add\_timeout** (Timeout \*, `l4_kernel_clock_t`) override  
*Returns `-L4_ENOSYS`, we have no timeout queue.*
- int **remove\_timeout** (Timeout \*) override  
*Returns `-L4_ENOSYS`, we have no timeout queue.*

## Public Member Functions inherited from `L4::lpc_svr::Server_iface`

- `Server_iface` ()  
*Make a server interface.*
- template<typename T>  
`L4::Cap< T >` **rcv\_cap** (int index) const  
*Get given receive buffer as typed capability.*
- `L4::Cap< void >` **rcv\_cap** (int index) const  
*Get receive cap with the given index as generic (void) type.*

## Additional Inherited Members

### Public Types inherited from [L4::lpc\\_svr::Server\\_iface](#)

- typedef [L4::Type\\_info::Demand](#) **Demand**  
*Data type expressing server-side demand for receive buffers.*

### Protected Member Functions inherited from [L4::Server< L4::lpc\\_svr::Default\\_loop\\_hooks >](#)

- [l4\\_msgtag\\_t](#) **reply\_n\_wait** ([l4\\_msgtag\\_t](#) reply, [l4\\_umword\\_t](#) \*p, [l4\\_utcb\\_t](#) \*)  
*Internal implementation for reply and wait.*

### Protected Member Functions inherited from [L4::lpc\\_svr::Br\\_manager\\_no\\_buffers](#)

- unsigned **first\_free\_br** () const  
*Returns 1 as first free buffer.*
- void **setup\_wait** ([l4\\_utcb\\_t](#) \*utcb, [L4::lpc\\_svr::Reply\\_mode](#))  
*Setup wait function for the server loop ([Server<>](#)).*

## 16.332.1 Detailed Description

```
template<typename LOOP_HOOKS = L4::lpc_svr::Default_loop_hooks>
class L4Re::Util::Registry_server< LOOP_HOOKS >
```

A server loop object which has a [Object\\_registry](#) included.

### Examples

[examples/clntsrv/src/server.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#), and [examples/libs/l4re/streammap/server.cc](#).

Definition at line 293 of file [object\\_registry](#).

## 16.332.2 Constructor & Destructor Documentation

### 16.332.2.1 Registry\_server() [1/3]

```
template<typename LOOP_HOOKS = L4::lpc_svr::Default_loop_hooks>
L4Re::Util::Registry_server< LOOP_HOOKS >::Registry_server () [inline]
```

Create a new server loop object for the main thread of the task.

#### Precondition

Must be called from the main thread or behaviour is undefined.

Definition at line 305 of file [object\\_registry](#).

### 16.332.2.2 Registry\_server() [2/3]

```
template<typename LOOP_HOOKS = L4::lpc_svr::Default_loop_hooks>
L4Re::Util::Registry_server< LOOP_HOOKS >::Registry_server (
    l4_utcb_t * ,
    L4::Cap< L4::Thread > server,
    L4::Cap< L4::Factory > factory) [inline]
```

Create a new server loop object for an arbitrary thread and factory.

#### Parameters



<i>server</i>	Capability to thread running the server loop.
<i>factory</i>	Capability to factory object used to create new IPC gates.

**Deprecated** Note that this variant of the constructor is deprecated, please do not supply the UTCB pointer, it's not used.

Definition at line 317 of file [object\\_registry](#).

### 16.332.2.3 Registry\_server() [3/3]

```
template<typename LOOP_HOOKS = L4::Ipc_svr::Default_loop_hooks>
L4Re::Util::Registry_server< LOOP_HOOKS >::Registry_server (
    L4::Cap< L4::Thread > server,
    L4::Cap< L4::Factory > factory) [inline]
```

Create a new server loop object for an arbitrary thread and factory.

#### Parameters

<i>server</i>	Capability to thread running the server loop.
<i>factory</i>	Capability to factory object used to create new IPC gates.

Definition at line 328 of file [object\\_registry](#).

## 16.332.3 Member Function Documentation

### 16.332.3.1 loop()

```
template<typename LOOP_HOOKS = L4::Ipc_svr::Default_loop_hooks>
void L4_NORETURN L4Re::Util::Registry_server< LOOP_HOOKS >::loop (
    l4_utcb_t * utcb = l4_utcb()) [inline]
```

Start the server loop.

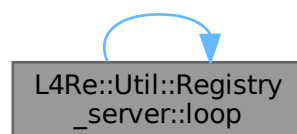
#### Parameters

<i>utcb</i>	The UTCB of the thread running the server loop, defaults to <a href="#">l4_utcb()</a> .
-------------	-----------------------------------------------------------------------------------------

Definition at line 344 of file [object\\_registry](#).

Referenced by [L4Re::Util::Registry\\_server< Loop\\_hooks >::loop\(\)](#).

Here is the caller graph for this function:



### 16.332.3.2 loop\_dbg()

```
template<typename LOOP_HOOKS = L4::Ipc_svr::Default_loop_hooks>
template<typename Printer>
void L4_NORETURN L4Re::Util::Registry_server< LOOP_HOOKS >::loop_dbg (
    Printer printer,
    l4_utcb_t * utcb = l4_utcb()) [inline]
```

Start the server loop with error printing.

#### Template Parameters

<i>Printer</i>	The printer type.
----------------	-------------------

#### Parameters

<i>printer</i>	The printer object on which printf() is called.
<i>utcb</i>	The UTCB of the thread running the server loop, defaults to <a href="#">l4_utcb()</a> .

Definition at line [356](#) of file [object\\_registry](#).

Referenced by [L4Re::Util::Registry\\_server< Loop\\_hooks >::loop\\_dbg\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

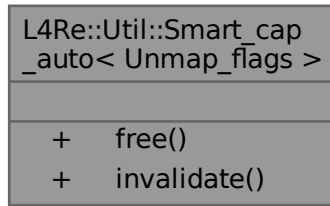
- `l4/re/util/object_registry`

## 16.333 L4Re::Util::Smart\_cap\_auto< Unmap\_flags > Class Template Reference

Helper for [Unique\\_cap](#) and [Unique\\_del\\_cap](#).

```
#include <cap_alloc>
```

Collaboration diagram for L4Re::Util::Smart\_cap\_auto< Unmap\_flags >:



#### Static Public Member Functions

- static void **free** ([L4::Cap\\_base](#) &c)  
*Free the provided capability.*
- static void **invalidate** ([L4::Cap\\_base](#) &c)  
*Invalidate the provided capability.*

#### 16.333.1 Detailed Description

```
template<unsigned long Unmap_flags = L4_FP_ALL_SPACES>
class L4Re::Util::Smart_cap_auto< Unmap_flags >
```

Helper for [Unique\\_cap](#) and [Unique\\_del\\_cap](#).

Definition at line 45 of file [cap\\_alloc](#).

The documentation for this class was generated from the following file:

- [l4/re/util/cap\\_alloc](#)

### 16.334 L4Re::Util::Smart\_count\_cap< Unmap\_flags > Class Template Reference

Helper for [Ref\\_cap](#) and [Ref\\_del\\_cap](#).

```
#include <cap_alloc>
```

Collaboration diagram for L4Re::Util::Smart\_count\_cap< Unmap\_flags >:

L4Re::Util::Smart_count_cap< Unmap_flags >	
+	free()
+	invalidate()
+	copy()

### Static Public Member Functions

- static void **free** ([L4::Cap\\_base](#) &c) noexcept  
*Free operation for [L4::Smart\\_cap](#) (decrement ref count and delete if 0).*
- static void **invalidate** ([L4::Cap\\_base](#) &c) noexcept  
*Invalidate operation for [L4::Smart\\_cap](#).*
- static [L4::Cap\\_base](#) **copy** ([L4::Cap\\_base](#) const &src)  
*Copy operation for [L4::Smart\\_cap](#) (increment ref count).*

#### 16.334.1 Detailed Description

```
template<unsigned long Unmap_flags = L4_FP_ALL_SPACES>
class L4Re::Util::Smart_count_cap< Unmap_flags >
```

Helper for [Ref\\_cap](#) and [Ref\\_del\\_cap](#).

Definition at line 76 of file [cap\\_alloc](#).

The documentation for this class was generated from the following file:

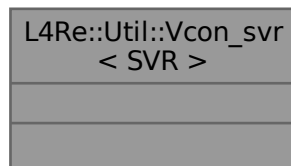
- [l4/re/util/cap\\_alloc](#)

### 16.335 L4Re::Util::Vcon\_svr< SVR > Class Template Reference

[Console](#) server template class.

```
#include <vcon_svr>
```

Collaboration diagram for L4Re::Util::Vcon\_svr< SVR >:



### 16.335.1 Detailed Description

```
template<typename SVR>
class L4Re::Util::Vcon_svr< SVR >
```

[Console](#) server template class.

This template uses `vcon_write()` and `vcon_read()` to get and deliver data from the implementor.

`vcon_read()` needs to update the status argument with the `L4_vcon_read_stat` flags, especially the `L4_VCON_READ_STAT_DONE` flag to indicate that there's nothing more to read for the other end.

`vcon_write()` gets the live data from the UTCB. Make sure to copy out the data before using the UTCB again.

The size parameter of both functions is given in bytes.

Definition at line [36](#) of file [vcon\\_svr](#).

The documentation for this class was generated from the following file:

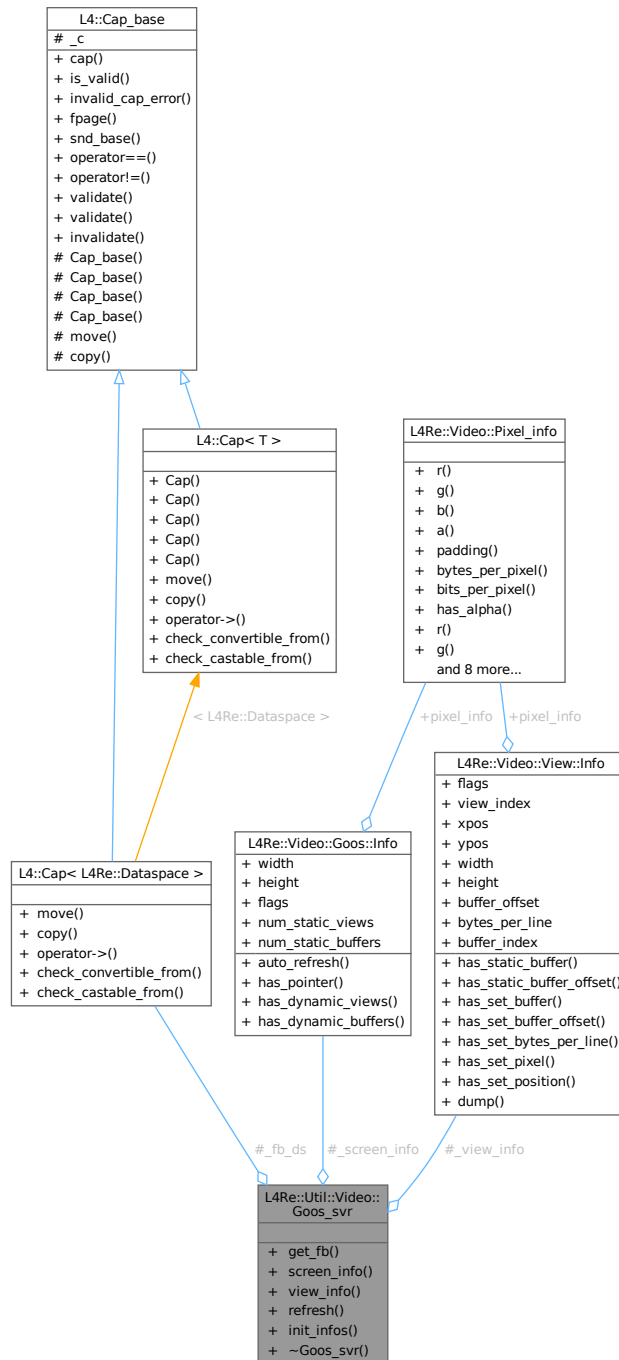
- `l4/re/util/vcon_svr`

## 16.336 L4Re::Util::Video::Goos\_svr Class Reference

Goos server class.

```
#include <goos_svr>
```

Collaboration diagram for L4Re::Util::Video::Goos\_svr:



## Public Member Functions

- **L4::Cap< L4Re::Dataspace >** **get\_fb ()** const  
Return framebuffer memory dataspace.
- **L4Re::Video::Goos::Info** const \* **screen\_info ()** const  
Goos information structure.
- **L4Re::Video::View::Info** const \* **view\_info ()** const

*View information structure.*

- virtual int [refresh](#) (int x, int y, int w, int h)

*Refresh area of the framebuffer.*

- void [init\\_infos](#) ()

*Initialize the view information structure of this object.*

- virtual  $\sim$ [Goos\\_svr](#) ()

*Destructor.*

## Protected Attributes

- [L4::Cap](#)< [L4Re::Dataspace](#) > [\\_fb\\_ds](#)

*Goos memory dataspace.*

- [L4Re::Video::Goos::Info](#) [\\_screen\\_info](#)

*Goos information.*

- [L4Re::Video::View::Info](#) [\\_view\\_info](#)

*View information.*

## 16.336.1 Detailed Description

Goos server class.

Definition at line 25 of file [goos\\_svr](#).

## 16.336.2 Member Function Documentation

### 16.336.2.1 [get\\_fb\(\)](#)

```
L4::Cap< L4Re::Dataspace > L4Re::Util::Video::Goos\_svr::get\_fb () const [inline]
```

Return framebuffer memory dataspace.

#### Returns

Goos memory dataspace

Definition at line 42 of file [goos\\_svr](#).

References [\\_fb\\_ds](#).

### 16.336.2.2 [init\\_infos\(\)](#)

```
void L4Re::Util::Video::Goos\_svr::init\_infos () [inline]
```

Initialize the view information structure of this object.

This function initializes the view info structure of this goos object based on the information in the goos information, i.e. the width, height and pixel\_info of the goos information has to contain valid values before calling [init\\_info\(\)](#).

Definition at line 78 of file [goos\\_svr](#).

References [\\_screen\\_info](#), and [\\_view\\_info](#).

### 16.336.2.3 refresh()

```
virtual int L4Re::Util::Video::Goos_svr::refresh (  
    int x,  
    int y,  
    int w,  
    int h) [inline], [virtual]
```

Refresh area of the framebuffer.

#### Parameters

---



<i>x</i>	X coordinate (pixels)
<i>y</i>	Y coordinate (pixels)
<i>w</i>	Width of area in pixels
<i>h</i>	Height of area in pixels

#### Returns

0 on success, negative error code otherwise

Definition at line 66 of file [goos\\_svr](#).

References [L4\\_ENOSYS](#).

#### 16.336.2.4 screen\_info()

```
L4Re::Video::Goos::Info const * L4Re::Util::Video::Goos_svr::screen_info () const [inline]
```

Goos information structure.

#### Returns

Return goos information structure.

Definition at line 48 of file [goos\\_svr](#).

References [\\_screen\\_info](#).

#### 16.336.2.5 view\_info()

```
L4Re::Video::View::Info const * L4Re::Util::Video::Goos_svr::view_info () const [inline]
```

View information structure.

#### Returns

Return view information structure.

Definition at line 54 of file [goos\\_svr](#).

References [\\_view\\_info](#).

The documentation for this class was generated from the following file:

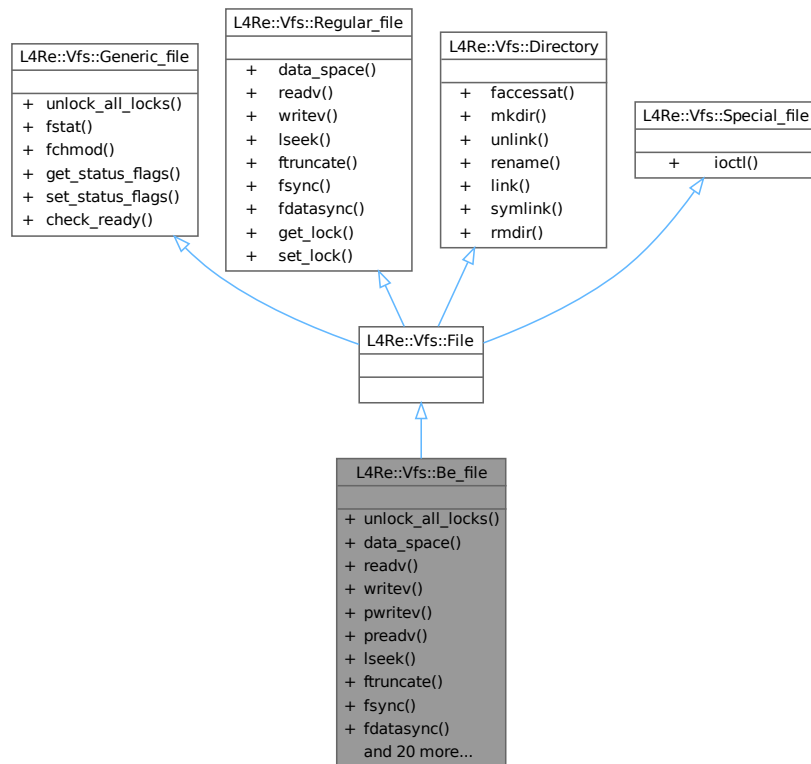
- [l4/re/util/video/goos\\_svr](#)

## 16.337 L4Re::Vfs::Be\_file Class Reference

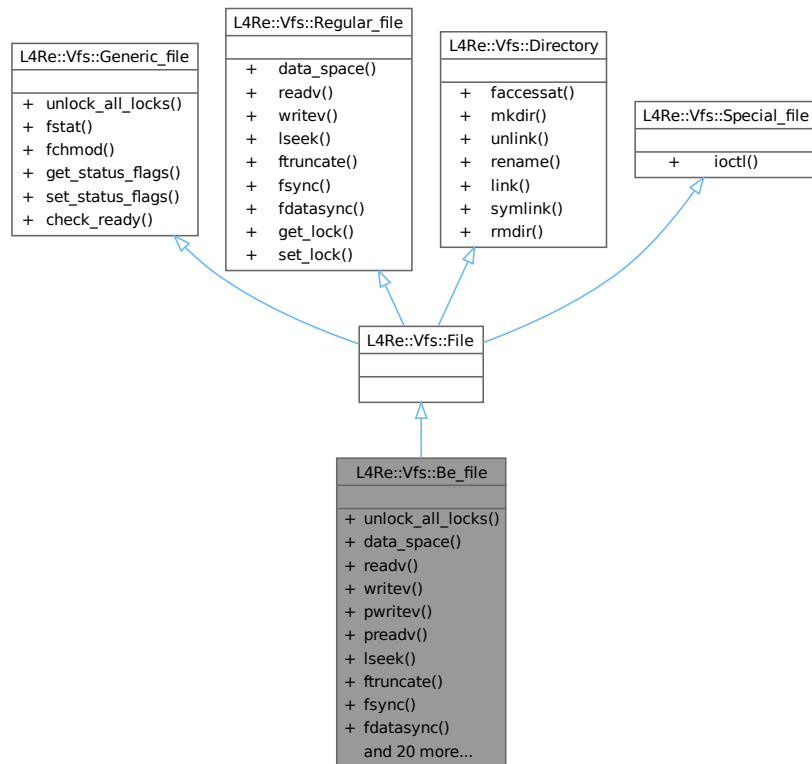
Boiler plate class for implementing an open file for [L4Re::Vfs](#).

```
#include <backend>
```

Inheritance diagram for L4Re::Vfs::Be\_file:



Collaboration diagram for L4Re::Vfs::Be\_file:



## Public Member Functions

- `int unlock_all_locks ()` noexcept override  
*Unlock all locks on the file.*
- `L4::Cap< L4Re::Dataspace > data_space ()` noexcept override  
*Get an [L4Re::Dataspace](#) object for the file.*
- `ssize_t readv (const struct iovec *, int)` noexcept override  
*Default backend for POSIX read and readv functions.*
- `ssize_t writev (const struct iovec *, int)` noexcept override  
*Default backend for POSIX write and writev functions.*
- `ssize_t pwritev (const struct iovec *, int, off64_t)` noexcept override  
*Default backend for POSIX pwrite and pwritev functions.*
- `ssize_t preadv (const struct iovec *, int, off64_t)` noexcept override  
*Default backend for POSIX pread and preadv functions.*
- `off64_t lseek (off64_t, int)` noexcept override  
*Default backend for POSIX seek and lseek functions.*
- `int ftruncate (off64_t)` noexcept override  
*Default backend for the POSIX truncate, ftruncate and similar functions.*
- `int fsync ()` const noexcept override  
*Default backend for POSIX fsync.*
- `int fdatsync ()` const noexcept override  
*Default backend for POSIX fdatsync.*

- int **ioctl** (unsigned long, va\_list) noexcept override  
*Default backend for POSIX ioctl.*
- int **fstat** (struct stat64 \*) const noexcept override  
*Get status information for the file.*
- int **fchmod** (mode\_t) noexcept override  
*Default backend for POSIX chmod and fchmod.*
- int **get\_status\_flags** () const noexcept override  
*Default backend for POSIX fcntl subfunctions.*
- int **set\_status\_flags** (long) noexcept override  
*Default backend for POSIX fcntl subfunctions.*
- int **get\_lock** (struct flock64 \*) noexcept override  
*Default backend for POSIX fcntl subfunctions.*
- int **set\_lock** (struct flock64 \*, bool) noexcept override  
*Default backend for POSIX fcntl subfunctions.*
- int **faccessat** (const char \*, int, int) noexcept override  
*Default backend for POSIX access and faccessat functions.*
- int **fchmodat** (const char \*, mode\_t, int) noexcept override  
*Default backend for POSIX fchmodat function.*
- int **utime** (const struct utimbuf \*) noexcept override  
*Default backend for POSIX utime.*
- int **utimes** (const struct timeval[2]) noexcept override  
*Default backend for POSIX utimes.*
- int **utimensat** (const char \*, const struct timespec[2], int) noexcept override  
*Default backend for POSIX utimensat.*
- int **mkdir** (const char \*, mode\_t) noexcept override  
*Default backend for POSIX mkdir and mkdirat.*
- int **unlink** (const char \*) noexcept override  
*Default backend for POSIX unlink, unlinkat.*
- int **rename** (const char \*, const char \*) noexcept override  
*Default backend for POSIX rename, renameat.*
- int **link** (const char \*, const char \*) noexcept override  
*Default backend for POSIX link, linkat.*
- int **symlink** (const char \*, const char \*) noexcept override  
*Default backend for POSIX symlink, symlinkat.*
- int **rmdir** (const char \*) noexcept override  
*Default backend for POSIX rmdir, rmdirat.*
- ssize\_t **readlink** (char \*, size\_t) override  
*Default backend for POSIX readlink, readlinkat.*
- bool **check\_ready** ([Ready\\_type](#)) noexcept override  
*Default implementation of a readiness check.*

### Additional Inherited Members

### Public Types inherited from [L4Re::Vfs::Generic\\_file](#)

- enum [Ready\\_type](#) : unsigned  
*Type of I/O operation/condition a file can indicate readiness.*

### 16.337.1 Detailed Description

Boiler plate class for implementing an open file for [L4Re::Vfs](#).

This class may be used as a base class for everything that a POSIX file descriptor may point to. This are things such as regular files, directories, special device files, streams, pipes, and so on.

#### Examples

[tmpfs/lib/src/fs.cc](#).

Definition at line 30 of file [backend](#).

### 16.337.2 Member Function Documentation

#### 16.337.2.1 `check_ready()`

```
bool L4Re::Vfs::Be_file::check_ready (
    Ready_type ) [inline], [override], [virtual], [noexcept]
```

Default implementation of a readiness check.

By default, we assume a file is not ready for an I/O operation/condition since the proper semantics of that relies on the backend.

#### Returns

Always false.

Implements [L4Re::Vfs::Generic\\_file](#).

Definition at line 217 of file [backend](#).

#### 16.337.2.2 `data_space()`

```
L4::Cap< L4Re::Dataspace > L4Re::Vfs::Be_file::data_space () [inline], [override], [virtual],
[noexcept]
```

Get an [L4Re::Dataspace](#) object for the file.

This is used as a backend for POSIX mmap and mmap2 functions.

#### Note

mmap is not possible if the function returns an invalid capability.

#### Returns

A capability to an [L4Re::Dataspace](#) that represents the file contents in an [L4Re](#) way.

Implements [L4Re::Vfs::Regular\\_file](#).

Definition at line 47 of file [backend](#).

References [L4::Cap\\_base::Invalid](#).

#### 16.337.2.3 `fstat()`

```
int L4Re::Vfs::Be_file::fstat (
    struct stat64 * buf) const [inline], [override], [virtual], [noexcept]
```

Get status information for the file.

This is the backend for POSIX fstat, stat, fstat64 and friends.

#### Parameters

---

out	buf	This buffer is filled with the status information.
-----	-----	----------------------------------------------------

#### Returns

0 on success, or <0 on error.

Implements [L4Re::Vfs::Generic\\_file](#).

Definition at line 86 of file [backend](#).

### 16.337.2.4 unlock\_all\_locks()

```
int L4Re::Vfs::Be_file::unlock_all_locks () [inline], [override], [virtual], [noexcept]
```

Unlock all locks on the file.

#### Note

All locks means all locks independent of which file the locks were taken by.

This method is called by the POSIX close implementation to get the POSIX semantics of releasing all locks taken by this application on a close for any fd referencing the real file.

#### Returns

0 on success, or <0 on error.

Implements [L4Re::Vfs::Generic\\_file](#).

Definition at line 43 of file [backend](#).

The documentation for this class was generated from the following file:

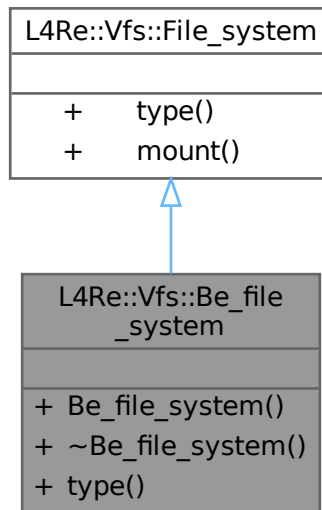
- [l4/l4re\\_vfs/backend](#)

## 16.338 L4Re::Vfs::Be\_file\_system Class Reference

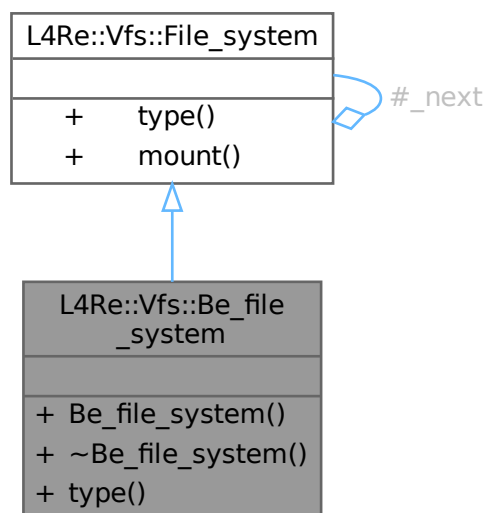
Boilerplate class for implementing a [L4Re::Vfs::File\\_system](#).

```
#include <backend>
```

Inheritance diagram for L4Re::Vfs::Be\_file\_system:



Collaboration diagram for L4Re::Vfs::Be\_file\_system:



## Public Member Functions

- [Be\\_file\\_system](#) (char const \*fstype) noexcept  
*Create a file-system object for the given fstype.*
- [~Be\\_file\\_system](#) () noexcept  
*Destroy a file-system object.*
- char const \* [type](#) () const noexcept override  
*Return the file-system type.*

## Public Member Functions inherited from [L4Re::Vfs::File\\_system](#)

- virtual int [mount](#) (char const \*source, unsigned long mountflags, void const \*data, [cxx::Ref\\_ptr](#)< [File](#) > \*dir) noexcept=0  
*Create a directory object dir representing source mounted with this file system.*

### 16.338.1 Detailed Description

Boilerplate class for implementing a [L4Re::Vfs::File\\_system](#).

This class already takes care of registering and unregistering the file system in the global registry and implements the [type\(\)](#) method.

#### Examples

[tmpfs/lib/src/fs.cc](#).

Definition at line 310 of file [backend](#).

### 16.338.2 Constructor & Destructor Documentation

#### 16.338.2.1 [Be\\_file\\_system](#)()

```
L4Re::Vfs::Be_file_system::Be_file_system (
    char const * fstype) [inline], [explicit], [noexcept]
```

Create a file-system object for the given *fstype*.

#### Parameters

<i>fstype</i>	The type that <a href="#">type()</a> shall return.
---------------	----------------------------------------------------

This constructor takes care of registering the file system in the registry of [L4Re::Vfs::vfs\\_ops](#).

Definition at line 324 of file [backend](#).



### 16.338.2.2 ~Be\_file\_system()

```
L4Re::Vfs::Be_file_system::~Be_file_system () [inline], [noexcept]
```

Destroy a file-system object.

This destructor takes care of removing this file system from the registry of L4Re::Vfs::vfs\_ops.

Definition at line 336 of file [backend](#).

## 16.338.3 Member Function Documentation

### 16.338.3.1 type()

```
char const * L4Re::Vfs::Be_file_system::type () const [inline], [override], [virtual], [noexcept]
```

Return the file-system type.

Returns the file-system type given as *fstype* in the constructor.

Implements [L4Re::Vfs::File\\_system](#).

Definition at line 346 of file [backend](#).

The documentation for this class was generated from the following file:

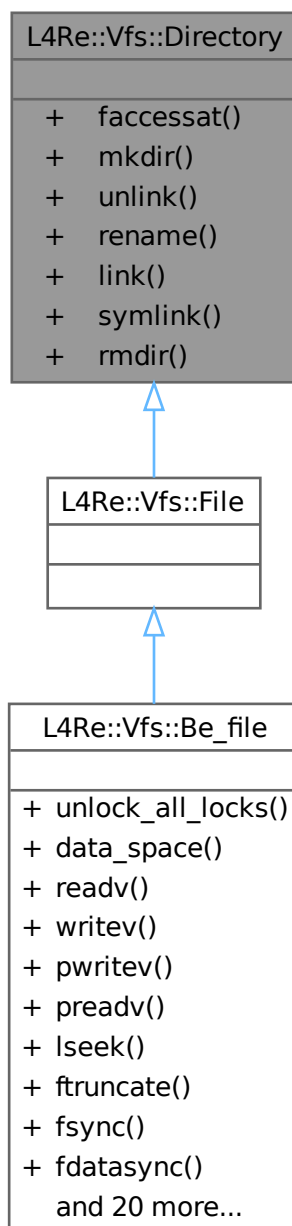
- l4/l4re\_vfs/backend

## 16.339 L4Re::Vfs::Directory Class Reference

Interface for a POSIX file that is a directory.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Directory:



Collaboration diagram for L4Re::Vfs::Directory:

L4Re::Vfs::Directory	
+	faccessat()
+	mkdir()
+	unlink()
+	rename()
+	link()
+	symlink()
+	rmdir()

## Public Member Functions

- virtual int [faccessat](#) (const char \*path, int mode, int flags) noexcept=0  
*Check access permissions on the given file.*
- virtual int [mkdir](#) (const char \*path, mode\_t mode) noexcept=0  
*Create a new subdirectory.*
- virtual int [unlink](#) (const char \*path) noexcept=0  
*Unlink the given file from that directory.*
- virtual int [rename](#) (const char \*src\_path, const char \*dst\_path) noexcept=0  
*Rename the given file.*
- virtual int [link](#) (const char \*src\_path, const char \*dst\_path) noexcept=0  
*Create a hard link (second name) for the given file.*
- virtual int [symlink](#) (const char \*src\_path, const char \*dst\_path) noexcept=0  
*Create a symbolic link for the given file.*
- virtual int [rmdir](#) (const char \*path) noexcept=0  
*Delete an empty directory.*

### 16.339.1 Detailed Description

Interface for a POSIX file that is a directory.

This interface provides functionality for directory files in the [L4Re::Vfs](#). However, real objects always use the combined [L4Re::Vfs::File](#) interface.

Definition at line 160 of file [vfs.h](#).

## 16.339.2 Member Function Documentation

### 16.339.2.1 faccessat()

```
virtual int L4Re::Vfs::Directory::faccessat (  
    const char * path,  
    int mode,  
    int flags) [pure virtual], [noexcept]
```

Check access permissions on the given file.

Backend function for POSIX access and faccessat functions.

#### Parameters

<i>path</i>	The path relative to this directory. Note: <i>path</i> is relative to this directory and may contain subdirectories.
<i>mode</i>	The access mode to check.
<i>flags</i>	The flags as in POSIX faccessat (AT_EACCESS, AT_SYMLINK_NOFOLLOW).

#### Returns

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 16.339.2.2 link()

```
virtual int L4Re::Vfs::Directory::link (  
    const char * src_path,  
    const char * dst_path) [pure virtual], [noexcept]
```

Create a hard link (second name) for the given file.

Backend for the POSIX link and linkat functions.

#### Parameters

<i>src_path</i>	The old name of the file. Note: <i>src_path</i> is relative to this directory and may contain subdirectories.
<i>dst_path</i>	The new (second) name for the file. Note: <i>dst_path</i> is relative to this directory and may contain subdirectories.

#### Returns

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 16.339.2.3 mkdir()

```
virtual int L4Re::Vfs::Directory::mkdir (  
    const char * path,  
    mode_t mode) [pure virtual], [noexcept]
```

Create a new subdirectory.

Backend for POSIX mkdir and mkdirat function calls.

#### Parameters

<i>path</i>	The name of the subdirectory to create. Note: <i>path</i> is relative to this directory and may contain subdirectories.
<i>mode</i>	The file mode to use for the new directory.

**Returns**

0 on success, or <0 on error. -ENOTDIR if this or some component in path is not a directory.

Implemented in [L4Re::Vfs::Be\\_file](#).

**16.339.2.4 rename()**

```
virtual int L4Re::Vfs::Directory::rename (  
    const char * src_path,  
    const char * dst_path) [pure virtual], [noexcept]
```

Rename the given file.

Backend for the POSIX rename, renameat functions.

**Parameters**

<i>src_path</i>	The old name of the file to rename. Note: <i>src_path</i> is relative to this directory and may contain subdirectories.
<i>dst_path</i>	The new name for the file. Note: <i>dst_path</i> is relative to this directory and may contain subdirectories.

**Returns**

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

**16.339.2.5 rmdir()**

```
virtual int L4Re::Vfs::Directory::rmdir (  
    const char * path) [pure virtual], [noexcept]
```

Delete an empty directory.

Backend for POSIX rmdir, rmdirat functions.

**Parameters**

<i>path</i>	The name of the directory to remove. Note: <i>path</i> is relative to this directory and may contain subdirectories.
-------------	----------------------------------------------------------------------------------------------------------------------

**Returns**

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 16.339.2.6 symlink()

```
virtual int L4Re::Vfs::Directory::symlink (
    const char * src_path,
    const char * dst_path) [pure virtual], [noexcept]
```

Create a symbolic link for the given file.

Backend for the POSIX symlink and symlinkat functions.

#### Parameters

<i>src_path</i>	The old name of the file. Note: <i>src_path</i> shall be an absolute path.
<i>dst_path</i>	The name for symlink. Note: <i>dst_path</i> is relative to this directory and may contain subdirectories.

#### Returns

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 16.339.2.7 unlink()

```
virtual int L4Re::Vfs::Directory::unlink (
    const char * path) [pure virtual], [noexcept]
```

Unlink the given file from that directory.

Backend for the POSIX unlink and unlinkat functions.

#### Parameters

<i>path</i>	The name of the file to unlink. Note: <i>path</i> is relative to this directory and may contain subdirectories.
-------------	-----------------------------------------------------------------------------------------------------------------

#### Returns

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

The documentation for this class was generated from the following file:

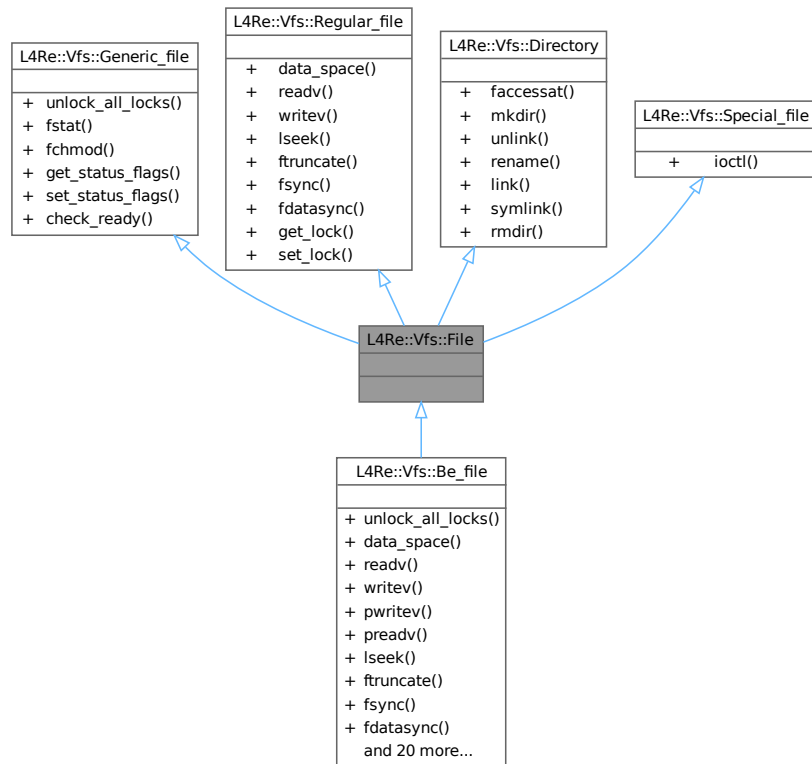
- l4/l4re\_vfs/vfs.h

## 16.340 L4Re::Vfs::File Class Reference

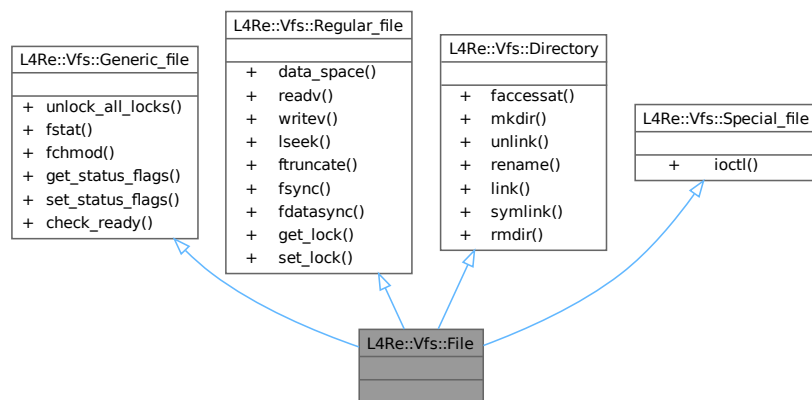
The basic interface for an open POSIX file.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::File:



Collaboration diagram for L4Re::Vfs::File:



## Additional Inherited Members

### Public Types inherited from [L4Re::Vfs::Generic\\_file](#)

- enum [Ready\\_type](#) : unsigned  
*Type of I/O operation/condition a file can indicate readiness.*

### Public Member Functions inherited from [L4Re::Vfs::Generic\\_file](#)

- virtual int [unlock\\_all\\_locks](#) () noexcept=0  
*Unlock all locks on the file.*
- virtual int [fstat](#) (struct stat64 \*buf) const noexcept=0  
*Get status information for the file.*
- virtual int [fchmod](#) (mode\_t) noexcept=0  
*Change POSIX access rights on the file.*
- virtual int [get\\_status\\_flags](#) () const noexcept=0  
*Get file status flags (fcntl F\_GETFL).*
- virtual int [set\\_status\\_flags](#) (long flags) noexcept=0  
*Set file status flags (fcntl F\_SETFL).*
- virtual bool [check\\_ready](#) ([Ready\\_type](#) rt) noexcept=0  
*Check whether the file is ready for an I/O operation/condition.*

### Public Member Functions inherited from [L4Re::Vfs::Regular\\_file](#)

- virtual [L4::Cap](#)< [L4Re::Dataspace](#) > [data\\_space](#) () noexcept=0  
*Get an [L4Re::Dataspace](#) object for the file.*
- virtual ssize\_t [readv](#) (const struct iovec \*, int iovcnt) noexcept=0  
*Read one or more blocks of data from the file.*
- virtual ssize\_t [writev](#) (const struct iovec \*, int iovcnt) noexcept=0  
*Write one or more blocks of data to the file.*
- virtual off64\_t [lseek](#) (off64\_t, int) noexcept=0  
*Change the file pointer.*
- virtual int [ftruncate](#) (off64\_t pos) noexcept=0  
*Truncate the file at the given position.*
- virtual int [fsync](#) () const noexcept=0  
*Sync the data and meta data to persistent storage.*
- virtual int [fdatasync](#) () const noexcept=0  
*Sync the data to persistent storage.*
- virtual int [get\\_lock](#) (struct flock64 \*lock) noexcept=0  
*Test if the given lock can be placed in the file.*
- virtual int [set\\_lock](#) (struct flock64 \*lock, bool wait) noexcept=0  
*Acquire or release the given lock on the file.*



## Public Member Functions inherited from [L4Re::Vfs::Directory](#)

- virtual int [faccessat](#) (const char \*path, int mode, int flags) noexcept=0  
*Check access permissions on the given file.*
- virtual int [mkdir](#) (const char \*path, mode\_t mode) noexcept=0  
*Create a new subdirectory.*
- virtual int [unlink](#) (const char \*path) noexcept=0  
*Unlink the given file from that directory.*
- virtual int [rename](#) (const char \*src\_path, const char \*dst\_path) noexcept=0  
*Rename the given file.*
- virtual int [link](#) (const char \*src\_path, const char \*dst\_path) noexcept=0  
*Create a hard link (second name) for the given file.*
- virtual int [symlink](#) (const char \*src\_path, const char \*dst\_path) noexcept=0  
*Create a symbolic link for the given file.*
- virtual int [rmdir](#) (const char \*path) noexcept=0  
*Delete an empty directory.*

## Public Member Functions inherited from [L4Re::Vfs::Special\\_file](#)

- virtual int [ioctl](#) (unsigned long cmd, va\_list args) noexcept=0  
*The famous IO control.*

### 16.340.1 Detailed Description

The basic interface for an open POSIX file.

An open POSIX file can be anything that hides behind a POSIX file descriptor. This means that even directories are files. An open file can be anything from a directory to a special device file so see [Generic\\_file](#), [Regular\\_file](#), [Directory](#), and [Special\\_file](#) for more information.

#### Note

For implementing a backend for the [L4Re::Vfs](#) [L4Re::Vfs::Be\\_file](#) may be used as a base class.

Definition at line 455 of file [vfs.h](#).

The documentation for this class was generated from the following file:

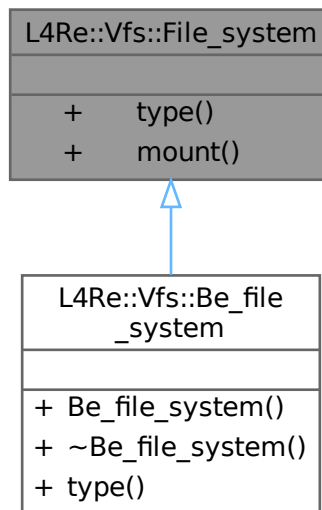
- [l4/l4re\\_vfs/vfs.h](#)

## 16.341 L4Re::Vfs::File\_system Class Reference

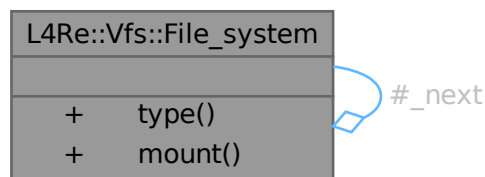
Basic interface for an [L4Re::Vfs](#) file system.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::File\_system:



Collaboration diagram for L4Re::Vfs::File\_system:



### Public Member Functions

- virtual char const \* [type](#) () const noexcept=0  
*Returns the type of the file system used in mount as fstype argument.*
- virtual int [mount](#) (char const \*source, unsigned long mountflags, void const \*data, [cxx::Ref\\_ptr](#)< [File](#) > \*dir) noexcept=0  
*Create a directory object dir representing source mounted with this file system.*

### 16.341.1 Detailed Description

Basic interface for an [L4Re::Vfs](#) file system.

#### Note

For implementing a special file system [L4Re::Vfs::Be\\_file\\_system](#) may be used as a base class.

The main purpose of this interface is to have a single object for each supported file-system type (e.g., ext2, vfat) that exists in the application and is registered at the [L4Re::Vfs::Fs](#) singleton available via [L4Re::Vfs::vfs\\_ops](#). Ultimately, the POSIX mount function calls the [File\\_system::mount](#) method matching the file-system type given in mount.

Definition at line 857 of file [vfs.h](#).

### 16.341.2 Member Function Documentation

#### 16.341.2.1 mount()

```
virtual int L4Re::Vfs::File_system::mount (
    char const * source,
    unsigned long mountflags,
    void const * data,
    cxx::Ref_ptr< File > * dir) [pure virtual], [noexcept]
```

Create a directory object *dir* representing *source* mounted with this file system.

#### Parameters

	<i>source</i>	The path to the source device to mount. This may also be some URL or anything file-system specific.
	<i>mountflags</i>	The mount flags as specified in the POSIX mount call.
	<i>data</i>	The data as specified in the POSIX mount call. The contents are file-system specific.
out	<i>dir</i>	A new directory object representing the file-system root directory.

#### Returns

0 on success, and <0 on error (e.g. -EINVAL).

References [mount\(\)](#).

Referenced by [mount\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.341.2.2 type()

```
virtual char const * L4Re::Vfs::File_system::type () const [pure virtual], [noexcept]
```

Returns the type of the file system used in mount as fstype argument.

#### Note

This method is already provided by [Be\\_file\\_system](#).

Implemented in [L4Re::Vfs::Be\\_file\\_system](#).

The documentation for this class was generated from the following file:

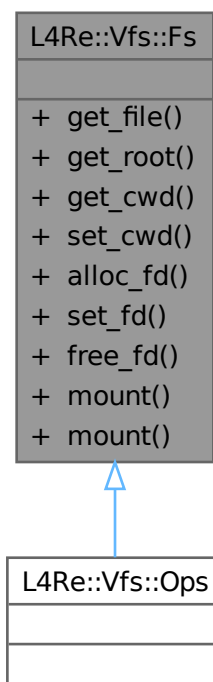
- `l4/l4re_vfs/vfs.h`

## 16.342 L4Re::Vfs::Fs Class Reference

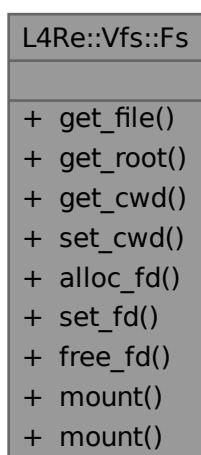
POSIX File-system related functionality.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Fs:



Collaboration diagram for L4Re::Vfs::Fs:



## Public Member Functions

- virtual `cxx::Ref_ptr< File > get_file` (int fd) noexcept=0  
*Get the `L4Re::Vfs::File` for the file descriptor fd.*
- virtual `cxx::Ref_ptr< File > get_root` () noexcept=0  
*Get the directory object for the application's root directory.*
- virtual `cxx::Ref_ptr< File > get_cwd` () noexcept  
*Get the directory object for the application's current working directory.*
- virtual void `set_cwd` (cxx::Ref\_ptr< File > const &) noexcept  
*Set the current working directory for the application.*
- virtual int `alloc_fd` (cxx::Ref\_ptr< File > const &f=cxx::Ref\_ptr<>::Nil) noexcept=0  
*Allocate the next free file descriptor.*
- virtual `cxx::Pair< cxx::Ref_ptr< File >, int > set_fd` (int fd, cxx::Ref\_ptr< File > const &f=cxx::Ref\_ptr<>::Nil) noexcept=0  
*Set the file object referenced by the file descriptor fd.*
- virtual `cxx::Ref_ptr< File > free_fd` (int fd) noexcept=0  
*Free the file descriptor fd.*
- virtual int `mount` (char const \*path, cxx::Ref\_ptr< File > const &dir) noexcept=0  
*Mount a given file object at the given global path in the VFS.*
- int `mount` (char const \*source, char const \*target, char const \*fstype, unsigned long mountflags, void const \*data) noexcept  
*Backend for the POSIX mount call.*

### 16.342.1 Detailed Description

POSIX File-system related functionality.

#### Note

This class usually exists as a singleton and as a superclass of `L4Re::Vfs::Ops` (

#### See also

`L4Re::Vfs::vfs_ops`).

Definition at line 946 of file `vfs.h`.

### 16.342.2 Member Function Documentation

#### 16.342.2.1 alloc\_fd()

```
virtual int L4Re::Vfs::Fs::alloc_fd (
    cxx::Ref_ptr< File > const & f = cxx::Ref_ptr<>::Nil) [pure virtual], [noexcept]
```

Allocate the next free file descriptor.

#### Parameters

<i>f</i>	The file to assign to that file descriptor.
----------	---------------------------------------------

**Returns**

The allocated file descriptor, or -EMFILE on error.

**16.342.2.2 free\_fd()**

```
virtual cxx::Ref_ptr< File > L4Re::Vfs::Fs::free_fd (
    int fd) [pure virtual], [noexcept]
```

Free the file descriptor *fd*.

**Parameters**

<i>fd</i>	The file descriptor to free.
-----------	------------------------------

**Returns**

A pointer to the file object that was assigned to the *fd*.

**16.342.2.3 get\_file()**

```
virtual cxx::Ref_ptr< File > L4Re::Vfs::Fs::get_file (
    int fd) [pure virtual], [noexcept]
```

Get the [L4Re::Vfs::File](#) for the file descriptor *fd*.

**Parameters**

<i>fd</i>	The POSIX file descriptor number.
-----------	-----------------------------------

**Returns**

A pointer to the [File](#) object, or 0 if *fd* is not open.

**16.342.2.4 mount()**

```
virtual int L4Re::Vfs::Fs::mount (
    char const * path,
    cxx::Ref_ptr< File > const & dir) [pure virtual], [noexcept]
```

Mount a given file object at the given global path in the VFS.

**Parameters**

<i>path</i>	The global path to mount <i>dir</i> at.
<i>dir</i>	A pointer to the file/directory object that shall be mounted at <i>path</i> .

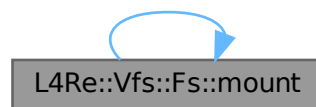
**Returns**

0 on success, or <0 on error.

References [mount\(\)](#).

Referenced by [mount\(\)](#), and [mount\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**16.342.2.5 set\_fd()**

```

virtual cxx::Pair< cxx::Ref_ptr< File >, int > L4Re::Vfs::Fs::set_fd (
    int fd,
    cxx::Ref_ptr< File > const & f = cxx::Ref_ptr<>::Nil) [pure virtual], [noexcept]
  
```

Set the file object referenced by the file descriptor *fd*.

**Parameters**

<i>fd</i>	The file descriptor to set to <i>f</i> .
-----------	------------------------------------------



<i>f</i>	The file object to assign.
----------	----------------------------

#### Returns

A pair of a pointer to the file object that was previously assigned to `fd` (`first`) and a return value (`second`). `second` contains `-#EBADF` if the passed file descriptor is outside the valid range. `first` contains a `Nil` pointer in that case. On success, `second` contains 0.

The documentation for this class was generated from the following files:

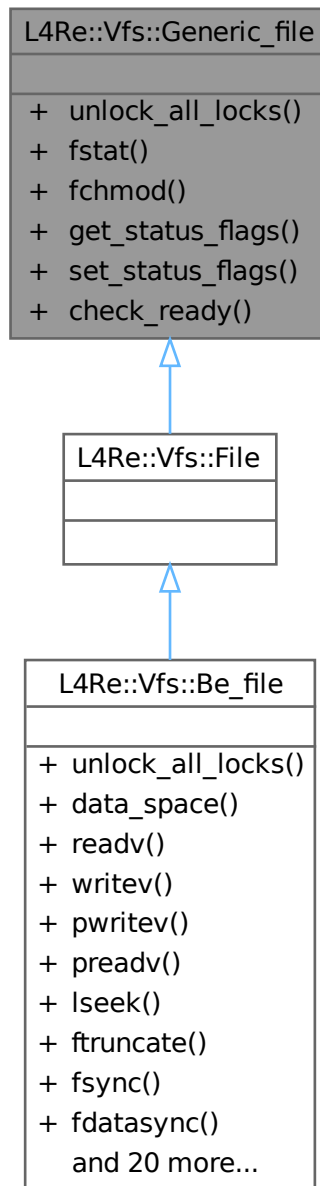
- `l4/l4re_vfs/vfs.h`
- `l4/l4re_vfs/impl/vfs_impl.h`

## 16.343 L4Re::Vfs::Generic\_file Class Reference

The common interface for an open POSIX file.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Generic\_file:



Collaboration diagram for L4Re::Vfs::Generic\_file:

L4Re::Vfs::Generic_file
<ul style="list-style-type: none"> <li>+ unlock_all_locks()</li> <li>+ fstat()</li> <li>+ fchmod()</li> <li>+ get_status_flags()</li> <li>+ set_status_flags()</li> <li>+ check_ready()</li> </ul>

## Public Types

- enum [Ready\\_type](#) : unsigned  
*Type of I/O operation/condition a file can indicate readiness.*

## Public Member Functions

- virtual int [unlock\\_all\\_locks](#) () noexcept=0  
*Unlock all locks on the file.*
- virtual int [fstat](#) (struct stat64 \*buf) const noexcept=0  
*Get status information for the file.*
- virtual int [fchmod](#) (mode\_t) noexcept=0  
*Change POSIX access rights on the file.*
- virtual int [get\\_status\\_flags](#) () const noexcept=0  
*Get file status flags (fcntl F\_GETFL).*
- virtual int [set\\_status\\_flags](#) (long flags) noexcept=0  
*Set file status flags (fcntl F\_SETFL).*
- virtual bool [check\\_ready](#) ([Ready\\_type](#) rt) noexcept=0  
*Check whether the file is ready for an I/O operation/condition.*

### 16.343.1 Detailed Description

The common interface for an open POSIX file.

This interface is common to all kinds of open files, independent of the file type (e.g., directory, regular file etc.). However, in the [L4Re::Vfs](#) the interface [File](#) is used for every real object.

See also

[L4Re::Vfs::File](#) for more information.

Definition at line 52 of file [vfs.h](#).

## 16.343.2 Member Enumeration Documentation

### 16.343.2.1 Ready\_type

```
enum L4Re::Vfs::Generic_file::Ready_type : unsigned
```

Type of I/O operation/condition a file can indicate readiness.

As defined by select() and similar functions.

Definition at line 60 of file [vfs.h](#).

## 16.343.3 Member Function Documentation

### 16.343.3.1 check\_ready()

```
virtual bool L4Re::Vfs::Generic_file::check_ready (
    Ready_type rt) [pure virtual], [noexcept]
```

Check whether the file is ready for an I/O operation/condition.

This method is used by the implementation of select() and similar functions.

#### Parameters

<i>rt</i>	Type of the I/O operation/condition to be ready, as defined by the select() and similar functions (Read, Write, Exception).
-----------	-----------------------------------------------------------------------------------------------------------------------------

#### Return values

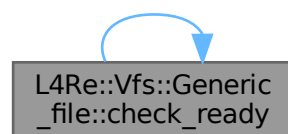
<i>true</i>	The file is ready for the given type of I/O operation/condition.
<i>false</i>	The file is not ready for the given type of I/O operation/condition.

Implemented in [L4Re::Vfs::Be\\_file](#).

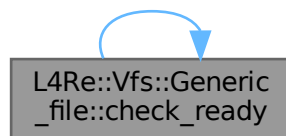
References [check\\_ready\(\)](#).

Referenced by [check\\_ready\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.343.3.2 fchmod()

```
virtual int L4Re::Vfs::Generic_file::fchmod (
    mode_t ) [pure virtual], [noexcept]
```

Change POSIX access rights on the file.

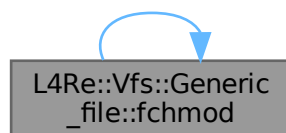
Backend for POSIX `chmod` and `fchmod`.

Implemented in [L4Re::Vfs::Be\\_file](#).

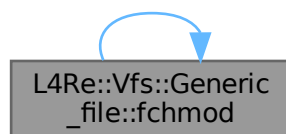
References [fchmod\(\)](#).

Referenced by [fchmod\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.343.3.3 `fstat()`

```
virtual int L4Re::Vfs::Generic_file::fstat (  
    struct stat64 * buf) const [pure virtual], [noexcept]
```

Get status information for the file.

This is the backend for POSIX `fstat`, `stat`, `fstat64` and friends.

#### Parameters

<code>out</code>	<code>buf</code>	This buffer is filled with the status information.
------------------	------------------	----------------------------------------------------

#### Returns

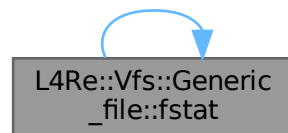
0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

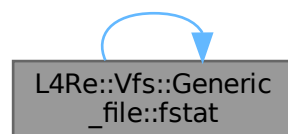
References [fstat\(\)](#).

Referenced by [fstat\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 16.343.3.4 get\_status\_flags()

```
virtual int L4Re::Vfs::Generic_file::get_status_flags () const [pure virtual], [noexcept]
```

Get file status flags (fcntl F\_GETFL).

This function is used by the fcntl implementation for the F\_GETFL command.

##### Returns

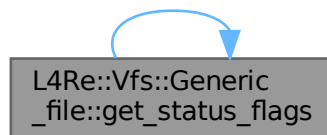
flags such as O\_RDONLY, O\_WRONLY, O\_RDWR, O\_DIRECT, O\_ASYNC, O\_NOATIME, O\_NONBLOCK, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

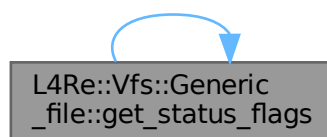
References [get\\_status\\_flags\(\)](#).

Referenced by [get\\_status\\_flags\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 16.343.3.5 set\_status\_flags()

```
virtual int L4Re::Vfs::Generic_file::set_status_flags (
    long flags) [pure virtual], [noexcept]
```

Set file status flags (fcntl F\_SETFL).

This function is used by the fcntl implementation for the F\_SETFL command.

##### Parameters

<i>flags</i>	The file status flags to set. This must be a combination of <code>O_RDONLY</code> , <code>O_WRONLY</code> , <code>O_RDWR</code> , <code>O_APPEND</code> , <code>O_ASYNC</code> , <code>O_DIRECT</code> , <code>O_NOATIME</code> , <code>O_NONBLOCK</code> .
--------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Note**

Creation flags such as `O_CREAT`, `O_EXCL`, `O_NOCTTY`, `O_TRUNC` are ignored.

**Returns**

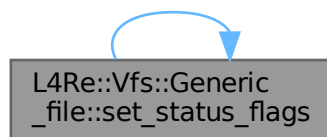
0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

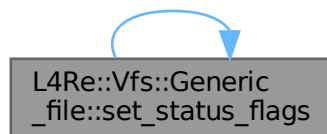
References [set\\_status\\_flags\(\)](#).

Referenced by [set\\_status\\_flags\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:





### 16.343.3.6 unlock\_all\_locks()

```
virtual int L4Re::Vfs::Generic_file::unlock_all_locks () [pure virtual], [noexcept]
```

Unlock all locks on the file.

#### Note

All locks means all locks independent of which file the locks were taken by.

This method is called by the POSIX close implementation to get the POSIX semantics of releasing all locks taken by this application on a close for any fd referencing the real file.

#### Returns

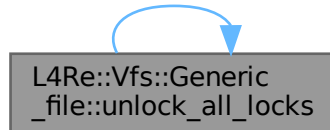
0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

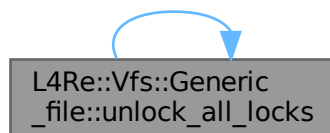
References [unlock\\_all\\_locks\(\)](#).

Referenced by [unlock\\_all\\_locks\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

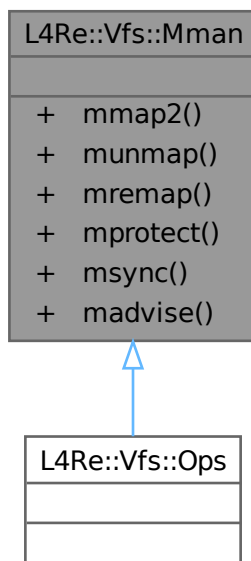
- `l4/l4re_vfs/vfs.h`

## 16.344 L4Re::Vfs::Mman Class Reference

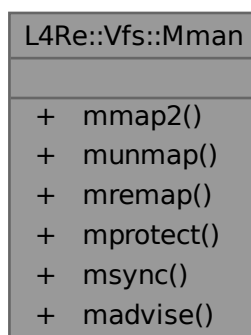
Interface for POSIX memory management.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Mman:



Collaboration diagram for L4Re::Vfs::Mman:



## Public Member Functions

- virtual int **mmap2** (void \*start, size\_t len, int prot, int flags, int fd, off\_t offset, void \*\*ptr) noexcept=0  
*Backend for the mmap2 system call.*
- virtual int **munmap** (void \*start, size\_t len) noexcept=0  
*Backend for the munmap system call.*
- virtual int **mremap** (void \*old, size\_t old\_sz, size\_t new\_sz, int flags, void \*\*new\_addr) noexcept=0  
*Backend for the mremap system call.*
- virtual int **mprotect** (const void \*a, size\_t sz, int prot) noexcept=0  
*Backend for the mprotect system call.*
- virtual int **msync** (void \*addr, size\_t len, int flags) noexcept=0  
*Backend for the msync system call.*
- virtual int **madvise** (void \*addr, size\_t len, int advice) noexcept=0  
*Backend for the madvise system call.*

### 16.344.1 Detailed Description

Interface for POSIX memory management.

#### Note

This interface usually exists as a singleton and as a superclass of [L4Re::Vfs::Ops](#).

An implementation for this interface is in [l4/l4re\\_vfs/impl/vfs\\_impl.h](#) and used by the l4re\_vfs library or by the VFS implementation in ldso.

Definition at line 773 of file [vfs.h](#).

The documentation for this class was generated from the following file:

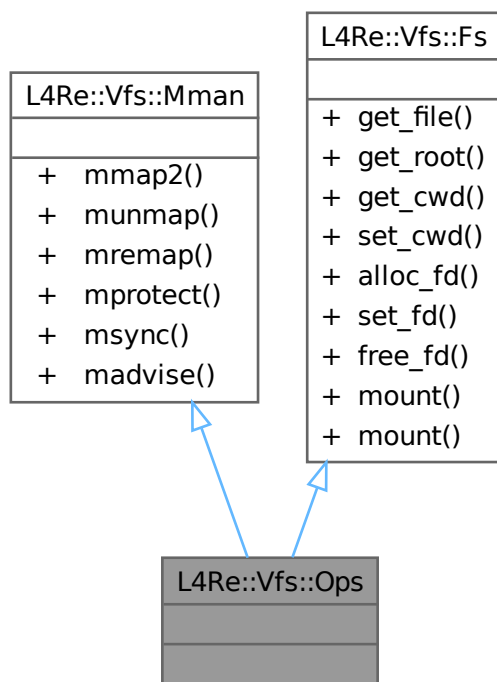
- l4/l4re\_vfs/vfs.h

## 16.345 L4Re::Vfs::Ops Class Reference

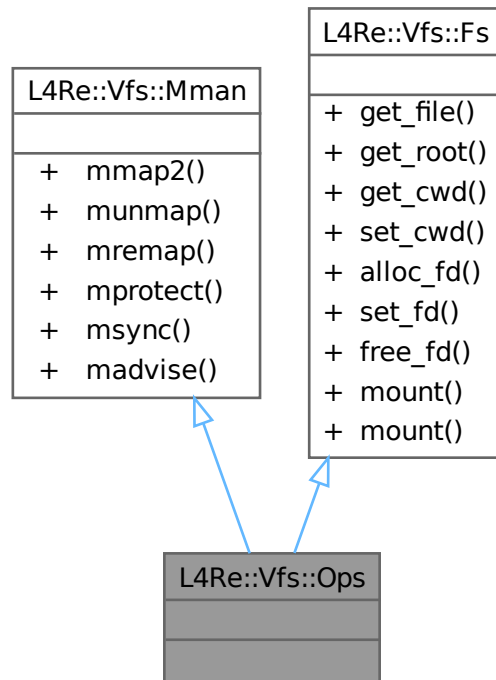
Interface for the POSIX backends of an application.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Ops:



Collaboration diagram for L4Re::Vfs::Ops:



### Additional Inherited Members

#### Public Member Functions inherited from L4Re::Vfs::Mman

- virtual int **mmap2** (void \*start, size\_t len, int prot, int flags, int fd, off\_t offset, void \*\*ptr) noexcept=0  
*Backend for the mmap2 system call.*
- virtual int **munmap** (void \*start, size\_t len) noexcept=0  
*Backend for the munmap system call.*
- virtual int **mremap** (void \*old, size\_t old\_sz, size\_t new\_sz, int flags, void \*\*new\_addr) noexcept=0  
*Backend for the mremap system call.*
- virtual int **mprotect** (const void \*a, size\_t sz, int prot) noexcept=0  
*Backend for the mprotect system call.*
- virtual int **msync** (void \*addr, size\_t len, int flags) noexcept=0  
*Backend for the msync system call.*
- virtual int **madvise** (void \*addr, size\_t len, int advice) noexcept=0  
*Backend for the madvice system call.*

#### Public Member Functions inherited from L4Re::Vfs::Fs

- virtual [cxx::Ref\\_ptr< File >](#) **get\_file** (int fd) noexcept=0  
*Get the [L4Re::Vfs::File](#) for the file descriptor fd.*

- virtual `cxx::Ref_ptr< File > get_root ()` noexcept=0  
*Get the directory object for the application's root directory.*
- virtual `cxx::Ref_ptr< File > get_cwd ()` noexcept  
*Get the directory object for the application's current working directory.*
- virtual void `set_cwd (cxx::Ref_ptr< File > const &)` noexcept  
*Set the current working directory for the application.*
- virtual int `alloc_fd (cxx::Ref_ptr< File > const &f=cxx::Ref_ptr<>::Nil)` noexcept=0  
*Allocate the next free file descriptor.*
- virtual `cxx::Pair< cxx::Ref_ptr< File >, int > set_fd (int fd, cxx::Ref_ptr< File > const &f=cxx::Ref_ptr<>::Nil)` noexcept=0  
*Set the file object referenced by the file descriptor fd.*
- virtual `cxx::Ref_ptr< File > free_fd (int fd)` noexcept=0  
*Free the file descriptor fd.*
- virtual int `mount (char const *path, cxx::Ref_ptr< File > const &dir)` noexcept=0  
*Mount a given file object at the given global path in the VFS.*
- int `mount (char const *source, char const *target, char const *fstype, unsigned long mountflags, void const *data)` noexcept  
*Backend for the POSIX mount call.*

### 16.345.1 Detailed Description

Interface for the POSIX backends of an application.

#### Note

There usually exists a single instance of this interface available via `L4Re::Vfs::vfs_ops` that is used for all kinds of C-Library functions.

Definition at line 1109 of file `vfs.h`.

The documentation for this class was generated from the following file:

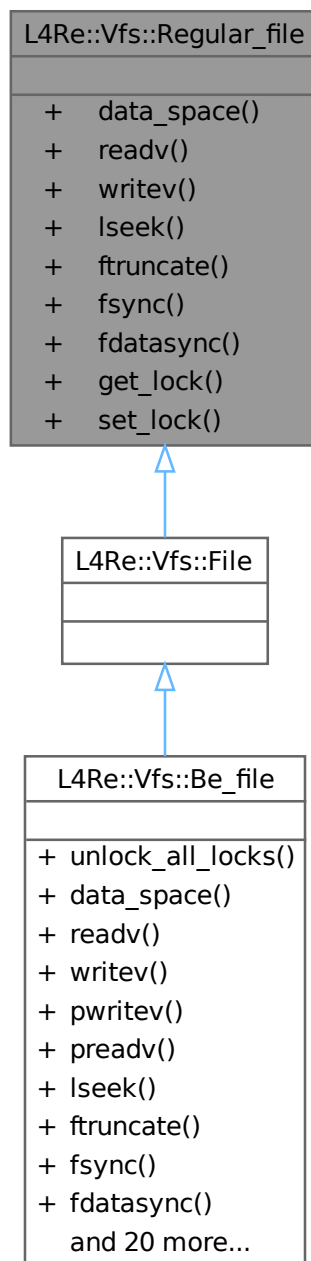
- `l4/l4re_vfs/vfs.h`

## 16.346 L4Re::Vfs::Regular\_file Class Reference

Interface for a POSIX file that provides regular file semantics.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Regular\_file:



Collaboration diagram for `L4Re::Vfs::Regular_file`:

L4Re::Vfs::Regular_file	
+	<code>data_space()</code>
+	<code>readv()</code>
+	<code>writew()</code>
+	<code>lseek()</code>
+	<code>ftruncate()</code>
+	<code>fsync()</code>
+	<code>fdatasync()</code>
+	<code>get_lock()</code>
+	<code>set_lock()</code>

### Public Member Functions

- virtual `L4::Cap< L4Re::Dataspace > data_space ()` noexcept=0  
*Get an `L4Re::Dataspace` object for the file.*
- virtual `ssize_t readv (const struct iovec *, int iovcnt)` noexcept=0  
*Read one or more blocks of data from the file.*
- virtual `ssize_t writew (const struct iovec *, int iovcnt)` noexcept=0  
*Write one or more blocks of data to the file.*
- virtual `off64_t lseek (off64_t, int)` noexcept=0  
*Change the file pointer.*
- virtual `int ftruncate (off64_t pos)` noexcept=0  
*Truncate the file at the given position.*
- virtual `int fsync ()` const noexcept=0  
*Sync the data and meta data to persistent storage.*
- virtual `int fdatasync ()` const noexcept=0  
*Sync the data to persistent storage.*
- virtual `int get_lock (struct flock64 *lock)` noexcept=0  
*Test if the given lock can be placed in the file.*
- virtual `int set_lock (struct flock64 *lock, bool wait)` noexcept=0  
*Acquire or release the given lock on the file.*

### 16.346.1 Detailed Description

Interface for a POSIX file that provides regular file semantics.

Real objects always use the combined `L4Re::Vfs::File` interface.

Definition at line 287 of file `vfs.h`.



## 16.346.2 Member Function Documentation

### 16.346.2.1 data\_space()

```
virtual L4::Cap< L4Re::Dataspace > L4Re::Vfs::Regular_file::data_space () [pure virtual],  
[noexcept]
```

Get an [L4Re::Dataspace](#) object for the file.

This is used as a backend for POSIX mmap and mmap2 functions.

#### Note

mmap is not possible if the function returns an invalid capability.

#### Returns

A capability to an [L4Re::Dataspace](#) that represents the file contents in an [L4Re](#) way.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 16.346.2.2 fdatsync()

```
virtual int L4Re::Vfs::Regular_file::fdatsync () const [pure virtual], [noexcept]
```

Sync the data to persistent storage.

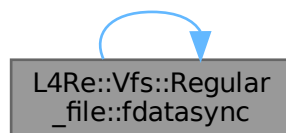
This is the backend for POSIX fdatsync.

Implemented in [L4Re::Vfs::Be\\_file](#).

References [fdatsync\(\)](#).

Referenced by [fdatsync\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.346.2.3 fsync()

```
virtual int L4Re::Vfs::Regular_file::fsync () const [pure virtual], [noexcept]
```

Sync the data and meta data to persistent storage.

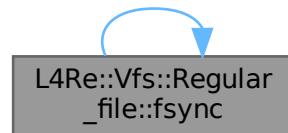
This is the backend for POSIX fsync.

Implemented in [L4Re::Vfs::Be\\_file](#).

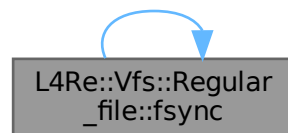
References [fsync\(\)](#).

Referenced by [fsync\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.346.2.4 ftruncate()

```
virtual int L4Re::Vfs::Regular_file::ftruncate (  
    off64_t pos) [pure virtual], [noexcept]
```

Truncate the file at the given position.

This function is the backend for truncate and friends.

#### Parameters

---

<i>pos</i>	The offset at which the file shall be truncated.
------------	--------------------------------------------------

**Returns**

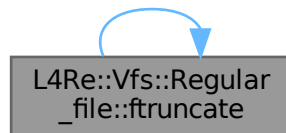
0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

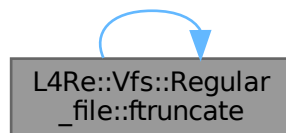
References [ftruncate\(\)](#).

Referenced by [ftruncate\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**16.346.2.5 get\_lock()**

```
virtual int L4Re::Vfs::Regular_file::get_lock (
    struct flock64 * lock) [pure virtual], [noexcept]
```

Test if the given lock can be placed in the file.

This function is used as backend for fcntl F\_GETLK commands.

**Parameters**

<i>lock</i>	The lock that shall be placed on the file. The <i>l_type</i> member will contain <code>F_UNLCK</code> if the lock could be placed.
-------------	------------------------------------------------------------------------------------------------------------------------------------

#### Returns

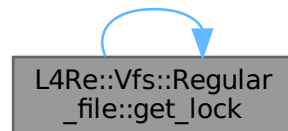
0 on success, <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

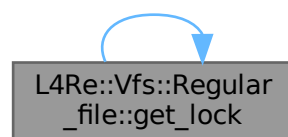
References [get\\_lock\(\)](#).

Referenced by [get\\_lock\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 16.346.2.6 lseek()

```
virtual off64_t L4Re::Vfs::Regular_file::lseek (
    off64_t ,
    int ) [pure virtual], [noexcept]
```

Change the file pointer.

This is the backend for POSIX seek, lseek and friends.

#### Returns

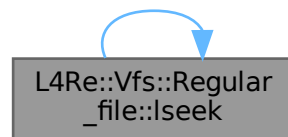
The new file position, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

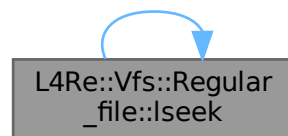
References [lseek\(\)](#).

Referenced by [lseek\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 16.346.2.7 readv()

```
virtual ssize_t L4Re::Vfs::Regular_file::readv (  
    const struct iovec * ,  
    int iovcnt) [pure virtual], [noexcept]
```

Read one or more blocks of data from the file.

This function acts as backend for POSIX read and readv calls and reads data starting from the `f_pos` pointer of that open file. The file pointer is advanced according to the number of bytes read.

**Returns**

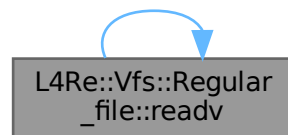
The number of bytes read from the file, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

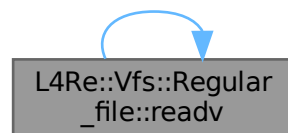
References [readv\(\)](#).

Referenced by [readv\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**16.346.2.8 set\_lock()**

```
virtual int L4Re::Vfs::Regular_file::set_lock (  
    struct flock64 * lock,  
    bool wait) [pure virtual], [noexcept]
```

Acquire or release the given lock on the file.

This function is used as backend for fcntl `F_SETLK` and `F_SETLKW` commands.

**Parameters**

<i>lock</i>	The lock that shall be placed on the file.
-------------	--------------------------------------------

<i>wait</i>	If true, then block if there is a conflicting lock on the file.
-------------	-----------------------------------------------------------------

**Returns**

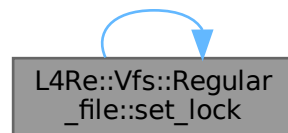
0 on success, <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

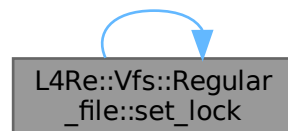
References [set\\_lock\(\)](#).

Referenced by [set\\_lock\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**16.346.2.9 writev()**

```
virtual ssize_t L4Re::Vfs::Regular_file::writev (
    const struct iovec * ,
    int iovcnt) [pure virtual], [noexcept]
```

Write one or more blocks of data to the file.

This function acts as backend for POSIX write and writev calls. The data is written starting at the current file pointer and the file pointer must be advanced according to the number of written bytes.

### Returns

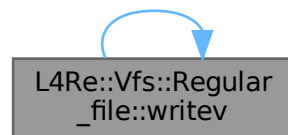
The number of bytes written to the file, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

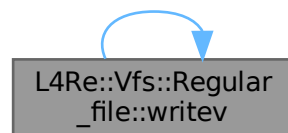
References [writev\(\)](#).

Referenced by [writev\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- `l4/l4re_vfs/vfs.h`

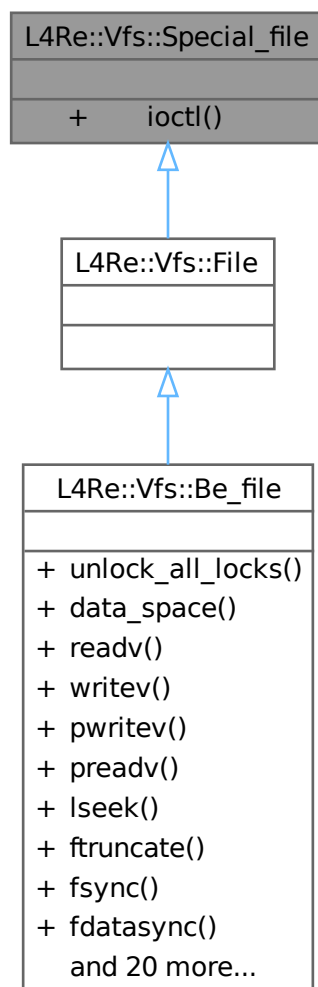
## 16.347 L4Re::Vfs::Special\_file Class Reference

Interface for a POSIX file that provides special file semantics.

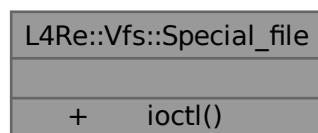
```
#include <vfs.h>
```



Inheritance diagram for L4Re::Vfs::Special\_file:



Collaboration diagram for L4Re::Vfs::Special\_file:



## Public Member Functions

- virtual int [ioctl](#) (unsigned long cmd, va\_list args) noexcept=0  
*The famous IO control.*

### 16.347.1 Detailed Description

Interface for a POSIX file that provides special file semantics.

Real objects always use the combined [L4Re::Vfs::File](#) interface.

Definition at line [420](#) of file [vfs.h](#).

### 16.347.2 Member Function Documentation

#### 16.347.2.1 ioctl()

```
virtual int L4Re::Vfs::Special_file::ioctl (  
    unsigned long cmd,  
    va_list args) [pure virtual], [noexcept]
```

The famous IO control.

Backend for POSIX generic object invocation ioctl.

#### Parameters

<i>cmd</i>	The ioctl command.
<i>args</i>	The arguments for the ioctl, usually some kind of pointer.

#### Returns

>=0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

The documentation for this class was generated from the following file:

- [l4/l4re\\_vfs/vfs.h](#)

## 16.348 L4Re::Video::Color\_component Class Reference

A color component.

```
#include <colors>
```

Collaboration diagram for L4Re::Video::Color\_component:

L4Re::Video::Color_component
<ul style="list-style-type: none"> <li>+ Color_component()</li> <li>+ Color_component()</li> <li>+ size()</li> <li>+ shift()</li> <li>+ operator==( )</li> <li>+ get()</li> <li>+ set()</li> <li>+ dump()</li> </ul>

### Public Member Functions

- **Color\_component ( )**  
*Constructor.*
- **Color\_component** (unsigned char bits, unsigned char shift)  
*Constructor.*
- unsigned char **size** ( ) const  
*Return the number of bits used by the component.*
- unsigned char **shift** ( ) const  
*Return the position of the component in the pixel.*
- bool **operator==** (Color\_component const &o) const  
*Compare for equality.*
- int **get** (unsigned long v) const  
*Get component from value (normalized to 16bits).*
- long unsigned **set** (int v) const  
*Transform 16bit normalized value to the component in the color space.*
- template<typename OUT>  
void **dump** (OUT &s) const  
*Dump information on the view information to a stream.*

### 16.348.1 Detailed Description

A color component.

Definition at line 21 of file [colors](#).

## 16.348.2 Constructor & Destructor Documentation

### 16.348.2.1 Color\_component()

```
L4Re::Video::Color_component::Color_component (
    unsigned char bits,
    unsigned char shift) [inline]
```

Constructor.

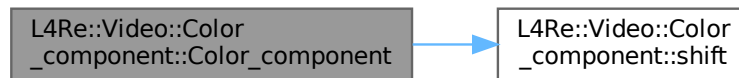
#### Parameters

<i>bits</i>	Number of bits used by the component
<i>shift</i>	Position in bits of the component in the pixel

Definition at line 36 of file [colors](#).

References [shift\(\)](#).

Here is the call graph for this function:



## 16.348.3 Member Function Documentation

### 16.348.3.1 dump()

```
template<typename OUT>
void L4Re::Video::Color_component::dump (
    OUT & s) const [inline]
```

Dump information on the view information to a stream.

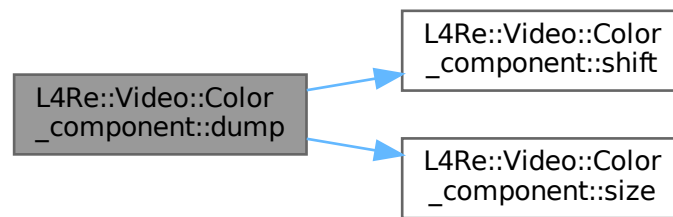
#### Parameters

<i>s</i>	Stream
----------	--------

Definition at line 81 of file [colors](#).

References [shift\(\)](#), and [size\(\)](#).

Here is the call graph for this function:



### 16.348.3.2 `get()`

```
int L4Re::Video::Color_component::get (  
    unsigned long v) const [inline]
```

Get component from value (normalized to 16bits).

#### Parameters

<i>v</i>	Value
----------	-------

#### Returns

Converted value

Definition at line 63 of file [colors](#).

### 16.348.3.3 `operator==()`

```
bool L4Re::Video::Color_component::operator== (  
    Color_component const & o) const [inline]
```

Compare for equality.

**Returns**

True if the same components are described, false if not.

Definition at line 55 of file [colors](#).

References [Color\\_component\(\)](#).

Here is the call graph for this function:

**16.348.3.4 set()**

```
long unsigned L4Re::Video::Color_component::set (
    int v) const [inline]
```

Transform 16bit normalized value to the component in the color space.

**Parameters**

<i>v</i>	Value return Converted value.
----------	-------------------------------

Definition at line 73 of file [colors](#).

**16.348.3.5 shift()**

```
unsigned char L4Re::Video::Color_component::shift () const [inline]
```

Return the position of the component in the pixel.

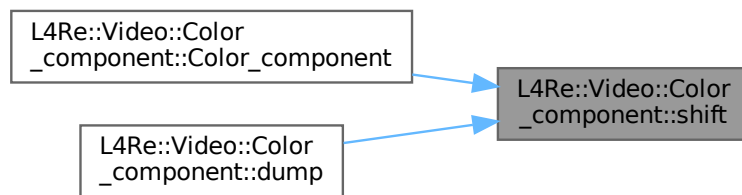
**Returns**

Position in bits of the component in the pixel

Definition at line 49 of file [colors](#).

Referenced by [Color\\_component\(\)](#), and [dump\(\)](#).

Here is the caller graph for this function:



### 16.348.3.6 size()

```
unsigned char L4Re::Video::Color_component::size () const [inline]
```

Return the number of bits used by the component.

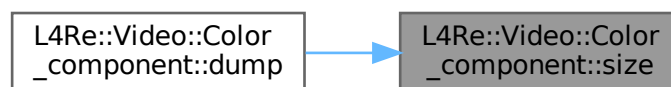
#### Returns

Number of bits used by the component

Definition at line 43 of file [colors](#).

Referenced by [dump\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

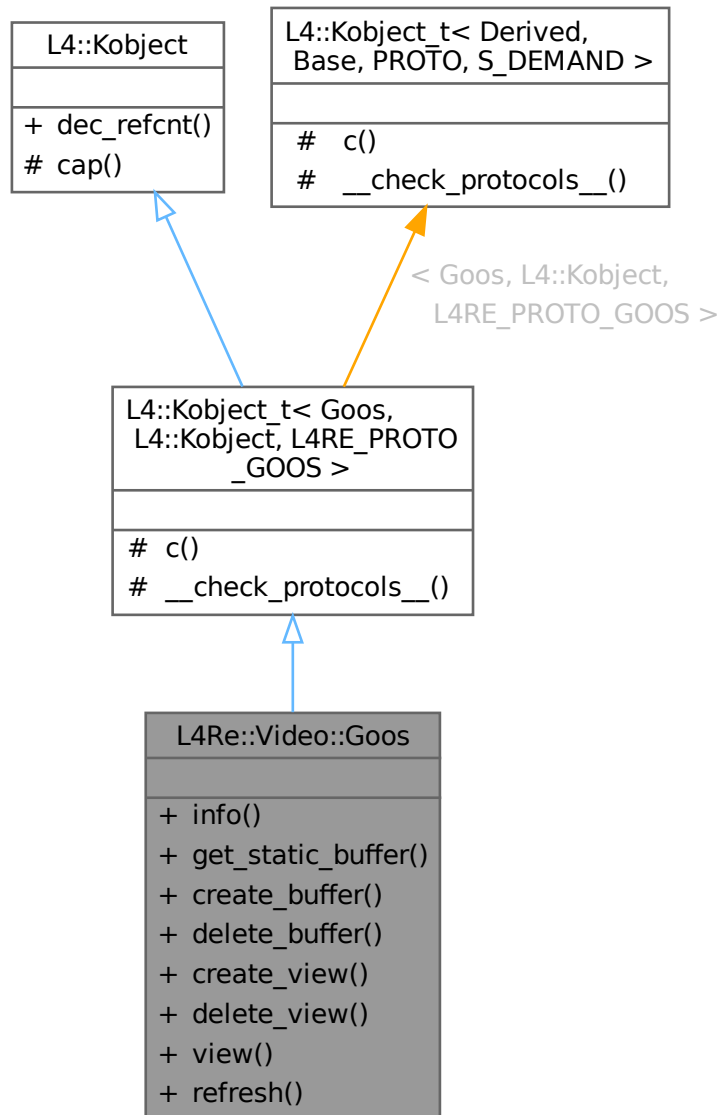
- `l4/re/video/colors`

## 16.349 L4Re::Video::Goos Class Reference

Class that abstracts framebuffers.

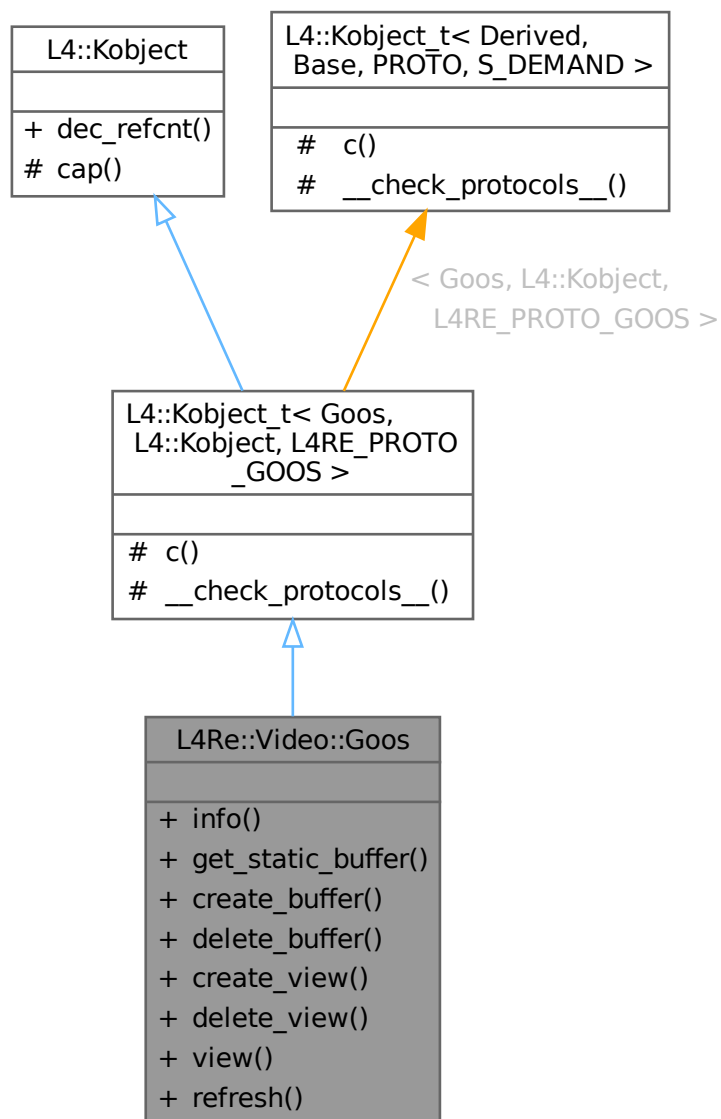
```
#include <goos>
```

Inheritance diagram for L4Re::Video::Goos:





Collaboration diagram for L4Re::Video::Goos:



## Data Structures

- struct [Info](#)

Information structure of a [Goos](#).

## Public Types

- enum [Flags](#) { [F\\_auto\\_refresh](#) = 0x01 , [F\\_pointer](#) = 0x02 , [F\\_dynamic\\_views](#) = 0x04 , [F\\_dynamic\\_buffers](#) = 0x08 }

[Flags](#) for a [Goos](#).

## Public Member Functions

- long **info** (Info \*info)  
*Return the Goos information of the Goos.*
- long **get\_static\_buffer** (unsigned idx, L4::lpc::Out< L4::Cap< L4Re::Dataspace > > rbuf)  
*Return a static buffer of a Goos.*
- long **create\_buffer** (unsigned long size, L4::lpc::Out< L4::Cap< L4Re::Dataspace > > rbuf)  
*Create a buffer.*
- long **delete\_buffer** (unsigned idx)  
*Delete a buffer.*
- int **create\_view** (View \*view, l4\_utcb\_t \*utcb=l4\_utcb()) const noexcept  
*Create a view.*
- int **delete\_view** (View const &v, l4\_utcb\_t \*utcb=l4\_utcb()) const noexcept  
*Delete a view.*
- View **view** (unsigned index) const noexcept  
*Return a view.*
- long **refresh** (int x, int y, int w, int h)  
*Trigger refreshing of the given area on the virtual screen.*

## Public Member Functions inherited from L4::Kobject

- l4\_msgtag\_t **dec\_refcnt** (l4\_mword\_t diff, l4\_utcb\_t \*utcb=l4\_utcb())  
*Decrement the in kernel reference counter for the object.*

## Additional Inherited Members

### Protected Types inherited from

**L4::Kobject\_t< Goos, L4::Kobject, L4RE\_PROTO\_GOOS >**

- typedef Goos **Class**  
*The target interface type (inheriting from Kobject\_t).*
- typedef Typeid::Iface< PROTO, Goos > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< \_\_Iface >, typename L4::Kobject::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Member Functions inherited from

**L4::Kobject\_t< Goos, L4::Kobject, L4RE\_PROTO\_GOOS >**

- L4::Cap< Class > **c** () const noexcept  
*Get the capability to ourselves.*

## Protected Member Functions inherited from L4::Kobject

- l4\_cap\_idx\_t **cap** () const noexcept  
*Return capability selector.*

## Static Protected Member Functions inherited from [L4::Kobject\\_t](#)< [Goos](#), [L4::Kobject](#), [L4RE\\_PROTO\\_GOOS](#) >

- static void `__check_protocols__()` noexcept  
*Helper to check for protocol conflicts.*

### 16.349.1 Detailed Description

[Class](#) that abstracts framebuffer.

A framebuffer is the pixel data that is displayed on a screen and a [Goos](#) object lets the user manipulate that data. A [Goos](#) makes use of two kinds of objects:

- Buffers in the form of [L4Re::Dataspace](#) objects. These hold the bytes for the pixel data.
- [L4Re::Video::View](#) objects.

Both can either be static, that is their number and configuration is fixed and determined by the framebuffer, or they can be dynamic, with the user allocating them.

Definition at line [223](#) of file [goos](#).

### 16.349.2 Member Enumeration Documentation

#### 16.349.2.1 Flags

enum [L4Re::Video::Goos::Flags](#)

[Flags](#) for a [Goos](#).

#### Enumerator

<code>F_auto_refresh</code>	The graphics display is automatically refreshed.
<code>F_pointer</code>	We have a mouse pointer.
<code>F_dynamic_views</code>	Supports dynamically allocated views.
<code>F_dynamic_buffers</code>	Supports dynamically allocated buffers.

Definition at line [228](#) of file [goos](#).

### 16.349.3 Member Function Documentation

#### 16.349.3.1 `create_buffer()`

```
long L4Re::Video::Goos::create_buffer (
    unsigned long size,
    L4::Ipc::Out< L4::Cap< L4Re::Dataspace > > rbuf)
```

Create a buffer.

#### Parameters

---

<i>size</i>	Size of buffer in bytes.
<i>rbuf</i>	Capability slot to point the buffer dataspace to.

### Return values

$\geq 0$	Success, the value returned is the buffer index.
$< 0$	Error

Referenced by [get\\_static\\_buffer\(\)](#).

Here is the caller graph for this function:



### 16.349.3.2 create\_view()

```
int L4Re::Video::Goos::create_view (
    View * view,
    l4_utcb_t * utcb = l4_utcb()) const [inline], [noexcept]
```

Create a view.

### Parameters

out	<i>view</i>	A view object.
	<i>utcb</i>	UTCB of the caller. This is a default parameter.

### Return values

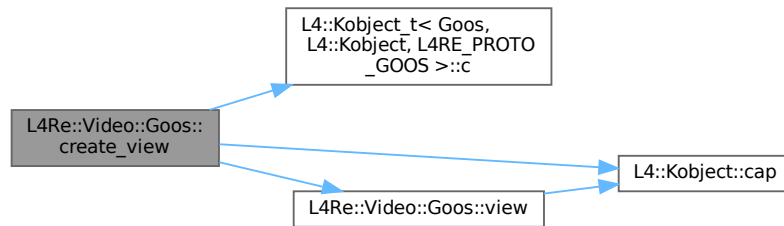
$\geq 0$	Success, the value returned is the view index.
$< 0$	Error

Definition at line 312 of file [goos](#).

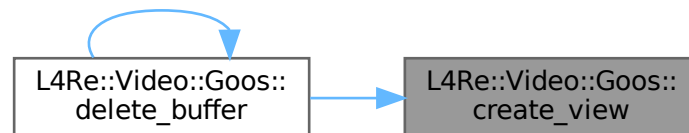
References [L4::Kobject\\_t< Goos, L4::Kobject, L4RE\\_PROTO\\_GOOS >::c\(\)](#), [L4::Kobject::cap\(\)](#), and [view\(\)](#).

Referenced by [delete\\_buffer\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.349.3.3 delete\_buffer()

```
long L4Re::Video::Goos::delete_buffer (
    unsigned idx)
```

Delete a buffer.

#### Parameters

<i>idx</i>	Buffer to delete.
------------	-------------------

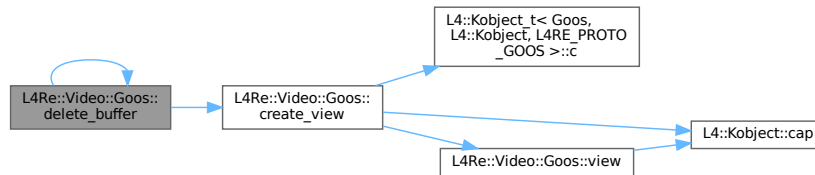
#### Return values

0	Success
<0	Error

References [create\\_view\(\)](#), and [delete\\_buffer\(\)](#).

Referenced by [delete\\_buffer\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 16.349.3.4 delete\_view()

```

int L4Re::Video::Goos::delete_view (
    View const & v,
    l4_utcb_t * utcb = l4_utcb()) const [inline], [noexcept]

```

Delete a view.

##### Parameters

<i>v</i>	The view object to delete.
<i>utcb</i>	UTCB of the caller. This is a default parameter.

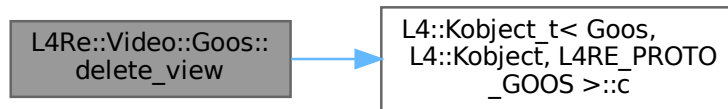
##### Return values

0	Success
<0	Error

Definition at line 332 of file [goos](#).

References [L4::Kobject\\_t< Goos, L4::Kobject, L4RE\\_PROTO\\_GOOS >::c\(\)](#).

Here is the call graph for this function:



### 16.349.3.5 get\_static\_buffer()

```
long L4Re::Video::Goos::get_static_buffer (
    unsigned idx,
    L4::Ipc::Out< L4::Cap< L4Re::Dataspace > > rbuf)
```

Return a static buffer of a [Goos](#).

#### Parameters

<i>idx</i>	Index of the static buffer.
<i>rbuf</i>	Capability slot to point the buffer dataspace to.

#### Return values

0	Success
<0	Error

References [create\\_buffer\(\)](#).

Referenced by [info\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.349.3.6 info()

```
long L4Re::Video::Goos::info (
    Info * info)
```

Return the [Goos](#) information of the [Goos](#).

#### Parameters

out	info	<a href="#">Goos</a> information structure pointer.
-----	------	-----------------------------------------------------

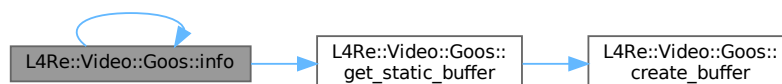
#### Return values

0	Success
<0	Error

References [get\\_static\\_buffer\(\)](#), and [info\(\)](#).

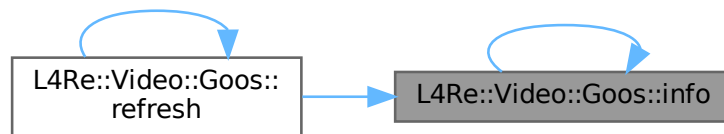
Referenced by [info\(\)](#), and [refresh\(\)](#).

Here is the call graph for this function:





Here is the caller graph for this function:



### 16.349.3.7 view()

```
View L4Re::Video::Goos::view (
    unsigned index) const [inline], [noexcept]
```

Return a view.

#### Parameters

<i>index</i>	Index of the view to return.
--------------	------------------------------

#### Returns

The view.

Definition at line 363 of file [goos](#).

References [L4::Kobject::cap\(\)](#).

Referenced by [create\\_view\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

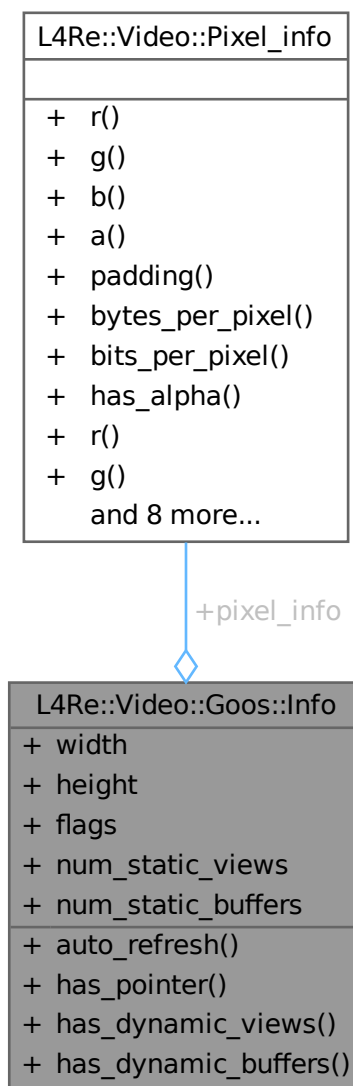
- `I4/re/video/goos`

## 16.350 L4Re::Video::Goos::Info Struct Reference

Information structure of a [Goos](#).

```
#include <goos>
```

Collaboration diagram for L4Re::Video::Goos::Info:



## Public Member Functions

- bool **auto\_refresh** () const  
Return whether this [Goos](#) does auto refreshing or the view refresh functions must be used to make changes visible.
- bool **has\_pointer** () const  
Return whether a pointer is used by the provider of the [Goos](#).
- bool **has\_dynamic\_views** () const  
Return whether dynamic view are supported.
- bool **has\_dynamic\_buffers** () const  
Return whether dynamic buffers are supported.

## Data Fields

- unsigned long **width**  
*Width.*
- unsigned long **height**  
*Height.*
- unsigned **flags**  
*Flags, see [Flags](#).*
- unsigned **num\_static\_views**  
*Number of static view.*
- unsigned **num\_static\_buffers**  
*Number of static buffers.*
- [Pixel\\_info](#) **pixel\_info**  
*Pixel information.*

### 16.350.1 Detailed Description

Information structure of a [Goos](#).

Definition at line [237](#) of file [goos](#).

The documentation for this struct was generated from the following file:

- [l4/re/video/goos](#)

## 16.351 L4Re::Video::Pixel\_info Class Reference

Pixel information.

```
#include <colors>
```

Collaboration diagram for L4Re::Video::Pixel\_info:

L4Re::Video::Pixel_info
<ul style="list-style-type: none"> <li>+ r()</li> <li>+ g()</li> <li>+ b()</li> <li>+ a()</li> <li>+ padding()</li> <li>+ bytes_per_pixel()</li> <li>+ bits_per_pixel()</li> <li>+ has_alpha()</li> <li>+ r()</li> <li>+ g()</li> <li>and 8 more...</li> </ul>

## Public Member Functions

- [Color\\_component](#) const & [r](#) () const  
*Return the red color compoment of the pixel.*
- [Color\\_component](#) const & [g](#) () const  
*Return the green color compoment of the pixel.*
- [Color\\_component](#) const & [b](#) () const  
*Return the blue color compoment of the pixel.*
- [Color\\_component](#) const & [a](#) () const  
*Return the alpha color compoment of the pixel.*
- [Color\\_component](#) const [padding](#) () const  
*Compute the padding pseudo component.*
- unsigned char [bytes\\_per\\_pixel](#) () const  
*Query size of pixel in bytes.*
- unsigned char [bits\\_per\\_pixel](#) () const  
*Number of bits of the pixel.*
- bool [has\\_alpha](#) () const  
*Return whether the pixel has an alpha channel.*
- void [r](#) ([Color\\_component](#) const &c)  
*Set the red color component of the pixel.*
- void [g](#) ([Color\\_component](#) const &c)  
*Set the green color component of the pixel.*
- void [b](#) ([Color\\_component](#) const &c)  
*Set the blue color component of the pixel.*
- void [a](#) ([Color\\_component](#) const &c)  
*Set the alpha color component of the pixel.*
- void [bytes\\_per\\_pixel](#) (unsigned char bpp)  
*Set the size of the pixel in bytes.*
- **Pixel\_info** ()  
*Constructor.*
- [Pixel\\_info](#) (unsigned char bpp, char [r](#), char [rs](#), char [g](#), char [gs](#), char [b](#), char [bs](#), char [a](#)=0, char [as](#)=0)  
*Constructor.*
- template<typename VBI>  
[Pixel\\_info](#) (VBI const \*vbi)  
*Convenience constructor.*
- bool [operator==](#) ([Pixel\\_info](#) const &o) const  
*Compare for complete equality of the color space.*
- template<typename OUT>  
void [dump](#) (OUT &s) const  
*Dump information on the pixel to a stream.*

## 16.351.1 Detailed Description

Pixel information.

This class wraps the information on a pixel, such as the size and position of each color component in the pixel.

Definition at line 94 of file [colors](#).

## 16.351.2 Constructor & Destructor Documentation

### 16.351.2.1 Pixel\_info() [1/2]

```
L4Re::Video::Pixel_info::Pixel_info (  
    unsigned char bpp,  
    char r,  
    char rs,  
    char g,  
    char gs,  
    char b,  
    char bs,  
    char a = 0,  
    char as = 0) [inline]
```

Constructor.

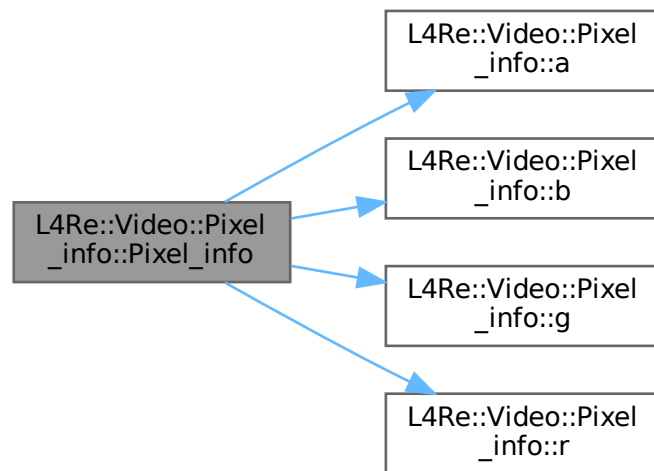
#### Parameters

<i>bpp</i>	Size of pixel in bytes.
<i>r</i>	Red component size.
<i>rs</i>	Red component shift.
<i>g</i>	Green component size.
<i>gs</i>	Green component shift.
<i>b</i>	Blue component size.
<i>bs</i>	Blue component shift.
<i>a</i>	Alpha component size, defaults to 0.
<i>as</i>	Alpha component shift, defaults to 0.

Definition at line 212 of file [colors](#).

References [a\(\)](#), [b\(\)](#), [g\(\)](#), and [r\(\)](#).

Here is the call graph for this function:



### 16.351.2.2 Pixel\_info() [2/2]

```
template<typename VBI>
L4Re::Video::Pixel_info::Pixel_info (
    VBI const * vbi) [inline], [explicit]
```

Convenience constructor.

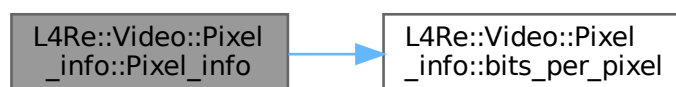
#### Parameters

<i>vbi</i>	Suitable information structure. Convenience constructor to create the pixel info from a VESA Framebuffer Info.
------------	----------------------------------------------------------------------------------------------------------------

Definition at line 224 of file [colors](#).

References [bits\\_per\\_pixel\(\)](#).

Here is the call graph for this function:



### 16.351.3 Member Function Documentation

#### 16.351.3.1 a() [1/2]

```
Color_component const & L4Re::Video::Pixel_info::a () const [inline]
```

Return the alpha color compoment of the pixel.

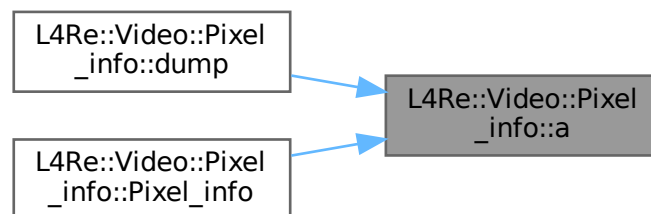
##### Returns

Alpha color component.

Definition at line 123 of file [colors](#).

Referenced by [dump\(\)](#), and [Pixel\\_info\(\)](#).

Here is the caller graph for this function:



#### 16.351.3.2 a() [2/2]

```
void L4Re::Video::Pixel_info::a (
    Color_component const & c) [inline]
```

Set the alpha color component of the pixel.

##### Parameters

<i>c</i>	Alpha color component.
----------	------------------------

Definition at line 187 of file [colors](#).



**16.351.3.3 b()** [1/2]

```
Color_component const & L4Re::Video::Pixel_info::b () const [inline]
```

Return the blue color compoment of the pixel.

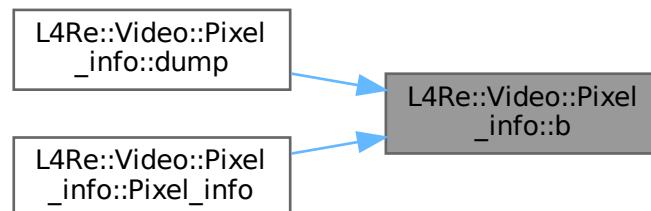
**Returns**

Blue color component.

Definition at line 117 of file [colors](#).

Referenced by [dump\(\)](#), and [Pixel\\_info\(\)](#).

Here is the caller graph for this function:

**16.351.3.4 b()** [2/2]

```
void L4Re::Video::Pixel_info::b (
    Color_component const & c) [inline]
```

Set the blue color component of the pixel.

**Parameters**

<code>c</code>	Blue color component.
----------------	-----------------------

Definition at line 181 of file [colors](#).

### 16.351.3.5 bits\_per\_pixel()

```
unsigned char L4Re::Video::Pixel_info::bits_per_pixel () const [inline]
```

Number of bits of the pixel.

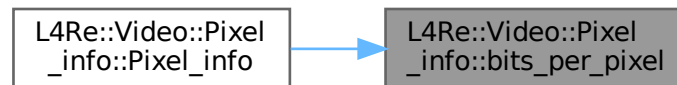
#### Returns

Number of bits used by the pixel.

Definition at line 156 of file [colors](#).

Referenced by [Pixel\\_info\(\)](#).

Here is the caller graph for this function:



### 16.351.3.6 bytes\_per\_pixel() [1/2]

```
unsigned char L4Re::Video::Pixel_info::bytes_per_pixel () const [inline]
```

Query size of pixel in bytes.

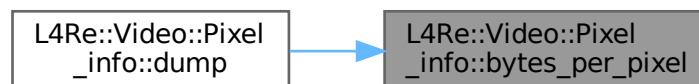
#### Returns

Size of pixel in bytes.

Definition at line 150 of file [colors](#).

Referenced by [dump\(\)](#).

Here is the caller graph for this function:



### 16.351.3.7 bytes\_per\_pixel() [2/2]

```
void L4Re::Video::Pixel_info::bytes_per_pixel (  
    unsigned char bpp) [inline]
```

Set the size of the pixel in bytes.

#### Parameters

<i>bpp</i>	Size of pixel in bytes.
------------	-------------------------

Definition at line 193 of file [colors](#).

### 16.351.3.8 dump()

```
template<typename OUT>
void L4Re::Video::Pixel_info::dump (
    OUT & s) const [inline]
```

Dump information on the pixel to a stream.

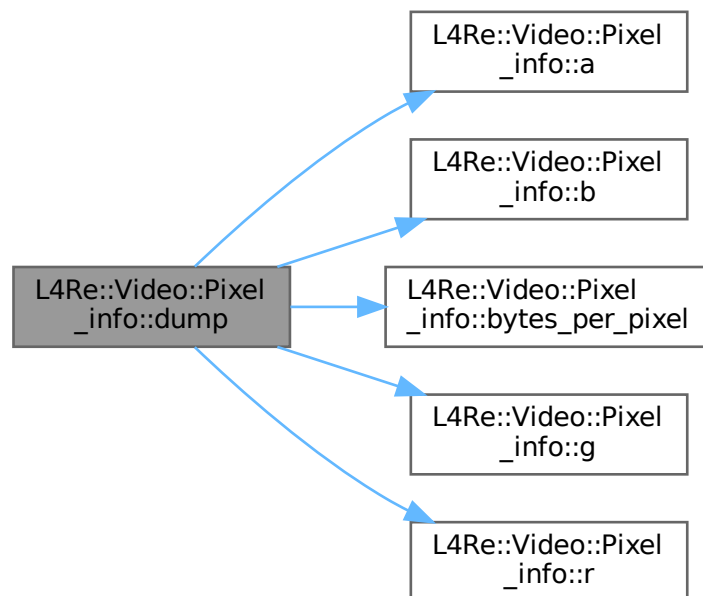
#### Parameters

<i>s</i>	Stream
----------	--------

Definition at line 246 of file [colors](#).

References [a\(\)](#), [b\(\)](#), [bytes\\_per\\_pixel\(\)](#), [g\(\)](#), and [r\(\)](#).

Here is the call graph for this function:



**16.351.3.9 g() [1/2]**

```
Color_component const & L4Re::Video::Pixel_info::g () const [inline]
```

Return the green color component of the pixel.

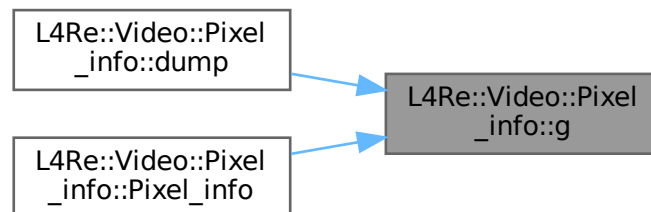
**Returns**

Green color component.

Definition at line 111 of file [colors](#).

Referenced by [dump\(\)](#), and [Pixel\\_info\(\)](#).

Here is the caller graph for this function:

**16.351.3.10 g() [2/2]**

```
void L4Re::Video::Pixel_info::g (
    Color_component const & c) [inline]
```

Set the green color component of the pixel.

**Parameters**

<code>c</code>	Green color component.
----------------	------------------------

Definition at line 175 of file [colors](#).

**16.351.3.11 has\_alpha()**

```
bool L4Re::Video::Pixel_info::has_alpha () const [inline]
```

Return whether the pixel has an alpha channel.

**Returns**

True if the pixel has an alpha channel, false if not.

Definition at line 163 of file [colors](#).

### 16.351.3.12 operator==()

```
bool L4Re::Video::Pixel_info::operator== (
    Pixel_info const & o) const [inline]
```

Compare for complete equality of the color space.

#### Parameters

<i>o</i>	A <a href="#">Pixel_info</a> to compare to.
----------	---------------------------------------------

#### Returns

true if the both [Pixel\\_info](#)'s are equal, false if not.

Definition at line 236 of file [colors](#).

References [Pixel\\_info\(\)](#).

Here is the call graph for this function:



### 16.351.3.13 padding()

```
Color_component const L4Re::Video::Pixel_info::padding () const [inline]
```

Compute the padding pseudo component.

The padding pseudo component represents the tailing bits that are reserved in RGB32 and similar pixel formats.

#### Returns

Padding pseudo component.

Definition at line 131 of file [colors](#).

**16.351.3.14** `r()` [1/2]

```
Color_component const & L4Re::Video::Pixel_info::r () const [inline]
```

Return the red color component of the pixel.

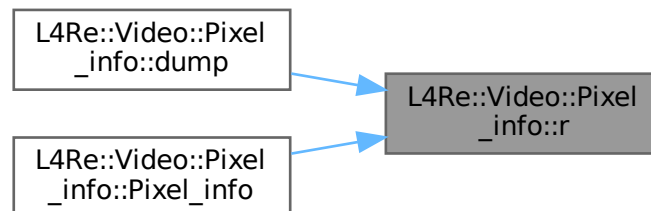
**Returns**

Red color component.

Definition at line 105 of file [colors](#).

Referenced by [dump\(\)](#), and [Pixel\\_info\(\)](#).

Here is the caller graph for this function:

**16.351.3.15** `r()` [2/2]

```
void L4Re::Video::Pixel_info::r (
    Color_component const & c) [inline]
```

Set the red color component of the pixel.

**Parameters**

<code>c</code>	Red color component.
----------------	----------------------

Definition at line 169 of file [colors](#).

The documentation for this class was generated from the following file:

- `l4/re/video/colors`

## 16.352 L4Re::Video::View Class Reference

[View](#) of a framebuffer.

```
#include <goos>
```

Collaboration diagram for L4Re::Video::View:

L4Re::Video::View
<ul style="list-style-type: none"> <li>+ info()</li> <li>+ set_info()</li> <li>+ set_viewport()</li> <li>+ stack()</li> <li>+ push_top()</li> <li>+ push_bottom()</li> <li>+ refresh()</li> <li>+ valid()</li> </ul>

### Data Structures

- struct [Info](#)  
*Information structure of a view.*

### Public Types

- enum [Flags](#) {  
[F\\_none](#) = 0x00 , [F\\_set\\_buffer](#) = 0x01 , [F\\_set\\_buffer\\_offset](#) = 0x02 , [F\\_set\\_bytes\\_per\\_line](#) = 0x04 ,  
[F\\_set\\_pixel](#) = 0x08 , [F\\_set\\_position](#) = 0x10 , [F\\_dyn\\_allocated](#) = 0x20 , [F\\_set\\_background](#) = 0x40 ,  
[F\\_set\\_flags](#) = 0x80 , [F\\_fully\\_dynamic](#) }  
*Flags on a view.*
- enum [V\\_flags](#) { [F\\_above](#) = 0x1000 , [F\\_flags\\_mask](#) = 0xff000 }  
*Property flags of a view.*

### Public Member Functions

- int [info](#) ([Info](#) \*info) const noexcept  
*Return the view information of the view.*
- int [set\\_info](#) ([Info](#) const &info) const noexcept  
*Set the information structure for this view.*
- int [set\\_viewport](#) (int scr\_x, int scr\_y, int w, int h, unsigned long buf\_offset) const noexcept  
*Set the position of the view in the [Goos](#).*

- int [stack](#) ([View](#) const &pivot, bool behind=true) const noexcept  
*Move this view in the view stack.*
- int [push\\_top](#) () const noexcept  
*Make this view the top-most view.*
- int [push\\_bottom](#) () const noexcept  
*Push this view the back.*
- int [refresh](#) (int x, int y, int w, int h) const noexcept  
*Refresh/Redraw the view.*
- bool [valid](#) () const  
*Return whether this view is valid.*

### 16.352.1 Detailed Description

[View](#) of a framebuffer.

A view is a rectangular subset of a framebuffer managed by a [Goos](#) object. The [Goos](#) orders multiple views in a stack which determines which view is on top in case they overlap. The view's pixel data is provided by a backing buffer, which must belong to the [Goos](#). It can be static or dynamically allocated, depending on the framebuffer.

See also

[L4Re::Video::Goos](#)

Definition at line 39 of file [goos](#).

### 16.352.2 Member Enumeration Documentation

#### 16.352.2.1 Flags

enum [L4Re::Video::View::Flags](#)

[Flags](#) on a view.

#### Enumerator

F_none	everything for this view is static (the VESA-FB case)
F_set_buffer	buffer object for this view can be changed
F_set_buffer_offset	buffer offset can be set
F_set_bytes_per_line	bytes per line can be set
F_set_pixel	pixel type can be set
F_set_position	position on screen can be set
F_dyn_allocated	<a href="#">View</a> is dynamically allocated.
F_set_background	Set view as background for session.



F_set_flags	Set view flags (.  See also <a href="#">V_flags</a> )
F_fully_dynamic	<a href="#">Flags</a> for a fully dynamic view.

Definition at line 59 of file [goos](#).

### 16.352.2.2 V\_flags

```
enum L4Re::Video::View::V_flags
```

Property flags of a view.

Such flags can be set or deleted with the [F\\_set\\_flags](#) operation using the [set\\_info\(\)](#) method.

#### Enumerator

F_above	Flag the view as stay on top.
F_flags_mask	Mask containing all possible property flags.

Definition at line 82 of file [goos](#).

## 16.352.3 Member Function Documentation

### 16.352.3.1 info()

```
int L4Re::Video::View::info (
    Info * info) const [inline], [noexcept]
```

Return the view information of the view.

#### Parameters

out	info	Information structure pointer.
-----	------	--------------------------------

#### Return values

0	Success
<0	Error

Definition at line 367 of file [goos](#).

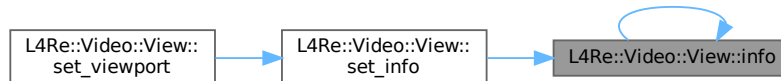
References [info\(\)](#).

Referenced by [info\(\)](#), and [set\\_info\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.352.3.2 refresh()

```

int L4Re::Video::View::refresh (
    int x,
    int y,
    int w,
    int h) const [inline], [noexcept]
  
```

Refresh/Redraw the view.

#### Parameters

<i>x</i>	X position.
<i>y</i>	Y position.
<i>w</i>	Width.
<i>h</i>	Height.

#### Return values

0	Success
<0	Error

Definition at line [379](#) of file [goos](#).

### 16.352.3.3 set\_info()

```
int L4Re::Video::View::set_info (
    Info const & info) const [inline], [noexcept]
```

Set the information structure for this view.

#### Parameters

<i>info</i>	Information structure.
-------------	------------------------

#### Return values

0	Success
<0	Error

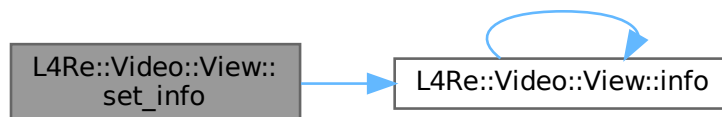
The function will also set the view port according to the values given in the information structure.

Definition at line 371 of file [goos](#).

References [info\(\)](#).

Referenced by [set\\_viewport\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.352.3.4 set\_viewport()

```
int L4Re::Video::View::set_viewport (
    int scr_x,
    int scr_y,
    int w,
    int h,
    unsigned long buf_offset) const [inline], [noexcept]
```

Set the position of the view in the [Goos](#).

#### Parameters

<i>scr_x</i>	X position
<i>scr_y</i>	Y position
<i>w</i>	Width
<i>h</i>	Height
<i>buf_offset</i>	Offset in the buffer in bytes

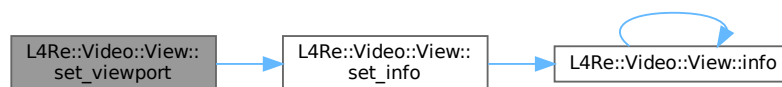
#### Return values

0	Success
<0	Error

Definition at line 383 of file [goos](#).

References [L4Re::Video::View::Info::buffer\\_index](#), [L4Re::Video::View::Info::buffer\\_offset](#), [L4Re::Video::View::Info::bytes\\_per\\_line](#), [F\\_set\\_buffer\\_offset](#), [F\\_set\\_position](#), [L4Re::Video::View::Info::flags](#), [L4Re::Video::View::Info::height](#), [L4Re::Video::View::Info::pixel\\_info](#), [set\\_info\(\)](#), [L4Re::Video::View::Info::view\\_index](#), [L4Re::Video::View::Info::width](#), [L4Re::Video::View::Info::xpos](#), and [L4Re::Video::View::Info::ypos](#).

Here is the call graph for this function:



### 16.352.3.5 stack()

```
int L4Re::Video::View::stack (
    View const & pivot,
    bool behind = true) const [inline], [noexcept]
```

Move this view in the view stack.

#### Parameters

<i>pivot</i>	<a href="#">View</a> to move relative to
<i>behind</i>	When true move the view behind the pivot view, if false move the view before the pivot view.

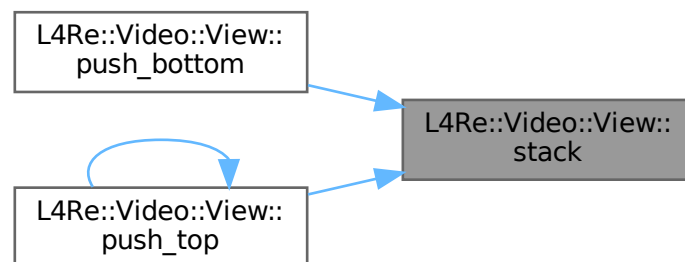
#### Return values

0	Success
<0	Error

Definition at line 375 of file [goos](#).

Referenced by [push\\_bottom\(\)](#), and [push\\_top\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

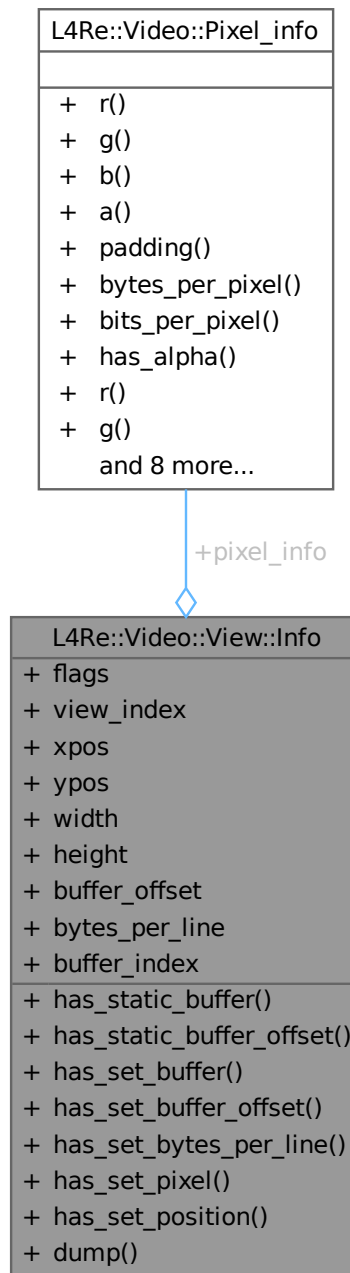
- `l4/re/video/goos`

## 16.353 L4Re::Video::View::Info Struct Reference

Information structure of a view.

```
#include <goos>
```

Collaboration diagram for L4Re::Video::View::Info:



## Public Member Functions

- bool **has\_static\_buffer** () const  
*Return whether the view has a static buffer.*
- bool **has\_static\_buffer\_offset** () const  
*Return whether the static buffer offset is available.*
- bool **has\_set\_buffer** () const

*Return whether a buffer is set.*

- bool **has\_set\_buffer\_offset** () const

*Return whether the given buffer offset is valid.*

- bool **has\_set\_bytes\_per\_line** () const

*Return whether the given bytes-per-line value is valid.*

- bool **has\_set\_pixel** () const

*Return whether the given pixel information is valid.*

- bool **has\_set\_position** () const

*Return whether the position information given is valid.*

- template<typename OUT>

void **dump** (OUT &s) const

*Dump information on the view information to a stream.*

## Data Fields

- unsigned **flags** = 0

*Flags, see [Flags](#) and [V\\_flags](#).*

- unsigned **view\_index** = 0

*Index of the view.*

- unsigned long **xpos** = 0

*X position in pixels of the view in the [Goos](#).*

- unsigned long **ypos** = 0

*Y position in pixels of the view in the [Goos](#).*

- unsigned long **width** = 0

*Width of the view in pixels.*

- unsigned long **height** = 0

*Height of the view in pixels.*

- unsigned long **buffer\_offset** = 0

*Offset in the memory buffer in bytes.*

- unsigned long **bytes\_per\_line** = 0

*Bytes per line.*

- [Pixel\\_info](#) **pixel\_info**

*Pixel information.*

- unsigned **buffer\_index** = 0

*Number of the buffer used for this view.*

## 16.353.1 Detailed Description

Information structure of a view.

Definition at line 91 of file [goos](#).

The documentation for this struct was generated from the following file:

- l4/re/video/goos

## 16.354 l4re\_aux\_t Struct Reference

Auxiliary descriptor.

```
#include <l4aux.h>
```

Collaboration diagram for l4re\_aux\_t:

l4re_aux_t
+ binary
+ kip_ds
+ dbg_lvl
+ ldr_flags
+ ldr_base

### Data Fields

- char const \* **binary**  
*Binary name.*
- [l4\\_cap\\_idx\\_t](#) **kip\_ds**  
*Data space of the KIP.*
- [l4\\_umword\\_t](#) **dbg\_lvl**  
*Debug levels for l4re.*
- [l4\\_umword\\_t](#) **ldr\_flags**  
*Flags for l4re, see [l4re\\_aux\\_ldr\\_flags\\_t](#).*
- [l4\\_addr\\_t](#) **ldr\_base**  
*Load offset of executable.*

### 16.354.1 Detailed Description

Auxiliary descriptor.

Definition at line 40 of file [l4aux.h](#).

The documentation for this struct was generated from the following file:

- [l4/re/l4aux.h](#)



## 16.355 l4re\_ds\_stats\_t Struct Reference

Information about the data space.

```
#include <dataspace.h>
```

Collaboration diagram for l4re\_ds\_stats\_t:

l4re_ds_stats_t	
+	size
+	flags

### Data Fields

- l4re\_ds\_size\_t **size**  
*size*
- l4re\_ds\_flags\_t **flags**  
*flags*

### 16.355.1 Detailed Description

Information about the data space.

Definition at line 39 of file [dataspace.h](#).

The documentation for this struct was generated from the following file:

- [l4re/c/dataspace.h](#)

## 16.356 l4re\_elf\_aux\_mword\_t Struct Reference

Auxiliary vector element for a single unsigned data word.

```
#include <elf_aux.h>
```

Collaboration diagram for l4re\_elf\_aux\_mword\_t:

l4re_elf_aux_mword_t	

### 16.356.1 Detailed Description

Auxiliary vector element for a single unsigned data word.

Definition at line 118 of file [elf\\_aux.h](#).

The documentation for this struct was generated from the following file:

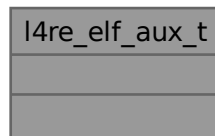
- [l4/re/elf\\_aux.h](#)

## 16.357 l4re\_elf\_aux\_t Struct Reference

Generic header for each auxiliary vector element.

```
#include <elf_aux.h>
```

Collaboration diagram for l4re\_elf\_aux\_t:



### 16.357.1 Detailed Description

Generic header for each auxiliary vector element.

Definition at line 98 of file [elf\\_aux.h](#).

The documentation for this struct was generated from the following file:

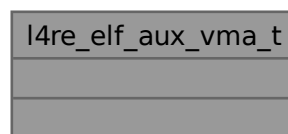
- [l4/re/elf\\_aux.h](#)

## 16.358 l4re\_elf\_aux\_vma\_t Struct Reference

Auxiliary vector element for a reserved virtual memory area.

```
#include <elf_aux.h>
```

Collaboration diagram for l4re\_elf\_aux\_vma\_t:



### 16.358.1 Detailed Description

Auxiliary vector element for a reserved virtual memory area.

Definition at line 107 of file [elf\\_aux.h](#).

The documentation for this struct was generated from the following file:

- [l4re/elf\\_aux.h](#)

## 16.359 l4re\_env\_cap\_entry\_t Struct Reference

Entry in the [L4Re](#) environment array for the named initial objects.

```
#include <env.h>
```

Collaboration diagram for l4re\_env\_cap\_entry\_t:

l4re_env_cap_entry_t
+ cap
+ flags
+ name
+ l4re_env_cap_entry_t()
+ l4re_env_cap_entry_t()

### Public Member Functions

- [l4re\\_env\\_cap\\_entry\\_t\(\)](#) [L4\\_NOTHROW](#)  
*Create an invalid entry.*
- [l4re\\_env\\_cap\\_entry\\_t](#) (char const \*n, [l4\\_cap\\_idx\\_t](#) c, [l4\\_umword\\_t](#) f=0) [L4\\_NOTHROW](#)  
*Create an entry with the name n, capability c, and flags f.*

### Data Fields

- [l4\\_cap\\_idx\\_t](#) cap  
*The capability selector for the object.*
- [l4\\_umword\\_t](#) flags  
*Flags for the object.*
- char **name** [16]  
*The name of the object.*

### 16.359.1 Detailed Description

Entry in the [L4Re](#) environment array for the named initial objects.

Definition at line [39](#) of file [env.h](#).

### 16.359.2 Constructor & Destructor Documentation

#### 16.359.2.1 `l4re_env_cap_entry_t()`

```
l4re_env_cap_entry_t::l4re_env_cap_entry_t (
    char const * n,
    l4_cap_idx_t c,
    l4_umword_t f = 0) [inline]
```

Create an entry with the name *n*, capability *c*, and flags *f*.

#### Parameters

<i>n</i>	is the name of the initial object.
<i>c</i>	is the capability selector that refers the initial object.
<i>f</i>	are the additional flags for the object.

Definition at line [70](#) of file [env.h](#).

References [cap](#), [flags](#), [L4\\_NOTHROW](#), and [name](#).

### 16.359.3 Field Documentation

#### 16.359.3.1 `flags`

```
l4_umword_t l4re_env_cap_entry_t::flags
```

Flags for the object.

#### Note

Currently unused, except as an end marker for the entry list.

Definition at line [50](#) of file [env.h](#).

Referenced by [l4re\\_env\\_cap\\_entry\\_t\(\)](#), [l4re\\_env\\_cap\\_entry\\_t\(\)](#), and [l4re\\_env\\_get\\_cap\\_l\(\)](#).

The documentation for this struct was generated from the following file:

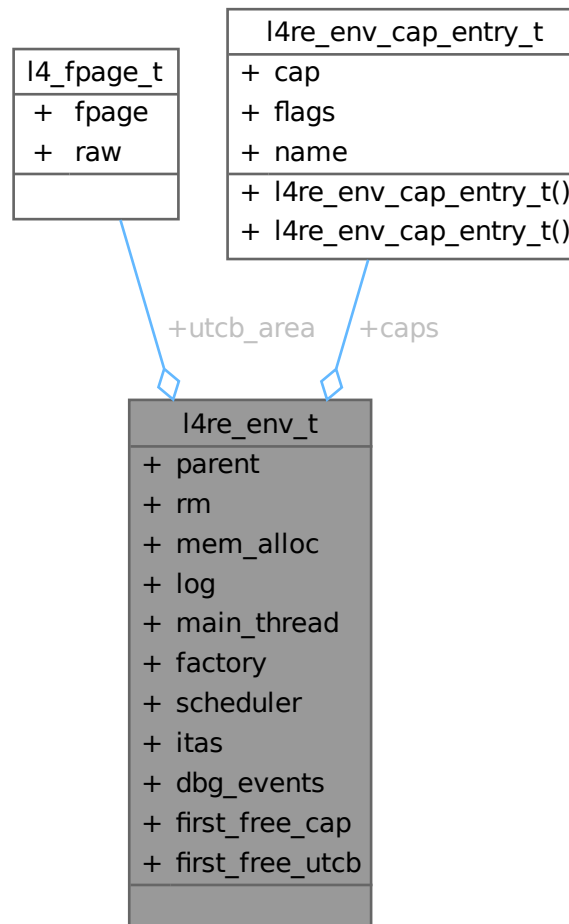
- [l4/re/env.h](#)

## 16.360 l4re\_env\_t Struct Reference

Initial environment data structure.

```
#include <env.h>
```

Collaboration diagram for l4re\_env\_t:



### Data Fields

- [l4\\_cap\\_idx\\_t](#) **parent**  
*Parent object-capability.*
- [l4\\_cap\\_idx\\_t](#) **rm**  
*Region map object-capability.*
- [l4\\_cap\\_idx\\_t](#) **mem\_alloc**  
*Memory allocator object-capability.*
- [l4\\_cap\\_idx\\_t](#) **log**  
*Logging object-capability.*

- [l4\\_cap\\_idx\\_t](#) **main\_thread**  
*Object-capability of the first user thread.*
- [l4\\_cap\\_idx\\_t](#) **factory**  
*Object-capability of the factory available to the task.*
- [l4\\_cap\\_idx\\_t](#) **scheduler**  
*Object capability for the scheduler set to use.*
- [l4\\_cap\\_idx\\_t](#) **itas**  
*ITAS services object-capability.*
- [l4\\_cap\\_idx\\_t](#) **dbg\_events**  
*Object-capability of the debug events service.*
- [l4\\_cap\\_idx\\_t](#) **first\_free\_cap**  
*First capability index available to the application.*
- [l4\\_fpage\\_t](#) **utcb\_area**  
*UTCB area of the task.*
- [l4\\_addr\\_t](#) **first\_free\_utcb**  
*First UTCB within the UTCB area available to the application.*
- [l4re\\_env\\_cap\\_entry\\_t](#) \* **caps**  
*Pointer to the first entry in the initial objects array which contains [l4re\\_env\\_cap\\_entry\\_t](#) elements.*

### 16.360.1 Detailed Description

Initial environment data structure.

See also

[Initial environment](#)

Definition at line 98 of file [env.h](#).

### 16.360.2 Field Documentation

#### 16.360.2.1 caps

```
l4re\_env\_cap\_entry\_t* l4re\_env\_t::caps
```

Pointer to the first entry in the initial objects array which contains [l4re\\_env\\_cap\\_entry\\_t](#) elements.

The array is terminated by an invalid entry with a `flags` value of `~0ul`.

Definition at line 117 of file [env.h](#).

The documentation for this struct was generated from the following file:

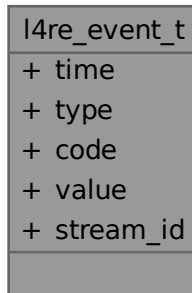
- [l4/re/env.h](#)

## 16.361 l4re\_event\_t Struct Reference

Event structure used in buffer.

```
#include <event.h>
```

Collaboration diagram for l4re\_event\_t:



### Data Fields

- long long **time**  
*Time stamp of the event.*
- unsigned short **type**  
*Type of the event.*
- unsigned short **code**  
*Code of the event.*
- int **value**  
*Value of the event.*
- [l4\\_umword\\_t](#) **stream\_id**  
*Stream ID.*

### 16.361.1 Detailed Description

Event structure used in buffer.

Definition at line 30 of file [event.h](#).

The documentation for this struct was generated from the following file:

- [l4/re/c/event.h](#)

## 16.362 l4re\_video\_color\_component\_t Struct Reference

Color component structure.

```
#include <colors.h>
```

Collaboration diagram for l4re\_video\_color\_component\_t:

l4re_video_color_component_t	
+	size
+	shift

### Data Fields

- unsigned char **size**  
*Size in bits.*
- unsigned char **shift**  
*offset in pixel*

### 16.362.1 Detailed Description

Color component structure.

Definition at line 20 of file [colors.h](#).

The documentation for this struct was generated from the following file:

- [l4re/c/video/colors.h](#)

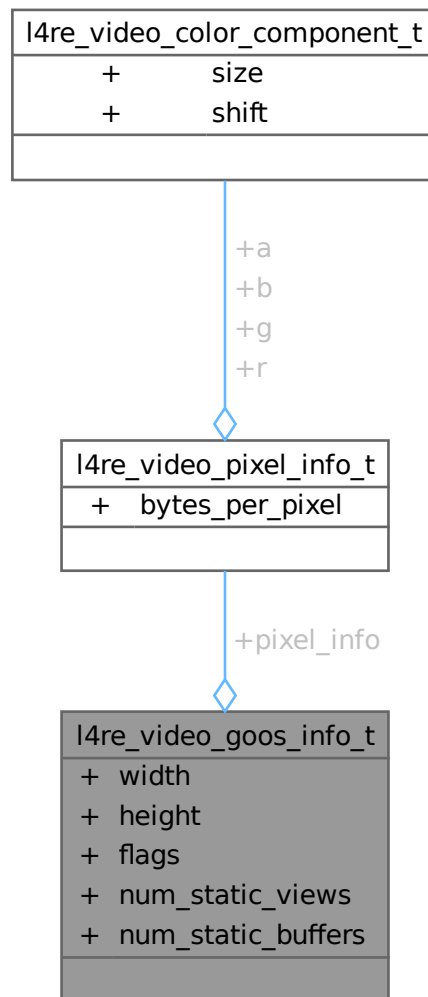
## 16.363 l4re\_video\_goos\_info\_t Struct Reference

Goos information structure.

```
#include <goos.h>
```



Collaboration diagram for l4re\_video\_goos\_info\_t:



## Data Fields

- unsigned long **width**  
*Width of the goos.*
- unsigned long **height**  
*Height of the goos.*
- unsigned **flags**  
*Flags of the framebuffer, see [l4re\\_video\\_goos\\_info\\_flags\\_t](#).*
- unsigned **num\_static\_views**  
*Number of static views.*
- unsigned **num\_static\_buffers**  
*Number of static buffers.*
- [l4re\\_video\\_pixel\\_info\\_t](#) **pixel\_info**  
*Pixel layout of the goos.*

### 16.363.1 Detailed Description

Goos information structure.

Definition at line 41 of file [goos.h](#).

The documentation for this struct was generated from the following file:

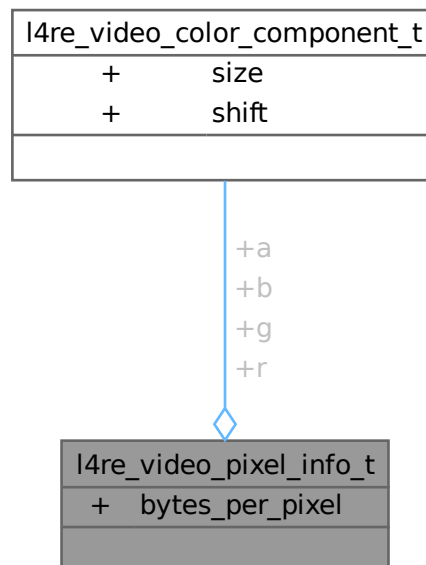
- [l4re/c/video/goos.h](#)

### 16.364 l4re\_video\_pixel\_info\_t Struct Reference

Pixel\_info structure.

```
#include <colors.h>
```

Collaboration diagram for l4re\_video\_pixel\_info\_t:



#### Data Fields

- [l4re\\_video\\_color\\_component\\_t](#) **a**  
*Colors.*
- unsigned char **bytes\_per\_pixel**  
*Bytes per pixel.*

### 16.364.1 Detailed Description

Pixel\_info structure.

Definition at line 30 of file [colors.h](#).

The documentation for this struct was generated from the following file:

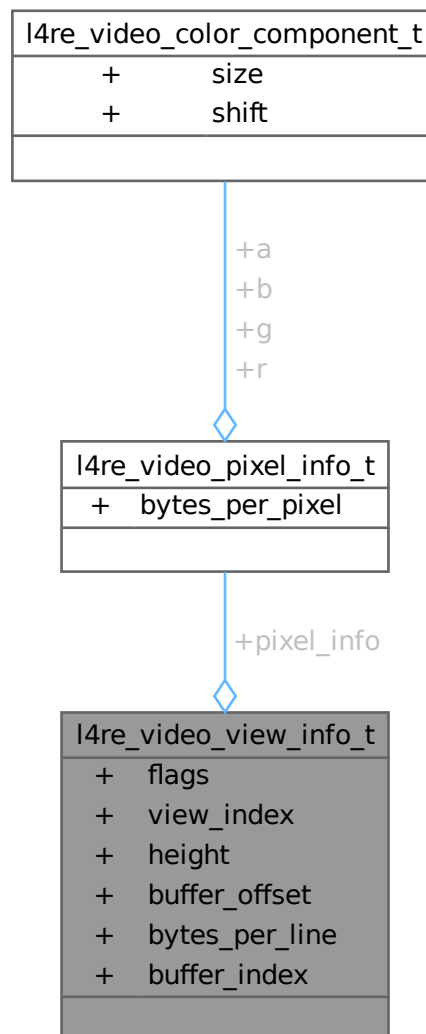
- [l4re/c/video/colors.h](#)

## 16.365 l4re\_video\_view\_info\_t Struct Reference

View information structure.

```
#include <view.h>
```

Collaboration diagram for l4re\_video\_view\_info\_t:



## Data Fields

- unsigned **flags**  
*Flags.*
- unsigned **view\_index**  
*Number of view in the goos.*
- unsigned long **height**  
*Position in goos and size of view.*
- unsigned long **buffer\_offset**  
*Memory offset in goos buffer.*
- unsigned long **bytes\_per\_line**  
*Size of line in view.*
- [l4re\\_video\\_pixel\\_info\\_t](#) **pixel\_info**  
*Pixel info.*
- unsigned **buffer\_index**  
*Number of buffer of goos.*

### 16.365.1 Detailed Description

View information structure.

Definition at line 49 of file [view.h](#).

The documentation for this struct was generated from the following file:

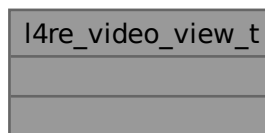
- [l4re/c/video/view.h](#)

### 16.366 l4re\_video\_view\_t Struct Reference

C representation of a goos view.

```
#include <view.h>
```

Collaboration diagram for `l4re_video_view_t`:



### 16.366.1 Detailed Description

C representation of a goos view.

A view is a visible rectangle that provides a view to the contents of a buffer (frame buffer) memory object and is placed on a real screen.

Definition at line 68 of file [view.h](#).

The documentation for this struct was generated from the following file:

- [l4/re/c/video/view.h](#)

## 16.367 l4shmc\_ringbuf\_head\_t Struct Reference

Head field of a ring buffer.

```
#include <ringbuf.h>
```

Collaboration diagram for l4shmc\_ringbuf\_head\_t:

l4shmc_ringbuf_head_t	
+	next_read
+	next_write
+	bytes_filled
+	sender_waits
+	data

### Data Fields

- unsigned **next\_read**  
*offset to next read packet*
- unsigned **next\_write**  
*offset to next write packet*
- unsigned **bytes\_filled**  
*bytes filled in buffer*
- unsigned **sender\_waits**  
*sender waiting?*
- char **data** []  
*tail pointer -> data*

### 16.367.1 Detailed Description

Head field of a ring buffer.

Definition at line 59 of file [ringbuf.h](#).

The documentation for this struct was generated from the following file:

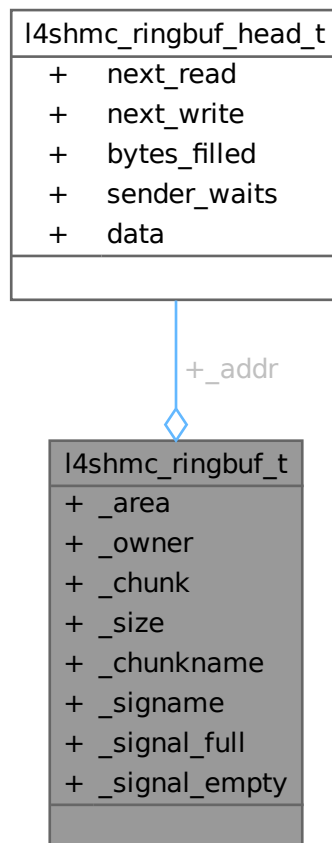
- [l4shmc/ringbuf.h](#)

## 16.368 l4shmc\_ringbuf\_t Struct Reference

Ring buffer.

```
#include <ringbuf.h>
```

Collaboration diagram for l4shmc\_ringbuf\_t:



**Data Fields**

- `l4shmc_area_t * _area`  
*L4SHM area this buffer is located in.*
- `l4_cap_idx_t _owner`  
*owner (attached to send/recv signal)*
- `l4shmc_chunk_t _chunk`  
*chunk descriptor*
- `unsigned _size`  
*chunk size // XXX do we need this?*
- `char * _chunkname`  
*name of the ring buffer chunk*
- `char * _signame`  
*base name of the ring buffer signals*
- `l4shmc_ringbuf_head_t * _addr`  
*pointer to ring buffer head*
- `l4shmc_signal_t _signal_full`  
*"full" signal - triggered when data is produced*
- `l4shmc_signal_t _signal_empty`  
*"empty" signal - triggered when data is consumed*

**16.368.1 Detailed Description**

Ring buffer.

Definition at line 85 of file [ringbuf.h](#).

The documentation for this struct was generated from the following file:

- [l4/shmc/ringbuf.h](#)

**16.369 l4util\_l4mod\_info Struct Reference**

Base module structure.

```
#include <l4mod.h>
```

Collaboration diagram for `l4util_l4mod_info`:

<code>l4util_l4mod_info</code>
+ flags
+ cmdline
+ mods_addr
+ mods_count
+ vbe_ctrl_info
+ vbe_mode_info

## Data Fields

- [l4\\_uint64\\_t](#) **flags**  
*Flags.*
- [l4\\_uint64\\_t](#) **cmdline**  
*Pointer to kernel command line.*
- [l4\\_uint64\\_t](#) **mods\_addr**  
*Module list.*
- [l4\\_uint32\\_t](#) **mods\_count**  
*Number of modules.*
- [l4\\_uint64\\_t](#) **vbe\_ctrl\_info**  
*VESA video info, valid if one of vbe\_ctrl\_info or vbe\_mode\_info is not zero.*
- [l4\\_uint64\\_t](#) **vbe\_mode\_info**  
*VESA video mode info.*

### 16.369.1 Detailed Description

Base module structure.

Definition at line 36 of file [l4mod.h](#).

### 16.369.2 Field Documentation

#### 16.369.2.1 vbe\_ctrl\_info

[l4\\_uint64\\_t](#) l4util\_l4mod\_info::vbe\_ctrl\_info

VESA video info, valid if one of vbe\_ctrl\_info or vbe\_mode\_info is not zero.

VESA video controller info

Definition at line 48 of file [l4mod.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/l4mod.h](#)

## 16.370 l4util\_l4mod\_mod Struct Reference

A single module.

```
#include <l4mod.h>
```

Collaboration diagram for l4util\_l4mod\_mod:

l4util_l4mod_mod
+ flags
+ mod_start
+ mod_end
+ cmdline



## Data Fields

- [l4\\_uint64\\_t](#) **flags**  
*Module flags (l4util\_l4mod\_mod\_info\_flag).*
- [l4\\_uint64\\_t](#) **mod\_start**  
*Starting address of module in memory.*
- [l4\\_uint64\\_t](#) **mod\_end**  
*End address of module in memory.*
- [l4\\_uint64\\_t](#) **cmdline**  
*Module command line.*

## 16.370.1 Detailed Description

A single module.

Definition at line 27 of file [l4mod.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/l4mod.h](#)

## 16.371 l4util\_mb\_addr\_range\_t Struct Reference

INT-15, AX=E820 style "AddressRangeDescriptor" ...with a "size" parameter on the front which is the structure size - 4, pointing to the next one, up until the full buffer length of the memory map has been reached.

```
#include <mb_info.h>
```

Collaboration diagram for l4util\_mb\_addr\_range\_t:

l4util_mb_addr_range_t	
+	struct_size
+	addr
+	size
+	type

## Data Fields

- [l4\\_uint32\\_t](#) **struct\_size**  
*Size of structure.*
- [l4\\_uint64\\_t](#) **addr**  
*Start address.*
- [l4\\_uint64\\_t](#) **size**  
*Size of memory range.*
- [l4\\_uint32\\_t](#) **type**  
*type of memory range*

### 16.371.1 Detailed Description

INT-15, AX=E820 style "AddressRangeDescriptor" ...with a "size" parameter on the front which is the structure size - 4, pointing to the next one, up until the full buffer length of the memory map has been reached.

Definition at line 48 of file [mb\\_info.h](#).

The documentation for this struct was generated from the following file:

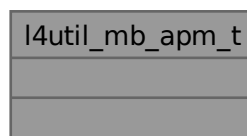
- [l4/util/mb\\_info.h](#)

## 16.372 l4util\_mb\_apm\_t Struct Reference

APM BIOS info.

```
#include <mb_info.h>
```

Collaboration diagram for l4util\_mb\_apm\_t:



### 16.372.1 Detailed Description

APM BIOS info.

Definition at line 90 of file [mb\\_info.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/mb\\_info.h](#)

## 16.373 l4util\_mb\_drive\_t Struct Reference

Drive Info structure.

```
#include <mb_info.h>
```

Collaboration diagram for l4util\_mb\_drive\_t:

l4util_mb_drive_t
+ drive_number
+ drive_mode
+ drive_cylinders
+ drive_heads
+ drive_sectors
+ drive_ports

### Data Fields

- [l4\\_uint8\\_t](#) **drive\_number**  
<The size of this structure.
- [l4\\_uint8\\_t](#) **drive\_mode**  
<The BIOS drive number.
- [l4\\_uint16\\_t](#) **drive\_cylinders**  
<The access mode (see below).
- [l4\\_uint8\\_t](#) **drive\_heads**  
<number of cylinders
- [l4\\_uint8\\_t](#) **drive\_sectors**  
<number of heads
- [l4\\_uint16\\_t](#) **drive\_ports** [0]  
<number of sectors per track

### 16.373.1 Detailed Description

Drive Info structure.

Definition at line 73 of file [mb\\_info.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/mb\\_info.h](#)

## 16.374 l4util\_mb\_info\_t Struct Reference

MultiBoot Info description.

```
#include <mb_info.h>
```

Collaboration diagram for l4util\_mb\_info\_t:

l4util_mb_info_t
+ flags
+ mem_lower
+ mem_upper
+ boot_device
+ cmdline
+ mods_count
+ mods_addr
+ tabsize
+ num
+ mmap_length
and 12 more...

### Data Fields

- [l4\\_uint32\\_t](#) **flags**  
*MultiBoot info version number.*
- [l4\\_uint32\\_t](#) **mem\_lower**  
*available memory below 1MB*
- [l4\\_uint32\\_t](#) **mem\_upper**  
*available memory starting from 1MB [KB]*
- [l4\\_uint32\\_t](#) **boot\_device**  
*"root" partition*
- [l4\\_uint32\\_t](#) **cmdline**  
*Kernel command line.*
- [l4\\_uint32\\_t](#) **mods\_count**  
*number of modules*
- [l4\\_uint32\\_t](#) **mods\_addr**  
*module list*
- [l4\\_uint32\\_t](#) **mmap\_length**  
*size of memory mapping buffer*
- [l4\\_uint32\\_t](#) **mmap\_addr**  
*address of memory mapping buffer*

- [l4\\_uint32\\_t drives\\_length](#)  
*size of drive info buffer*
- [l4\\_uint32\\_t drives\\_addr](#)  
*address of driver info buffer*
- [l4\\_uint32\\_t config\\_table](#)  
*ROM configuration table.*
- [l4\\_uint32\\_t boot\\_loader\\_name](#)  
*Boot Loader Name.*
- [l4\\_uint32\\_t apm\\_table](#)  
*APM table.*
- [l4\\_uint32\\_t vbe\\_ctrl\\_info](#)  
*VESA video controller info.*
- [l4\\_uint32\\_t vbe\\_mode\\_info](#)  
*VESA video mode info.*
- [l4\\_uint16\\_t vbe\\_mode](#)  
*VESA video mode number.*
- [l4\\_uint16\\_t vbe\\_interface\\_seg](#)  
*VESA segment of prot BIOS interface.*
- [l4\\_uint16\\_t vbe\\_interface\\_off](#)  
*VESA offset of prot BIOS interface.*
- [l4\\_uint16\\_t vbe\\_interface\\_len](#)  
*VESA lenght of prot BIOS interface.*

### 16.374.1 Detailed Description

MultiBoot Info description.

This is the struct passed to the boot image. This is done by placing its address in the EAX register.

Definition at line 247 of file [mb\\_info.h](#).

The documentation for this struct was generated from the following file:

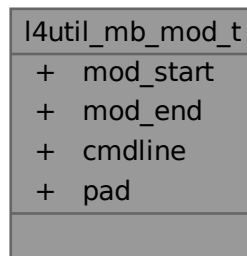
- [l4/util/mb\\_info.h](#)

## 16.375 l4util\_mb\_mod\_t Struct Reference

The structure type "mod\_list" is used by the [multiboot\\_info](#) structure.

```
#include <mb_info.h>
```

Collaboration diagram for `l4util_mb_mod_t`:



### Data Fields

- [l4\\_uint32\\_t](#) **mod\_start**  
*Starting address of module in memory.*
- [l4\\_uint32\\_t](#) **mod\_end**  
*End address of module in memory.*
- [l4\\_uint32\\_t](#) **cmdline**  
*Module command line.*
- [l4\\_uint32\\_t](#) **pad**  
*padding to take it to 16 bytes*

## 16.375.1 Detailed Description

The structure type "mod\_list" is used by the [multiboot\\_info](#) structure.

Definition at line 33 of file [mb\\_info.h](#).

The documentation for this struct was generated from the following file:

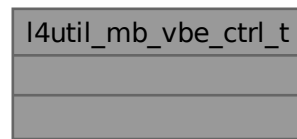
- [l4/util/mb\\_info.h](#)

## 16.376 l4util\_mb\_vbe\_ctrl\_t Struct Reference

VBE controller information.

```
#include <mb_info.h>
```

Collaboration diagram for l4util\_mb\_vbe\_ctrl\_t:



### 16.376.1 Detailed Description

VBE controller information.

Definition at line 106 of file [mb\\_info.h](#).

The documentation for this struct was generated from the following file:

- l4/util/[mb\\_info.h](#)

## 16.377 l4util\_mb\_vbe\_mode\_t Struct Reference

VBE mode information.

```
#include <mb_info.h>
```

Collaboration diagram for `l4util_mb_vbe_mode_t`:

<code>l4util_mb_vbe_mode_t</code>
+ mode_attributes
+ win_a_attributes
+ win_b_attributes
+ win_granularity
+ win_size
+ win_a_segment
+ win_b_segment
+ win_func
+ bytes_per_scanline
+ x_resolution
+ y_resolution
+ x_char_size
+ y_char_size
+ number_of_planes
+ bits_per_pixel
+ number_of_banks
+ memory_model
+ bank_size
+ number_of_image_pages
+ reserved0
+ red_mask_size
+ red_field_position
+ green_mask_size
+ green_field_position
+ blue_mask_size
+ blue_field_position
+ reserved_mask_size
+ reserved_field_position
+ direct_color_mode_info
+ phys_base
+ reserved1
+ reversed2
+ linear_bytes_per_scanline
+ banked_number_of_image_pages
+ linear_number_of_image_pages
+ linear_red_mask_size
+ linear_red_field_position
+ linear_green_mask_size
+ linear_green_field_position
+ linear_blue_mask_size
+ linear_blue_field_position
+ linear_reserved_mask_size
+ linear_reserved_field_position
+ max_pixel_clock
+ reserved3
* mode_attributes
* win_a_attributes
* win_b_attributes
* win_granularity
* win_size
* win_a_segment
* win_b_segment
* win_func
* bytes_per_scanline
* x_resolution
* y_resolution
* x_char_size
* y_char_size
* number_of_planes
* bits_per_pixel
* number_of_banks
* memory_model
* bank_size
* number_of_image_pages
* reserved0
* red_mask_size
* red_field_position
* green_mask_size
* green_field_position
* blue_mask_size
* blue_field_position
* reserved_mask_size
* reserved_field_position
* direct_color_mode_info
* phys_base
* reserved1
* reversed2
* linear_bytes_per_scanline
* banked_number_of_image_pages
* linear_number_of_image_pages
* linear_red_mask_size
* linear_red_field_position
* linear_green_mask_size
* linear_green_field_position
* linear_blue_mask_size
* linear_blue_field_position
* linear_reserved_mask_size
* linear_reserved_field_position
* max_pixel_clock
* reserved3

## Data Fields

### all VESA versions

- [l4\\_uint16\\_t mode\\_attributes](#)  
*Mode attributes.*
- [l4\\_uint8\\_t win\\_a\\_attributes](#)  
*Window A attributes.*



- [l4\\_uint8\\_t win\\_b\\_attributes](#)  
*Window B attributes.*
- [l4\\_uint16\\_t win\\_granularity](#)  
*Window granularity.*
- [l4\\_uint16\\_t win\\_size](#)  
*Window size.*
- [l4\\_uint16\\_t win\\_a\\_segment](#)  
*Window A start segment.*
- [l4\\_uint16\\_t win\\_b\\_segment](#)  
*Window B start segment.*
- [l4\\_uint32\\_t win\\_func](#)  
*Real mode pointer to window function.*
- [l4\\_uint16\\_t bytes\\_per\\_scanline](#)  
*Bytes per scan line.*

#### >= VESA version 1.2

- [l4\\_uint16\\_t x\\_resolution](#)  
*Horizontal resolution in pixels or characters.*
- [l4\\_uint16\\_t y\\_resolution](#)  
*Vertical resolution in pixels or characters.*
- [l4\\_uint8\\_t x\\_char\\_size](#)  
*Character cell width in pixels.*
- [l4\\_uint8\\_t y\\_char\\_size](#)  
*Character cell height in pixels.*
- [l4\\_uint8\\_t number\\_of\\_planes](#)  
*Number of memory planes.*
- [l4\\_uint8\\_t bits\\_per\\_pixel](#)  
*Bits per pixel.*
- [l4\\_uint8\\_t number\\_of\\_banks](#)  
*Number of banks.*
- [l4\\_uint8\\_t memory\\_model](#)  
*Memory model type.*
- [l4\\_uint8\\_t bank\\_size](#)  
*Bank size in KiB.*
- [l4\\_uint8\\_t number\\_of\\_image\\_pages](#)  
*Number of images.*
- [l4\\_uint8\\_t reserved0](#)  
*Reserved for page function.*

#### direct color

- [l4\\_uint8\\_t red\\_mask\\_size](#)  
*Size of direct color red mask in bits.*
- [l4\\_uint8\\_t red\\_field\\_position](#)  
*Bit position of LSB of red mask.*
- [l4\\_uint8\\_t green\\_mask\\_size](#)  
*Size of direct color green mask in bits.*
- [l4\\_uint8\\_t green\\_field\\_position](#)  
*Bit position of LSB of green mask.*
- [l4\\_uint8\\_t blue\\_mask\\_size](#)  
*Size of direct color blue mask in bits.*
- [l4\\_uint8\\_t blue\\_field\\_position](#)  
*Bit position of LSB of blue mask.*
- [l4\\_uint8\\_t reserved\\_mask\\_size](#)  
*Size of direct color reserved mask in bits.*
- [l4\\_uint8\\_t reserved\\_field\\_position](#)  
*Bit position of LSB of reserved mask.*

- [l4\\_uint8\\_t direct\\_color\\_mode\\_info](#)

*Direct color mode attributes.*

#### >= VESA version 2.0

- [l4\\_uint32\\_t phys\\_base](#)  
*Physical address for flat memory memory frame buffer.*
- [l4\\_uint32\\_t reserved1](#)  
*Reserved – always set to 0.*
- [l4\\_uint16\\_t reversed2](#)  
*Reserved – always set to 0.*

#### >= VESA version 3.0

- [l4\\_uint16\\_t linear\\_bytes\\_per\\_scanline](#)  
*Bytes per scan line for linear modes.*
- [l4\\_uint8\\_t banked\\_number\\_of\\_image\\_pages](#)  
*Number of images for banked modes.*
- [l4\\_uint8\\_t linear\\_number\\_of\\_image\\_pages](#)  
*Number of images for linear modes.*
- [l4\\_uint8\\_t linear\\_red\\_mask\\_size](#)  
*Size of direct color red mask (linear modes).*
- [l4\\_uint8\\_t linear\\_red\\_field\\_position](#)  
*Bit position of LSB of red mask (linear modes).*
- [l4\\_uint8\\_t linear\\_green\\_mask\\_size](#)  
*Size of direct color green mask (linear modes).*
- [l4\\_uint8\\_t linear\\_green\\_field\\_position](#)  
*Bit position of LSB of green mask (linear modes).*
- [l4\\_uint8\\_t linear\\_blue\\_mask\\_size](#)  
*Size of direct color blue mask (linear modes).*
- [l4\\_uint8\\_t linear\\_blue\\_field\\_position](#)  
*Bit position of LSB of blue mask (linear modes).*
- [l4\\_uint8\\_t linear\\_reserved\\_mask\\_size](#)  
*Size of direct color reserved mask (linear modes).*
- [l4\\_uint8\\_t linear\\_reserved\\_field\\_position](#)  
*Bit position of LSB of reserved mask (linear modes).*
- [l4\\_uint32\\_t max\\_pixel\\_clock](#)  
*Maximum pixel clock (in Hz) for graphics mode.*
- [l4\\_uint8\\_t reserved3](#) [190]  
*Reserved (padding).*

### 16.377.1 Detailed Description

VBE mode information.

Definition at line 125 of file [mb\\_info.h](#).

The documentation for this struct was generated from the following file:

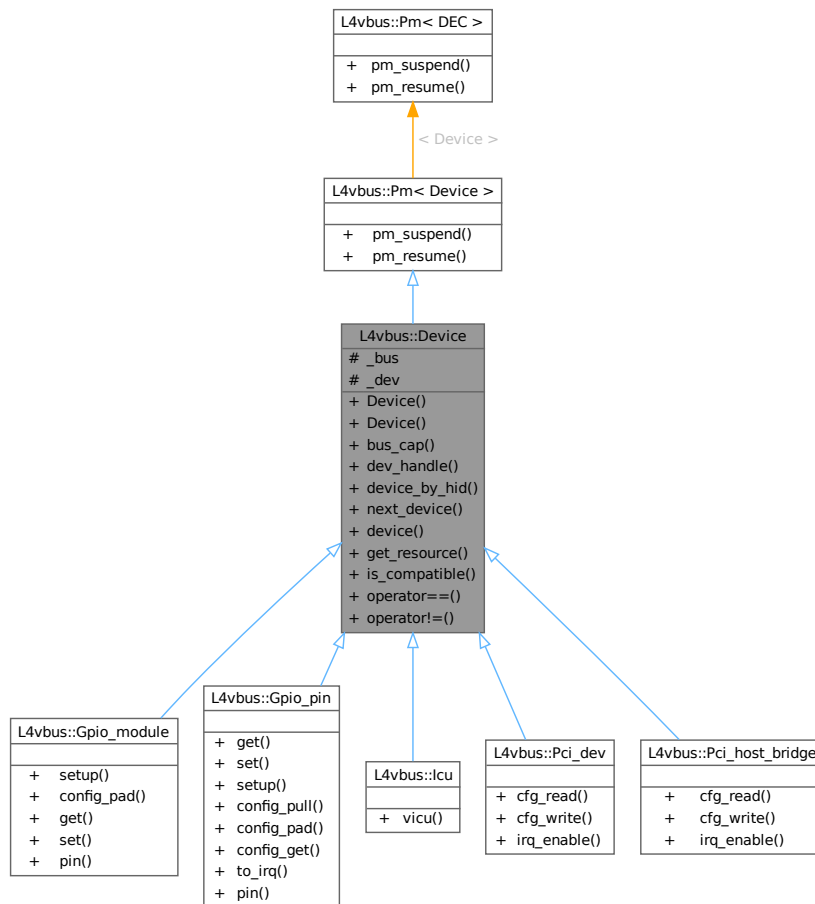
- [l4/util/mb\\_info.h](#)

## 16.378 L4vbus::Device Class Reference

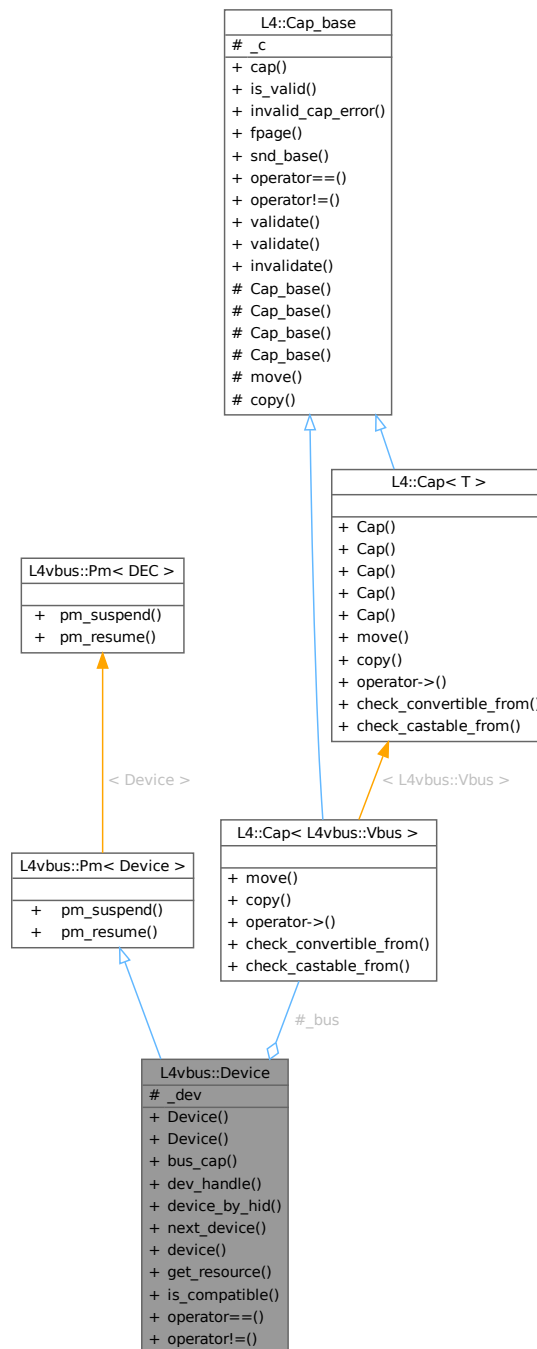
Device on a [L4vbus::Vbus](#).

```
#include <vbus>
```

Inheritance diagram for L4vbus::Device:



Collaboration diagram for L4vbus::Device:



## Public Member Functions

- **Device ( )**  
Construct a new vbus device using the NULL device `L4VBUS_NULL`.
- **Device (L4::Cap< Vbus > bus, l4vbus\_device\_handle\_t dev)**  
Construct a new vbus device using a device handle.
- **L4::Cap< Vbus > bus\_cap ( ) const**

- Access the [Vbus](#) capability of the underlying virtual bus.

  - [l4vbus\\_device\\_handle\\_t dev\\_handle](#) () const

Access the device handle of this device.
- int [device\\_by\\_hid](#) ([Device](#) \*child, char const \*hid, int depth=L4VBUS\_MAX\_DEPTH, [l4vbus\\_device\\_t](#) \*devinfo=0) const

Find a device by the hardware interface identifier (HID).
- int [next\\_device](#) ([Device](#) \*child, int depth=L4VBUS\_MAX\_DEPTH, [l4vbus\\_device\\_t](#) \*devinfo=0) const

Find next child following *child*.
- int [device](#) ([l4vbus\\_device\\_t](#) \*devinfo) const

Obtain detailed information about a [Vbus](#) device.
- int [get\\_resource](#) (unsigned res\_idx, [l4vbus\\_resource\\_t](#) \*res) const

Obtain the resource description of an individual device resource.
- int [is\\_compatible](#) (char const \*cid) const

Check if the given device has a compatibility ID (CID) or HID that matches *cid*.
- bool [operator==](#) ([Device](#) const &o) const

Test if two devices are the same [Vbus](#) device.
- bool [operator!=](#) ([Device](#) const &o) const

Test if two [Vbus](#) devices are not the same.

## Public Member Functions inherited from [L4vbus::Pm](#)< [Device](#) >

- int [pm\\_suspend](#) () const

Suspend the device.
- int [pm\\_resume](#) () const

Resume the device.

## Protected Attributes

- [L4::Cap](#)< [Vbus](#) > [\\_bus](#)

The [Vbus](#) capability where this device is located on.
- [l4vbus\\_device\\_handle\\_t](#) [\\_dev](#)

The device handle for this device.

## 16.378.1 Detailed Description

[Device](#) on a [L4vbus::Vbus](#).

Definition at line 83 of file [vbus](#).

## 16.378.2 Constructor & Destructor Documentation

### 16.378.2.1 Device()

```
L4vbus::Device::Device (
    L4::Cap< Vbus > bus,
    l4vbus\_device\_handle\_t dev) [inline]
```

Construct a new vbus device using a device handle.

Specifying the special root bus device handle [L4VBUS\\_ROOT\\_BUS](#) forms the root device of the corresponding vbus, see [Vbus::root](#).

## Parameters

<i>bus</i>	The vbus capability where this device is assigned.
<i>dev</i>	The device handle of the device.

Definition at line 100 of file [vbus](#).

References [\\_bus](#), and [\\_dev](#).

### 16.378.3 Member Function Documentation

#### 16.378.3.1 bus\_cap()

```
L4::Cap< Vbus > L4vbus::Device::bus_cap () const [inline]
```

Access the [Vbus](#) capability of the underlying virtual bus.

##### Returns

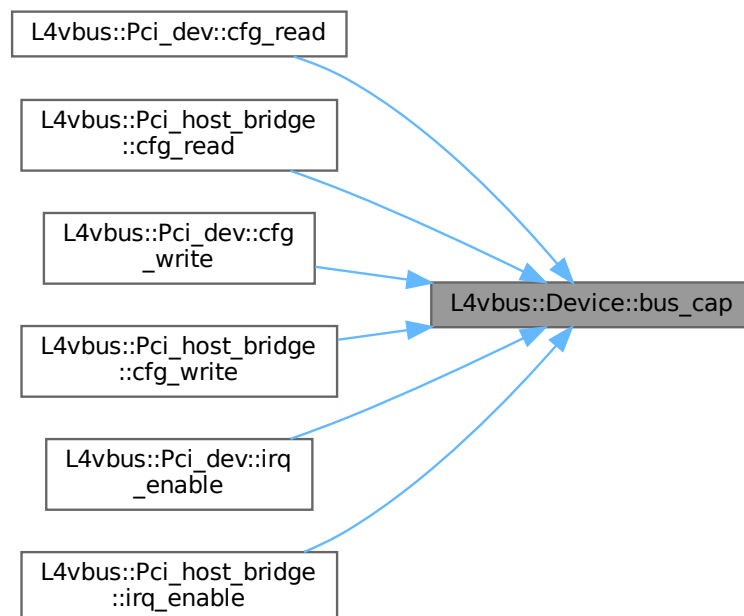
the capability to the underlying [Vbus](#).

Definition at line 107 of file [vbus](#).

References [\\_bus](#).

Referenced by [L4vbus::Pci\\_dev::cfg\\_read\(\)](#), [L4vbus::Pci\\_host\\_bridge::cfg\\_read\(\)](#), [L4vbus::Pci\\_dev::cfg\\_write\(\)](#), [L4vbus::Pci\\_host\\_bridge::cfg\\_write\(\)](#), [L4vbus::Pci\\_dev::irq\\_enable\(\)](#), and [L4vbus::Pci\\_host\\_bridge::irq\\_enable\(\)](#).

Here is the caller graph for this function:



### 16.378.3.2 dev\_handle()

```
l4vbus_device_handle_t L4vbus::Device::dev_handle () const [inline]
```

Access the device handle of this device.

#### Returns

the device handle for this device.

The device handle is used to directly address the device on its virtual bus.

Definition at line 116 of file [vbus](#).

References [\\_dev](#).

### 16.378.3.3 device()

```
int L4vbus::Device::device (  
    l4vbus_device_t * devinfo) const [inline]
```

Obtain detailed information about a [Vbus](#) device.

#### Parameters

out	<i>devinfo</i>	Information structure which contains details about the device. The pointer might be NULL.
-----	----------------	-------------------------------------------------------------------------------------------

#### Return values

0	Success.
-L4_ENODEV	No device with the given device handle <i>dev</i> could be found.

Definition at line 189 of file [vbus](#).

References [\\_bus](#), [\\_dev](#), and [l4vbus\\_get\\_device\(\)](#).

Here is the call graph for this function:



### 16.378.3.4 device\_by\_hid()

```
int L4vbus::Device::device_by_hid (
    Device * child,
    char const * hid,
    int depth = L4VBUS_MAX_DEPTH,
    l4vbus_device_t * devinfo = 0) const [inline]
```

Find a device by the hardware interface identifier (HID).

This function searches the vbus for a device with the given HID and returns a handle to the first matching device. The HID usually conforms to an ACPI HID or a Linux device tree compatible identifier.

It is possible to have multiple devices with the same HID on a vbus. In order to find all matching devices this function has to be called repeatedly with `child` pointing to the device found in the previous iteration. The iteration starts at `child` that might be any device node in the tree.

#### Parameters

in, out	<i>child</i>	Handle of the device from where in the device tree the search should start. To start searching from the beginning <code>child</code> must be initialized using the default ( <a href="#">L4VBUS_NULL</a> ). If a matching device is found, its handle is returned through this parameter.
	<i>hid</i>	HID of the device
	<i>depth</i>	Maximum depth for the recursive lookup
out	<i>devinfo</i>	<a href="#">Device</a> information structure (might be NULL)

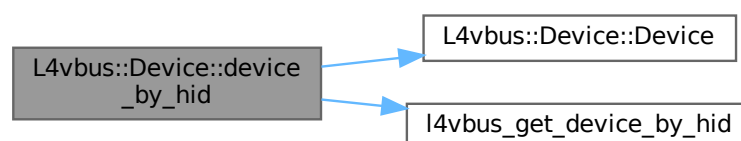
#### Return values

<code>&gt;=0</code>	A device with the given HID was found.
<code>-L4_ENOENT</code>	No device with the given HID could be found on the vbus.
<code>-L4_EINVAL</code>	Invalid or no HID provided.
<code>-L4_ENODEV</code>	Function called on a non-existing device.

Definition at line [148](#) of file [vbus](#).

References [\\_bus](#), [\\_dev](#), [Device\(\)](#), and [l4vbus\\_get\\_device\\_by\\_hid\(\)](#).

Here is the call graph for this function:





### 16.378.3.5 get\_resource()

```
int L4vbus::Device::get_resource (
    unsigned res_idx,
    l4vbus_resource_t * res) const [inline]
```

Obtain the resource description of an individual device resource.

#### Parameters

	<i>res_idx</i>	Index of the resource for which the resource description should be returned. The total number of resources for a device is available in the <a href="#">l4vbus_device_t</a> structure that is returned by <a href="#">L4vbus::Device::device_by_hid()</a> and <a href="#">L4vbus::Device::next_device()</a> .
out	<i>res</i>	Descriptor of the resource.

This function returns the resource descriptor of an individual device resource selected by the *res\_idx* parameter.

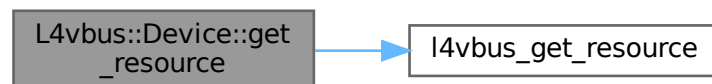
#### Return values

0	Success.
-L4_ENOENT	Invalid resource index <i>res_idx</i> .

Definition at line 209 of file [vbus](#).

References [\\_bus](#), [\\_dev](#), and [l4vbus\\_get\\_resource\(\)](#).

Here is the call graph for this function:



### 16.378.3.6 is\_compatible()

```
int L4vbus::Device::is_compatible (
    char const * cid) const [inline]
```

Check if the given device has a compatibility ID (CID) or HID that matches *cid*.

#### Parameters

<i>cid</i>	the compatibility ID to test
------------	------------------------------

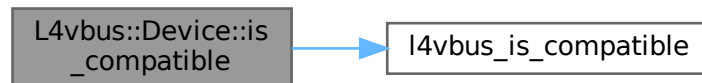
**Returns**

1 when the given ID (*cid*) matches this device, 0 when the given ID does not match, <0 on error.

Definition at line 223 of file [vbus](#).

References [\\_bus](#), [\\_dev](#), and [l4vbus\\_is\\_compatible\(\)](#).

Here is the call graph for this function:

**16.378.3.7 next\_device()**

```

int L4vbus::Device::next_device (
    Device * child,
    int depth = L4VBUS_MAX_DEPTH,
    l4vbus_device_t * devinfo = 0) const [inline]
  
```

Find next child following `child`.

**Parameters**

in, out	<i>child</i>	Handle of the device that precedes the device that shall be returned. To start from the beginning, <i>child</i> must be initialized with <a href="#">L4VBUS_NULL</a> ( <a href="#">Device::Device</a> ). If a device is found, its handle is returned through this parameter.
	<i>depth</i>	Depth to look for
out	<i>devinfo</i>	<a href="#">Device</a> information (might be NULL)

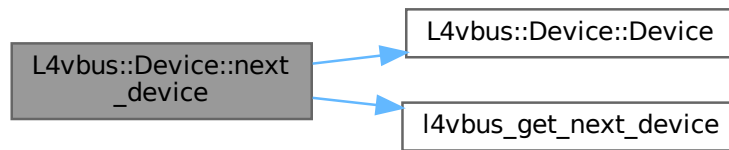
**Returns**

0 on success, else failure

Definition at line 171 of file [vbus](#).

References [\\_bus](#), [\\_dev](#), [Device\(\)](#), and [l4vbus\\_get\\_next\\_device\(\)](#).

Here is the call graph for this function:



### 16.378.3.8 `operator!=()`

```
bool L4vbus::Device::operator!= (
    Device const & o) const [inline]
```

Test if two [Vbus](#) devices are not the same.

#### Returns

true if the two devices are different, false else.

Definition at line [239](#) of file [vbus](#).

References [\\_bus](#), [\\_dev](#), and [Device\(\)](#).

Here is the call graph for this function:



### 16.378.3.9 `operator==()`

```
bool L4vbus::Device::operator== (
    Device const & o) const [inline]
```

Test if two devices are the same [Vbus](#) device.

#### Returns

true if the two devices are the same, false else.

Definition at line 230 of file [vbus](#).

References [\\_bus](#), [\\_dev](#), and [Device\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

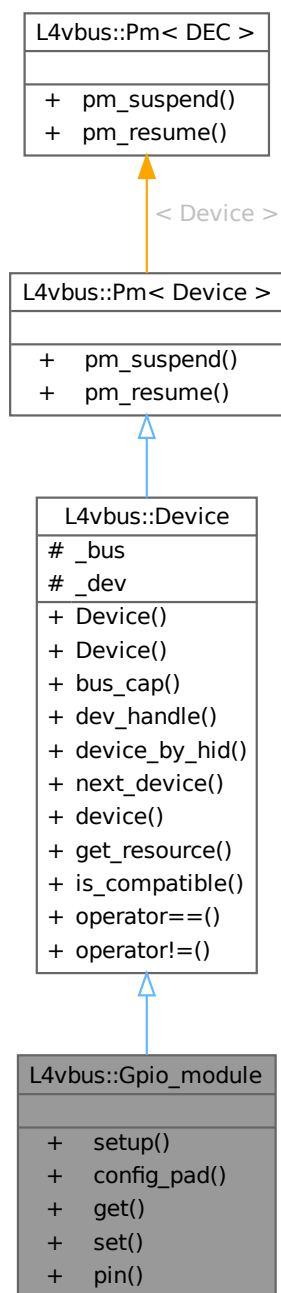
- `I4/vbus/vbus`

## 16.379 L4vbus::Gpio\_module Class Reference

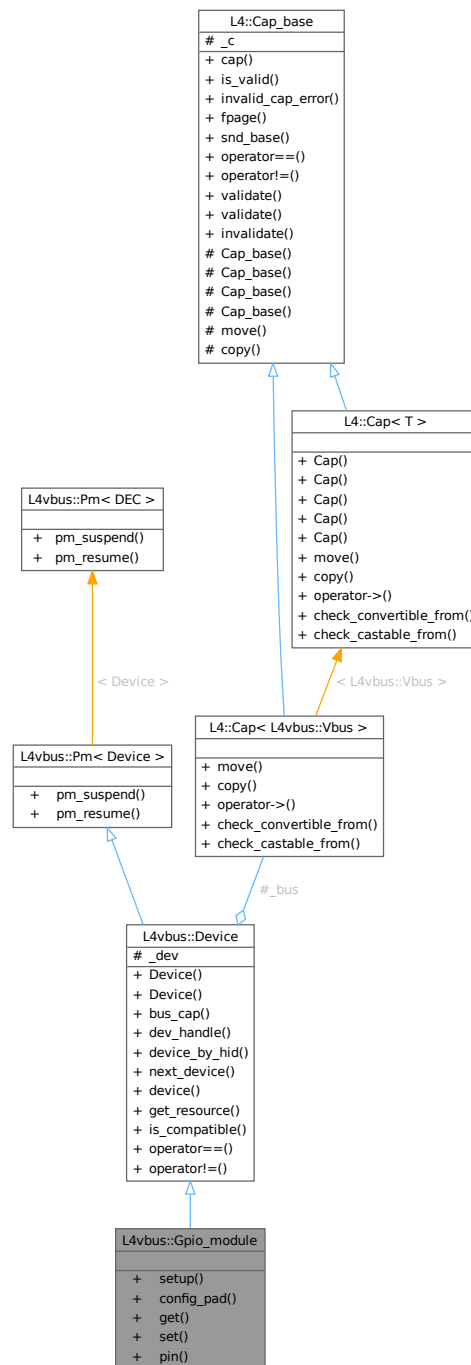
A [Gpio\\_module](#) groups multiple GPIO pins together.

```
#include <vbus_gpio>
```

Inheritance diagram for L4vbus::Gpio\_module:



Collaboration diagram for L4vbus::Gpio\_module:



## Data Structures

- struct [Pin\\_slice](#)

*A slice of the pins provided by this module.*

## Public Member Functions

- int [setup](#) ([Pin\\_slice](#) const &mask, unsigned mode, unsigned value) const

- *Configure function of multiple GPIO pins at once.*  
int [config\\_pad](#) ([Pin\\_slice](#) const &mask, unsigned func, unsigned value) const
- *Hardware specific configuration function for multiple GPIO pins.*  
int [get](#) (unsigned offset, unsigned \*data) const
- *Read values of multiple GPIO pins at once.*  
int [set](#) ([Pin\\_slice](#) const &mask, unsigned data)
- *Set multiple GPIO output pins at once.*  
[Gpio\\_pin](#) [pin](#) (unsigned pin) const
- *Get [Gpio\\_pin](#) for a specific pin of this [Gpio\\_module](#).*

## Public Member Functions inherited from [L4vbus::Device](#)

- [Device](#) ()  
*Construct a new vbus device using the NULL device [L4VBUS\\_NULL](#).*
- [Device](#) ([L4::Cap](#)< [Vbus](#) > bus, [l4vbus\\_device\\_handle\\_t](#) dev)  
*Construct a new vbus device using a device handle.*
- [L4::Cap](#)< [Vbus](#) > [bus\\_cap](#) () const  
*Access the [Vbus](#) capability of the underlying virtual bus.*
- [l4vbus\\_device\\_handle\\_t](#) [dev\\_handle](#) () const  
*Access the device handle of this device.*
- int [device\\_by\\_hid](#) ([Device](#) \*child, char const \*hid, int depth=[L4VBUS\\_MAX\\_DEPTH](#), [l4vbus\\_device\\_t](#) \*devinfo=0) const  
*Find a device by the hardware interface identifier (HID).*
- int [next\\_device](#) ([Device](#) \*child, int depth=[L4VBUS\\_MAX\\_DEPTH](#), [l4vbus\\_device\\_t](#) \*devinfo=0) const  
*Find next child following [child](#).*
- int [device](#) ([l4vbus\\_device\\_t](#) \*devinfo) const  
*Obtain detailed information about a [Vbus](#) device.*
- int [get\\_resource](#) (unsigned res\_idx, [l4vbus\\_resource\\_t](#) \*res) const  
*Obtain the resource description of an individual device resource.*
- int [is\\_compatible](#) (char const \*cid) const  
*Check if the given device has a compatibility ID (CID) or HID that matches [cid](#).*
- bool [operator==](#) ([Device](#) const &o) const  
*Test if two devices are the same [Vbus](#) device.*
- bool [operator!=](#) ([Device](#) const &o) const  
*Test if two [Vbus](#) devices are not the same.*

## Public Member Functions inherited from [L4vbus::Pm](#)< [Device](#) >

- int [pm\\_suspend](#) () const  
*Suspend the device.*
- int [pm\\_resume](#) () const  
*Resume the device.*

## Additional Inherited Members

## Protected Attributes inherited from [L4vbus::Device](#)

- [L4::Cap](#)< [Vbus](#) > [\\_bus](#)  
*The [Vbus](#) capability where this device is located on.*
- [l4vbus\\_device\\_handle\\_t](#) [\\_dev](#)  
*The device handle for this device.*

### 16.379.1 Detailed Description

A [Gpio\\_module](#) groups multiple GPIO pins together.

Definition at line 133 of file [vbus\\_gpio](#).

### 16.379.2 Member Function Documentation

#### 16.379.2.1 config\_pad()

```
int L4vbus::Gpio_module::config_pad (  
    Pin_slice const & mask,  
    unsigned func,  
    unsigned value) const [inline]
```

Hardware specific configuration function for multiple GPIO pins.

##### Parameters

<i>mask</i>	Mask of GPIO pins to configure. A bit set to 1 configures this pin. A maximum of 32 pins can be configured at once. The real number depends on the hardware and the driver implementation.
<i>func</i>	Hardware specific configuration register, usually offset to the GPIO chip's base address.
<i>value</i>	Value which is written into the hardware specific configuration register for the specified pins

##### Returns

0 if OK, error code otherwise

Definition at line 185 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_multi\\_config\\_pad\(\)](#).

Here is the call graph for this function:



#### 16.379.2.2 get()

```
int L4vbus::Gpio_module::get (  
    unsigned offset,  
    unsigned * data) const [inline]
```

Read values of multiple GPIO pins at once.

##### Parameters



	<i>offset</i>	Pin corresponding to the LSB in <i>data</i> . Note: allowed may be hardware specific.
out	<i>data</i>	Each bit returns the value (0 or 1) for the corresponding GPIO pin. The value of pins that are not accessible is undefined.

#### Returns

0 if OK, error code otherwise

Definition at line 201 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_multi\\_get\(\)](#).

Here is the call graph for this function:



#### 16.379.2.3 pin()

```
Gpio_pin L4vbus::Gpio_module::pin (  
    unsigned pin) const [inline]
```

Get [Gpio\\_pin](#) for a specific pin of this [Gpio\\_module](#).

#### Parameters

<i>pin</i>	GPIO pin number to get <a href="#">Gpio_pin</a> for.
------------	------------------------------------------------------

#### Returns

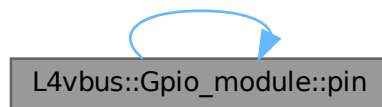
[Gpio\\_pin](#)

Definition at line 229 of file [vbus\\_gpio](#).

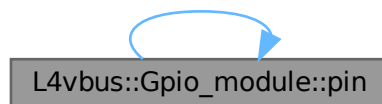
References [pin\(\)](#).

Referenced by [pin\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 16.379.2.4 set()

```
int L4vbus::Gpio_module::set (
    Pin_slice const & mask,
    unsigned data) [inline]
```

Set multiple GPIO output pins at once.

##### Parameters

<i>mask</i>	Mask of GPIO pins to set. A bit set to 1 selects this pin. A maximum of 32 pins can be set at once. The real number depends on the hardware and the driver implementation.
<i>data</i>	Each bit corresponds to the GPIO pin in <i>mask</i> . The value of each bit is written to the GPIO pin if its bit in <i>mask</i> is set.

##### Returns

0 if OK, error code otherwise

Definition at line 217 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_multi\\_set\(\)](#).

Here is the call graph for this function:



### 16.379.2.5 setup()

```
int L4vbus::Gpio_module::setup (
    Pin_slice const & mask,
    unsigned mode,
    unsigned value) const [inline]
```

Configure function of multiple GPIO pins at once.

#### Parameters

<i>mask</i>	Mask of GPIO pins to configure. A bit set to 1 configures this pin. A maximum of 32 pins can be configured at once. The real number depends on the hardware and the driver implementation.
<i>mode</i>	GPIO function, see <a href="#">L4vbus_gpio_generic_func</a> for generic functions. Hardware specific functions must be provided in the lower 8 bits.
<i>value</i>	Optional value to set the GPIO pins to if they are configured as output pins

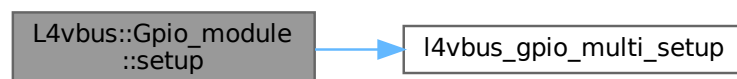
#### Returns

0 if OK, error code otherwise

Definition at line 166 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_multi\\_setup\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

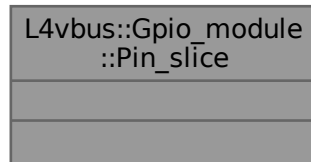
- [l4/vbus/vbus\\_gpio](#)

## 16.380 L4vbus::Gpio\_module::Pin\_slice Struct Reference

A slice of the pins provided by this module.

```
#include <vbus_gpio>
```

Collaboration diagram for L4vbus::Gpio\_module::Pin\_slice:



### 16.380.1 Detailed Description

A slice of the pins provided by this module.

Data type to specify a selection of pins for the 'multi' methods.

Definition at line [146](#) of file [vbus\\_gpio](#).

The documentation for this struct was generated from the following file:

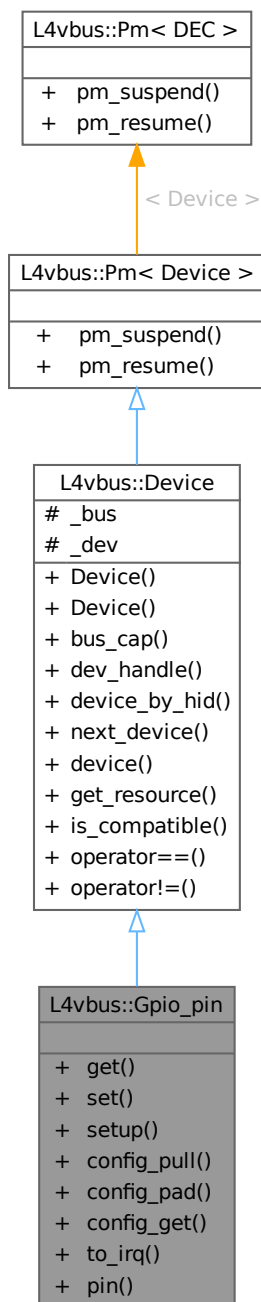
- l4/vbus/vbus\_gpio

## 16.381 L4vbus::Gpio\_pin Class Reference

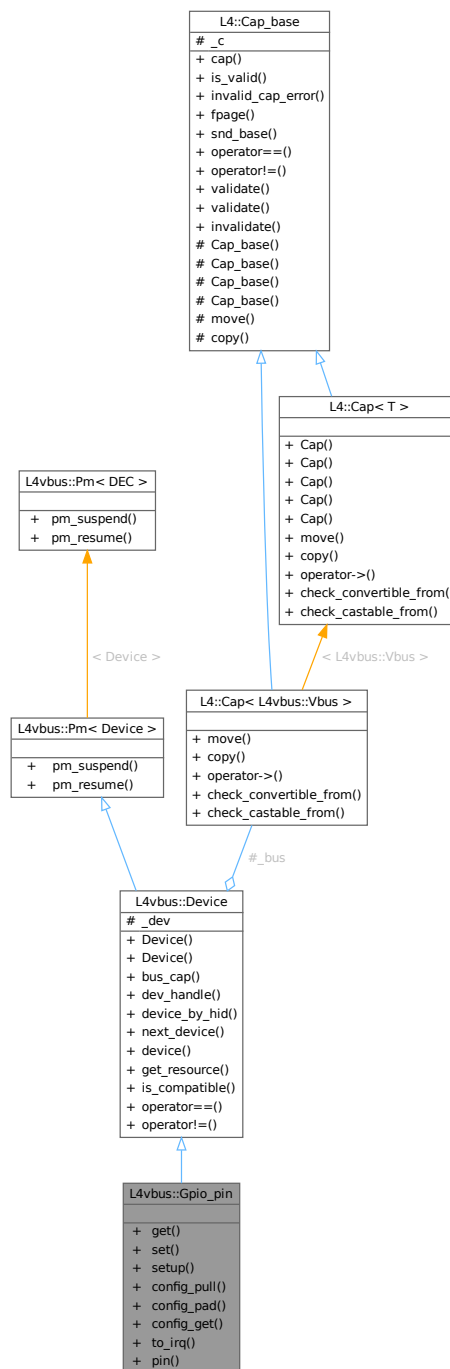
A GPIO pin.

```
#include <vbus_gpio>
```

Inheritance diagram for L4vbus::Gpio\_pin:



Collaboration diagram for L4vbus::Gpio\_pin:



## Public Member Functions

- `int` `get` ( ) const  
Read value of GPIO input pin.
- `int` `set` (int value) const  
Set GPIO output pin.
- `int` `setup` (unsigned mode, unsigned value) const

- *Configure the function of a GPIO pin.*
- `int config_pull` (unsigned mode) const  
*Generic function to set pull up/down mode.*
- `int config_pad` (unsigned func, unsigned value) const  
*Hardware specific configuration function.*
- `int config_get` (unsigned func, unsigned \*value) const  
*Read hardware specific configuration.*
- `int to_irq` () const  
*Create IRQ for GPIO pin.*
- `unsigned pin` () const  
*Get pin number.*

## Public Member Functions inherited from L4vbus::Device

- `Device` ()  
*Construct a new vbus device using the NULL device L4VBUS\_NULL.*
- `Device` (L4::Cap< Vbus > bus, l4vbus\_device\_handle\_t dev)  
*Construct a new vbus device using a device handle.*
- `L4::Cap< Vbus > bus_cap` () const  
*Access the Vbus capability of the underlying virtual bus.*
- `l4vbus_device_handle_t dev_handle` () const  
*Access the device handle of this device.*
- `int device_by_hid` (Device \*child, char const \*hid, int depth=L4VBUS\_MAX\_DEPTH, l4vbus\_device\_t \*devinfo=0) const  
*Find a device by the hardware interface identifier (HID).*
- `int next_device` (Device \*child, int depth=L4VBUS\_MAX\_DEPTH, l4vbus\_device\_t \*devinfo=0) const  
*Find next child following child.*
- `int device` (l4vbus\_device\_t \*devinfo) const  
*Obtain detailed information about a Vbus device.*
- `int get_resource` (unsigned res\_idx, l4vbus\_resource\_t \*res) const  
*Obtain the resource description of an individual device resource.*
- `int is_compatible` (char const \*cid) const  
*Check if the given device has a compatibility ID (CID) or HID that matches cid.*
- `bool operator==` (Device const &o) const  
*Test if two devices are the same Vbus device.*
- `bool operator!=` (Device const &o) const  
*Test if two Vbus devices are not the same.*

## Public Member Functions inherited from L4vbus::Pm< Device >

- `int pm_suspend` () const  
*Suspend the device.*
- `int pm_resume` () const  
*Resume the device.*

## Additional Inherited Members

### Protected Attributes inherited from [L4vbus::Device](#)

- [L4::Cap< Vbus > \\_bus](#)  
*The [Vbus](#) capability where this device is located on.*
- [l4vbus\\_device\\_handle\\_t \\_dev](#)  
*The device handle for this device.*

## 16.381.1 Detailed Description

A GPIO pin.

Definition at line 26 of file [vbus\\_gpio](#).

## 16.381.2 Member Function Documentation

### 16.381.2.1 `config_get()`

```
int L4vbus::Gpio_pin::config_get (
    unsigned func,
    unsigned * value) const [inline]
```

Read hardware specific configuration.

#### Parameters

	<i>func</i>	Hardware specific configuration register to read from. Usually this is an offset to the GPIO chip's base address.
out	<i>value</i>	The configuration value.

#### Returns

0 if OK, error code otherwise

Definition at line 102 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_config\\_get\(\)](#).

Here is the call graph for this function:





### 16.381.2.2 config\_pad()

```
int L4vbus::Gpio_pin::config_pad (  
    unsigned func,  
    unsigned value) const [inline]
```

Hardware specific configuration function.

#### Parameters

---

<i>func</i>	Hardware specific configuration register, usually offset to the GPIO chip's base address
<i>value</i>	Value which is written into the hardware specific configuration register for the specified pin

#### Returns

0 if OK, error code otherwise

Definition at line 89 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_config\\_pad\(\)](#).

Here is the call graph for this function:



#### 16.381.2.3 config\_pull()

```
int L4vbus::Gpio_pin::config_pull (  
    unsigned mode) const [inline]
```

Generic function to set pull up/down mode.

#### Parameters

<i>mode</i>	mode for pull up/down resistors, see <a href="#">L4vbus_gpio_pull_modes</a>
-------------	-----------------------------------------------------------------------------

#### Returns

0 if OK, error code otherwise

Definition at line 75 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_config\\_pull\(\)](#).

Here is the call graph for this function:



#### 16.381.2.4 get()

```
int L4vbus::Gpio_pin::get () const [inline]
```

Read value of GPIO input pin.

##### Returns

Value of GPIO pin (usually 0 or 1), negative error code otherwise.

Definition at line 38 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_get\(\)](#).

Here is the call graph for this function:



#### 16.381.2.5 pin()

```
unsigned L4vbus::Gpio_pin::pin () const [inline]
```

Get pin number.

##### Returns

GPIO pin number

Definition at line 122 of file [vbus\\_gpio](#).

#### 16.381.2.6 set()

```
int L4vbus::Gpio_pin::set (  
    int value) const [inline]
```

Set GPIO output pin.

##### Parameters

<i>value</i>	Value to write to the GPIO pin (usually 0 or 1)
--------------	-------------------------------------------------

**Returns**

0 if OK, error code otherwise

Definition at line 49 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_set\(\)](#).

Here is the call graph for this function:

**16.381.2.7 setup()**

```
int L4vbus::Gpio_pin::setup (  
    unsigned mode,  
    unsigned value) const [inline]
```

Configure the function of a GPIO pin.

**Parameters**

<i>mode</i>	GPIO function, see <a href="#">L4vbus_gpio_generic_func</a> for generic functions. Hardware specific functions must be provided in the lower 8 bits.
<i>value</i>	Optional value to set the GPIO pin to if it is configured as an output pin

**Returns**

0 if OK, error code otherwise

Definition at line 64 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_setup\(\)](#).

Here is the call graph for this function:



### 16.381.2.8 to\_irq()

```
int L4vbus::Gpio_pin::to_irq () const [inline]
```

Create IRQ for GPIO pin.

#### Returns

IRQ number if OK, negative error code otherwise

Definition at line 112 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_to\\_irq\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

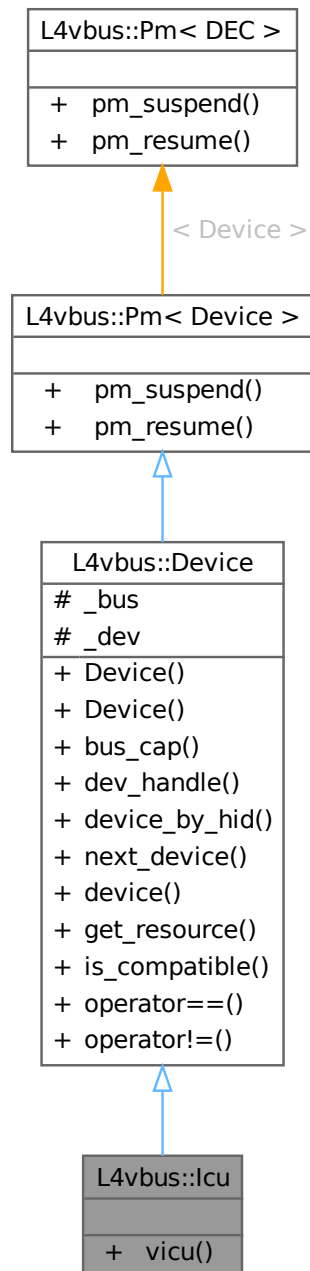
- [l4/vbus/vbus\\_gpio](#)

## 16.382 L4vbus::lcu Class Reference

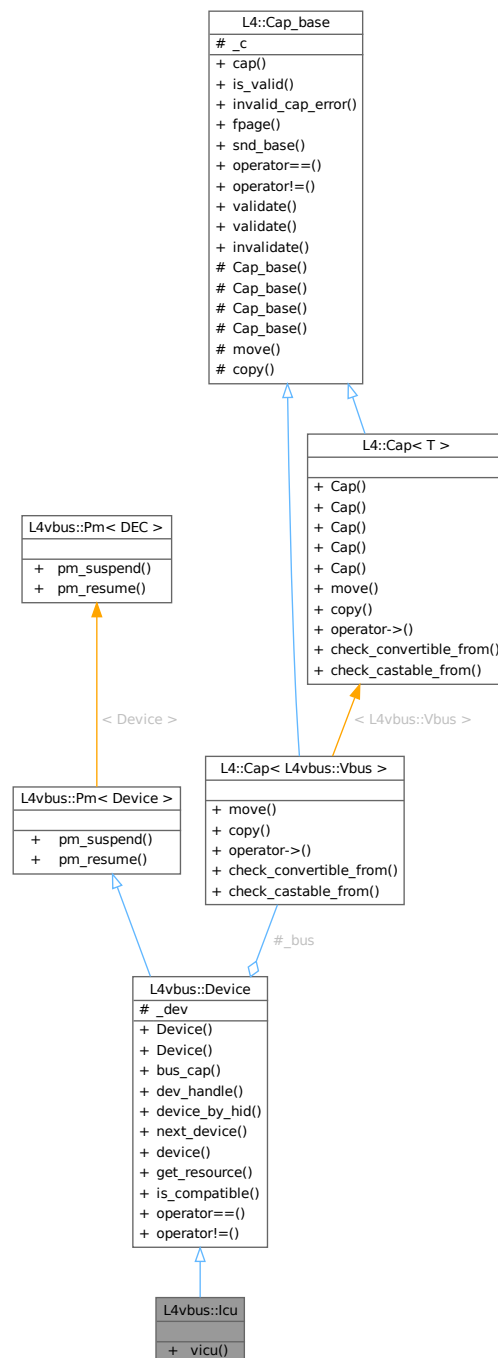
[Vbus](#) Interrupt controller API.

```
#include <vbus>
```

Inheritance diagram for L4vbus::lcu:



Collaboration diagram for L4vbus::lcu:



## Public Types

- enum `Src_types` { `Src_dev_handle` = `L4VBUS_ICU_SRC_DEV_HANDLE` }  
Flags that can be used with the ICU on a vbus device.

## Public Member Functions

- int `vicu` (`L4::Cap< L4::lcu >` icu) const

Request an [L4::Icu](#) capability for this [Vbus](#)'s virtual ICU.

## Public Member Functions inherited from [L4vbus::Device](#)

- [Device](#) ()  
Construct a new vbus device using the NULL device [L4VBUS\\_NULL](#).
- [Device](#) ([L4::Cap](#)< [Vbus](#) > bus, [l4vbus\\_device\\_handle\\_t](#) dev)  
Construct a new vbus device using a device handle.
- [L4::Cap](#)< [Vbus](#) > [bus\\_cap](#) () const  
Access the [Vbus](#) capability of the underlying virtual bus.
- [l4vbus\\_device\\_handle\\_t](#) [dev\\_handle](#) () const  
Access the device handle of this device.
- int [device\\_by\\_hid](#) ([Device](#) \*child, char const \*hid, int depth=L4VBUS\_MAX\_DEPTH, [l4vbus\\_device\\_t](#) \*devinfo=0) const  
Find a device by the hardware interface identifier (HID).
- int [next\\_device](#) ([Device](#) \*child, int depth=L4VBUS\_MAX\_DEPTH, [l4vbus\\_device\\_t](#) \*devinfo=0) const  
Find next child following *child*.
- int [device](#) ([l4vbus\\_device\\_t](#) \*devinfo) const  
Obtain detailed information about a [Vbus](#) device.
- int [get\\_resource](#) (unsigned res\_idx, [l4vbus\\_resource\\_t](#) \*res) const  
Obtain the resource description of an individual device resource.
- int [is\\_compatible](#) (char const \*cid) const  
Check if the given device has a compatibility ID (CID) or HID that matches *cid*.
- bool [operator==](#) ([Device](#) const &o) const  
Test if two devices are the same [Vbus](#) device.
- bool [operator!=](#) ([Device](#) const &o) const  
Test if two [Vbus](#) devices are not the same.

## Public Member Functions inherited from [L4vbus::Pm](#)< [Device](#) >

- int [pm\\_suspend](#) () const  
Suspend the device.
- int [pm\\_resume](#) () const  
Resume the device.

## Additional Inherited Members

## Protected Attributes inherited from [L4vbus::Device](#)

- [L4::Cap](#)< [Vbus](#) > [\\_bus](#)  
The [Vbus](#) capability where this device is located on.
- [l4vbus\\_device\\_handle\\_t](#) [\\_dev](#)  
The device handle for this device.



## 16.382.1 Detailed Description

[Vbus](#) Interrupt controller API.

Every [Vbus](#) contains a virtual interrupt control unit that manages the IRQs of the devices on the bus. This class provides access to a capability to an [L4::Icu](#) object which can then be used to interface with the IRQs.

See also

[L4::Icu](#)

Definition at line 260 of file [vbus](#).

## 16.382.2 Member Enumeration Documentation

### 16.382.2.1 Src\_types

```
enum L4vbus::Icu::Src_types
```

Flags that can be used with the ICU on a vbus device.

#### Enumerator

Src_dev_handle	Flag to denote that the value should be interpreted as a device handle. This flag may be used in the <code>source</code> parameter in <a href="#">L4::Icu::msi_info()</a> to denote that the ICU should interpret the source ID as a device handle.
----------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Definition at line 264 of file [vbus](#).

## 16.382.3 Member Function Documentation

### 16.382.3.1 vicu()

```
int L4vbus::Icu::vicu (  
    L4::Cap< L4::Icu > icu) const [inline]
```

Request an [L4::Icu](#) capability for this [Vbus](#)'s virtual ICU.

#### Parameters

<i>out</i>	<i>icu</i>	Capability slot where the <a href="#">L4::Icu</a> capability shall be stored.
------------	------------	-------------------------------------------------------------------------------

#### Return values

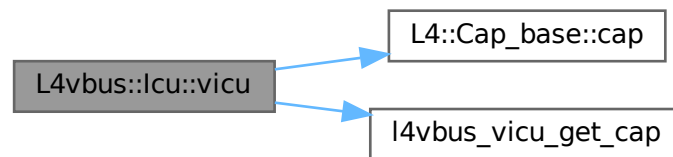
0	Success.
---	----------

<i>otherwise</i>	IPC error.
------------------	------------

Definition at line 285 of file [vbus](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), [L4::Cap\\_base::cap\(\)](#), and [l4vbus\\_vicu\\_get\\_cap\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

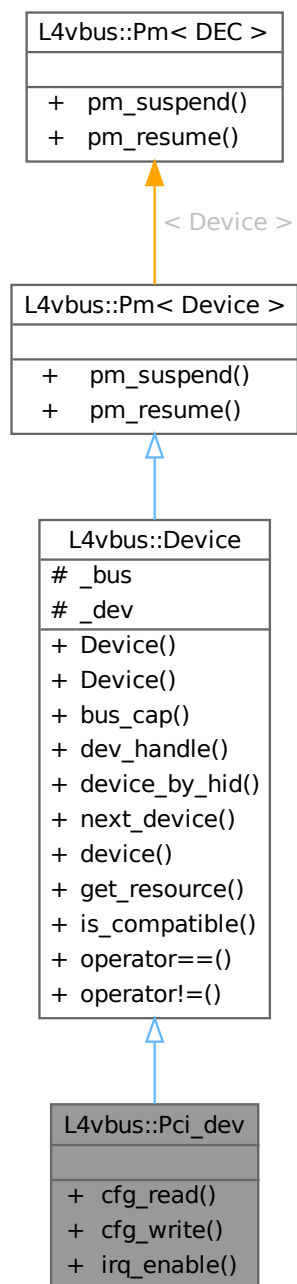
- `l4/vbus/vbus`

## 16.383 L4vbus::Pci\_dev Class Reference

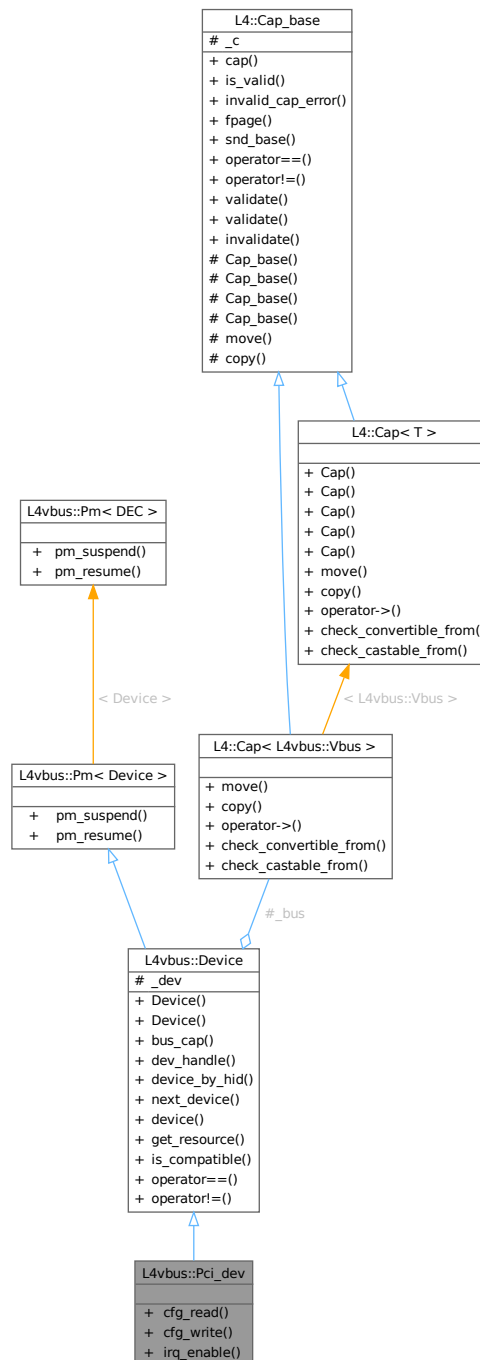
A PCI device.

```
#include <vbus_pci>
```

Inheritance diagram for L4vbus::Pci\_dev:



Collaboration diagram for L4vbus::Pci\_dev:



## Public Member Functions

- `int cfg_read (l4_uint32_t reg, l4_uint32_t *value, l4_uint32_t width) const`  
Read from the device's vPCI configuration space.
- `int cfg_write (l4_uint32_t reg, l4_uint32_t value, l4_uint32_t width) const`  
Write to the device's vPCI configuration space.
- `int irq_enable (unsigned char *trigger, unsigned char *polarity) const`  
Enable the device's PCI interrupt.

## Public Member Functions inherited from L4vbus::Device

- **Device** ()  
*Construct a new vbus device using the NULL device L4VBUS\_NULL.*
- **Device** (L4::Cap< Vbus > bus, l4vbus\_device\_handle\_t dev)  
*Construct a new vbus device using a device handle.*
- **L4::Cap< Vbus > bus\_cap** () const  
*Access the Vbus capability of the underlying virtual bus.*
- **l4vbus\_device\_handle\_t dev\_handle** () const  
*Access the device handle of this device.*
- **int device\_by\_hid** (Device \*child, char const \*hid, int depth=L4VBUS\_MAX\_DEPTH, l4vbus\_device\_t \*devinfo=0) const  
*Find a device by the hardware interface identifier (HID).*
- **int next\_device** (Device \*child, int depth=L4VBUS\_MAX\_DEPTH, l4vbus\_device\_t \*devinfo=0) const  
*Find next child following child.*
- **int device** (l4vbus\_device\_t \*devinfo) const  
*Obtain detailed information about a Vbus device.*
- **int get\_resource** (unsigned res\_idx, l4vbus\_resource\_t \*res) const  
*Obtain the resource description of an individual device resource.*
- **int is\_compatible** (char const \*cid) const  
*Check if the given device has a compatibility ID (CID) or HID that matches cid.*
- **bool operator==** (Device const &o) const  
*Test if two devices are the same Vbus device.*
- **bool operator!=** (Device const &o) const  
*Test if two Vbus devices are not the same.*

## Public Member Functions inherited from L4vbus::Pm< Device >

- **int pm\_suspend** () const  
*Suspend the device.*
- **int pm\_resume** () const  
*Resume the device.*

## Additional Inherited Members

## Protected Attributes inherited from L4vbus::Device

- **L4::Cap< Vbus > \_bus**  
*The Vbus capability where this device is located on.*
- **l4vbus\_device\_handle\_t \_dev**  
*The device handle for this device.*

### 16.383.1 Detailed Description

A PCI device.

Definition at line 93 of file `vbus_pci`.

## 16.383.2 Member Function Documentation

### 16.383.2.1 `cfg_read()`

```
int L4vbus::Pci_dev::cfg_read (
    14_uint32_t reg,
    14_uint32_t * value,
    14_uint32_t width) const [inline]
```

Read from the device's vPCI configuration space.

#### Parameters

	<i>reg</i>	Register in configuration space to read
out	<i>value</i>	Value that has been read
	<i>width</i>	Width to read in bits (e.g. 8, 16, 32)

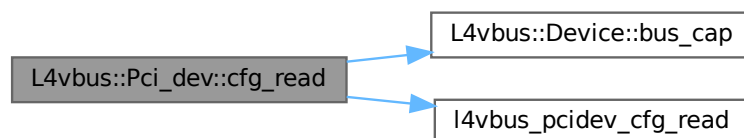
#### Returns

0 on success, else failure

Definition at line 105 of file `vbus_pci`.

References `L4vbus::Device::_dev`, `L4vbus::Device::bus_cap()`, and `l4vbus_pcidev_cfg_read()`.

Here is the call graph for this function:



### 16.383.2.2 `cfg_write()`

```
int L4vbus::Pci_dev::cfg_write (
    14_uint32_t reg,
    14_uint32_t value,
    14_uint32_t width) const [inline]
```

Write to the device's vPCI configuration space.

#### Parameters

<i>reg</i>	Register in configuration space to write
<i>value</i>	Value to write
<i>width</i>	Width to write in bits (e.g. 8, 16, 32)

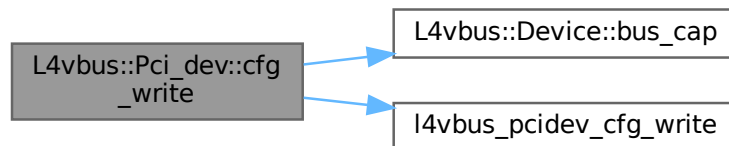
#### Returns

0 on success, else failure

Definition at line 121 of file [vbus\\_pci](#).

References [L4vbus::Device::\\_dev](#), [L4vbus::Device::bus\\_cap\(\)](#), and [l4vbus\\_pciddev\\_cfg\\_write\(\)](#).

Here is the call graph for this function:



#### 16.383.2.3 irq\_enable()

```
int L4vbus::Pci_dev::irq_enable (  
    unsigned char * trigger,  
    unsigned char * polarity) const [inline]
```

Enable the device's PCI interrupt.

#### Parameters

out	<i>trigger</i>	False if interrupt is level-triggered
out	<i>polarity</i>	True if interrupt is of low polarity

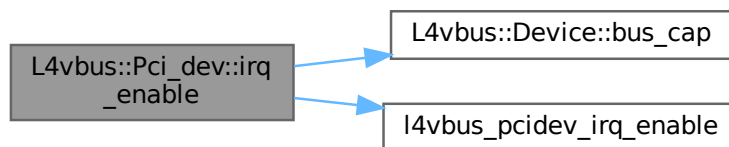
### Returns

On success: Interrupt line to be used, else failure

Definition at line 137 of file [vbus\\_pci](#).

References [L4vbus::Device::\\_dev](#), [L4vbus::Device::bus\\_cap\(\)](#), and [l4vbus\\_pcidev\\_irq\\_enable\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- `l4/vbus/vbus_pci`

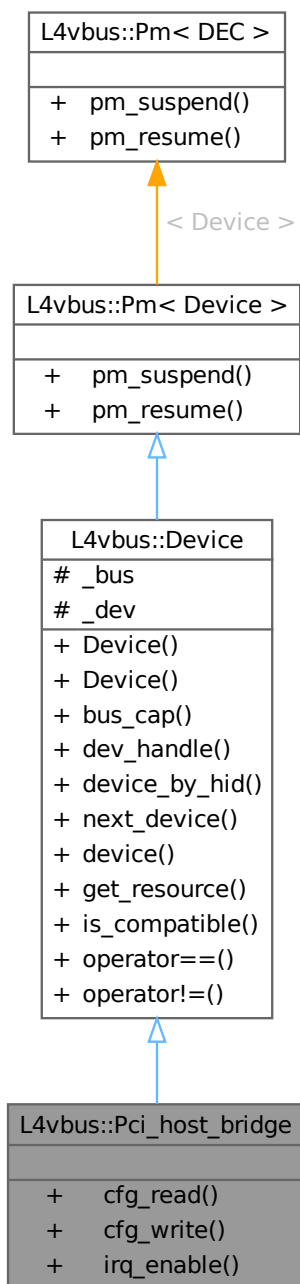
## 16.384 L4vbus::Pci\_host\_bridge Class Reference

A Pci host bridge.

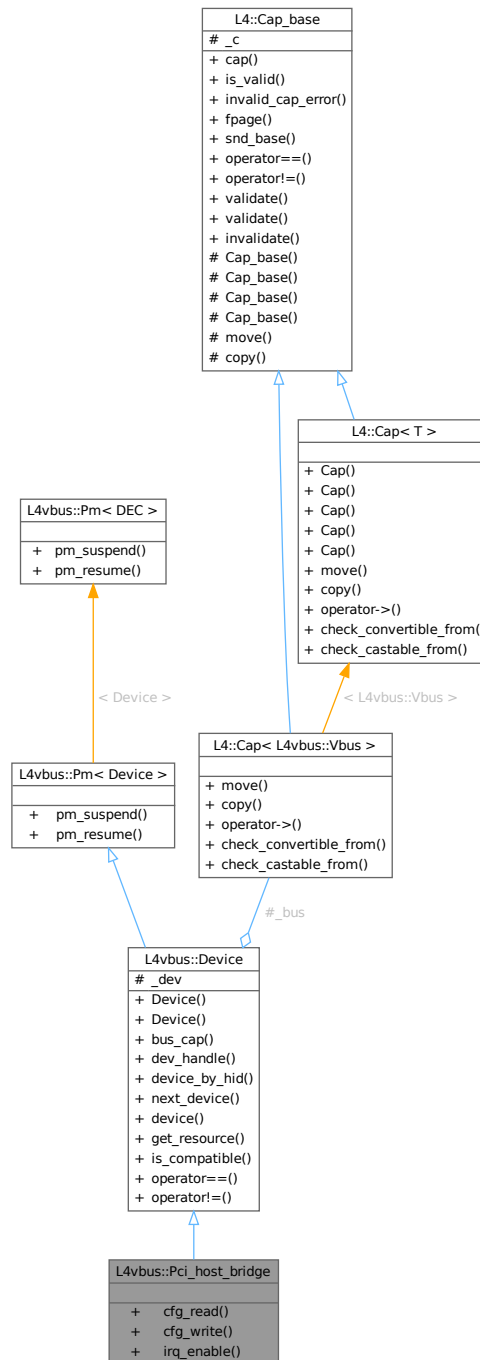
```
#include <vbus_pci>
```



Inheritance diagram for L4vbus::Pci\_host\_bridge:



Collaboration diagram for L4vbus::Pci\_host\_bridge:



## Public Member Functions

- `int cfg_read (l4_uint32_t bus, l4_uint32_t devfn, l4_uint32_t reg, l4_uint32_t *value, l4_uint32_t width) const`  
Read from the vPCI configuration space using the PCI root bridge.
- `int cfg_write (l4_uint32_t bus, l4_uint32_t devfn, l4_uint32_t reg, l4_uint32_t value, l4_uint32_t width) const`  
Write to the vPCI configuration space using the PCI root bridge.

- int [irq\\_enable](#) ([l4\\_uint32\\_t](#) bus, [l4\\_uint32\\_t](#) devfn, int pin, unsigned char \*trigger, unsigned char \*polarity) const

*Enable PCI interrupt for a specific device using the PCI root bridge.*

## Public Member Functions inherited from [L4vbus::Device](#)

- [Device](#) ()  
*Construct a new vbus device using the NULL device [L4VBUS\\_NULL](#).*
- [Device](#) ([L4::Cap](#)< [Vbus](#) > bus, [l4vbus\\_device\\_handle\\_t](#) dev)  
*Construct a new vbus device using a device handle.*
- [L4::Cap](#)< [Vbus](#) > [bus\\_cap](#) () const  
*Access the [Vbus](#) capability of the underlying virtual bus.*
- [l4vbus\\_device\\_handle\\_t](#) [dev\\_handle](#) () const  
*Access the device handle of this device.*
- int [device\\_by\\_hid](#) ([Device](#) \*child, char const \*hid, int depth=[L4VBUS\\_MAX\\_DEPTH](#), [l4vbus\\_device\\_t](#) \*devinfo=0) const  
*Find a device by the hardware interface identifier (HID).*
- int [next\\_device](#) ([Device](#) \*child, int depth=[L4VBUS\\_MAX\\_DEPTH](#), [l4vbus\\_device\\_t](#) \*devinfo=0) const  
*Find next child following *child*.*
- int [device](#) ([l4vbus\\_device\\_t](#) \*devinfo) const  
*Obtain detailed information about a [Vbus](#) device.*
- int [get\\_resource](#) (unsigned res\_idx, [l4vbus\\_resource\\_t](#) \*res) const  
*Obtain the resource description of an individual device resource.*
- int [is\\_compatible](#) (char const \*cid) const  
*Check if the given device has a compatibility ID (CID) or HID that matches cid.*
- bool [operator==](#) ([Device](#) const &o) const  
*Test if two devices are the same [Vbus](#) device.*
- bool [operator!=](#) ([Device](#) const &o) const  
*Test if two [Vbus](#) devices are not the same.*

## Public Member Functions inherited from [L4vbus::Pm](#)< [Device](#) >

- int [pm\\_suspend](#) () const  
*Suspend the device.*
- int [pm\\_resume](#) () const  
*Resume the device.*

## Additional Inherited Members

## Protected Attributes inherited from [L4vbus::Device](#)

- [L4::Cap](#)< [Vbus](#) > [\\_bus](#)  
*The [Vbus](#) capability where this device is located on.*
- [l4vbus\\_device\\_handle\\_t](#) [\\_dev](#)  
*The device handle for this device.*

### 16.384.1 Detailed Description

A Pci host bridge.

Definition at line 25 of file [vbus\\_pci](#).

### 16.384.2 Member Function Documentation

#### 16.384.2.1 `cfg_read()`

```
int L4vbus::Pci_host_bridge::cfg_read (
    14_uint32_t bus,
    14_uint32_t devfn,
    14_uint32_t reg,
    14_uint32_t * value,
    14_uint32_t width) const [inline]
```

Read from the vPCI configuration space using the PCI root bridge.

#### Parameters

	<i>bus</i>	Bus number
	<i>devfn</i>	Device id (upper 16bit) and function (lower 16bit)
	<i>reg</i>	Register in configuration space to read
out	<i>value</i>	Value that has been read
	<i>width</i>	Width to read in bits (e.g. 8, 16, 32)

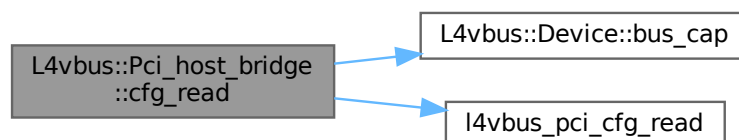
#### Returns

0 on success, else failure

Definition at line 39 of file [vbus\\_pci](#).

References [L4vbus::Device::\\_dev](#), [L4vbus::Device::bus\\_cap\(\)](#), and [l4vbus\\_pci\\_cfg\\_read\(\)](#).

Here is the call graph for this function:



### 16.384.2.2 cfg\_write()

```
int L4vbus::Pci_host_bridge::cfg_write (
    14_uint32_t bus,
    14_uint32_t devfn,
    14_uint32_t reg,
    14_uint32_t value,
    14_uint32_t width) const [inline]
```

Write to the vPCI configuration space using the PCI root bridge.

#### Parameters

<i>bus</i>	Bus number
<i>devfn</i>	<a href="#">Device</a> id (upper 16bit) and function (lower 16bit)
<i>reg</i>	Register in configuration space to write
<i>value</i>	Value to write
<i>width</i>	Width to write in bits (e.g. 8, 16, 32)

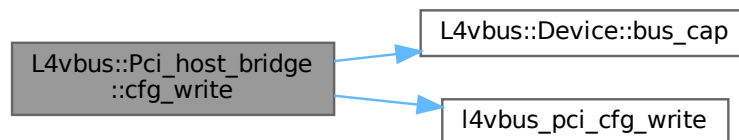
#### Returns

0 on success, else failure

Definition at line 58 of file [vbus\\_pci](#).

References [L4vbus::Device::\\_dev](#), [L4vbus::Device::bus\\_cap\(\)](#), and [l4vbus\\_pci\\_cfg\\_write\(\)](#).

Here is the call graph for this function:



### 16.384.2.3 irq\_enable()

```
int L4vbus::Pci_host_bridge::irq_enable (
    14_uint32_t bus,
    14_uint32_t devfn,
    int pin,
    unsigned char * trigger,
    unsigned char * polarity) const [inline]
```

Enable PCI interrupt for a specific device using the PCI root bridge.

#### Parameters

	<i>bus</i>	Bus number
	<i>devfn</i>	<a href="#">Device</a> id (upper 16bit) and function (lower 16bit)
	<i>pin</i>	Interrupt pin (normally as reported in configuration register INTR)
out	<i>trigger</i>	False if interrupt is level-triggered
out	<i>polarity</i>	True if interrupt is of low polarity

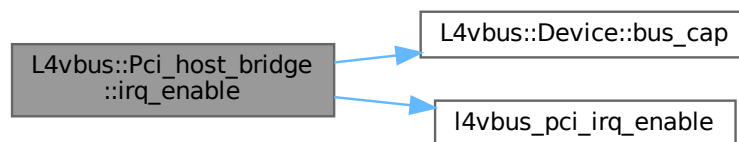
#### Returns

On success: Interrupt line to be used, else failure

Definition at line 79 of file [vbus\\_pci](#).

References [L4vbus::Device::\\_dev](#), [L4vbus::Device::bus\\_cap\(\)](#), and [l4vbus\\_pci\\_irq\\_enable\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

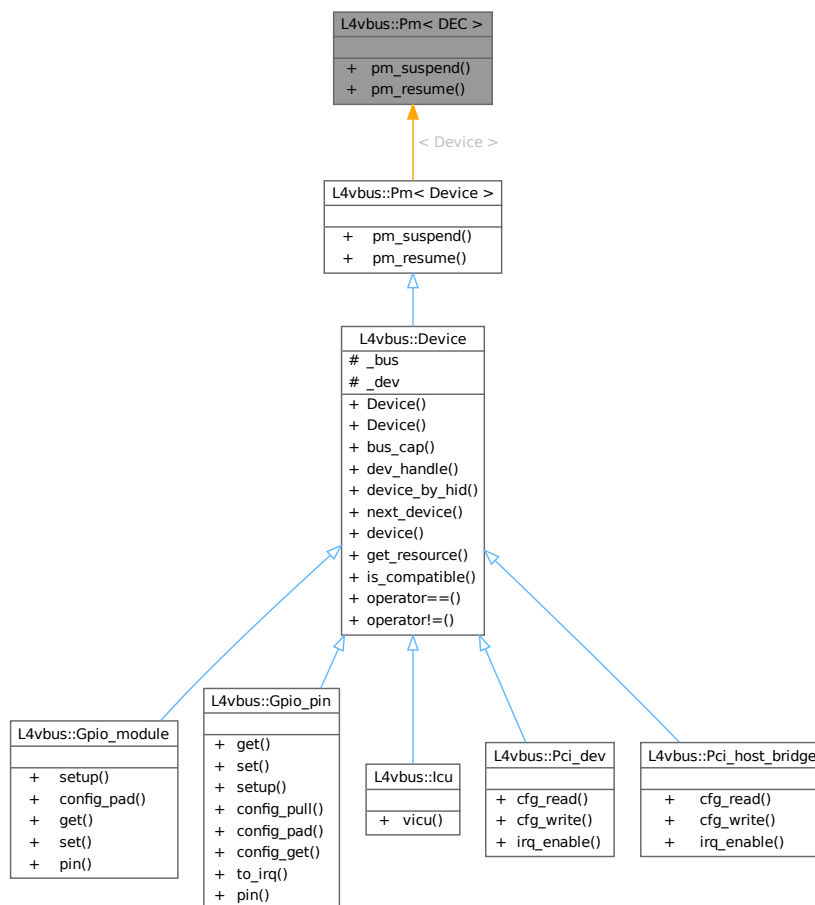
- `l4/vbus/vbus_pci`

## 16.385 L4vbus::Pm< DEC > Class Template Reference

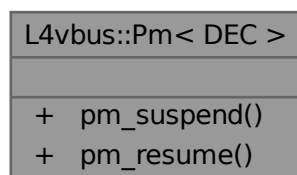
Power-management API mixin.

```
#include <vbus>
```

Inheritance diagram for L4vbus::Pm< DEC >:



Collaboration diagram for L4vbus::Pm< DEC >:



## Public Member Functions

- int [pm\\_suspend](#) () const  
*Suspend the device.*
- int [pm\\_resume](#) () const  
*Resume the device.*

### 16.385.1 Detailed Description

```
template<typename DEC>
class L4vbus::Pm< DEC >
```

Power-management API mixin.

Devices that inherit from this mixin provide an API to be suspended and resumed.

Definition at line 50 of file [vbus](#).

### 16.385.2 Member Function Documentation

#### 16.385.2.1 pm\_resume()

```
template<typename DEC>
int L4vbus::Pm< DEC >::pm_resume () const [inline]
```

Resume the device.

Switches the device from low-power mode to normal operation and restores the saved state.

#### Return values

0	Success.
---	----------

Definition at line 74 of file [vbus](#).

References [l4vbus\\_pm\\_resume\(\)](#).

Here is the call graph for this function:



#### 16.385.2.2 pm\_suspend()

```
template<typename DEC>
int L4vbus::Pm< DEC >::pm_suspend () const [inline]
```

Suspend the device.

Saves the state of the device and puts it into a low-power mode.

#### Return values



0	Success.
---	----------

Definition at line 63 of file [vbus](#).

References [l4vbus\\_pm\\_suspend\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

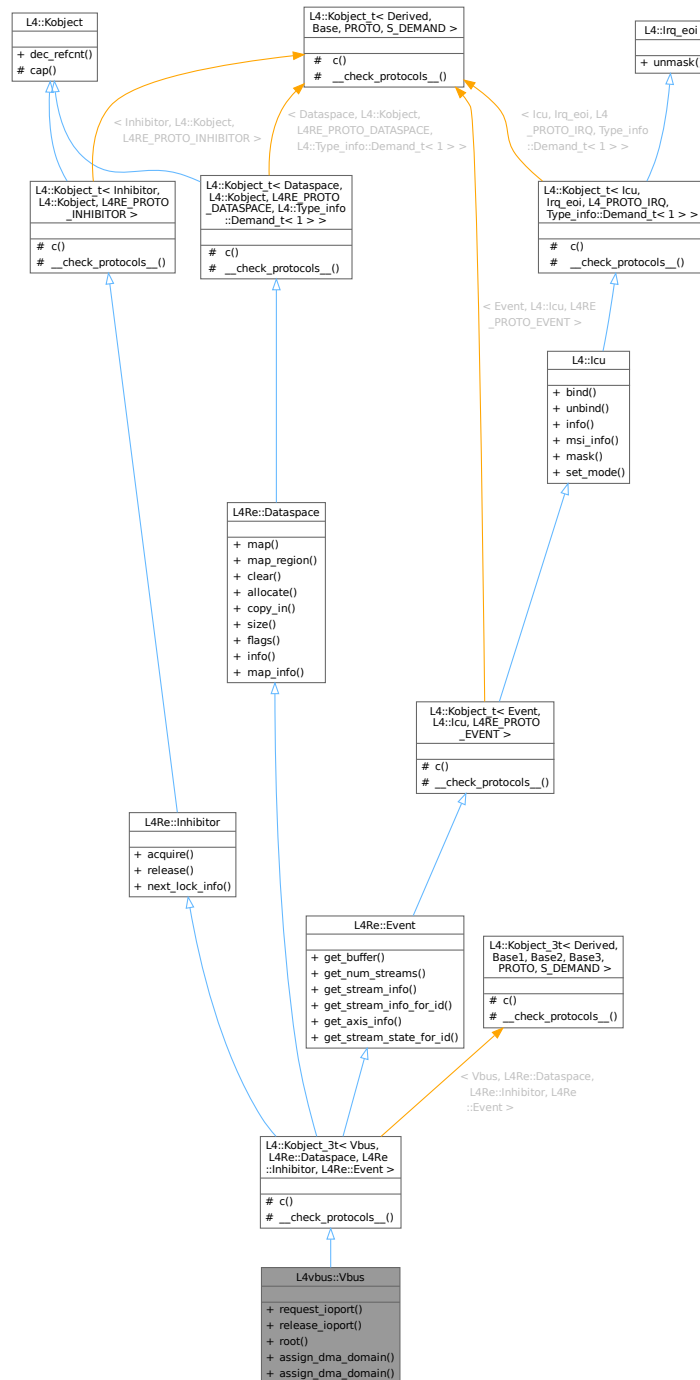
- `I4/vbus/vbus`

## 16.386 L4vbus::Vbus Class Reference

The virtual bus ([Vbus](#)) interface.

```
#include <vbus>
```

Inheritance diagram for L4vbus::Vbus:





- Generated for L4Re by Doxygen

*Get the root device of the device tree of this bus.*

- int [assign\\_dma\\_domain](#) (unsigned domain\_id, unsigned flags, [L4::Cap](#)< [L4Re::Dma\\_space](#) > dma\_space) const

*Bind or unbind an [L4Re::Dma\\_space](#) to a DMA domain.*

- int [assign\\_dma\\_domain](#) (unsigned domain\_id, unsigned flags, [L4::Cap](#)< [L4::Task](#) > dma\_space) const

*Bind or unbind a kernel [DMA space](#) to a DMA domain.*

## Public Member Functions inherited from [L4Re::Dataspace](#)

- long [map](#) (Offset offset, Flags flags, Map\_addr local\_addr, Map\_addr min\_addr, Map\_addr max\_addr, [L4::Cap](#)< [L4::Task](#) > dst=[L4::Cap](#)< [L4::Task](#) >::Invalid) const noexcept

*Request a flexpage mapping from the dataspace.*

- long [map\\_region](#) (Offset offset, Flags flags, Map\_addr min\_addr, Map\_addr max\_addr, [L4::Cap](#)< [L4::Task](#) > dst=[L4::Cap](#)< [L4::Task](#) >::Invalid) const noexcept

*Map a part of a dataspace into a local memory area.*

- long [clear](#) (Offset offset, Size size)

*Clear parts of a dataspace.*

- long [allocate](#) (Offset offset, Size size)

*Allocate a range in the dataspace.*

- long [copy\\_in](#) (Offset dst\_offs, [L4::lpc::Cap](#)< [Dataspace](#) > src, Offset src\_offs, Size size)

*Copy contents from another dataspace.*

- Size [size](#) () const noexcept

*Get size of a dataspace.*

- Flags [flags](#) () const noexcept

*Get flags of the dataspace.*

- long [info](#) ([Stats](#) \*stats)

*Get information on the dataspace.*

- long [map\\_info](#) ([l4\\_addr\\_t](#) \*start\_addr, [l4\\_addr\\_t](#) \*end\_addr)

*Get mapping range of dataspace.*

## Public Member Functions inherited from [L4::Kobject](#)

- [l4\\_msgtag\\_t](#) [dec\\_refcnt](#) ([l4\\_mword\\_t](#) diff, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)())

*Decrement the in kernel reference counter for the object.*

## Public Member Functions inherited from [L4Re::Inhibitor](#)

- long [acquire](#) ([l4\\_umword\\_t](#) id, [L4::lpc::String](#)<> reason)

*Acquire a specific inhibitor lock.*

- long [release](#) ([l4\\_umword\\_t](#) id)

*Release a specific inhibitor lock.*

- long [next\\_lock\\_info](#) (char \*name, unsigned len, [l4\\_mword\\_t](#) current\_id=-1, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)())

*Get information for the next available inhibitor lock.*

## Public Member Functions inherited from L4Re::Event

- long `get_buffer` (L4::lpc::Out< L4::Cap< Dataspace > > ds)  
*Get event signal buffer.*
- long `get_num_streams` ()  
*Get number of event streams.*
- long `get_stream_info` (int idx, Event\_stream\_info \*info)  
*Get event stream infos.*
- long `get_stream_info_for_id` (l4\_umword\_t stream\_id, Event\_stream\_info \*info)  
*Get event stream infos.*
- long `get_axis_info` (l4\_umword\_t stream\_id, unsigned naxes, unsigned const \*axis, Event\_absinfo \*info) const noexcept  
*Get event stream axis infos.*
- long `get_stream_state_for_id` (l4\_umword\_t stream\_id, Event\_stream\_state \*state)  
*Get event stream state.*

## Public Member Functions inherited from L4::Icu

- `l4_msgtag_t bind` (unsigned irqnum, L4::Cap< Triggerable > irq, l4\_utcb\_t \*utcb=l4\_utcb()) noexcept  
*Bind an interrupt line of an interrupt controller to an interrupt object.*
- `l4_msgtag_t unbind` (unsigned irqnum, L4::Cap< Triggerable > irq, l4\_utcb\_t \*utcb=l4\_utcb()) noexcept  
*Remove binding of an interrupt line from the interrupt controller object.*
- `l4_msgtag_t info` (l4\_icu\_info\_t \*info, l4\_utcb\_t \*utcb=l4\_utcb()) noexcept  
*Get information about the ICU features.*
- `l4_msgtag_t msi_info` (l4\_umword\_t irqnum, l4\_uint64\_t source, l4\_icu\_msi\_info\_t \*msi\_info)  
*Get MSI info about IRQ.*
- `l4_msgtag_t mask` (unsigned irqnum, l4\_umword\_t \*label=0, l4\_timeout\_t to=L4\_IPC\_NEVER, l4\_utcb\_t \*utcb=l4\_utcb()) noexcept  
*Mask an IRQ line.*
- `l4_msgtag_t set_mode` (unsigned irqnum, l4\_umword\_t mode, l4\_utcb\_t \*utcb=l4\_utcb()) noexcept  
*Set interrupt mode.*

## Public Member Functions inherited from L4::Irq\_eoi

- `l4_msgtag_t unmask` (unsigned irqnum, l4\_umword\_t \*label=0, l4\_timeout\_t to=L4\_IPC\_NEVER, l4\_utcb\_t \*utcb=l4\_utcb()) noexcept  
*Unmask the given interrupt line.*

## Additional Inherited Members

## Public Types inherited from L4Re::Inhibitor

- enum { `Name_max` = 20 }

## Protected Types inherited from

**L4::Kobject\_3t**< **Vbus**, **L4Re::Dataspace**, **L4Re::Inhibitor**, **L4Re::Event** >

- typedef Vbus **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< **PROTO\_ANY**, Vbus > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, Typeid::Merge\_list< typename L4Re::↵  
 Dataspace::\_\_Iface\_list, Typeid::Merge\_list< typename L4Re::Inhibitor::\_\_Iface\_list, typename L4Re::↵  
 Event::\_\_Iface\_list > > > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Types inherited from

**L4::Kobject\_t**< **Dataspace**, **L4::Kobject**, **L4RE\_PROTO\_DATASPACE**, **L4::Type\_info::Demand\_t**< 1 > >

- typedef Dataspace **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< **PROTO**, Dataspace > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename L4::Kobject::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Types inherited from

**L4::Kobject\_t**< **Inhibitor**, **L4::Kobject**, **L4RE\_PROTO\_INHIBITOR** >

- typedef Inhibitor **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< **PROTO**, Inhibitor > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename L4::Kobject::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Types inherited from **L4::Kobject\_t**< **Event**, **L4::lcu**, **L4RE\_PROTO\_EVENT** >

- typedef Event **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< **PROTO**, Event > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename L4::lcu::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

## Protected Types inherited from

**L4::Kobject\_t**< **lcu**, **lrc\_eoi**, **L4\_PROTO\_IRQ**, **Type\_info::Demand\_t**< 1 > >

- typedef lcu **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< **PROTO**, lcu > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename lrc\_eoi::\_\_Iface\_list > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

**Protected Member Functions inherited from****L4::Kobject\_3t< Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event >**

- **L4::Cap< Class > c ()** const noexcept

*Get the capability to ourselves.***Protected Member Functions inherited from****L4::Kobject\_t< Dataspace, L4::Kobject, L4RE\_PROTO\_DATASPACE, L4::Type\_info::Demand\_t< 1 > >**

- **L4::Cap< Class > c ()** const noexcept

*Get the capability to ourselves.***Protected Member Functions inherited from L4::Kobject**

- **l4\_cap\_idx\_t cap ()** const noexcept

*Return capability selector.***Protected Member Functions inherited from****L4::Kobject\_t< Inhibitor, L4::Kobject, L4RE\_PROTO\_INHIBITOR >**

- **L4::Cap< Class > c ()** const noexcept

*Get the capability to ourselves.***Protected Member Functions inherited from****L4::Kobject\_t< Event, L4::lcu, L4RE\_PROTO\_EVENT >**

- **L4::Cap< Class > c ()** const noexcept

*Get the capability to ourselves.***Protected Member Functions inherited from****L4::Kobject\_t< lcu, Irq\_eoi, L4\_PROTO\_IRQ, Type\_info::Demand\_t< 1 > >**

- **L4::Cap< Class > c ()** const noexcept

*Get the capability to ourselves.***Static Protected Member Functions inherited from****L4::Kobject\_3t< Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event >**

- static void **\_\_check\_protocols\_\_ ()** noexcept

*Helper to check for protocol conflicts.***Static Protected Member Functions inherited from****L4::Kobject\_t< Dataspace, L4::Kobject, L4RE\_PROTO\_DATASPACE, L4::Type\_info::Demand\_t< 1 > >**

- static void **\_\_check\_protocols\_\_ ()** noexcept

*Helper to check for protocol conflicts.*

**Static Protected Member Functions inherited from****[L4::Kobject\\_t< Inhibitor, L4::Kobject, L4RE\\_PROTO\\_INHIBITOR >](#)**

- static void `__check_protocols__()` noexcept

*Helper to check for protocol conflicts.***Static Protected Member Functions inherited from****[L4::Kobject\\_t< Event, L4::lcu, L4RE\\_PROTO\\_EVENT >](#)**

- static void `__check_protocols__()` noexcept

*Helper to check for protocol conflicts.***Static Protected Member Functions inherited from****[L4::Kobject\\_t< lcu, Irq\\_eoi, L4\\_PROTO\\_IRQ, Type\\_info::Demand\\_t< 1 > >](#)**

- static void `__check_protocols__()` noexcept

*Helper to check for protocol conflicts.***16.386.1 Detailed Description**

The virtual bus ([Vbus](#)) interface.

See also

[L4Re Vbus API](#)

Definition at line [298](#) of file [vbus](#).

**16.386.2 Member Function Documentation****16.386.2.1 `assign_dma_domain()` [1/2]**

```
int L4vbus::Vbus::assign_dma_domain (
    unsigned domain_id,
    unsigned flags,
    L4::Cap< L4::Task > dma_space) const [inline]
```

Bind or unbind a kernel [DMA space](#) to a DMA domain.

**Parameters**

<i>domain_id</i>	DMA domain ID (resource address of DMA domain found on the vBUS). If the value is ~0U the DMA space of the whole vBUS is used.
<i>flags</i>	A combination of <a href="#">L4vbus_dma_domain_assign_flags</a> .
<i>dma_space</i>	The DMA space capability to bind or unbind, this must be a kernel <a href="#">DMA space</a> ( <a href="#">L4::Task</a> created with <code>L4_PROTO_DMA_SPACE</code> )

**Return values**

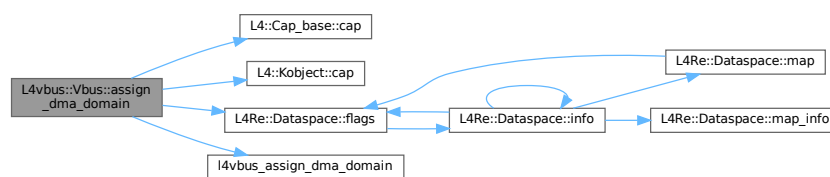


0	Operation completed successfully.
-L4_ENOENT	The vbus does not support a global DMA domain or no DMA domain could be found.
-L4_EINVAL	Invalid argument used.
-L4_EBUSY	DMA domain is already active, this means another DMA space is already assigned.

Definition at line 382 of file [vbus](#).

References [L4::Cap\\_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), [L4Re::Dataspace::flags\(\)](#), [l4vbus\\_assign\\_dma\\_domain\(\)](#), [L4VBUS\\_DMAD\\_KERNEL\\_DMA\\_SPACE](#), and [L4VBUS\\_DMAD\\_L4RE\\_DMA\\_SPACE](#).

Here is the call graph for this function:



### 16.386.2.2 assign\_dma\_domain() [2/2]

```

int L4vbus::Vbus::assign_dma_domain (
    unsigned domain_id,
    unsigned flags,
    L4::Cap< L4Re::Dma_space > dma_space) const [inline]

```

Bind or unbind an [L4Re::Dma\\_space](#) to a DMA domain.

#### Parameters

<i>domain_id</i>	DMA domain ID (resource address of DMA domain found on the vBUS). If the value is ~0U the DMA space of the whole vBUS is used.
<i>flags</i>	A combination of <a href="#">L4vbus_dma_domain_assign_flags</a> .
<i>dma_space</i>	The DMA space capability to bind or unbind, this must be an <a href="#">L4Re::Dma_space</a>

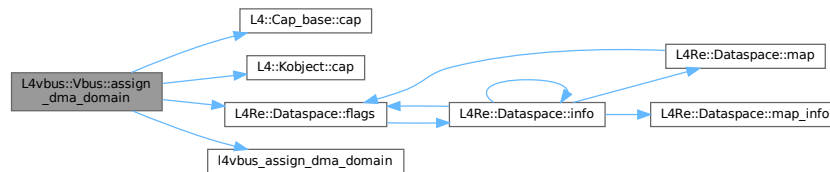
#### Return values

0	Operation completed successfully.
-L4_ENOENT	The vbus does not support a global DMA domain or no DMA domain could be found.
-L4_EINVAL	Invalid argument used.
-L4_EBUSY	DMA domain is already active, this means another DMA space is already assigned.

Definition at line 357 of file [vbus](#).

References [L4::Cap\\_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), [L4Re::Dataspace::flags\(\)](#), [l4vbus\\_assign\\_dma\\_domain\(\)](#), [L4VBUS\\_DMAD\\_KERNEL\\_DMA\\_SPACE](#), and [L4VBUS\\_DMAD\\_L4RE\\_DMA\\_SPACE](#).

Here is the call graph for this function:



### 16.386.2.3 release\_ioport()

```
int L4vbus::Vbus::release_ioport (
    l4vbus_resource_t * res) const [inline]
```

Release the given IO port resource from the bus.

#### Parameters

in	res	The IO port resource to be released from the bus.
----	-----	---------------------------------------------------

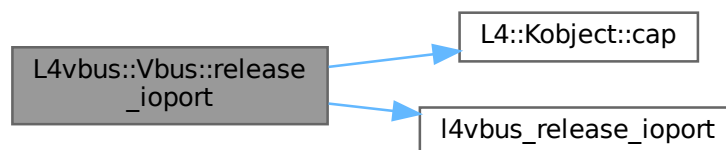
#### Returns

>=0 on success, <0 on error.

Definition at line 323 of file [vbus](#).

References [L4::Kobject::cap\(\)](#), and [l4vbus\\_release\\_ioport\(\)](#).

Here is the call graph for this function:



### 16.386.2.4 request\_ioport()

```
int L4vbus::Vbus::request_ioport (
    l4vbus_resource_t * res) const [inline]
```

Request the given IO port resource from the bus.

#### Parameters

<i>in</i>	<i>res</i>	The IO port resource to be requested from the bus.
-----------	------------	----------------------------------------------------

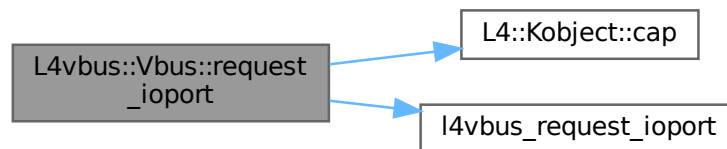
**Return values**

<i>0</i>	Success.
<i>-L4_EINVAL</i>	Resource is not an IO port resource.
<i>-L4_ENOENT</i>	No matching IO port resource found.

Definition at line 311 of file [vbus](#).

References [L4::Kobject::cap\(\)](#), and [l4vbus\\_request\\_ioport\(\)](#).

Here is the call graph for this function:

**16.386.2.5 root()**

```
Device L4vbus::Vbus::root () const [inline]
```

Get the root device of the device tree of this bus.

The root device is usually the starting point for iterating the bus, see [Device::next\\_device](#).

**Returns**

A [Vbus](#) device representing the root of the device tree.

Definition at line 336 of file [vbus](#).

References [L4::Kobject::cap\(\)](#), and [L4VBUS\\_ROOT\\_BUS](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

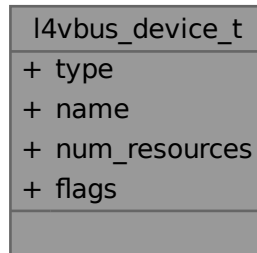
- `l4/vbus/vbus`

## 16.387 l4vbus\_device\_t Struct Reference

Detailed information about a vbus device.

```
#include <vbus_types.h>
```

Collaboration diagram for l4vbus\_device\_t:



### Data Fields

- [l4\\_uint32\\_t](#) **type**  
*Bitfield of supported sub-interfaces, see [l4vbus\\_iface\\_type\\_t](#).*
- char **name** [L4VBUS\_DEV\_NAME\_LEN]  
*Name.*
- unsigned **num\_resources**  
*Number of resources for this device.*
- unsigned **flags**  
*Flags, see [l4vbus\\_device\\_flags\\_t](#).*

### 16.387.1 Detailed Description

Detailed information about a vbus device.

Definition at line 80 of file [vbus\\_types.h](#).

The documentation for this struct was generated from the following file:

- [l4/vbus/vbus\\_types.h](#)

## 16.388 l4vbus\_resource\_t Struct Reference

Description of a single vbus resource.

```
#include <vbus_types.h>
```

Collaboration diagram for l4vbus\_resource\_t:

l4vbus_resource_t	
+	type
+	flags
+	start
+	end
+	provider
+	id

### Data Fields

- [l4\\_uint16\\_t](#) **type**  
*Resource type, see [l4vbus\\_resource\\_type\\_t](#).*
- [l4\\_uint16\\_t](#) **flags**  
*Flags.*
- [l4vbus\\_paddr\\_t](#) **start**  
*Start of resource range.*
- [l4vbus\\_paddr\\_t](#) **end**  
*End of resource range (inclusive).*
- [l4vbus\\_device\\_handle\\_t](#) **provider**  
*Device handle of the provider of the resource.*
- [l4\\_uint32\\_t](#) **id**  
*Resource ID (4 bytes), usually a 4 letter ASCII name is used.*

### 16.388.1 Detailed Description

Description of a single vbus resource.

Definition at line 23 of file [vbus\\_types.h](#).

The documentation for this struct was generated from the following file:

- [l4/vbus/vbus\\_types.h](#)

## 16.389 L4vcpu::State Class Reference

C++ implementation of state word in the vCPU area.

```
#include <vcpu>
```

Collaboration diagram for L4vcpu::State:

L4vcpu::State	
+	State()
+	add()
+	clear()
+	set()

### Public Member Functions

- [State](#) (unsigned v)  
*Initialize state.*
- void [add](#) (unsigned bits) throw ()  
*Add flags.*
- void [clear](#) (unsigned bits) throw ()  
*Clear flags.*
- void [set](#) (unsigned v) throw ()  
*Set flags.*

### 16.389.1 Detailed Description

C++ implementation of state word in the vCPU area.

Definition at line [24](#) of file [vcpu](#).

### 16.389.2 Constructor & Destructor Documentation

#### 16.389.2.1 State()

```
L4vcpu::State::State (
    unsigned v) [inline], [explicit]
```

Initialize state.

#### Parameters

---

<i>v</i>	Initial state.
----------	----------------

Definition at line 34 of file [vcpu](#).

## 16.389.3 Member Function Documentation

### 16.389.3.1 add()

```
void L4vcpu::State::add (  
    unsigned bits) throw ( )    [inline]
```

Add flags.

#### Parameters

<i>bits</i>	Bits to add to the word.
-------------	--------------------------

Definition at line 41 of file [vcpu](#).

### 16.389.3.2 clear()

```
void L4vcpu::State::clear (  
    unsigned bits) throw ( )    [inline]
```

Clear flags.

#### Parameters

<i>bits</i>	Bits to clear in the word.
-------------	----------------------------

Definition at line 48 of file [vcpu](#).

### 16.389.3.3 set()

```
void L4vcpu::State::set (  
    unsigned v) throw ( )    [inline]
```

Set flags.

#### Parameters

<i>v</i>	Set the word to the value of <i>v</i> .
----------	-----------------------------------------

Definition at line 55 of file [vcpu](#).

The documentation for this class was generated from the following file:

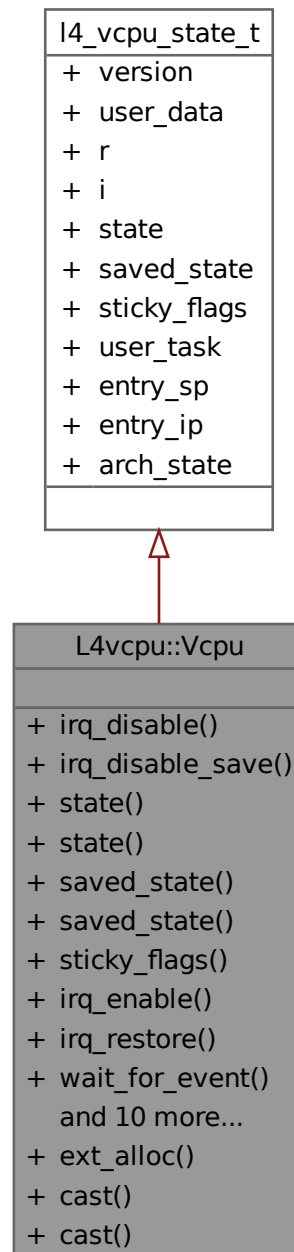
- [I4/vcpu/vcpu](#)

## 16.390 L4vcpu::Vcpu Class Reference

C++ implementation of the vCPU save state area.

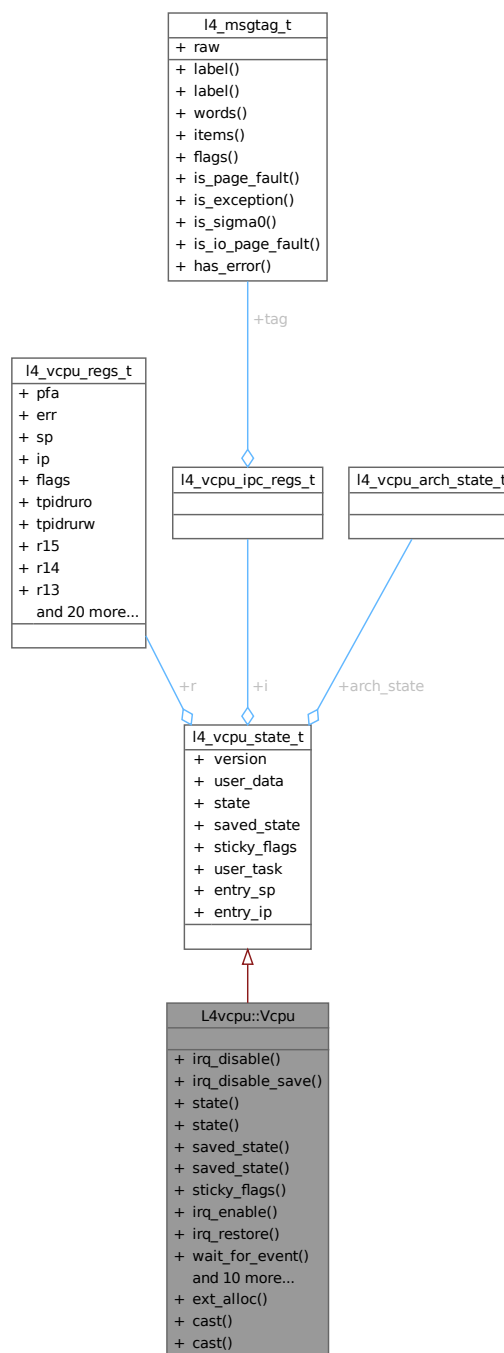
```
#include <vcpu>
```

Inheritance diagram for L4vcpu::Vcpu:





Collaboration diagram for L4vcpu::Vcpu:



## Public Member Functions

- void **irq\_disable** () throw ()  
*Disable the vCPU for event delivery.*
- unsigned **irq\_disable\_save** () throw ()  
*Disable the vCPU for event delivery and return previous state.*
- **State** \* **state** () throw ()

- Get state word.*

  - [State state](#) () const throw ()

*Get state word.*
- [State \\* saved\\_state](#) () throw ()

*Get saved\_state word.*
- [State saved\\_state](#) () const throw ()

*Get saved\_state word.*
- [l4\\_uint16\\_t sticky\\_flags](#) () const throw ()

*Get sticky flags.*
- void [irq\\_enable](#) ([l4\\_utcb\\_t](#) \*utcb, [l4vcpu\\_event\\_hndl\\_t](#) do\_event\_work\_cb, [l4vcpu\\_setup\\_ipc\\_t](#) setup\_ipc) throw ()

*Enable the vCPU for event delivery.*
- void [irq\\_restore](#) (unsigned s, [l4\\_utcb\\_t](#) \*utcb, [l4vcpu\\_event\\_hndl\\_t](#) do\_event\_work\_cb, [l4vcpu\\_setup\\_ipc\\_t](#) setup\_ipc) throw ()

*Restore a previously saved IRQ/event state.*
- void [wait\\_for\\_event](#) ([l4\\_utcb\\_t](#) \*utcb, [l4vcpu\\_event\\_hndl\\_t](#) do\_event\_work\_cb, [l4vcpu\\_setup\\_ipc\\_t](#) setup\_ipc) throw ()

*Wait for event.*
- void [task](#) ([L4::Cap](#)< [L4::Task](#) > const task=[L4::Cap](#)< [L4::Task](#) >::Invalid) throw ()

*Set the task of the vCPU.*
- int [is\\_page\\_fault\\_entry](#) () const

*Return whether the entry reason was a page fault.*
- int [is\\_irq\\_entry](#) () const

*Return whether the entry reason was an IRQ/IPC message.*
- [l4\\_vcpu\\_regs\\_t](#) \* r () throw ()

*Return pointer to register state.*
- [l4\\_vcpu\\_regs\\_t](#) const \* r () const throw ()

*Return pointer to register state.*
- [l4\\_vcpu\\_ipc\\_regs\\_t](#) \* i () throw ()

*Return pointer to IPC state.*
- [l4\\_vcpu\\_ipc\\_regs\\_t](#) const \* i () const throw ()

*Return pointer to IPC state.*
- void [entry\\_sp](#) ([l4\\_umword\\_t](#) sp)

*Set vCPU entry stack pointer.*
- void [entry\\_ip](#) ([l4\\_umword\\_t](#) ip)

*Set vCPU entry instruction pointer.*
- void [print\\_state](#) (const char \*prefix="") const throw ()

*Print the state of the vCPU.*

## Static Public Member Functions

- static int [ext\\_alloc](#) ([Vcpu](#) \*\*vcpu, [l4\\_addr\\_t](#) \*ext\_state, [L4::Cap](#)< [L4::Task](#) > task=[L4Re::Env::env](#)() ->task(), [L4::Cap](#)< [L4Re::Rm](#) > rm=[L4Re::Env::env](#)() ->rm()) throw ()

*Allocate state area for an extended vCPU.*
- static [Vcpu](#) \* [cast](#) (void \*x) throw ()

*Cast a void pointer to a class pointer.*
- static [Vcpu](#) \* [cast](#) ([l4\\_addr\\_t](#) x) throw ()

*Cast an address to a class pointer.*

## 16.390.1 Detailed Description

C++ implementation of the vCPU save state area.

Definition at line 65 of file [vcpu](#).

## 16.390.2 Member Function Documentation

### 16.390.2.1 `cast()` [1/2]

```
Vcpu * L4vcpu::Vcpu::cast (
    l4_addr_t x) throw ( )    [inline], [static]
```

Cast an address to a class pointer.

#### Parameters

<i>x</i>	Pointer.
----------	----------

#### Returns

Pointer to [Vcpu](#) class.

Definition at line 269 of file [vcpu](#).

### 16.390.2.2 `cast()` [2/2]

```
Vcpu * L4vcpu::Vcpu::cast (
    void * x) throw ( )    [inline], [static]
```

Cast a void pointer to a class pointer.

#### Parameters

<i>x</i>	Pointer.
----------	----------

**Returns**

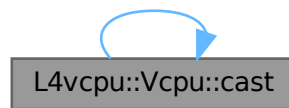
Pointer to [Vcpu](#) class.

Definition at line [259](#) of file [vcpu](#).

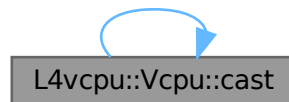
References [cast\(\)](#).

Referenced by [cast\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**16.390.2.3 entry\_ip()**

```
void L4vcpu::Vcpu::entry_ip (  
    l4\_umword\_t ip) [inline]
```

Set vCPU entry instruction pointer.

**Parameters**

<i>ip</i>	Instruction pointer address to set.
-----------	-------------------------------------

Definition at line [232](#) of file [vcpu](#).

References [l4\\_vcpu\\_state\\_t::entry\\_ip](#).

#### 16.390.2.4 entry\_sp()

```
void L4vcpu::Vcpu::entry_sp (  
    14_umword_t sp) [inline]
```

Set vCPU entry stack pointer.

##### Parameters

---

<i>sp</i>	Stack pointer address to set.
-----------	-------------------------------

**Note**

The value is only used when entering from a user-task.

Definition at line 225 of file `vcpu`.

References `l4_vcpu_state_t::entry_sp`.

**16.390.2.5 ext\_alloc()**

```
int L4vcpu::Vcpu::ext_alloc (
    Vcpu ** vcpu,
    l4_addr_t * ext_state,
    L4::Cap< L4::Task > task = L4Re::Env::env() ->task(),
    L4::Cap< L4Re::Rm > rm = L4Re::Env::env() ->rm() throw ( )    [static]
```

Allocate state area for an extended vCPU.

**Parameters**

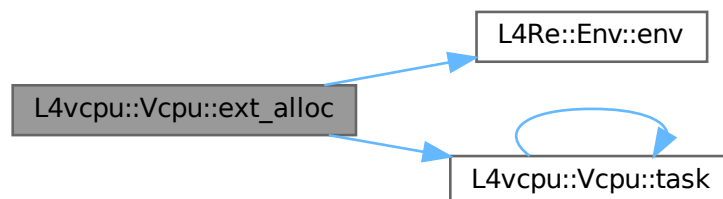
out	<i>vcpu</i>	Allocated vcpu-state area.
out	<i>ext_state</i>	Allocated extended vcpu-state area.
	<i>task</i>	Task to use for allocation, defaults to own task.
	<i>rm</i>	Region manager to use for allocation defaults to standard region manager.

**Returns**

0 for success, error code otherwise

References `L4Re::Env::env()`, and `task()`.

Here is the call graph for this function:



**16.390.2.6 i()** [1/2]

```
l4_vcpu_ipc_regs_t * L4vcpu::Vcpu::i () throw ( ) [inline]
```

Return pointer to IPC state.

**Returns**

Pointer to IPC state.

Definition at line 209 of file [vcpu](#).

References [l4\\_vcpu\\_state\\_t::i](#).

**16.390.2.7 i()** [2/2]

```
l4_vcpu_ipc_regs_t const * L4vcpu::Vcpu::i () const throw ( ) [inline]
```

Return pointer to IPC state.

**Returns**

Pointer to IPC state.

Definition at line 216 of file [vcpu](#).

References [l4\\_vcpu\\_state\\_t::i](#).

**16.390.2.8 irq\_disable\_save()**

```
unsigned L4vcpu::Vcpu::irq_disable_save () throw ( ) [inline]
```

Disable the vCPU for event delivery and return previous state.

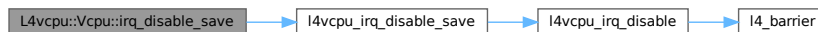
**Returns**

IRQ state before disabling IRQs.

Definition at line 78 of file [vcpu](#).

References [l4vcpu\\_irq\\_disable\\_save\(\)](#).

Here is the call graph for this function:

**16.390.2.9 irq\_enable()**

```
void L4vcpu::Vcpu::irq_enable (
    l4_utcb_t * utcb,
    l4vcpu_event_hndl_t do_event_work_cb,
    l4vcpu_setup_ipc_t setup_ipc) throw ( ) [inline]
```

Enable the vCPU for event delivery.

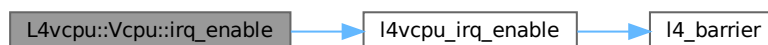
**Parameters**

<i>utcb</i>	The UTCB to use.
<i>do_event_work_cb</i>	Call-back function that is called in case an event (such as an interrupt) is pending.
<i>setup_ipc</i>	Call-back function that is called before an IPC operation is called, and before event delivery is enabled.

Definition at line 135 of file `vcpu`.

References [l4vcpu\\_irq\\_enable\(\)](#).

Here is the call graph for this function:



### 16.390.2.10 `irq_restore()`

```

void L4vcpu::Vcpu::irq_restore (
    unsigned s,
    l4_utcb_t * utcb,
    l4vcpu_event_hdl_t do_event_work_cb,
    l4vcpu_setup_ipc_t setup_ipc) throw ( )    [inline]
  
```

Restore a previously saved IRQ/event state.

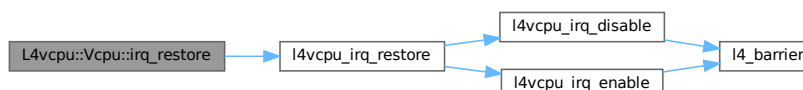
#### Parameters

<i>s</i>	IRQ state to be restored.
<i>utcb</i>	The UTCB to use.
<i>do_event_work_cb</i>	Call-back function that is called in case an event (such as an interrupt) is pending.
<i>setup_ipc</i>	Call-back function that is called before an IPC operation is called, and before event delivery is enabled.

Definition at line 150 of file `vcpu`.

References [l4vcpu\\_irq\\_restore\(\)](#).

Here is the call graph for this function:





### 16.390.2.11 is\_irq\_entry()

```
int L4vcpu::Vcpu::is_irq_entry () const [inline]
```

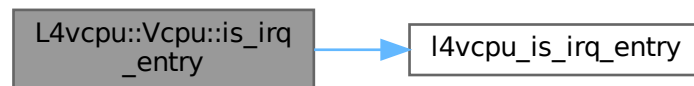
Return whether the entry reason was an IRQ/IPC message.

return 0 if not, !=0 otherwise.

Definition at line 188 of file [vcpu](#).

References [l4vcpu\\_is\\_irq\\_entry\(\)](#).

Here is the call graph for this function:



### 16.390.2.12 is\_page\_fault\_entry()

```
int L4vcpu::Vcpu::is_page_fault_entry () const [inline]
```

Return whether the entry reason was a page fault.

return 0 if not, !=0 otherwise.

Definition at line 181 of file [vcpu](#).

References [l4vcpu\\_is\\_page\\_fault\\_entry\(\)](#).

Here is the call graph for this function:



**16.390.2.13** `r()` [1/2]

```
l4_vcpu_regs_t * L4vcpu::Vcpu::r () throw ( ) [inline]
```

Return pointer to register state.

**Returns**

Pointer to register state.

Definition at line 195 of file `vcpu`.

References `l4_vcpu_state_t::r`.

**16.390.2.14** `r()` [2/2]

```
l4_vcpu_regs_t const * L4vcpu::Vcpu::r () const throw ( ) [inline]
```

Return pointer to register state.

**Returns**

Pointer to register state.

Definition at line 202 of file `vcpu`.

References `l4_vcpu_state_t::r`.

**16.390.2.15** `saved_state()` [1/2]

```
State * L4vcpu::Vcpu::saved_state () throw ( ) [inline]
```

Get `saved_state` word.

**Returns**

Pointer to `saved_state` word in the vCPU

Definition at line 106 of file `vcpu`.

References `l4_vcpu_state_t::saved_state`.

**16.390.2.16** `saved_state()` [2/2]

```
State L4vcpu::Vcpu::saved_state () const throw ( ) [inline]
```

Get `saved_state` word.

**Returns**

Pointer to `saved_state` word in the vCPU

Definition at line 116 of file `vcpu`.

References `l4_vcpu_state_t::saved_state`.

**16.390.2.17 state()** [1/2]

```
State * L4vcpu::Vcpu::state () throw ( ) [inline]
```

Get state word.

**Returns**

Pointer to state word in the vCPU

Definition at line 88 of file [vcpu](#).

References [l4\\_vcpu\\_state\\_t::state](#).

**16.390.2.18 state()** [2/2]

```
State L4vcpu::Vcpu::state () const throw ( ) [inline]
```

Get state word.

**Returns**

Pointer to state word in the vCPU

Definition at line 99 of file [vcpu](#).

References [l4\\_vcpu\\_state\\_t::state](#).

**16.390.2.19 task()**

```
void L4vcpu::Vcpu::task (
    L4::Cap< L4::Task > const task = L4::Cap<L4::Task>::Invalid) throw ( ) [inline]
```

Set the task of the vCPU.

**Parameters**

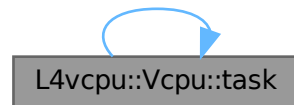
<i>task</i>	Task to set, defaults to invalid task.
-------------	----------------------------------------

Definition at line 174 of file [vcpu](#).

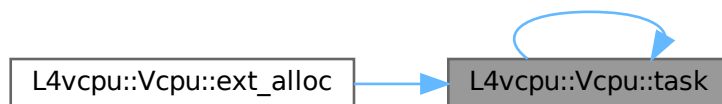
References [L4::Cap\\_base::Invalid](#), [task\(\)](#), and [l4\\_vcpu\\_state\\_t::user\\_task](#).

Referenced by [ext\\_alloc\(\)](#), and [task\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.390.2.20 wait\_for\_event()

```

void L4vcpu::Vcpu::wait_for_event (
    l4_utcb_t * utcb,
    l4vcpu_event_hndl_t do_event_work_cb,
    l4vcpu_setup_ipc_t setup_ipc) throw ( )    [inline]
  
```

Wait for event.

#### Parameters

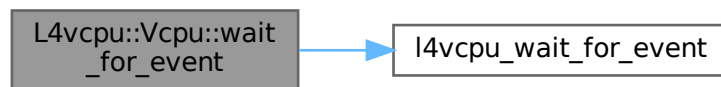
<i>utcb</i>	The UTCB to use.
<i>do_event_work_cb</i>	Call-back function that is called in case an event (such as an interrupt) is pending.
<i>setup_ipc</i>	Call-back function that is called before an IPC operation is called.

Note that event delivery remains disabled after this function returns.

Definition at line 166 of file `vcpu`.

References `l4vcpu_wait_for_event()`.

Here is the call graph for this function:



The documentation for this class was generated from the following file:

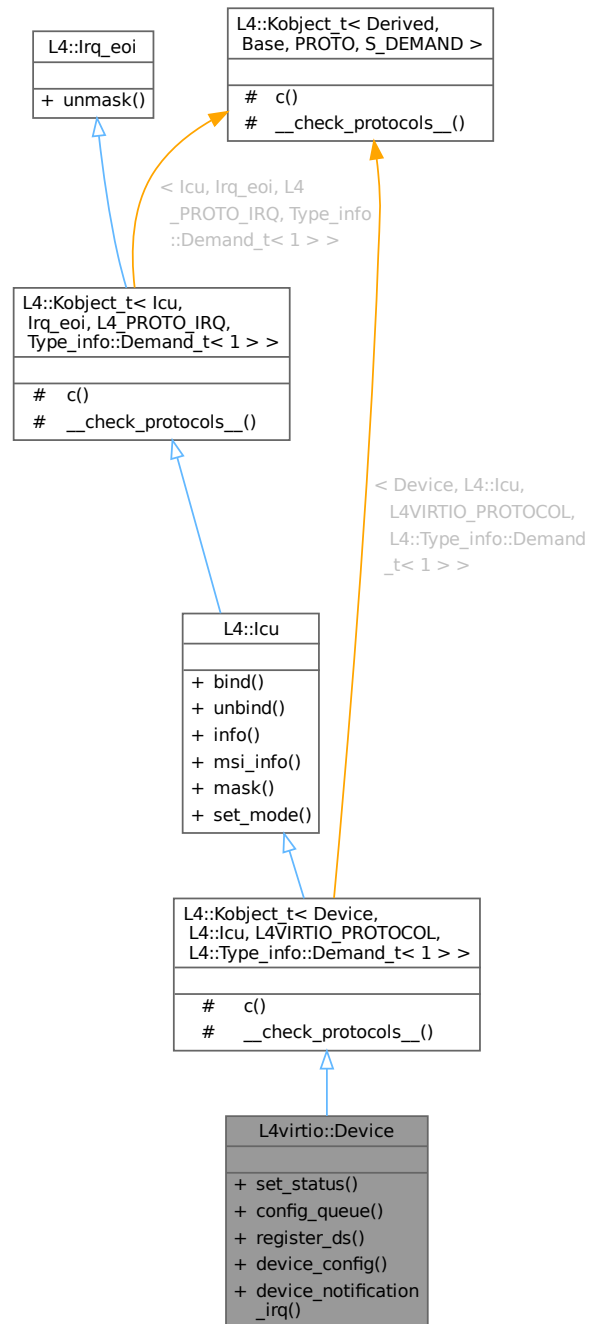
- [l4/vcpu/vcpu](#)

## 16.391 L4virtio::Device Class Reference

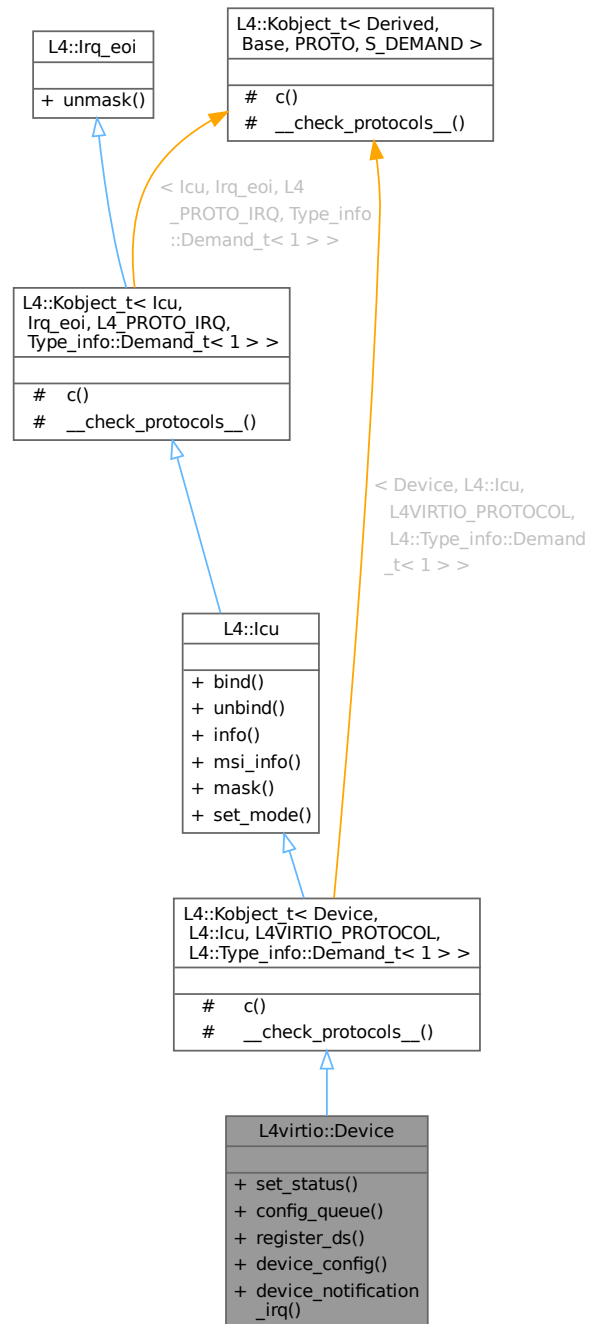
IPC interface for virtio over [L4](#) IPC.

```
#include <l4virtio>
```

Inheritance diagram for L4virtio::Device:



Collaboration diagram for L4virtio::Device:



## Public Member Functions

- long [set\\_status](#) (unsigned status)  
*Write the VIRTIO status register.*
- long [config\\_queue](#) (unsigned queue)  
*Trigger queue configuration of the given queue.*

- long `register_ds` (`L4::lpc::Cap< L4Re::Dataspace >` ds\_cap, `l4_uint64_t` base, `l4_umword_t` offset, `l4_umword_t` size)  
*Register a shared data space with VIRTIO host.*
- long `device_config` (`L4::lpc::Out< L4::Cap< L4Re::Dataspace > >` config\_ds, `l4_addr_t` \*ds\_offset)  
*Get the dataspace with the [L4virtio](#) configuration page.*
- long `device_notification_irq` (unsigned index, `L4::lpc::Out< L4::Cap< L4::Triggerable > >` irq)  
*Get the notification interrupt corresponding to the given index.*

## Public Member Functions inherited from `L4::lcu`

- `l4_msgtag_t` `bind` (unsigned irqnum, `L4::Cap< Triggerable >` irq, `l4_utcb_t` \*utcb=`l4_utcb()`) noexcept  
*Bind an interrupt line of an interrupt controller to an interrupt object.*
- `l4_msgtag_t` `unbind` (unsigned irqnum, `L4::Cap< Triggerable >` irq, `l4_utcb_t` \*utcb=`l4_utcb()`) noexcept  
*Remove binding of an interrupt line from the interrupt controller object.*
- `l4_msgtag_t` `info` (`l4_icu_info_t` \*info, `l4_utcb_t` \*utcb=`l4_utcb()`) noexcept  
*Get information about the ICU features.*
- `l4_msgtag_t` `msi_info` (`l4_umword_t` irqnum, `l4_uint64_t` source, `l4_icu_msi_info_t` \*msi\_info)  
*Get MSI info about IRQ.*
- `l4_msgtag_t` `mask` (unsigned irqnum, `l4_umword_t` \*label=0, `l4_timeout_t` to=`L4_IPC_NEVER`, `l4_utcb_t` \*utcb=`l4_utcb()`) noexcept  
*Mask an IRQ line.*
- `l4_msgtag_t` `set_mode` (unsigned irqnum, `l4_umword_t` mode, `l4_utcb_t` \*utcb=`l4_utcb()`) noexcept  
*Set interrupt mode.*

## Public Member Functions inherited from `L4::lrq_eoi`

- `l4_msgtag_t` `unmask` (unsigned irqnum, `l4_umword_t` \*label=0, `l4_timeout_t` to=`L4_IPC_NEVER`, `l4_utcb_t` \*utcb=`l4_utcb()`) noexcept  
*Unmask the given interrupt line.*

## Additional Inherited Members

### Protected Types inherited from

`L4::Kobject_t< Device, L4::lcu, L4VIRTIO_PROTOCOL, L4::Type_info::Demand_t< 1 > >`

- typedef Device **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< PROTO, Device > **\_\_iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_iface** >, typename L4::lcu::\_\_iface\_list > **\_\_iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Types inherited from

`L4::Kobject_t< lcu, lrq_eoi, L4_PROTO_IRQ, Type_info::Demand_t< 1 > >`

- typedef lcu **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< PROTO, lcu > **\_\_iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_iface** >, typename lrq\_eoi::\_\_iface\_list > **\_\_iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*



**Protected Member Functions inherited from****L4::Kobject\_t< Device, L4::lcu, L4VIRTIO\_PROTOCOL, L4::Type\_info::Demand\_t< 1 > >**

- **L4::Cap< Class > c ()** const noexcept  
*Get the capability to ourselves.*

**Protected Member Functions inherited from****L4::Kobject\_t< lcu, Irq\_eoi, L4\_PROTO\_IRQ, Type\_info::Demand\_t< 1 > >**

- **L4::Cap< Class > c ()** const noexcept  
*Get the capability to ourselves.*

**Static Protected Member Functions inherited from****L4::Kobject\_t< Device, L4::lcu, L4VIRTIO\_PROTOCOL, L4::Type\_info::Demand\_t< 1 > >**

- static void **\_\_check\_protocols\_\_ ()** noexcept  
*Helper to check for protocol conflicts.*

**Static Protected Member Functions inherited from****L4::Kobject\_t< lcu, Irq\_eoi, L4\_PROTO\_IRQ, Type\_info::Demand\_t< 1 > >**

- static void **\_\_check\_protocols\_\_ ()** noexcept  
*Helper to check for protocol conflicts.*

**16.391.1 Detailed Description**

IPC interface for virtio over [L4](#) IPC.

The [L4virtio](#) protocol is an adaption of the mmio virtio transport 1.0(4). This interface allows to exchange the necessary resources: device configuration page, notification interrupts and dataspace for payload.

Notification interrupts can be configured independently for changes to the configuration space and each queue through special L4virtio-specific `notify_index` fields in the config page and queue configuration. The interface distinguishes between device-to-driver and driver-to-device notification interrupts.

Device-to-driver interrupts are configured via the ICU interface. The device announces the maximum number of supported interrupts via `lcu::info()`. The driver can then bind interrupts using `lcu::bind()`.

Driver-to-device interrupts must be requested from the device through [device\\_notification\\_irq\(\)](#).

Definition at line 39 of file [l4virtio](#).

**16.391.2 Member Function Documentation****16.391.2.1 config\_queue()**

```
long L4virtio::Device::config_queue (
    unsigned queue)
```

Trigger queue configuration of the given queue.

Usually all queues are configured when the status is written to running. However, in some cases queues shall be disabled or enabled dynamically, in this case this function triggers a reconfiguration from the shared memory register of the queue config.

**Parameters**

<i>queue</i>	Queue index for the queue to be configured.
--------------	---------------------------------------------

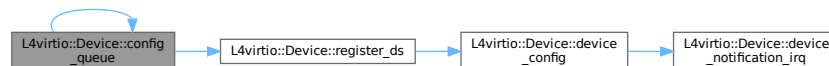
### Return values

0	on success.
-L4_EIO	The queue's status is invalid.
-L4_ERANGE	The queue index exceeds the number of queues.
-L4_EINVAL	Otherwise.

References [config\\_queue\(\)](#), [L4VIRTIO\\_OP\\_REGISTER\\_DS](#), and [register\\_ds\(\)](#).

Referenced by [config\\_queue\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.391.2.2 device\_config()

```

long L4virtio::Device::device_config (
    L4::Ipc::Out< L4::Cap< L4Re::Dataspace > > config_ds,
    l4_addr_t * ds_offset)

```

Get the dataspace with the [L4virtio](#) configuration page.

### Parameters

<i>config_ds</i>	Capability for receiving the dataspace capability for the shared L4-VIRTIO config data space.
------------------	-----------------------------------------------------------------------------------------------

<i>ds_offset</i>	Offset into the dataspace where the device configuration structure starts.
------------------	----------------------------------------------------------------------------

References [device\\_notification\\_irq\(\)](#), and [L4VIRTIO\\_OP\\_GET\\_DEVICE\\_IRQ](#).

Referenced by [register\\_ds\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.391.2.3 device\_notification\_irq()

```

long L4virtio::Device::device_notification_irq (
    unsigned index,
    L4::Ipc::Out< L4::Cap< L4::Triggerable > > irq)

```

Get the notification interrupt corresponding to the given index.

#### Parameters

	<i>index</i>	Index of the interrupt.
out	<i>irq</i>	Triggerable for the given index.

#### Return values

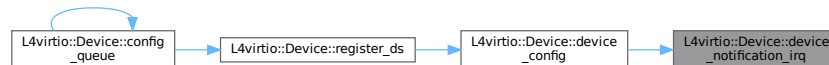
<i>L4_EOK</i>	Success.
<i>L4_ENOSYS</i>	IRQ notification not supported by device.
<0	Other error.

An index is only guaranteed to return an IRQ object when the index is set in one of the device notify index fields. The device must return the same interrupt for a given index as long as the index is in use. If an index disappears as a result of a configuration change and then is reused later, the interrupt is not guaranteed to be the same.

Interrupts must always be rerequested after a device reset.

Referenced by [device\\_config\(\)](#).

Here is the caller graph for this function:



#### 16.391.2.4 register\_ds()

```

long L4virtio::Device::register_ds (
    L4::Ipc::Cap< L4Re::Dataspace > ds_cap,
    l4_uint64_t base,
    l4_umword_t offset,
    l4_umword_t size)

```

Register a shared data space with VIRTIO host.

##### Parameters

<i>ds_cap</i>	Dataspace capability to register. The lower 8 bits determine the rights mask with which the guest's rights are masked during the registration of the dataspace at the VIRTIO host.
<i>base</i>	VIRTIO guest physical start address of shared memory region
<i>offset</i>	Offset within the data space that is attached to the given <i>base</i> in the guest physical memory.
<i>size</i>	Size of the memory region in the guest

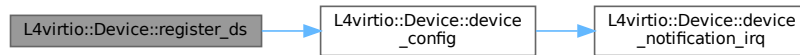
##### Return values

<i>L4_EOK</i>	Operation successful.
<i>-L4_EINVAL</i>	The <i>ds_cap</i> capability is invalid, does not refer to a valid dataspace, is not a trusted dataspace if trusted dataspace validation is enabled, or <i>size</i> and <i>offset</i> specify an invalid region.
<i>-L4_ENOMEM</i>	The limit of dataspaces that can be registered has been reached or no capability slot could be allocated.
<i>-L4_ERANGE</i>	<i>offset</i> is larger than the size of the dataspace.
<i>&lt;0</i>	Any error returned by the dataspace when queried for information during setup or any error returned by the region manager from attaching the dataspace.

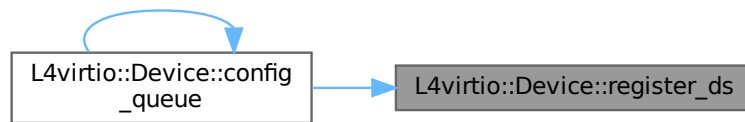
References [device\\_config\(\)](#), and [L4VIRTIO\\_OP\\_DEVICE\\_CONFIG](#).

Referenced by [config\\_queue\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.391.2.5 set\_status()

```
long L4virtio::Device::set_status (
    unsigned status)
```

Write the VIRTIO status register.

#### Parameters

<i>status</i>	Status word to write to the VIRTIO status.
---------------	--------------------------------------------

#### Return values

0	on success.
---	-------------

#### Note

All other registers are accessed via shared memory.

References [L4VIRTIO\\_OP\\_CONFIG\\_QUEUE](#), and [set\\_status\(\)](#).

Referenced by [set\\_status\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

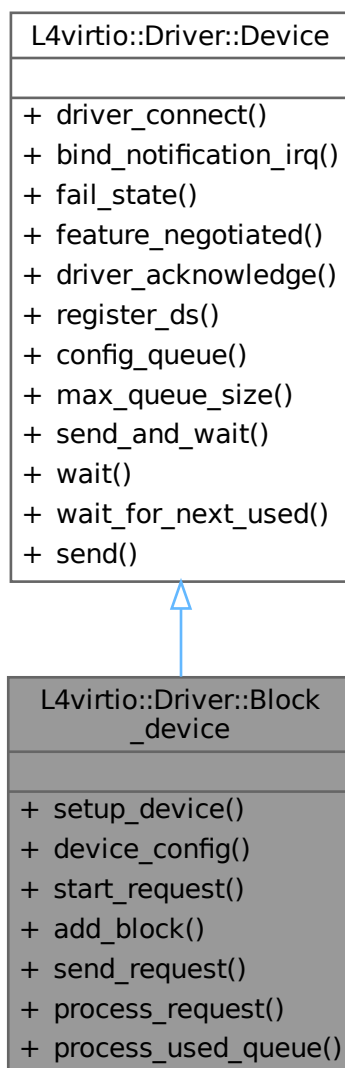
- `I4/I4virtio/I4virtio`

## 16.392 L4virtio::Driver::Block\_device Class Reference

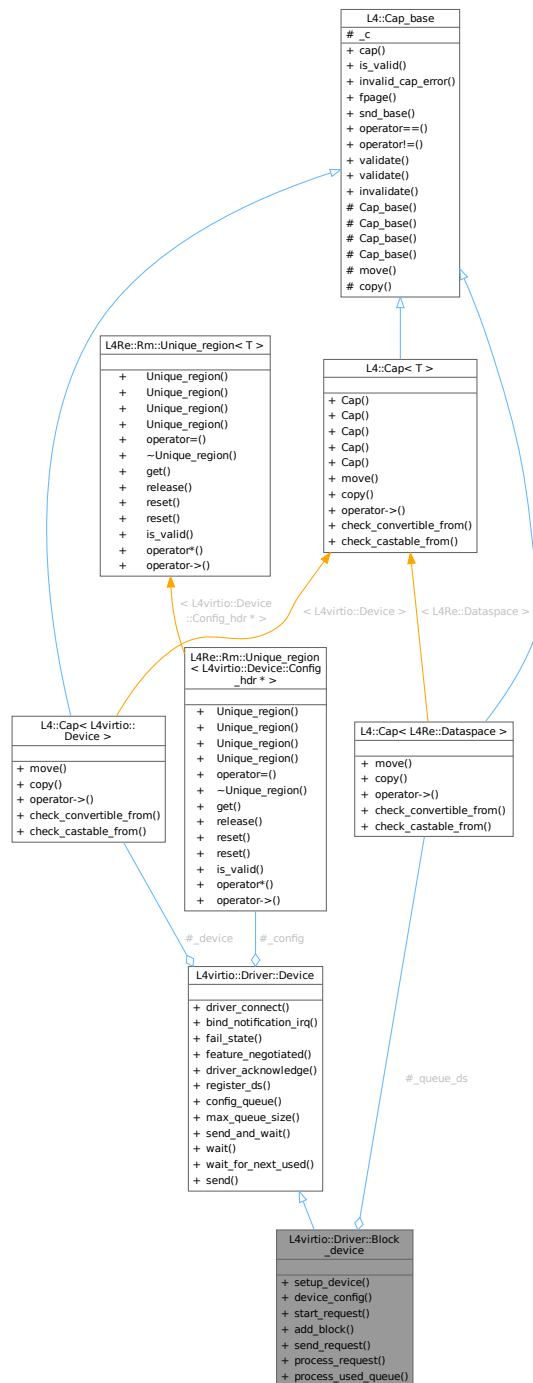
Simple class for accessing a virtio block device synchronously.

```
#include <virtio-block>
```

Inheritance diagram for L4virtio::Driver::Block\_device:



Collaboration diagram for L4virtio::Driver::Block\_device:



## Data Structures

- class `Handle`  
*Handle* to an ongoing request.



## Public Member Functions

- void [setup\\_device](#) ([L4::Cap](#)< [L4virtio::Device](#) > srvcap, [l4\\_size\\_t](#) usermem, void \*\*userdata, [Ptr](#)< void > &user\_devaddr, [L4::Cap](#)< [L4Re::Dataspace](#) > qds=[L4::Cap](#)< [L4Re::Dataspace](#) >(), [l4\\_uint32\\_t](#) fmask0=-1U, [l4\\_uint32\\_t](#) fmask1=-1U)  
*Establish a connection to the device and set up shared memory.*
- [l4virtio\\_block\\_config\\_t](#) const & **device\_config** () const  
*Return a reference to the device configuration.*
- [Handle](#) start\_request ([l4\\_uint64\\_t](#) sector, [l4\\_uint32\\_t](#) type, Callback callback)  
*Start the setup of a new request.*
- int [add\\_block](#) ([Handle](#) handle, [Ptr](#)< void > addr, [l4\\_uint32\\_t](#) size)  
*Add a data block to a request that has already been set up.*
- int [send\\_request](#) ([Handle](#) handle)  
*Process request asynchronously.*
- int [process\\_request](#) ([Handle](#) handle)  
*Process request synchronously.*
- void [process\\_used\\_queue](#) ()  
*Process and free all items in the used queue.*

## Public Member Functions inherited from [L4virtio::Driver::Device](#)

- void [driver\\_connect](#) ([L4::Cap](#)< [L4virtio::Device](#) > srvcap, bool manage\_notify=true)  
*Contacts the device and starts the initial handshake.*
- int [bind\\_notification\\_irq](#) (unsigned index, [L4::Cap](#)< [L4::Triggerable](#) > irq) const  
*Register a triggerable to receive notifications from the device.*
- bool **fail\_state** () const  
*Return true if the device is in a fail state.*
- bool [feature\\_negotiated](#) (unsigned int feat) const  
*Check if a particular feature bit was negotiated with the device.*
- int [driver\\_acknowledge](#) ()  
*Finalize handshake with the device.*
- int [register\\_ds](#) ([L4::Cap](#)< [L4Re::Dataspace](#) > ds, [l4\\_umword\\_t](#) offset, [l4\\_umword\\_t](#) size, [l4\\_uint64\\_t](#) \*devaddr)  
*Share a dataspace with the device.*
- int [config\\_queue](#) (int num, unsigned size, [l4\\_uint64\\_t](#) desc\_addr, [l4\\_uint64\\_t](#) avail\_addr, [l4\\_uint64\\_t](#) used\_↔addr)  
*Send the virtqueue configuration to the device.*
- int [max\\_queue\\_size](#) (int num) const  
*Maximum queue size allowed by the device.*
- int [send\\_and\\_wait](#) ([Virtqueue](#) &queue, [l4\\_uint16\\_t](#) descno)  
*Send a request to the device and wait for it to be processed.*
- int [wait](#) (int index) const  
*Wait for a notification from the device.*
- int [wait\\_for\\_next\\_used](#) ([Virtqueue](#) &queue, [l4\\_uint32\\_t](#) \*len=nullptr) const  
*Wait for the next item to arrive in the used queue and return it.*
- void [send](#) ([Virtqueue](#) &queue, [l4\\_uint16\\_t](#) descno)  
*Send a request to the device.*

### 16.392.1 Detailed Description

Simple class for accessing a virtio block device synchronously.

Definition at line 36 of file [virtio-block](#).

### 16.392.2 Member Function Documentation

#### 16.392.2.1 add\_block()

```
int L4virtio::Driver::Block_device::add_block (
    Handle handle,
    Ptr< void > addr,
    l4_uint32_t size) [inline]
```

Add a data block to a request that has already been set up.

#### Parameters

<i>handle</i>	<a href="#">Handle</a> to request previously set up with <a href="#">start_request()</a> .
<i>addr</i>	Address of data block in device address space.
<i>size</i>	Size of data block.

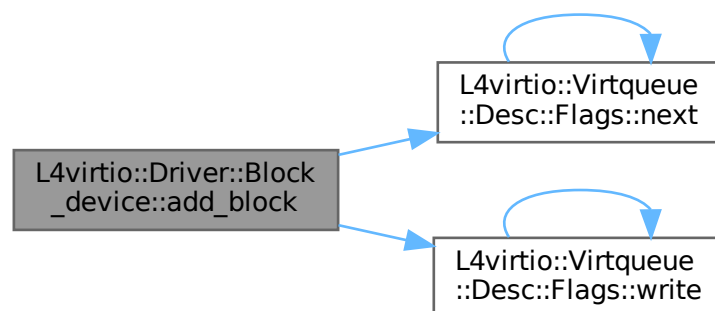
#### Return values

<i>L4_OK</i>	Block was successfully added.
<i>-L4_EAGAIN</i>	No descriptors available. Try again later.

Definition at line 229 of file [virtio-block](#).

References [L4virtio::Virtqueue::Desc::addr](#), [L4virtio::Virtqueue::Desc::flags](#), [L4\\_EAGAIN](#), [L4\\_EOK](#), [L4virtio::Virtqueue::Desc::len](#), [L4virtio::Virtqueue::Desc::Flags::next\(\)](#), [L4virtio::Virtqueue::Desc::next](#), [L4virtio::Virtqueue::Desc::Flags::raw](#), and [L4virtio::Virtqueue::Desc::Flags::write\(\)](#).

Here is the call graph for this function:



### 16.392.2.2 process\_request()

```
int L4virtio::Driver::Block_device::process_request (
    Handle handle) [inline]
```

Process request synchronously.

#### Parameters

<i>handle</i>	Handle to request to process.
---------------	-------------------------------

#### Return values

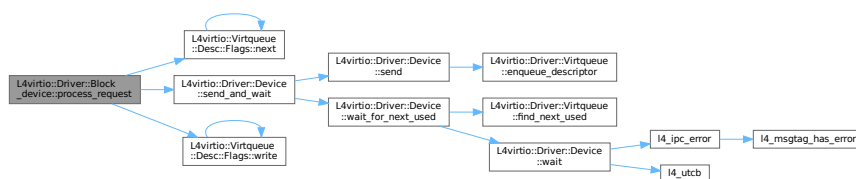
<i>L4_EOK</i>	Request processed successfully.
<i>-L4_EAGAIN</i>	No descriptors available. Try again later.
<i>-L4_EIO</i>	IO error during request processing.
<i>-L4_ENOSYS</i>	Unsupported request.
<i>&lt;0</i>	Another unspecified error occurred.

Sends a request to the driver that was previously set up with [start\\_request\(\)](#) and [add\\_block\(\)](#) and wait for it to be executed.

Definition at line 306 of file [virtio-block](#).

References [L4virtio::Virtqueue::Desc::addr](#), [L4virtio::Virtqueue::Desc::flags](#), [L4\\_EAGAIN](#), [L4\\_EINVAL](#), [L4\\_EIO](#), [L4\\_ENOSYS](#), [L4\\_EOK](#), [L4VIRTIO\\_BLOCK\\_S\\_IOERR](#), [L4VIRTIO\\_BLOCK\\_S\\_OK](#), [L4VIRTIO\\_BLOCK\\_S\\_UNSUPP](#), [L4virtio::Virtqueue::Desc::len](#), [L4virtio::Virtqueue::Desc::Flags::next\(\)](#), [L4virtio::Virtqueue::Desc::next](#), [L4virtio::Virtqueue::Desc::Flags::write\(\)](#), [L4virtio::Driver::Device::send\\_and\\_wait\(\)](#), and [L4virtio::Virtqueue::Desc::Flags::write\(\)](#).

Here is the call graph for this function:



### 16.392.2.3 process\_used\_queue()

```
void L4virtio::Driver::Block_device::process_used_queue () [inline]
```

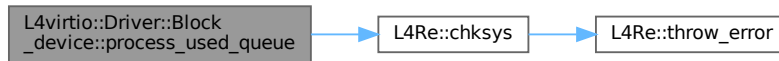
Process and free all items in the used queue.

If the request has a callback registered it is called after the item has been removed from the queue.

Definition at line 357 of file [virtio-block](#).

References [L4Re::chksys\(\)](#), and [L4\\_ENOSYS](#).

Here is the call graph for this function:



#### 16.392.2.4 send\_request()

```
int L4virtio::Driver::Block_device::send_request (
    Handle handle) [inline]
```

Process request asynchronously.

##### Parameters

<i>handle</i>	<a href="#">Handle</a> to request to send to the device
---------------	---------------------------------------------------------

##### Return values

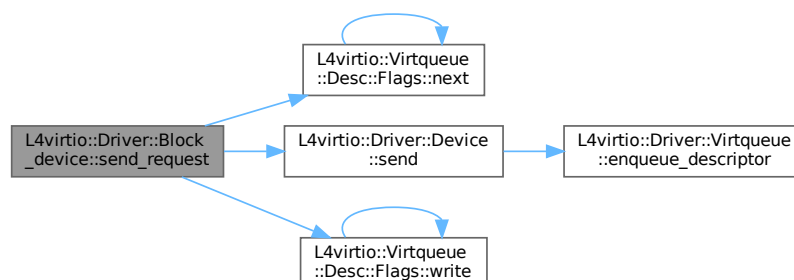
<i>L4_OK</i>	Request was successfully scheduled.
<i>-L4_EAGAIN</i>	No descriptors available. Try again later.

Sends a request to the driver that was previously set up with [start\\_request\(\)](#) and [add\\_block\(\)](#) and wait for it to be executed.

Definition at line 265 of file [virtio-block](#).

References [L4virtio::Virtqueue::Desc::addr](#), [L4virtio::Virtqueue::Desc::flags](#), [L4\\_EAGAIN](#), [L4\\_EOK](#), [L4virtio::Virtqueue::Desc::len](#), [L4virtio::Virtqueue::Desc::Flags::next\(\)](#), [L4virtio::Virtqueue::Desc::next](#), [L4virtio::Virtqueue::Desc::Flags::raw](#), [L4virtio::Driver::Device::send\(\)](#), and [L4virtio::Virtqueue::Desc::Flags::write\(\)](#).

Here is the call graph for this function:



## 16.392.2.5 setup\_device()

```
void L4virtio::Driver::Block_device::setup_device (
    L4::Cap< L4virtio::Device > srvcap,
    l4_size_t usermem,
    void ** userdata,
    Ptr< void > & user_devaddr,
    L4::Cap< L4Re::Dataspace > qds = L4::Cap<L4Re::Dataspace>(),
    l4_uint32_t fmask0 = -1U,
    l4_uint32_t fmask1 = -1U) [inline]
```

Establish a connection to the device and set up shared memory.

## Parameters

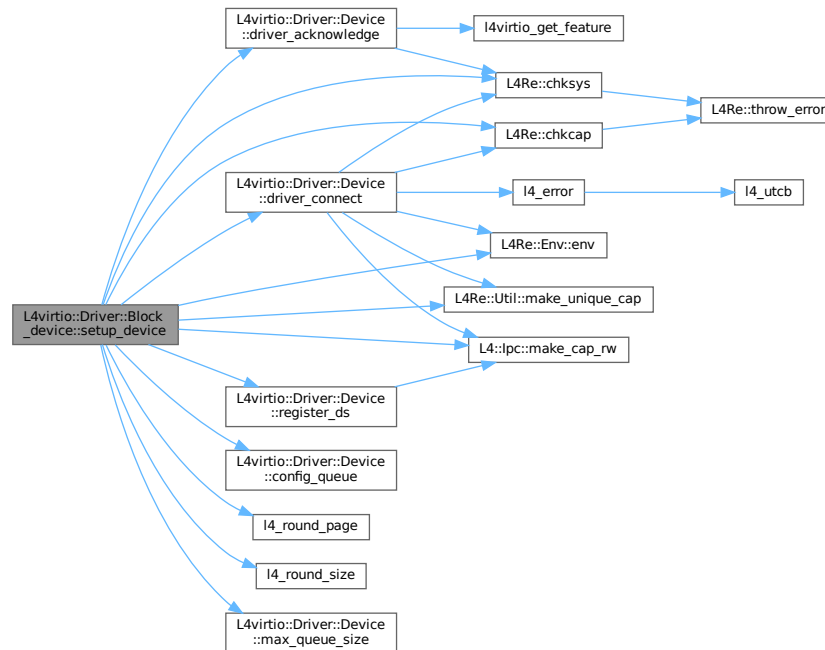
	<i>srvcap</i>	IPC capability of the channel to the server.
	<i>usermem</i>	Size of additional memory to share with device.
out	<i>userdata</i>	Pointer to the region of user-usable memory.
out	<i>user_devaddr</i>	Address of user-usable memory in device address space.
	<i>qds</i>	External queue dataspace. If this capability is invalid, the function will attempt to allocate a dataspace on its own. Note that the external queue dataspace must be large enough.
	<i>fmask0</i>	Feature bits 0..31 that the driver supports.
	<i>fmask1</i>	Feature bits 32..63 that the driver supports.

This function starts a handshake with the device and sets up the virtqueues for communication and the additional data structures for the block device. It will also allocate and share additional memory that the caller then can use freely, i.e. normally this memory would be used as a reception buffer. The caller may also decide to not make use of this convenience function and request 0 bytes in usermem. Then it has to allocate the block buffers for sending/receiving payload manually and share them using [register\\_ds\(\)](#).

Definition at line 92 of file [virtio-block](#).

References [L4Re::chkcap\(\)](#), [L4Re::chksys\(\)](#), [L4virtio::Driver::Device::config\\_queue\(\)](#), [L4Re::Mem\\_alloc::Continuous](#), [L4virtio::Driver::Device::driver\\_acknowledge\(\)](#), [L4virtio::Driver::Device::driver\\_connect\(\)](#), [L4Re::Env::env\(\)](#), [L4\\_EINVAL](#), [L4\\_ENODEV](#), [L4\\_PAGESHIFT](#), [l4\\_round\\_page\(\)](#), [l4\\_round\\_size\(\)](#), [L4VIRTIO\\_ID\\_BLOCK](#), [L4::lpc::make\\_cap\\_rw\(\)](#), [L4Re::Util::make\\_unique\\_cap\(\)](#), [L4virtio::Driver::Device::max\\_queue\\_size\(\)](#), [L4Re::Mem\\_alloc::Pinned](#), [L4virtio::Driver::Device::register\\_ds\(\)](#), [L4Re::Rm::F::RW](#), and [L4Re::Rm::F::Search\\_addr](#).

Here is the call graph for this function:



### 16.392.2.6 start\_request()

```

Handle L4virtio::Driver::Block_device::start_request (
    14_uint64_t sector,
    14_uint32_t type,
    Callback callback) [inline]
  
```

Start the setup of a new request.

#### Parameters

<i>sector</i>	First sector to write to/read from.
<i>type</i>	Request type.
<i>callback</i>	Function to call, when the request is finished. May be 0 for synchronous requests.

Definition at line 191 of file [virtio-block](#).

References [L4virtio::Virtqueue::Desc::addr](#), [L4virtio::Virtqueue::Desc::flags](#), [I4virtio\\_block\\_header\\_t::ioprio](#), [L4virtio::Virtqueue::Desc::len](#), [L4virtio::Virtqueue::Desc::Flags::raw](#), [I4virtio\\_block\\_header\\_t::sector](#), and [I4virtio\\_block\\_header\\_t::type](#)

The documentation for this class was generated from the following file:

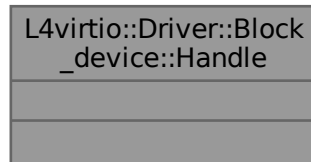
- [I4/I4virtio/client/virtio-block](#)

## 16.393 L4virtio::Driver::Block\_device::Handle Class Reference

[Handle](#) to an ongoing request.

```
#include <virtio-block>
```

Collaboration diagram for L4virtio::Driver::Block\_device::Handle:



### 16.393.1 Detailed Description

[Handle](#) to an ongoing request.

Definition at line 56 of file [virtio-block](#).

The documentation for this class was generated from the following file:

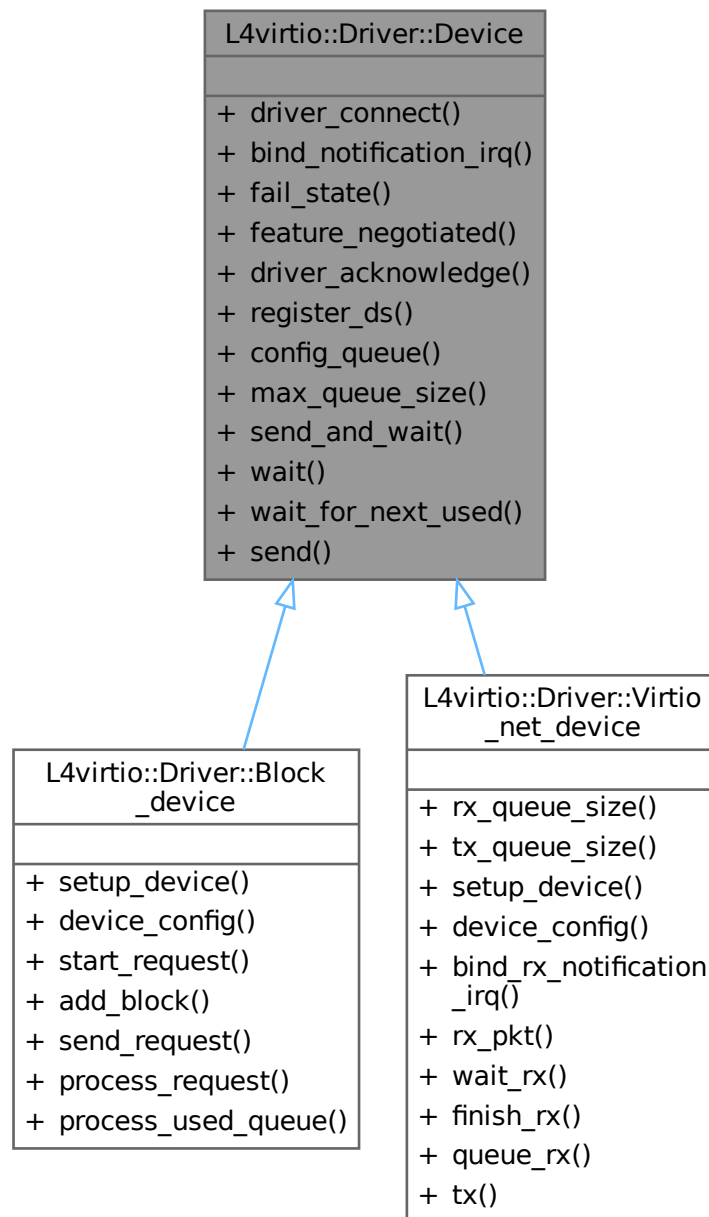
- l4/l4virtio/client/virtio-block

## 16.394 L4virtio::Driver::Device Class Reference

Client-side implementation for a general virtio device.

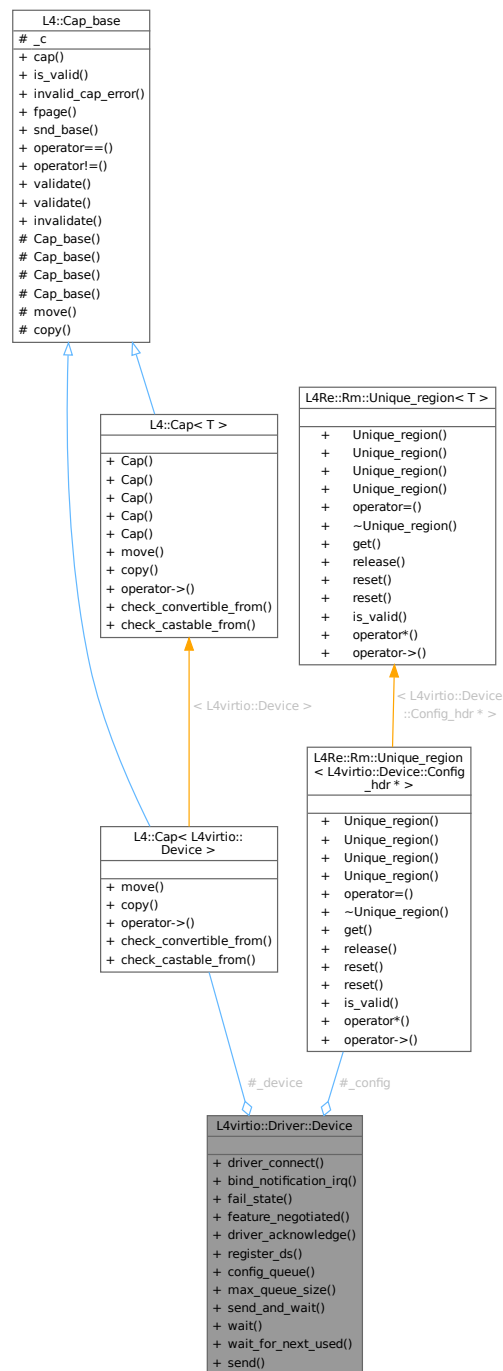
```
#include <l4virtio>
```

Inheritance diagram for L4virtio::Driver::Device:





Collaboration diagram for L4virtio::Driver::Device:



## Public Member Functions

- void `driver_connect` (`L4::Cap< L4virtio::Device >` `srvcap`, bool `manage_notify=true`)  
Contacts the device and starts the initial handshake.
- int `bind_notification_irq` (unsigned index, `L4::Cap< L4::Triggerable >` `irq`) const  
Register a triggerable to receive notifications from the device.
- bool `fail_state` () const

- Return true if the device is in a fail state.*

  - bool `feature_negotiated` (unsigned int feat) const

*Check if a particular feature bit was negotiated with the device.*
- int `driver_acknowledge` ()

*Finalize handshake with the device.*
- int `register_ds` (L4::Cap< L4Re::Dataspace > ds, l4\_umword\_t offset, l4\_umword\_t size, l4\_uint64\_t \*devaddr)

*Share a dataspace with the device.*
- int `config_queue` (int num, unsigned size, l4\_uint64\_t desc\_addr, l4\_uint64\_t avail\_addr, l4\_uint64\_t used\_addr)

*Send the virtqueue configuration to the device.*
- int `max_queue_size` (int num) const

*Maximum queue size allowed by the device.*
- int `send_and_wait` (Virtqueue &queue, l4\_uint16\_t descno)

*Send a request to the device and wait for it to be processed.*
- int `wait` (int index) const

*Wait for a notification from the device.*
- int `wait_for_next_used` (Virtqueue &queue, l4\_uint32\_t \*len=NULLPTR) const

*Wait for the next item to arrive in the used queue and return it.*
- void `send` (Virtqueue &queue, l4\_uint16\_t descno)

*Send a request to the device.*

## 16.394.1 Detailed Description

Client-side implementation for a general virtio device.

Definition at line 31 of file `l4virtio`.

## 16.394.2 Member Function Documentation

### 16.394.2.1 bind\_notification\_irq()

```
int L4virtio::Driver::Device::bind_notification_irq (
    unsigned index,
    L4::Cap< L4::Triggerable > irq) const [inline]
```

Register a triggerable to receive notifications from the device.

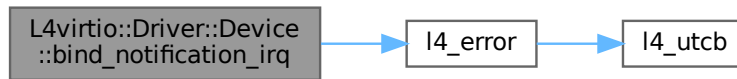
#### Parameters

	<i>index</i>	Index of the interrupt.
out	<i>irq</i>	Triggerable to register for notifications.

Definition at line 129 of file `l4virtio`.

References `l4_error()`.

Here is the call graph for this function:



### 16.394.2.2 config\_queue()

```

int L4virtio::Driver::Device::config_queue (
    int num,
    unsigned size,
    l4_uint64_t desc_addr,
    l4_uint64_t avail_addr,
    l4_uint64_t used_addr) [inline]
  
```

Send the virtqueue configuration to the device.

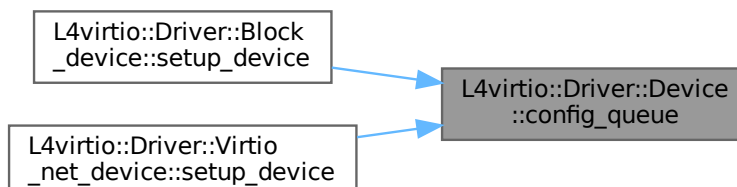
#### Parameters

<i>num</i>	Number of queue to configure.
<i>size</i>	Size of rings in the queue, must be a power of 2)
<i>desc_addr</i>	Address of descriptor table (device address)
<i>avail_addr</i>	Address of available ring (device address)
<i>used_addr</i>	Address of used ring (device address)

Definition at line 212 of file [l4virtio](#).

Referenced by [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#), and [L4virtio::Driver::Virtio\\_net\\_device::setup\\_device\(\)](#).

Here is the caller graph for this function:



### 16.394.2.3 driver\_acknowledge()

```
int L4virtio::Driver::Device::driver_acknowledge () [inline]
```

Finalize handshake with the device.

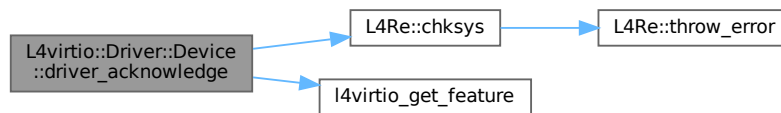
Must be called after all queues have been set up and before the first request is sent. It is still possible to add more shared dataspace after the handshake has been finished.

Definition at line 156 of file [l4virtio](#).

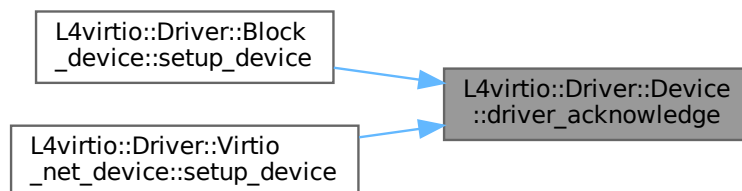
References [L4Re::chksys\(\)](#), [L4\\_EINVAL](#), [L4\\_EIO](#), [L4\\_ENODEV](#), [L4\\_EOK](#), [L4VIRTIO\\_FEATURE\\_VERSION\\_1](#), [l4virtio\\_get\\_feature\(\)](#), [L4VIRTIO\\_STATUS\\_DRIVER\\_OK](#), and [L4VIRTIO\\_STATUS\\_FEATURES\\_OK](#).

Referenced by [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#), and [L4virtio::Driver::Virtio\\_net\\_device::setup\\_device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.394.2.4 driver\_connect()

```
void L4virtio::Driver::Device::driver_connect (
    L4::Cap< L4virtio::Device > srvcap,
    bool manage_notify = true) [inline]
```

Contacts the device and starts the initial handshake.

#### Parameters

---

<i>srvcap</i>	Capability for device communication.
<i>manage_notify</i>	Set up a semaphore for notifications from the device. See below.

### Exceptions

<a href="#">L4::Runtime_error</a>	if the initialisation fails
-----------------------------------	-----------------------------

This function contacts the server, sets up the notification channels and the configuration dataspace. After this is done, the caller can set up any dataspaces it needs. The initialisation then needs to be finished by calling [driver\\_acknowledge\(\)](#).

Per default this function creates and registers a semaphore for receiving notification from the device. This semaphore is used in the blocking functions [send\\_and\\_wait\(\)](#), [wait\(\)](#) and [next\\_used\(\)](#).

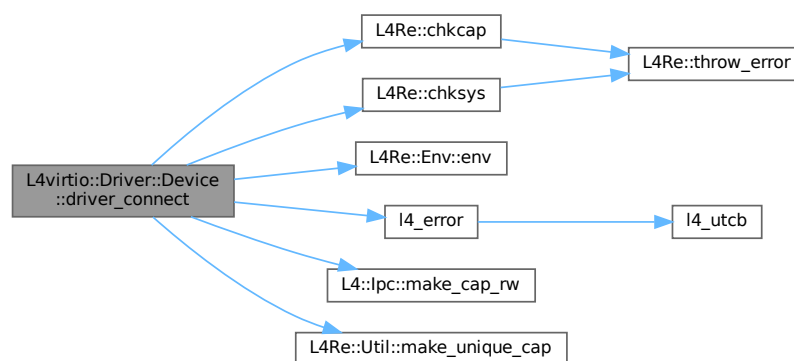
When `manage_notify` is false, then the caller may manually register and handle notification interrupts from the device. This is for example useful, when the client runs in an application with a server loop.

Definition at line 56 of file [l4virtio](#).

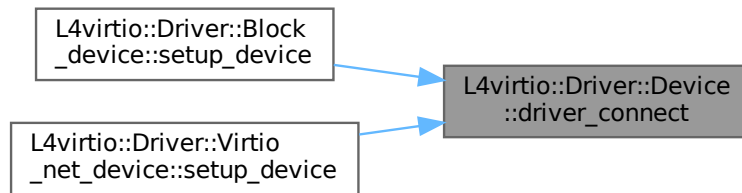
References [L4Re::chkcap\(\)](#), [L4Re::chksys\(\)](#), [L4Re::Env::env\(\)](#), [L4\\_EINVAL](#), [L4\\_EIO](#), [L4\\_ENODEV](#), [l4\\_error\(\)](#), [L4\\_PAGEMASK](#), [L4\\_PAGESHIFT](#), [L4\\_PAGESIZE](#), [L4\\_SUPERPAGESIZE](#), [L4VIRTIO\\_STATUS\\_ACKNOWLEDGE](#), [L4VIRTIO\\_STATUS\\_DRIVER](#), [L4::lpc::make\\_cap\\_rw\(\)](#), [L4Re::Util::make\\_unique\\_cap\(\)](#), [L4Re::Rm::F::RW](#), and [L4Re::Rm::F::Search\\_addr](#).

Referenced by [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#), and [L4virtio::Driver::Virtio\\_net\\_device::setup\\_device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.394.2.5 feature\_negotiated()

```
bool L4virtio::Driver::Device::feature_negotiated (
    unsigned int feat) const [inline]
```

Check if a particular feature bit was negotiated with the device.

The result is only valid after [driver\\_acknowledge\(\)](#) was called (when the handshake with the device was completed).

#### Parameters

<i>feat</i>	The feature bit.
-------------	------------------

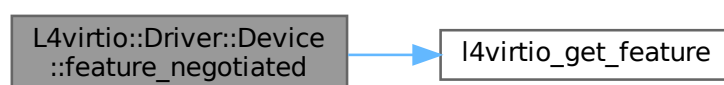
#### Return values

<i>true</i>	The feature is supported by both driver and device.
<i>false</i>	The feature is not supported by the driver and/or device.

Definition at line [145](#) of file [l4virtio](#).

References [l4virtio\\_get\\_feature\(\)](#).

Here is the call graph for this function:



### 16.394.2.6 max\_queue\_size()

```
int L4virtio::Driver::Device::max_queue_size (
    int num) const [inline]
```

Maximum queue size allowed by the device.

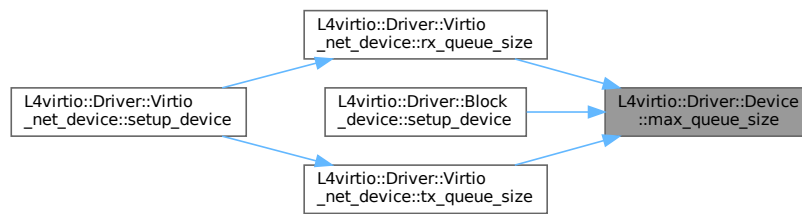
#### Parameters

<i>num</i>	Number of queue for which to determine the maximum size.
------------	----------------------------------------------------------

Definition at line 230 of file [l4virtio](#).

Referenced by [L4virtio::Driver::Virtio\\_net\\_device::rx\\_queue\\_size\(\)](#), [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#), and [L4virtio::Driver::Virtio\\_net\\_device::tx\\_queue\\_size\(\)](#).

Here is the caller graph for this function:



### 16.394.2.7 register\_ds()

```
int L4virtio::Driver::Device::register_ds (
    L4::Cap< L4Re::Dataspace > ds,
    l4_umword_t offset,
    l4_umword_t size,
    l4_uint64_t * devaddr) [inline]
```

Share a dataspace with the device.

#### Parameters

<i>ds</i>	Dataspace to share with the device.
<i>offset</i>	Offset in dataspace where the shared part starts.
<i>size</i>	Total size in bytes of the shared space.
<i>devaddr</i>	Start of shared space in the device address space.

Although this function allows to share only a part of the given dataspace for convenience, the granularity of sharing is always the dataspace level. Thus, the remainder of the dataspace is not protected from the device.

When communicating with the device, addresses must be given with respect to the device address space. This is not the same as the virtual address space of the client in order to not leak information about the address space layout.

Definition at line 196 of file [l4virtio](#).

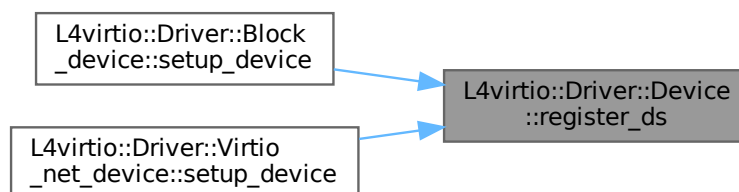
References [L4::lpc::make\\_cap\\_rw\(\)](#).

Referenced by [L4virtio::Driver::Block\\_device::setup\\_device\(\)](#), and [L4virtio::Driver::Virtio\\_net\\_device::setup\\_device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.394.2.8 send()

```
void L4virtio::Driver::Device::send (
    Virtqueue & queue,
    l4_uint16_t descno) [inline]
```

Send a request to the device.

#### Parameters

<i>queue</i>	Queue that contains the request in its descriptor table
<i>descno</i>	Index of first entry in descriptor table where

Definition at line 312 of file [l4virtio](#).



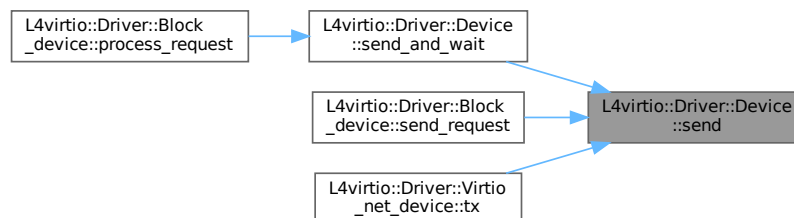
References [L4virtio::Driver::Virtqueue::enqueue\\_descriptor\(\)](#).

Referenced by [send\\_and\\_wait\(\)](#), [L4virtio::Driver::Block\\_device::send\\_request\(\)](#), and [L4virtio::Driver::Virtio\\_net\\_device::tx\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.394.2.9 send\_and\_wait()

```

int L4virtio::Driver::Device::send_and_wait (
    Virtqueue & queue,
    l4_uint16_t descno) [inline]
  
```

Send a request to the device and wait for it to be processed.

#### Parameters

<i>queue</i>	Queue that contains the request in its descriptor table
<i>descno</i>	Index of first entry in descriptor table where

This function provides a simple mechanism to send requests synchronously. It must not be used with other requests at the same time as it directly waits for a notification on the device irq cap.

**Precondition**

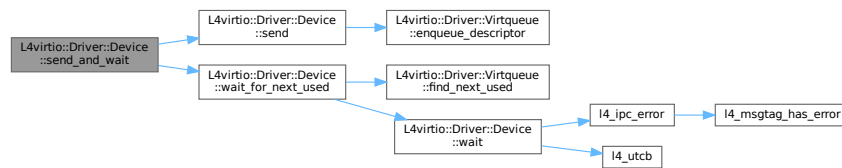
`driver_connect()` was called with `manage_notify`.

Definition at line 247 of file `l4virtio`.

References `L4_EINVAL`, `L4_EOK`, `send()`, and `wait_for_next_used()`.

Referenced by `L4virtio::Driver::Block_device::process_request()`.

Here is the call graph for this function:



Here is the caller graph for this function:

**16.394.2.10 wait()**

```
int L4virtio::Driver::Device::wait (
    int index) const [inline]
```

Wait for a notification from the device.

**Parameters**

<i>index</i>	Notification slot to wait for.
--------------	--------------------------------

**Precondition**

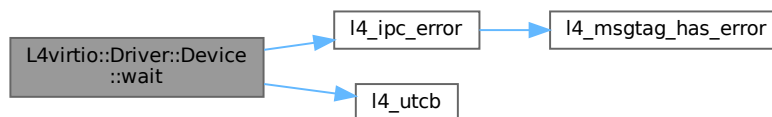
`driver_connect()` was called with `manage_notify`.

Definition at line 268 of file `l4virtio`.

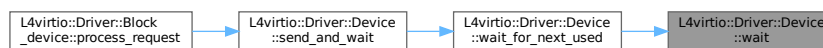
References `L4_EEXIST`, `l4_ipc_error()`, and `l4_utcb()`.

Referenced by `wait_for_next_used()`.

Here is the call graph for this function:



Here is the caller graph for this function:

**16.394.2.11 wait\_for\_next\_used()**

```

int L4virtio::Driver::Device::wait_for_next_used (
    Virtqueue & queue,
    l4_uint32_t * len = nullptr) const [inline]
  
```

Wait for the next item to arrive in the used queue and return it.

**Parameters**

	<i>queue</i>	A queue.
out	<i>len</i>	(optional) Size of valid data in finished block. Note that this is the value reported by the device, which may set it to a value that is larger than the original buffer size.

**Return values**

$\geq 0$	Descriptor number of item removed from used queue.
$< 0$	IPC error while waiting for notification.

The call blocks until the next item is available in the used queue.

**Precondition**

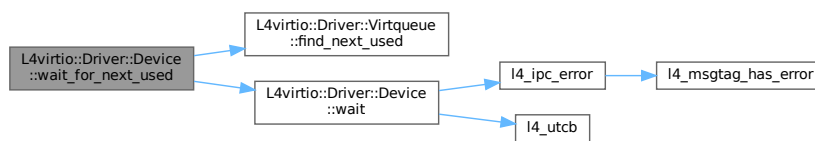
[driver\\_connect\(\)](#) was called with `manage_notify`.

Definition at line 291 of file [l4virtio](#).

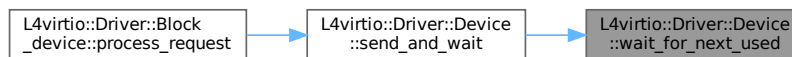
References [L4virtio::Driver::Virtqueue::find\\_next\\_used\(\)](#), and [wait\(\)](#).

Referenced by [send\\_and\\_wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

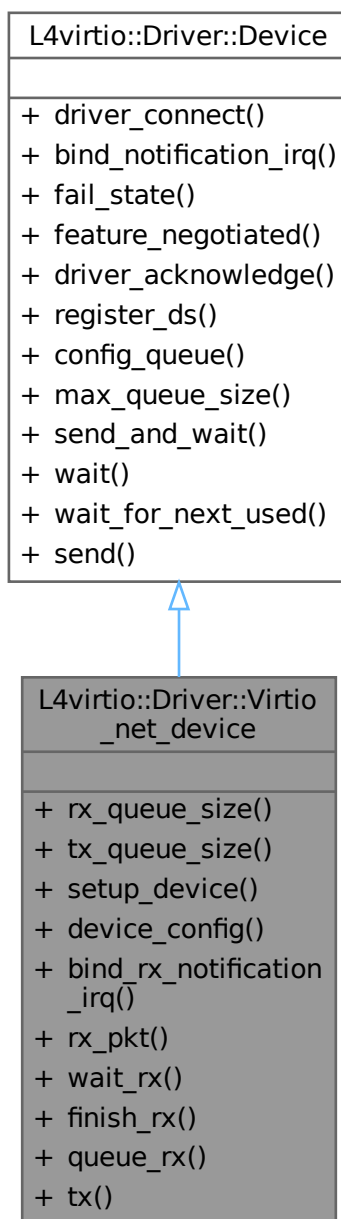
- `I4/I4virtio/client/I4virtio`

## 16.395 L4virtio::Driver::Virtio\_net\_device Class Reference

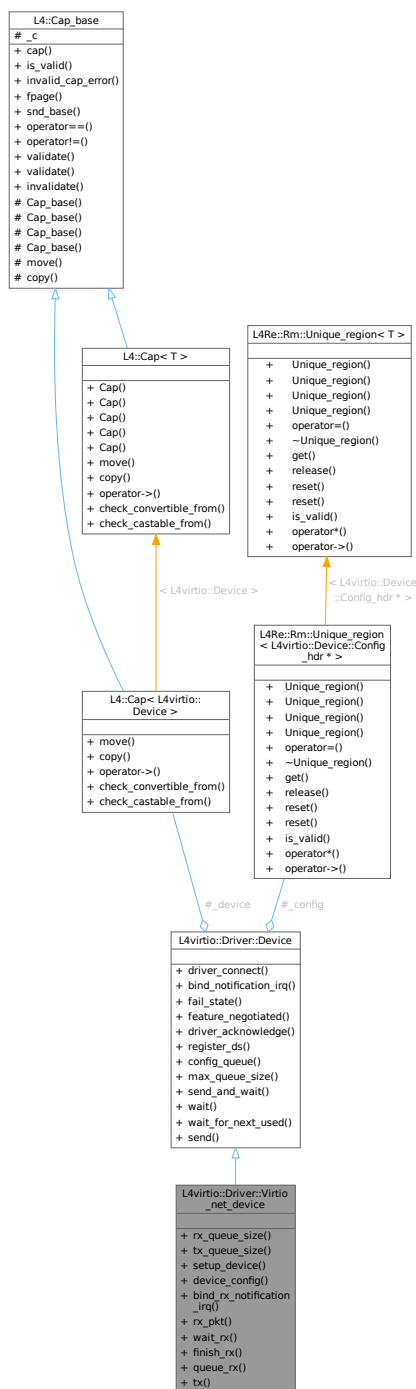
Simple class for accessing a virtio net device.

```
#include <virtio-net>
```

Inheritance diagram for L4virtio::Driver::Virtio\_net\_device:



Collaboration diagram for L4virtio::Driver::Virtio\_net\_device:



## Data Structures

- struct [Packet](#)

Structure for a network packet (header including data) with maximum size, assuming that no extra features have been negotiated.

## Public Member Functions

- int [rx\\_queue\\_size](#) () const  
*Return the maximum receive queue size allowed by the device.*
- int [tx\\_queue\\_size](#) () const  
*Return the maximum transmit queue size allowed by the device.*
- void [setup\\_device](#) (L4::Cap< L4virtio::Device > srvcap)  
*Establish a connection to the device and set up shared memory.*
- [l4virtio\\_net\\_config\\_t](#) const & [device\\_config](#) () const  
*Return a reference to the device configuration.*
- int [bind\\_rx\\_notification\\_irq](#) (L4::Cap< L4::Thread > thread, [l4\\_umword\\_t](#) label)  
*Bind the rx notification IRQ to the specified thread.*
- [Packet](#) & [rx\\_pkt](#) ([l4\\_uint16\\_t](#) descno)  
*Return a reference to the RX packet buffer of the specified descriptor, e.g.*
- [l4\\_uint16\\_t](#) [wait\\_rx](#) ([l4\\_uint32\\_t](#) \*len=nullptr)  
*Block until a network packet has been received from the device and return the descriptor number.*
- void [finish\\_rx](#) ([l4\\_uint16\\_t](#) descno)  
*Free an RX descriptor number to make it available for the RX queue again.*
- void [queue\\_rx](#) ()  
*Queue new available descriptors in the RX queue.*
- bool [tx](#) (std::function< [l4\\_uint32\\_t](#)([Packet](#) &)> prepare)  
*Attempt to allocate a descriptor in the TX queue and transmit the packet, after calling the prepare callback.*

## Public Member Functions inherited from L4virtio::Driver::Device

- void [driver\\_connect](#) (L4::Cap< L4virtio::Device > srvcap, bool manage\_notify=true)  
*Contacts the device and starts the initial handshake.*
- int [bind\\_notification\\_irq](#) (unsigned index, L4::Cap< L4::Triggerable > irq) const  
*Register a triggerable to receive notifications from the device.*
- bool [fail\\_state](#) () const  
*Return true if the device is in a fail state.*
- bool [feature\\_negotiated](#) (unsigned int feat) const  
*Check if a particular feature bit was negotiated with the device.*
- int [driver\\_acknowledge](#) ()  
*Finalize handshake with the device.*
- int [register\\_ds](#) (L4::Cap< L4Re::Dataspace > ds, [l4\\_umword\\_t](#) offset, [l4\\_umword\\_t](#) size, [l4\\_uint64\\_t](#) \*devaddr)  
*Share a dataspace with the device.*
- int [config\\_queue](#) (int num, unsigned size, [l4\\_uint64\\_t](#) desc\_addr, [l4\\_uint64\\_t](#) avail\_addr, [l4\\_uint64\\_t](#) used\_↔ addr)  
*Send the virtqueue configuration to the device.*
- int [max\\_queue\\_size](#) (int num) const  
*Maximum queue size allowed by the device.*
- int [send\\_and\\_wait](#) ([Virtqueue](#) &queue, [l4\\_uint16\\_t](#) descno)  
*Send a request to the device and wait for it to be processed.*
- int [wait](#) (int index) const  
*Wait for a notification from the device.*
- int [wait\\_for\\_next\\_used](#) ([Virtqueue](#) &queue, [l4\\_uint32\\_t](#) \*len=nullptr) const  
*Wait for the next item to arrive in the used queue and return it.*
- void [send](#) ([Virtqueue](#) &queue, [l4\\_uint16\\_t](#) descno)  
*Send a request to the device.*

### 16.395.1 Detailed Description

Simple class for accessing a virtio net device.

Definition at line 30 of file [virtio-net](#).

### 16.395.2 Member Function Documentation

#### 16.395.2.1 `bind_rx_notification_irq()`

```
int L4virtio::Driver::Virtio_net_device::bind_rx_notification_irq (
    L4::Cap< L4::Thread > thread,
    l4_umword_t label) [inline]
```

Bind the rx notification IRQ to the specified thread.

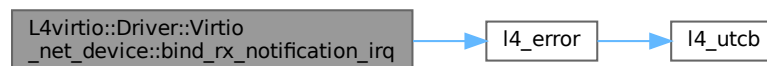
##### Parameters

<i>thread</i>	Thread to bind the notification IRQ to.
<i>label</i>	Label to assign to the IRQ.

Definition at line 169 of file [virtio-net](#).

References [l4\\_error\(\)](#).

Here is the call graph for this function:



#### 16.395.2.2 `finish_rx()`

```
void L4virtio::Driver::Virtio_net_device::finish_rx (
    l4_uint16_t descno) [inline]
```

Free an RX descriptor number to make it available for the RX queue again.

##### Parameters

<i>descno</i>	Descriptor number in the virtio queue.
---------------	----------------------------------------

Usually [queue\\_rx\(\)](#) should be called afterwards to queue the freed descriptor(s).

Definition at line 230 of file [virtio-net](#).



### 16.395.2.3 rx\_pkt()

```
Packet & L4virtio::Driver::Virtio_net_device::rx_pkt (
    l4_uint16_t descno) [inline]
```

Return a reference to the RX packet buffer of the specified descriptor, e.g.

from [wait\\_rx\(\)](#).

#### Parameters

<i>descno</i>	Descriptor number in the virtio queue.
---------------	----------------------------------------

Definition at line 180 of file [virtio-net](#).

### 16.395.2.4 rx\_queue\_size()

```
int L4virtio::Driver::Virtio_net_device::rx_queue_size () const [inline]
```

Return the maximum receive queue size allowed by the device.

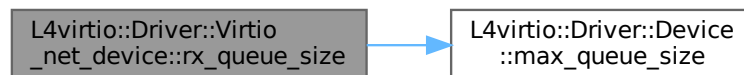
[wait\\_rx\(\)](#) will return a descriptor number that is smaller than this size.

Definition at line 47 of file [virtio-net](#).

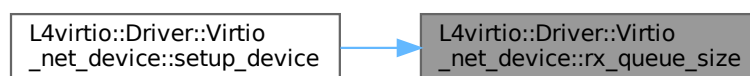
References [L4virtio::Driver::Device::max\\_queue\\_size\(\)](#).

Referenced by [setup\\_device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.395.2.5 setup\_device()

```
void L4virtio::Driver::Virtio_net_device::setup_device (  
    L4::Cap< L4virtio::Device > srvcap) [inline]
```

Establish a connection to the device and set up shared memory.

#### Parameters

---

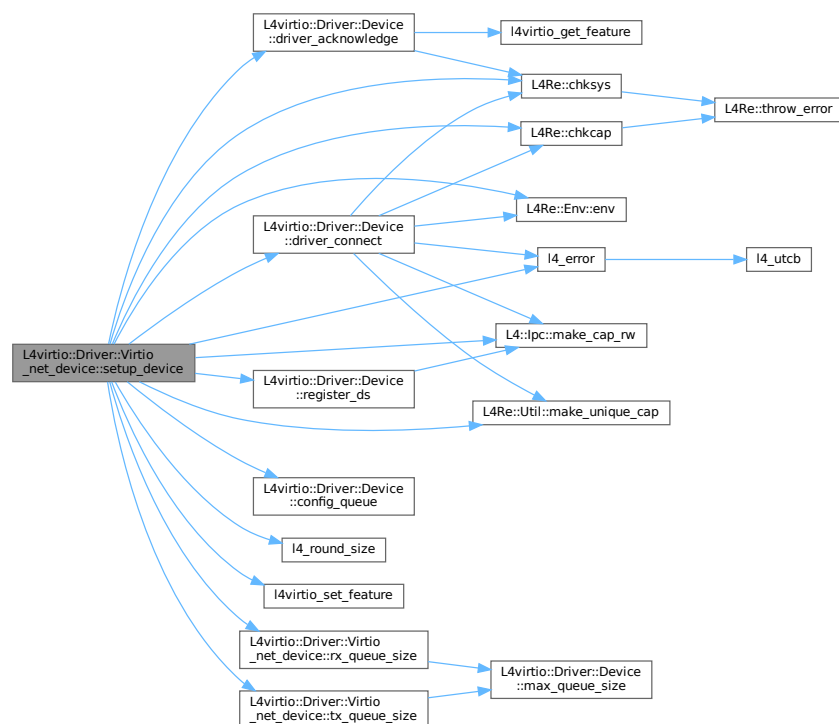
<i>srvcap</i>	IPC capability of the channel to the server.
---------------	----------------------------------------------

This function starts a handshake with the device and sets up the virtqueues for communication and the additional data structures for the network device.

Definition at line 66 of file [virtio-net](#).

References [L4Re::chkcap\(\)](#), [L4Re::chksys\(\)](#), [L4virtio::Driver::Device::config\\_queue\(\)](#), [L4Re::Mem\\_alloc::Continuous](#), [L4virtio::Driver::Device::driver\\_acknowledge\(\)](#), [L4virtio::Driver::Device::driver\\_connect\(\)](#), [L4Re::Env::env\(\)](#), [L4\\_EINVAL](#), [L4\\_ENODEV](#), [l4\\_error\(\)](#), [L4\\_PAGESHIFT](#), [l4\\_round\\_size\(\)](#), [L4VIRTIO\\_FEATURE\\_VERSION\\_1](#), [L4VIRTIO\\_ID\\_NET](#), [l4virtio\\_set\\_feature\(\)](#), [L4::lpc::make\\_cap\\_rw\(\)](#), [L4Re::Util::make\\_unique\\_cap\(\)](#), [L4Re::Mem\\_alloc::Pinned](#), [L4virtio::Driver::Device::register\\_ds\(\)](#), [L4Re::Rm::F::RW](#), [rx\\_queue\\_size\(\)](#), [L4Re::Rm::F::Search\\_addr](#), and [tx\\_queue\\_size\(\)](#).

Here is the call graph for this function:



### 16.395.2.6 tx()

```
bool L4virtio::Driver::Virtio_net_device::tx (
    std::function< l4_uint32_t(Packet &)> prepare) [inline]
```

Attempt to allocate a descriptor in the TX queue and transmit the packet, after calling the prepare callback.

#### Parameters

<i>prepare</i>	Function that fills the packet with data, should return the length of the data copied to the packet.
----------------	------------------------------------------------------------------------------------------------------

### Return values

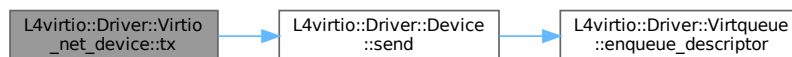
<i>true</i>	The packet was queued.
<i>false</i>	TX queue is full.

The prepare callback should fill the packet with data and return the length of the packet data (without the size of the virtio-net packet header).

Definition at line 260 of file [virtio-net](#).

References [L4virtio::Driver::Device::send\(\)](#).

Here is the call graph for this function:



### 16.395.2.7 tx\_queue\_size()

```
int L4virtio::Driver::Virtio_net_device::tx_queue_size () const [inline]
```

Return the maximum transmit queue size allowed by the device.

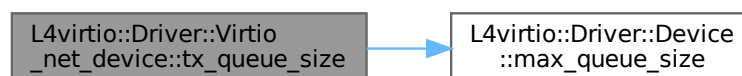
[tx\(\)](#) will fail if the amount of queued packets exceeds this size.

Definition at line 54 of file [virtio-net](#).

References [L4virtio::Driver::Device::max\\_queue\\_size\(\)](#).

Referenced by [setup\\_device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.395.2.8 wait\_rx()

```

14_uint16_t L4virtio::Driver::Virtio_net_device::wait_rx (
    14_uint32_t * len = nullptr) [inline]
  
```

Block until a network packet has been received from the device and return the descriptor number.

#### Precondition

The calling thread must be bound to the rx notification IRQ via `bind_rx_notification_irq()`.

#### Parameters

out	len	(optional) Length of valid data in RX packet.
-----	-----	-----------------------------------------------

#### Returns

Descriptor number of received packet.

The packet data can be obtained with `rx_pkt()`. `finish_rx()` should be called after the packet buffer can be returned to the RX queue.

Definition at line 202 of file `virtio-net`.

References `L4Re::chksys()`.

Here is the call graph for this function:



The documentation for this class was generated from the following file:

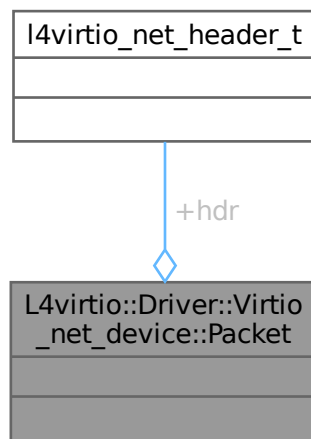
- `l4/l4virtio/client/virtio-net`

## 16.396 L4virtio::Driver::Virtio\_net\_device::Packet Struct Reference

Structure for a network packet (header including data) with maximum size, assuming that no extra features have been negotiated.

```
#include <virtio-net>
```

Collaboration diagram for L4virtio::Driver::Virtio\_net\_device::Packet:



### 16.396.1 Detailed Description

Structure for a network packet (header including data) with maximum size, assuming that no extra features have been negotiated.

Definition at line 37 of file [virtio-net](#).

The documentation for this struct was generated from the following file:

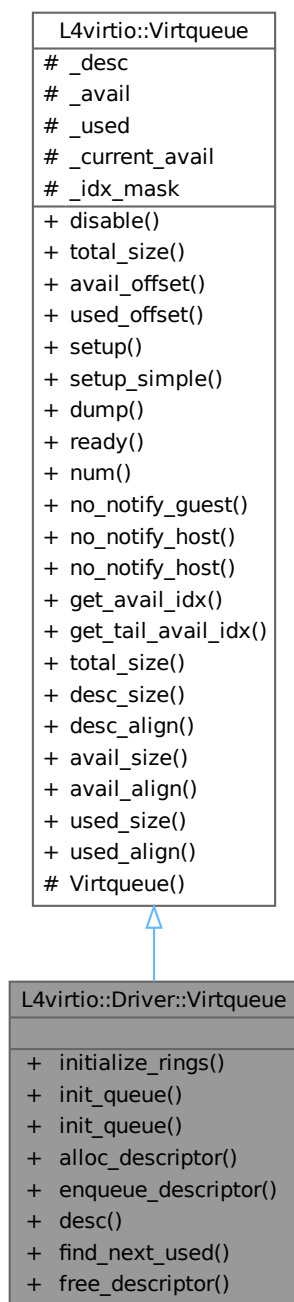
- `I4/I4virtio/client/virtio-net`

## 16.397 L4virtio::Driver::Virtqueue Class Reference

Driver-side implementation of a [Virtqueue](#).

```
#include <virtqueue>
```

Inheritance diagram for L4virtio::Driver::Virtqueue:







- Initialize this virtqueue.*
- [l4\\_uint16\\_t alloc\\_descriptor](#) ()  
*Allocate and return an unused descriptor from the descriptor table.*
- void [enqueue\\_descriptor](#) ([l4\\_uint16\\_t](#) descno)  
*Enqueue a descriptor in the available ring.*
- [Desc & desc](#) ([l4\\_uint16\\_t](#) descno)  
*Return a reference to a descriptor in the descriptor table.*
- [l4\\_uint16\\_t find\\_next\\_used](#) ([l4\\_uint32\\_t](#) \*len=nullptr)  
*Return the next finished block.*
- void [free\\_descriptor](#) ([l4\\_uint16\\_t](#) head, [l4\\_uint16\\_t](#) tail)  
*Free a chained list of descriptors in the descriptor queue.*

## Public Member Functions inherited from [L4virtio::Virtqueue](#)

- void [disable](#) ()  
*Completely disable the queue.*
- unsigned long [total\\_size](#) () const  
*Calculate the total size of this virtqueue.*
- unsigned long [avail\\_offset](#) () const  
*Get the offset of the available ring from the descriptor table.*
- unsigned long [used\\_offset](#) () const  
*Get the offset of the used ring from the descriptor table.*
- void [setup](#) (unsigned [num](#), void \*desc, void \*avail, void \*used)  
*Enable this queue.*
- void [setup\\_simple](#) (unsigned [num](#), void \*ring)  
*Enable this queue.*
- void [dump](#) ([Desc](#) const \*d) const  
*Dump descriptors for this queue.*
- bool [ready](#) () const  
*Test if this queue is in working state.*
- unsigned [num](#) () const
- bool [no\\_notify\\_guest](#) () const  
*Get the no IRQ flag of this queue.*
- bool [no\\_notify\\_host](#) () const  
*Get the no notify flag of this queue.*
- void [no\\_notify\\_host](#) (bool value)  
*Set the no-notify flag for this queue.*
- [l4\\_uint16\\_t](#) [get\\_avail\\_idx](#) () const  
*Get available index from available ring (for debugging).*
- [l4\\_uint16\\_t](#) [get\\_tail\\_avail\\_idx](#) () const  
*Get tail-available index stored in local state (for debugging).*

## Additional Inherited Members

## Public Types inherited from [L4virtio::Virtqueue](#)

- enum  
*Fixed alignment values for different parts of a virtqueue.*

## Static Public Member Functions inherited from [L4virtio::Virtqueue](#)

- static unsigned long [total\\_size](#) (unsigned [num](#))  
*Calculate the total size for a virtqueue of the given dimensions.*
- static unsigned long [desc\\_size](#) (unsigned [num](#))  
*Calculate the size of the descriptor table for [num](#) entries.*
- static unsigned long [desc\\_align](#) ()  
*Get the alignment in zero LSBs needed for the descriptor table.*
- static unsigned long [avail\\_size](#) (unsigned [num](#))  
*Calculate the size of the available ring for [num](#) entries.*
- static unsigned long [avail\\_align](#) ()  
*Get the alignment in zero LSBs needed for the available ring.*
- static unsigned long [used\\_size](#) (unsigned [num](#))  
*Calculate the size of the used ring for [num](#) entries.*
- static unsigned long [used\\_align](#) ()  
*Get the alignment in zero LSBs needed for the used ring.*

## Protected Member Functions inherited from [L4virtio::Virtqueue](#)

- [Virtqueue](#) ()=default  
*Create a disabled virtqueue.*

## Protected Attributes inherited from [L4virtio::Virtqueue](#)

- [Desc](#) \* [\\_desc](#) = nullptr  
*pointer to descriptor table, NULL if queue is off.*
- [Avail](#) \* [\\_avail](#) = nullptr  
*pointer to available ring.*
- [Used](#) \* [\\_used](#) = nullptr  
*pointer to used ring.*
- [l4\\_uint16\\_t](#) [\\_current\\_avail](#) = 0  
*The life counter for the queue.*
- [l4\\_uint16\\_t](#) [\\_idx\\_mask](#) = 0  
*mask used for indexing into the descriptor table and the rings.*

### 16.397.1 Detailed Description

Driver-side implementation of a [Virtqueue](#).

Adds function for managing the descriptor list, enqueueing new and dequeueing finished requests.

#### Note

The [Virtqueue](#) implementation is not thread-safe.

Definition at line 470 of file [virtqueue](#).

## 16.397.2 Member Function Documentation

### 16.397.2.1 alloc\_descriptor()

```
l4_uint16_t L4virtio::Driver::Virtqueue::alloc_descriptor () [inline]
```

Allocate and return an unused descriptor from the descriptor table.

The descriptor will be removed from the free list, the content should be considered undefined. After use, it needs to be freed using [free\\_descriptor\(\)](#).

#### Returns

The index of the reserved descriptor or Virtqueue::Eq if no free descriptor is available.

Note: the implementation uses  $(2^{16} - 1)$  as the end of queue marker. That means that the final entry in the queue can not be allocated iff the queue size is  $2^{16}$ .

Definition at line 558 of file [virtqueue](#).

References [L4virtio::Virtqueue::\\_desc](#).

### 16.397.2.2 desc()

```
Desc & L4virtio::Driver::Virtqueue::desc (
    l4_uint16_t descno) [inline]
```

Return a reference to a descriptor in the descriptor table.

#### Parameters

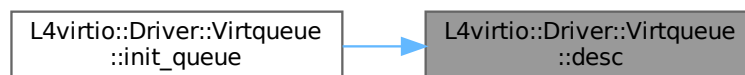
<i>descno</i>	Index of the descriptor, expected to be in correct range.
---------------	-----------------------------------------------------------

Definition at line 590 of file [virtqueue](#).

References [L4virtio::Virtqueue::\\_desc](#), and [L4virtio::Virtqueue::\\_idx\\_mask](#).

Referenced by [init\\_queue\(\)](#).

Here is the caller graph for this function:



### 16.397.2.3 enqueue\_descriptor()

```
void L4virtio::Driver::Virtqueue::enqueue_descriptor (
    l4_uint16_t descno) [inline]
```

Enqueue a descriptor in the available ring.

#### Parameters

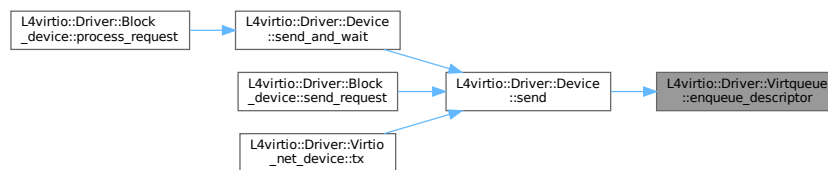
<i>descno</i>	Index of the head descriptor to enqueue.
---------------	------------------------------------------

Definition at line 574 of file [virtqueue](#).

References [L4virtio::Virtqueue::\\_avail](#), and [L4virtio::Virtqueue::\\_idx\\_mask](#).

Referenced by [L4virtio::Driver::Device::send\(\)](#).

Here is the caller graph for this function:



#### 16.397.2.4 find\_next\_used()

```

14_uint16_t L4virtio::Driver::Virtqueue::find_next_used (
    14_uint32_t * len = nullptr) [inline]
  
```

Return the next finished block.

##### Parameters

out	<i>len</i>	(optional) Size of valid data in finished block. Note that this is the value reported by the device, which may set it to a value that is larger than the original buffer size.
-----	------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

##### Returns

Index of the head or `Virtqueue::Eoq` if no used element is currently available.

Definition at line 609 of file [virtqueue](#).

References [L4virtio::Virtqueue::\\_current\\_avail](#), [L4virtio::Virtqueue::\\_idx\\_mask](#), and [L4virtio::Virtqueue::\\_used](#).

Referenced by [L4virtio::Driver::Device::wait\\_for\\_next\\_used\(\)](#).

Here is the caller graph for this function:



### 16.397.2.5 free\_descriptor()

```
void L4virtio::Driver::Virtqueue::free_descriptor (
    14_uint16_t head,
    14_uint16_t tail) [inline]
```

Free a chained list of descriptors in the descriptor queue.

#### Parameters

---

<i>head</i>	Index of the first element in the descriptor chain.
<i>tail</i>	Index of the last element in the descriptor chain.

Simply takes the descriptor chain and prepends it to the beginning of the free list. Assumes that the list has been correctly chained.

Definition at line 631 of file [virtqueue](#).

References [L4virtio::Virtqueue::\\_desc](#), and [L4virtio::Virtqueue::\\_idx\\_mask](#).

### 16.397.2.6 `init_queue()` [1/2]

```
void L4virtio::Driver::Virtqueue::init_queue (
    unsigned num,
    void * base) [inline]
```

Initialize this virtqueue.

#### Parameters

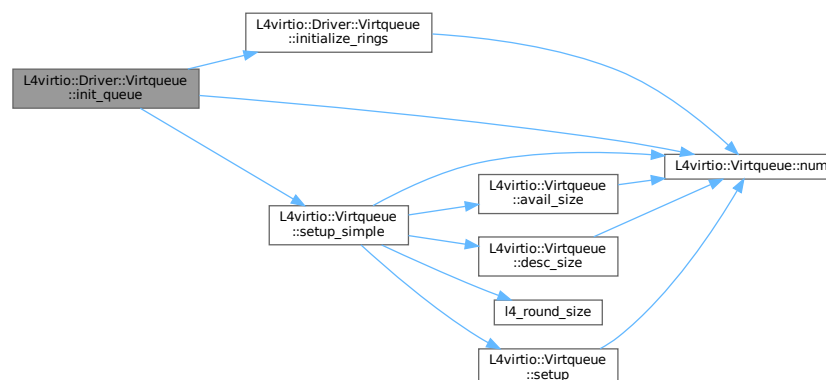
<i>num</i>	The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2).
<i>base</i>	The base address for the queue data structure.

This function sets up the memory and initializes the freelist.

Definition at line 537 of file [virtqueue](#).

References [initialize\\_rings\(\)](#), [L4virtio::Virtqueue::num\(\)](#), and [L4virtio::Virtqueue::setup\\_simple\(\)](#).

Here is the call graph for this function:



### 16.397.2.7 init\_queue() [2/2]

```
void L4virtio::Driver::Virtqueue::init_queue (
    unsigned num,
    void * desc,
    void * avail,
    void * used) [inline]
```

Initialize this virtqueue.

#### Parameters

---

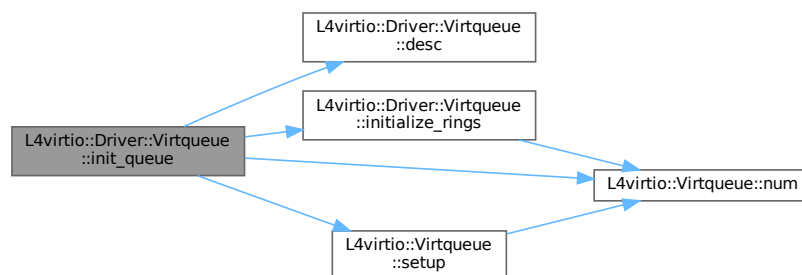
<i>num</i>	The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2).
<i>desc</i>	The address of the descriptor table. (Must be <code>Desc_align</code> aligned and at least <code>desc_size(num)</code> bytes in size.)
<i>avail</i>	The address of the available ring. (Must be <code>Avail_align</code> aligned and at least <code>avail_size(num)</code> bytes in size.)
<i>used</i>	The address of the used ring. (Must be <code>Used_align</code> aligned and at least <code>used_size(num)</code> bytes in size.)

This function sets up the memory and initializes the freelist.

Definition at line 522 of file [virtqueue](#).

References [desc\(\)](#), [initialize\\_rings\(\)](#), [L4virtio::Virtqueue::num\(\)](#), and [L4virtio::Virtqueue::setup\(\)](#).

Here is the call graph for this function:



### 16.397.2.8 initialize\_rings()

```
void L4virtio::Driver::Virtqueue::initialize_rings (
    unsigned num) [inline]
```

Initialize the descriptor table and the index structures of this queue.

#### Parameters

<i>num</i>	The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2).
------------	--------------------------------------------------------------------------------------------------------------



**Precondition**

The queue must be set up correctly with [setup\(\)](#) or [setup\\_simple\(\)](#).

Definition at line [494](#) of file [virtqueue](#).

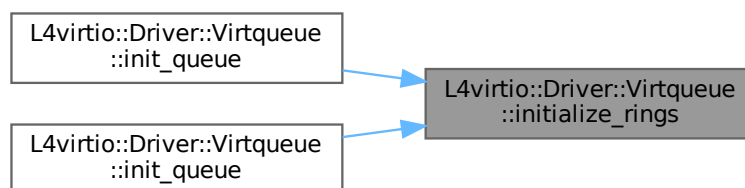
References [L4virtio::Virtqueue::\\_avail](#), [L4virtio::Virtqueue::\\_desc](#), [L4virtio::Virtqueue::\\_used](#), and [L4virtio::Virtqueue::num\(\)](#).

Referenced by [init\\_queue\(\)](#), and [init\\_queue\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

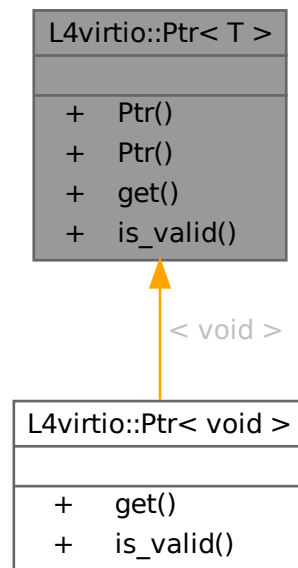
- [I4/I4virtio/virtqueue](#)

## 16.398 L4virtio::Ptr< T > Class Template Reference

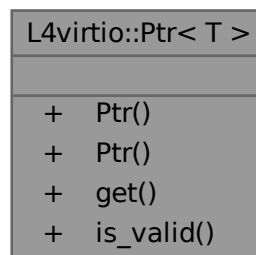
Pointer used in virtio descriptors.

```
#include <virtqueue>
```

Inheritance diagram for L4virtio::Ptr< T >:



Collaboration diagram for L4virtio::Ptr< T >:



## Public Types

- enum `Invalid_type` { `Invalid` }  
*Type for making an invalid (NULL) `Ptr`.*

## Public Member Functions

- `Ptr` (`Invalid_type`)

Make and invalid [Ptr](#).

- **Ptr** ([l4\\_uint64\\_t](#) vm\_addr)

Make a [Ptr](#) from a raw 64bit address.

- [l4\\_uint64\\_t](#) get () const
- bool [is\\_valid](#) () const

## 16.398.1 Detailed Description

```
template<typename T>
class L4virtio::Ptr< T >
```

Pointer used in virtio descriptors.

As the descriptor contain guest addresses these pointers cannot be dereferenced directly.

Definition at line 47 of file [virtqueue](#).

## 16.398.2 Member Enumeration Documentation

### 16.398.2.1 Invalid\_type

```
template<typename T>
enum L4virtio::Ptr::Invalid_type
```

Type for making an invalid (NULL) [Ptr](#).

#### Enumerator

Invalid	Use to set a <a href="#">Ptr</a> to invalid (NULL).
---------	-----------------------------------------------------

Definition at line 51 of file [virtqueue](#).

## 16.398.3 Member Function Documentation

### 16.398.3.1 get()

```
template<typename T>
l4_uint64_t L4virtio::Ptr< T >::get () const [inline]
```

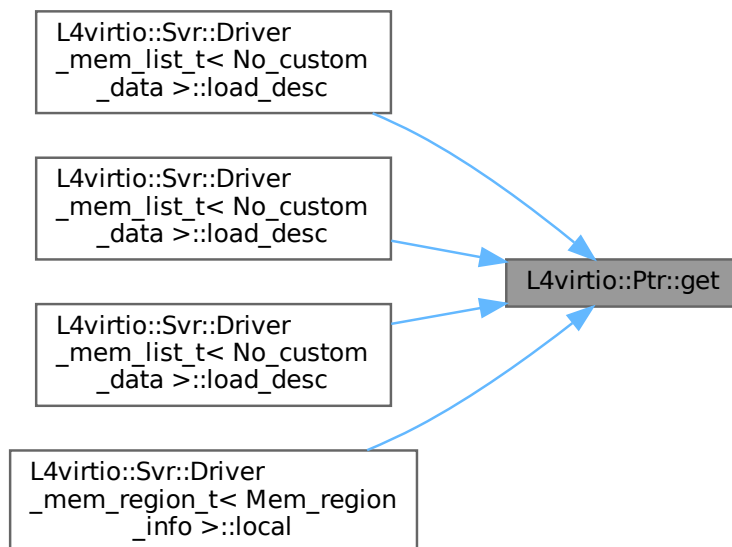
**Returns**

The raw 64bit address of the stored pointer.

Definition at line 62 of file [virtqueue](#).

Referenced by [L4virtio::Svr::Driver\\_mem\\_list\\_t< No\\_custom\\_data >::load\\_desc\(\)](#), [L4virtio::Svr::Driver\\_mem\\_list\\_t< No\\_custom\\_data >::load\\_desc\(\)](#), [L4virtio::Svr::Driver\\_mem\\_list\\_t< No\\_custom\\_data >::load\\_desc\(\)](#), and [L4virtio::Svr::Driver\\_mem\\_region\\_t< Mem\\_region\\_info >::local](#).

Here is the caller graph for this function:

**16.398.3.2 is\_valid()**

```
template<typename T>
bool L4virtio::Ptr< T >::is_valid () const [inline]
```

**Returns**

true if the stored pointer is valid (not NULL).

Definition at line 65 of file [virtqueue](#).

The documentation for this class was generated from the following file:

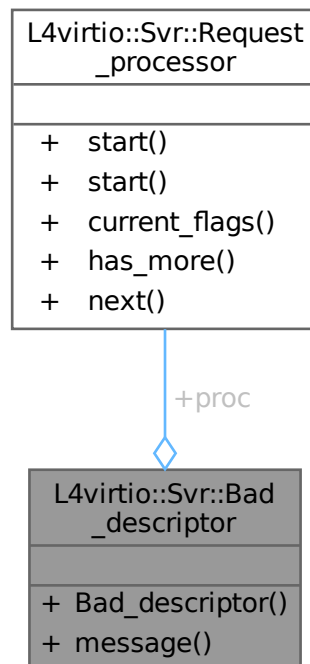
- `l4/l4virtio/virtqueue`

## 16.399 L4virtio::Svr::Bad\_descriptor Struct Reference

Exception used by Queue to indicate descriptor errors.

```
#include <virtio>
```

Collaboration diagram for L4virtio::Svr::Bad\_descriptor:



### Public Types

- enum [Error](#) {  
[Bad\\_address](#) , [Bad\\_rights](#) , [Bad\\_flags](#) , [Bad\\_next](#) ,  
[Bad\\_size](#) }  
*The error code.*

### Public Member Functions

- [Bad\\_descriptor](#) ([Request\\_processor](#) const \*[proc](#), [Error](#) e)  
*Make a bad descriptor exception.*
- char const \* [message](#) () const  
*Get a human readable description of the error code.*

### Data Fields

- [Request\\_processor](#) const \* **proc**  
*The processor that triggered the exception.*

### 16.399.1 Detailed Description

Exception used by Queue to indicate descriptor errors.

Definition at line 397 of file [virtio](#).

### 16.399.2 Member Enumeration Documentation

#### 16.399.2.1 Error

```
enum L4virtio::Svr::Bad_descriptor::Error
```

The error code.

##### Enumerator

Bad_address	Address cannot be translated.
Bad_rights	Missing access rights on memory.
Bad_flags	Invalid combination of descriptor flags.
Bad_next	Invalid next index.
Bad_size	Invalid size of memory block.

Definition at line 400 of file [virtio](#).

### 16.399.3 Constructor & Destructor Documentation

#### 16.399.3.1 Bad\_descriptor()

```
L4virtio::Svr::Bad_descriptor::Bad_descriptor (  
    Request_processor const * proc,  
    Error e) [inline]
```

Make a bad descriptor exception.

##### Parameters

<i>proc</i>	The request processor causing the exception
<i>e</i>	The error code.

Definition at line 421 of file [virtio](#).

References [proc](#).

## 16.399.4 Member Function Documentation

### 16.399.4.1 message()

```
char const * L4virtio::Svr::Bad_descriptor::message () const [inline]
```

Get a human readable description of the error code.

#### Returns

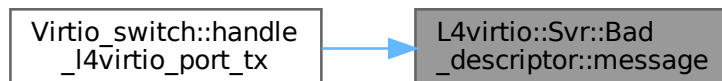
Message describing the error.

Definition at line 430 of file [virtio](#).

References [Bad\\_address](#), [Bad\\_flags](#), [Bad\\_next](#), [Bad\\_rights](#), and [Bad\\_size](#).

Referenced by [Virtio\\_switch::handle\\_l4virtio\\_port\\_tx\(\)](#).

Here is the caller graph for this function:



The documentation for this struct was generated from the following file:

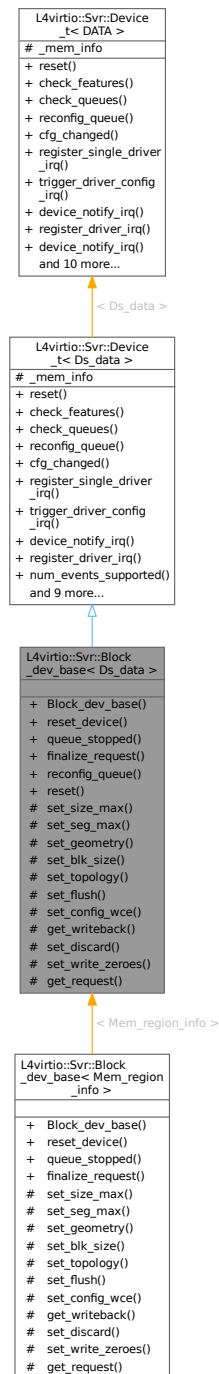
- [l4/l4virtio/server/virtio](#)

## 16.400 L4virtio::Svr::Block\_dev\_base< Ds\_data > Class Template Reference

Base class for virtio block devices.

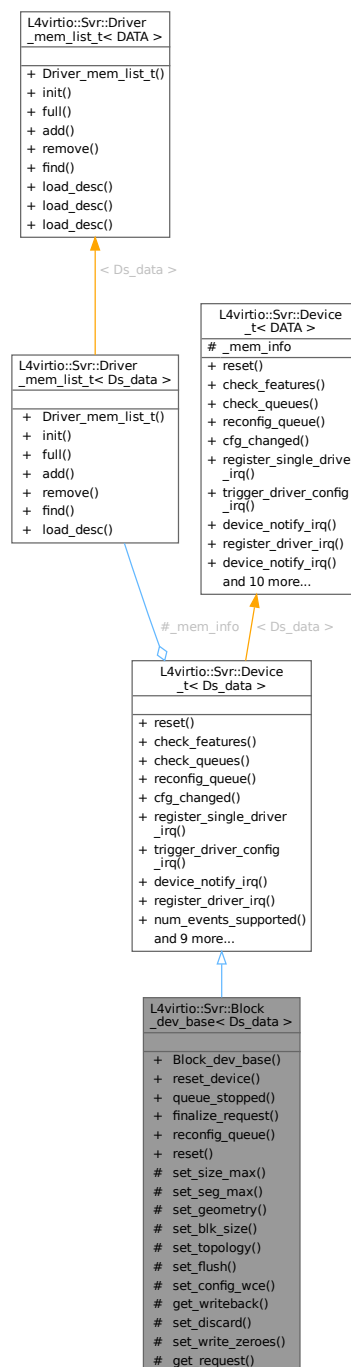
```
#include <virtio-block>
```

Inheritance diagram for L4virtio::Svr::Block\_dev\_base< Ds\_data >:





Collaboration diagram for L4virtio::Svr::Block\_dev\_base< Ds\_data >:



## Public Member Functions

- **Block\_dev\_base** (`l4_uint32_t` vendor, unsigned queue\_size, `l4_uint64_t` capacity, bool read\_only)  
*Create a new virtio block device.*
- virtual void **reset\_device** ()=0  
*Reset the actual hardware device.*
- virtual bool **queue\_stopped** ()=0

- *Return true, if the queues should not be processed further.*
- void **finalize\_request** (cxx::unique\_ptr< [Request](#) > req, unsigned sz, [l4\\_uint8\\_t](#) status=L4VIRTIO\_BLOCK\_S\_OK)  
*Releases resources related to a request and notifies the client.*
- int **reconfig\_queue** (unsigned idx) override  
*callback for client queue-config request*
- void **reset** () override  
*reset callback, called for doing a device reset*

## Public Member Functions inherited from [L4virtio::Svr::Device\\_t< Ds\\_data >](#)

- virtual bool **check\_features** ()  
*callback for checking the subset of accepted features*
- virtual void **cfg\_changed** (unsigned)  
*callback for client device configuration changes*
- virtual [L4::Cap< L4::Irq >](#) **device\_notify\_irq** () const  
*callback to gather the device notification IRQ (old-style)*
- virtual void **register\_driver\_irq** (unsigned idx)  
*Callback for registering an notification IRQ (multi IRQ).*
- virtual unsigned **num\_events\_supported** () const  
*Return the highest notification index supported.*
- [Device\\_t](#) ([Dev\\_config](#) \*dev\_config)  
*Make a device for the given config.*
- [Mem\\_list](#) const \* **mem\_info** () const  
*Get the memory region list used for this device.*
- void **reset\_queue\_config** (unsigned idx, unsigned num\_max, bool inc\_generation=false)  
*Trigger reset for the configuration space for queue idx.*
- void **init\_mem\_info** (unsigned num)  
*Initialize the memory region list to the given maximum.*
- void **device\_error** ()  
*Transition device into DEVICE\_NEEDS\_RESET state.*
- bool **setup\_queue** ([Virtqueue](#) \*q, unsigned qn, unsigned num\_max)  
*Enable/disable the specified queue.*
- bool **handle\_mem\_cmd\_write** ()  
*Check for a value in the cmd register and handle a write.*
- void **enable\_trusted\_ds\_validation** ()  
*Enable trusted dataspace validation.*
- void **add\_trusted\_dataspaces** (std::shared\_ptr< [Ds\\_vector](#) const > ds)  
*Provide a list of trusted dataspaces that can be used for validation.*

## Protected Member Functions

- void **set\_size\_max** ([l4\\_uint32\\_t](#) sz)  
*Sets the maximum size of any single segment reported to client.*
- void **set\_seg\_max** ([l4\\_uint32\\_t](#) sz)  
*Sets the maximum number of segments in a request that is reported to client.*
- void **set\_geometry** ([l4\\_uint16\\_t](#) cylinders, [l4\\_uint8\\_t](#) heads, [l4\\_uint8\\_t](#) sectors)  
*Set disk geometry that is reported to the client.*
- void **set\_blk\_size** ([l4\\_uint32\\_t](#) sz)  
*Sets block disk size to be reported to the client.*

- void [set\\_topology](#) ([l4\\_uint8\\_t](#) physical\_block\_exp, [l4\\_uint8\\_t](#) alignment\_offset, [l4\\_uint32\\_t](#) min\_io\_size, [l4\\_uint32\\_t](#) opt\_io\_size)  
*Sets the I/O alignment information reported back to the client.*
- void **set\_flush** ()  
*Enables the flush command.*
- void [set\\_config\\_wce](#) ([l4\\_uint8\\_t](#) writeback)  
*Sets cache mode and enables the writeback toggle.*
- [l4\\_uint8\\_t](#) [get\\_writeback](#) ()  
*Get the writeback field from the configuration space.*
- void [set\\_discard](#) ([l4\\_uint32\\_t](#) max\_discard\_sectors, [l4\\_uint32\\_t](#) max\_discard\_seg, [l4\\_uint32\\_t](#) discard\_↔ sector\_alignment)  
*Sets constraints for and enables the discard command.*
- void [set\\_write\\_zeroes](#) ([l4\\_uint32\\_t](#) max\_write\_zeroes\_sectors, [l4\\_uint32\\_t](#) max\_write\_zeroes\_seg, [l4\\_uint8\\_t](#) write\_zeroes\_may\_unmap)  
*Sets constraints for and enables the write zeroes command.*
- [cxx::unique\\_ptr](#)< [Request](#) > **get\_request** ()  
*Return one request if available.*

### Additional Inherited Members

### Protected Attributes inherited from [L4virtio::Svr::Device\\_t](#)< [Ds\\_data](#) >

- [Mem\\_list](#) \_mem\_info  
*Memory region list.*

## 16.400.1 Detailed Description

```
template<typename Ds_data>
class L4virtio::Svr::Block_dev_base< Ds_data >
```

Base class for virtio block devices.

Use this class as a base to implement your own specific block device.

Definition at line 259 of file [virtio-block](#).

## 16.400.2 Constructor & Destructor Documentation

### 16.400.2.1 Block\_dev\_base()

```
template<typename Ds_data>
L4virtio::Svr::Block_dev_base< Ds_data >::Block_dev_base (
    l4\_uint32\_t vendor,
    unsigned queue\_size,
    l4\_uint64\_t capacity,
    bool read\_only) [inline]
```

Create a new virtio block device.

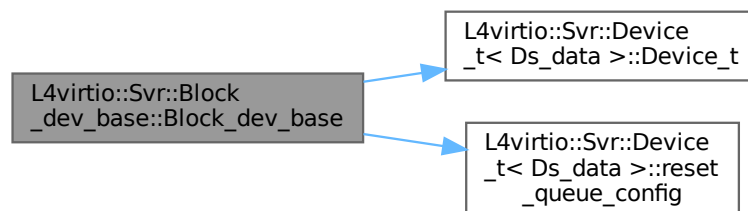
### Parameters

<i>vendor</i>	Vendor ID
<i>queue_size</i>	Number of entries to provide in avail and used queue.
<i>capacity</i>	Size of the device in 512-byte sectors.
<i>read_only</i>	True, if the device should not be writable.

Definition at line 444 of file [virtio-block](#).

References [L4virtio::Svr::Device\\_t< Ds\\_data >::Device\\_t\(\)](#), [L4VIRTIO\\_FEATURE\\_VERSION\\_1](#), [L4VIRTIO\\_ID\\_BLOCK](#), and [L4virtio::Svr::Device\\_t< Ds\\_data >::reset\\_queue\\_config\(\)](#).

Here is the call graph for this function:



### 16.400.3 Member Function Documentation

#### 16.400.3.1 finalize\_request()

```

template<typename Ds_data>
void L4virtio::Svr::Block_dev_base< Ds_data >::finalize_request (
    cxx::unique_ptr< Request > req,
    unsigned sz,
    l4_uint8_t status = L4VIRTIO_BLOCK_S_OK) [inline]
  
```

Releases resources related to a request and notifies the client.

#### Parameters

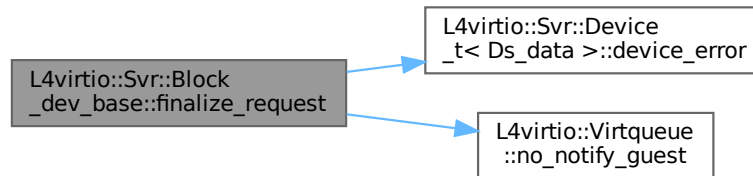
<i>req</i>	Pointer to request that has finished.
<i>sz</i>	Number of bytes consumed.
<i>status</i>	Status of request (see <a href="#">L4virtio_block_status</a> ).

This function must be called when an asynchronous request finishes, either successfully or with an error. The status byte in the request must have been set prior to calling it.

Definition at line 483 of file [virtio-block](#).

References [L4virtio::Svr::Device\\_t< Ds\\_data >::device\\_error\(\)](#), [L4VIRTIO\\_BLOCK\\_S\\_OK](#), [L4VIRTIO\\_IRQ\\_STATUS\\_VRING](#), and [L4virtio::Virtqueue::no\\_notify\\_guest\(\)](#).

Here is the call graph for this function:



### 16.400.3.2 get\_writeback()

```
template<typename Ds_data>
l4_uint8_t L4virtio::Svr::Block_dev_base< Ds_data >::get_writeback () [inline], [protected]
```

Get the writeback field from the configuration space.

#### Returns

Value of the writeback field.

Definition at line 388 of file [virtio-block](#).

### 16.400.3.3 set\_blk\_size()

```
template<typename Ds_data>
void L4virtio::Svr::Block_dev_base< Ds_data >::set_blk_size (
    l4_uint32_t sz) [inline], [protected]
```

Sets block disk size to be reported to the client.

Setting this does not change the logical sector size used for addressing the device.

Definition at line 332 of file [virtio-block](#).

### 16.400.3.4 set\_config\_wce()

```
template<typename Ds_data>
void L4virtio::Svr::Block_dev_base< Ds_data >::set_config_wce (
    l4_uint8_t writeback) [inline], [protected]
```

Sets cache mode and enables the writeback toggle.

#### Parameters

---

<i>writeback</i>	Mode of the cache (0 for writethrough, 1 for writeback).
------------------	----------------------------------------------------------

Definition at line 375 of file [virtio-block](#).

#### 16.400.3.5 set\_discard()

```
template<typename Ds_data>
void L4virtio::Svr::Block_dev_base< Ds_data >::set_discard (
    l4_uint32_t max_discard_sectors,
    l4_uint32_t max_discard_seg,
    l4_uint32_t discard_sector_alignment) [inline], [protected]
```

Sets constraints for and enables the discard command.

##### Parameters

<i>max_discard_sectors</i>	Maximum discard sectors size.
<i>max_discard_seg</i>	Maximum discard segment number.
<i>discard_sector_alignment</i>	Can be used by the driver when splitting a request based on alignment.

Definition at line 402 of file [virtio-block](#).

#### 16.400.3.6 set\_size\_max()

```
template<typename Ds_data>
void L4virtio::Svr::Block_dev_base< Ds_data >::set_size_max (
    l4_uint32_t sz) [inline], [protected]
```

Sets the maximum size of any single segment reported to client.

The limit is also applied to any incoming requests. Requests with larger segments result in an IO error being reported to the client. That means that `process_request()` can safely make the assumption that all segments in the received request are smaller.

Definition at line 290 of file [virtio-block](#).

#### 16.400.3.7 set\_topology()

```
template<typename Ds_data>
void L4virtio::Svr::Block_dev_base< Ds_data >::set_topology (
    l4_uint8_t physical_block_exp,
    l4_uint8_t alignment_offset,
    l4_uint32_t min_io_size,
    l4_uint32_t opt_io_size) [inline], [protected]
```

Sets the I/O alignment information reported back to the client.

##### Parameters

<i>physical_block_exp</i>	Number of logical blocks per physical block(log2)
<i>alignment_offset</i>	Offset of the first aligned logical block
<i>min_io_size</i>	Suggested minimum I/O size in blocks
<i>opt_io_size</i>	Optimal I/O size in blocks

Definition at line 348 of file [virtio-block](#).

### 16.400.3.8 set\_write\_zeroes()

```
template<typename Ds_data>
void L4virtio::Svr::Block_dev_base< Ds_data >::set_write_zeroes (
    14_uint32_t max_write_zeroes_sectors,
    14_uint32_t max_write_zeroes_seg,
    14_uint8_t write_zeroes_may_unmap) [inline], [protected]
```

Sets constraints for and enables the write zeroes command.

#### Parameters

<i>max_write_zeroes_sectors</i>	Maximum write zeroes sectors size.
<i>max_write_zeroes_seg</i>	maximum write zeroes segment number.
<i>write_zeroes_may_unmap</i>	Set if a write zeroes request can result in deallocating one or more sectors.

Definition at line 422 of file [virtio-block](#).

The documentation for this class was generated from the following file:

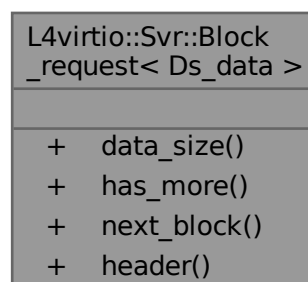
- l4/l4virtio/server/virtio-block

## 16.401 L4virtio::Svr::Block\_request< Ds\_data > Class Template Reference

A request to read or write data.

```
#include <virtio-block>
```

Collaboration diagram for L4virtio::Svr::Block\_request< Ds\_data >:



## Public Member Functions

- unsigned [data\\_size](#) () const  
*Compute the total size of the data in the request.*
- bool **has\_more** ()  
*Check if the request contains more data blocks.*
- Data\_block [next\\_block](#) ()  
*Return next block in scatter-gather list.*
- [l4virtio\\_block\\_header\\_t](#) const & **header** () const  
*Return the block request header.*

### 16.401.1 Detailed Description

```
template<typename Ds_data>
class L4virtio::Svr::Block_request< Ds_data >
```

A request to read or write data.

Definition at line 28 of file [virtio-block](#).

### 16.401.2 Member Function Documentation

#### 16.401.2.1 data\_size()

```
template<typename Ds_data>
unsigned L4virtio::Svr::Block_request< Ds_data >::data_size () const [inline]
```

Compute the total size of the data in the request.

#### Return values

Size	in bytes or 0 if there was an error.
------	--------------------------------------

#### Exceptions

<a href="#">L4::Runtime_error(-L4_EIO)</a>	Request has a bad format.
--------------------------------------------	---------------------------

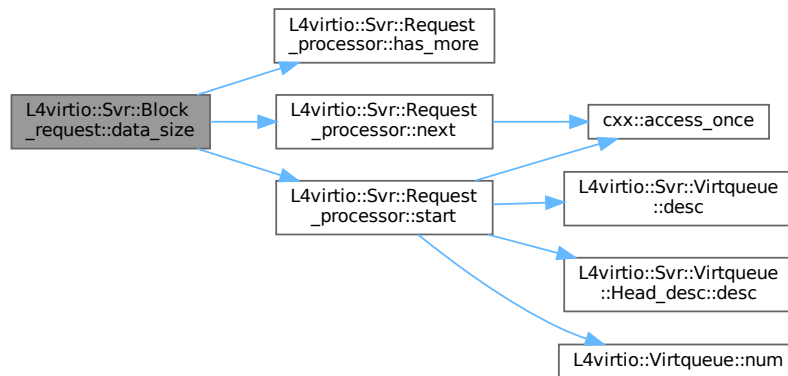
Note that this operation is relatively expensive as it has to iterate over the complete list of blocks.

Definition at line 63 of file [virtio-block](#).

References [L4virtio::Svr::Request\\_processor::has\\_more\(\)](#), [L4\\_EIO](#), [L4virtio::Svr::Request\\_processor::next\(\)](#), and [L4virtio::Svr::Request\\_processor::start\(\)](#).



Here is the call graph for this function:



### 16.401.2.2 next\_block()

```
template<typename Ds_data>
Data_block L4virtio::Svr::Block_request< Ds_data >::next_block () [inline]
```

Return next block in scatter-gather list.

#### Returns

Information about the next data block.

#### Exceptions

<a href="#"><i>L4::Runtime_error</i></a>	No more data block is available.
<a href="#"><i>Bad_descriptor</i></a>	Virtio request is corrupted.

Definition at line 113 of file [virtio-block](#).

References [L4virtio::Svr::Bad\\_descriptor::Bad\\_size](#), and [L4\\_EEXIST](#).

The documentation for this class was generated from the following file:

- `I4/I4virtio/server/virtio-block`

## 16.402 L4virtio::Svr::Console::Control\_message Struct Reference

Virtio console control message.

```
#include <virtio-console>
```

Collaboration diagram for L4virtio::Svr::Console::Control\_message:

L4virtio::Svr::Console ::Control_message	
+	id
+	event
+	value

### Public Types

- enum [Events](#) {  
[Device\\_ready](#) = 0 , [Device\\_add](#) = 1 , [Device\\_remove](#) = 2 , [Port\\_ready](#) = 3 ,  
[Console\\_port](#) = 4 , [Resize](#) = 5 , [Port\\_open](#) = 6 , [Port\\_name](#) = 7 }  
*Possible control events.*

### Data Fields

- [l4\\_uint32\\_t](#) **id**  
*Port number.*
- [l4\\_uint16\\_t](#) **event**  
*Control event, see [Events](#).*
- [l4\\_uint16\\_t](#) **value**  
*Extra information.*

### 16.402.1 Detailed Description

Virtio console control message.

Definition at line [31](#) of file [virtio-console](#).

### 16.402.2 Member Enumeration Documentation

#### 16.402.2.1 Events

```
enum L4virtio::Svr::Console::Control_message::Events
```

Possible control events.

#### Enumerator

---

Device_ready	Sent by driver at initialization.
Device_add	Sent by device to create new ports.
Device_remove	Sent by device to remove added ports.
Port_ready	Sent by driver as response to <a href="#">Device_add</a> .
Console_port	Sent by device to nominate port as console port.
Resize	Sent by device to indicate a console size change.
Port_open	Sent by device and driver to indicate whether a port is open.
Port_name	Sent by device to tag a port.

Definition at line 34 of file [virtio-console](#).

The documentation for this struct was generated from the following file:

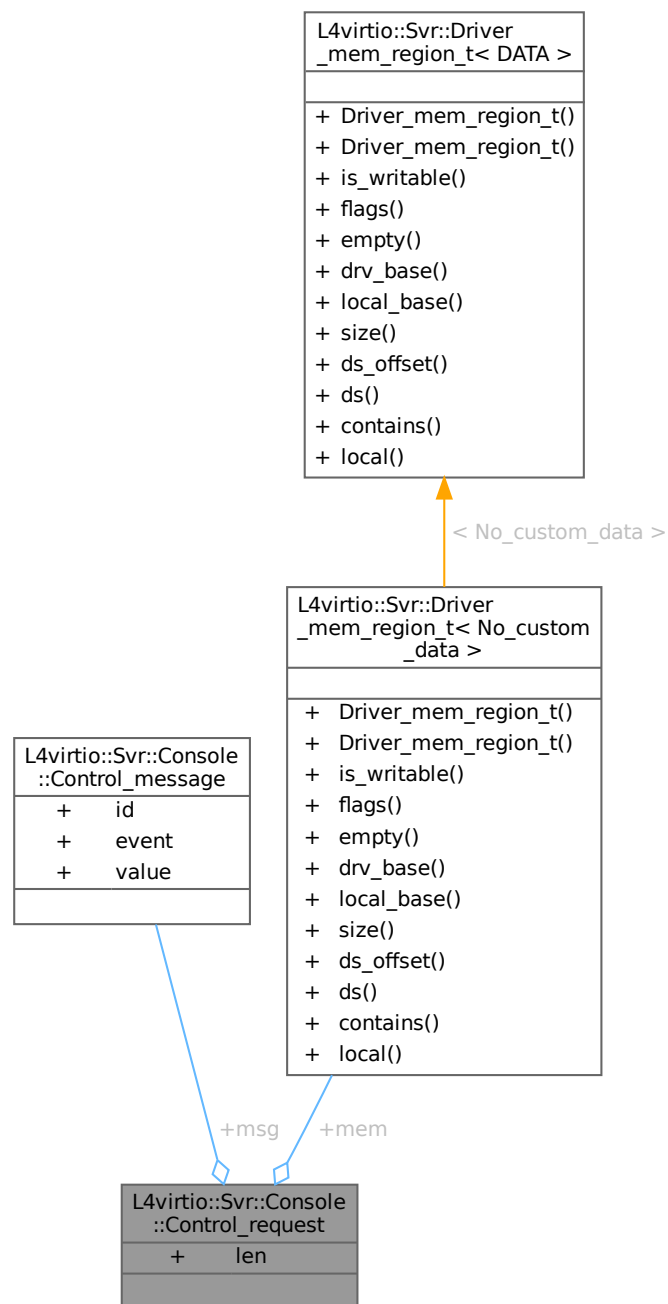
- l4/l4virtio/server/virtio-console

## 16.403 L4virtio::Svr::Console::Control\_request Struct Reference

Specialised `Virtqueue::Request` providing access to control message payload.

```
#include <virtio-console>
```

Collaboration diagram for L4virtio::Svr::Console::Control\_request:



## Data Fields

- **Control\_message \* msg**  
Virtual address of the data block (in device space).
- **l4\_uint32\_t len**  
Length of datablock in bytes.
- **Driver\_mem\_region \* mem**  
Pointer to driver memory region.

### 16.403.1 Detailed Description

Specialised `Virtqueue::Request` providing access to control message payload.

Definition at line 65 of file [virtio-console](#).

The documentation for this struct was generated from the following file:

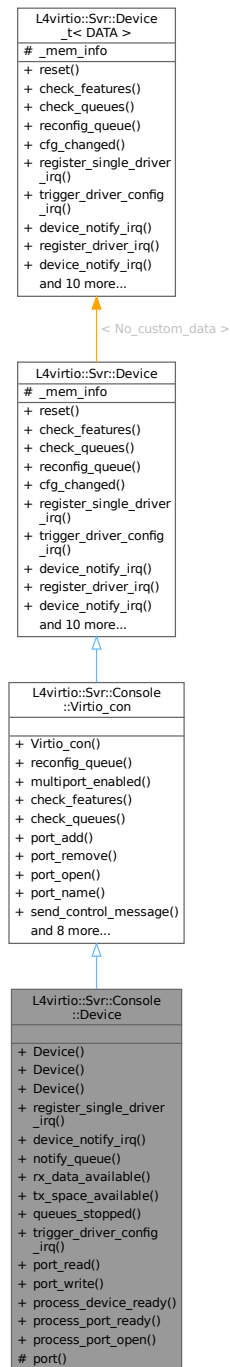
- `l4/l4virtio/server/virtio-console`

## 16.404 L4virtio::Svr::Console::Device Class Reference

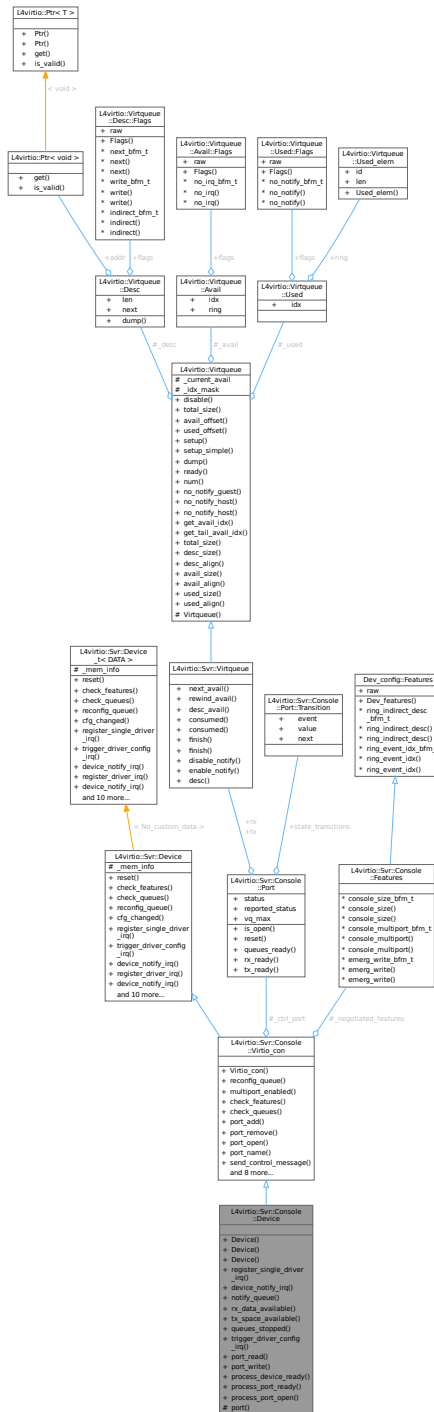
Base class implementing a virtio console device with L4Re-based notification handling.

```
#include <virtio-console-device>
```

Inheritance diagram for L4virtio::Svr::Console::Device:



Collaboration diagram for L4virtio::Svr::Console::Device:



## Public Member Functions

- [Device](#) (unsigned vq\_max)  
Create a new console device.
- [Device](#) (unsigned vq\_max, unsigned ports)  
Create a new console device.
- [Device](#) (cx::static\_vector< unsigned > const &vq\_max\_nums)

- Create a new console [Device](#).
- void **register\_single\_driver\_irq** () override  
callback for registering a single guest IRQ for all queues (old-style)
- [L4::Cap](#)< [L4::Irq](#) > **device\_notify\_irq** () const override  
callback to gather the device notification IRQ (old-style)
- void **notify\_queue** ([Virtqueue](#) \*queue) override  
Notify queue of available data.
- virtual void **rx\_data\_available** (unsigned [port](#))=0  
Callback to notify that new data is available to be read from port.
- virtual void **tx\_space\_available** (unsigned [port](#))=0  
Callback to notify that data can be written to port.
- virtual bool **queues\_stopped** ()  
Return true, if the queues should not be processed further.
- void **trigger\_driver\_config\_irq** () override  
callback for triggering configuration change notification IRQ
- unsigned **port\_read** (char \*buf, unsigned len, unsigned [port](#)=0)  
Read data from port.
- unsigned **port\_write** (char const \*buf, unsigned len, unsigned [port](#)=0)  
Write data to port.
- void **process\_device\_ready** ([l4\\_uint16\\_t](#) value) override  
Callback called on `DEVICE_READY` event.
- void **process\_port\_ready** ([l4\\_uint32\\_t](#) id, [l4\\_uint16\\_t](#) value) override  
Callback called on `PORT_READY` event.
- virtual void **process\_port\_open** ([l4\\_uint32\\_t](#) id, [l4\\_uint16\\_t](#) value)  
Callback called on `PORT_OPEN` event.

## Public Member Functions inherited from [L4virtio::Svr::Console::Virtio\\_con](#)

- [Virtio\\_con](#) (unsigned max\_ports, bool enable\_multiport)  
Create a new multiport console device.
- int **reconfig\_queue** (unsigned index) override  
callback for client queue-config request
- bool **multiport\_enabled** () const  
Return true if the multiport feature is enabled and control queues are available.
- bool **check\_features** (void) override  
callback for checking the subset of accepted features
- bool **check\_queues** () override  
callback for checking if the queues at `DRIVER_OK` transition
- int **port\_add** (unsigned idx)  
Send a `DEVICE_ADD` message and update the internal state.
- int **port\_remove** (unsigned idx)  
Send a `DEVICE_REMOVE` message and update the internal state.
- int **port\_open** (unsigned idx, bool open)  
Send a `PORT_OPEN` message and update the internal state.
- int **port\_name** (unsigned idx, char const \*name)  
Send a `PORT_NAME` message to announce the port name.
- int **send\_control\_message** ([l4\\_uint32\\_t](#) idx, [l4\\_uint16\\_t](#) event, [l4\\_uint16\\_t](#) value=0, const char \*name=0)  
Send control message to driver.
- int **handle\_control\_message** ()  
Handle control message received from the driver.
- void **reset** () override  
reset callback, called for doing a device reset
- virtual void **reset\_device** ()  
Reset the state of the actual console device.



## Public Member Functions inherited from L4virtio::Svr::Device\_t< No\_custom\_data >

- virtual void **cfg\_changed** (unsigned)  
*callback for client device configuration changes*
- virtual void **register\_driver\_irq** (unsigned idx)  
*Callback for registering an notification IRQ (multi IRQ).*
- virtual L4::Cap< L4::Irq > **device\_notify\_irq** (unsigned idx)  
*Callback to gather the device notification IRQ (multi IRQ).*
- virtual unsigned **num\_events\_supported** () const  
*Return the highest notification index supported.*
- **Device\_t** (Dev\_config \*dev\_config)  
*Make a device for the given config.*
- **Mem\_list** const \* **mem\_info** () const  
*Get the memory region list used for this device.*
- void **reset\_queue\_config** (unsigned idx, unsigned num\_max, bool inc\_generation=false)  
*Trigger reset for the configuration space for queue idx.*
- void **init\_mem\_info** (unsigned num)  
*Initialize the memory region list to the given maximum.*
- void **device\_error** ()  
*Transition device into DEVICE\_NEEDS\_RESET state.*
- bool **setup\_queue** (Virtqueue \*q, unsigned qn, unsigned num\_max)  
*Enable/disable the specified queue.*
- bool **handle\_mem\_cmd\_write** ()  
*Check for a value in the cmd register and handle a write.*
- void **enable\_trusted\_ds\_validation** ()  
*Enable trusted dataspace validation.*
- void **add\_trusted\_dataspaces** (std::shared\_ptr< Ds\_vector const > ds)  
*Provide a list of trusted dataspaces that can be used for validation.*

## Protected Member Functions

- **Port** \* **port** (unsigned idx) override  
*Return the specified port.*

## Additional Inherited Members

## Protected Attributes inherited from L4virtio::Svr::Device\_t< No\_custom\_data >

- **Mem\_list** **mem\_info**  
*Memory region list.*

### 16.404.1 Detailed Description

Base class implementing a virtio console device with L4Re-based notification handling.

This console device is derived from [Virtio\\_con](#) and already includes functionality to handle interrupts and notify drivers. If an interrupt is received, all the necessary interaction with the virtqueues is performed and only the actual data processing has to be done by the derived class. By default all available ports are added and an "open"-request of a port by the driver is automatically acknowledged. The derived class can optionally change this behaviour by overriding [process\\_device\\_ready\(\)](#), [process\\_port\\_ready\(\)](#) and [process\\_port\\_open\(\)](#).

This class provides a stream-based interface to access the port data with edge-triggered notification callbacks. If a port receives data from the driver the derived class is notified with the [rx\\_data\\_available\(\)](#) callback. The actual data can be retrieved by [port\\_read\(\)](#). If there was not enough data to be read, the call will return the available partial data. Only then will the [rx\\_data\\_available\(\)](#) callback be triggered again.

Data on a port may be transmitted by [port\\_write\(\)](#). If there were not enough buffers available, only a part of the data will be transmitted. Once there are new buffers available, the [tx\\_space\\_available\(\)](#) callback will be invoked. This callback will be called again only after a previous [port\\_write\(\)](#) was not able to send all requested data.

Use this class as a base to provide your own high-level console device. You must derive from this class as well as [L4::Epiface\\_t<..., L4virtio::Device>](#). For a working device the [irq\\_iface\(\)](#) must be registered too. A typical implementation might look like the following:

```
class My_console
: public L4virtio::Svr::Console::Device,
  public L4::Epiface_t<My_console, L4virtio::Device>
{
public:
    My_console(L4Re::Util::Object_registry *r)
    : L4virtio::Svr::Console::Device(0x100)
    {
        init_mem_info(4);
        L4Re::chkcap(r->register_irq_obj(irq_iface()), "virtio notification IRQ");
    }

    void rx_data_available(unsigned port) override
    {
        // call port_read() to fetch available data
    }

    void tx_space_available(unsigned port) override
    {
        // can call port_write() to send (pending) data
    }
};

My_console console(registry);
registry->register_obj(&console, ...);
```

The maximum number of memory regions ([init\\_mem\\_info\(\)](#)) should correlate with the number of supported ports.

Definition at line 118 of file [virtio-console-device](#).

### 16.404.2 Constructor & Destructor Documentation

#### 16.404.2.1 Device() [1/3]

```
L4virtio::Svr::Console::Device::Device (
    unsigned vq_max) [inline], [explicit]
```

Create a new console device.

#### Parameters

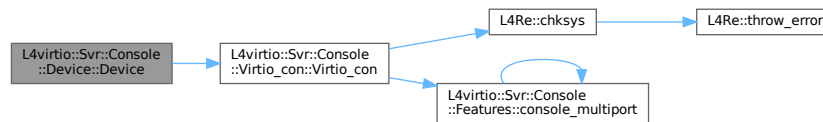
<i>vq_max</i>	Maximum number of buffers in data queues.
---------------	-------------------------------------------

Create a console device with no multiport support, i.e. control queues are disabled.

Definition at line 145 of file [virtio-console-device](#).

References [L4virtio::Svr::Console::Virtio\\_con::Virtio\\_con\(\)](#).

Here is the call graph for this function:



### 16.404.2.2 Device() [2/3]

```
L4virtio::Svr::Console::Device::Device (
    unsigned vq_max,
    unsigned ports) [inline], [explicit]
```

Create a new console device.

#### Parameters

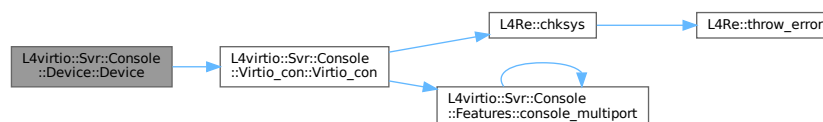
<i>vq_max</i>	Maximum number of buffers in data queues.
<i>ports</i>	Number of ports (maximum 32).

Create a console device with multiport support, i.e. control queues are enabled.

Definition at line 163 of file [virtio-console-device](#).

References [L4virtio::Svr::Console::Virtio\\_con::Virtio\\_con\(\)](#).

Here is the call graph for this function:



### 16.404.2.3 Device() [3/3]

```
L4virtio::Svr::Console::Device::Device (
    cxx::static_vector< unsigned > const & vq_max_nums) [inline], [explicit]
```

Create a new console [Device](#).

#### Parameters

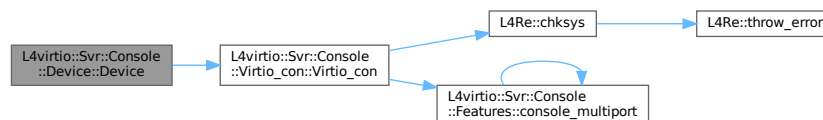
<code>vq_max_nums</code>	Maximum number of buffers in data queues, given as a <code>cx::static_vector</code> with one entry per port.
--------------------------	--------------------------------------------------------------------------------------------------------------

Create a console device with multiport support, i.e. control queues are enabled.

Definition at line 182 of file `virtio-console-device`.

References `L4virtio::Svr::Console::Virtio_con::Virtio_con()`.

Here is the call graph for this function:



## 16.404.3 Member Function Documentation

### 16.404.3.1 notify\_queue()

```
void L4virtio::Svr::Console::Device::notify_queue (
    Virtqueue * queue) [inline], [override], [virtual]
```

Notify queue of available data.

#### Parameters

<code>queue</code>	<code>Virtqueue</code> to notify.
--------------------	-----------------------------------

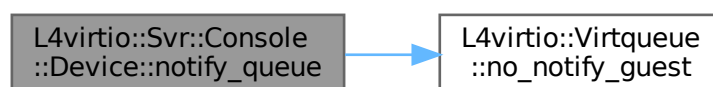
This callback is called whenever data is sent to `queue`. It is the responsibility of the derived class to perform all necessary notification actions, e.g. triggering guest interrupts.

Implements `L4virtio::Svr::Console::Virtio_con`.

Definition at line 202 of file `virtio-console-device`.

References `L4VIRTIO_IRQ_STATUS_VRING`, and `L4virtio::Virtqueue::no_notify_guest()`.

Here is the call graph for this function:



### 16.404.3.2 port()

```
Port * L4virtio::Svr::Console::Device::port (
    unsigned port) [inline], [override], [protected], [virtual]
```

Return the specified port.

#### Parameters

<i>port</i>	Port number.
-------------	--------------

#### Precondition

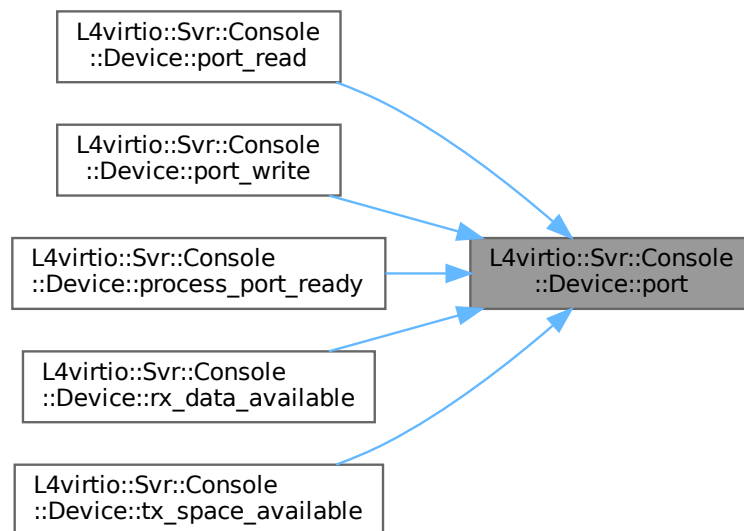
Port number must be lower than the configured maximum number of ports.

Implements [L4virtio::Svr::Console::Virtio\\_con](#).

Definition at line 450 of file [virtio-console-device](#).

Referenced by [port\\_read\(\)](#), [port\\_write\(\)](#), [process\\_port\\_ready\(\)](#), [rx\\_data\\_available\(\)](#), and [tx\\_space\\_available\(\)](#).

Here is the caller graph for this function:



### 16.404.3.3 port\_read()

```
unsigned L4virtio::Svr::Console::Device::port_read (  
    char * buf,  
    unsigned len,  
    unsigned port = 0) [inline]
```

Read data from port.

Will read up to *len* bytes from *port* into *buf*. Returns the number of bytes read, which may be less if not enough data was available. If all data was read, the [rx\\_data\\_available\(\)](#) callback will be invoked the next time the driver queues new data for the port. The callback won't be called again until all data was consumed again.

#### Parameters

---

<i>buf</i>	The destination buffer
<i>len</i>	Size of the buffer
<i>port</i>	<a href="#">Port</a> index to read data from

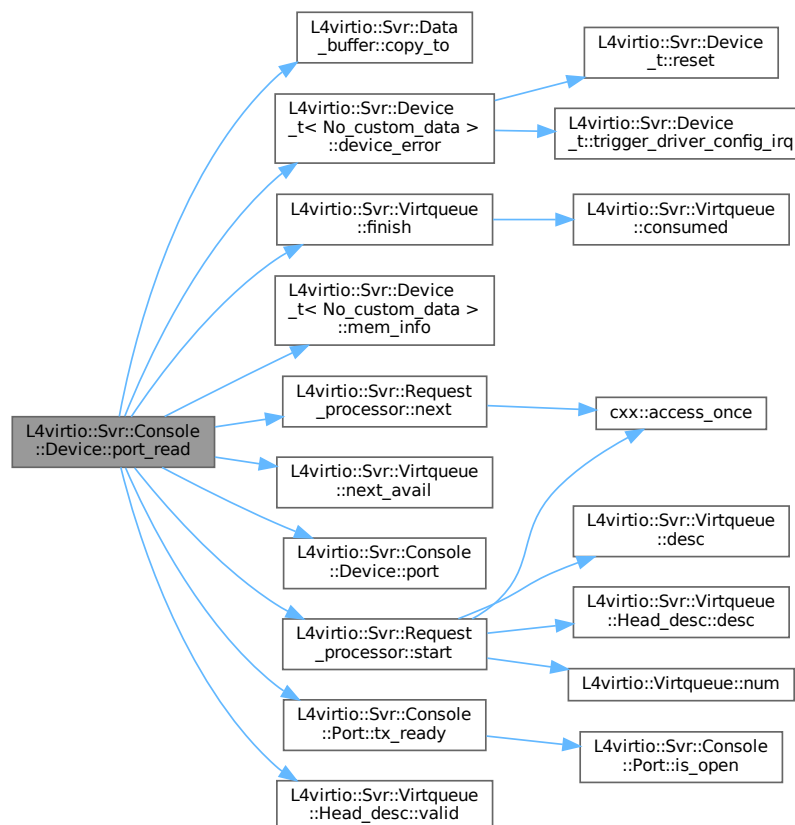
**Returns**

Number of bytes read

Definition at line 272 of file [virtio-console-device](#).

References [L4virtio::Svr::Data\\_buffer::copy\\_to\(\)](#), [L4virtio::Svr::Device\\_t< No\\_custom\\_data >::device\\_error\(\)](#), [L4virtio::Svr::Virtqueue::finish\(\)](#), [L4virtio::Svr::Data\\_buffer::left](#), [L4virtio::Svr::Device\\_t< No\\_custom\\_data >::mem\\_info\(\)](#), [L4virtio::Svr::Request\\_processor::next\(\)](#), [L4virtio::Svr::Virtqueue::next\\_avail\(\)](#), [port\(\)](#), [L4virtio::Svr::Data\\_buffer::pos](#), [L4virtio::Svr::Console::Device\\_port::request](#), [L4virtio::Svr::Console::Device\\_port::rp](#), [L4virtio::Svr::Console::Device\\_port::src](#), [L4virtio::Svr::Request\\_processor::start\(\)](#), [L4virtio::Svr::Console::Port::tx](#), [L4virtio::Svr::Console::Port::tx\\_ready\(\)](#), and [L4virtio::Svr::Virtqueue::Head\\_desc::valid\(\)](#).

Here is the call graph for this function:



#### 16.404.3.4 port\_write()

```
unsigned L4virtio::Svr::Console::Device::port_write (  
    char const * buf,  
    unsigned len,  
    unsigned port = 0) [inline]
```

Write data to port.

Will write up to *len* bytes to *port* from *buf*. Returns the number of bytes written, which may be less if not enough virtio buffers were available. If not all data could be written, the [tx\\_space\\_available\(\)](#) callback will be invoked the next time the driver queues new receive buffers for the port. The callback won't be called again until all receive buffers were filled again.

##### Parameters

<i>buf</i>	The source buffer
<i>len</i>	Size of the buffer
<i>port</i>	<a href="#">Port</a> index to write data to

##### Returns

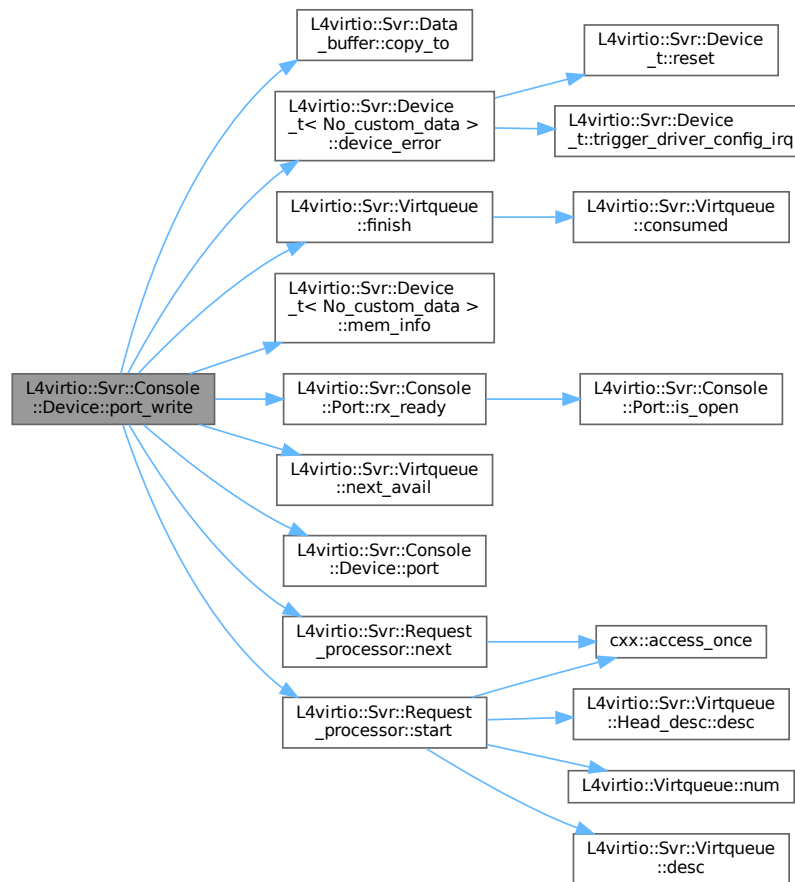
Number of bytes written

Definition at line [341](#) of file [virtio-console-device](#).

References [L4virtio::Svr::Data\\_buffer::copy\\_to\(\)](#), [L4virtio::Svr::Device\\_t< No\\_custom\\_data >::device\\_error\(\)](#), [L4virtio::Svr::Virtqueue::finish\(\)](#), [L4virtio::Svr::Data\\_buffer::left](#), [L4virtio::Svr::Device\\_t< No\\_custom\\_data >::mem\\_info\(\)](#), [L4virtio::Svr::Request\\_processor::next\(\)](#), [L4virtio::Svr::Virtqueue::next\\_avail\(\)](#), [port\(\)](#), [L4virtio::Svr::Data\\_buffer::pos](#), [L4virtio::Svr::Console::Port::rx](#), [L4virtio::Svr::Console::Port::rx\\_ready\(\)](#), and [L4virtio::Svr::Request\\_processor::start\(\)](#).



Here is the call graph for this function:



### 16.404.3.5 process\_device\_ready()

```
void L4virtio::Svr::Console::Device::process_device_ready (
    uint16_t value) [inline], [override], [virtual]
```

Callback called on DEVICE\_READY event.

#### Parameters

<i>value</i>	The value field of the control message, indicating if the initialization was successful.
--------------	------------------------------------------------------------------------------------------

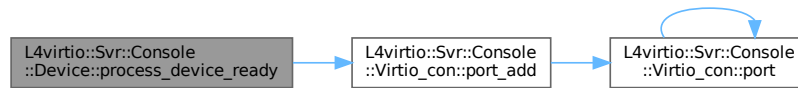
By default, this function adds all ports if the driver indicates successful initialization. Override this function to perform custom actions for a DEVICE\_READY event. It is then your responsibility to inform the driver about usable ports, see [port\\_add\(\)](#).

Implements [L4virtio::Svr::Console::Virtio\\_con](#).

Definition at line 399 of file [virtio-console-device](#).

References [L4virtio::Svr::Console::Virtio\\_con::port\\_add\(\)](#).

Here is the call graph for this function:



#### 16.404.3.6 process\_port\_open()

```
virtual void L4virtio::Svr::Console::Device::process_port_open (
    14_uint32_t id,
    14_uint16_t value) [inline], [virtual]
```

Callback called on PORT\_OPEN event.

##### Parameters

<i>id</i>	The id field of the control message, i.e. the port number.
<i>value</i>	The value field of the control message, indicating if the port was opened or closed.

The default implementation does nothing. Override it to implement some custom logic to respond to open/close events of the driver.

Implements [L4virtio::Svr::Console::Virtio\\_con](#).

Definition at line 443 of file [virtio-console-device](#).

#### 16.404.3.7 process\_port\_ready()

```
void L4virtio::Svr::Console::Device::process_port_ready (
    14_uint32_t id,
    14_uint16_t value) [inline], [override], [virtual]
```

Callback called on PORT\_READY event.

##### Parameters

<i>id</i>	The id field of the control message, i.e. the port number.
<i>value</i>	The value field of the control message, indicating if the initialization was successful.

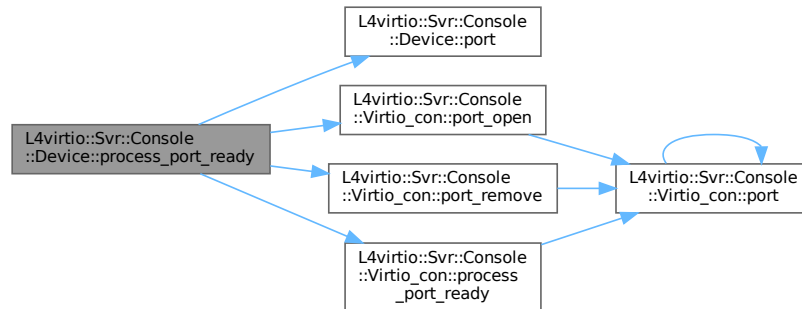
By default, this function opens the port if the driver is ready. Otherwise, the port is removed if the driver failed to set it up correctly. Override this function to perform custom actions for a PORT\_READY event, *after* the generic management of the base class. It is then your responsibility to inform the driver about connected or unusable ports. See [port\\_open\(\)](#) and [port\\_remove\(\)](#).

Reimplemented from [L4virtio::Svr::Console::Virtio\\_con](#).

Definition at line 422 of file [virtio-console-device](#).

References [port\(\)](#), [L4virtio::Svr::Console::Port::Port\\_failed](#), [L4virtio::Svr::Console::Virtio\\_con::port\\_open\(\)](#), [L4virtio::Svr::Console::Port::Port\\_ready](#), [L4virtio::Svr::Console::Virtio\\_con::port\\_remove\(\)](#), [L4virtio::Svr::Console::Virtio\\_con::process\\_port\\_ready](#) and [L4virtio::Svr::Console::Port::status](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

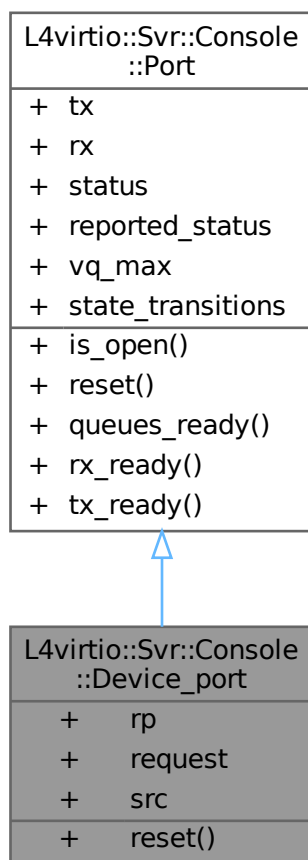
- `l4/l4virtio/server/virtio-console-device`

## 16.405 L4virtio::Svr::Console::Device\_port Struct Reference

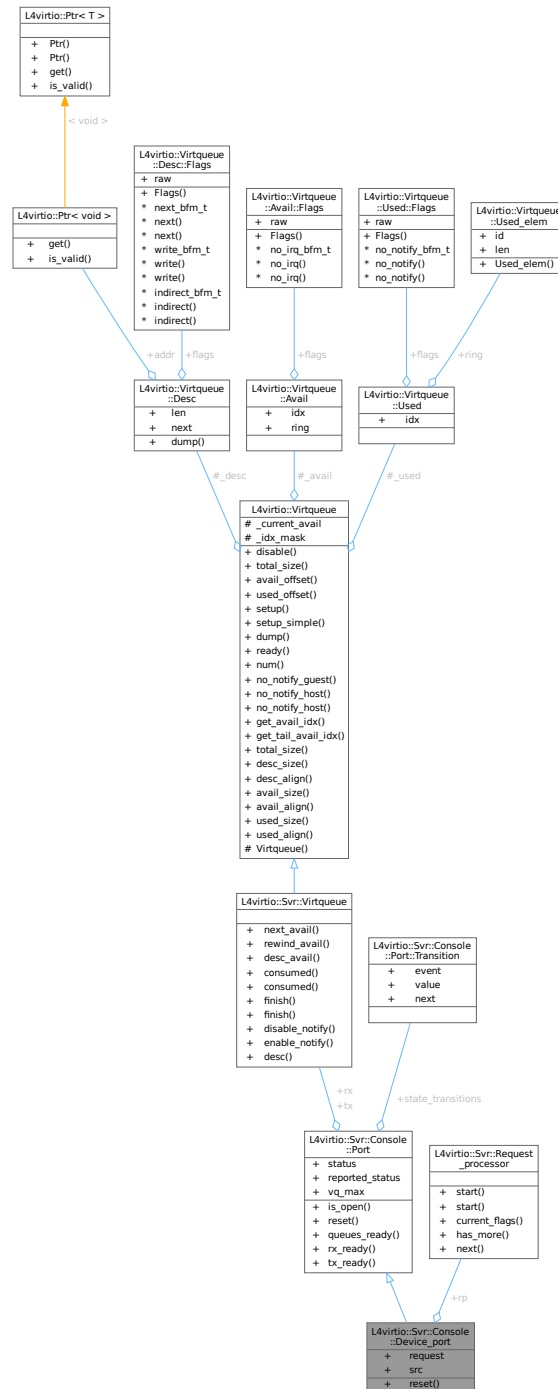
A console port with associated read/write state.

```
#include <virtio-console-device>
```

Inheritance diagram for L4virtio::Svr::Console::Device\_port:



Collaboration diagram for L4virtio::Svr::Console::Device\_port:



## Public Member Functions

- void **reset** () override  
*Reset the port to the initial state and disable its virtqueues.*

## Public Member Functions inherited from L4virtio::Svr::Console::Port

- bool **is\_open** () const

- *Check that the port is open.*
- bool **queues\_ready** () const  
*Check that both virtqueues are set up correctly.*
- bool **rx\_ready** () const  
*Check that device implementation may write to receive queues.*
- bool **tx\_ready** () const  
*Check that device implementation may read from transmit queues.*

## Data Fields

- [Request\\_processor](#) **rp**  
*Request processor associated with current request.*
- Virtqueue::Request **request**  
*Current virtio tx queue request.*
- [Buffer](#) **src**  
*Source data block to process.*

## Data Fields inherited from [L4virtio::Svr::Console::Port](#)

- [Virtqueue](#) **tx**  
*Receiveq of the port.*
- [Virtqueue](#) **rx**  
*Transmitq of the port.*
- [Port\\_status](#) **status**  
*State the port is in.*
- [Port\\_status](#) **reported\_status**  
*State the port was last reported.*
- unsigned **vq\_max**  
*Maximum queue sizes for this port.*

## Additional Inherited Members

## Public Types inherited from [L4virtio::Svr::Console::Port](#)

- enum [Port\\_status](#) {  
  [Port\\_disabled](#) = 0 , [Port\\_added](#) , [Port\\_ready](#) , [Port\\_open](#) ,  
  [Port\\_failed](#) , [Port\\_num\\_states](#) }  
*Possible states of a virtio console port.*
- enum  
*Size of control queues, also used as default size.*

## Static Public Attributes inherited from [L4virtio::Svr::Console::Port](#)

- static constexpr [Transition](#) **state\_transitions** [[Port\\_num\\_states](#)][[Port\\_num\\_states](#)]  
*State transition table from last report state to current state.*

### 16.405.1 Detailed Description

A console port with associated read/write state.

Tracks the notification of the device implementation and holds the state when receiving data from the driver.

Definition at line 26 of file [virtio-console-device](#).

The documentation for this struct was generated from the following file:

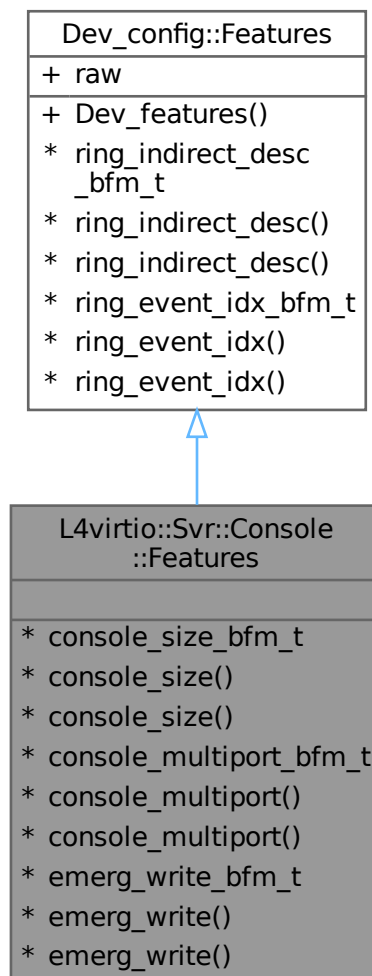
- l4/l4virtio/server/virtio-console-device

## 16.406 L4virtio::Svr::Console::Features Struct Reference

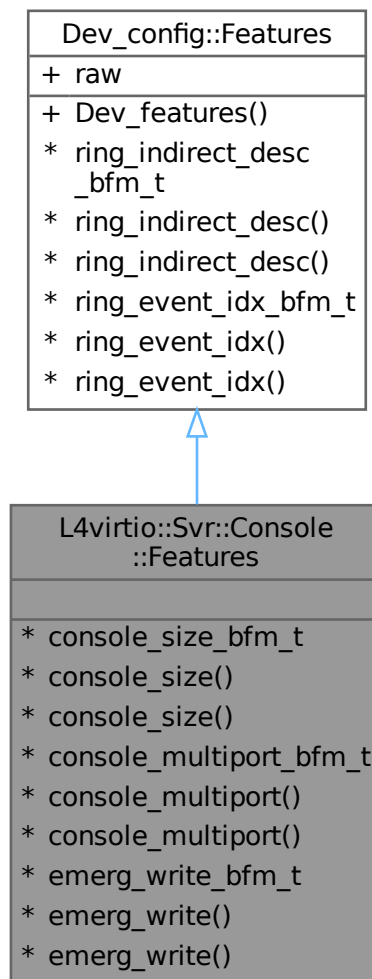
Virtio console specific feature bits.

```
#include <virtio-console>
```

Inheritance diagram for L4virtio::Svr::Console::Features:



Collaboration diagram for L4virtio::Svr::Console::Features:



- typedef `cxx::Bitfield< decltype(raw), 0, 0 >` `console_size_bfm_t`  
Configuration `cols` and `rows` are valid.
- constexpr `console_size_bfm_t::Val console_size ()` const  
Get the `console_size` bits (0 to 0) of `raw`.
- constexpr `console_size_bfm_t::Ref console_size ()`  
Get a reference to the `console_size` bits (0 to 0) of `raw`.
- typedef `cxx::Bitfield< decltype(raw), 1, 1 >` `console_multiport_bfm_t`  
`Device` has support for multiple ports.
- constexpr `console_multiport_bfm_t::Val console_multiport ()` const  
Get the `console_multiport` bits (1 to 1) of `raw`.
- constexpr `console_multiport_bfm_t::Ref console_multiport ()`  
Get a reference to the `console_multiport` bits (1 to 1) of `raw`.



- typedef [cxx::Bitfield](#)< decltype([raw](#)), 2, 2 > [emerg\\_write\\_bfm\\_t](#)  
*Device has support for emergency write.*
- constexpr [emerg\\_write\\_bfm\\_t::Val](#) [emerg\\_write](#) () const  
*Get the [emerg\\_write](#) bits (2 to 2) of [raw](#).*
- constexpr [emerg\\_write\\_bfm\\_t::Ref](#) [emerg\\_write](#) ()  
*Get a reference to the [emerg\\_write](#) bits (2 to 2) of [raw](#).*

### Additional Inherited Members

- typedef [cxx::Bitfield](#)< decltype([raw](#)), 28, 28 > [ring\\_indirect\\_desc\\_bfm\\_t](#)  
*Type to access the [ring\\_indirect\\_desc](#) bits (28 to 28) of [raw](#).*
- typedef [cxx::Bitfield](#)< decltype([raw](#)), 29, 29 > [ring\\_event\\_idx\\_bfm\\_t](#)  
*Type to access the [ring\\_event\\_idx](#) bits (29 to 29) of [raw](#).*

### Public Member Functions inherited from [L4virtio::Svr::Dev\\_features](#)

- [Dev\\_features](#) ([l4\\_uint32\\_t](#) v)  
*Make Features from a raw bitmap.*
- constexpr [ring\\_indirect\\_desc\\_bfm\\_t::Val](#) [ring\\_indirect\\_desc](#) () const  
*Get the [ring\\_indirect\\_desc](#) bits (28 to 28) of [raw](#).*
- constexpr [ring\\_indirect\\_desc\\_bfm\\_t::Ref](#) [ring\\_indirect\\_desc](#) ()  
*Get a reference to the [ring\\_indirect\\_desc](#) bits (28 to 28) of [raw](#).*
- constexpr [ring\\_event\\_idx\\_bfm\\_t::Val](#) [ring\\_event\\_idx](#) () const  
*Get the [ring\\_event\\_idx](#) bits (29 to 29) of [raw](#).*
- constexpr [ring\\_event\\_idx\\_bfm\\_t::Ref](#) [ring\\_event\\_idx](#) ()  
*Get a reference to the [ring\\_event\\_idx](#) bits (29 to 29) of [raw](#).*

### Data Fields inherited from [L4virtio::Svr::Dev\\_features](#)

- [l4\\_uint32\\_t](#) [raw](#)  
*The raw value of the features bitmap.*

## 16.406.1 Detailed Description

Virtio console specific feature bits.

Definition at line 18 of file [virtio-console](#).

## 16.406.2 Member Typedef Documentation

### 16.406.2.1 console\_multiport\_bfm\_t

```
typedef cxx::Bitfield<decltype(raw), 1, 1> L4virtio::Svr::Console::Features::console_multiport_bfm_t
```

Device has support for multiple ports.

Type to access the `console_multiport` bits (1 to 1) of `raw`.

Definition at line 25 of file `virtio-console`.

### 16.406.2.2 console\_size\_bfm\_t

```
typedef cxx::Bitfield<decltype(raw), 0, 0> L4virtio::Svr::Console::Features::console_size_bfm_t
```

Configuration `cols` and `rows` are valid.

Type to access the `console_size` bits (0 to 0) of `raw`.

Definition at line 23 of file `virtio-console`.

### 16.406.2.3 emerg\_write\_bfm\_t

```
typedef cxx::Bitfield<decltype(raw), 2, 2> L4virtio::Svr::Console::Features::emerg_write_bfm_t
```

Device has support for emergency write.

Type to access the `emerg_write` bits (2 to 2) of `raw`.

Definition at line 27 of file `virtio-console`.

The documentation for this struct was generated from the following file:

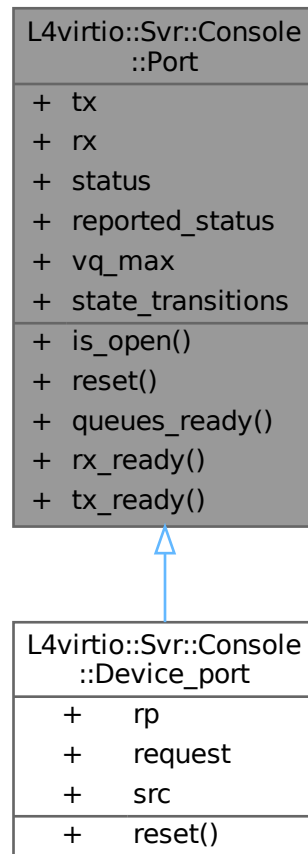
- `I4/I4virtio/server/virtio-console`

## 16.407 L4virtio::Svr::Console::Port Struct Reference

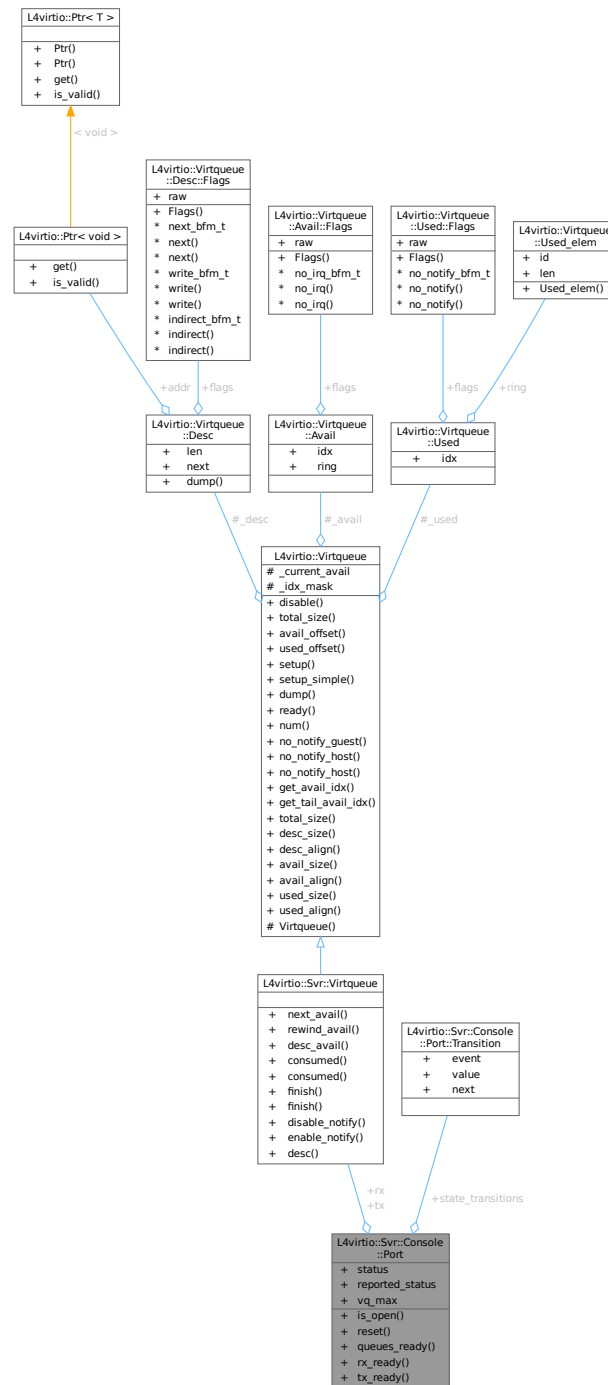
Representation of a Virtio console port.

```
#include <virtio-console>
```

Inheritance diagram for L4virtio::Svr::Console::Port:



Collaboration diagram for L4virtio::Svr::Console::Port:



## Data Structures

- struct [Transition](#)

*State transition from last report state to current state.*

## Public Types

- enum [Port\\_status](#) {  
    [Port\\_disabled](#) = 0 , [Port\\_added](#) , [Port\\_ready](#) , [Port\\_open](#) ,  
    [Port\\_failed](#) , [Port\\_num\\_states](#) }

*Possible states of a virtio console port.*

- enum  
*Size of control queues, also used as default size.*

## Public Member Functions

- bool **is\_open** () const  
*Check that the port is open.*
- virtual void **reset** ()  
*Reset the port to the initial state and disable its virtqueues.*
- bool **queues\_ready** () const  
*Check that both virtqueues are set up correctly.*
- bool **rx\_ready** () const  
*Check that device implementation may write to receive queues.*
- bool **tx\_ready** () const  
*Check that device implementation may read from transmit queues.*

## Data Fields

- [Virtqueue](#) **tx**  
*Receiveq of the port.*
- [Virtqueue](#) **rx**  
*Transmitq of the port.*
- [Port\\_status](#) **status**  
*State the port is in.*
- [Port\\_status](#) **reported\_status**  
*State the port was last reported.*
- unsigned **vq\_max**  
*Maximum queue sizes for this port.*

## Static Public Attributes

- static constexpr [Transition](#) **state\_transitions** [[Port\\_num\\_states](#)][[Port\\_num\\_states](#)]  
*State transition table from last report state to current state.*



### 16.407.3 Field Documentation

#### 16.407.3.1 state\_transitions

```
Transition L4virtio::Svr::Console::Port::state_transitions[Port_num_states][Port_num_states]  
[static], [constexpr]
```

State transition table from last report state to current state.

Not all transitions can be made directly. For example, if the last reported state was `Port_disabled` and the current state is `Port_open`, the device has to send two messages: `Control_message::Device_add` and `Control_message::Port_open`. This is expressed by going through an intermediate state (`Port_ready`) on the reporting side.

For the purpose of the driver there are only three coarse states:

1. The port does not exist (`Port_disabled`).
2. The port exists but is closed on the device side (`Port_added`, `Port_ready`, `Port_failed`).
3. The port exists and is open on the device side (`Port_open`).

The state transition table with `Port_added`, `Port_ready` and `Port_failed` as current state are thus identical.

Definition at line 195 of file `virtio-console`.

The documentation for this struct was generated from the following file:

- `l4/l4virtio/server/virtio-console`

## 16.408 L4virtio::Svr::Console::Port::Transition Struct Reference

State transition from last report state to current state.

```
#include <virtio-console>
```

Collaboration diagram for L4virtio::Svr::Console::Port::Transition:

L4virtio::Svr::Console ::Port::Transition	
+	event
+	value
+	next

### Data Fields

- [l4\\_int16\\_t](#) **event**  
*[Control\\_message::Events](#) or  $< 0$  if no event is sent.*
- [l4\\_uint16\\_t](#) **value**  
*Extra information.*
- [Port\\_status](#) **next**  
*Next [Port\\_status](#) state.*

## 16.408.1 Detailed Description

State transition from last report state to current state.

Definition at line [169](#) of file [virtio-console](#).

The documentation for this struct was generated from the following file:

- [l4/l4virtio/server/virtio-console](#)

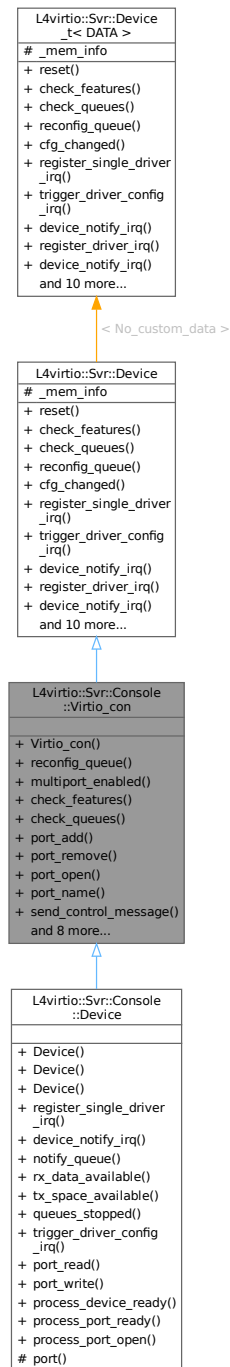
## 16.409 L4virtio::Svr::Console::Virtio\_con Class Reference

Base class implementing a virtio console functionality.

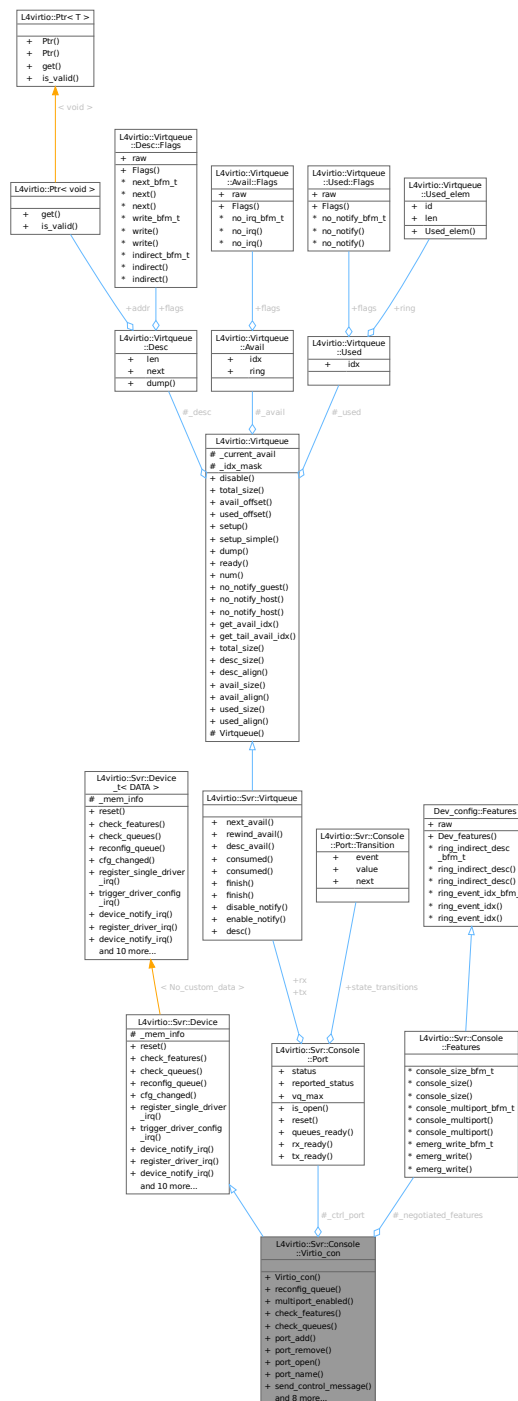
```
#include <virtio-console>
```



Inheritance diagram for L4virtio::Svr::Console::Virtio\_con:



Collaboration diagram for L4virtio::Svr::Console::Virtio\_con:



## Public Member Functions

- **Virtio\_con** (unsigned max\_ports, bool enable\_multiport)  
Create a new multiport console device.
- int **reconfig\_queue** (unsigned index) override  
callback for client queue-config request
- bool **multiport\_enabled** () const

- Return true if the multiport feature is enabled and control queues are available.*

  - bool **check\_features** (void) override  
*callback for checking the subset of accepted features*
  - bool **check\_queues** () override  
*callback for checking if the queues at DRIVER\_OK transition*
  - int **port\_add** (unsigned idx)  
*Send a DEVICE\_ADD message and update the internal state.*
  - int **port\_remove** (unsigned idx)  
*Send a DEVICE\_REMOVE message and update the internal state.*
  - int **port\_open** (unsigned idx, bool open)  
*Send a PORT\_OPEN message and update the internal state.*
  - int **port\_name** (unsigned idx, char const \*name)  
*Send a PORT\_NAME message to announce the port name.*
  - int **send\_control\_message** (l4\_uint32\_t idx, l4\_uint16\_t event, l4\_uint16\_t value=0, const char \*name=0)  
*Send control message to driver.*
  - int **handle\_control\_message** ()  
*Handle control message received from the driver.*
  - void **reset** () override  
*reset callback, called for doing a device reset*
  - virtual void **reset\_device** ()  
*Reset the state of the actual console device.*
  - virtual void **notify\_queue** (Virtqueue \*queue)=0  
*Notify queue of available data.*
  - virtual Port \* **port** (unsigned port)=0  
*Return the specified port.*
  - virtual void **process\_device\_ready** (l4\_uint16\_t value)=0  
*Callback called on DEVICE\_READY event.*
  - virtual void **process\_port\_ready** (l4\_uint32\_t id, l4\_uint16\_t value)  
*Callback called on PORT\_READY event.*
  - virtual void **process\_port\_open** (l4\_uint32\_t id, l4\_uint16\_t value)=0  
*Callback called on PORT\_OPEN event.*

## Public Member Functions inherited from L4virtio::Svr::Device\_t< No\_custom\_data >

- virtual void **cfg\_changed** (unsigned)  
*callback for client device configuration changes*
- virtual void **register\_single\_driver\_irq** ()  
*callback for registering a single guest IRQ for all queues (old-style)*
- virtual void **trigger\_driver\_config\_irq** ()=0  
*callback for triggering configuration change notification IRQ*
- virtual L4::Cap< L4::Irq > **device\_notify\_irq** () const  
*callback to gather the device notification IRQ (old-style)*
- virtual void **register\_driver\_irq** (unsigned idx)  
*Callback for registering an notification IRQ (multi IRQ).*
- virtual L4::Cap< L4::Irq > **device\_notify\_irq** (unsigned idx)  
*Callback to gather the device notification IRQ (multi IRQ).*
- virtual unsigned **num\_events\_supported** () const  
*Return the highest notification index supported.*
- **Device\_t** (Dev\_config \*dev\_config)  
*Make a device for the given config.*

- `Mem_list` const \* `mem_info` () const  
*Get the memory region list used for this device.*
- void `reset_queue_config` (unsigned idx, unsigned num\_max, bool inc\_generation=false)  
*Trigger reset for the configuration space for queue idx.*
- void `init_mem_info` (unsigned num)  
*Initialize the memory region list to the given maximum.*
- void `device_error` ()  
*Transition device into `DEVICE_NEEDS_RESET` state.*
- bool `setup_queue` (`Virtqueue` \*q, unsigned qn, unsigned num\_max)  
*Enable/disable the specified queue.*
- bool `handle_mem_cmd_write` ()  
*Check for a value in the `cmd` register and handle a write.*
- void `enable_trusted_ds_validation` ()  
*Enable trusted dataspace validation.*
- void `add_trusted_dataspaces` (std::shared\_ptr< Ds\_vector const > ds)  
*Provide a list of trusted dataspaces that can be used for validation.*

### Additional Inherited Members

### Protected Attributes inherited from `L4virtio::Svr::Device_t< No_custom_data >`

- `Mem_list` \_mem\_info  
*Memory region list.*

## 16.409.1 Detailed Description

Base class implementing a virtio console functionality.

It is possible to activate the MULTIPORT feature, in which case incoming control messages need to be dispatched by calling `handle_control_message()`. The derived class must additionally override `process_device_ready()`, `process_port_ready()` and `process_port_open()` to implement the actual behaviour. The derived class has the following responsibilities:

- inform the driver about usable ports once the device is ready as signaled in `process_device_ready()`, see the wrapper `port_add()`.
- inform the driver about unusable ports, see the wrapper `port_remove()`.
- react to open/close events, see the wrapper `port_open()`.

This implementation provides no means to handle interrupts or notify guests, therefore derived classes have to provide this functionality, see `notify_queue()` and `handle_control_message()`. Similarly, all interaction with data queues has to be implemented. Memory for port structures must be managed by the implementor as well.

Use this class as a base to implement your own specific console device.

Definition at line 267 of file `virtio-console`.

## 16.409.2 Constructor & Destructor Documentation

### 16.409.2.1 Virtio\_con()

```
L4virtio::Svr::Console::Virtio_con::Virtio_con (
    unsigned max_ports,
    bool enable_multiport) [inline], [explicit]
```

Create a new multiport console device.

### Parameters

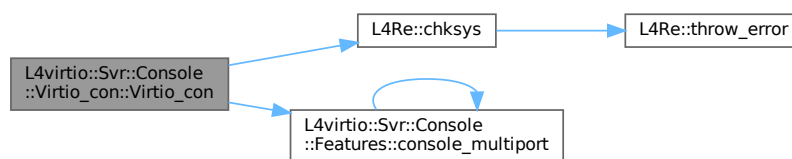
<i>max_ports</i>	Maximum number of ports the device should be able to handle (ignored when <code>enable_multiport</code> is false).
<i>enable_multiport</i>	Enable the control queue for dynamic handling of ports.

Definition at line 293 of file [virtio-console](#).

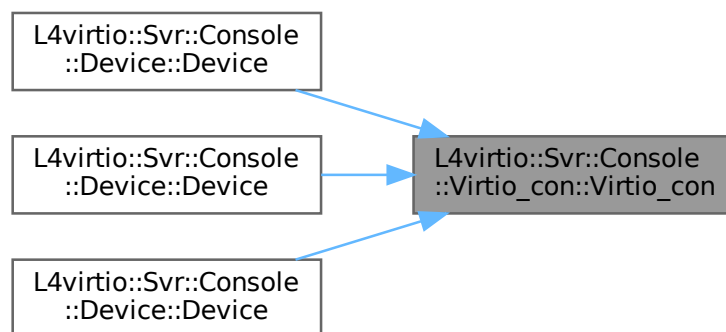
References [L4Re::chksys\(\)](#), [L4virtio::Svr::Console::Features::console\\_multiport\(\)](#), [L4\\_EINVAL](#), [L4VIRTIO\\_ID\\_CONSOLE](#), and [L4virtio::Svr::Dev\\_features::raw](#).

Referenced by [L4virtio::Svr::Console::Device::Device\(\)](#), [L4virtio::Svr::Console::Device::Device\(\)](#), and [L4virtio::Svr::Console::Device::Device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



## 16.409.3 Member Function Documentation

### 16.409.3.1 `handle_control_message()`

```
int L4virtio::Svr::Console::Virtio_con::handle_control_message () [inline]
```

Handle control message received from the driver.

#### Return values

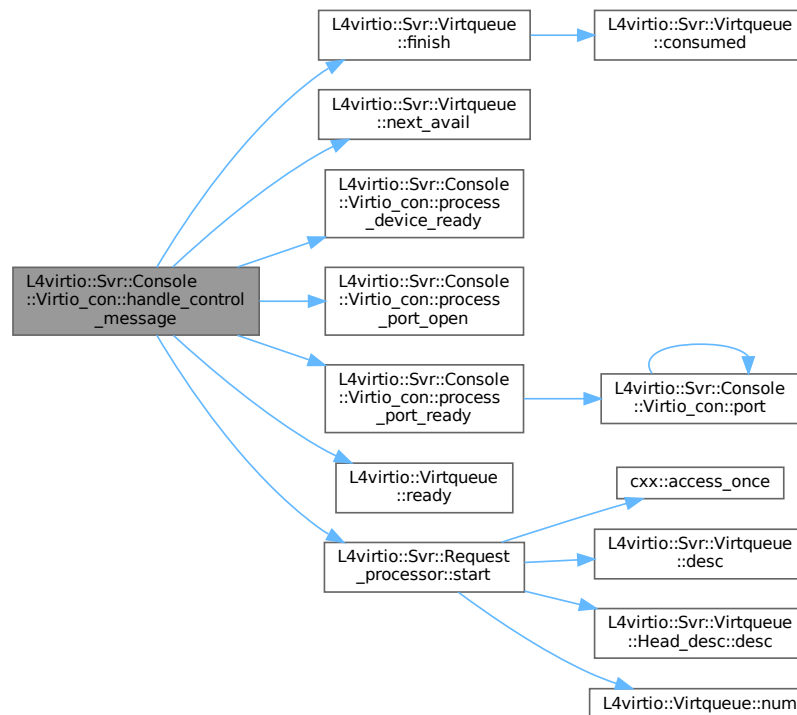
<code>L4_EOK</code>	Message has been handled.
<code>-L4_ENODEV</code>	Control queue is not ready.
<code>-L4_EINVAL</code>	Received an unexpected control event.

This function performs the basic handling of control messages from the driver. It does all necessary work with the control queues and performs some sanity checks. All other work is deferred to the derived class, see `process_device_ready()`, `process_port_ready()` and `process_port_open()`.

Definition at line 536 of file `virtio-console`.

References `L4virtio::Svr::Console::Control_message::Device_ready`, `L4virtio::Svr::Console::Control_message::event`, `L4virtio::Svr::Virtqueue::finish()`, `L4virtio::Svr::Console::Control_message::id`, `L4_EINVAL`, `L4_ENODEV`, `L4_EOK`, `L4virtio::Svr::Console::Control_request::len`, `L4virtio::Svr::Console::Control_request::msg`, `L4virtio::Svr::Virtqueue::next_avail()`, `L4virtio::Svr::Console::Port::Port_disabled`, `L4virtio::Svr::Console::Control_message::Port_open`, `L4virtio::Svr::Console::Port::Port_op`, `L4virtio::Svr::Console::Control_message::Port_ready`, `process_device_ready()`, `process_port_open()`, `process_port_ready()`, `L4virtio::Virtqueue::ready()`, `L4virtio::Svr::Request_processor::start()`, and `L4virtio::Svr::Console::Control_message::value`.

Here is the call graph for this function:



### 16.409.3.2 notify\_queue()

```
virtual void L4virtio::Svr::Console::Virtio_con::notify_queue (
    Virtqueue * queue) [pure virtual]
```

Notify queue of available data.

#### Parameters

<i>queue</i>	<a href="#">Virtqueue</a> to notify.
--------------	--------------------------------------

This callback is called whenever data is sent to `queue`. It is the responsibility of the derived class to perform all necessary notification actions, e.g. triggering guest interrupts.

Implemented in [L4virtio::Svr::Console::Device](#).

### 16.409.3.3 port()

```
virtual Port * L4virtio::Svr::Console::Virtio_con::port (
    unsigned port) [pure virtual]
```

Return the specified port.

#### Parameters

<i>port</i>	<a href="#">Port</a> number.
-------------	------------------------------

#### Precondition

[Port](#) number must be lower than the configured maximum number of ports.

Implemented in [L4virtio::Svr::Console::Device](#).

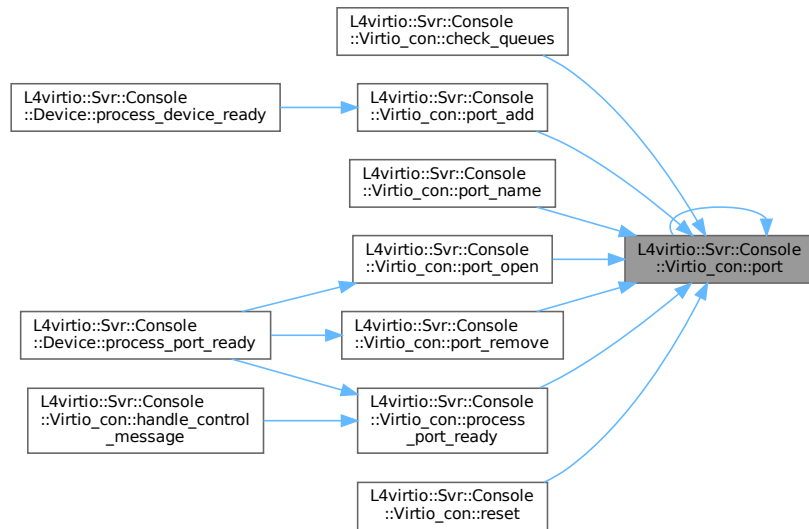
References [port\(\)](#).

Referenced by [check\\_queues\(\)](#), [port\(\)](#), [port\\_add\(\)](#), [port\\_name\(\)](#), [port\\_open\(\)](#), [port\\_remove\(\)](#), [process\\_port\\_ready\(\)](#), and [reset\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 16.409.3.4 port\_add()

```
int L4virtio::Svr::Console::Virtio_con::port_add (
    unsigned idx) [inline]
```

Send a DEVICE\_ADD message and update the internal state.

#### Parameters

<i>idx</i>	Port that should be added.
------------	----------------------------

#### Return values

<i>L4_EOK</i>	Message has been sent.
<i>-L4_EPERM</i>	Invalid state transition.

#### Precondition

*idx* must be smaller than the configured number of ports.

Port must not already exist.

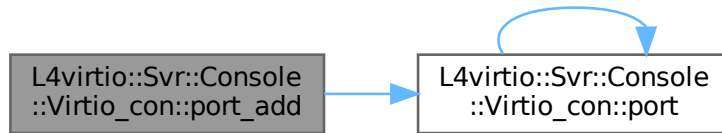
Definition at line 379 of file [virtio-console](#).

References [L4\\_EOK](#), [L4\\_EPERM](#), [port\(\)](#), [L4virtio::Svr::Console::Port::Port\\_added](#), [L4virtio::Svr::Console::Port::Port\\_disabled](#), and [L4virtio::Svr::Console::Port::status](#).

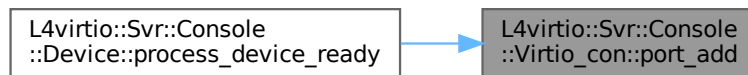


Referenced by [L4virtio::Svr::Console::Device::process\\_device\\_ready\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.409.3.5 port\_name()

```
int L4virtio::Svr::Console::Virtio_con::port_name (
    unsigned idx,
    char const * name) [inline]
```

Send a PORT\_NAME message to announce the port name.

#### Parameters

<i>idx</i>	<a href="#">Port</a> that should be opened or closed.
<i>name</i>	The port name

#### Return values

<i>L4_EOK</i>	Message has been sent.
<i>-L4_EPERM</i>	Control message is not allowed in the current state.

#### Returns

Errors from [send\\_control\\_message\(\)](#)

**Precondition**

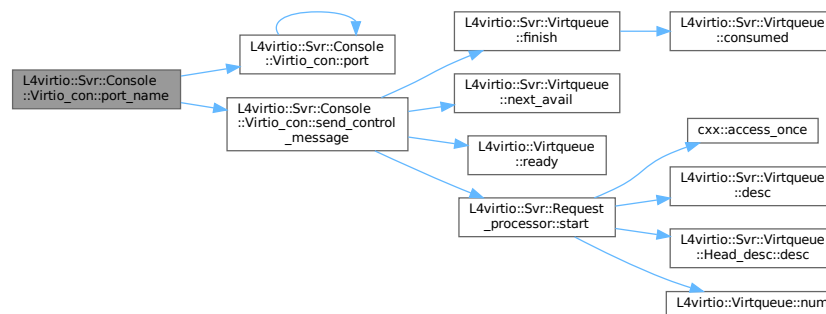
`idx` must be smaller than the configured number of ports.

[Port](#) must already exist.

Definition at line 455 of file [virtio-console](#).

References [L4\\_EPERM](#), [port\(\)](#), [L4virtio::Svr::Console::Port::Port\\_disabled](#), [L4virtio::Svr::Console::Control\\_message::Port\\_name](#), [send\\_control\\_message\(\)](#), and [L4virtio::Svr::Console::Port::status](#).

Here is the call graph for this function:

**16.409.3.6 port\_open()**

```

int L4virtio::Svr::Console::Virtio_con::port_open (
    unsigned idx,
    bool open) [inline]

```

Send a PORT\_OPEN message and update the internal state.

**Parameters**

<i>idx</i>	<a href="#">Port</a> that should be opened or closed.
<i>open</i>	Open or close port.

**Return values**

<i>L4_EOK</i>	Message has been sent.
<i>-L4_EPERM</i>	Invalid state transition.

**Precondition**

`idx` must be smaller than the configured number of ports.

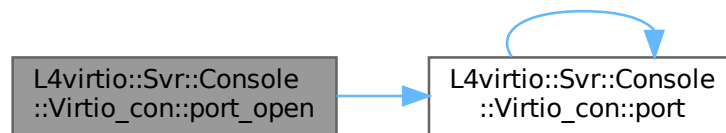
[Port](#) must be ready when opening or open when closing.

Definition at line 428 of file [virtio-console](#).

References [L4\\_EOK](#), [L4\\_EPERM](#), [port\(\)](#), [L4virtio::Svr::Console::Port::Port\\_open](#), [L4virtio::Svr::Console::Port::Port\\_ready](#), and [L4virtio::Svr::Console::Port::status](#).

Referenced by [L4virtio::Svr::Console::Device::process\\_port\\_ready\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**16.409.3.7 port\_remove()**

```
int L4virtio::Svr::Console::Virtio_con::port_remove (
    unsigned idx) [inline]
```

Send a `DEVICE_REMOVE` message and update the internal state.

**Parameters**

<code>idx</code>	<a href="#">Port</a> that should be removed.
------------------	----------------------------------------------

**Return values**

<code>L4_EOK</code>	Message has been sent.
<code>-L4_EPERM</code>	Invalid state transition.

**Precondition**

`idx` must be smaller than the configured number of ports.

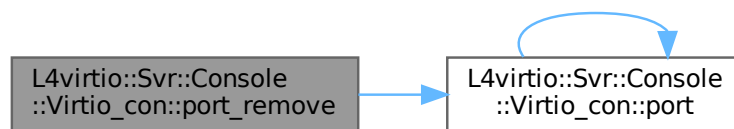
`Port` must already exist.

Definition at line 403 of file `virtio-console`.

References `L4_EOK`, `L4_EPERM`, `port()`, `L4virtio::Svr::Console::Port::Port_disabled`, and `L4virtio::Svr::Console::Port::status`.

Referenced by `L4virtio::Svr::Console::Device::process_port_ready()`.

Here is the call graph for this function:



Here is the caller graph for this function:

**16.409.3.8 process\_device\_ready()**

```
virtual void L4virtio::Svr::Console::Virtio_con::process_device_ready (
    l4_uint16_t value) [pure virtual]
```

Callback called on `DEVICE_READY` event.

**Parameters**

<i>value</i>	The value field of the control message, indicating if the initialization was successful.
--------------	------------------------------------------------------------------------------------------

Needs to be overridden by the derived class if the MULTIPORT feature is enabled. Control messages may be sent only after the driver has successfully initialized the device.

Implemented in [L4virtio::Svr::Console::Device](#).

Referenced by [handle\\_control\\_message\(\)](#).

Here is the caller graph for this function:



### 16.409.3.9 process\_port\_open()

```
virtual void L4virtio::Svr::Console::Virtio_con::process_port_open (
    l4_uint32_t id,
    l4_uint16_t value) [pure virtual]
```

Callback called on PORT\_OPEN event.

#### Parameters

<i>id</i>	The id field of the control message, i.e. the port number.
<i>value</i>	The value field of the control message, indicating if the port was opened or closed.

Signal that an application has opened the port. Can to be overridden by the derived class if the MULTIPORT feature is enabled.

Implemented in [L4virtio::Svr::Console::Device](#).

Referenced by [handle\\_control\\_message\(\)](#).

Here is the caller graph for this function:



### 16.409.3.10 process\_port\_ready()

```
virtual void L4virtio::Svr::Console::Virtio_con::process_port_ready (
    14_uint32_t id,
    14_uint16_t value) [inline], [virtual]
```

Callback called on PORT\_READY event.

#### Parameters

<i>id</i>	The id field of the control message, i.e. the port number.
<i>value</i>	The value field of the control message, indicating if the initialization was successful.

May be overridden by the derived class if the MULTIPORT feature is enabled. This default implementation just sets the status of the port according to the driver message.

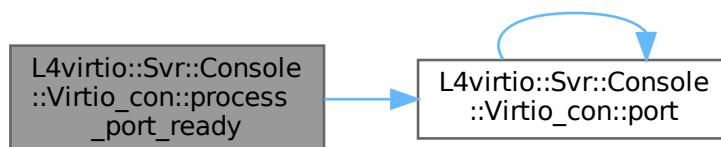
Reimplemented in [L4virtio::Svr::Console::Device](#).

Definition at line 698 of file [virtio-console](#).

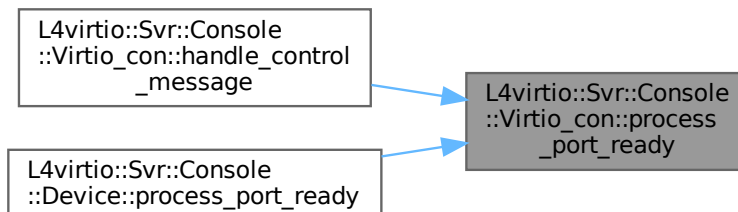
References [port\(\)](#), [L4virtio::Svr::Console::Port::Port\\_added](#), [L4virtio::Svr::Console::Port::Port\\_failed](#), [L4virtio::Svr::Console::Port::Port\\_ready](#), and [L4virtio::Svr::Console::Port::status](#).

Referenced by [handle\\_control\\_message\(\)](#), and [L4virtio::Svr::Console::Device::process\\_port\\_ready\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



**16.409.3.11 reset\_device()**

```
virtual void L4virtio::Svr::Console::Virtio_con::reset_device () [inline], [virtual]
```

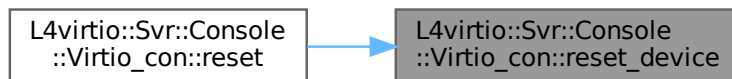
Reset the state of the actual console device.

This callback is called at the end of `reset()`, allowing the derived class to reset internal state.

Definition at line 652 of file `virtio-console`.

Referenced by `reset()`.

Here is the caller graph for this function:

**16.409.3.12 send\_control\_message()**

```
int L4virtio::Svr::Console::Virtio_con::send_control_message (
    14_uint32_t idx,
    14_uint16_t event,
    14_uint16_t value = 0,
    const char * name = 0) [inline]
```

Send control message to driver.

**Parameters**

<i>idx</i>	Port number.
<i>event</i>	Kind of control event.
<i>value</i>	Extra information for the event.
<i>name</i>	Name to be used for Port_name message

**Return values**

<i>L4_EOK</i>	Message has been sent.
<i>-L4_ENODEV</i>	Control queue is not ready.
<i>-L4_EBUSY</i>	Currently no descriptor available in the control queue.
<i>-L4_ENOMEM</i>	Client-issued descriptor too small. Device will be set to failed state.

**Precondition**

`port` must be smaller than the configured number of ports.

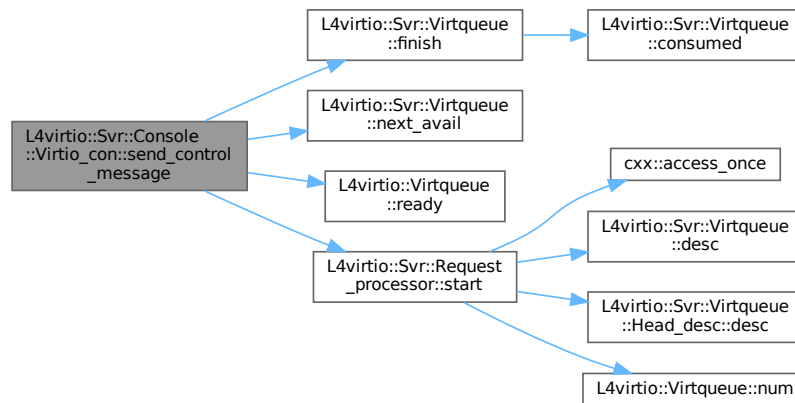
The convenience functions `port_add()`, `port_remove()` and `port_open()` should cover the most use cases and are the preferred way of communication with the driver. If you use this function directly, it is your responsibility to guarantee no invalid control messages are sent to the driver.

Definition at line 487 of file `virtio-console`.

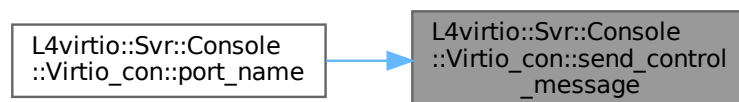
References `L4virtio::Svr::Virtqueue::finish()`, `L4_EBUSY`, `L4_ENODEV`, `L4_ENOMEM`, `L4_EOK`, `L4virtio::Svr::Console::Control_request::msg`, `L4virtio::Svr::Virtqueue::next_avail()`, `L4virtio::Svr::Console::Control_message::Port_name`, `L4virtio::Virtqueue::ready()`, and `L4virtio::Svr::Request_processor::start()`.

Referenced by `port_name()`.

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- `l4/l4virtio/server/virtio-console`

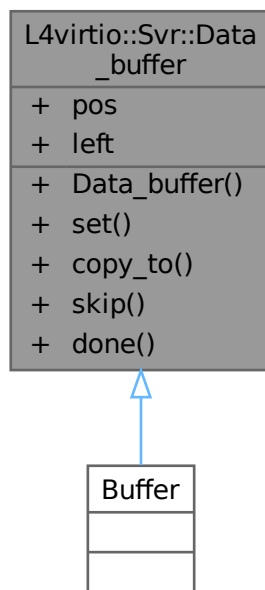


## 16.410 L4virtio::Svr::Data\_buffer Struct Reference

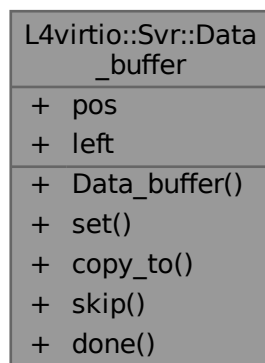
Abstract data buffer.

```
#include <virtio>
```

Inheritance diagram for L4virtio::Svr::Data\_buffer:



Collaboration diagram for L4virtio::Svr::Data\_buffer:



## Public Member Functions

- `template<typename T>`  
`Data_buffer` (`T *p`)  
*Create buffer for object `p`.*
- `template<typename T>`  
`void set` (`T *p`)  
*Set buffer for object `p`.*
- `l4_uint32_t copy_to` (`Data_buffer *dst`, `l4_uint32_t max=UINT_MAX`)  
*Copy contents from this buffer to the destination buffer.*
- `l4_uint32_t skip` (`l4_uint32_t bytes`)  
*Skip given number of bytes in this buffer.*
- `bool done` () `const`  
*Check if there are no more bytes left in the buffer.*

## Data Fields

- `char * pos`  
*Current buffer position.*
- `l4_uint32_t left`  
*Bytes left in buffer.*

## 16.410.1 Detailed Description

Abstract data buffer.

Definition at line 306 of file [virtio](#).

## 16.410.2 Constructor & Destructor Documentation

### 16.410.2.1 Data\_buffer()

```
template<typename T>
L4virtio::Svr::Data_buffer::Data_buffer (
    T * p) [inline], [explicit]
```

Create buffer for object `p`.

## Template Parameters

<code>T</code>	Type of object (implicit)
----------------	---------------------------

## Parameters

<code>p</code>	Pointer to object.
----------------	--------------------

The buffer shall point to the start of the object `p` and the size `left` is `sizeof(T)`.

Definition at line 323 of file [virtio](#).

References [left](#), and [pos](#).

### 16.410.3 Member Function Documentation

#### 16.410.3.1 copy\_to()

```
l4_uint32_t L4virtio::Svr::Data_buffer::copy_to (
    Data_buffer * dst,
    l4_uint32_t max = UINT_MAX) [inline]
```

Copy contents from this buffer to the destination buffer.

##### Parameters

<i>dst</i>	Destination buffer.
<i>max</i>	(optional) Maximum number of bytes to copy.

##### Returns

the number of bytes copied.

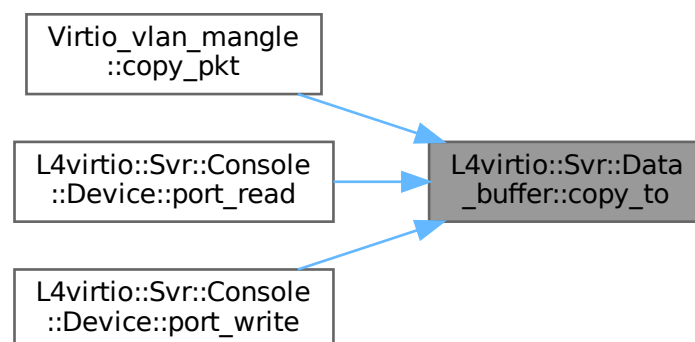
This function copies at most `max` bytes from this to `dst`. If `max` is omitted, copies the maximum number of bytes available that fit `dst`.

Definition at line 354 of file [virtio](#).

References [left](#), and [pos](#).

Referenced by [Virtio\\_vlan\\_mangle::copy\\_pkt\(\)](#), [L4virtio::Svr::Console::Device::port\\_read\(\)](#), and [L4virtio::Svr::Console::Device::port\\_w](#)

Here is the caller graph for this function:



**16.410.3.2 done()**

```
bool L4virtio::Svr::Data_buffer::done () const [inline]
```

Check if there are no more bytes left in the buffer.

**Returns**

true if there are no more bytes left in the buffer.

Definition at line 388 of file [virtio](#).

References [left](#).

**16.410.3.3 set()**

```
template<typename T>
void L4virtio::Svr::Data_buffer::set (
    T * p) [inline]
```

Set buffer for object p.

**Template Parameters**

<i>T</i>	Type of object (implicit)
----------	---------------------------

**Parameters**

<i>p</i>	Pointer to object.
----------	--------------------

The buffer shall point to the start of the object p and the size left is sizeof(T).

Definition at line 337 of file [virtio](#).

References [left](#), and [pos](#).

**16.410.3.4 skip()**

```
l4_uint32_t L4virtio::Svr::Data_buffer::skip (
    l4_uint32_t bytes) [inline]
```

Skip given number of bytes in this buffer.

**Parameters**

<i>bytes</i>	Number of bytes that shall be skipped.
--------------	----------------------------------------

### Returns

The number of bytes skipped.

Try to skip the given number of bytes in this buffer, if there are less bytes left in the buffer that given then at most left bytes are skipped and the amount is returned.

Definition at line 375 of file [virtio](#).

References [left](#), and [pos](#).

Referenced by [Virtio\\_vlan\\_mangle::copy\\_pkt\(\)](#).

Here is the caller graph for this function:



The documentation for this struct was generated from the following file:

- l4/l4virtio/server/virtio

## 16.411 L4virtio::Svr::Dev\_config Class Reference

Abstraction for L4-Virtio device config memory.

```
#include <l4virtio>
```

Collaboration diagram for L4virtio::Svr::Dev\_config:



## Public Member Functions

- [Dev\\_config](#) ([l4\\_uint32\\_t](#) vendor, [l4\\_uint32\\_t](#) device, unsigned cfg\_size, [l4\\_uint32\\_t](#) num\_queues=0)  
*Create a L4-Virtio config data space.*
- [Dev\\_config](#) (Cfg\_cap const &cfg, [l4\\_addr\\_t](#) cfg\_offset, [l4\\_uint32\\_t](#) vendor, [l4\\_uint32\\_t](#) device, unsigned cfg\_size, [l4\\_uint32\\_t](#) num\_queues=0)  
*Setup an L4-Virtio config space in an existing data space.*
- [l4\\_uint32\\_t num\\_queues](#) () const  
*Return the number of queues currently usable.*
- [l4\\_uint32\\_t guest\\_features](#) (unsigned idx) const  
*Return a specific set of guest features.*
- [l4\\_uint32\\_t negotiated\\_features](#) (unsigned idx) const  
*Compute a specific set of negotiated features.*
- [Status status](#) () const  
*Get current device status (trusted).*
- [l4\\_uint32\\_t get\\_cmd](#) () const  
*Get the value from the cmd register.*
- void [reset\\_cmd](#) ()  
*Reset the cmd register after execution of a command.*
- void [set\\_status](#) ([Status status](#))  
*Set device status register.*
- void [add\\_irq\\_status](#) ([l4\\_uint32\\_t status](#))  
*Adds irq status bit.*
- void [set\\_device\\_needs\\_reset](#) ()  
*Set DEVICE\_NEEDS\_RESET bit in device status register.*

- bool [change\\_queue\\_config](#) ([l4\\_uint32\\_t](#) num\_queues)  
*Setup new queue configuration.*
- [l4virtio\\_config\\_queue\\_t](#) volatile const \* [qconfig](#) (unsigned index) const  
*Get queue read-only config data for queue with the given index.*
- void [reset\\_hdr](#) (bool inc\_generation=false) const  
*Reset the config header to the initial contents.*
- bool [reset\\_queue](#) (unsigned index, unsigned num\_max, bool inc\_generation=false) const  
*Reset queue config for the given queue.*
- [l4virtio\\_config\\_hdr\\_t](#) const volatile \* [hdr](#) () const  
*Get a read-only pointer to the config header.*
- [L4::Cap](#)< [L4Re::Dataspace](#) > [ds](#) () const  
*Get data-space capability for the shared config data space.*
- [l4\\_addr\\_t](#) [ds\\_offset](#) () const  
*Return the offset into the config dataspace where the device configuration starts.*

### 16.411.1 Detailed Description

Abstraction for L4-Virtio device config memory.

Virtio defines a device configuration mechanism, L4-Virtio implements this mechanism based on shared memory a [set\\_status\(\)](#) and a [config\\_queue\(\)](#) call. This class provides an abstraction for L4-Virtio host implementations to establish such a shared memory data space and providing the necessary contents and access functions.

Definition at line 52 of file [l4virtio](#).

### 16.411.2 Constructor & Destructor Documentation

#### 16.411.2.1 Dev\_config() [1/2]

```
L4virtio::Svr::Dev_config::Dev_config (
    l4\_uint32\_t vendor,
    l4\_uint32\_t device,
    unsigned cfg_size,
    l4\_uint32\_t num_queues = 0) [inline]
```

Create a L4-Virtio config data space.

#### Parameters

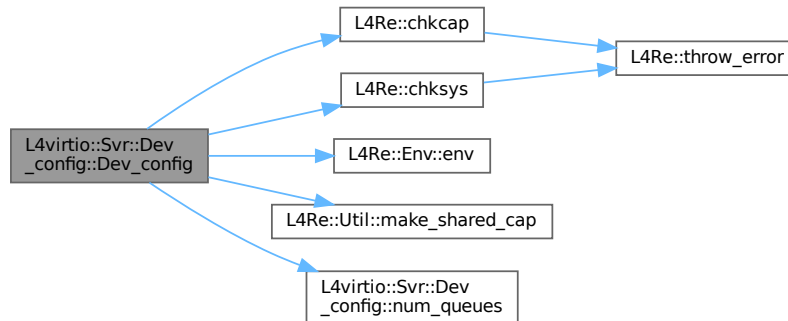
<i>vendor</i>	The vendor ID to store in config header.
<i>device</i>	The device ID to store in config header.
<i>cfg_size</i>	The size of the device-specific config data in bytes.
<i>num_queues</i>	The number of queues provided by the device.

This constructor allocates a data space used for L4-virtio config attaches the data space to the local address space and writes the initial contents to the config header.

Definition at line 112 of file [l4virtio](#).

References [L4Re::chkcap\(\)](#), [L4Re::chksys\(\)](#), [L4Re::Env::env\(\)](#), [L4\\_PAGESIZE](#), [L4Re::Util::make\\_shared\\_cap\(\)](#), and [num\\_queues\(\)](#).

Here is the call graph for this function:



### 16.411.2.2 Dev\_config() [2/2]

```

L4virtio::Svr::Dev_config::Dev_config (
    Cfg_cap const & cfg,
    14_addr_t  cfg_offset,
    14_uint32_t vendor,
    14_uint32_t device,
    unsigned  cfg_size,
    14_uint32_t num_queues = 0) [inline]
  
```

Setup an L4-Virtio config space in an existing data space.

#### Parameters

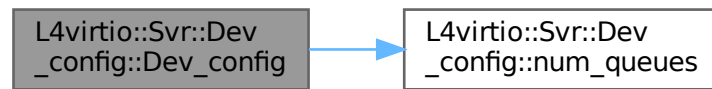
<i>cfg</i>	Dataspace that should hold the L4-Virtio configuration.
<i>cfg_offset</i>	Offset into the dataspace where the configuration starts.
<i>vendor</i>	The vendor ID to store in config header.
<i>device</i>	The device ID to store in config header.
<i>cfg_size</i>	The size of the device-specific config data in bytes.
<i>num_queues</i>	The number of queues provided by the device.

Definition at line 146 of file [l4virtio](#).

References [L4\\_PAGESIZE](#), and [num\\_queues\(\)](#).



Here is the call graph for this function:



### 16.411.3 Member Function Documentation

#### 16.411.3.1 add\_irq\_status()

```
void L4virtio::Svr::Dev_config::add_irq_status (
    14_uint32_t status) [inline]
```

Adds irq status bit.

##### Parameters

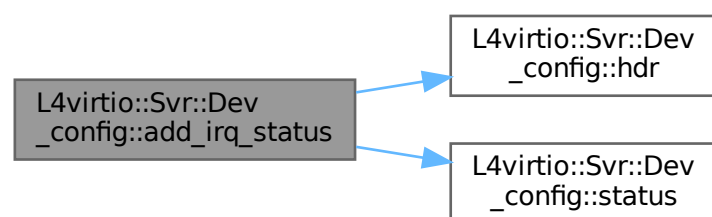
<i>status</i>	The value to add to the irq status register.
---------------	----------------------------------------------

This function adds the status bit to the irq status register.

Definition at line 265 of file `l4virtio`.

References `hdr()`, and `status()`.

Here is the call graph for this function:



#### 16.411.3.2 change\_queue\_config()

```
bool L4virtio::Svr::Dev_config::change_queue_config (
    14_uint32_t num_queues) [inline]
```

Setup new queue configuration.

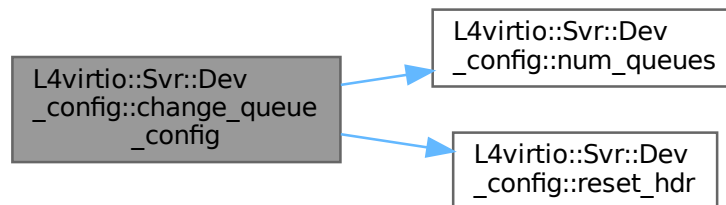
##### Parameters

<code>num_queues</code>	The number of queues provided by the device.
-------------------------	----------------------------------------------

Definition at line 286 of file [l4virtio](#).

References [L4\\_PAGESIZE](#), [num\\_queues\(\)](#), and [reset\\_hdr\(\)](#).

Here is the call graph for this function:



### 16.411.3.3 ds()

```
L4::Cap< L4Re::Dataspace > L4virtio::Svr::Dev_config::ds () const [inline]
```

Get data-space capability for the shared config data space.

Returns

Capability for the shared config data space.

Definition at line 375 of file [l4virtio](#).

### 16.411.3.4 get\_cmd()

```
l4_uint32_t L4virtio::Svr::Dev_config::get_cmd () const [inline]
```

Get the value from the `cmd` register.

Note, the most significant eight bits are the command (0 is nothing to do). The upper eight bit are reset to zero after the command was handled.

Definition at line 230 of file [l4virtio](#).

References [hdr\(\)](#).

Here is the call graph for this function:



### 16.411.3.5 guest\_features()

```
l4_uint32_t L4virtio::Svr::Dev_config::guest_features (
    unsigned idx) const [inline]
```

Return a specific set of guest features.

#### Parameters

<i>idx</i>	Index into the guest features array.
------------	--------------------------------------

#### Return values

<i>The</i>	selected set of guest features.
------------	---------------------------------

This function returns a specific 32bit set of features enabled by the guest/driver. *idx* is the index in the guest features array, resp. the 32 bit set to return.

Definition at line 198 of file [l4virtio](#).

### 16.411.3.6 hdr()

```
l4virtio_config_hdr_t const volatile * L4virtio::Svr::Dev_config::hdr () const [inline]
```

Get a read-only pointer to the config header.

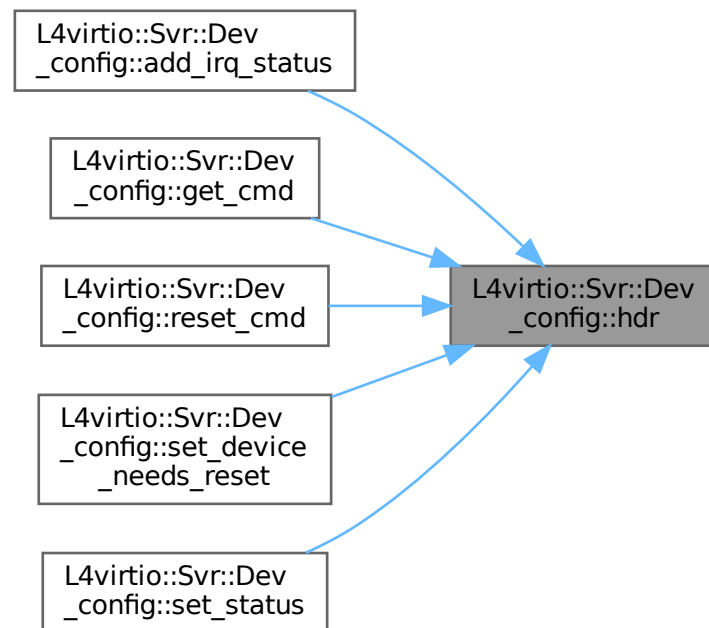
#### Returns

Read-only pointer to the shared config header.

Definition at line 368 of file [l4virtio](#).

Referenced by [add\\_irq\\_status\(\)](#), [get\\_cmd\(\)](#), [reset\\_cmd\(\)](#), [set\\_device\\_needs\\_reset\(\)](#), and [set\\_status\(\)](#).

Here is the caller graph for this function:



### 16.411.3.7 negotiated\_features()

```
l4_uint32_t L4virtio::Svr::Dev_config::negotiated_features (
    unsigned idx) const [inline]
```

Compute a specific set of negotiated features.

#### Parameters

<i>idx</i>	Index into the guest/host features array.
------------	-------------------------------------------

#### Return values

<i>The</i>	selected set of negotiated features.
------------	--------------------------------------

This function returns a specific 32-bit set of features negotiated by the guest/driver and host/device. *idx* is the index in the guest/host features array, resp. the 32-bit set to return.

Definition at line 212 of file [l4virtio](#).

### 16.411.3.8 qconfig()

```
l4virtio_config_queue_t volatile const * L4virtio::Svr::Dev_config::qconfig (
    unsigned index) const [inline]
```

Get queue read-only config data for queue with the given *index*.

#### Parameters

<i>index</i>	The index of the queue.
--------------	-------------------------

#### Returns

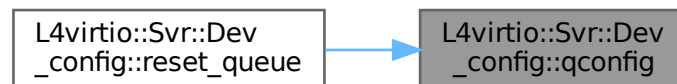
Read-only pointer to the config of the queue with the given *index*, or NULL if *index* is out of range.

Definition at line 303 of file [l4virtio](#).

References [L4\\_UNLIKELY](#).

Referenced by [reset\\_queue\(\)](#).

Here is the caller graph for this function:



### 16.411.3.9 reset\_cmd()

```
void L4virtio::Svr::Dev_config::reset_cmd () [inline]
```

Reset the `cmd` register after execution of a command.

This function resets the `cmd` register in order for the client to detect that the command was executed by the device.

Definition at line 241 of file [l4virtio](#).

References [hdr\(\)](#).

Here is the call graph for this function:



### 16.411.3.10 reset\_queue()

```
bool L4virtio::Svr::Dev_config::reset_queue (
    unsigned index,
    unsigned num_max,
    bool inc_generation = false) const [inline]
```

Reset queue config for the given queue.

#### Parameters

<i>index</i>	The index of the queue to reset.
<i>num_max</i>	The maximum number of descriptors supported by this queue.
<i>inc_generation</i>	The config generation will be incremented when this is true.

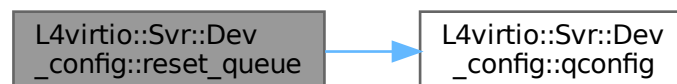
#### Returns

true on success, or false when *index* is out of range.

Definition at line 345 of file [l4virtio](#).

References [L4\\_UNLIKELY](#), [l4virtio\\_config\\_queue\\_t::num](#), [l4virtio\\_config\\_queue\\_t::num\\_max](#), [qconfig\(\)](#), and [l4virtio\\_config\\_queue\\_t::ready](#).

Here is the call graph for this function:



### 16.411.3.11 set\_device\_needs\_reset()

```
void L4virtio::Svr::Dev_config::set_device_needs_reset () [inline]
```

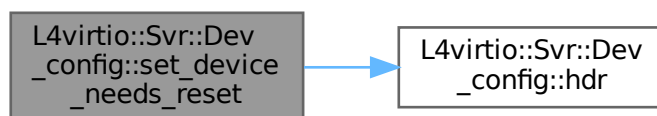
Set `DEVICE_NEEDS_RESET` bit in device status register.

This function sets the internal status register and also the status register in the shared memory to `DEVICE_NEEDS_RESET`.

Definition at line 276 of file [l4virtio](#).

References [hdr\(\)](#).

Here is the call graph for this function:



### 16.411.3.12 set\_status()

```
void L4virtio::Svr::Dev_config::set_status (  
    Status status) [inline]
```

Set device status register.

#### Parameters

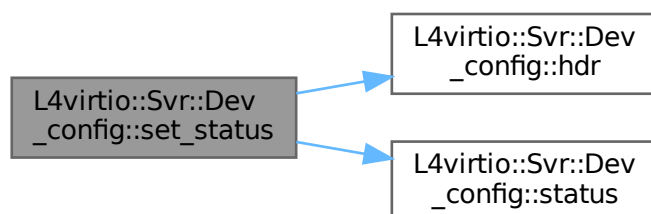
<i>status</i>	The new value for the device status register.
---------------	-----------------------------------------------

This function sets the internal status register and also the status register in the shared memory to *status*.

Definition at line 253 of file [l4virtio](#).

References [hdr\(\)](#), and [status\(\)](#).

Here is the call graph for this function:



**16.411.3.13 status()**

```
Status L4virtio::Svr::Dev_config::status () const [inline]
```

Get current device status (trusted).

**Returns**

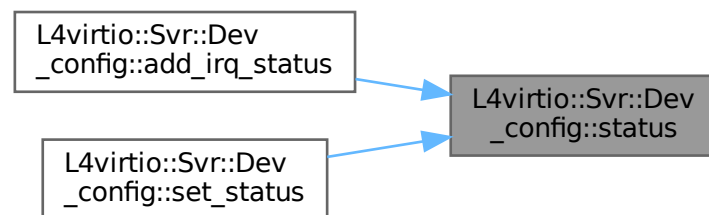
Current device status register (trusted).

The status returned by this function is value stored internally and cannot be written by the guest (i.e., the value can be taken as trusted.)

Definition at line 222 of file [l4virtio](#).

Referenced by [add\\_irq\\_status\(\)](#), and [set\\_status\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- `l4/l4virtio/server/l4virtio`

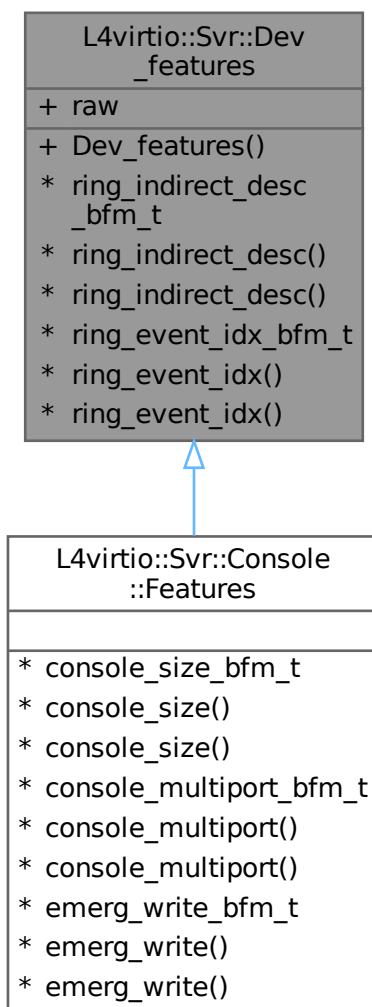
**16.412 L4virtio::Svr::Dev\_features Struct Reference**

Type for device feature bitmap.

```
#include <virtio>
```



Inheritance diagram for L4virtio::Svr::Dev\_features:



Collaboration diagram for L4virtio::Svr::Dev\_features:

L4virtio::Svr::Dev_features
+ raw
+ Dev_features()
* ring_indirect_desc_bfm_t
* ring_indirect_desc()
* ring_indirect_desc()
* ring_event_idx_bfm_t
* ring_event_idx()
* ring_event_idx()

## Public Member Functions

- **Dev\_features** ([l4\\_uint32\\_t](#) v)  
*Make Features from a raw bitmap.*

## Data Fields

- [l4\\_uint32\\_t](#) raw  
*The raw value of the features bitmap.*
- typedef [cxx::Bitfield](#)< decltype(raw), 28, 28 > **ring\_indirect\_desc\_bfm\_t**  
*Type to access the [ring\\_indirect\\_desc](#) bits (28 to 28) of raw.*
- constexpr [ring\\_indirect\\_desc\\_bfm\\_t::Val](#) **ring\_indirect\_desc** () const  
*Get the [ring\\_indirect\\_desc](#) bits (28 to 28) of raw.*
- constexpr [ring\\_indirect\\_desc\\_bfm\\_t::Ref](#) **ring\_indirect\_desc** ()  
*Get a reference to the [ring\\_indirect\\_desc](#) bits (28 to 28) of raw.*
- typedef [cxx::Bitfield](#)< decltype(raw), 29, 29 > **ring\_event\_idx\_bfm\_t**  
*Type to access the [ring\\_event\\_idx](#) bits (29 to 29) of raw.*
- constexpr [ring\\_event\\_idx\\_bfm\\_t::Val](#) **ring\_event\_idx** () const  
*Get the [ring\\_event\\_idx](#) bits (29 to 29) of raw.*
- constexpr [ring\\_event\\_idx\\_bfm\\_t::Ref](#) **ring\_event\_idx** ()  
*Get a reference to the [ring\\_event\\_idx](#) bits (29 to 29) of raw.*

### 16.412.1 Detailed Description

Type for device feature bitmap.

Definition at line 66 of file [virtio](#).

The documentation for this struct was generated from the following file:

- l4/l4virtio/server/virtio

## 16.413 L4virtio::Svr::Dev\_status Struct Reference

Type of the device status register.

```
#include <virtio>
```

Collaboration diagram for L4virtio::Svr::Dev\_status:

L4virtio::Svr::Dev_status
+ raw
+ Dev_status()
+ running()
* acknowledged_bfm_t
* acknowledged()
* acknowledged()
* driver_bfm_t
* driver()
* driver()
* driver_ok_bfm_t
* driver_ok()
* driver_ok()
* features_ok_bfm_t
* features_ok()
* features_ok()
* fail_state_bfm_t
* fail_state()
* fail_state()
* device_needs_reset_bfm_t
* device_needs_reset()
* device_needs_reset()
* failed_bfm_t
* failed()
* failed()

## Public Member Functions

- **Dev\_status** ([l4\\_uint32\\_t](#) v)  
*Make Status from raw value.*
- bool **running** () const  
*Check if the device is in running state.*

## Data Fields

- unsigned char **raw**  
*Raw value of the VIRTIO device status register.*
- typedef [cxx::Bitfield](#)< decltype([raw](#)), 0, 0 > **acked\_bfm\_t**  
*Type to access the [acked](#) bits (0 to 0) of [raw](#).*
- constexpr [acked\\_bfm\\_t::Val](#) **acked** () const  
*Get the [acked](#) bits (0 to 0) of [raw](#).*
- constexpr [acked\\_bfm\\_t::Ref](#) **acked** ()  
*Get a reference to the [acked](#) bits (0 to 0) of [raw](#).*
- typedef [cxx::Bitfield](#)< decltype([raw](#)), 1, 1 > **driver\_bfm\_t**  
*Type to access the [driver](#) bits (1 to 1) of [raw](#).*
- constexpr [driver\\_bfm\\_t::Val](#) **driver** () const  
*Get the [driver](#) bits (1 to 1) of [raw](#).*
- constexpr [driver\\_bfm\\_t::Ref](#) **driver** ()  
*Get a reference to the [driver](#) bits (1 to 1) of [raw](#).*
- typedef [cxx::Bitfield](#)< decltype([raw](#)), 2, 2 > **driver\_ok\_bfm\_t**  
*Type to access the [driver\\_ok](#) bits (2 to 2) of [raw](#).*
- constexpr [driver\\_ok\\_bfm\\_t::Val](#) **driver\_ok** () const  
*Get the [driver\\_ok](#) bits (2 to 2) of [raw](#).*
- constexpr [driver\\_ok\\_bfm\\_t::Ref](#) **driver\_ok** ()  
*Get a reference to the [driver\\_ok](#) bits (2 to 2) of [raw](#).*
- typedef [cxx::Bitfield](#)< decltype([raw](#)), 3, 3 > **features\_ok\_bfm\_t**  
*Type to access the [features\\_ok](#) bits (3 to 3) of [raw](#).*
- constexpr [features\\_ok\\_bfm\\_t::Val](#) **features\_ok** () const  
*Get the [features\\_ok](#) bits (3 to 3) of [raw](#).*
- constexpr [features\\_ok\\_bfm\\_t::Ref](#) **features\_ok** ()  
*Get a reference to the [features\\_ok](#) bits (3 to 3) of [raw](#).*
- typedef [cxx::Bitfield](#)< decltype([raw](#)), 6, 7 > **fail\_state\_bfm\_t**  
*Type to access the [fail\\_state](#) bits (6 to 7) of [raw](#).*
- constexpr [fail\\_state\\_bfm\\_t::Val](#) **fail\_state** () const  
*Get the [fail\\_state](#) bits (6 to 7) of [raw](#).*
- constexpr [fail\\_state\\_bfm\\_t::Ref](#) **fail\_state** ()  
*Get a reference to the [fail\\_state](#) bits (6 to 7) of [raw](#).*
- typedef [cxx::Bitfield](#)< decltype([raw](#)), 6, 6 > **device\_needs\_reset\_bfm\_t**

Type to access the *device\_needs\_reset* bits (6 to 6) of *raw*.

- constexpr [device\\_needs\\_reset\\_bfm\\_t::Val](#) **device\_needs\_reset** () const

Get the *device\_needs\_reset* bits (6 to 6) of *raw*.

- constexpr [device\\_needs\\_reset\\_bfm\\_t::Ref](#) **device\_needs\_reset** ()

Get a reference to the *device\_needs\_reset* bits (6 to 6) of *raw*.

- typedef [cxx::Bitfield](#)< decltype(*raw*), 7, 7 > **failed\_bfm\_t**

Type to access the *failed* bits (7 to 7) of *raw*.

- constexpr [failed\\_bfm\\_t::Val](#) **failed** () const

Get the *failed* bits (7 to 7) of *raw*.

- constexpr [failed\\_bfm\\_t::Ref](#) **failed** ()

Get a reference to the *failed* bits (7 to 7) of *raw*.

## 16.413.1 Detailed Description

Type of the device status register.

Definition at line 32 of file [virtio](#).

## 16.413.2 Member Function Documentation

### 16.413.2.1 running()

```
bool L4virtio::Svr::Dev_status::running () const [inline]
```

Check if the device is in running state.

#### Returns

true if the device is in running state.

The device is in running state when [acked\(\)](#), [driver\(\)](#), [features\\_ok\(\)](#), and [driver\\_ok\(\)](#) return true, and [device\\_needs\\_reset\(\)](#) and [failed\(\)](#) return false.

Definition at line 57 of file [virtio](#).

References [raw](#).

The documentation for this struct was generated from the following file:

- I4/I4virtio/server/virtio



Collaboration diagram for L4virtio::Svr::Device\_t< DATA >:

L4virtio::Svr::Device_t< DATA >
# _mem_info
+ reset() + check_features() + check_queues() + reconfig_queue() + cfg_changed() + register_single_driver_irq() + trigger_driver_config_irq() + device_notify_irq() + register_driver_irq() + device_notify_irq() and 10 more...

## Public Member Functions

- virtual void **reset** ()=0  
*reset callback, called for doing a device reset*
- virtual bool **check\_features** ()  
*callback for checking the subset of accepted features*
- virtual bool **check\_queues** ()=0  
*callback for checking if the queues at DRIVER\_OK transition*
- virtual int **reconfig\_queue** (unsigned idx)=0  
*callback for client queue-config request*
- virtual void **cfg\_changed** (unsigned)  
*callback for client device configuration changes*
- virtual void **register\_single\_driver\_irq** ()  
*callback for registering a single guest IRQ for all queues (old-style)*
- virtual void **trigger\_driver\_config\_irq** ()=0  
*callback for triggering configuration change notification IRQ*
- virtual [L4::Cap](#)< [L4::Irq](#) > **device\_notify\_irq** () const  
*callback to gather the device notification IRQ (old-style)*
- virtual void [register\\_driver\\_irq](#) (unsigned idx)  
*Callback for registering an notification IRQ (multi IRQ).*
- virtual [L4::Cap](#)< [L4::Irq](#) > **device\_notify\_irq** (unsigned idx)  
*Callback to gather the device notification IRQ (multi IRQ).*
- virtual unsigned **num\_events\_supported** () const  
*Return the highest notification index supported.*

- **Device\_t** ([Dev\\_config](#) \*dev\_config)  
*Make a device for the given config.*
- **Mem\_list** const \* **mem\_info** () const  
*Get the memory region list used for this device.*
- void **reset\_queue\_config** (unsigned idx, unsigned num\_max, bool inc\_generation=false)  
*Trigger reset for the configuration space for queue idx.*
- void **init\_mem\_info** (unsigned num)  
*Initialize the memory region list to the given maximum.*
- void **device\_error** ()  
*Transition device into DEVICE\_NEEDS\_RESET state.*
- bool **setup\_queue** ([Virtqueue](#) \*q, unsigned qn, unsigned num\_max)  
*Enable/disable the specified queue.*
- bool **handle\_mem\_cmd\_write** ()  
*Check for a value in the cmd register and handle a write.*
- void **enable\_trusted\_ds\_validation** ()  
*Enable trusted dataspace validation.*
- void **add\_trusted\_dataspaces** (std::shared\_ptr< Ds\_vector const > ds)  
*Provide a list of trusted dataspace that can be used for validation.*

## Protected Attributes

- **Mem\_list** **mem\_info**  
*Memory region list.*

## 16.414.1 Detailed Description

```
template<typename DATA>
class L4virtio::Svr::Device_t< DATA >
```

Server-side L4-VIRTIO device stub.

This stub supports new-style multi-event registration (using `get_device_config()`, `bind()` and `get_device_notification_irq()`).

Definition at line 801 of file [l4virtio](#).

## 16.414.2 Member Function Documentation

### 16.414.2.1 add\_trusted\_dataspaces()

```
template<typename DATA>
void L4virtio::Svr::Device_t< DATA >::add_trusted_dataspaces (
    std::shared_ptr< Ds_vector const > ds) [inline]
```

Provide a list of trusted dataspace that can be used for validation.

#### Parameters

---



<i>ds</i>	list of trusted dataspace.
-----------	----------------------------

Definition at line 1198 of file [l4virtio](#).

#### 16.414.2.2 device\_error()

```
template<typename DATA>
void L4virtio::Svr::Device_t< DATA >::device_error () [inline]
```

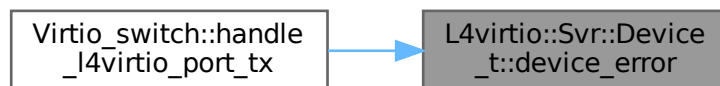
Transition device into DEVICE\_NEEDS\_RESET state.

This function does a full reset, sets the DEVICE\_NEEDS\_RESET bit in the device status register, triggering a guest config IRQ if necessary. The driver still needs to perform its own reset and initialization sequence.

Definition at line 1024 of file [l4virtio](#).

Referenced by [Virtio\\_switch::handle\\_l4virtio\\_port\\_tx\(\)](#).

Here is the caller graph for this function:



#### 16.414.2.3 device\_notify\_irq()

```
template<typename DATA>
virtual L4::Cap< L4::Irq > L4virtio::Svr::Device_t< DATA >::device_notify_irq (
    unsigned idx) [inline], [virtual]
```

Callback to gather the device notification IRQ (multi IRQ).

The default implementation maps to the implementation for single IRQ notification points.

Definition at line 874 of file [l4virtio](#).

#### 16.414.2.4 handle\_mem\_cmd\_write()

```
template<typename DATA>
bool L4virtio::Svr::Device_t< DATA >::handle_mem_cmd_write () [inline]
```

Check for a value in the `cmd` register and handle a write.

This function checks for a value in the `cmd` register and executes the command if there is any, or returns false if there was no command.

Execution of the command is signaled by a zero in the `cmd` register.

Definition at line 1154 of file [l4virtio](#).

#### 16.414.2.5 `init_mem_info()`

```
template<typename DATA>
void L4virtio::Svr::Device_t< DATA >::init_mem_info (
    unsigned num) [inline]
```

Initialize the memory region list to the given maximum.

##### Parameters

<i>num</i>	Maximum number of memory regions that can be managed.
------------	-------------------------------------------------------

Definition at line 1012 of file [l4virtio](#).

#### 16.414.2.6 `register_driver_irq()`

```
template<typename DATA>
virtual void L4virtio::Svr::Device_t< DATA >::register_driver_irq (
    unsigned idx) [inline], [virtual]
```

Callback for registering an notification IRQ (multi IRQ).

The default implementation maps to the implementation for single IRQ notification points.

Definition at line 860 of file [l4virtio](#).

#### 16.414.2.7 `reset_queue_config()`

```
template<typename DATA>
void L4virtio::Svr::Device_t< DATA >::reset_queue_config (
    unsigned idx,
    unsigned num_max,
    bool inc_generation = false) [inline]
```

Trigger reset for the configuration space for queue *idx*.

##### Parameters

<i>idx</i>	The queue index to reset.
<i>num_max</i>	Maximum number of entries in this queue.
<i>inc_generation</i>	The config generation will be incremented when this is true.

This function resets the driver-readable configuration space for the queue with the given index. The queue configuration is reset to all 0, and the maximum number of entries in the queue is set to *num\_max*.

Definition at line 1002 of file [l4virtio](#).

### 16.414.2.8 setup\_queue()

```
template<typename DATA>
bool L4virtio::Svr::Device_t< DATA >::setup_queue (
    Virtqueue * q,
    unsigned qn,
    unsigned num_max) [inline]
```

Enable/disable the specified queue.

#### Parameters

---

<i>q</i>	Pointer to the ring that represents the virtqueue internally.
<i>qn</i>	Index of the queue.
<i>num_max</i>	Maximum number of supported entries in this queue.

### Returns

true for success.

- This function calculates the parameters of the virtqueue from the clients configuration space values, checks the accessibility of the queue data structures and initializes *q* to ready state when all checks succeeded.

Definition at line 1047 of file [l4virtio](#).

The documentation for this class was generated from the following file:

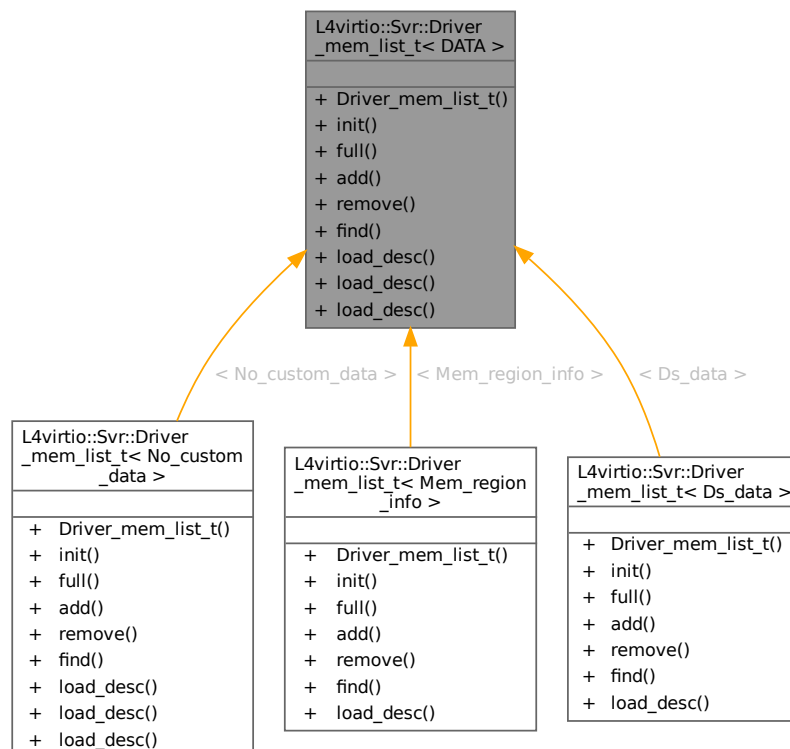
- [l4/l4virtio/server/l4virtio](#)

## 16.415 L4virtio::Svr::Driver\_mem\_list\_t< DATA > Class Template Reference

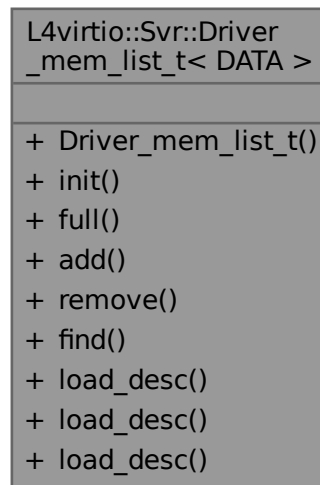
List of driver memory regions assigned to a single L4-VIRTIO transport instance.

```
#include <l4virtio>
```

Inheritance diagram for L4virtio::Svr::Driver\_mem\_list\_t< DATA >:



Collaboration diagram for L4virtio::Svr::Driver\_mem\_list\_t< DATA >:



## Public Types

- typedef [L4Re::Util::Unique\\_cap< L4Re::Dataspace >](#) **Ds\_cap**  
type for storing a data-space capability internally

## Public Member Functions

- **Driver\_mem\_list\_t ()**  
Make an empty, zero capacity list.
- void **init** (unsigned max)  
Make a fresh list with capacity max.
- bool **full** () const
- [Mem\\_region](#) const \* **add** ([l4\\_uint64\\_t](#) drv\_base, [l4\\_umword\\_t](#) size, [l4\\_addr\\_t](#) offset, [Ds\\_cap](#) &&ds)  
Add a new region to the list.
- void **remove** ([Mem\\_region](#) const \*r)  
Remove the given region from the list.
- [Mem\\_region](#) \* **find** ([l4\\_uint64\\_t](#) base, [l4\\_umword\\_t](#) size) const  
Find memory region containing the given driver address region.
- void **load\_desc** ([Virtqueue::Desc](#) const &desc, [Request\\_processor](#) const \*p, [Virtqueue::Desc](#) const \*\*table) const  
Default implementation for loading an indirect descriptor.
- void **load\_desc** ([Virtqueue::Desc](#) const &desc, [Request\\_processor](#) const \*p, [Mem\\_region](#) const \*\*data) const  
Default implementation returning the Driver\_mem\_region.
- template<typename ARG>  
void **load\_desc** ([Virtqueue::Desc](#) const &desc, [Request\\_processor](#) const \*p, ARG \*data) const  
Default implementation returning generic information.

### 16.415.1 Detailed Description

```
template<typename DATA>
class L4virtio::Svr::Driver_mem_list_t< DATA >
```

List of driver memory regions assigned to a single L4-VIRTIO transport instance.

#### Note

The regions added to this list *must* never overlap.

Definition at line 629 of file [l4virtio](#).

### 16.415.2 Member Function Documentation

#### 16.415.2.1 add()

```
template<typename DATA>
Mem_region const * L4virtio::Svr::Driver_mem_list_t< DATA >::add (
    l4_uint64_t drv_base,
    l4_umword_t size,
    l4_addr_t offset,
    Ds_cap && ds) [inline]
```

Add a new region to the list.

#### Parameters

<i>drv_base</i>	Driver base address of the region.
<i>size</i>	Size of the region in bytes.
<i>offset</i>	Offset within the data space attached to <i>drv_base</i> .
<i>ds</i>	Data space backing the driver memory.

#### Returns

A pointer to the new region.

Definition at line 669 of file [l4virtio](#).

#### 16.415.2.2 find()

```
template<typename DATA>
Mem_region * L4virtio::Svr::Driver_mem_list_t< DATA >::find (
    l4_uint64_t base,
    l4_umword_t size) const [inline]
```

Find memory region containing the given driver address region.

#### Parameters

<i>base</i>	Driver base address.
<i>size</i>	Size of the region.

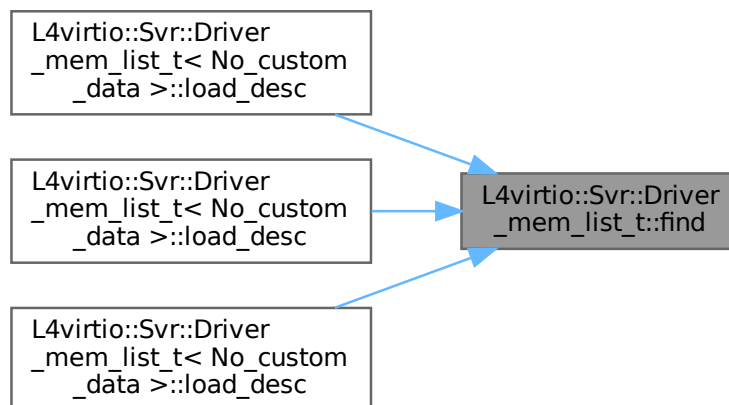
#### Returns

Pointer to the region containing the given region, NULL if none is found.

Definition at line 703 of file [l4virtio](#).

Referenced by [L4virtio::Svr::Driver\\_mem\\_list\\_t< No\\_custom\\_data >::load\\_desc\(\)](#), [L4virtio::Svr::Driver\\_mem\\_list\\_t< No\\_custom\\_data >::load\\_desc\(\)](#) and [L4virtio::Svr::Driver\\_mem\\_list\\_t< No\\_custom\\_data >::load\\_desc\(\)](#).

Here is the caller graph for this function:



#### 16.415.2.3 full()

```
template<typename DATA>
bool L4virtio::Svr::Driver_mem_list_t< DATA >::full () const [inline]
```

#### Returns

True if the remaining capacity is 0.

Definition at line 658 of file [l4virtio](#).

Referenced by [L4virtio::Svr::Driver\\_mem\\_list\\_t< No\\_custom\\_data >::add\(\)](#).

Here is the caller graph for this function:



#### 16.415.2.4 init()

```
template<typename DATA>
void L4virtio::Svr::Driver_mem_list_t< DATA >::init (
    unsigned max) [inline]
```

Make a fresh list with capacity *max*.

##### Parameters

<i>max</i>	The capacity of this vector.
------------	------------------------------

Definition at line 650 of file [l4virtio](#).

#### 16.415.2.5 load\_desc() [1/3]

```
template<typename DATA>
template<typename ARG>
void L4virtio::Svr::Driver_mem_list_t< DATA >::load_desc (
    Virtqueue::Desc const & desc,
    Request_processor const * p,
    ARG * data) const [inline]
```

Default implementation returning generic information.

##### Template Parameters

<i>ARG</i>	Abstract argument type used with <a href="#">Request_processor::start()</a> and <a href="#">Request_processor::next()</a> to deliver the result of loading a descriptor. This type must provide a constructor taking three arguments: (1) pointer to a <code>Driver_mem_region</code> , (2) the <a href="#">Virtqueue::Desc</a> descriptor, and (3) a pointer to the calling <a href="#">Request_processor</a> .
------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

##### Parameters

	<i>desc</i>	The descriptor to load
--	-------------	------------------------



	<i>p</i>	The request processor calling us
out	<i>data</i>	Shall be assigned to ARG(mem, desc, p)

### Exceptions

<a href="#"><i>Bad_descriptor</i></a>	The descriptor address could not be translated.
---------------------------------------	-------------------------------------------------

Definition at line 764 of file [l4virtio](#).

#### 16.415.2.6 load\_desc() [2/3]

```
template<typename DATA>
void L4virtio::Svr::Driver_mem_list_t< DATA >::load_desc (
    Virtqueue::Desc const & desc,
    Request_processor const * p,
    Mem_region const ** data) const [inline]
```

Default implementation returning the Driver\_mem\_region.

### Parameters

	<i>desc</i>	The descriptor to load
	<i>p</i>	The request processor calling us
out	<i>data</i>	Shall be set to a pointer to the Driver_mem_region that covers the descriptor.

### Exceptions

<a href="#"><i>Bad_descriptor</i></a>	The descriptor address could not be translated.
---------------------------------------	-------------------------------------------------

Definition at line 737 of file [l4virtio](#).

#### 16.415.2.7 load\_desc() [3/3]

```
template<typename DATA>
void L4virtio::Svr::Driver_mem_list_t< DATA >::load_desc (
    Virtqueue::Desc const & desc,
    Request_processor const * p,
    Virtqueue::Desc const ** table) const [inline]
```

Default implementation for loading an indirect descriptor.

### Parameters

	<i>desc</i>	The descriptor to load
--	-------------	------------------------

	<i>p</i>	The request processor calling us
out	<i>table</i>	Shall be set to the loaded descriptor table

### Exceptions

<a href="#">Bad_descriptor</a>	The descriptor address could not be translated.
--------------------------------	-------------------------------------------------

Definition at line 717 of file [l4virtio](#).

#### 16.415.2.8 remove()

```
template<typename DATA>
void L4virtio::Svr::Driver_mem_list_t< DATA >::remove (
    Mem_region const * r) [inline]
```

Remove the given region from the list.

### Parameters

<i>r</i>	The region to remove (result from <a href="#">add()</a> , or <a href="#">find()</a> ).
----------	----------------------------------------------------------------------------------------

Definition at line 683 of file [l4virtio](#).

The documentation for this class was generated from the following file:

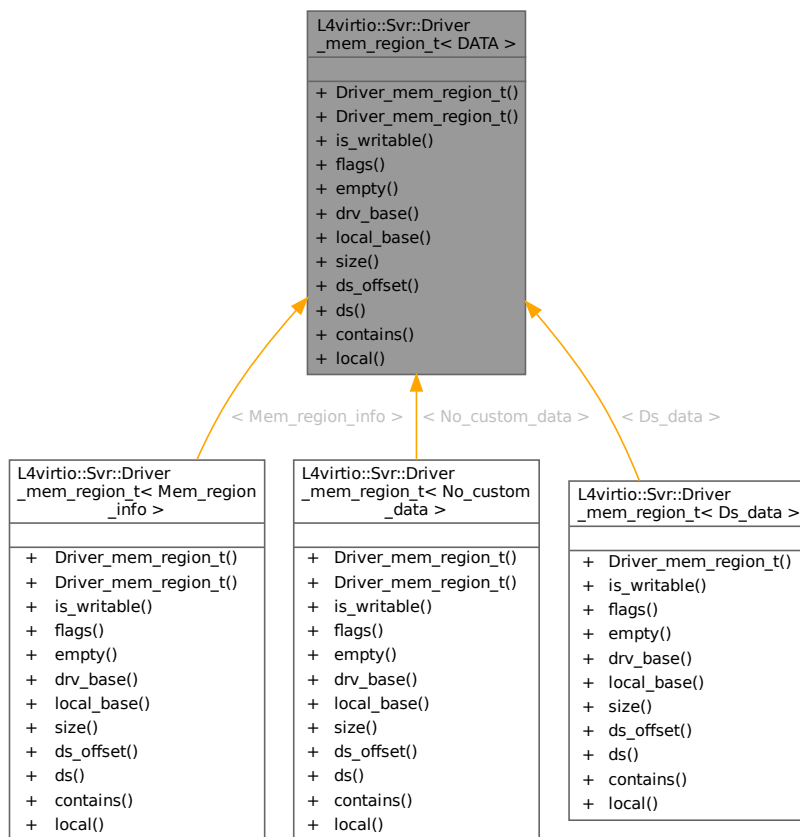
- [l4/l4virtio/server/l4virtio](#)

## 16.416 L4virtio::Svr::Driver\_mem\_region\_t< DATA > Class Template Reference

Region of driver memory, that shall be managed locally.

```
#include <l4virtio>
```

Inheritance diagram for L4virtio::Svr::Driver\_mem\_region\_t< DATA >:



Collaboration diagram for L4virtio::Svr::Driver\_mem\_region\_t< DATA >:

L4virtio::Svr::Driver_mem_region_t< DATA >
<ul style="list-style-type: none"> <li>+ Driver_mem_region_t()</li> <li>+ Driver_mem_region_t()</li> <li>+ is_writable()</li> <li>+ flags()</li> <li>+ empty()</li> <li>+ drv_base()</li> <li>+ local_base()</li> <li>+ size()</li> <li>+ ds_offset()</li> <li>+ ds()</li> <li>+ contains()</li> <li>+ local()</li> </ul>

## Public Member Functions

- **Driver\_mem\_region\_t ()**  
*Make default empty memory region.*
- **Driver\_mem\_region\_t (l4\_uint64\_t drv\_base, l4\_umword\_t size, l4\_addr\_t offset, Ds\_cap &&ds)**  
*Make a local memory region for the given driver values.*
- bool **is\_writable ()** const
- Flags **flags ()** const
- bool **empty ()** const
- **l4\_uint64\_t drv\_base ()** const
- **l4\_addr\_t local\_base ()** const
- **l4\_umword\_t size ()** const
- **l4\_addr\_t ds\_offset ()** const
- **L4::Cap< L4Re::Dataspace > ds ()** const
- bool **contains (l4\_uint64\_t base, l4\_umword\_t size)** const  
*Test if the given driver address range is within this region.*
- template<typename T>  
**T \* local (Ptr< T > p)** const  
*Get the local address for driver address p.*

### 16.416.1 Detailed Description

```
template<typename DATA>
class L4virtio::Svr::Driver_mem_region_t< DATA >
```

Region of driver memory, that shall be managed locally.

## Template Parameters

---

<i>DATA</i>	Class defining additional information
-------------	---------------------------------------

Definition at line 450 of file [l4virtio](#).

## 16.416.2 Constructor & Destructor Documentation

### 16.416.2.1 Driver\_mem\_region\_t()

```
template<typename DATA>
L4virtio::Svr::Driver_mem_region_t< DATA >::Driver_mem_region_t (
    l4_uint64_t drv_base,
    l4_umword_t size,
    l4_addr_t offset,
    Ds_cap && ds) [inline]
```

Make a local memory region for the given driver values.

#### Parameters

<i>drv_base</i>	Base address of the memory region used by the driver.
<i>size</i>	Size of the memory region.
<i>offset</i>	Offset within the data space that is mapped to <i>drv_base</i> within the driver.
<i>ds</i>	Data space capability backing the memory.

This constructor attaches the region of given data space to the local address space and stores the corresponding data for later reference.

Definition at line 498 of file [l4virtio](#).

## 16.416.3 Member Function Documentation

### 16.416.3.1 contains()

```
template<typename DATA>
bool L4virtio::Svr::Driver_mem_region_t< DATA >::contains (
    l4_uint64_t base,
    l4_umword_t size) const [inline]
```

Test if the given driver address range is within this region.

#### Parameters

<i>base</i>	The driver base address.
<i>size</i>	The size of the region to lookup.

#### Returns

true if the given driver address region is contained in this region, false else.

Definition at line 592 of file [l4virtio](#).

### 16.416.3.2 drv\_base()

```
template<typename DATA>
l4_uint64_t L4virtio::Svr::Driver_mem_region_t< DATA >::drv_base () const [inline]
```

#### Returns

The base address used by the driver.

Definition at line 571 of file [l4virtio](#).

### 16.416.3.3 ds()

```
template<typename DATA>
L4::Cap< L4Re::Dataspace > L4virtio::Svr::Driver_mem_region_t< DATA >::ds () const [inline]
```

#### Returns

The data space capability for this region.

Definition at line 583 of file [l4virtio](#).

### 16.416.3.4 ds\_offset()

```
template<typename DATA>
l4_addr_t L4virtio::Svr::Driver_mem_region_t< DATA >::ds_offset () const [inline]
```

#### Returns

The offset within the data space.

Definition at line 580 of file [l4virtio](#).

### 16.416.3.5 empty()

```
template<typename DATA>
bool L4virtio::Svr::Driver_mem_region_t< DATA >::empty () const [inline]
```

#### Returns

True if the region is empty (size == 0), false otherwise.

Definition at line 567 of file [l4virtio](#).

**16.416.3.6 flags()**

```
template<typename DATA>
Flags L4virtio::Svr::Driver_mem_region_t< DATA >::flags () const [inline]
```

**Returns**

The flags for this region.

Definition at line 564 of file [l4virtio](#).

**16.416.3.7 is\_writable()**

```
template<typename DATA>
bool L4virtio::Svr::Driver_mem_region_t< DATA >::is_writable () const [inline]
```

**Returns**

True if the region is writable, false otherwise.

Definition at line 561 of file [l4virtio](#).

**16.416.3.8 local()**

```
template<typename DATA>
template<typename T>
T * L4virtio::Svr::Driver_mem_region_t< DATA >::local (
    Ptr< T > p) const [inline]
```

Get the local address for driver address *p*.

**Parameters**

<i>p</i>	Driver address to translate.
----------	------------------------------

**Precondition**

*p must* be contained in this region.

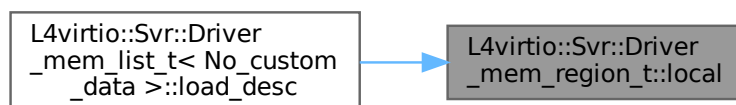
**Returns**

Local address for the given driver address *p*.

Definition at line 616 of file [l4virtio](#).

Referenced by [L4virtio::Svr::Driver\\_mem\\_list\\_t< No\\_custom\\_data >::load\\_desc\(\)](#).

Here is the caller graph for this function:





### 16.416.3.9 local\_base()

```
template<typename DATA>
l4_addr_t L4virtio::Svr::Driver_mem_region_t< DATA >::local_base () const [inline]
```

#### Returns

The local base address.

Definition at line 574 of file [l4virtio](#).

### 16.416.3.10 size()

```
template<typename DATA>
l4_umword_t L4virtio::Svr::Driver_mem_region_t< DATA >::size () const [inline]
```

#### Returns

The size of the region in bytes.

Definition at line 577 of file [l4virtio](#).

The documentation for this class was generated from the following file:

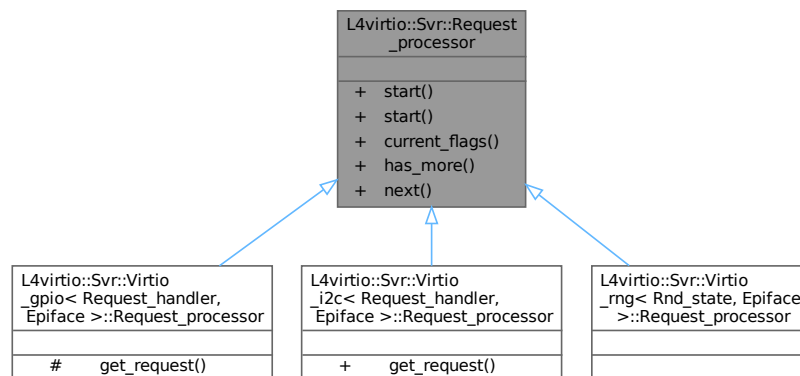
- [l4/l4virtio/server/l4virtio](#)

## 16.417 L4virtio::Svr::Request\_processor Class Reference

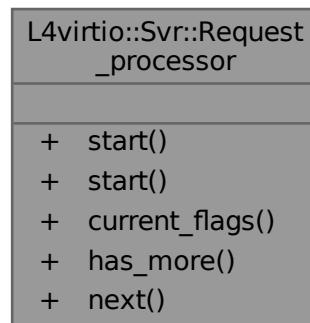
Encapsulate the state for processing a VIRTIO request.

```
#include <virtio>
```

Inheritance diagram for L4virtio::Svr::Request\_processor:



Collaboration diagram for L4virtio::Svr::Request\_processor:



## Public Member Functions

- `template<typename DESC_MAN, typename ... ARGS>`  
`void start (DESC_MAN *dm, Virtqueue *ring, Virtqueue::Head_desc const &request, ARGS... args)`  
*Start processing a new request.*
- `template<typename DESC_MAN, typename ... ARGS>`  
`Virtqueue::Request const & start (DESC_MAN *dm, Virtqueue::Request const &request, ARGS... args)`  
*Start processing a new request.*
- `Virtqueue::Desc::Flags current_flags () const`  
*Get the flags of the currently processed descriptor.*
- `bool has_more () const`  
*Are there more chained descriptors?*
- `template<typename DESC_MAN, typename ... ARGS>`  
`bool next (DESC_MAN *dm, ARGS... args)`  
*Switch to the next descriptor in a descriptor chain.*

### 16.417.1 Detailed Description

Encapsulate the state for processing a VIRTIO request.

A VIRTIO request is a possibly chained list of descriptors retrieved from the available ring of a virtqueue, using `Virtqueue::next_avail()`.

The descriptor processing depends on helper (DESC\_MAN) for interpreting the descriptors in the context of the device implementation.

DESC\_MAN has to provide the functionality to safely dereference a descriptor from a descriptor list.

The following methods must be provided by DESC\_MAN:

- `DESC_MAN::load_desc (Virtqueue::Desc const &desc,`  
`Request_processor const *proc,`  
`Virtqueue::Desc const **table)`

This function is used to dereference `desc` as an indirect descriptor table, and must return a pointer to an indirect descriptor table.

- `DESC_MAN::load_desc(Virtqueue::Desc const &desc, Request_processor const *proc, ...)`

This function is used to dereference a descriptor as a normal data buffer, and '...' are the arguments that are passed to `start()` and `next()`.

Definition at line 472 of file `virtio`.

## 16.417.2 Member Function Documentation

### 16.417.2.1 `current_flags()`

```
Virtqueue::Desc::Flags L4virtio::Svr::Request_processor::current_flags () const [inline]
```

Get the flags of the currently processed descriptor.

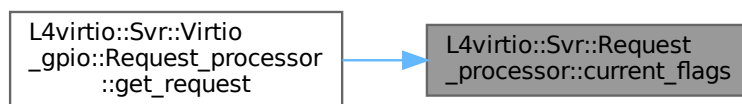
#### Returns

The flags of the currently processed descriptor.

Definition at line 545 of file `virtio`.

Referenced by `L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::Request_processor::get_request()`.

Here is the caller graph for this function:



### 16.417.2.2 `has_more()`

```
bool L4virtio::Svr::Request_processor::has_more () const [inline]
```

Are there more chained descriptors?

#### Returns

true if there are more chained descriptors in the current request.

Definition at line 553 of file `virtio`.

Referenced by `L4virtio::Svr::Block_request< Ds_data >::data_size()`.

Here is the caller graph for this function:



**16.417.2.3 next()**

```
template<typename DESC_MAN, typename ... ARGS>
bool L4virtio::Svr::Request_processor::next (
    DESC_MAN * dm,
    ARGS... args) [inline]
```

Switch to the next descriptor in a descriptor chain.

**Template Parameters**

<i>DESC_MAN</i>	Type of descriptor manager (implicit).
-----------------	----------------------------------------

**Parameters**

<i>dm</i>	Descriptor manager that is used to translate VIRTIO descriptor addresses.
<i>args</i>	Extra arguments passed to dm->load_desc()

**Return values**

<i>true</i>	A next descriptor is available.
<i>false</i>	No descriptor available.

**Exceptions**

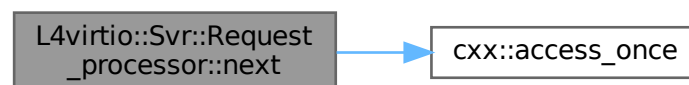
<a href="#"><i>Bad_descriptor</i></a>	The <a href="#">next</a> index of this descriptor is invalid.
---------------------------------------	---------------------------------------------------------------

Definition at line 570 of file [virtio](#).

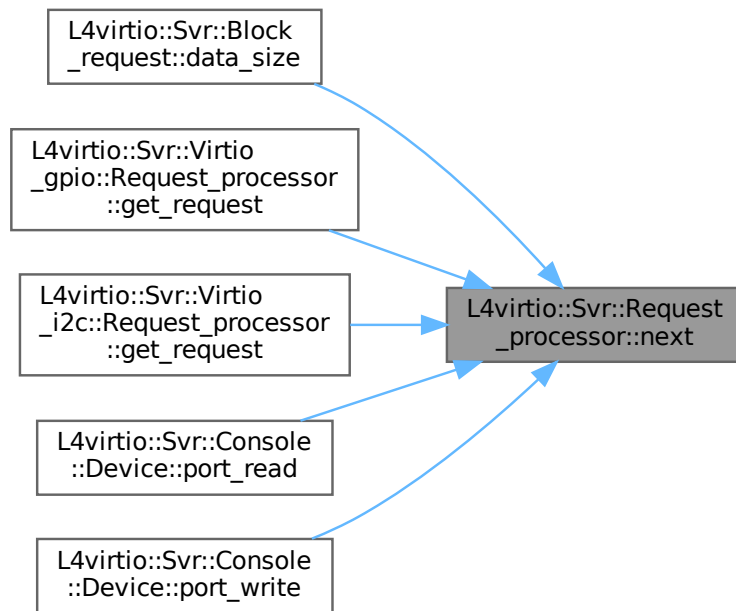
References [cxx::access\\_once\(\)](#), [L4virtio::Svr::Bad\\_descriptor::Bad\\_flags](#), [L4virtio::Svr::Bad\\_descriptor::Bad\\_next](#), and [L4\\_UNLIKELY](#).

Referenced by [L4virtio::Svr::Block\\_request<Ds\\_data>::data\\_size\(\)](#), [L4virtio::Svr::Virtio\\_gpio<Request\\_handler, Epiface>::Request\\_handler::get\\_request\(\)](#), [L4virtio::Svr::Virtio\\_i2c<Request\\_handler, Epiface>::Request\\_processor::get\\_request\(\)](#), [L4virtio::Svr::Console::Device::port\\_read\(\)](#) and [L4virtio::Svr::Console::Device::port\\_write\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 16.417.2.4 start() [1/2]

```

template<typename DESC_MAN, typename ... ARGS>
void L4virtio::Svr::Request_processor::start (
    DESC_MAN * dm,
    Virtqueue * ring,
    Virtqueue::Head_desc const & request,
    ARGS... args) [inline]

```

Start processing a new request.

#### Template Parameters

<code>DESC_MAN</code>	Type of descriptor manager (implicit).
-----------------------	----------------------------------------

#### Parameters

<code>dm</code>	Descriptor manager that is used to translate VIRTIO descriptor addresses.
<code>ring</code>	VIRTIO ring of the request.
<code>request</code>	VIRTIO request from <a href="#">Virtqueue::next_avail()</a>
<code>args</code>	Extra arguments passed to <code>dm-&gt;load_desc()</code>

**Precondition**

The given request must be valid.

**Exceptions**

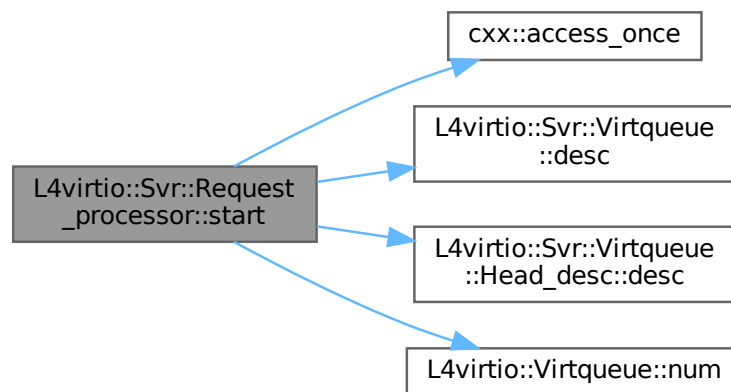
<a href="#"><i>Bad_descriptor</i></a>	The descriptor has an invalid size or <code>load_desc()</code> has thrown an exception by itself.
---------------------------------------	---------------------------------------------------------------------------------------------------

Definition at line 501 of file [virtio](#).

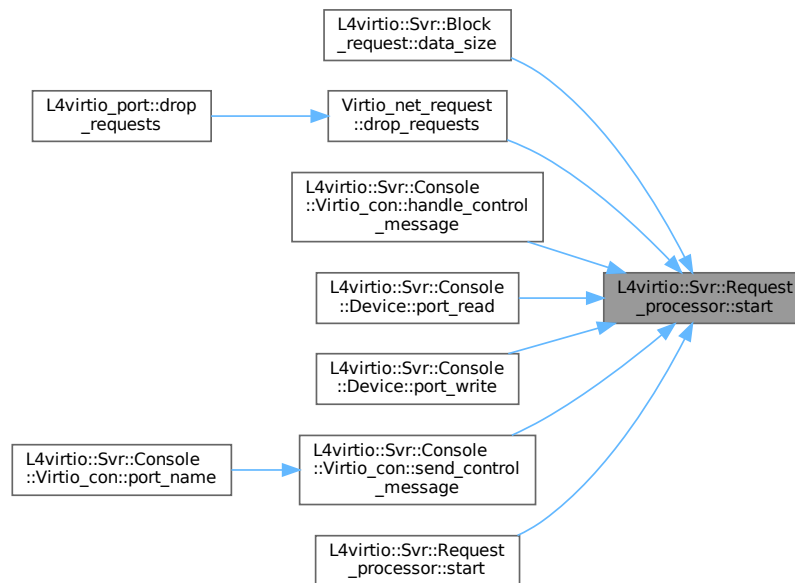
References [cxx::access\\_once\(\)](#), [L4virtio::Svr::Bad\\_descriptor::Bad\\_size](#), [L4virtio::Svr::Virtqueue::desc\(\)](#), [L4virtio::Svr::Virtqueue::Head\\_desc::desc\(\)](#), [L4\\_UNLIKELY](#), and [L4virtio::Virtqueue::num\(\)](#).

Referenced by [L4virtio::Svr::Block\\_request<Ds\\_data>::data\\_size\(\)](#), [Virtio\\_net\\_request::drop\\_requests\(\)](#), [L4virtio::Svr::Console::Virtio\\_con::handle\\_control\\_message\(\)](#), [L4virtio::Svr::Console::Device::port\\_read\(\)](#), [L4virtio::Svr::Console::Device::port\\_write\(\)](#), [L4virtio::Svr::Console::Virtio\\_con::send\\_control\\_message\(\)](#), and [start\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.417.2.5 start() [2/2]

```

template<typename DESC_MAN, typename ... ARGS>
Virtqueue::Request const & L4virtio::Svr::Request_processor::start (
    DESC_MAN * dm,
    Virtqueue::Request const & request,
    ARGS... args) [inline]

```

Start processing a new request.

#### Template Parameters

<code>DESC_MAN</code>	Type of descriptor manager (implicit).
-----------------------	----------------------------------------

#### Parameters

<code>dm</code>	Descriptor manager that is used to translate VIRTIO descriptor addresses.
<code>request</code>	VIRTIO request from <a href="#">Virtqueue::next_avail()</a>
<code>args</code>	Extra arguments passed to <code>dm-&gt;load_desc()</code>

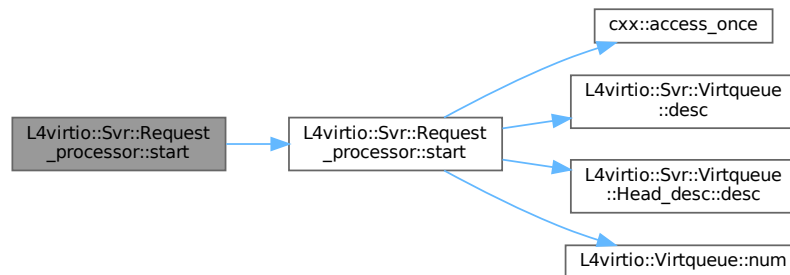
**Precondition**

The given request must be valid.

Definition at line 534 of file [virtio](#).

References [start\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

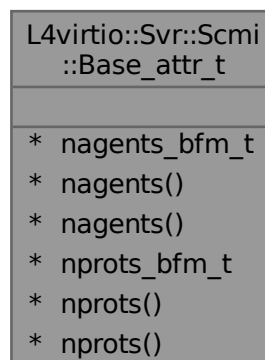
- `l4/l4virtio/server/virtio`

## 16.418 L4virtio::Svr::Scmi::Base\_attr\_t Struct Reference

SCMI base protocol attributes.

```
#include <virtio-scmi-device>
```

Collaboration diagram for L4virtio::Svr::Scmi::Base\_attr\_t:





- typedef [cxx::Bitfield](#)< decltype(attr\_raw), 8, 15 > **nagents\_bfm\_t**  
Type to access the *nagents* bits (8 to 15) of *attr\_raw*.
- constexpr [nagents\\_bfm\\_t::Val](#) **nagents** () const  
Get the *nagents* bits (8 to 15) of *attr\_raw*.
- constexpr [nagents\\_bfm\\_t::Ref](#) **nagents** ()  
Get a reference to the *nagents* bits (8 to 15) of *attr\_raw*.
  
- typedef [cxx::Bitfield](#)< decltype(attr\_raw), 0, 7 > **nprots\_bfm\_t**  
Type to access the *nprots* bits (0 to 7) of *attr\_raw*.
- constexpr [nprots\\_bfm\\_t::Val](#) **nprots** () const  
Get the *nprots* bits (0 to 7) of *attr\_raw*.
- constexpr [nprots\\_bfm\\_t::Ref](#) **nprots** ()  
Get a reference to the *nprots* bits (0 to 7) of *attr\_raw*.

### 16.418.1 Detailed Description

SCMI base protocol attributes.

Definition at line 90 of file [virtio-scmi-device](#).

The documentation for this struct was generated from the following file:

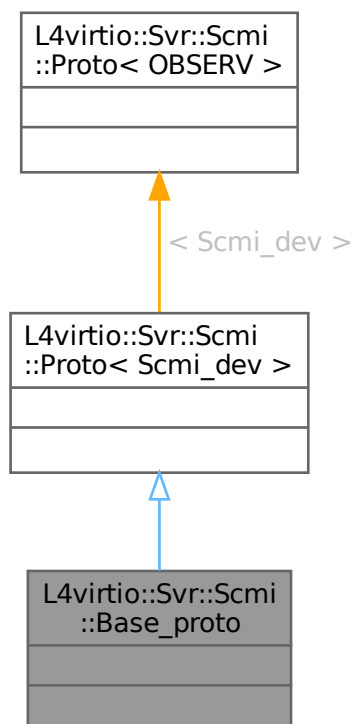
- I4/I4virtio/server/virtio-scmi-device

## 16.419 L4virtio::Svr::Scmi::Base\_proto Class Reference

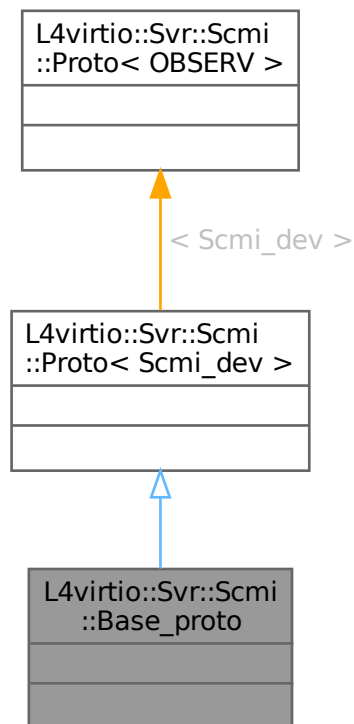
Base class for the SCMI base protocol.

```
#include <virtio-scmi-device>
```

Inheritance diagram for L4virtio::Svr::Scmi::Base\_proto:



Collaboration diagram for L4virtio::Svr::Scmi::Base\_proto:



### 16.419.1 Detailed Description

Base class for the SCMI base protocol.

Use this class as a base to implement the base protocol.

Definition at line 458 of file [virtio-scmi-device](#).

The documentation for this class was generated from the following file:

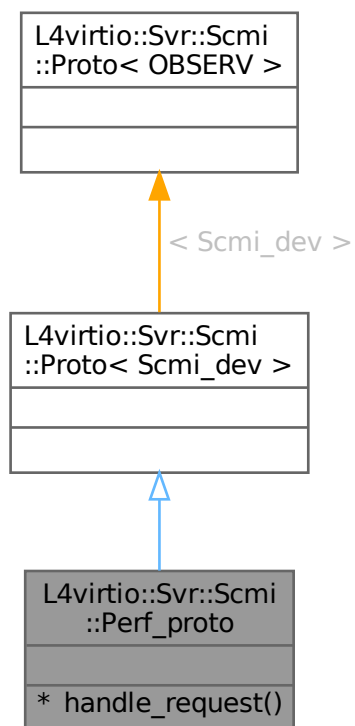
- I4/I4virtio/server/virtio-scmi-device

## 16.420 L4virtio::Svr::Scmi::Perf\_proto Class Reference

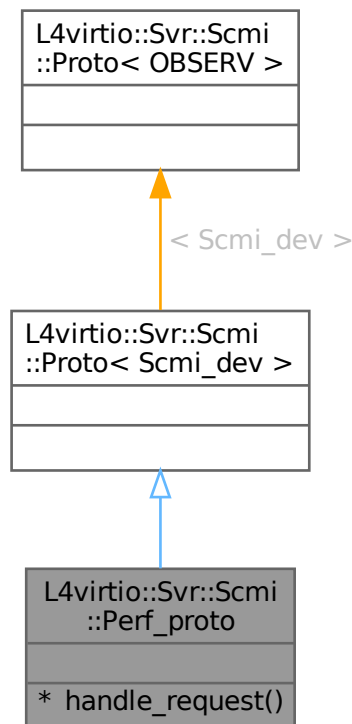
Base class for the SCMI performance protocol.

```
#include <virtio-scmi-device>
```

Inheritance diagram for L4virtio::Svr::Scmi::Perf\_proto:



Collaboration diagram for L4virtio::Svr::Scmi::Perf\_proto:



### 16.420.1 Detailed Description

Base class for the SCMI performance protocol.

Use this class as a base to implement the performance protocol.

If you want to use this from a Uvmm Linux guest, the device tree needs to look something like this:

```

firmware {
    scmi {
        compatible = "arm,scmi-virtio";

        #address-cells = <1>;
        #size-cells = <0>;

        cpufreq: protocol@13 {
            reg = <0x13>;
            #clock-cells = <1>;
        };
    };
};
....

cpu@0 {
    device_type = "cpu";
    reg = <0x0>;
    clocks = <&cpufreq 0>; // domain_id
};
  
```

Definition at line 662 of file [virtio-scmi-device](#).

The documentation for this class was generated from the following file:

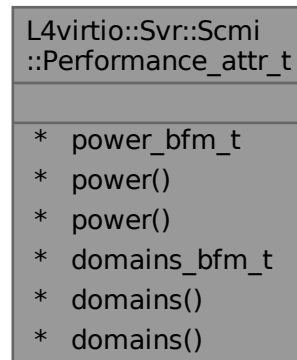
- `I4/I4virtio/server/virtio-scmi-device`

## 16.421 L4virtio::Svr::Scmi::Performance\_attr\_t Struct Reference

SCMI performance protocol attributes.

```
#include <virtio-scmi-device>
```

Collaboration diagram for L4virtio::Svr::Scmi::Performance\_attr\_t:



- typedef [cxx::Bitfield](#)< decltype(attr\_raw), 16, 16 > **power\_bfm\_t**  
Type to access the [power](#) bits (16 to 16) of *attr\_raw*.
- constexpr [power\\_bfm\\_t::Val](#) **power** () const  
Get the [power](#) bits (16 to 16) of *attr\_raw*.
- constexpr [power\\_bfm\\_t::Ref](#) **power** ()  
Get a reference to the [power](#) bits (16 to 16) of *attr\_raw*.
- typedef [cxx::Bitfield](#)< decltype(attr\_raw), 0, 15 > **domains\_bfm\_t**  
Type to access the [domains](#) bits (0 to 15) of *attr\_raw*.
- constexpr [domains\\_bfm\\_t::Val](#) **domains** () const  
Get the [domains](#) bits (0 to 15) of *attr\_raw*.
- constexpr [domains\\_bfm\\_t::Ref](#) **domains** ()  
Get a reference to the [domains](#) bits (0 to 15) of *attr\_raw*.

### 16.421.1 Detailed Description

SCMI performance protocol attributes.

Definition at line 112 of file [virtio-scmi-device](#).

The documentation for this struct was generated from the following file:

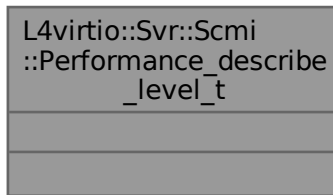
- l4/l4virtio/server/virtio-scmi-device

## 16.422 L4virtio::Svr::Scmi::Performance\_describe\_level\_t Struct Reference

SCMI performance describe level.

```
#include <virtio-scmi-device>
```

Collaboration diagram for L4virtio::Svr::Scmi::Performance\_describe\_level\_t:



### 16.422.1 Detailed Description

SCMI performance describe level.

Definition at line 150 of file [virtio-scmi-device](#).

The documentation for this struct was generated from the following file:

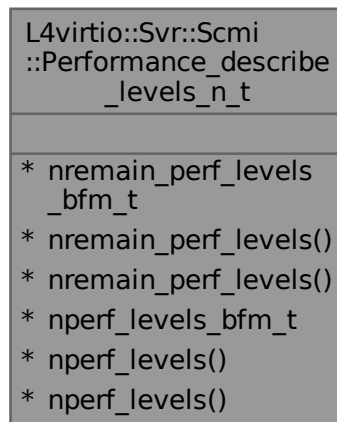
- I4/I4virtio/server/virtio-scmi-device

## 16.423 L4virtio::Svr::Scmi::Performance\_describe\_levels\_n\_t Struct Reference

SCMI performance describe levels numbers.

```
#include <virtio-scmi-device>
```

Collaboration diagram for L4virtio::Svr::Scmi::Performance\_describe\_levels\_n\_t:



- typedef [cxx::Bitfield](#)< decltype(num\_levels\_raw), 16, 31 > **nremain\_perf\_levels\_bfm\_t**  
*Type to access the [nremain\\_perf\\_levels](#) bits (16 to 31) of num\_levels\_raw.*
- constexpr [nremain\\_perf\\_levels\\_bfm\\_t::Val](#) **nremain\_perf\_levels** () const  
*Get the [nremain\\_perf\\_levels](#) bits (16 to 31) of num\_levels\_raw.*
- constexpr [nremain\\_perf\\_levels\\_bfm\\_t::Ref](#) **nremain\_perf\_levels** ()  
*Get a reference to the [nremain\\_perf\\_levels](#) bits (16 to 31) of num\_levels\_raw.*
- typedef [cxx::Bitfield](#)< decltype(num\_levels\_raw), 0, 11 > **nperf\_levels\_bfm\_t**  
*Type to access the [nperf\\_levels](#) bits (0 to 11) of num\_levels\_raw.*
- constexpr [nperf\\_levels\\_bfm\\_t::Val](#) **nperf\_levels** () const  
*Get the [nperf\\_levels](#) bits (0 to 11) of num\_levels\_raw.*
- constexpr [nperf\\_levels\\_bfm\\_t::Ref](#) **nperf\_levels** ()  
*Get a reference to the [nperf\\_levels](#) bits (0 to 11) of num\_levels\_raw.*

### 16.423.1 Detailed Description

SCMI performance describe levels numbers.

Definition at line [142](#) of file [virtio-scmi-device](#).

The documentation for this struct was generated from the following file:

- l4/l4virtio/server/virtio-scmi-device



## 16.424 L4virtio::Svr::Scmi::Performance\_domain\_attr\_t Struct Reference

SCMI performance domain protocol attributes.

```
#include <virtio-scmi-device>
```

Collaboration diagram for L4virtio::Svr::Scmi::Performance\_domain\_attr\_t:

L4virtio::Svr::Scmi ::Performance_domain _attr_t
<ul style="list-style-type: none"> <li>* set_limits_bfm_t</li> <li>* set_limits()</li> <li>* set_limits()</li> <li>* set_perf_level_bfm_t</li> <li>* set_perf_level()</li> <li>* set_perf_level()</li> <li>* perf_limits_change_notify_bfm_t</li> <li>* perf_limits_change_notify()</li> <li>* perf_limits_change_notify()</li> <li>* perf_level_change_notify_bfm_t</li> <li>* perf_level_change_notify()</li> <li>* perf_level_change_notify()</li> <li>* fast_channel_bfm_t</li> <li>* fast_channel()</li> <li>* fast_channel()</li> <li>* rate_limit_bfm_t</li> <li>* rate_limit()</li> <li>* rate_limit()</li> </ul>

- typedef [cxx::Bitfield](#)< decltype(attr\_raw), 31, 31 > **set\_limits\_bfm\_t**  
Type to access the [set\\_limits](#) bits (31 to 31) of *attr\_raw*.
- constexpr [set\\_limits\\_bfm\\_t::Val](#) **set\_limits** () const  
Get the [set\\_limits](#) bits (31 to 31) of *attr\_raw*.
- constexpr [set\\_limits\\_bfm\\_t::Ref](#) **set\_limits** ()  
Get a reference to the [set\\_limits](#) bits (31 to 31) of *attr\_raw*.

- typedef [cxx::Bitfield](#)< decltype(attr\_raw), 30, 30 > **set\_perf\_level\_bfm\_t**  
Type to access the [set\\_perf\\_level](#) bits (30 to 30) of [attr\\_raw](#).
- constexpr [set\\_perf\\_level\\_bfm\\_t::Val](#) **set\_perf\_level** () const  
Get the [set\\_perf\\_level](#) bits (30 to 30) of [attr\\_raw](#).
- constexpr [set\\_perf\\_level\\_bfm\\_t::Ref](#) **set\_perf\_level** ()  
Get a reference to the [set\\_perf\\_level](#) bits (30 to 30) of [attr\\_raw](#).
  
- typedef [cxx::Bitfield](#)< decltype(attr\_raw), 29, 29 > **perf\_limits\_change\_notify\_bfm\_t**  
Type to access the [perf\\_limits\\_change\\_notify](#) bits (29 to 29) of [attr\\_raw](#).
- constexpr [perf\\_limits\\_change\\_notify\\_bfm\\_t::Val](#) **perf\_limits\_change\_notify** () const  
Get the [perf\\_limits\\_change\\_notify](#) bits (29 to 29) of [attr\\_raw](#).
- constexpr [perf\\_limits\\_change\\_notify\\_bfm\\_t::Ref](#) **perf\_limits\_change\_notify** ()  
Get a reference to the [perf\\_limits\\_change\\_notify](#) bits (29 to 29) of [attr\\_raw](#).
  
- typedef [cxx::Bitfield](#)< decltype(attr\_raw), 28, 28 > **perf\_level\_change\_notify\_bfm\_t**  
Type to access the [perf\\_level\\_change\\_notify](#) bits (28 to 28) of [attr\\_raw](#).
- constexpr [perf\\_level\\_change\\_notify\\_bfm\\_t::Val](#) **perf\_level\_change\_notify** () const  
Get the [perf\\_level\\_change\\_notify](#) bits (28 to 28) of [attr\\_raw](#).
- constexpr [perf\\_level\\_change\\_notify\\_bfm\\_t::Ref](#) **perf\_level\_change\_notify** ()  
Get a reference to the [perf\\_level\\_change\\_notify](#) bits (28 to 28) of [attr\\_raw](#).
  
- typedef [cxx::Bitfield](#)< decltype(attr\_raw), 27, 27 > **fast\_channel\_bfm\_t**  
Type to access the [fast\\_channel](#) bits (27 to 27) of [attr\\_raw](#).
- constexpr [fast\\_channel\\_bfm\\_t::Val](#) **fast\_channel** () const  
Get the [fast\\_channel](#) bits (27 to 27) of [attr\\_raw](#).
- constexpr [fast\\_channel\\_bfm\\_t::Ref](#) **fast\_channel** ()  
Get a reference to the [fast\\_channel](#) bits (27 to 27) of [attr\\_raw](#).
  
- typedef [cxx::Bitfield](#)< decltype(rate\_limit\_raw), 0, 19 > **rate\_limit\_bfm\_t**  
Type to access the [rate\\_limit](#) bits (0 to 19) of [rate\\_limit\\_raw](#).
- constexpr [rate\\_limit\\_bfm\\_t::Val](#) **rate\_limit** () const  
Get the [rate\\_limit](#) bits (0 to 19) of [rate\\_limit\\_raw](#).
- constexpr [rate\\_limit\\_bfm\\_t::Ref](#) **rate\_limit** ()  
Get a reference to the [rate\\_limit](#) bits (0 to 19) of [rate\\_limit\\_raw](#).

### 16.424.1 Detailed Description

SCMI performance domain protocol attributes.

Definition at line 124 of file [virtio-scmi-device](#).

The documentation for this struct was generated from the following file:

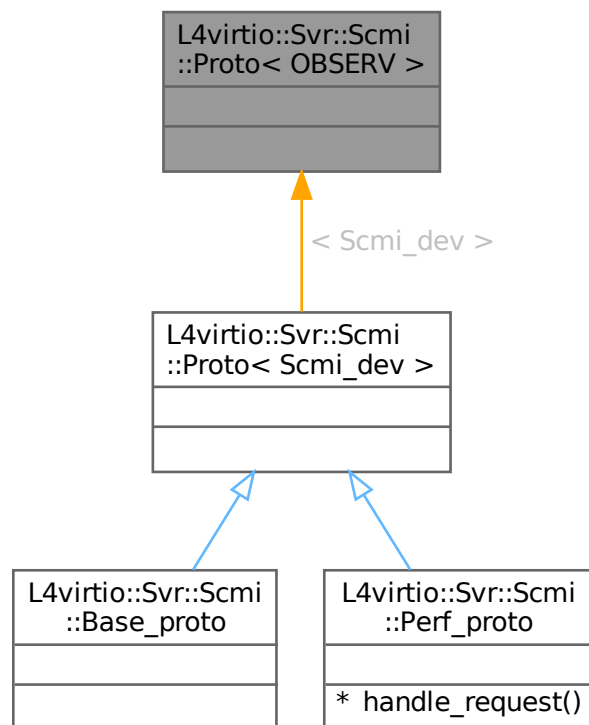
- I4/I4virtio/server/virtio-scmi-device

## 16.425 L4virtio::Svr::Scmi::Proto< OBSERV > Struct Template Reference

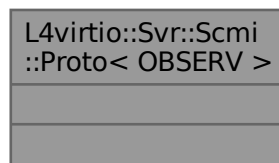
Base class for all protocols.

```
#include <virtio-scmi-device>
```

Inheritance diagram for L4virtio::Svr::Scmi::Proto< OBSERV >:



Collaboration diagram for L4virtio::Svr::Scmi::Proto< OBSERV >:



### 16.425.1 Detailed Description

```
template<typename OBSERV>
struct L4virtio::Svr::Scmi::Proto< OBSERV >
```

Base class for all protocols.

Defines an interface for processing the virtio buffers for the implemented protocol.

Definition at line 299 of file [virtio-scmi-device](#).

The documentation for this struct was generated from the following file:

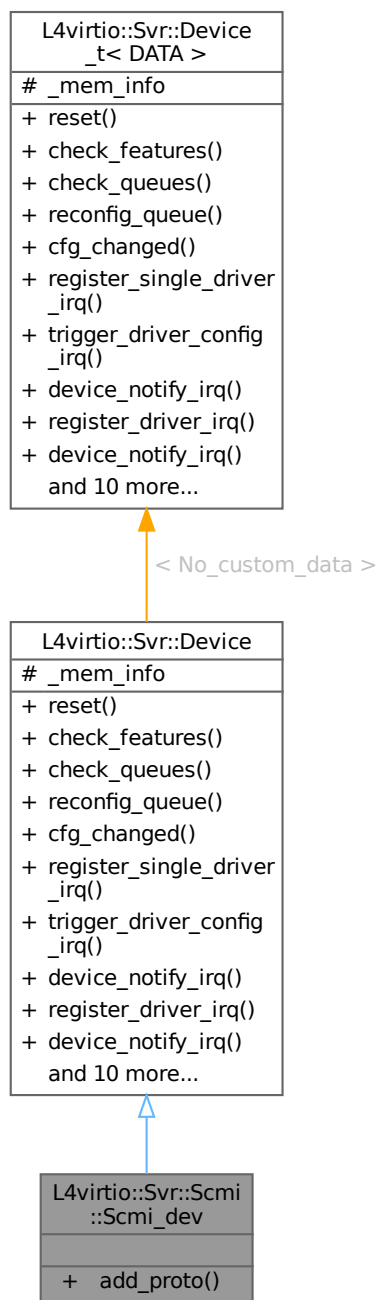
- l4/l4virtio/server/virtio-scmi-device

## 16.426 L4virtio::Svr::Scmi::Scmi\_dev Class Reference

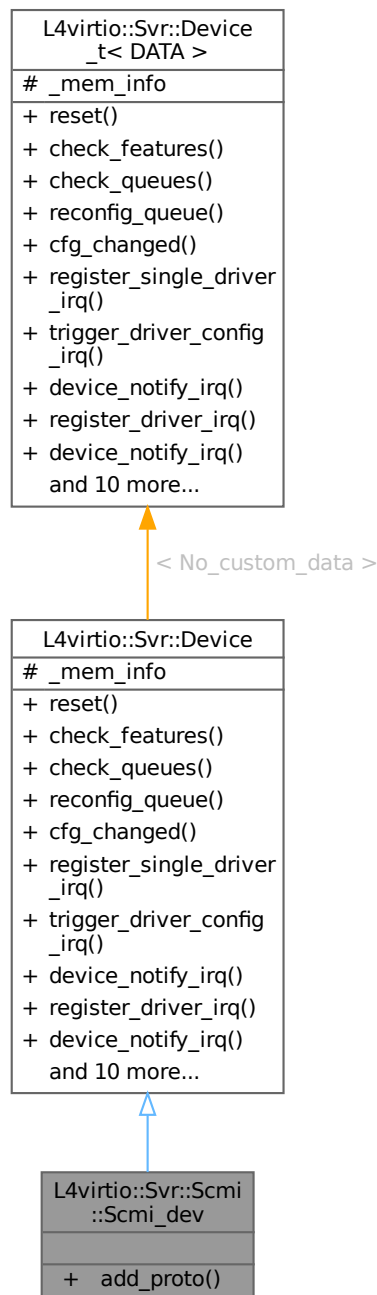
A server implementation of the virtio-scmi protocol.

```
#include <virtio-scmi-device>
```

Inheritance diagram for L4virtio::Svr::Scmi::Scmi\_dev:



Collaboration diagram for L4virtio::Svr::Scmi::Scmi\_dev:



### Public Member Functions

- void **add\_proto** (I4\_uint32\_t id, Proto< Scmi\_dev > \*proto)

*Add an actual protocol implementation with the given id to the server.*

### Public Member Functions inherited from L4virtio::Svr::Device\_t< No\_custom\_data >

- virtual bool **check\_features** ()

- callback for checking the subset of accepted features*
- virtual void **cfg\_changed** (unsigned)
  - callback for client device configuration changes*
- virtual void **register\_driver\_irq** (unsigned idx)
  - Callback for registering an notification IRQ (multi IRQ).*
- virtual **L4::Cap**< **L4::Irq** > **device\_notify\_irq** (unsigned idx)
  - Callback to gather the device notification IRQ (multi IRQ).*
- virtual unsigned **num\_events\_supported** () const
  - Return the highest notification index supported.*
- **Device\_t** (**Dev\_config** \*dev\_config)
  - Make a device for the given config.*
- **Mem\_list** const \* **mem\_info** () const
  - Get the memory region list used for this device.*
- void **reset\_queue\_config** (unsigned idx, unsigned num\_max, bool inc\_generation=false)
  - Trigger reset for the configuration space for queue idx.*
- void **init\_mem\_info** (unsigned num)
  - Initialize the memory region list to the given maximum.*
- void **device\_error** ()
  - Transition device into DEVICE\_NEEDS\_RESET state.*
- bool **setup\_queue** (**Virtqueue** \*q, unsigned qn, unsigned num\_max)
  - Enable/disable the specified queue.*
- bool **handle\_mem\_cmd\_write** ()
  - Check for a value in the cmd register and handle a write.*
- void **enable\_trusted\_ds\_validation** ()
  - Enable trusted dataspace validation.*
- void **add\_trusted\_dataspaces** (std::shared\_ptr< Ds\_vector const > ds)
  - Provide a list of trusted dataspaces that can be used for validation.*

### Additional Inherited Members

### Protected Attributes inherited from **L4virtio::Svr::Device\_t**< **No\_custom\_data** >

- **Mem\_list** **\_mem\_info**
  - Memory region list.*

## 16.426.1 Detailed Description

A server implementation of the virtio-scmi protocol.

Use this class as a base to implement your own specific SCMI device.

SCMI defines multiple protocols which can be optionally handled. This server implementation is flexible enough to handle any combination of them. The user of this server has to deviate from the provided **Proto** classes (for the protocols he want to handle) and needs to implement the required callbacks.

Right now, support for the base and the performance protocol is provided.

The base protocol is mandatory.

If you want to use this from a Uvmm Linux guest, the device tree needs to look something like this:

```

firmware {
    scmi {
        compatible = "arm,scmi-virtio";

        #address-cells = <1>;
        #size-cells = <0>;

        // ... supported protocols ...
    };
};

```

Definition at line 336 of file [virtio-scmi-device](#).

The documentation for this class was generated from the following file:

- I4/I4virtio/server/virtio-scmi-device

## 16.427 L4virtio::Svr::Scmi::Scmi\_hdr\_t Struct Reference

SCMI header.

```
#include <virtio-scmi-device>
```

Collaboration diagram for L4virtio::Svr::Scmi::Scmi\_hdr\_t:

L4virtio::Svr::Scmi ::Scmi_hdr_t
<ul style="list-style-type: none"> <li>* token_bfm_t</li> <li>* token()</li> <li>* token()</li> <li>* protocol_id_bfm_t</li> <li>* protocol_id()</li> <li>* protocol_id()</li> <li>* message_type_bfm_t</li> <li>* message_type()</li> <li>* message_type()</li> <li>* message_id_bfm_t</li> <li>* message_id()</li> <li>* message_id()</li> </ul>

- typedef [cxx::Bitfield](#)< decltype(hdr\_raw), 18, 27 > **token\_bfm\_t**  
Type to access the *token* bits (18 to 27) of *hdr\_raw*.
- constexpr [token\\_bfm\\_t::Val](#) **token** () const  
Get the *token* bits (18 to 27) of *hdr\_raw*.



- constexpr [token\\_bfm\\_t::Ref](#) token ()  
*Get a reference to the [token](#) bits (18 to 27) of [hdr\\_raw](#).*
- typedef [cxx::Bitfield](#)< decltype(hdr\_raw), 10, 17 > **protocol\_id\_bfm\_t**  
*Type to access the [protocol\\_id](#) bits (10 to 17) of [hdr\\_raw](#).*
- constexpr [protocol\\_id\\_bfm\\_t::Val](#) protocol\_id () const  
*Get the [protocol\\_id](#) bits (10 to 17) of [hdr\\_raw](#).*
- constexpr [protocol\\_id\\_bfm\\_t::Ref](#) protocol\_id ()  
*Get a reference to the [protocol\\_id](#) bits (10 to 17) of [hdr\\_raw](#).*
- typedef [cxx::Bitfield](#)< decltype(hdr\_raw), 8, 9 > **message\_type\_bfm\_t**  
*Type to access the [message\\_type](#) bits (8 to 9) of [hdr\\_raw](#).*
- constexpr [message\\_type\\_bfm\\_t::Val](#) message\_type () const  
*Get the [message\\_type](#) bits (8 to 9) of [hdr\\_raw](#).*
- constexpr [message\\_type\\_bfm\\_t::Ref](#) message\_type ()  
*Get a reference to the [message\\_type](#) bits (8 to 9) of [hdr\\_raw](#).*
- typedef [cxx::Bitfield](#)< decltype(hdr\_raw), 0, 7 > **message\_id\_bfm\_t**  
*Type to access the [message\\_id](#) bits (0 to 7) of [hdr\\_raw](#).*
- constexpr [message\\_id\\_bfm\\_t::Val](#) message\_id () const  
*Get the [message\\_id](#) bits (0 to 7) of [hdr\\_raw](#).*
- constexpr [message\\_id\\_bfm\\_t::Ref](#) message\_id ()  
*Get a reference to the [message\\_id](#) bits (0 to 7) of [hdr\\_raw](#).*

### 16.427.1 Detailed Description

SCMI header.

Definition at line 45 of file [virtio-scmi-device](#).

The documentation for this struct was generated from the following file:

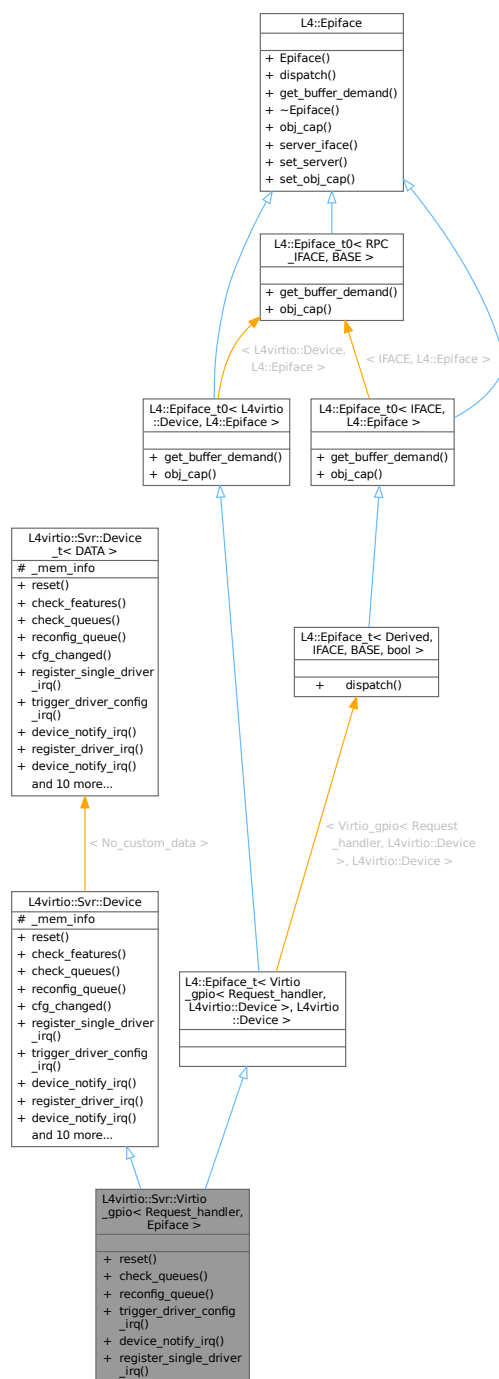
- l4/l4virtio/server/virtio-scmi-device

## 16.428 L4virtio::Svr::Virtio\_gpio< Request\_handler, Epiface > Class Template Reference

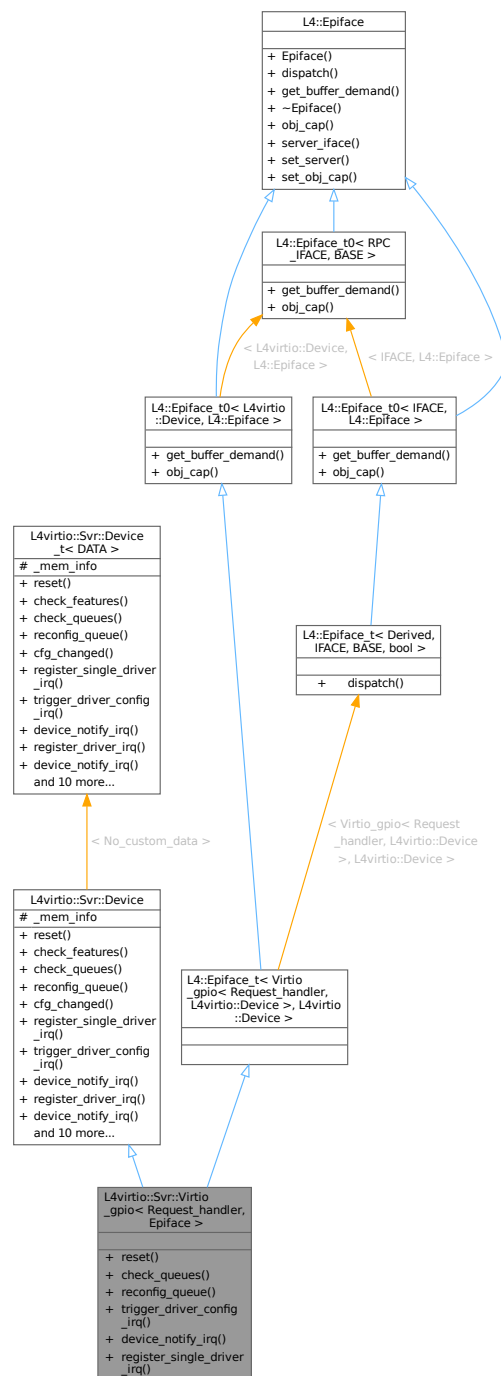
A server implementation of the virtio-gpio protocol.

```
#include <virtio-gpio-device>
```

Inheritance diagram for L4virtio::Svr::Virtio\_gpio< Request\_handler, Epiface >:



Collaboration diagram for L4virtio::Svr::Virtio\_gpio< Request\_handler, Epiface >:



## Data Structures

- struct [Irq\\_handler](#)  
Handler for an gpio pin irq.
- struct [Host\\_irq](#)  
Handler for the host irq.
- struct [Request\\_processor](#)  
Generic handler for the Virtio requests.

## Public Member Functions

- void **reset** () override  
*reset callback, called for doing a device reset*
- bool **check\_queues** () override  
*callback for checking if the queues at DRIVER\_OK transition*
- int **reconfig\_queue** (unsigned idx) override  
*callback for client queue-config request*
- void **trigger\_driver\_config\_irq** () override  
*callback for triggering configuration change notification IRQ*
- [L4::Cap](#) < [L4::lrq](#) > **device\_notify\_irq** () const override  
*callback to gather the device notification IRQ (old-style)*
- void **register\_single\_driver\_irq** () override  
*callback for registering a single guest IRQ for all queues (old-style)*

## Public Member Functions inherited from [L4virtio::Svr::Device\\_t](#) < [No\\_custom\\_data](#) >

- virtual bool **check\_features** ()  
*callback for checking the subset of accepted features*
- virtual void **cfg\_changed** (unsigned)  
*callback for client device configuration changes*
- virtual void **register\_driver\_irq** (unsigned idx)  
*Callback for registering an notification IRQ (multi IRQ).*
- virtual [L4::Cap](#) < [L4::lrq](#) > **device\_notify\_irq** (unsigned idx)  
*Callback to gather the device notification IRQ (multi IRQ).*
- virtual unsigned **num\_events\_supported** () const  
*Return the highest notification index supported.*
- [Device\\_t](#) ([Dev\\_config](#) \*dev\_config)  
*Make a device for the given config.*
- [Mem\\_list](#) const \* **mem\_info** () const  
*Get the memory region list used for this device.*
- void **reset\_queue\_config** (unsigned idx, unsigned num\_max, bool inc\_generation=false)  
*Trigger reset for the configuration space for queue idx.*
- void **init\_mem\_info** (unsigned num)  
*Initialize the memory region list to the given maximum.*
- void **device\_error** ()  
*Transition device into DEVICE\_NEEDS\_RESET state.*
- bool **setup\_queue** ([Virtqueue](#) \*q, unsigned qn, unsigned num\_max)  
*Enable/disable the specified queue.*
- bool **handle\_mem\_cmd\_write** ()  
*Check for a value in the cmd register and handle a write.*
- void **enable\_trusted\_ds\_validation** ()  
*Enable trusted dataspace validation.*
- void **add\_trusted\_dataspaces** (std::shared\_ptr < [Ds\\_vector](#) const > ds)  
*Provide a list of trusted dataspaces that can be used for validation.*

## Public Member Functions inherited from [L4::Epiface\\_t0](#) < [L4virtio::Device](#), [L4::Epiface](#) >

- [Type\\_info::Demand](#) **get\_buffer\_demand** () const  
*Get the server-side buffer demand based in IFACE.*
- [Cap](#) < [L4virtio::Device](#) > **obj\_cap** () const  
*Get the (typed) capability to this object.*

## Public Member Functions inherited from L4::Epiface

- **Epiface** ()  
*Make a server object.*
- virtual ~**Epiface** ()=0  
*Destroy the object.*
- Stored\_cap **obj\_cap** () const  
*Get the capability to the kernel object belonging to this object.*
- **Server\_iface** \* **server\_iface** () const  
*Get pointer to server interface at which the object is currently registered.*
- int **set\_server** (**Server\_iface** \*srv, **Cap**< void > cap, bool managed=false)  
*Set server registration info for the object.*
- void **set\_obj\_cap** (**Cap**< void > const &cap)  
*Deprecated server registration function.*

## Additional Inherited Members

## Public Types inherited from L4::Epiface\_t0< L4virtio::Device, L4::Epiface >

- typedef L4virtio::Device **Interface**  
*Data type of the IPC interface definition.*

## Public Types inherited from L4::Epiface

- typedef **lpc\_svr::Server\_iface** **Server\_iface**  
*Type for abstract server interface.*
- typedef **lpc\_svr::Server\_iface::Demand** **Demand**  
*Type for server-side receive buffer demand.*

## Protected Attributes inherited from L4virtio::Svr::Device\_t< No\_custom\_data >

- **Mem\_list** **\_mem\_info**  
*Memory region list.*

## 16.428.1 Detailed Description

```
template<typename Request_handler, typename Epiface = L4virtio::Device>
class L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >
```

A server implementation of the virtio-gpio protocol.

## Template Parameters

---

<i>Request_handler</i>	<p>The type that is used to handle incoming requests. Needs to have:</p> <ul style="list-style-type: none"> <li>• <code>bool get_direction(l4_uint16_t gpio, l4_uint8_t *dir)</code></li> <li>• <code>bool set_direction(l4_uint16_t gpio, l4_uint8_t dir)</code></li> <li>• <code>bool get_value(l4_uint16_t gpio, l4_uint8_t *val)</code></li> <li>• <code>bool set_value(l4_uint16_t gpio, l4_uint8_t val)</code></li> <li>• <code>bool set_irq_type(l4_uint16_t gpio, l4_uint8_t mode)</code></li> <li>• <code>bool enable_irq(l4_uint16_t gpio, std::shared_ptr&lt;Virtio_gpio::Irq_handler&gt; const &amp;hdl)</code> functions.</li> </ul>
<i>Epiface</i>	The Epiface to derive from. Defaults to <a href="#">L4virtio::Device</a> .

Definition at line 142 of file [virtio-gpio-device](#).

The documentation for this class was generated from the following file:

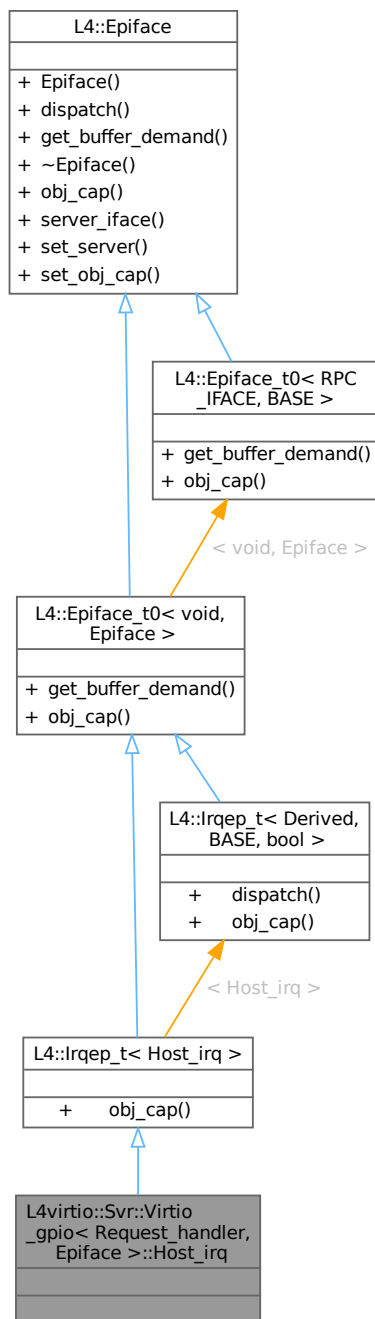
- `l4/l4virtio/server/virtio-gpio-device`

## 16.429 L4virtio::Svr::Virtio\_gpio< Request\_handler, Epiface >::Host\_irq Struct Reference

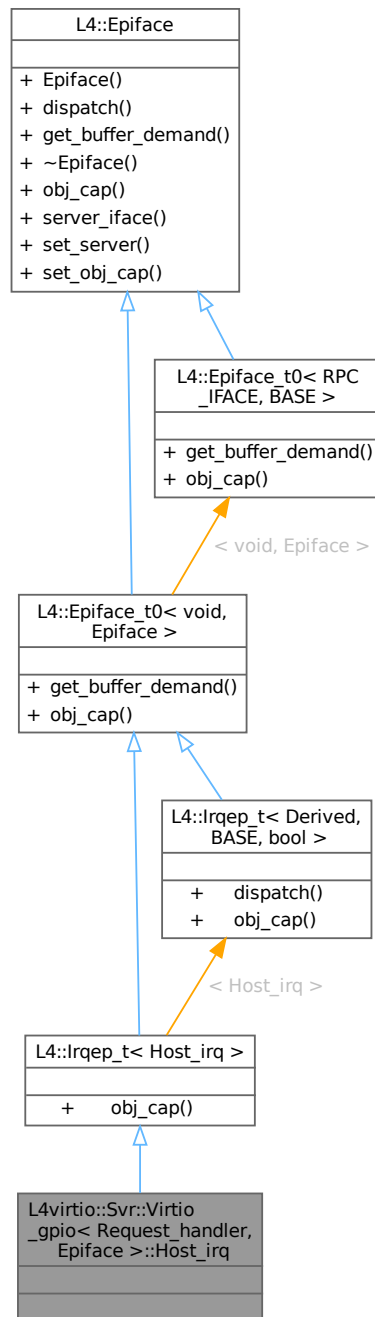
Handler for the host irq.

```
#include <virtio-gpio-device>
```

Inheritance diagram for L4virtio::Svr::Virtio\_gpio< Request\_handler, Epiface >::Host\_irq:



Collaboration diagram for L4virtio::Svr::Virtio\_gpio< Request\_handler, Epiface >::Host\_irq:



### Additional Inherited Members

### Public Types inherited from [L4::Epiface\\_t0< void, Epiface >](#)

- typedef void **Interface**

*Data type of the IPC interface definition.*



## Public Types inherited from L4::Epiface

- typedef [lpc\\_svr::Server\\_iface](#) **Server\_iface**  
*Type for abstract server interface.*
- typedef [lpc\\_svr::Server\\_iface::Demand](#) **Demand**  
*Type for server-side receive buffer demand.*

## Public Member Functions inherited from L4::lrqep\_t< Host\_irq >

- [Cap< L4::lrq >](#) **obj\_cap** () const  
*Get the (typed) capability to this object.*

## Public Member Functions inherited from L4::Epiface\_t0< void, Epiface >

- [Type\\_info::Demand](#) **get\_buffer\_demand** () const  
*Get the server-side buffer demand based in IFACE.*
- [Cap< void >](#) **obj\_cap** () const  
*Get the (typed) capability to this object.*

## Public Member Functions inherited from L4::Epiface

- **Epiface** ()  
*Make a server object.*
- virtual **~Epiface** ()=0  
*Destroy the object.*
- Stored\_cap **obj\_cap** () const  
*Get the capability to the kernel object belonging to this object.*
- [Server\\_iface](#) \* **server\_iface** () const  
*Get pointer to server interface at which the object is currently registered.*
- int **set\_server** ([Server\\_iface](#) \*srv, [Cap< void >](#) cap, bool managed=false)  
*Set server registration info for the object.*
- void **set\_obj\_cap** ([Cap< void >](#) const &cap)  
*Deprecated server registration function.*

### 16.429.1 Detailed Description

```
template<typename Request_handler, typename Epiface = L4virtio::Device>
struct L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::Host_irq
```

Handler for the host irq.

An [L4::lrqep\\_t](#) to handle irq's send to the server.

Definition at line 198 of file [virtio-gpio-device](#).

The documentation for this struct was generated from the following file:

- l4/l4virtio/server/virtio-gpio-device

## 16.430 L4virtio::Svr::Virtio\_gpio< Request\_handler, Epiface >::Irq\_handler Struct Reference

Handler for an gpio pin irq.

```
#include <virtio-gpio-device>
```

Collaboration diagram for L4virtio::Svr::Virtio\_gpio< Request\_handler, Epiface >::Irq\_handler:



### 16.430.1 Detailed Description

```
template<typename Request_handler, typename Epiface = L4virtio::Device>
struct L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::Irq_handler
```

Handler for an gpio pin irq.

This notifies the virtio client that an gpio pin irq happened or the operation was canceled.

This needs to be called by any server implementation of the [Virtio\\_gpio](#) class, after an irq was enabled with the enable method of the Gpio\_request\_handler.

Definition at line 166 of file [virtio-gpio-device](#).

The documentation for this struct was generated from the following file:

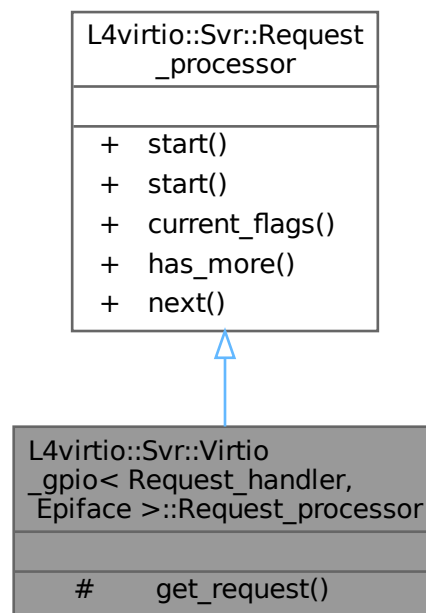
- I4/I4virtio/server/virtio-gpio-device

## 16.431 L4virtio::Svr::Virtio\_gpio< Request\_handler, Epiface >::Request\_processor Struct Reference

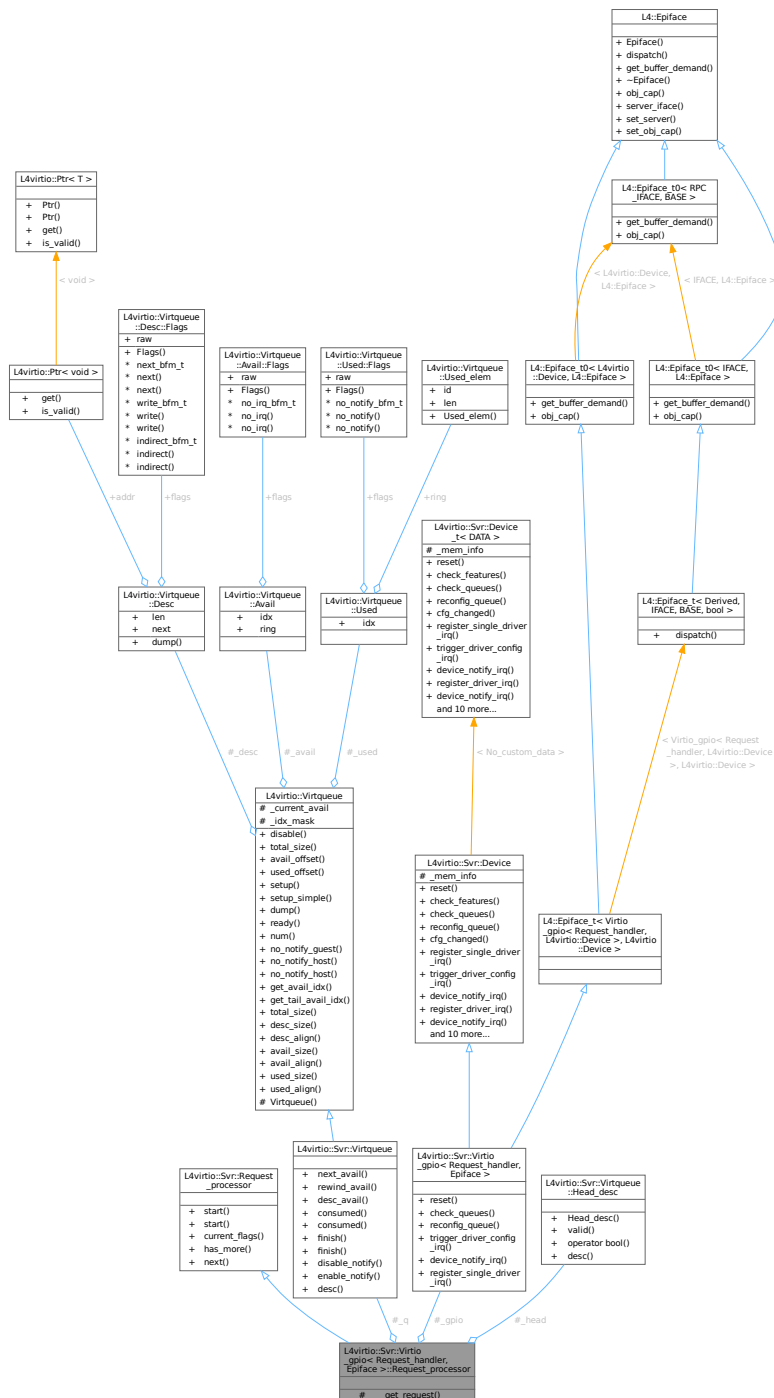
Generic handler for the Virtio requests.

```
#include <virtio-gpio-device>
```

Inheritance diagram for L4virtio::Svr::Virtio\_gpio< Request\_handler, Epiface >::Request\_processor:



Collaboration diagram for L4virtio::Svr::Virtio\_gpio< Request\_handler, Epiface >::Request\_processor:



## Protected Member Functions

- template<typename T>  
T **get\_request** ()

The driver prepares the GPIO request in two data parts: 1st: in\_hdr 2rd: out\_hdr.

## Additional Inherited Members

### Public Member Functions inherited from L4virtio::Svr::Request\_processor

- template<typename DESC\_MAN, typename ... ARGS>  
void [start](#) (DESC\_MAN \*dm, [Virtqueue](#) \*ring, [Virtqueue::Head\\_desc](#) const &request, ARGS... args)  
*Start processing a new request.*
- template<typename DESC\_MAN, typename ... ARGS>  
[Virtqueue::Request](#) const & [start](#) (DESC\_MAN \*dm, [Virtqueue::Request](#) const &request, ARGS... args)  
*Start processing a new request.*
- [Virtqueue::Desc::Flags](#) [current\\_flags](#) () const  
*Get the flags of the currently processed descriptor.*
- bool [has\\_more](#) () const  
*Are there more chained descriptors?*
- template<typename DESC\_MAN, typename ... ARGS>  
bool [next](#) (DESC\_MAN \*dm, ARGS... args)  
*Switch to the next descriptor in a descriptor chain.*

## 16.431.1 Detailed Description

template<typename Request\_handler, typename [Epiface](#) = L4virtio::Device>  
struct L4virtio::Svr::Virtio\_gpio< Request\_handler, Epiface >::Request\_processor

Generic handler for the Virtio requests.

Definition at line 213 of file [virtio-gpio-device](#).

## 16.431.2 Member Function Documentation

### 16.431.2.1 get\_request()

```
template<typename Request_handler, typename Epiface = L4virtio::Device>
template<typename T>
T L4virtio::Svr::Virtio\_gpio< Request\_handler, Epiface >::Request\_processor::get\_request ()
[inline], [protected]
```

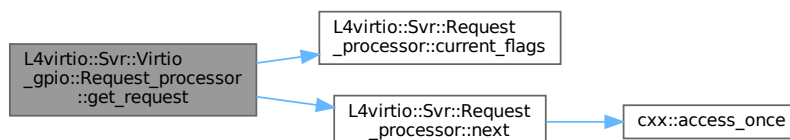
The driver prepares the GPIO request in two data parts: 1st: in\_hdr 2rd: out\_hdr.

This parses the two Data\_buffers and create the Gpio\_\* structure.

Definition at line 241 of file [virtio-gpio-device](#).

References [L4virtio::Svr::Request\\_processor::current\\_flags\(\)](#), and [L4virtio::Svr::Request\\_processor::next\(\)](#).

Here is the call graph for this function:



The documentation for this struct was generated from the following file:

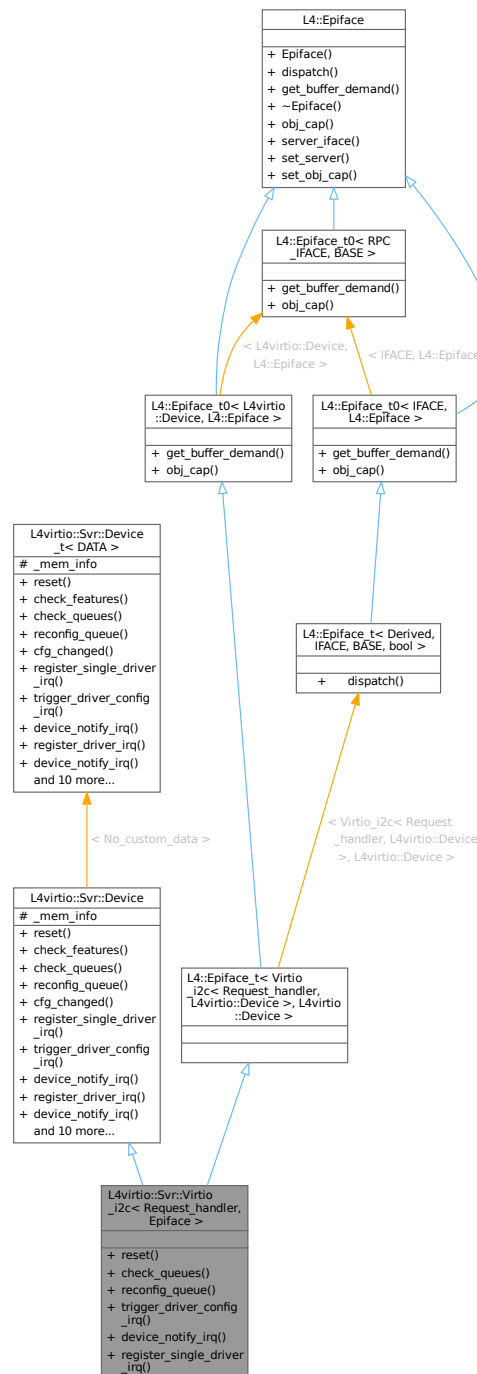
- I4/I4virtio/server/virtio-gpio-device

## 16.432 L4virtio::Svr::Virtio\_i2c< Request\_handler, Epiface > Class Template Reference

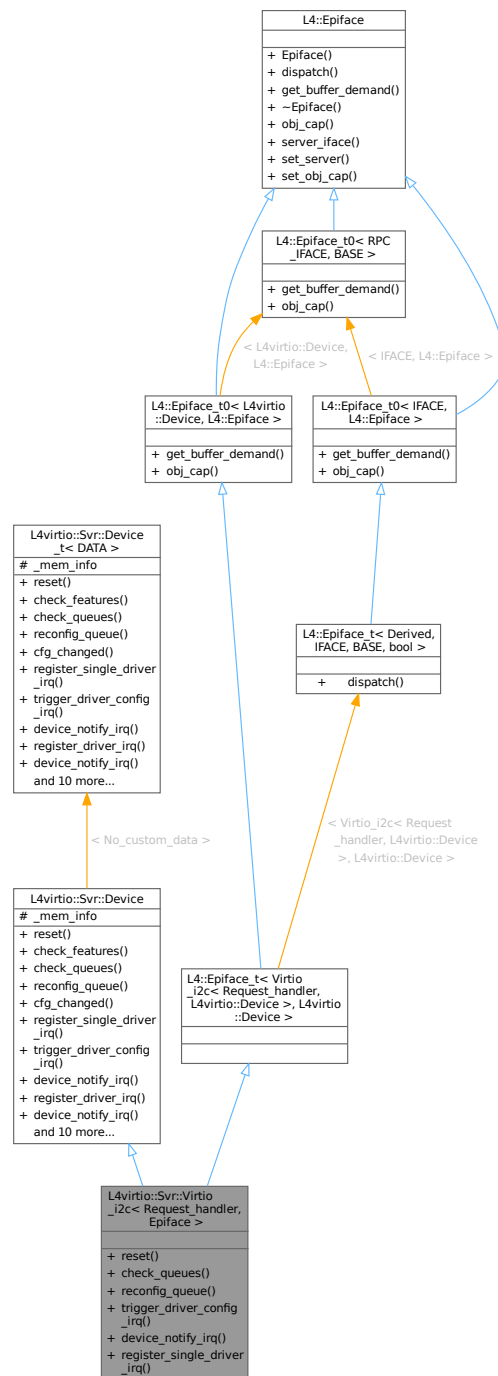
A server implementation of the virtio-i2c protocol.

```
#include <virtio-i2c-device>
```

Inheritance diagram for L4virtio::Svr::Virtio\_i2c< Request\_handler, Epiface >:



Collaboration diagram for L4virtio::Svr::Virtio\_i2c< Request\_handler, Epiface >:



## Data Structures

- class `Host_irq`  
*Handler for the host irq.*
- class `Request_processor`  
*Handler for the Virtio requests.*

## Public Member Functions

- void **reset** () override  
*reset callback, called for doing a device reset*
- bool **check\_queues** () override  
*callback for checking if the queues at DRIVER\_OK transition*
- int **reconfig\_queue** (unsigned idx) override  
*callback for client queue-config request*
- void **trigger\_driver\_config\_irq** () override  
*callback for triggering configuration change notification IRQ*
- [L4::Cap](#) < [L4::Irq](#) > **device\_notify\_irq** () const override  
*callback to gather the device notification IRQ (old-style)*
- void **register\_single\_driver\_irq** () override  
*callback for registering a single guest IRQ for all queues (old-style)*

## Public Member Functions inherited from [L4virtio::Svr::Device\\_t](#) < [No\\_custom\\_data](#) >

- virtual bool **check\_features** ()  
*callback for checking the subset of accepted features*
- virtual void **cfg\_changed** (unsigned)  
*callback for client device configuration changes*
- virtual void **register\_driver\_irq** (unsigned idx)  
*Callback for registering an notification IRQ (multi IRQ).*
- virtual [L4::Cap](#) < [L4::Irq](#) > **device\_notify\_irq** (unsigned idx)  
*Callback to gather the device notification IRQ (multi IRQ).*
- virtual unsigned **num\_events\_supported** () const  
*Return the highest notification index supported.*
- [Device\\_t](#) ([Dev\\_config](#) \*dev\_config)  
*Make a device for the given config.*
- [Mem\\_list](#) const \* **mem\_info** () const  
*Get the memory region list used for this device.*
- void **reset\_queue\_config** (unsigned idx, unsigned num\_max, bool inc\_generation=false)  
*Trigger reset for the configuration space for queue idx.*
- void **init\_mem\_info** (unsigned num)  
*Initialize the memory region list to the given maximum.*
- void **device\_error** ()  
*Transition device into DEVICE\_NEEDS\_RESET state.*
- bool **setup\_queue** ([Virtqueue](#) \*q, unsigned qn, unsigned num\_max)  
*Enable/disable the specified queue.*
- bool **handle\_mem\_cmd\_write** ()  
*Check for a value in the cmd register and handle a write.*
- void **enable\_trusted\_ds\_validation** ()  
*Enable trusted dataspace validation.*
- void **add\_trusted\_dataspaces** (std::shared\_ptr < [Ds\\_vector](#) const > ds)  
*Provide a list of trusted dataspaces that can be used for validation.*

## Public Member Functions inherited from [L4::Epiface\\_t0](#) < [L4virtio::Device](#), [L4::Epiface](#) >

- [Type\\_info::Demand](#) **get\_buffer\_demand** () const  
*Get the server-side buffer demand based in IFACE.*
- [Cap](#) < [L4virtio::Device](#) > **obj\_cap** () const  
*Get the (typed) capability to this object.*



## Public Member Functions inherited from L4::Epiface

- **Epiface** ()  
*Make a server object.*
- virtual ~**Epiface** ()=0  
*Destroy the object.*
- Stored\_cap **obj\_cap** () const  
*Get the capability to the kernel object belonging to this object.*
- **Server\_iface** \* **server\_iface** () const  
*Get pointer to server interface at which the object is currently registered.*
- int **set\_server** (**Server\_iface** \*srv, **Cap**< void > cap, bool managed=false)  
*Set server registration info for the object.*
- void **set\_obj\_cap** (**Cap**< void > const &cap)  
*Deprecated server registration function.*

## Additional Inherited Members

## Public Types inherited from L4::Epiface\_t0< L4virtio::Device, L4::Epiface >

- typedef L4virtio::Device **Interface**  
*Data type of the IPC interface definition.*

## Public Types inherited from L4::Epiface

- typedef **lpc\_svr::Server\_iface** **Server\_iface**  
*Type for abstract server interface.*
- typedef **lpc\_svr::Server\_iface::Demand** **Demand**  
*Type for server-side receive buffer demand.*

## Protected Attributes inherited from L4virtio::Svr::Device\_t< No\_custom\_data >

- **Mem\_list** **\_mem\_info**  
*Memory region list.*

## 16.432.1 Detailed Description

```
template<typename Request_handler, typename Epiface = L4virtio::Device>
class L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >
```

A server implementation of the virtio-i2c protocol.

### Template Parameters

<i>Request_handler</i>	The type that is used to handle incoming requests. Needs to have <code>handle_read(l4_uint8_t *, unsigned)</code> and <code>handle_write(l4_uint8_t const *, unsigned)</code> functions.
------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<i>Epiface</i>	The Epiface to derive from. Defaults to <a href="#">L4virtio::Device</a> .
----------------	----------------------------------------------------------------------------

Definition at line 86 of file [virtio-i2c-device](#).

The documentation for this class was generated from the following file:

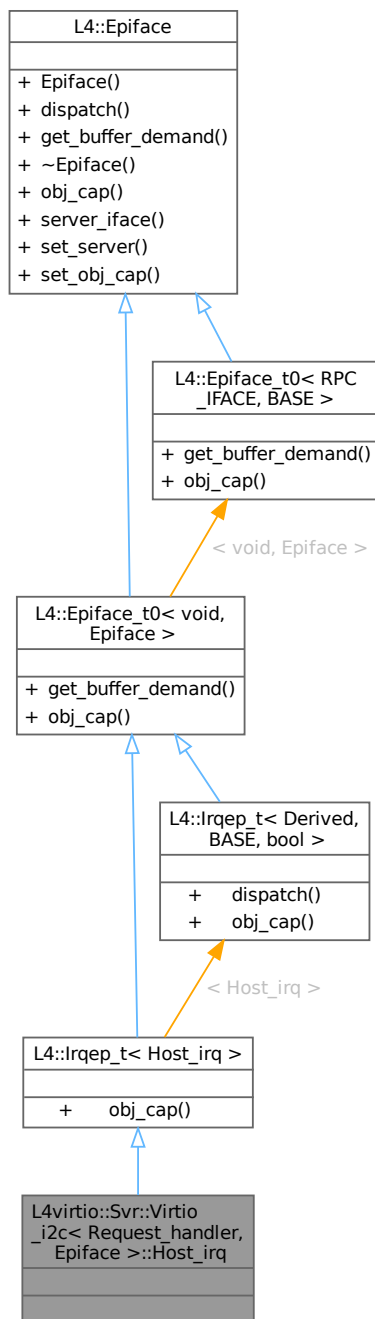
- l4/l4virtio/server/virtio-i2c-device

### 16.433 L4virtio::Svr::Virtio\_i2c< Request\_handler, Epiface >::Host\_irq Class Reference

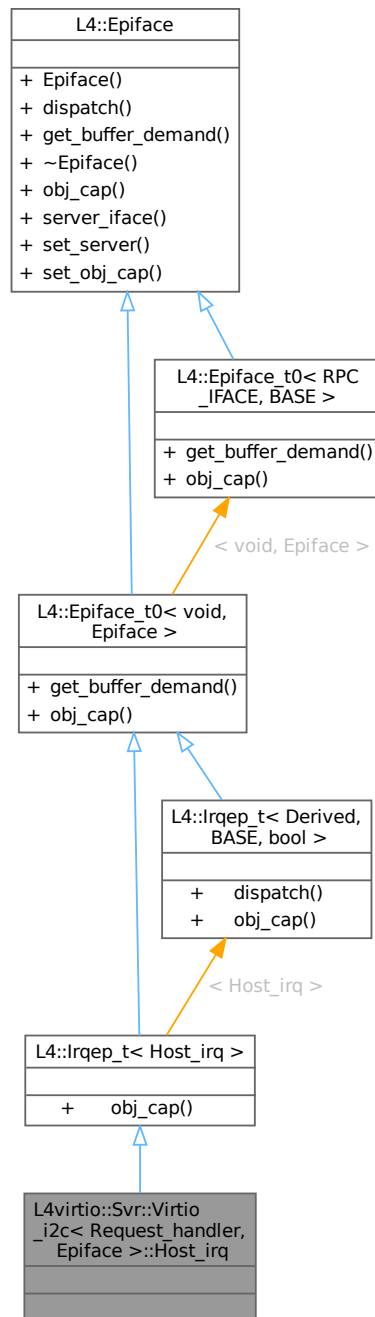
Handler for the host irq.

```
#include <virtio-i2c-device>
```

Inheritance diagram for L4virtio::Svr::Virtio\_i2c< Request\_handler, Epiface >::Host\_irq:



Collaboration diagram for L4virtio::Svr::Virtio\_i2c< Request\_handler, Epiface >::Host\_irq:



### Additional Inherited Members

### Public Types inherited from **L4::Epiface\_t0< void, Epiface >**

- typedef void **Interface**

*Data type of the IPC interface definition.*

## Public Types inherited from L4::Epiface

- typedef [lpc\\_svr::Server\\_iface](#) **Server\_iface**  
*Type for abstract server interface.*
- typedef [lpc\\_svr::Server\\_iface::Demand](#) **Demand**  
*Type for server-side receive buffer demand.*

## Public Member Functions inherited from L4::lrqep\_t< Host\_irq >

- [Cap< L4::lrq >](#) **obj\_cap** () const  
*Get the (typed) capability to this object.*

## Public Member Functions inherited from L4::Epiface\_t0< void, Epiface >

- [Type\\_info::Demand](#) **get\_buffer\_demand** () const  
*Get the server-side buffer demand based in IFACE.*
- [Cap< void >](#) **obj\_cap** () const  
*Get the (typed) capability to this object.*

## Public Member Functions inherited from L4::Epiface

- **Epiface** ()  
*Make a server object.*
- virtual **~Epiface** ()=0  
*Destroy the object.*
- Stored\_cap **obj\_cap** () const  
*Get the capability to the kernel object belonging to this object.*
- [Server\\_iface](#) \* **server\_iface** () const  
*Get pointer to server interface at which the object is currently registered.*
- int **set\_server** ([Server\\_iface](#) \*srv, [Cap< void >](#) cap, bool managed=false)  
*Set server registration info for the object.*
- void **set\_obj\_cap** ([Cap< void >](#) const &cap)  
*Deprecated server registration function.*

### 16.433.1 Detailed Description

```
template<typename Request_handler, typename Epiface = L4virtio::Device>
class L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >::Host_irq
```

Handler for the host irq.

An [L4::lrqep\\_t](#) to handle irq's send to the server.

Definition at line 106 of file [virtio-i2c-device](#).

The documentation for this class was generated from the following file:

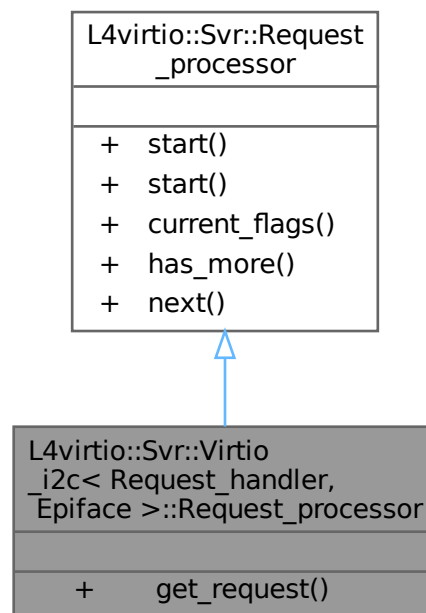
- I4/I4virtio/server/virtio-i2c-device

## 16.434 L4virtio::Svr::Virtio\_i2c< Request\_handler, Epiface >::Request\_processor Class Reference

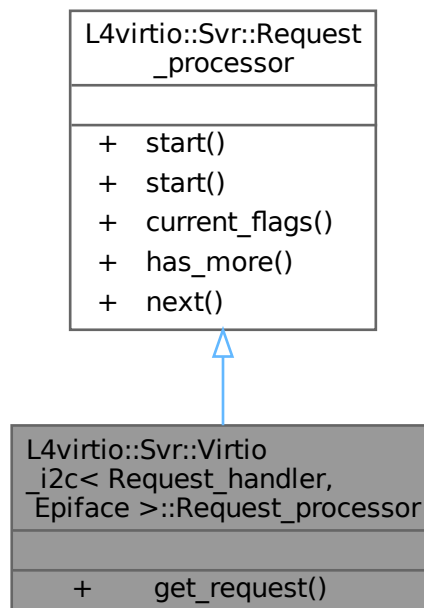
Handler for the Virtio requests.

```
#include <virtio-i2c-device>
```

Inheritance diagram for L4virtio::Svr::Virtio\_i2c< Request\_handler, Epiface >::Request\_processor:



Collaboration diagram for L4virtio::Svr::Virtio\_i2c< Request\_handler, Epiface >::Request\_processor:



## Public Member Functions

- I2c\_req [get\\_request](#) ()

*Linux prepares the I2C request in three data parts: 1st: out\_hdr 2nd: buffer (optional) 3rd: in\_hdr.*

## Public Member Functions inherited from [L4virtio::Svr::Request\\_processor](#)

- template<typename DESC\_MAN, typename ... ARGS>  
void [start](#) (DESC\_MAN \*dm, [Virtqueue](#) \*ring, [Virtqueue::Head\\_desc](#) const &request, ARGS... args)  
*Start processing a new request.*
- template<typename DESC\_MAN, typename ... ARGS>  
[Virtqueue::Request](#) const & [start](#) (DESC\_MAN \*dm, [Virtqueue::Request](#) const &request, ARGS... args)  
*Start processing a new request.*
- [Virtqueue::Desc::Flags](#) [current\\_flags](#) () const  
*Get the flags of the currently processed descriptor.*
- bool [has\\_more](#) () const  
*Are there more chained descriptors?*
- template<typename DESC\_MAN, typename ... ARGS>  
bool [next](#) (DESC\_MAN \*dm, ARGS... args)  
*Switch to the next descriptor in a descriptor chain.*

### 16.434.1 Detailed Description

```
template<typename Request_handler, typename Epiface = L4virtio::Device>
class L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >::Request_processor
```

Handler for the Virtio requests.

Definition at line 123 of file [virtio-i2c-device](#).

### 16.434.2 Member Function Documentation

#### 16.434.2.1 get\_request()

```
template<typename Request_handler, typename Epiface = L4virtio::Device>
I2c_req L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >::Request_processor::get_request
() [inline]
```

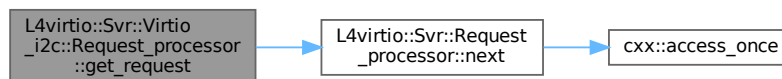
Linux prepares the I2C request in three data parts: 1st: out\_hdr 2nd: buffer (optional) 3rd: in\_hdr.

This parses the three Data\_buffers and recreate the virtio\_i2c\_req structure.

Definition at line 172 of file [virtio-i2c-device](#).

References [L4virtio::Svr::Data\\_buffer::left](#), [L4virtio::Svr::Request\\_processor::next\(\)](#), and [L4virtio::Svr::Data\\_buffer::pos](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- [l4/l4virtio/server/virtio-i2c-device](#)

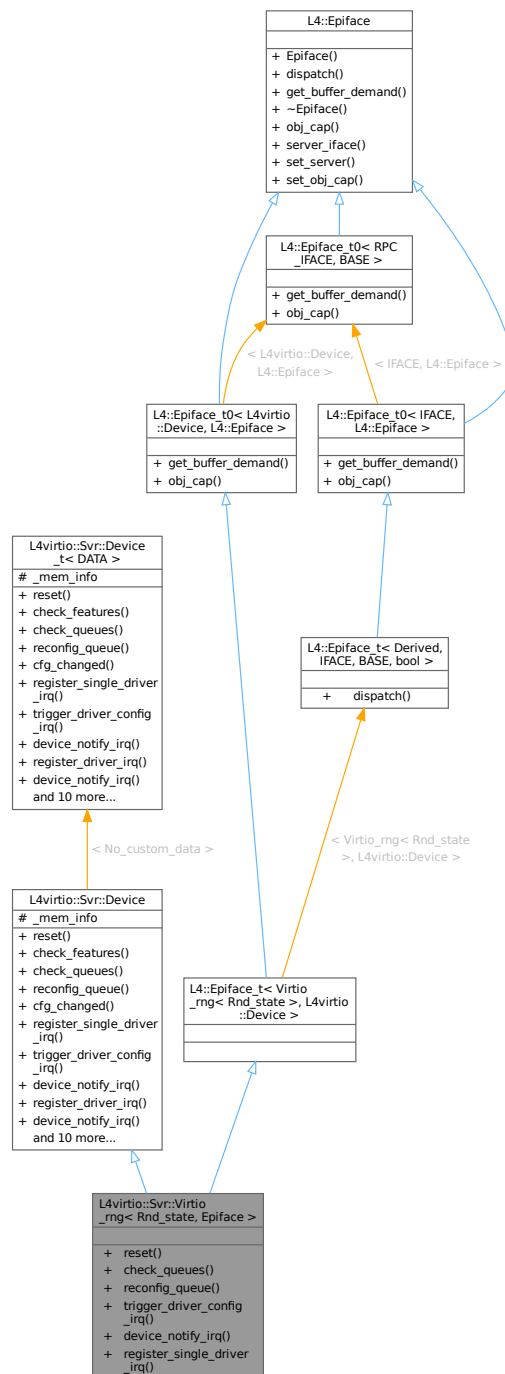


## 16.435 L4virtio::Svr::Virtio\_rng< Rnd\_state, Epiface > Class Template Reference

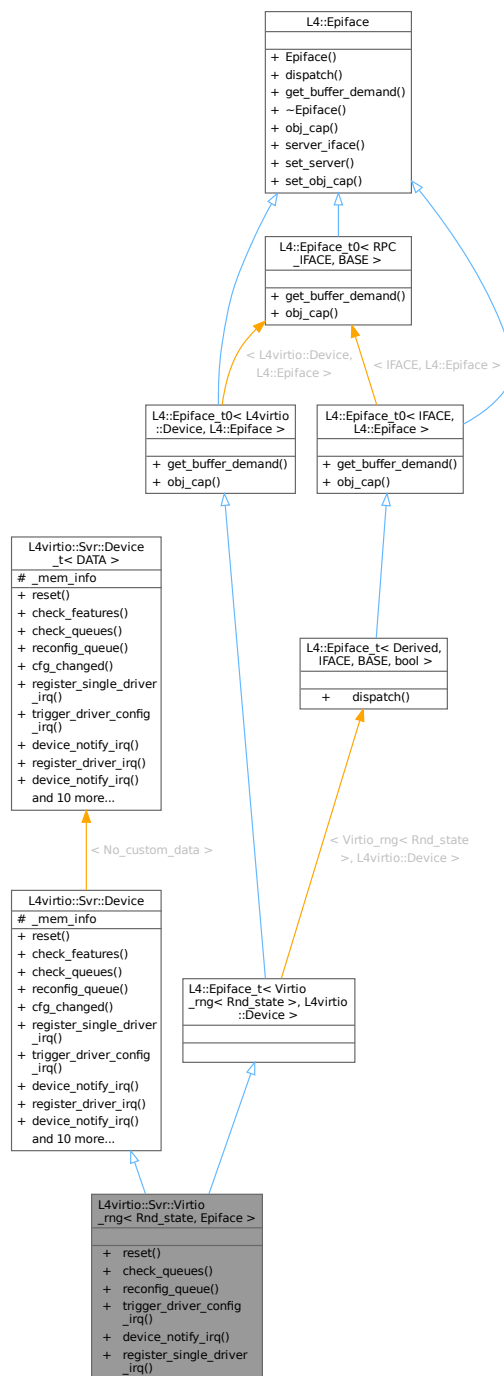
A server implementation of the virtio-rng protocol.

```
#include <virtio-rng-device>
```

Inheritance diagram for L4virtio::Svr::Virtio\_rng< Rnd\_state, Epiface >:



Collaboration diagram for L4virtio::Svr::Virtio\_rng< Rnd\_state, Epiface >:



## Data Structures

- class [Host\\_irq](#)  
Handler for the host irq.
- class [Request\\_processor](#)  
Handler for the Virtio requests.

**Public Member Functions**

- void **reset** () override  
*reset callback, called for doing a device reset*
- bool **check\_queues** () override  
*callback for checking if the queues at DRIVER\_OK transition*
- int **reconfig\_queue** (unsigned idx) override  
*callback for client queue-config request*
- void **trigger\_driver\_config\_irq** () override  
*callback for triggering configuration change notification IRQ*
- L4::Cap< L4::lrq > **device\_notify\_irq** () const override  
*callback to gather the device notification IRQ (old-style)*
- void **register\_single\_driver\_irq** () override  
*callback for registering a single guest IRQ for all queues (old-style)*

**Public Member Functions inherited from L4virtio::Svr::Device\_t< No\_custom\_data >**

- virtual bool **check\_features** ()  
*callback for checking the subset of accepted features*
- virtual void **cfg\_changed** (unsigned)  
*callback for client device configuration changes*
- virtual void **register\_driver\_irq** (unsigned idx)  
*Callback for registering an notification IRQ (multi IRQ).*
- virtual L4::Cap< L4::lrq > **device\_notify\_irq** (unsigned idx)  
*Callback to gather the device notification IRQ (multi IRQ).*
- virtual unsigned **num\_events\_supported** () const  
*Return the highest notification index supported.*
- **Device\_t** (**Dev\_config** \*dev\_config)  
*Make a device for the given config.*
- **Mem\_list** const \* **mem\_info** () const  
*Get the memory region list used for this device.*
- void **reset\_queue\_config** (unsigned idx, unsigned num\_max, bool inc\_generation=false)  
*Trigger reset for the configuration space for queue idx.*
- void **init\_mem\_info** (unsigned num)  
*Initialize the memory region list to the given maximum.*
- void **device\_error** ()  
*Transition device into DEVICE\_NEEDS\_RESET state.*
- bool **setup\_queue** (**Virtqueue** \*q, unsigned qn, unsigned num\_max)  
*Enable/disable the specified queue.*
- bool **handle\_mem\_cmd\_write** ()  
*Check for a value in the cmd register and handle a write.*
- void **enable\_trusted\_ds\_validation** ()  
*Enable trusted dataspace validation.*
- void **add\_trusted\_dataspaces** (std::shared\_ptr< Ds\_vector const > ds)  
*Provide a list of trusted dataspaces that can be used for validation.*

**Public Member Functions inherited from L4::Epiface\_t0< L4virtio::Device, L4::Epiface >**

- **Type\_info::Demand** **get\_buffer\_demand** () const  
*Get the server-side buffer demand based in IFACE.*
- **Cap**< L4virtio::Device > **obj\_cap** () const  
*Get the (typed) capability to this object.*

## Public Member Functions inherited from [L4::Epiface](#)

- **Epiface** ()  
*Make a server object.*
- virtual **~Epiface** ()=0  
*Destroy the object.*
- Stored\_cap **obj\_cap** () const  
*Get the capability to the kernel object belonging to this object.*
- [Server\\_iface](#) \* **server\_iface** () const  
*Get pointer to server interface at which the object is currently registered.*
- int **set\_server** ([Server\\_iface](#) \*srv, [Cap](#)< void > cap, bool managed=false)  
*Set server registration info for the object.*
- void **set\_obj\_cap** ([Cap](#)< void > const &cap)  
*Deprecated server registration function.*

## Additional Inherited Members

## Public Types inherited from [L4::Epiface\\_t0](#)< [L4virtio::Device](#), [L4::Epiface](#) >

- typedef [L4virtio::Device](#) **Interface**  
*Data type of the IPC interface definition.*

## Public Types inherited from [L4::Epiface](#)

- typedef [lpc\\_svr::Server\\_iface](#) **Server\_iface**  
*Type for abstract server interface.*
- typedef [lpc\\_svr::Server\\_iface::Demand](#) **Demand**  
*Type for server-side receive buffer demand.*

## Protected Attributes inherited from [L4virtio::Svr::Device\\_t](#)< [No\\_custom\\_data](#) >

- [Mem\\_list](#) **\_mem\_info**  
*Memory region list.*

## 16.435.1 Detailed Description

```
template<typename Rnd_state, typename Epiface = L4virtio::Device>
class L4virtio::Svr::Virtio\_rng< Rnd\_state, Epiface >
```

A server implementation of the virtio-rng protocol.

## Template Parameters

---

<i>Rnd_state</i>	The type that implements the random data generation. <code>Rnd_state::get_random(int len, unsigned char *buf)</code> is called to get len random bytes written into buf TODO: virtio-rng supports providing less random bytes then requested. This API currently does not support that, as I do not have a test case.
<i>Epiface</i>	The Epiface to derive from. Defaults to <code>L4virtio::Device</code> .

Definition at line 33 of file [virtio-rng-device](#).

The documentation for this class was generated from the following file:

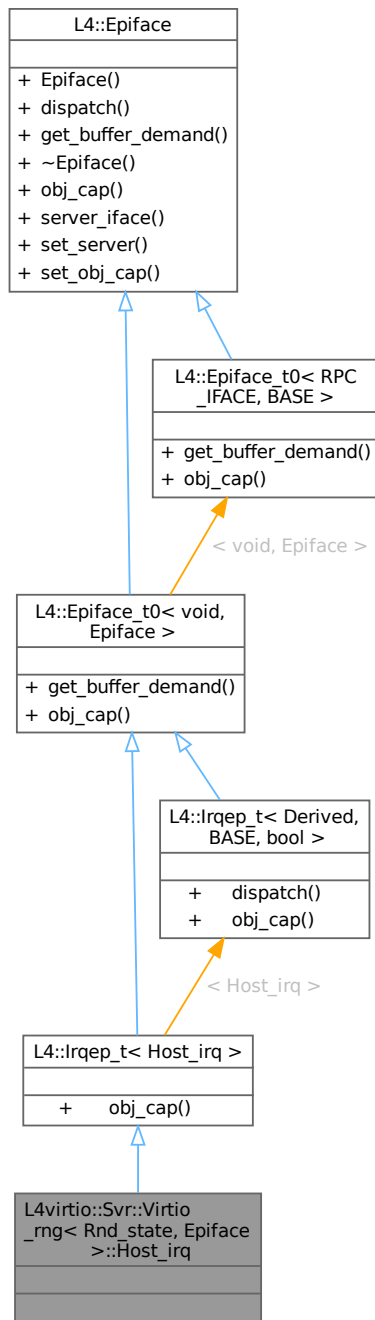
- I4/I4virtio/server/virtio-rng-device

## 16.436 L4virtio::Svr::Virtio\_rng< Rnd\_state, Epiface >::Host\_irq Class Reference

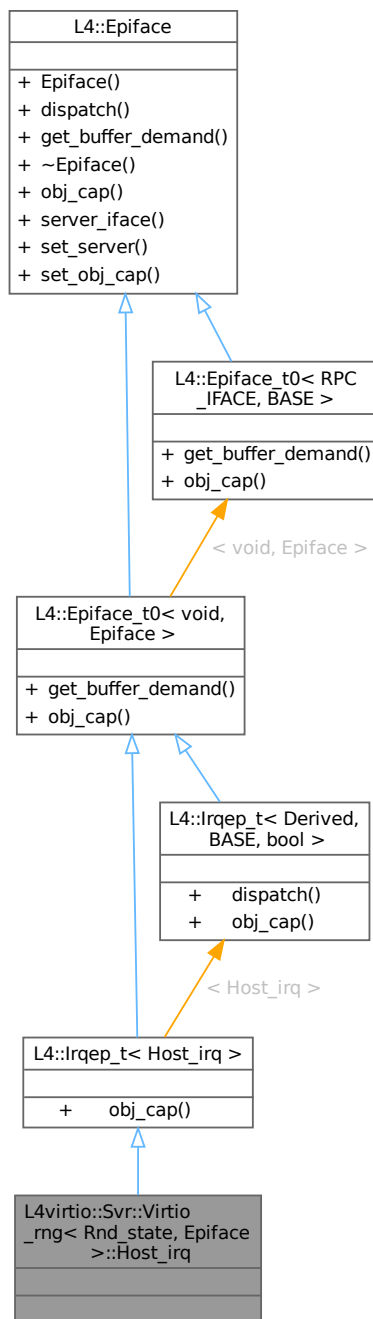
Handler for the host irq.

```
#include <virtio-rng-device>
```

Inheritance diagram for L4virtio::Svr::Virtio\_rng< Rnd\_state, Epiface >::Host\_irq:



Collaboration diagram for L4virtio::Svr::Virtio\_rng< Rnd\_state, Epiface >::Host\_irq:



#### Additional Inherited Members

#### Public Types inherited from **L4::Epiface\_t0< void, Epiface >**

- typedef void **Interface**

*Data type of the IPC interface definition.*

## Public Types inherited from [L4::Epiface](#)

- typedef [lpc\\_svr::Server\\_iface](#) **Server\_iface**  
*Type for abstract server interface.*
- typedef [lpc\\_svr::Server\\_iface::Demand](#) **Demand**  
*Type for server-side receive buffer demand.*

## Public Member Functions inherited from [L4::lrqep\\_t< Host\\_irq >](#)

- [Cap< L4::lrq > obj\\_cap](#) () const  
*Get the (typed) capability to this object.*

## Public Member Functions inherited from [L4::Epiface\\_t0< void, Epiface >](#)

- [Type\\_info::Demand](#) **get\_buffer\_demand** () const  
*Get the server-side buffer demand based in IFACE.*
- [Cap< void > obj\\_cap](#) () const  
*Get the (typed) capability to this object.*

## Public Member Functions inherited from [L4::Epiface](#)

- **Epiface** ()  
*Make a server object.*
- virtual **~Epiface** ()=0  
*Destroy the object.*
- Stored\_cap [obj\\_cap](#) () const  
*Get the capability to the kernel object belonging to this object.*
- [Server\\_iface](#) \* [server\\_iface](#) () const  
*Get pointer to server interface at which the object is currently registered.*
- int **set\_server** ([Server\\_iface](#) \*srv, [Cap< void > cap](#), bool managed=false)  
*Set server registration info for the object.*
- void **set\_obj\_cap** ([Cap< void > const &cap](#))  
*Deprecated server registration function.*

### 16.436.1 Detailed Description

```
template<typename Rnd_state, typename Epiface = L4virtio::Device>
class L4virtio::Svr::Virtio_rng< Rnd_state, Epiface >::Host_irq
```

Handler for the host irq.

An [L4::lrqep\\_t](#) to handle irq's send to the server.

Definition at line 51 of file [virtio-rng-device](#).

The documentation for this class was generated from the following file:

- I4/I4virtio/server/virtio-rng-device

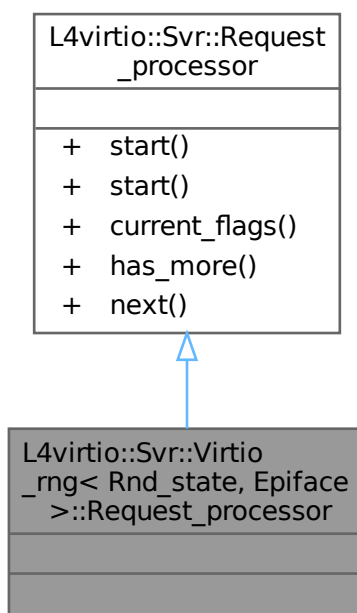


## 16.437 L4virtio::Svr::Virtio\_rng< Rnd\_state, Epiface >::Request\_processor Class Reference

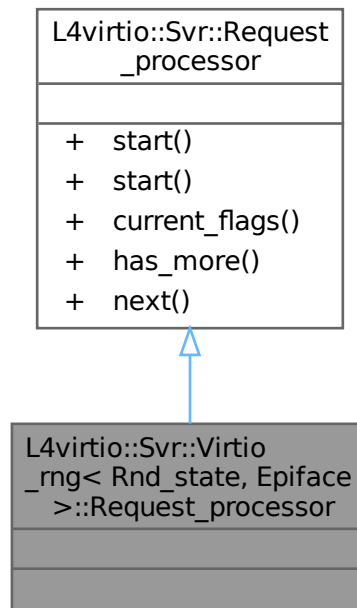
Handler for the Virtio requests.

```
#include <virtio-rng-device>
```

Inheritance diagram for L4virtio::Svr::Virtio\_rng< Rnd\_state, Epiface >::Request\_processor:



Collaboration diagram for L4virtio::Svr::Virtio\_rng< Rnd\_state, Epiface >::Request\_processor:



### Additional Inherited Members

### Public Member Functions inherited from L4virtio::Svr::Request\_processor

- template<typename DESC\_MAN, typename ... ARGS>  
void [start](#) (DESC\_MAN \*dm, [Virtqueue](#) \*ring, [Virtqueue::Head\\_desc](#) const &request, ARGS... args)  
*Start processing a new request.*
- template<typename DESC\_MAN, typename ... ARGS>  
[Virtqueue::Request](#) const & [start](#) (DESC\_MAN \*dm, [Virtqueue::Request](#) const &request, ARGS... args)  
*Start processing a new request.*
- [Virtqueue::Desc::Flags](#) [current\\_flags](#) () const  
*Get the flags of the currently processed descriptor.*
- bool [has\\_more](#) () const  
*Are there more chained descriptors?*
- template<typename DESC\_MAN, typename ... ARGS>  
bool [next](#) (DESC\_MAN \*dm, ARGS... args)  
*Switch to the next descriptor in a descriptor chain.*

## 16.437.1 Detailed Description

```
template<typename Rnd_state, typename Epiface = L4virtio::Device>
class L4virtio::Svr::Virtio_rng< Rnd_state, Epiface >::Request_processor
```

Handler for the Virtio requests.

Definition at line 68 of file [virtio-rng-device](#).

The documentation for this class was generated from the following file:

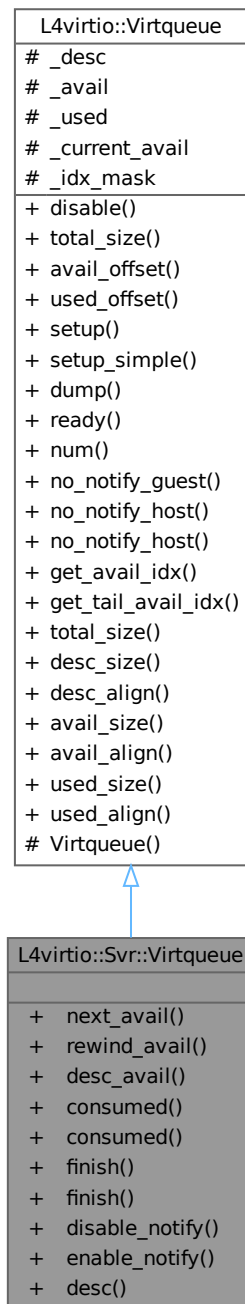
- l4/l4virtio/server/virtio-rng-device

## 16.438 L4virtio::Svr::Virtqueue Class Reference

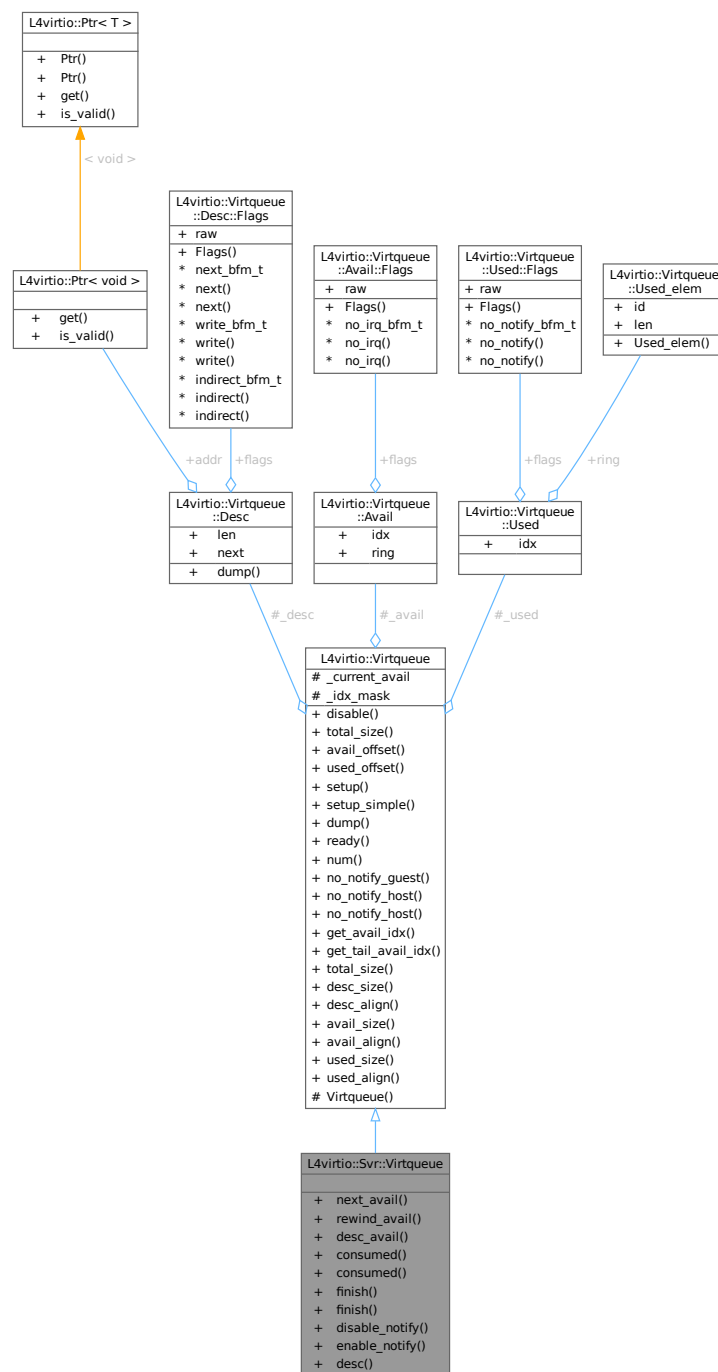
[Virtqueue](#) implementation for the device.

```
#include <virtio>
```

Inheritance diagram for L4virtio::Svr::Virtqueue:



Collaboration diagram for L4virtio::Svr::Virtqueue:



## Data Structures

- class [Head\\_desc](#)  
*VIRTIO request, essentially a descriptor from the available ring.*

## Public Member Functions

- Request [next\\_avail](#) ()

- Get the next available descriptor from the available ring.*

    - void `rewind_avail` (`Head_desc` const &d)

*Return unfinished descriptors to the available ring, i.e.*

  - bool `desc_avail` () const
- Test for available descriptors.*
- void `consumed` (`Head_desc` const &r, `l4_uint32_t` len=0)
- Put the given descriptor into the used ring.*
- template<typename ITER>  
void `consumed` (ITER const &begin, ITER const &end)
- Put multiple descriptors into the used ring.*
- template<typename QUEUE\_OBSERVER>  
void `finish` (`Head_desc` &d, QUEUE\_OBSERVER \*o, `l4_uint32_t` len=0)
- Add a descriptor to the used ring, and notify an observer.*
- template<typename ITER, typename QUEUE\_OBSERVER>  
void `finish` (ITER const &begin, ITER const &end, QUEUE\_OBSERVER \*o)
- Add a range of descriptors to the used ring, and notify an observer once.*
- void `disable_notify` ()
- Set the 'no notify' flag for this queue.*
- void `enable_notify` ()
- Clear the 'no notify' flag for this queue.*
- `Desc` const \* `desc` (unsigned idx) const
- Get a descriptor from the descriptor list.*

## Public Member Functions inherited from `L4virtio::Virtqueue`

- void `disable` ()
- Completely disable the queue.*
- unsigned long `total_size` () const
- Calculate the total size of this virtqueue.*
- unsigned long `avail_offset` () const
- Get the offset of the available ring from the descriptor table.*
- unsigned long `used_offset` () const
- Get the offset of the used ring from the descriptor table.*
- void `setup` (unsigned `num`, void \*desc, void \*avail, void \*used)
- Enable this queue.*
- void `setup_simple` (unsigned `num`, void \*ring)
- Enable this queue.*
- void `dump` (`Desc` const \*d) const
- Dump descriptors for this queue.*
- bool `ready` () const
- Test if this queue is in working state.*
- unsigned `num` () const
- bool `no_notify_guest` () const
- Get the no IRQ flag of this queue.*
- bool `no_notify_host` () const
- Get the no notify flag of this queue.*
- void `no_notify_host` (bool value)
- Set the no-notify flag for this queue.*
- `l4_uint16_t` `get_avail_idx` () const
- Get available index from available ring (for debugging).*
- `l4_uint16_t` `get_tail_avail_idx` () const
- Get tail-available index stored in local state (for debugging).*

## Additional Inherited Members

### Public Types inherited from L4virtio::Virtqueue

- enum  
*Fixed alignment values for different parts of a virtqueue.*

### Static Public Member Functions inherited from L4virtio::Virtqueue

- static unsigned long `total_size` (unsigned `num`)  
*Calculate the total size for a virtqueue of the given dimensions.*
- static unsigned long `desc_size` (unsigned `num`)  
*Calculate the size of the descriptor table for `num` entries.*
- static unsigned long `desc_align` ()  
*Get the alignment in zero LSBs needed for the descriptor table.*
- static unsigned long `avail_size` (unsigned `num`)  
*Calculate the size of the available ring for `num` entries.*
- static unsigned long `avail_align` ()  
*Get the alignment in zero LSBs needed for the available ring.*
- static unsigned long `used_size` (unsigned `num`)  
*Calculate the size of the used ring for `num` entries.*
- static unsigned long `used_align` ()  
*Get the alignment in zero LSBs needed for the used ring.*

### Protected Member Functions inherited from L4virtio::Virtqueue

- `Virtqueue` ()=default  
*Create a disabled virtqueue.*

### Protected Attributes inherited from L4virtio::Virtqueue

- `Desc * _desc` = nullptr  
*pointer to descriptor table, NULL if queue is off.*
- `Avail * _avail` = nullptr  
*pointer to available ring.*
- `Used * _used` = nullptr  
*pointer to used ring.*
- `l4_uint16_t _current_avail` = 0  
*The life counter for the queue.*
- `l4_uint16_t _idx_mask` = 0  
*mask used for indexing into the descriptor table and the rings.*

## 16.438.1 Detailed Description

`Virtqueue` implementation for the device.

This class represents a single virtqueue, with a local running available index.

#### Note

The `Virtqueue` implementation is not thread-safe.

Definition at line 87 of file `virtio`.

## 16.438.2 Member Function Documentation

### 16.438.2.1 consumed() [1/2]

```
void L4virtio::Svr::Virtqueue::consumed (
    Head_desc const & r,
    l4_uint32_t len = 0) [inline]
```

Put the given descriptor into the used ring.

#### Parameters

<i>r</i>	Request that shall be marked as finished.
<i>len</i>	The total number of bytes written.

#### Precondition

queue must be in working state.

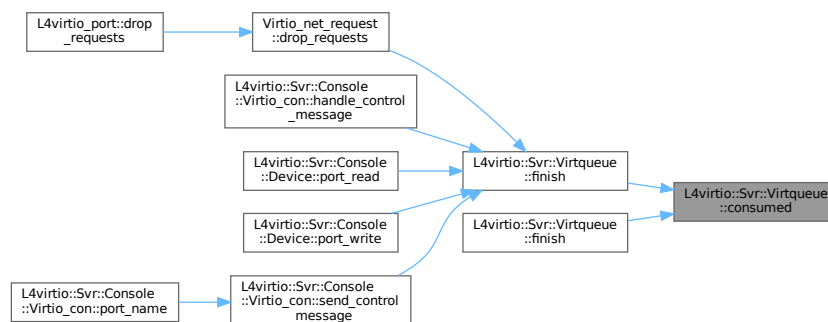
*r* must be a valid request from this queue.

Definition at line 190 of file [virtio](#).

References [L4virtio::Virtqueue::\\_desc](#), [L4virtio::Virtqueue::\\_idx\\_mask](#), and [L4virtio::Virtqueue::\\_used](#).

Referenced by [finish\(\)](#), and [finish\(\)](#).

Here is the caller graph for this function:



### 16.438.2.2 consumed() [2/2]

```
template<typename ITER>
void L4virtio::Svr::Virtqueue::consumed (
    ITER const & begin,
    ITER const & end) [inline]
```

Put multiple descriptors into the used ring.

A range of descriptors, specified by `begin` and `end` iterators is added. Each iterator points to a struct that has a first member that is a [Head\\_desc](#) and a second member that is the corresponding number of bytes written.

#### Template Parameters



<i>ITER</i>	The type of the iterator (inferred).
-------------	--------------------------------------

### Parameters

<i>begin</i>	Iterator pointing to first new descriptor.
<i>end</i>	Iterator pointing to one past last entry.

### Precondition

queue must be in working state.

Definition at line 213 of file [virtio](#).

References [L4virtio::Virtqueue::\\_desc](#), [L4virtio::Virtqueue::\\_idx\\_mask](#), and [L4virtio::Virtqueue::\\_used](#).

### 16.438.2.3 desc()

```
Desc const * L4virtio::Svr::Virtqueue::desc (
    unsigned idx) const [inline]
```

Get a descriptor from the descriptor list.

### Parameters

<i>idx</i>	The index of the descriptor.
------------	------------------------------

### Precondition

`idx < num`

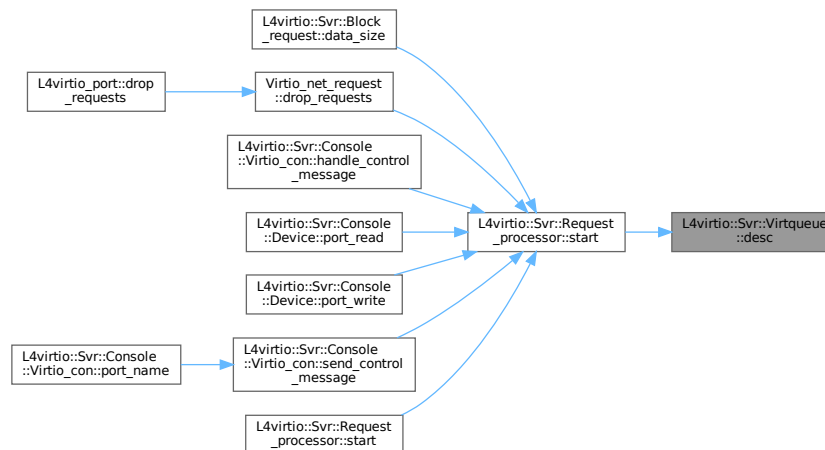
queue must be in working state

Definition at line 298 of file [virtio](#).

References [L4virtio::Virtqueue::\\_desc](#).

Referenced by [L4virtio::Svr::Request\\_processor::start\(\)](#).

Here is the caller graph for this function:



#### 16.438.2.4 desc\_avail()

```
bool L4virtio::Svr::Virtqueue::desc_avail () const [inline]
```

Test for available descriptors.

##### Returns

true if there are descriptors available, false if not.

##### Precondition

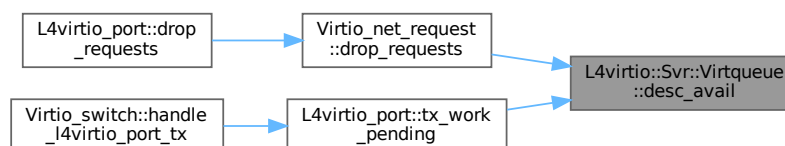
The queue must be in working state.

Definition at line 175 of file [virtio](#).

References [L4virtio::Virtqueue::\\_avail](#), and [L4virtio::Virtqueue::\\_current\\_avail](#).

Referenced by [Virtio\\_net\\_request::drop\\_requests\(\)](#), and [L4virtio\\_port::tx\\_work\\_pending\(\)](#).

Here is the caller graph for this function:



### 16.438.2.5 disable\_notify()

```
void L4virtio::Svr::Virtqueue::disable_notify () [inline]
```

Set the 'no notify' flag for this queue.

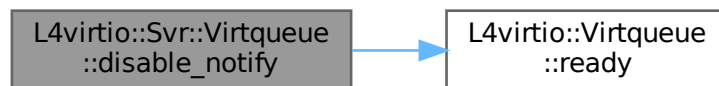
This function may be called on a disabled queue.

Definition at line 273 of file [virtio](#).

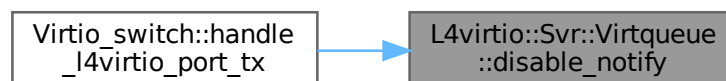
References [L4virtio::Virtqueue::\\_used](#), [L4\\_LIKELY](#), and [L4virtio::Virtqueue::ready\(\)](#).

Referenced by [Virtio\\_switch::handle\\_l4virtio\\_port\\_tx\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.438.2.6 enable\_notify()

```
void L4virtio::Svr::Virtqueue::enable_notify () [inline]
```

Clear the 'no notify' flag for this queue.

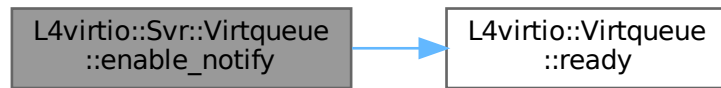
This function may be called on a disabled queue.

Definition at line 284 of file [virtio](#).

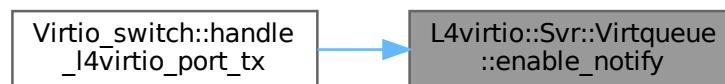
References [L4virtio::Virtqueue::\\_used](#), [L4\\_LIKELY](#), and [L4virtio::Virtqueue::ready\(\)](#).

Referenced by [Virtio\\_switch::handle\\_l4virtio\\_port\\_tx\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.438.2.7 finish() [1/2]

```

template<typename QUEUE_OBSERVER>
void L4virtio::Svr::Virtqueue::finish (
    Head_desc & d,
    QUEUE_OBSERVER * o,
    l4_uint32_t len = 0) [inline]
  
```

Add a descriptor to the used ring, and notify an observer.

#### Template Parameters

<code>QUEUE_OBSERVER</code>	The type of the observer (inferred).
-----------------------------	--------------------------------------

#### Parameters

<code>d</code>	descriptor of the request that is to be marked as finished.
<code>o</code>	Pointer to the observer that is notified.
<code>len</code>	Number of bytes written for this request.

**Precondition**

- queue must be in working state.
- d must be a valid request from this queue.

Definition at line 240 of file [virtio](#).

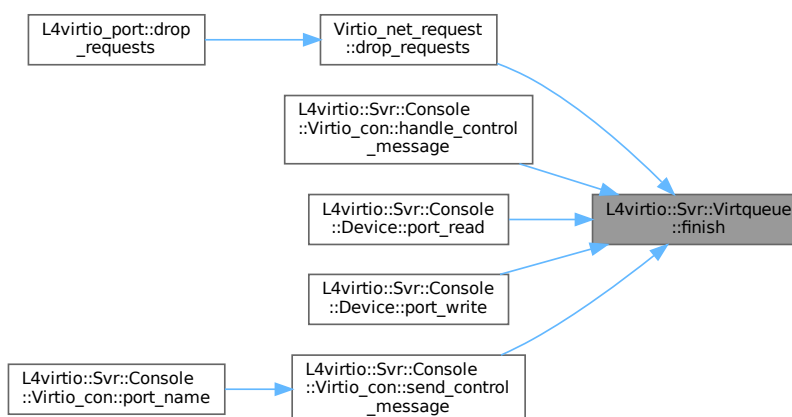
References [consumed\(\)](#).

Referenced by [Virtio\\_net\\_request::drop\\_requests\(\)](#), [L4virtio::Svr::Console::Virtio\\_con::handle\\_control\\_message\(\)](#), [L4virtio::Svr::Console::Device::port\\_read\(\)](#), [L4virtio::Svr::Console::Device::port\\_write\(\)](#), and [L4virtio::Svr::Console::Virtio\\_con::send\\_control\\_message\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**16.438.2.8 finish() [2/2]**

```

template<typename ITER, typename QUEUE_OBSERVER>
void L4virtio::Svr::Virtqueue::finish (
    ITER const & begin,
    ITER const & end,
    QUEUE_OBSERVER * o) [inline]
  
```

Add a range of descriptors to the used ring, and notify an observer once.

The iterators are passed to [consumed<ITER>\(ITER const &, ITER const &\)](#), and the requirements detailed there apply.

**Template Parameters**

<i>ITER</i>	type of the iterator (inferred)
<i>QUEUE_OBSERVER</i>	the type of the observer (inferred).

### Parameters

<i>begin</i>	iterator pointing to first element.
<i>end</i>	iterator pointing to one past last element.
<i>o</i>	pointer to the observer that is notified.

### Precondition

queue must be in working state.

Definition at line 262 of file [virtio](#).

References [consumed\(\)](#).

Here is the call graph for this function:



### 16.438.2.9 next\_avail()

```
Request L4virtio::Svr::Virtqueue::next_avail () [inline]
```

Get the next available descriptor from the available ring.

### Precondition

The queue must be in working state.

### Returns

A Request for the next available descriptor, the Request is invalid if there are no descriptors in the available ring.

**Note**

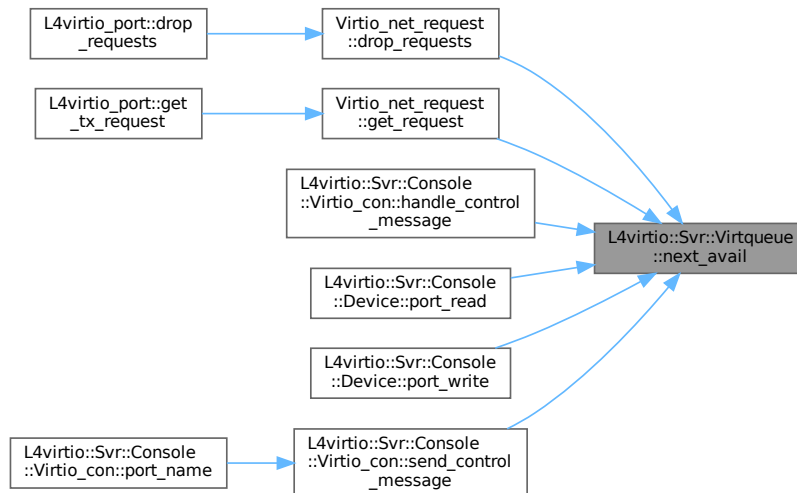
The return value must be checked even when a previous `desc_avail()` returned true.

Definition at line 136 of file `virtio`.

References `L4virtio::Virtqueue::_avail`, `L4virtio::Virtqueue::_current_avail`, `L4virtio::Virtqueue::_idx_mask`, and `L4_LIKELY`.

Referenced by `Virtio_net_request::drop_requests()`, `Virtio_net_request::get_request()`, `L4virtio::Svr::Console::Virtio_con::handle_control_message`, `L4virtio::Svr::Console::Device::port_read()`, `L4virtio::Svr::Console::Device::port_write()`, and `L4virtio::Svr::Console::Virtio_con::send_control_message`.

Here is the caller graph for this function:

**16.438.2.10 rewind\_avail()**

```
void L4virtio::Svr::Virtqueue::rewind_avail (
    Head_desc const & d) [inline]
```

Return unfinished descriptors to the available ring, i.e.

reset the local next index of the available ring to the given descriptor.

**Parameters**

<i>d</i>	descriptor of the request that is to be marked as finished.
----------	-------------------------------------------------------------

**Precondition**

queue must be in working state.

*d* must be a valid request from this queue, obtained via `next_avail()`, that has not yet been finished, and in addition, no descriptors following it have been finished.

Definition at line 160 of file `virtio`.

References `L4virtio::Virtqueue::_current_avail`, `L4virtio::Virtqueue::_desc`, and `L4virtio::Virtqueue::_idx_mask`.

The documentation for this class was generated from the following file:

- `I4/I4virtio/server/virtio`

## 16.439 L4virtio::Svr::Virtqueue::Head\_desc Class Reference

VIRTIO request, essentially a descriptor from the available ring.

```
#include <virtio>
```

Collaboration diagram for L4virtio::Svr::Virtqueue::Head\_desc:

L4virtio::Svr::Virtqueue ::Head_desc	
+	Head_desc()
+	valid()
+	operator bool()
+	desc()

### Public Member Functions

- **Head\_desc** ()  
*Make invalid (NULL) request.*
- bool **valid** () const
- **operator bool** () const
- **Desc** const \* **desc** () const

### 16.439.1 Detailed Description

VIRTIO request, essentially a descriptor from the available ring.

Definition at line 93 of file [virtio](#).

### 16.439.2 Member Function Documentation

#### 16.439.2.1 desc()

```
Desc const * L4virtio::Svr::Virtqueue::Head_desc::desc () const [inline]
```



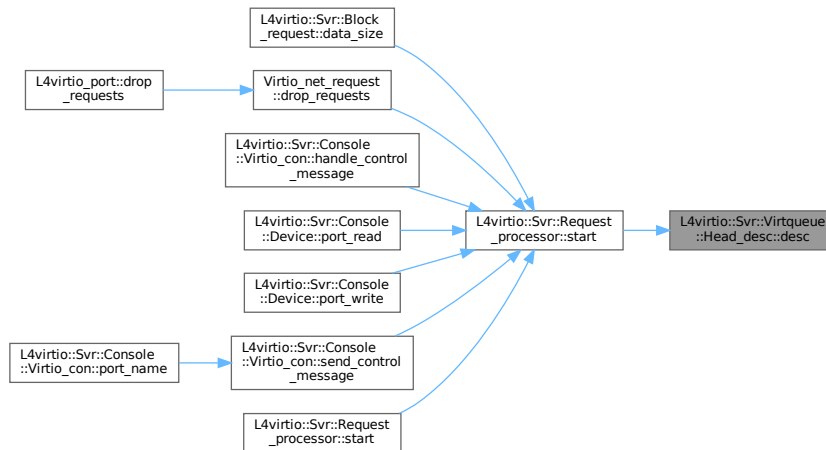
**Returns**

Pointer to the head descriptor of the request.

Definition at line 112 of file [virtio](#).

Referenced by [L4virtio::Svr::Request\\_processor::start\(\)](#).

Here is the caller graph for this function:

**16.439.2.2 operator bool()**

```
L4virtio::Svr::Virtqueue::Head_desc::operator bool () const [inline], [explicit]
```

**Returns**

True if the request is valid (not NULL).

Definition at line 108 of file [virtio](#).

References [valid\(\)](#).

Here is the call graph for this function:



### 16.439.2.3 valid()

```
bool L4virtio::Svr::Virtqueue::Head_desc::valid () const [inline]
```

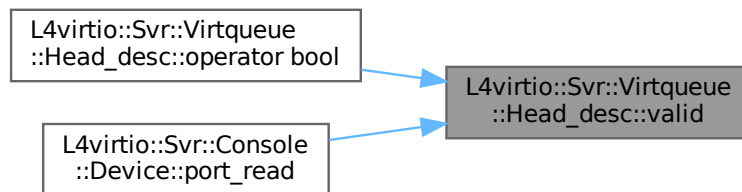
#### Returns

True if the request is valid (not NULL).

Definition at line 105 of file [virtio](#).

Referenced by [operator bool\(\)](#), and [L4virtio::Svr::Console::Device::port\\_read\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

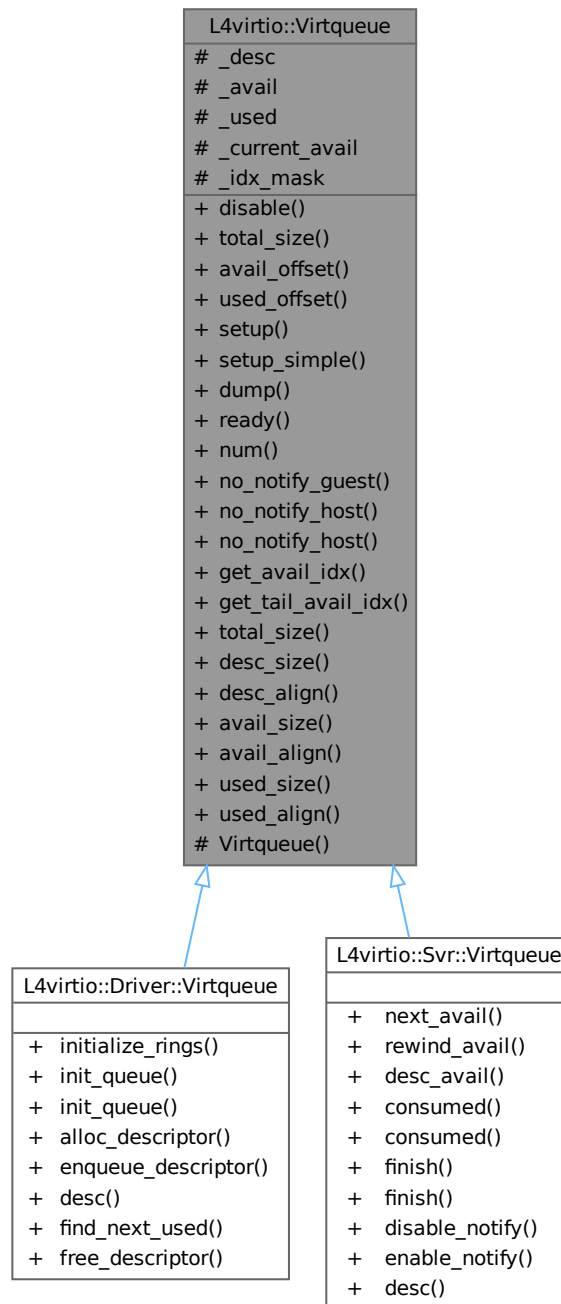
- `I4/I4virtio/server/virtio`

## 16.440 L4virtio::Virtqueue Class Reference

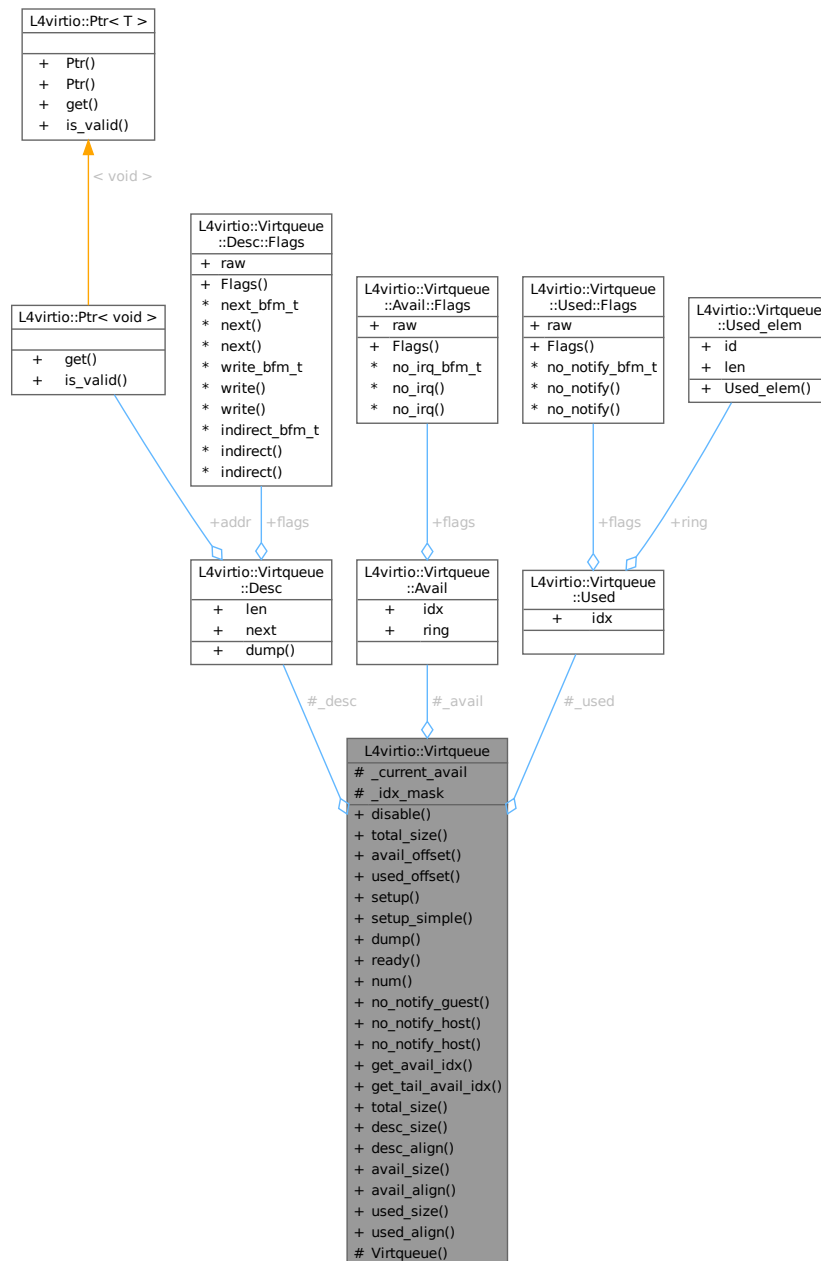
Low-level [Virtqueue](#).

```
#include <virtqueue>
```

Inheritance diagram for L4virtio::Virtqueue:



Collaboration diagram for L4virtio::Virtqueue:



## Data Structures

- class [Desc](#)  
*Descriptor in the descriptor table.*
- class [Avail](#)  
*Type of available ring, this is read-only for the host.*
- struct [Used\\_elem](#)  
*Type of an element of the used ring.*
- class [Used](#)  
*Used ring.*

## Public Types

- enum  
*Fixed alignment values for different parts of a virtqueue.*

## Public Member Functions

- void `disable` ()  
*Completely disable the queue.*
- unsigned long `total_size` () const  
*Calculate the total size of this virtqueue.*
- unsigned long `avail_offset` () const  
*Get the offset of the available ring from the descriptor table.*
- unsigned long `used_offset` () const  
*Get the offset of the used ring from the descriptor table.*
- void `setup` (unsigned `num`, void \*`desc`, void \*`avail`, void \*`used`)  
*Enable this queue.*
- void `setup_simple` (unsigned `num`, void \*`ring`)  
*Enable this queue.*
- void `dump` (Desc const \*`d`) const  
*Dump descriptors for this queue.*
- bool `ready` () const  
*Test if this queue is in working state.*
- unsigned `num` () const
- bool `no_notify_guest` () const  
*Get the no IRQ flag of this queue.*
- bool `no_notify_host` () const  
*Get the no notify flag of this queue.*
- void `no_notify_host` (bool `value`)  
*Set the no-notify flag for this queue.*
- `l4_uint16_t` `get_avail_idx` () const  
*Get available index from available ring (for debugging).*
- `l4_uint16_t` `get_tail_avail_idx` () const  
*Get tail-available index stored in local state (for debugging).*

## Static Public Member Functions

- static unsigned long `total_size` (unsigned `num`)  
*Calculate the total size for a virtqueue of the given dimensions.*
- static unsigned long `desc_size` (unsigned `num`)  
*Calculate the size of the descriptor table for `num` entries.*
- static unsigned long `desc_align` ()  
*Get the alignment in zero LSBs needed for the descriptor table.*
- static unsigned long `avail_size` (unsigned `num`)  
*Calculate the size of the available ring for `num` entries.*
- static unsigned long `avail_align` ()  
*Get the alignment in zero LSBs needed for the available ring.*
- static unsigned long `used_size` (unsigned `num`)  
*Calculate the size of the used ring for `num` entries.*
- static unsigned long `used_align` ()  
*Get the alignment in zero LSBs needed for the used ring.*

## Protected Member Functions

- **Virtqueue** ()=default  
*Create a disabled virtqueue.*

## Protected Attributes

- **Desc** \* **\_desc** = nullptr  
*pointer to descriptor table, NULL if queue is off.*
- **Avail** \* **\_avail** = nullptr  
*pointer to available ring.*
- **Used** \* **\_used** = nullptr  
*pointer to used ring.*
- **l4\_uint16\_t** **\_current\_avail** = 0  
*The life counter for the queue.*
- **l4\_uint16\_t** **\_idx\_mask** = 0  
*mask used for indexing into the descriptor table and the rings.*

### 16.440.1 Detailed Description

Low-level [Virtqueue](#).

This class represents a single virtqueue, with a local running available index.

#### Note

The [Virtqueue](#) implementation is not thread-safe.

Definition at line 80 of file [virtqueue](#).

### 16.440.2 Member Function Documentation

#### 16.440.2.1 [avail\\_align\(\)](#)

```
unsigned long L4virtio::Virtqueue::avail_align () [inline], [static]
```

Get the alignment in zero LSBs needed for the available ring.

#### Returns

The alignment in zero LSBs needed for an available ring.

Definition at line 287 of file [virtqueue](#).

#### 16.440.2.2 [avail\\_size\(\)](#)

```
unsigned long L4virtio::Virtqueue::avail_size (
    unsigned num) [inline], [static]
```

Calculate the size of the available ring for [num](#) entries.

#### Parameters

---

<i>num</i>	The number of entries in the available ring.
------------	----------------------------------------------

### Returns

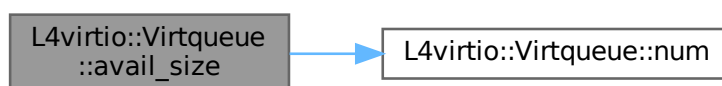
The size in bytes needed for an available ring with *num* entries.

Definition at line 279 of file [virtqueue](#).

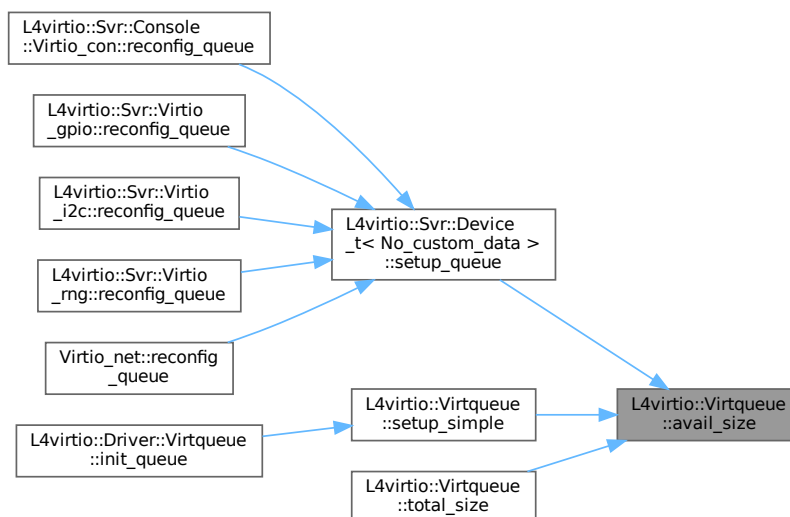
References [num\(\)](#).

Referenced by [L4virtio::Svr::Device\\_t< No\\_custom\\_data >::setup\\_queue\(\)](#), [setup\\_simple\(\)](#), and [total\\_size\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.440.2.3 desc\_align()

```
unsigned long L4virtio::Virtqueue::desc_align () [inline], [static]
```

Get the alignment in zero LSBs needed for the descriptor table.

#### Returns

The alignment in zero LSBs needed for a descriptor table.

Definition at line 269 of file [virtqueue](#).

### 16.440.2.4 desc\_size()

```
unsigned long L4virtio::Virtqueue::desc_size (  
    unsigned num) [inline], [static]
```

Calculate the size of the descriptor table for [num](#) entries.

#### Parameters

<i>num</i>	The number of entries in the descriptor table.
------------	------------------------------------------------

#### Returns

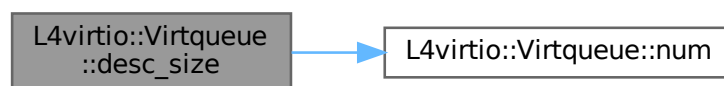
The size in bytes needed for a descriptor table with [num](#) entries.

Definition at line 261 of file [virtqueue](#).

References [num\(\)](#).

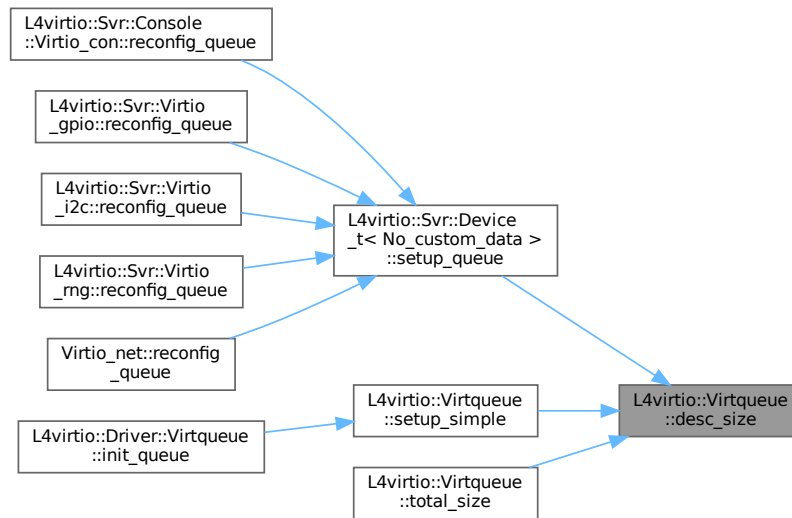
Referenced by [L4virtio::Svr::Device\\_t< No\\_custom\\_data >::setup\\_queue\(\)](#), [setup\\_simple\(\)](#), and [total\\_size\(\)](#).

Here is the call graph for this function:





Here is the caller graph for this function:



#### 16.440.2.5 disable()

```
void L4virtio::Virtqueue::disable () [inline]
```

Completely disable the queue.

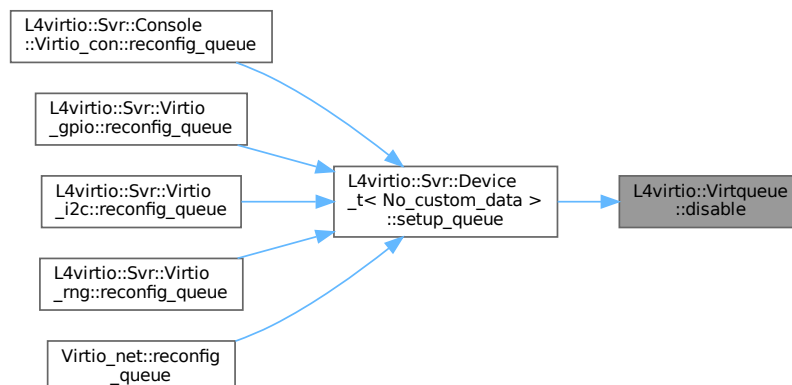
[setup\(\)](#) must be used to enable the queue again.

Definition at line 224 of file [virtqueue](#).

References [\\_desc](#).

Referenced by [L4virtio::Svr::Device\\_t< No\\_custom\\_data >::setup\\_queue\(\)](#).

Here is the caller graph for this function:



### 16.440.2.6 dump()

```
void L4virtio::Virtqueue::dump (
    Desc const * d) const [inline]
```

Dump descriptors for this queue.

#### Precondition

the queue must be in working state.

Definition at line 392 of file [virtqueue](#).

References [\\_desc](#), and [L4virtio::Virtqueue::Desc::dump\(\)](#).

Here is the call graph for this function:



### 16.440.2.7 get\_avail\_idx()

```
l4_uint16_t L4virtio::Virtqueue::get_avail_idx () const [inline]
```

Get available index from available ring (for debugging).

#### Precondition

Queue must be in a working state.

#### Returns

current index in the available ring (shared between device model and device driver).

Definition at line 449 of file [virtqueue](#).

References [\\_avail](#).

### 16.440.2.8 get\_tail\_avail\_idx()

```
l4_uint16_t L4virtio::Virtqueue::get_tail_avail_idx () const [inline]
```

Get tail-available index stored in local state (for debugging).

#### Returns

current tail index for the the available ring.

Definition at line 456 of file [virtqueue](#).

References [\\_current\\_avail](#).

### 16.440.2.9 no\_notify\_guest()

```
bool L4virtio::Virtqueue::no_notify_guest () const [inline]
```

Get the no IRQ flag of this queue.

#### Precondition

queue must be in working state.

#### Returns

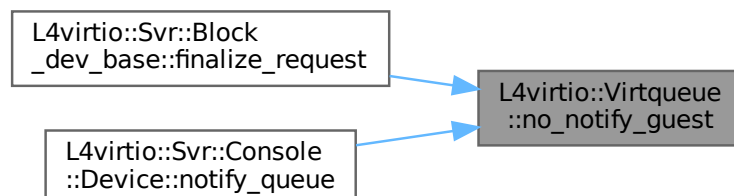
true if the guest does not want to get IRQs (currently).

Definition at line 414 of file [virtqueue](#).

References [\\_avail](#).

Referenced by [L4virtio::Svr::Block\\_dev\\_base<Ds\\_data>::finalize\\_request\(\)](#), and [L4virtio::Svr::Console::Device::notify\\_queue\(\)](#).

Here is the caller graph for this function:



**16.440.2.10 no\_notify\_host()** [1/2]

```
bool L4virtio::Virtqueue::no_notify_host () const [inline]
```

Get the no notify flag of this queue.

**Precondition**

queue must be in working state.

**Returns**

true if the host does not want to get IRQs (currently).

Definition at line 426 of file [virtqueue](#).

References [\\_used](#).

**16.440.2.11 no\_notify\_host()** [2/2]

```
void L4virtio::Virtqueue::no_notify_host (  
    bool value) [inline]
```

Set the no-notify flag for this queue.

**Precondition**

Queue must be in a working state.

Definition at line 436 of file [virtqueue](#).

References [\\_used](#).

**16.440.2.12 num()**

```
unsigned L4virtio::Virtqueue::num () const [inline]
```

## Returns

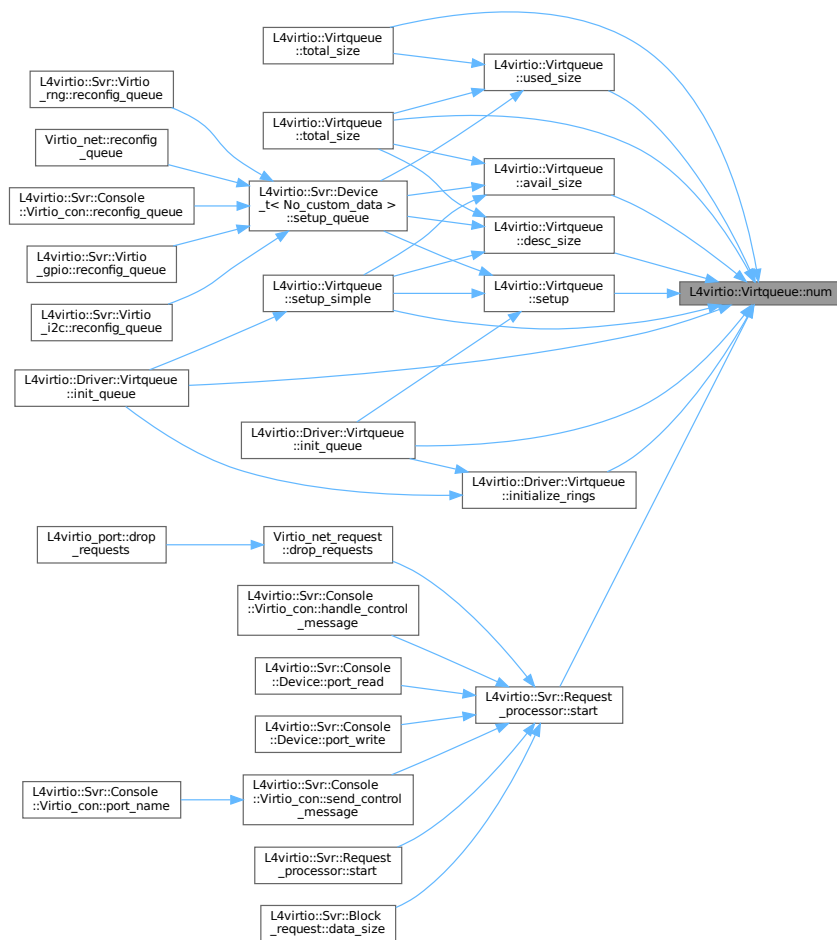
The number of entries in the ring.

Definition at line 404 of file [virtqueue](#).

References [\\_idx\\_mask](#).

Referenced by [avail\\_size\(\)](#), [desc\\_size\(\)](#), [L4virtio::Driver::Virtqueue::init\\_queue\(\)](#), [L4virtio::Driver::Virtqueue::init\\_queue\(\)](#), [L4virtio::Driver::Virtqueue::initialize\\_rings\(\)](#), [setup\(\)](#), [setup\\_simple\(\)](#), [L4virtio::Svr::Request\\_processor::start\(\)](#), [total\\_size\(\)](#), [total\\_size\(\)](#), and [used\\_size\(\)](#).

Here is the caller graph for this function:



## 16.440.2.13 ready()

```
bool L4virtio::Virtqueue::ready () const [inline]
```

Test if this queue is in working state.

**Returns**

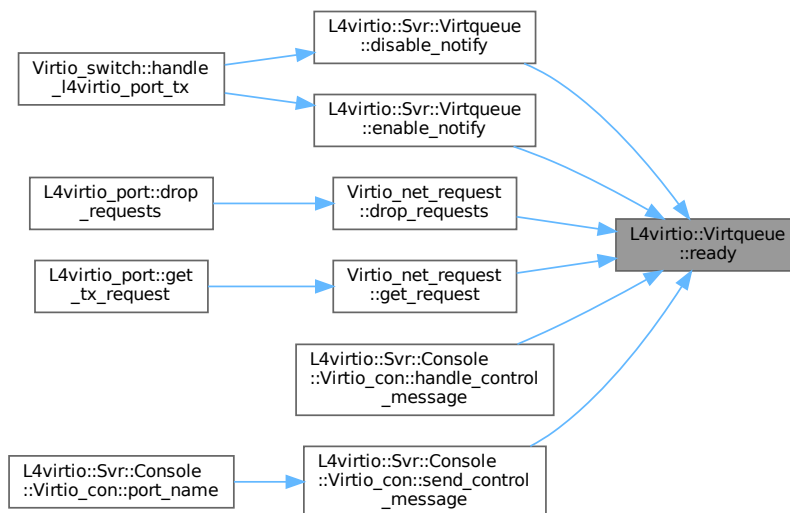
true when the queue is in working state, false else.

Definition at line 400 of file [virtqueue](#).

References [\\_desc](#), and [L4\\_LIKELY](#).

Referenced by [L4virtio::Svr::Virtqueue::disable\\_notify\(\)](#), [Virtio\\_net\\_request::drop\\_requests\(\)](#), [L4virtio::Svr::Virtqueue::enable\\_notify\(\)](#), [Virtio\\_net\\_request::get\\_request\(\)](#), [L4virtio::Svr::Console::Virtio\\_con::handle\\_control\\_message\(\)](#), and [L4virtio::Svr::Console::Virtio\\_con::port\\_name\(\)](#).

Here is the caller graph for this function:

**16.440.2.14 setup()**

```

void L4virtio::Virtqueue::setup (
    unsigned num,
    void * desc,
    void * avail,
    void * used) [inline]

```

Enable this queue.

**Parameters**

<i>num</i>	The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2).
<i>desc</i>	The address of the descriptor table. (Must be Desc_align aligned and at least desc_size(num) bytes in size.)
<i>avail</i>	The address of the available ring. (Must be Avail_align aligned and at least avail_size(num) bytes in size.)

<i>used</i>	The address of the used ring. (Must be <code>Used_align</code> aligned and at least <code>used_size(num)</code> bytes in size.)
-------------	---------------------------------------------------------------------------------------------------------------------------------

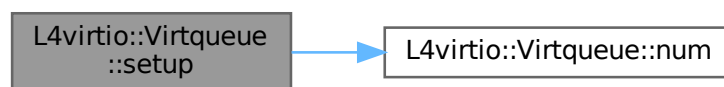
Due to the data type of the descriptors, the queue can have a maximum size of  $2^{16}$ .

Definition at line 349 of file [virtqueue](#).

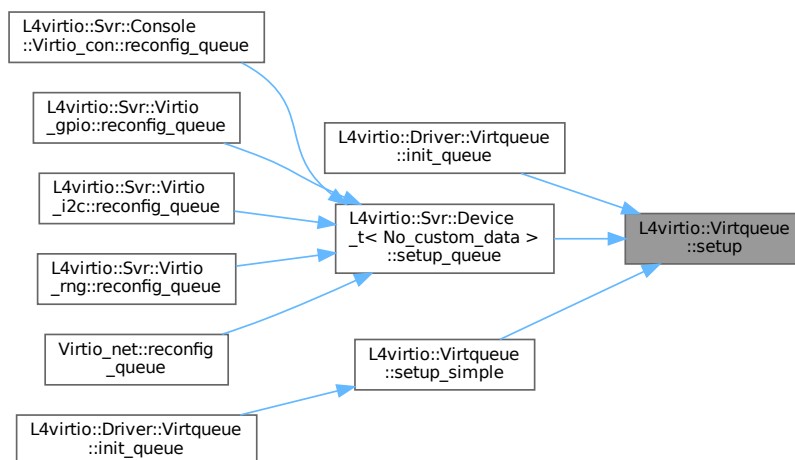
References [\\_avail](#), [\\_current\\_avail](#), [\\_desc](#), [\\_idx\\_mask](#), [\\_used](#), [L4\\_EINVAL](#), and [num\(\)](#).

Referenced by [L4virtio::Driver::Virtqueue::init\\_queue\(\)](#), [L4virtio::Svr::Device\\_t< No\\_custom\\_data >::setup\\_queue\(\)](#), and [setup\\_simple\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.440.2.15 setup\_simple()

```

void L4virtio::Virtqueue::setup_simple (
    unsigned num,
    void * ring) [inline]
  
```

Enable this queue.

#### Parameters

<i>num</i>	The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2).
<i>ring</i>	The base address for the queue data structure. The memory block at <code>ring</code> must be at least <code>total_size(num)</code> bytes in size and have an alignment of <code>Desc_align(desc_align())</code> bits.

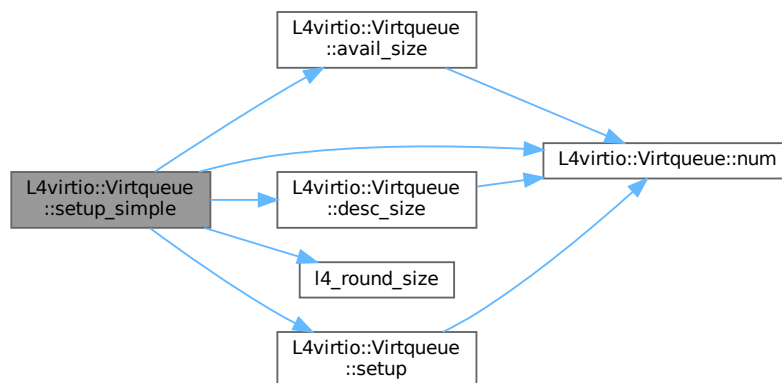
Due to the data type of the descriptors, the queue can have a maximum size of  $2^{16}$ .

Definition at line 378 of file [virtqueue](#).

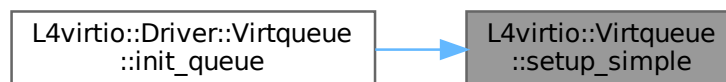
References [avail\\_size\(\)](#), [desc\\_size\(\)](#), [l4\\_round\\_size\(\)](#), [num\(\)](#), and [setup\(\)](#).

Referenced by [L4virtio::Driver::Virtqueue::init\\_queue\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 16.440.2.16 total\_size() [1/2]

```
unsigned long L4virtio::Virtqueue::total_size () const [inline]
```

Calculate the total size of this virtqueue.



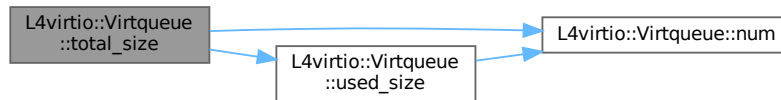
**Precondition**

The queue has been set up.

Definition at line 314 of file [virtqueue](#).

References [\\_desc](#), [\\_used](#), [num\(\)](#), and [used\\_size\(\)](#).

Here is the call graph for this function:

**16.440.2.17 total\_size() [2/2]**

```

unsigned long L4virtio::Virtqueue::total_size (
    unsigned num) [inline], [static]
  
```

Calculate the total size for a virtqueue of the given dimensions.

**Parameters**

<i>num</i>	The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2).
------------	--------------------------------------------------------------------------------------------------------------

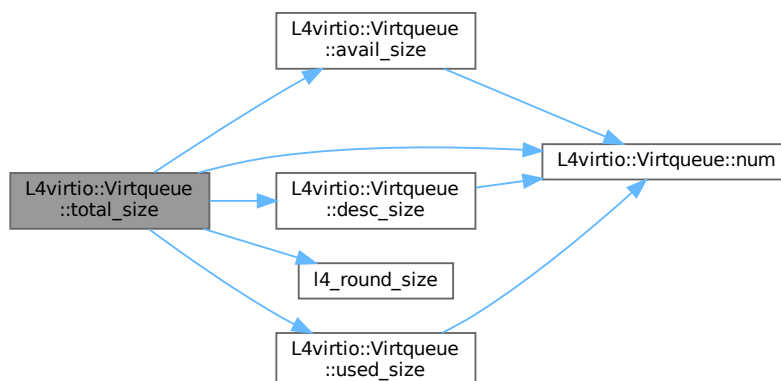
**Returns**

The total size in bytes of the queue data structures.

Definition at line 245 of file [virtqueue](#).

References [avail\\_size\(\)](#), [desc\\_size\(\)](#), [l4\\_round\\_size\(\)](#), [num\(\)](#), and [used\\_size\(\)](#).

Here is the call graph for this function:



#### 16.440.2.18 `used_align()`

```
unsigned long L4virtio::Virtqueue::used_align () [inline], [static]
```

Get the alignment in zero LSBs needed for the used ring.

##### Returns

The alignment in zero LSBs needed for an used ring.

Definition at line 306 of file [virtqueue](#).

#### 16.440.2.19 `used_size()`

```
unsigned long L4virtio::Virtqueue::used_size (  
    unsigned num) [inline], [static]
```

Calculate the size of the used ring for [num](#) entries.

##### Parameters

<i>num</i>	The number of entries in the used ring.
------------	-----------------------------------------

##### Returns

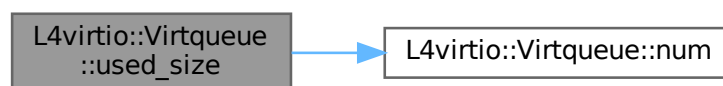
The size in bytes needed for an used ring with [num](#) entries.

Definition at line 298 of file [virtqueue](#).

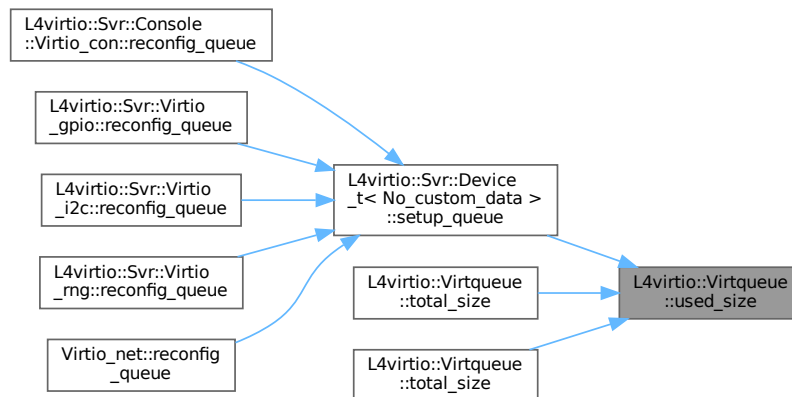
References [num\(\)](#).

Referenced by [L4virtio::Svr::Device\\_t< No\\_custom\\_data >::setup\\_queue\(\)](#), [total\\_size\(\)](#), and [total\\_size\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

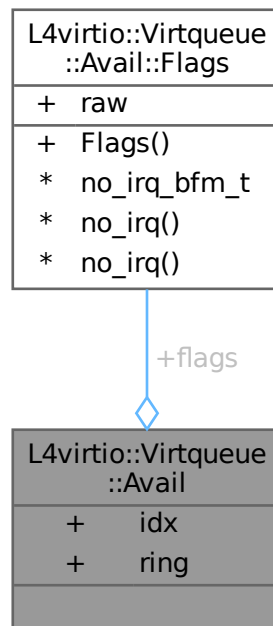
- `I4/I4virtio/virtqueue`

## 16.441 L4virtio::Virtqueue::Avail Class Reference

Type of available ring, this is read-only for the host.

```
#include <virtqueue>
```

Collaboration diagram for L4virtio::Virtqueue::Avail:



## Data Structures

- struct [Flags](#)  
*Flags of the available ring.*

## Data Fields

- [Flags](#) **flags**  
*flags of available ring*
- [l4\\_uint16\\_t](#) **idx**  
*available index written by guest*
- [l4\\_uint16\\_t](#) **ring** []  
*array of available descriptor indexes.*

## 16.441.1 Detailed Description

Type of available ring, this is read-only for the host.

Definition at line [128](#) of file [virtqueue](#).

The documentation for this class was generated from the following file:

- [l4/l4virtio/virtqueue](#)

## 16.442 L4virtio::Virtqueue::Avail::Flags Struct Reference

[Flags](#) of the available ring.

```
#include <virtqueue>
```

Collaboration diagram for L4virtio::Virtqueue::Avail::Flags:

L4virtio::Virtqueue ::Avail::Flags	
+	raw
+	Flags()
*	no_irq_bfm_t
*	no_irq()
*	no_irq()

### Public Member Functions

- **Flags** ([l4\\_uint16\\_t](#) v)  
*Make [Flags](#) from the raw value.*

### Data Fields

- [l4\\_uint16\\_t](#) raw  
*raw 16bit flags value of the available ring.*
- typedef [cxx::Bitfield](#)< decltype(raw), 0, 0 > [no\\_irq\\_bfm\\_t](#)  
*Guest does not want to receive interrupts when requests are finished.*
- constexpr [no\\_irq\\_bfm\\_t::Val](#) no\_irq () const  
*Get the [no\\_irq](#) bits (0 to 0) of [raw](#).*
- constexpr [no\\_irq\\_bfm\\_t::Ref](#) no\_irq ()  
*Get a reference to the [no\\_irq](#) bits (0 to 0) of [raw](#).*

### 16.442.1 Detailed Description

[Flags](#) of the available ring.

Definition at line 134 of file [virtqueue](#).

## 16.442.2 Member Typedef Documentation

### 16.442.2.1 no\_irq\_bfm\_t

```
typedef cxx::Bitfield<decltype(raw), 0, 0> L4virtio::Virtqueue::Avail::Flags::no\_irq\_bfm\_t
```

Guest does not want to receive interrupts when requests are finished.

Type to access the [no\\_irq](#) bits (0 to 0) of [raw](#).

Definition at line [143](#) of file [virtqueue](#).

The documentation for this struct was generated from the following file:

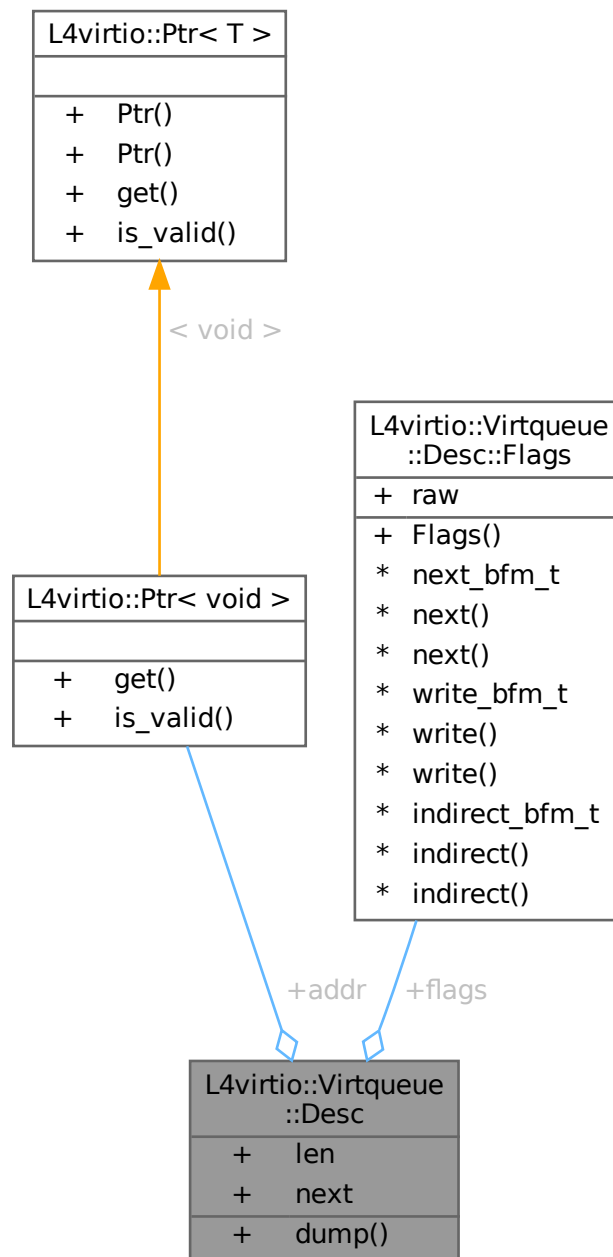
- [l4/l4virtio/virtqueue](#)

## 16.443 L4virtio::Virtqueue::Desc Class Reference

Descriptor in the descriptor table.

```
#include <virtqueue>
```

Collaboration diagram for L4virtio::Virtqueue::Desc:



## Data Structures

- struct [Flags](#)  
*Type for descriptor flags.*

## Public Member Functions

- void **dump** (unsigned idx) const

*Dump a single descriptor.*

## Data Fields

- [Ptr](#)< void > **addr**  
*Address stored in descriptor.*
- [l4\\_uint32\\_t](#) **len**  
*Length of described buffer.*
- [Flags](#) **flags**  
*Descriptor flags.*
- [l4\\_uint16\\_t](#) **next**  
*Index of the next chained descriptor.*

## 16.443.1 Detailed Description

Descriptor in the descriptor table.

Definition at line 86 of file [virtqueue](#).

The documentation for this class was generated from the following file:

- l4/l4virtio/virtqueue

## 16.444 L4virtio::Virtqueue::Desc::Flags Struct Reference

Type for descriptor flags.

```
#include <virtqueue>
```

Collaboration diagram for L4virtio::Virtqueue::Desc::Flags:

L4virtio::Virtqueue ::Desc::Flags
+ raw
+ Flags()
* next_bfm_t
* next()
* next()
* write_bfm_t
* write()
* write()
* indirect_bfm_t
* indirect()
* indirect()



## Public Member Functions

- **Flags** ([l4\\_uint16\\_t](#) v)  
*Make [Flags](#) from raw 16bit value.*

## Data Fields

- [l4\\_uint16\\_t](#) **raw**  
*raw flags value of a virtio descriptor.*
- typedef [cxx::Bitfield](#)< decltype([raw](#)), 0, 0 > [next\\_bfm\\_t](#)  
*Part of a descriptor chain which is continued with the next field.*
- constexpr [next\\_bfm\\_t::Val](#) **next** () const  
*Get the [next](#) bits (0 to 0) of [raw](#).*
- constexpr [next\\_bfm\\_t::Ref](#) **next** ()  
*Get a reference to the [next](#) bits (0 to 0) of [raw](#).*
- typedef [cxx::Bitfield](#)< decltype([raw](#)), 1, 1 > [write\\_bfm\\_t](#)  
*Block described by this descriptor is writeable.*
- constexpr [write\\_bfm\\_t::Val](#) **write** () const  
*Get the [write](#) bits (1 to 1) of [raw](#).*
- constexpr [write\\_bfm\\_t::Ref](#) **write** ()  
*Get a reference to the [write](#) bits (1 to 1) of [raw](#).*
- typedef [cxx::Bitfield](#)< decltype([raw](#)), 2, 2 > [indirect\\_bfm\\_t](#)  
*Indirect descriptor, block contains a list of descriptors.*
- constexpr [indirect\\_bfm\\_t::Val](#) **indirect** () const  
*Get the [indirect](#) bits (2 to 2) of [raw](#).*
- constexpr [indirect\\_bfm\\_t::Ref](#) **indirect** ()  
*Get a reference to the [indirect](#) bits (2 to 2) of [raw](#).*

## 16.444.1 Detailed Description

Type for descriptor flags.

Definition at line 92 of file [virtqueue](#).

## 16.444.2 Member Typedef Documentation

### 16.444.2.1 indirect\_bfm\_t

```
typedef cxx::Bitfield<decltype(raw), 2, 2> L4virtio::Virtqueue::Desc::Flags::indirect_bfm_t
```

Indirect descriptor, block contains a list of descriptors.

Type to access the [indirect](#) bits (2 to 2) of [raw](#).

Definition at line 105 of file [virtqueue](#).

### 16.444.2.2 next\_bfm\_t

```
typedef cxx::Bitfield<decltype(raw), 0, 0> L4virtio::Virtqueue::Desc::Flags::next_bfm_t
```

Part of a descriptor chain which is continued with the next field.

Type to access the `next` bits (0 to 0) of `raw`.

Definition at line 101 of file `virtqueue`.

### 16.444.2.3 write\_bfm\_t

```
typedef cxx::Bitfield<decltype(raw), 1, 1> L4virtio::Virtqueue::Desc::Flags::write_bfm_t
```

Block described by this descriptor is writeable.

Type to access the `write` bits (1 to 1) of `raw`.

Definition at line 103 of file `virtqueue`.

The documentation for this struct was generated from the following file:

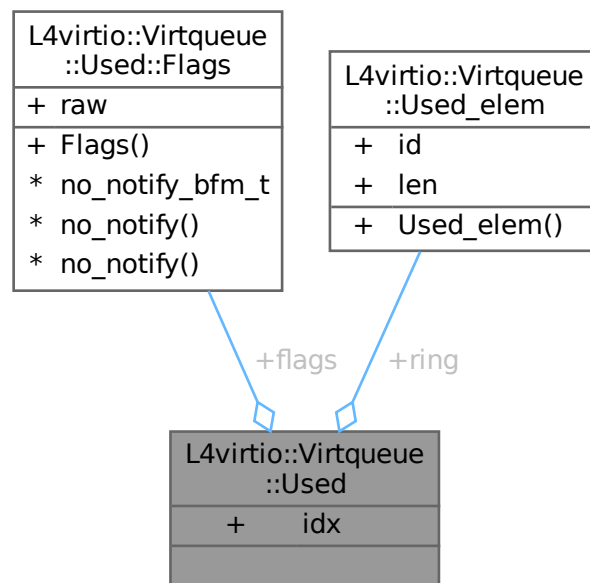
- `l4/l4virtio/virtqueue`

## 16.445 L4virtio::Virtqueue::Used Class Reference

`Used` ring.

```
#include <virtqueue>
```

Collaboration diagram for `L4virtio::Virtqueue::Used`:



## Data Structures

- struct [Flags](#)  
*flags for the used ring.*

## Data Fields

- [Flags](#) **flags**  
*flags of the used ring.*
- [l4\\_uint16\\_t](#) **idx**  
*index of the last entry in the ring.*
- [Used\\_elem](#) **ring** []  
*array of used descriptors.*

### 16.445.1 Detailed Description

[Used](#) ring.

Definition at line 173 of file [virtqueue](#).

The documentation for this class was generated from the following file:

- l4/l4virtio/virtqueue

## 16.446 L4virtio::Virtqueue::Used::Flags Struct Reference

flags for the used ring.

```
#include <virtqueue>
```

Collaboration diagram for L4virtio::Virtqueue::Used::Flags:

L4virtio::Virtqueue ::Used::Flags
+ raw
+ Flags()
* no_notify_bfm_t
* no_notify()
* no_notify()

## Public Member Functions

- **Flags** ([l4\\_uint16\\_t](#) v)  
*make [Flags](#) from raw value*

## Data Fields

- [l4\\_uint16\\_t](#) **raw**  
*raw flags value as specified by virtio.*
- typedef [cxx::Bitfield](#)< decltype([raw](#)), 0, 0 > **no\_notify\_bfm\_t**  
*host does not want to be notified when new requests have been queued.*
- constexpr [no\\_notify\\_bfm\\_t::Val](#) **no\_notify** () const  
*Get the [no\\_notify](#) bits (0 to 0) of [raw](#).*
- constexpr [no\\_notify\\_bfm\\_t::Ref](#) **no\_notify** ()  
*Get a reference to the [no\\_notify](#) bits (0 to 0) of [raw](#).*

### 16.446.1 Detailed Description

flags for the used ring.

Definition at line 179 of file [virtqueue](#).

### 16.446.2 Member Typedef Documentation

#### 16.446.2.1 no\_notify\_bfm\_t

```
typedef cxx::Bitfield<decltype(raw), 0, 0> L4virtio::Virtqueue::Used::Flags::no\_notify\_bfm\_t
```

host does not want to be notified when new requests have been queued.

Type to access the [no\\_notify](#) bits (0 to 0) of [raw](#).

Definition at line 188 of file [virtqueue](#).

The documentation for this struct was generated from the following file:

- [l4/l4virtio/virtqueue](#)

## 16.447 L4virtio::Virtqueue::Used\_elem Struct Reference

Type of an element of the used ring.

```
#include <virtqueue>
```

Collaboration diagram for L4virtio::Virtqueue::Used\_elem:

L4virtio::Virtqueue ::Used_elem
+ id
+ len
+ Used_elem()

### Public Member Functions

- [Used\\_elem](#) ([l4\\_uint16\\_t id](#), [l4\\_uint32\\_t len](#))  
*Initialize a used ring element.*

### Data Fields

- [l4\\_uint32\\_t id](#)  
*descriptor index*
- [l4\\_uint32\\_t len](#)  
*length field*

### 16.447.1 Detailed Description

Type of an element of the used ring.

Definition at line [154](#) of file [virtqueue](#).

### 16.447.2 Constructor & Destructor Documentation

#### 16.447.2.1 Used\_elem()

```
L4virtio::Virtqueue::Used_elem::Used_elem (
    l4_uint16_t id,
    l4_uint32_t len) [inline]
```

Initialize a used ring element.

#### Parameters

---

<i>id</i>	The index of the descriptor to be marked as used.
<i>len</i>	The total bytes written into the buffer of the descriptor chain.

Definition at line 165 of file [virtqueue](#).

References [id](#), and [len](#).

The documentation for this struct was generated from the following file:

- [l4/l4virtio/virtqueue](#)

## 16.448 l4virtio\_block\_config\_t Struct Reference

Device configuration for block devices.

```
#include <virtio_block.h>
```

Collaboration diagram for l4virtio\_block\_config\_t:

l4virtio_block_config_t	
+	capacity
+	size_max
+	seg_max
+	blk_size

### Data Fields

- [l4\\_uint64\\_t](#) **capacity**  
*Capacity of device in 512-byte sectors.*
- [l4\\_uint32\\_t](#) **size\_max**  
*Maximum size of a single segment.*
- [l4\\_uint32\\_t](#) **seg\_max**  
*Maximum number of segments per request.*
- [l4\\_uint32\\_t](#) **blk\_size**  
*Block size of underlying disk.*

### 16.448.1 Detailed Description

Device configuration for block devices.

Definition at line 68 of file [virtio\\_block.h](#).

The documentation for this struct was generated from the following file:

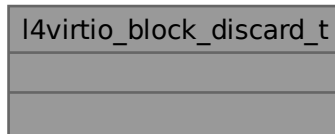
- l4/l4virtio/virtio\_block.h

## 16.449 l4virtio\_block\_discard\_t Struct Reference

Structure used for the write zeroes and discard commands.

```
#include <virtio_block.h>
```

Collaboration diagram for l4virtio\_block\_discard\_t:



### 16.449.1 Detailed Description

Structure used for the write zeroes and discard commands.

Definition at line 58 of file [virtio\\_block.h](#).

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtio\_block.h

## 16.450 l4virtio\_block\_header\_t Struct Reference

Header structure of a request for a block device.

```
#include <virtio_block.h>
```

Collaboration diagram for l4virtio\_block\_header\_t:

l4virtio_block_header_t	
+	type
+	ioprio
+	sector

### Data Fields

- [l4\\_uint32\\_t](#) **type**  
*Kind of request, see [L4virtio\\_block\\_operations](#).*
- [l4\\_uint32\\_t](#) **ioprio**  
*Priority (unused).*
- [l4\\_uint64\\_t](#) **sector**  
*First sector to read/write.*

### 16.450.1 Detailed Description

Header structure of a request for a block device.

Definition at line 42 of file [virtio\\_block.h](#).

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtio\_block.h



## 16.451 l4virtio\_config\_hdr\_t Struct Reference

L4-VIRTIO config header, provided in shared data space.

```
#include <virtio.h>
```

Collaboration diagram for l4virtio\_config\_hdr\_t:

l4virtio_config_hdr_t
+ magic
+ version
+ device
+ vendor
+ dev_features
+ num_queues
+ queues_offset

### Data Fields

- [l4\\_uint32\\_t](#) **magic**  
*magic value (must be 'virt').*
- [l4\\_uint32\\_t](#) **version**  
*VIRTIO version.*
- [l4\\_uint32\\_t](#) **device**  
*device ID*
- [l4\\_uint32\\_t](#) **vendor**  
*vendor ID*
- [l4\\_uint32\\_t](#) **dev\_features**  
*device features windows selected by device\_feature\_sel*
- [l4\\_uint32\\_t](#) **num\_queues**  
*number of virtqueues*
- [l4\\_uint32\\_t](#) **queues\_offset**  
*offset of virtqueue config array*

### 16.451.1 Detailed Description

L4-VIRTIO config header, provided in shared data space.

Definition at line 132 of file [virtio.h](#).

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtio.h

## 16.452 l4virtio\_config\_queue\_t Struct Reference

Queue configuration entry.

```
#include <virtio.h>
```

Collaboration diagram for l4virtio\_config\_queue\_t:

l4virtio_config_queue_t
+ num_max
+ num
+ ready
+ driver_notify_index
+ desc_addr
+ avail_addr
+ used_addr
+ device_notify_index

### Data Fields

- [l4\\_uint16\\_t](#) **num\_max**  
*R: maximum number of descriptors supported by this queue.*
- [l4\\_uint16\\_t](#) **num**  
*RW: number of descriptors configured for this queue.*
- [l4\\_uint16\\_t](#) **ready**  
*RW: queue ready flag (read-write).*
- [l4\\_uint16\\_t](#) **driver\_notify\_index**  
*W: Event index to be used for device notifications (device to driver).*
- [l4\\_uint64\\_t](#) **desc\_addr**  
*W: address of descriptor table.*
- [l4\\_uint64\\_t](#) **avail\_addr**  
*W: address of available ring.*
- [l4\\_uint64\\_t](#) **used\_addr**  
*W: address of used ring.*
- [l4\\_uint16\\_t](#) **device\_notify\_index**  
*R: Event index to be used by the driver (driver to device).*

### 16.452.1 Detailed Description

Queue configuration entry.

An array of such entries is available at the `l4virtio_config_hdr_t::queues_offset` in the config data space.

Consistency rules for the queue config are:

- A driver might read `num_max` at any time.
- A driver must write to `num`, `desc_addr`, `avail_addr`, and `used_addr` only when `ready` is zero (0). Values in these fields are validated and used by the device only after successfully setting `ready` to one (1), either by the IPC or by `L4VIRTIO_CMD_CFG_QUEUE`.
- The value of `device_notify_index` is valid only when `ready` is one.
- The driver might write to `device_notify_index` at any time, however the change is guaranteed to take effect after a successful `L4VIRTIO_CMD_CFG_QUEUE` or after a `config_queue` IPC. Note, the change might also have immediate effect, depending on the device implementation.

Definition at line 223 of file `virtio.h`.

The documentation for this struct was generated from the following file:

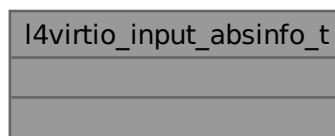
- `l4/l4virtio/virtio.h`

## 16.453 l4virtio\_input\_absinfo\_t Struct Reference

Information about the absolute axis in the underlying evdev implementation.

```
#include <virtio_input.h>
```

Collaboration diagram for `l4virtio_input_absinfo_t`:



### 16.453.1 Detailed Description

Information about the absolute axis in the underlying evdev implementation.

Definition at line 33 of file `virtio_input.h`.

The documentation for this struct was generated from the following file:

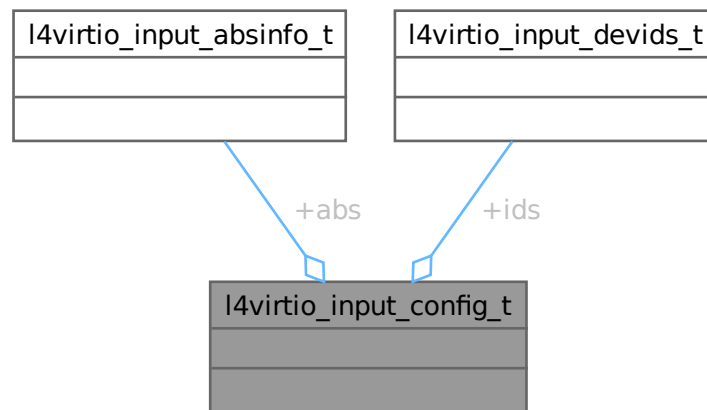
- `l4/l4virtio/virtio_input.h`

## 16.454 l4virtio\_input\_config\_t Struct Reference

Device configuration for input devices.

```
#include <virtio_input.h>
```

Collaboration diagram for l4virtio\_input\_config\_t:



### 16.454.1 Detailed Description

Device configuration for input devices.

Definition at line 56 of file [virtio\\_input.h](#).

The documentation for this struct was generated from the following file:

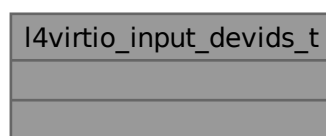
- l4/l4virtio/virtio\_input.h

## 16.455 l4virtio\_input\_devids\_t Struct Reference

Device ID information for the device.

```
#include <virtio_input.h>
```

Collaboration diagram for l4virtio\_input\_devids\_t:



### 16.455.1 Detailed Description

Device ID information for the device.

Definition at line 45 of file [virtio\\_input.h](#).

The documentation for this struct was generated from the following file:

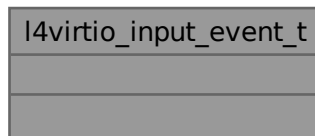
- l4/l4virtio/virtio\_input.h

## 16.456 l4virtio\_input\_event\_t Struct Reference

Single event in event or status queue.

```
#include <virtio_input.h>
```

Collaboration diagram for l4virtio\_input\_event\_t:



### 16.456.1 Detailed Description

Single event in event or status queue.

Definition at line 74 of file [virtio\\_input.h](#).

The documentation for this struct was generated from the following file:

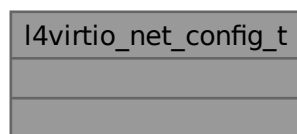
- l4/l4virtio/virtio\_input.h

## 16.457 l4virtio\_net\_config\_t Struct Reference

Device configuration for network devices.

```
#include <virtio_net.h>
```

Collaboration diagram for l4virtio\_net\_config\_t:



### 16.457.1 Detailed Description

Device configuration for network devices.

Definition at line 34 of file [virtio\\_net.h](#).

The documentation for this struct was generated from the following file:

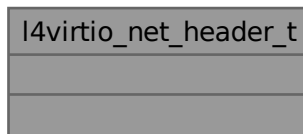
- l4/l4virtio/virtio\_net.h

## 16.458 l4virtio\_net\_header\_t Struct Reference

Header structure of a request for a network device.

```
#include <virtio_net.h>
```

Collaboration diagram for l4virtio\_net\_header\_t:



### 16.458.1 Detailed Description

Header structure of a request for a network device.

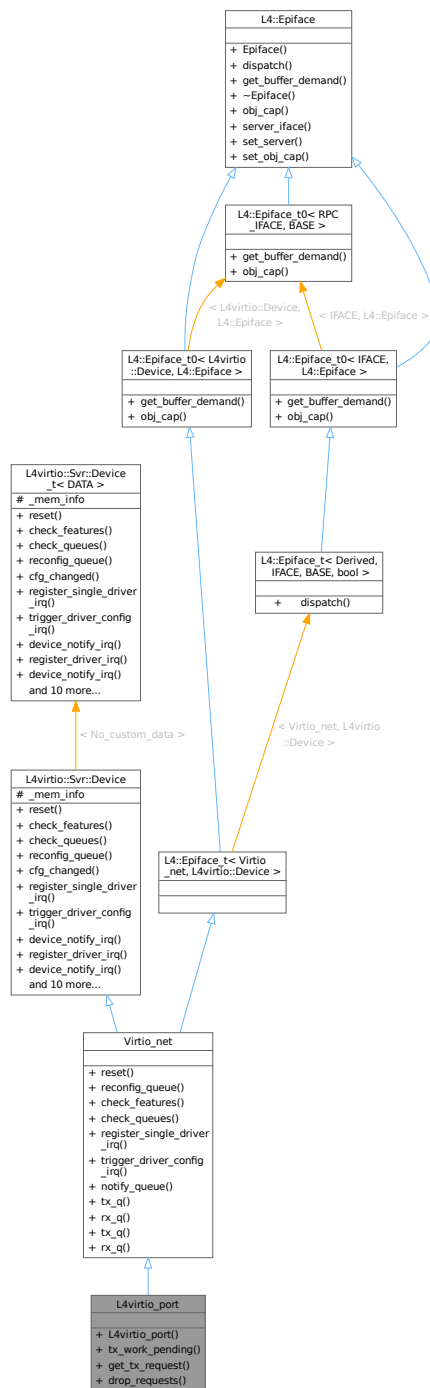
Definition at line 20 of file [virtio\\_net.h](#).

The documentation for this struct was generated from the following file:

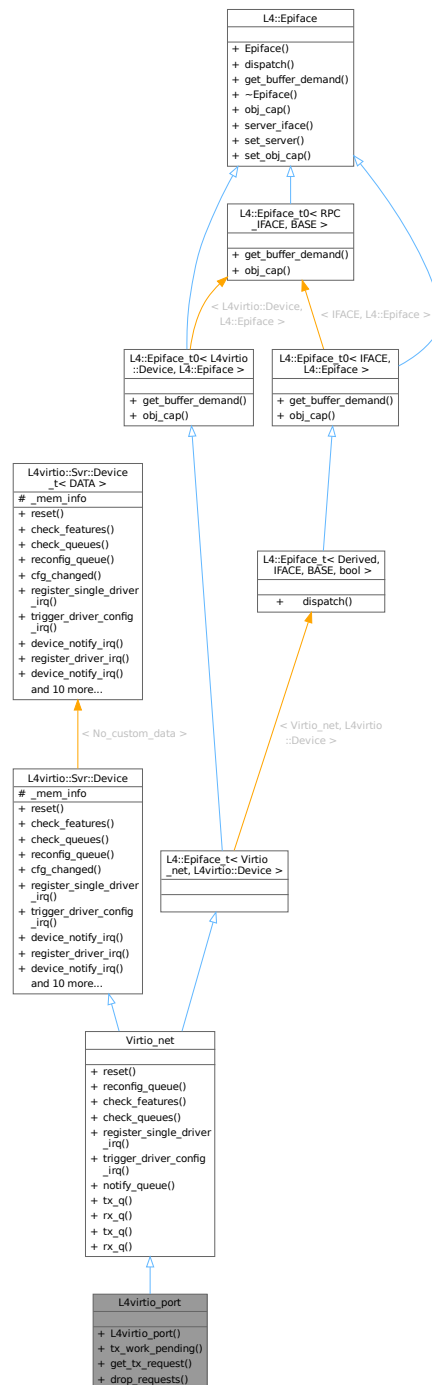
- l4/l4virtio/virtio\_net.h

A Port on the Virtio Net Switch.

Inheritance diagram for L4virtio\_port:



Collaboration diagram for L4virtio\_port:



## Public Member Functions

- **L4virtio\_port** (unsigned vq\_max, unsigned num\_ds, char const \*name, [l4\\_uint8\\_t](#) const \*mac)  
Create a Virtio net port object.
- bool **tx\_work\_pending** () const  
Check whether there is any work pending on the transmission queue.
- std::optional< [Virtio\\_net\\_request](#) > **get\_tx\_request** ()



*Get one request from the transmission queue.*

- void **drop\_requests** ()

*Drop all requests pending in the transmission queue.*

## Public Member Functions inherited from Virtio\_net

- void **reset** () override  
*reset callback, called for doing a device reset*
- int **reconfig\_queue** (unsigned index) override  
*callback for client queue-config request*
- bool **check\_features** () override  
*callback for checking the subset of accepted features*
- bool **check\_queues** () override  
*Check whether both virtqueues are ready.*
- void **register\_single\_driver\_irq** () override  
*Save the \_kick\_guest\_irq that the client sent via device\_notification\_irq().*
- void **trigger\_driver\_config\_irq** () override  
*callback for triggering configuration change notification IRQ*
- void **notify\_queue** (L4virtio::Svr::Virtqueue \*queue)  
*Trigger the \_kick\_guest\_irq IRQ.*
- Virtqueue \* **tx\_q** ()  
*Getter for the transmission queue.*
- Virtqueue \* **rx\_q** ()  
*Getter for the receive queue.*
- Virtqueue const \* **tx\_q** () const  
*Getter for the transmission queue.*
- Virtqueue const \* **rx\_q** () const  
*Getter for the receive queue.*

## Public Member Functions inherited from L4virtio::Svr::Device\_t< No\_custom\_data >

- virtual void **cfg\_changed** (unsigned)  
*callback for client device configuration changes*
- virtual L4::Cap< L4::Irq > **device\_notify\_irq** () const  
*callback to gather the device notification IRQ (old-style)*
- virtual void **register\_driver\_irq** (unsigned idx)  
*Callback for registering an notification IRQ (multi IRQ).*
- virtual L4::Cap< L4::Irq > **device\_notify\_irq** (unsigned idx)  
*Callback to gather the device notification IRQ (multi IRQ).*
- virtual unsigned **num\_events\_supported** () const  
*Return the highest notification index supported.*
- **Device\_t** (Dev\_config \*dev\_config)  
*Make a device for the given config.*
- **Mem\_list** const \* **mem\_info** () const  
*Get the memory region list used for this device.*
- void **reset\_queue\_config** (unsigned idx, unsigned num\_max, bool inc\_generation=false)  
*Trigger reset for the configuration space for queue idx.*
- void **init\_mem\_info** (unsigned num)  
*Initialize the memory region list to the given maximum.*

- void `device_error` ()  
*Transition device into `DEVICE_NEEDS_RESET` state.*
- bool `setup_queue` (`Virtqueue` \*q, unsigned qn, unsigned num\_max)  
*Enable/disable the specified queue.*
- bool `handle_mem_cmd_write` ()  
*Check for a value in the `cmd` register and handle a write.*
- void `enable_trusted_ds_validation` ()  
*Enable trusted dataspace validation.*
- void `add_trusted_dataspaces` (std::shared\_ptr< Ds\_vector const > ds)  
*Provide a list of trusted dataspaces that can be used for validation.*

## Public Member Functions inherited from `L4::Epiface_t0< L4virtio::Device, L4::Epiface >`

- `Type_info::Demand` `get_buffer_demand` () const  
*Get the server-side buffer demand based in IFACE.*
- `Cap< L4virtio::Device >` `obj_cap` () const  
*Get the (typed) capability to this object.*

## Public Member Functions inherited from `L4::Epiface`

- `Epiface` ()  
*Make a server object.*
- virtual `~Epiface` ()=0  
*Destroy the object.*
- Stored\_cap `obj_cap` () const  
*Get the capability to the kernel object belonging to this object.*
- `Server_iface` \* `server_iface` () const  
*Get pointer to server interface at which the object is currently registered.*
- int `set_server` (`Server_iface` \*srv, `Cap`< void > cap, bool managed=false)  
*Set server registration info for the object.*
- void `set_obj_cap` (`Cap`< void > const &cap)  
*Deprecated server registration function.*

## Additional Inherited Members

## Public Types inherited from `L4::Epiface_t0< L4virtio::Device, L4::Epiface >`

- typedef `L4virtio::Device` `Interface`  
*Data type of the IPC interface definition.*

## Public Types inherited from `L4::Epiface`

- typedef `lpc_svr::Server_iface` `Server_iface`  
*Type for abstract server interface.*
- typedef `lpc_svr::Server_iface::Demand` `Demand`  
*Type for server-side receive buffer demand.*

## Protected Attributes inherited from [L4virtio::Svr::Device\\_t< No\\_custom\\_data >](#)

- [Mem\\_list\\_mem\\_info](#)  
*Memory region list.*

### 16.459.1 Detailed Description

A Port on the Virtio Net Switch.

A Port object gets created by `Virtio_factory::op_create()`. This function actually only instantiates objects of the types `Switch_port` and `Monitor_port`. The created Port registers itself at the switch's server. Usually, the IPC call for port creation comes from `ned`. To finalize the setup, the client has to initialize the port during the virtio initialization phase. To do this, the client registers a dataspace for queues and buffers and provides an IRQ to notify the client on incoming network requests.

Definition at line 36 of file [port\\_l4virtio.h](#).

### 16.459.2 Member Function Documentation

#### 16.459.2.1 drop\_requests()

```
void L4virtio_port::drop_requests () [inline]
```

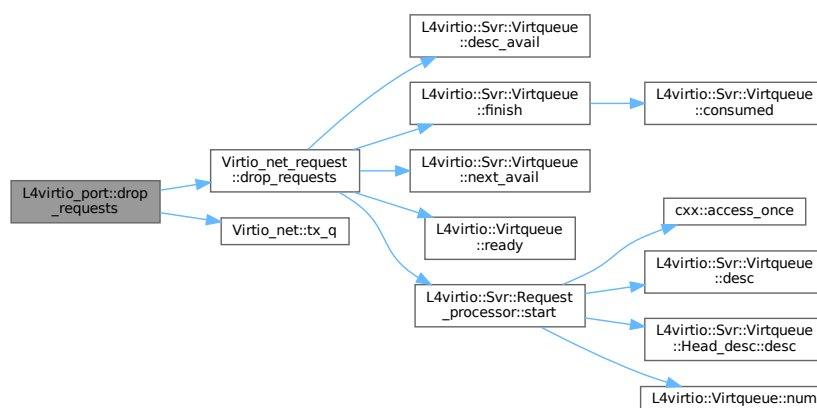
Drop all requests pending in the transmission queue.

This is used for monitor ports, which are not allowed to send packets.

Definition at line 103 of file [port\\_l4virtio.h](#).

References [Virtio\\_net\\_request::drop\\_requests\(\)](#), and [Virtio\\_net::tx\\_q\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

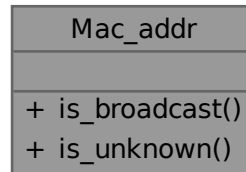
- `pkg/virtio-net-switch/server/switch/port_l4virtio.h`

## 16.460 Mac\_addr Class Reference

A wrapper class around the value of a MAC address.

```
#include <mac_addr.h>
```

Collaboration diagram for Mac\_addr:



### Public Member Functions

- bool **is\_broadcast** () const  
*Check if MAC address is a broadcast or multicast address.*
- bool **is\_unknown** () const  
*Check if the MAC address is not yet known.*

### 16.460.1 Detailed Description

A wrapper class around the value of a MAC address.

Definition at line 19 of file [mac\\_addr.h](#).

The documentation for this class was generated from the following file:

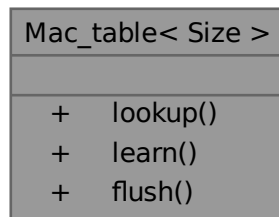
- [pkg/virtio-net-switch/server/switch/mac\\_addr.h](#)

## 16.461 Mac\_table< Size > Class Template Reference

[Mac\\_table](#) manages a 1:n association between ports and MAC addresses.

```
#include <mac_table.h>
```

Collaboration diagram for Mac\_table< Size >:



### Public Member Functions

- Port\_iface \* [lookup](#) (Mac\_addr dst, l4\_uint16\_t vlan\_id) const  
*Find the destination port for a MAC address and VLAN id.*
- void [learn](#) (Mac\_addr src, Port\_iface \*port, l4\_uint16\_t vlan\_id)  
*Learn a MAC address (add it to the MAC table).*
- void [flush](#) (Port\_iface \*port)  
*Flush all associations with a given port.*

### 16.461.1 Detailed Description

```
template<std::size_t Size = 1024U>
class Mac_table< Size >
```

[Mac\\_table](#) manages a 1:n association between ports and MAC addresses.

There are different types of devices which might be attached to a port. For a normal device the switch sees exactly one MAC address per port - the MAC address of the device attached to it. But there might be other devices like software bridges attached to the port sending packets with different MAC addresses to the port. Therefore the switch has to manage a 1:n association between ports and MAC addresses. The MAC table manages this association.

When a packet comes in we need to find the destination port for the packet and therefore perform a lookup based on the MAC address.

To prevent unbounded growth of the lookup table, the number of entries is limited. Replacement is done on a round-robin basis. If the capacity was reached, the oldest entry is evicted.

Definition at line 40 of file [mac\\_table.h](#).

### 16.461.2 Member Function Documentation

#### 16.461.2.1 flush()

```
template<std::size_t Size = 1024U>
void Mac_table< Size >::flush (
    Port_iface * port) [inline]
```

Flush all associations with a given port.

#### Parameters

<i>port</i>	Pointer to port that is to be flushed
-------------	---------------------------------------

This function removes all references to a given port from the MAC table. Since we manage a 1:n association between ports and MAC addresses there might be more than one entry for a given port and we have to iterate over the whole array to delete every reference to the port.

Definition at line 129 of file [mac\\_table.h](#).

### 16.461.2.2 learn()

```
template<std::size_t Size = 1024U>
void Mac_table< Size >::learn (
    Mac_addr src,
    Port_iface * port,
    14_uint16_t vlan_id) [inline]
```

Learn a MAC address (add it to the MAC table).

#### Parameters

<i>src</i>	MAC address
<i>port</i>	Pointer to the port object that can be used to reach MAC address src
<i>vlan↔ _id</i>	VLAN id of the packet destination.

Will evict the oldest learned address from the table if the maximum capacity was reached and if the MAC address was not known yet. The source port of the table entry is always updated to cope with clients that move between ports.

Definition at line 78 of file [mac\\_table.h](#).

References [L4\\_UNLIKELY](#), and [lookup\(\)](#).

Here is the call graph for this function:



### 16.461.2.3 lookup()

```
template<std::size_t Size = 1024U>
Port_iface * Mac_table< Size >::lookup (
    Mac_addr dst,
    14_uint16_t vlan_id) const [inline]
```

Find the destination port for a MAC address and VLAN id.

#### Parameters

<i>dst</i>	MAC address
<i>vlan</i> <i>_id</i>	VLAN id

### Return values

<i>nullptr</i>	The MAC address is not known (yet)
<i>other</i>	Pointer to the destination port

Definition at line 58 of file [mac\\_table.h](#).

Referenced by [learn\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

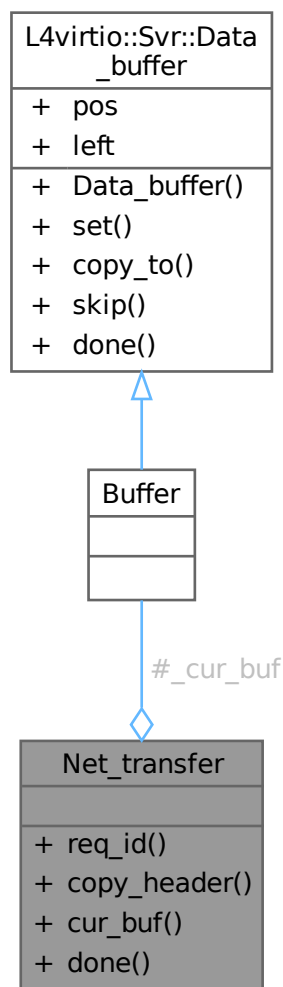
- `pkg/virtio-net-switch/server/switch/mac_table.h`

## 16.462 Net\_transfer Class Reference

A network request to only a single destination.

```
#include <request.h>
```

Collaboration diagram for Net\_transfer:



### Public Member Functions

- void const \* **req\_id** () const  
*Identifier for the underlying Net\_request, used for logging purposes.*
- virtual void **copy\_header** (Virtio\_net::Hdr \*dst\_header) const =0  
*Populate the virtio-net header for the destination.*
- **Buffer** & **cur\_buf** ()  
*Buffer containing (a part of) the packet data.*
- virtual bool **done** ()=0  
*Check whether the transfer has been completed, i.e.*

### 16.462.1 Detailed Description

A network request to only a single destination.



A `Net_request` can have multiple destinations (being a broadcast request, for example). That is why it is processed by multiple `Net_transfers`, each representing the delivery to a single destination port.

`Port_iface::handle_request` uses the `Net_transfer` to move one packet to the destination of the request.

Definition at line 33 of file [request.h](#).

## 16.462.2 Member Function Documentation

### 16.462.2.1 `cur_buf()`

```
Buffer & Net_transfer::cur_buf () [inline]
```

`Buffer` containing (a part of) the packet data.

Once emptied, a call to `done ()` might replenish the buffer, in case the net request consisted of multiple chained buffers.

Definition at line 54 of file [request.h](#).

### 16.462.2.2 `done()`

```
virtual bool Net_transfer::done () [pure virtual]
```

Check whether the transfer has been completed, i.e.

the entire packet data has been copied.

#### Return values

<i>false</i>	There is remaining packet data that needs to be copied.
<i>true</i>	The entire packet data has been copied.

#### Exceptions

<a href="#">L4virtio::Svr::Bad_descriptor</a>	Exception raised in SRC port queue.
-----------------------------------------------	-------------------------------------

The documentation for this class was generated from the following file:

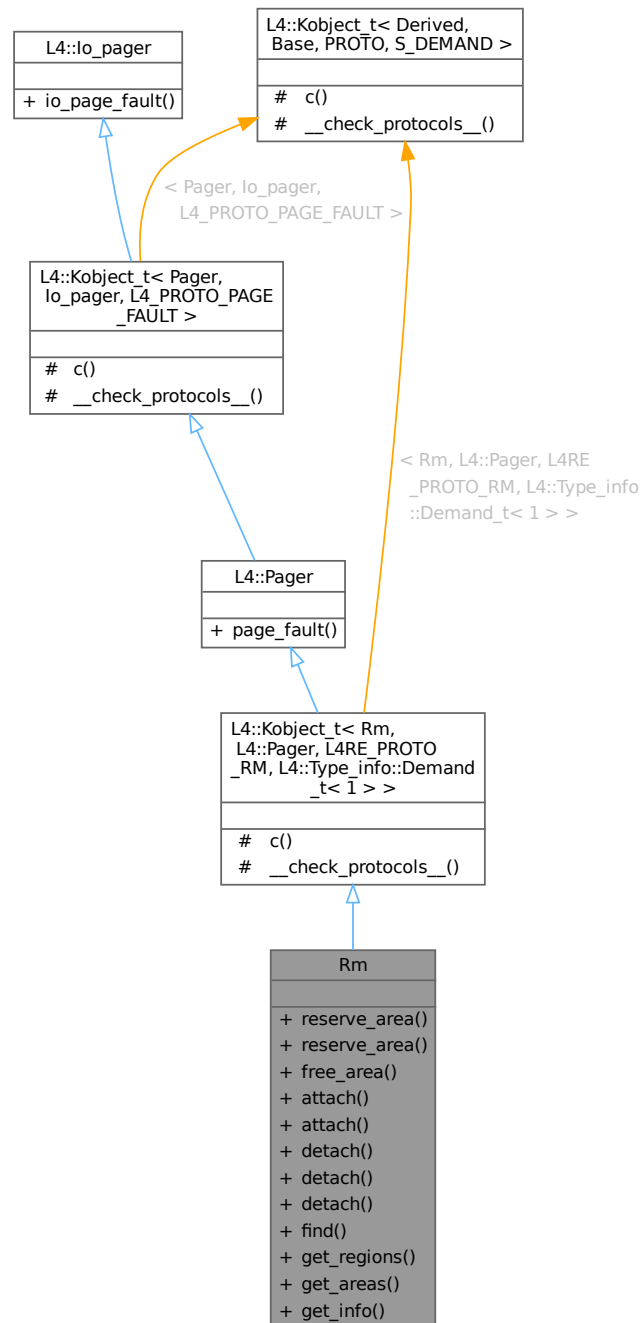
- `pkg/virtio-net-switch/server/switch/request.h`

## 16.463 Rm Class Reference

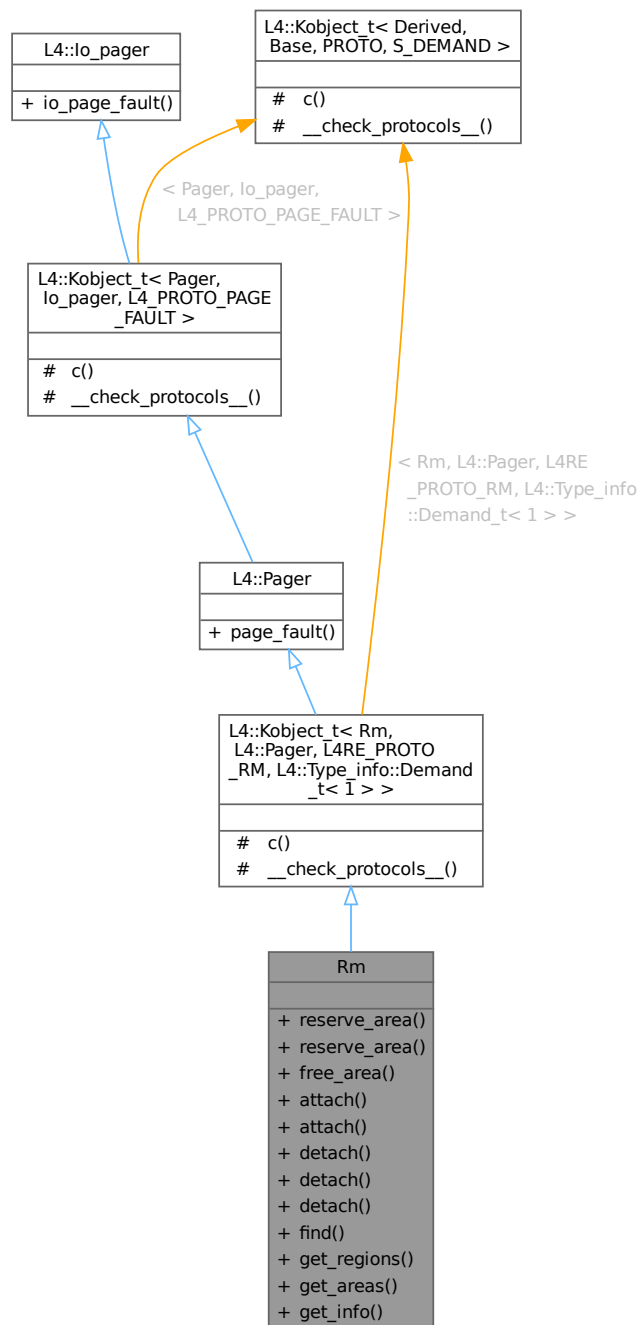
Region map.

```
#include <l4/re/rm>
```

Inheritance diagram for Rm:



Collaboration diagram for Rm:



## Data Structures

- struct [F](#)  
*Rm flags definitions.*
- class [Unique\\_region](#)  
*Unique region.*
- struct [Region](#)

*A region is a range of virtual addresses which is backed by content.*

- struct [Area](#)

*An area is a range of virtual addresses which is reserved, see [L4Re::Rm::reserve\\_area\(\)](#).*

## Public Types

- enum [Detach\\_result](#) { [Detached\\_ds](#) = 0 , [Kept\\_ds](#) = 1 , [Split\\_ds](#) = 2 , [Detach\\_result\\_mask](#) = 3 , [Detach\\_again](#) = 4 }
- Result values for detach operation.*
- enum [Region\\_flag\\_shifts](#) { [Caching\\_shift](#) = Dataspace::F::Caching\_shift }
- Region flag shifts.*
- enum [Detach\\_flags](#) { [Detach\\_exact](#) = 1 , [Detach\\_overlap](#) = 2 , [Detach\\_keep](#) = 4 }
- Flags for detach operation.*

## Public Member Functions

- long [reserve\\_area](#) ([l4\\_addr\\_t](#) \*start, unsigned long size, Flags flags=[Flags\(0\)](#), unsigned char align=[L4\\_PAGESHIFT](#)) const noexcept
- Reserve the given area in the region map.*
- template<typename T>  
long [reserve\\_area](#) (T \*\*start, unsigned long size, Flags flags=[Flags\(0\)](#), unsigned char align=[L4\\_PAGESHIFT](#)) const noexcept
- Reserve the given area in the region map.*
- long [free\\_area](#) ([l4\\_addr\\_t](#) addr)
- Free an area from the region map.*
- long [attach](#) ([l4\\_addr\\_t](#) \*start, unsigned long size, Flags flags, [L4::lpc::Cap](#)< Dataspace > mem, Offset offs=0, unsigned char align=[L4\\_PAGESHIFT](#), [L4::Cap](#)< [L4::Task](#) > const task=[L4::Cap](#)< [L4::Task](#) >::Invalid, char const \*name=nullptr, Offset backing\_offset=0) const noexcept
- Attach a data space to a region.*
- template<typename T>  
long [attach](#) (T \*\*start, unsigned long size, Flags flags, [L4::lpc::Cap](#)< Dataspace > mem, Offset offs=0, unsigned char align=[L4\\_PAGESHIFT](#), [L4::Cap](#)< [L4::Task](#) > const task=[L4::Cap](#)< [L4::Task](#) >::Invalid, char const \*name=nullptr, Offset backing\_offset=0) const noexcept
- Attach a data space to a region.*
- int [detach](#) ([l4\\_addr\\_t](#) addr, [L4::Cap](#)< Dataspace > \*mem, [L4::Cap](#)< [L4::Task](#) > const &task=This\_task) const noexcept
- Detach and unmap a region from the address space.*
- int [detach](#) (void \*addr, [L4::Cap](#)< Dataspace > \*mem, [L4::Cap](#)< [L4::Task](#) > const &task=This\_task) const noexcept
- Detach and unmap a region from the address space.*
- int [detach](#) ([l4\\_addr\\_t](#) start, unsigned long size, [L4::Cap](#)< Dataspace > \*mem, [L4::Cap](#)< [L4::Task](#) > const &task) const noexcept
- Detach and unmap all parts of the regions within the specified interval.*
- long [find](#) ([l4\\_addr\\_t](#) \*addr, unsigned long \*size, Offset \*offset, [L4Re::Rm::Flags](#) \*flags, [L4::Cap](#)< Dataspace > \*m) noexcept
- Find a region given an address and size.*
- long [get\\_regions](#) ([l4\\_addr\\_t](#) start, [L4::lpc::Ret\\_array](#)< [Region](#) > regions)
- Return the list of regions whose starting addresses are higher or equal to start in the address space managed by this region map.*
- long [get\\_areas](#) ([l4\\_addr\\_t](#) start, [L4::lpc::Ret\\_array](#)< [Area](#) > areas)
- Return the list of areas whose starting addresses are higher or equal to start in the address space managed by this region map.*
- long [get\\_info](#) ([l4\\_addr\\_t](#) addr, [L4::lpc::String](#)< char > &name, Offset &backing\_offset)
- Return auxiliary information of a region.*

## Public Member Functions inherited from [L4::Pager](#)

- [l4\\_msgtag\\_t page\\_fault](#) ([l4\\_umword\\_t](#) pfa, [l4\\_umword\\_t](#) pc, [L4::lpc::Rcv\\_fpage](#) rwin, [L4::lpc::Opt<L4::lpc::Snd\\_fpage & > fp](#))

*Page-fault protocol message.*

## Public Member Functions inherited from [L4::io\\_pager](#)

- [l4\\_msgtag\\_t io\\_page\\_fault](#) ([l4\\_fpage\\_t](#) io\_pfa, [l4\\_umword\\_t](#) pc, [L4::lpc::Rcv\\_fpage](#) rwin, [L4::lpc::Opt<L4::lpc::Snd\\_fpage & > fp](#))

*IO page fault protocol message.*

## Additional Inherited Members

### Protected Types inherited from

[L4::Kobject\\_t< Rm, L4::Pager, L4RE\\_PROTO\\_RM, L4::Type\\_info::Demand\\_t< 1 > >](#)

- typedef [Rm](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< PROTO, [Rm](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename [L4::Pager::\\_\\_Iface\\_list](#) > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Types inherited from

[L4::Kobject\\_t< Pager, io\\_pager, L4\\_PROTO\\_PAGE\\_FAULT >](#)

- typedef [Pager](#) **Class**  
*The target interface type (inheriting from [Kobject\\_t](#)).*
- typedef Typeid::Iface< PROTO, [Pager](#) > **\_\_Iface**  
*The interface description for the derived class.*
- typedef Typeid::Merge\_list< Typeid::Iface\_list< **\_\_Iface** >, typename [io\\_pager::\\_\\_Iface\\_list](#) > **\_\_Iface\_list**  
*The list of all RPC interfaces provided directly or through inheritance.*

### Protected Member Functions inherited from

[L4::Kobject\\_t< Rm, L4::Pager, L4RE\\_PROTO\\_RM, L4::Type\\_info::Demand\\_t< 1 > >](#)

- [L4::Cap< Class > c](#) () const noexcept  
*Get the capability to ourselves.*

### Protected Member Functions inherited from

[L4::Kobject\\_t< Pager, io\\_pager, L4\\_PROTO\\_PAGE\\_FAULT >](#)

- [L4::Cap< Class > c](#) () const noexcept  
*Get the capability to ourselves.*

**Static Protected Member Functions inherited from****[L4::Kobject\\_t< Rm, L4::Pager, L4RE\\_PROTO\\_RM, L4::Type\\_info::Demand\\_t< 1 > >](#)**

- static void **\_\_check\_protocols\_\_** () noexcept

*Helper to check for protocol conflicts.***Static Protected Member Functions inherited from****[L4::Kobject\\_t< Pager, lo\\_pager, L4\\_PROTO\\_PAGE\\_FAULT >](#)**

- static void **\_\_check\_protocols\_\_** () noexcept

*Helper to check for protocol conflicts.***16.463.1 Detailed Description**[Region](#) map.

See also

[Region map API](#) .Definition at line [81](#) of file [rm](#).**16.463.2 Member Enumeration Documentation****16.463.2.1 Detach\_flags**enum [L4Re::Rm::Detach\\_flags](#)

Flags for detach operation.

**Enumerator**

Detach_exact	Do an unmap of the exact region given.
Detach_overlap	Do an unmap of all overlapping regions.
Detach_keep	Do not free the detached data space, ignore the <a href="#">F::Detach_free</a> .

Definition at line [221](#) of file [rm](#).**16.463.2.2 Detach\_result**enum [L4Re::Rm::Detach\\_result](#)

Result values for detach operation.

**Enumerator**

Detached_ds	Detached data sapce.
Kept_ds	Kept data space.
Split_ds	Splitted data space, and done.
Detach_again	Detached data space, more to do.

Definition at line 89 of file [rm](#).

### 16.463.2.3 Region\_flag\_shifts

```
enum L4Re::Rm::Region_flag_shifts
```

[Region](#) flag shifts.

#### Enumerator

Caching_shift	Start of <a href="#">Rm</a> cache bits.
---------------	-----------------------------------------

Definition at line 101 of file [rm](#).

## 16.463.3 Member Function Documentation

### 16.463.3.1 attach() [1/2]

```
long L4Re::Rm::attach (
    l4_addr_t * start,
    unsigned long size,
    Rm::Flags flags,
    L4::Ipc::Cap< Dataspace > mem,
    Rm::Offset offs = 0,
    unsigned char align = L4_PAGESHIFT,
    L4::Cap< L4::Task > const task = L4::Cap<L4::Task>::Invalid,
    char const * name = nullptr,
    Rm::Offset backing_offset = 0) const [noexcept]
```

Attach a data space to a region.

#### Parameters

in, out	<i>start</i>	Virtual start address where the region manager shall attach the data space. Will be rounded down to the nearest start of a page. If <a href="#">L4Re::Rm::F::Search_addr</a> is given this value is used as the start address to search for a free virtual memory region and the resulting address is returned here. If <a href="#">L4Re::Rm::F::In_area</a> is given the value is used as a selector for the area (see <a href="#">L4Re::Rm::reserve_area</a> ) to attach the data space to.
	<i>size</i>	Size of the data space to attach (in bytes). Will be rounded up to the nearest multiple of the page size.

	<i>flags</i>	The flags control how and with which rights the dataspace is attached to the region. See <a href="#">L4Re::Rm::F::Attach_flags</a> and <a href="#">L4Re::Rm::F::Region_flags</a> . The caller must specify the desired rights of the attached region explicitly. The default set of rights is empty. If the <a href="#">F::Eager_map</a> flag is set this function may also return <a href="#">L4Re::Dataspace::map</a> error codes if the mapping fails.
	<i>mem</i>	Data space.
	<i>offs</i>	Offset into the data space to use.
	<i>align</i>	Alignment of the virtual region, log2-size, default: a page ( <a href="#">L4_PAGESHIFT</a> ). This is only meaningful if the <a href="#">L4Re::Rm::F::Search_addr</a> flag is used.
	<i>task</i>	Optional destination task of mapping if <a href="#">F::Eager_map</a> flag was passed. If invalid, the mapping is established in the current task. This parameter is only useful if the region manager is for a foreign task.
	<i>name</i>	Optional name of the region.
	<i>backing_offset</i>	Optional value describing an offset into the backing store of this region.

### Return values

0	Success
-L4_ENOENT	No area could be found (see <a href="#">L4Re::Rm::F::In_area</a> )
-L4_EPERM	Operation not allowed.
-L4_EINVAL	
-L4_EADDRNOTAVAIL	The given address is not available.
<0	IPC errors

Makes the whole or parts of a data space visible in the virtual memory of the corresponding task. The corresponding region in the virtual address space is backed with the contents of the dataspace.

### Note

When searching for a free place in the virtual address space, the space between *start* and the end of the virtual address space is searched.

There is no region object created, instead the region is defined by a virtual address within this range (see [L4Re::Rm::find](#)).

Definition at line 34 of file [rm\\_impl.h](#).

### 16.463.3.2 attach() [2/2]

```
template<typename T>
long L4Re::Rm::attach (
    T ** start,
    unsigned long size,
    Flags flags,
    L4::Ipc::Cap< Dataspace > mem,
    Offset offs = 0,
    unsigned char align = L4_PAGESHIFT,
    L4::Cap< L4::Task > const task = L4::Cap<L4::Task>::Invalid,
```



```
char const * name = nullptr,  
Offset backing_offset = 0) const [inline], [noexcept]
```

Attach a data space to a region.

#### Parameters

---

<i>in, out</i>	<i>start</i>	Virtual start address where the region manager shall attach the data space. Will be rounded down to the nearest start of a page. If <a href="#">L4Re::Rm::F::Search_addr</a> is given this value is used as the start address to search for a free virtual memory region and the resulting address is returned here. If <a href="#">L4Re::Rm::F::In_area</a> is given the value is used as a selector for the area (see <a href="#">L4Re::Rm::reserve_area</a> ) to attach the data space to.
	<i>size</i>	Size of the data space to attach (in bytes). Will be rounded up to the nearest multiple of the page size.
	<i>flags</i>	The flags control how and with which rights the dataspace is attached to the region. See <a href="#">L4Re::Rm::F::Attach_flags</a> and <a href="#">L4Re::Rm::F::Region_flags</a> . The caller must specify the desired rights of the attached region explicitly. The default set of rights is empty. If the <a href="#">F::Eager_map</a> flag is set this function may also return <a href="#">L4Re::Dataspace::map</a> error codes if the mapping fails.
	<i>mem</i>	Data space.
	<i>offs</i>	Offset into the data space to use.
	<i>align</i>	Alignment of the virtual region, log2-size, default: a page ( <a href="#">L4_PAGESHIFT</a> ). This is only meaningful if the <a href="#">L4Re::Rm::F::Search_addr</a> flag is used.
	<i>task</i>	Optional destination task of mapping if <a href="#">F::Eager_map</a> flag was passed. If invalid, the mapping is established in the current task. This parameter is only useful if the region manager is for a foreign task.
	<i>name</i>	Optional name of the region.
	<i>backing_offset</i>	Optional value describing an offset into the backing store of this region.

### Return values

0	Success
-L4_ENOENT	No area could be found (see <a href="#">L4Re::Rm::F::In_area</a> )
-L4_EPERM	Operation not allowed.
-L4_EINVAL	
-L4_EADDRNOTAVAIL	The given address is not available.
<0	IPC errors

Makes the whole or parts of a data space visible in the virtual memory of the corresponding task. The corresponding region in the virtual address space is backed with the contents of the dataspace.

### Note

When searching for a free place in the virtual address space, the space between *start* and the end of the virtual address space is searched.

There is no region object created, instead the region is defined by a virtual address within this range (see [L4Re::Rm::find](#)).

Definition at line 409 of file [rm](#).

### 16.463.3.3 detach() [1/3]

```
int L4Re::Rm::detach (
    l4_addr_t addr,
```

```

L4::Cap< Dataspace > * mem,
L4::Cap< L4::Task > const & task = This_task) const [inline], [noexcept]

```

Detach and unmap a region from the address space.

### Parameters

	<i>addr</i>	Virtual address of region, any address within the region is valid.
out	<i>mem</i>	Dataspace that is affected. Give 0 if not interested.
	<i>task</i>	This argument specifies the task where the pages are unmapped. Provide <a href="#">L4::Cap&lt;L4::Task&gt;::Invalid</a> for none. The default is the current task.

### Return values

<a href="#">L4Re::Rm::Detach_result</a>	On success.
<a href="#">-L4_ENOENT</a>	No region found.
<a href="#">&lt;0</a>	IPC errors

Frees a region in the virtual address space given by *addr* (address type). The corresponding part of the address space is now available again.

Definition at line 765 of file [rm](#).

#### 16.463.3.4 detach() [2/3]

```

int L4Re::Rm::detach (
    l4_addr_t start,
    unsigned long size,
    L4::Cap< Dataspace > * mem,
    L4::Cap< L4::Task > const & task) const [inline], [noexcept]

```

Detach and unmap all parts of the regions within the specified interval.

### Parameters

	<i>start</i>	Start of area to detach, must be within region.
	<i>size</i>	Size of of area to detach (in bytes).
out	<i>mem</i>	Dataspace that is affected. Give 0 if not interested.
	<i>task</i>	This argument specifies the task where the pages are unmapped. Provide <a href="#">L4::Cap&lt;L4::Task&gt;::Invalid</a> for none. The default is the current task.

### Return values

<a href="#">L4Re::Rm::Detach_result</a>	On success.
<a href="#">-L4_ENOENT</a>	No region found.
<a href="#">&lt;0</a>	IPC errors

Frees all regions within the interval given by start and size. If a region overlaps the start or the end of the interval this region is only detached partly. If the interval is within one region the original region is split up into two separate regions.

Definition at line 778 of file [rm](#).

### 16.463.3.5 detach() [3/3]

```
int L4Re::Rm::detach (
    void * addr,
    L4::Cap< Dataspace > * mem,
    L4::Cap< L4::Task > const & task = This_task) const [inline], [noexcept]
```

Detach and unmap a region from the address space.

#### Parameters

	<i>addr</i>	Virtual address of region, any address within the region is valid.
out	<i>mem</i>	Dataspace that is affected. Give 0 if not interested.
	<i>task</i>	This argument specifies the task where the pages are unmapped. Provide <a href="#">L4::Cap&lt;L4::Task&gt;::Invalid</a> for none. The default is the current task.

#### Return values

<a href="#">L4Re::Rm::Detach_result</a>	On success.
<a href="#">-L4_ENOENT</a>	No region found.
<a href="#">&lt;0</a>	IPC errors

Frees a region in the virtual address space given by addr (address type). The corresponding part of the address space is now available again.

Definition at line 770 of file [rm](#).

### 16.463.3.6 find()

```
long L4Re::Rm::find (
    l4_addr_t * addr,
    unsigned long * size,
    Offset * offset,
    L4Re::Rm::Flags * flags,
    L4::Cap< Dataspace > * m) [inline], [noexcept]
```

Find a region given an address and size.

#### Parameters

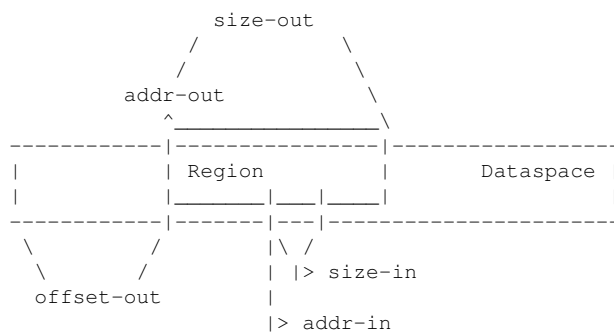
in, out	<i>addr</i>	Address to look for. Returns the start address of the found region.
---------	-------------	---------------------------------------------------------------------

<i>in, out</i>	<i>size</i>	Size of the area to look for (in bytes). Returns the size of the found region (in bytes).
<i>out</i>	<i>offset</i>	Offset at the beginning of the region within the associated dataspace.
<i>out</i>	<i>flags</i>	Region flags, see <a href="#">F::Region_flags</a> (and <a href="#">F::In_area</a> ).
<i>out</i>	<i>m</i>	Associated dataspace or paging service.

### Return values

<i>0</i>	Success
<i>-L4_EPERM</i>	Operation not allowed.
<i>-L4_ENOENT</i>	No region found.
<i>&lt;0</i>	IPC errors

This function returns the properties of the region that contains the area described by the `addr` and `size` parameter. If no such region is found but a reserved area, the area is returned and [F::In\\_area](#) is set in `flags`. Note, in the case of an area the `offset` and `m` return values are invalid.



### Note

The value of the `size` input parameter should be 1 to assure that a region can be determined unambiguously.

Definition at line [672](#) of file [rm](#).

### 16.463.3.7 free\_area()

```
long L4Re::Rm::free_area (
    l4_addr_t addr)
```

Free an area from the region map.

### Parameters

<i>addr</i>	An address within the area to free.
-------------	-------------------------------------

### Return values

0	Success
-L4_ENOENT	No area found.
<0	IPC errors

**Note**

The data spaces that are attached to that area are not detached by this operation.

**See also**

[reserve\\_area\(\)](#) for more information about areas.

**16.463.3.8 get\_areas()**

```
long L4Re::Rm::get_areas (
    l4_addr_t start,
    L4::Ipc::Ret_array< Area > areas)
```

Return the list of areas whose starting addresses are higher or equal to `start` in the address space managed by this region map.

**Parameters**

	<i>start</i>	Virtual address from where to start searching.
out	<i>areas</i>	List of areas found in this region map.

**Return values**

$\geq 0$	Number of returned areas in the <code>areas</code> array.
<0	IPC errors

**Note**

The returned list of areas might not be complete and the caller shall use the function repeatedly with a start address one larger than the end address of the last area from the previous call.

**16.463.3.9 get\_info()**

```
long L4Re::Rm::get_info (
    l4_addr_t addr,
    L4::Ipc::String< char > & name,
    Offset & backing_offset)
```

Return auxiliary information of a region.

This is a debugging feature and might not be available.

**Parameters**

	<i>addr</i>	Virtual address of the region.
out	<i>name</i>	Name of the region.
out	<i>backing_offset</i>	Backing offset information.

**Return values**

0	Success
-L4_ENOENT	Region not found.
-L4_ENOSYS	Function not available.
<0	IPC errors

**16.463.3.10 get\_regions()**

```
long L4Re::Rm::get_regions (
    l4_addr_t start,
    L4::Ipc::Ret_array< Region > regions)
```

Return the list of regions whose starting addresses are higher or equal to *start* in the address space managed by this region map.

**Parameters**

	<i>start</i>	Virtual address from where to start searching.
out	<i>regions</i>	List of regions found in this region map.

**Return values**

>=0	Number of returned regions in the <i>regions</i> array.
<0	IPC errors

**Note**

The returned list of regions might not be complete and the caller shall use the function repeatedly with a start address one larger than the end address of the last region from the previous call.

**16.463.3.11 reserve\_area() [1/2]**

```
long L4Re::Rm::reserve_area (
    l4_addr_t * start,
    unsigned long size,
    Flags flags = Flags(0),
    unsigned char align = L4_PAGESHIFT) const [inline], [noexcept]
```

Reserve the given area in the region map.

**Parameters**

<i>in, out</i>	<i>start</i>	The virtual start address of the area to reserve. Returns the start address of the area.
	<i>size</i>	The size of the area to reserve (in bytes).
	<i>flags</i>	Flags for the reserved area (see <a href="#">L4Re::Rm::F::Region_flags</a> and <a href="#">L4Re::Rm::F::Attach_flags</a> ).
	<i>align</i>	Alignment of area if searched as bits (log2 value).

### Return values

<i>0</i>	Success
<i>-L4_EADDRNOTAVAIL</i>	The given area cannot be reserved.
<i>&lt;0</i>	IPC errors

This function reserves an area within the virtual address space managed by the region map. There are two kinds of areas available:

- Reserved areas (*flags* = [L4Re::Rm::F::Reserved](#)), where no data spaces can be attached
- Special purpose areas (*flags* = 0), where data spaces can be attached to the area via the [L4Re::Rm::F::In\\_area](#) flag and a start address within the area itself.

### Note

When searching for a free place in the virtual address space (with *flags* = [L4Re::Rm::F::Search\\_addr](#)), the space between *start* and the end of the virtual address space is searched.

Definition at line [280](#) of file [rm](#).

### 16.463.3.12 `reserve_area()` [2/2]

```
template<typename T>
long L4Re::Rm::reserve_area (
    T ** start,
    unsigned long size,
    Flags flags = Flags(0),
    unsigned char align = L4\_PAGESHIFT) const [inline], [noexcept]
```

Reserve the given area in the region map.

### Parameters

<i>in, out</i>	<i>start</i>	The virtual start address of the area to reserve. Returns the start address of the area.
	<i>size</i>	The size of the area to reserve (in bytes).
	<i>flags</i>	Flags for the reserved area (see <a href="#">F::Region_flags</a> and <a href="#">F::Attach_flags</a> ).
	<i>align</i>	Alignment of area if searched as bits (log2 value).

### Return values



0	Success
-L4_EADDRNOTAVAIL	The given area cannot be reserved.
<0	IPC errors

For more information, please refer to the analogous function

See also

[L4Re::Rm::reserve\\_area](#).

Definition at line 306 of file [rm](#).

The documentation for this class was generated from the following files:

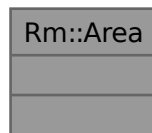
- [l4/re/rm](#)
- [l4/re/impl/rm\\_impl.h](#)

## 16.464 Rm::Area Struct Reference

An area is a range of virtual addresses which is reserved, see [L4Re::Rm::reserve\\_area\(\)](#).

```
#include <rm>
```

Collaboration diagram for Rm::Area:



### 16.464.1 Detailed Description

An area is a range of virtual addresses which is reserved, see [L4Re::Rm::reserve\\_area\(\)](#).

See also

[Region map API](#)

Definition at line 699 of file [rm](#).

The documentation for this struct was generated from the following file:

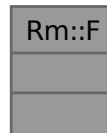
- [l4/re/rm](#)

## 16.465 Rm::F Struct Reference

Rm flags definitions.

```
#include <rm>
```

Collaboration diagram for Rm::F:



### Public Types

- enum [Attach\\_flags](#) : l4\_uint32\_t {  
[Search\\_addr](#) = 0x20000 , [In\\_area](#) = 0x40000 , [Eager\\_map](#) = 0x80000 , [No\\_eager\\_map](#) = 0x100000 ,  
[Attach\\_mask](#) = 0x1f0000 }  
*Flags for attach operation.*
- enum [Region\\_flags](#) : l4\_uint16\_t {  
[Rights\\_mask](#) = 0x0f , [R](#) = Dataspace::F::R , [W](#) = Dataspace::F::W , [X](#) = Dataspace::F::X ,  
[RW](#) = Dataspace::F::RW , [RX](#) = Dataspace::F::RX , [RWX](#) = Dataspace::F::RWX , [Kernel](#) = 0x100 ,  
[Detach\\_free](#) = 0x200 , [Pager](#) = 0x400 , [Reserved](#) = 0x800 , [Caching\\_mask](#) = Dataspace::F::Caching\_mask ,  
[Cache\\_normal](#) = Dataspace::F::Normal , [Cache\\_buffered](#) = Dataspace::F::Bufferable , [Cache\\_uncached](#) =  
Dataspace::F::Uncacheable , [Ds\\_map\\_mask](#) = 0xff ,  
[Region\\_flags\\_mask](#) = 0xffff }  
*Region flags (permissions, cacheability, special).*

### 16.465.1 Detailed Description

Rm flags definitions.

Definition at line 108 of file [rm](#).

### 16.465.2 Member Enumeration Documentation

#### 16.465.2.1 Attach\_flags

```
enum L4Re::Rm::F::Attach_flags : l4_uint32_t
```

Flags for attach operation.

#### Enumerator

Search_addr	Search for a suitable address range.
In_area	Search only in area, or map into area.
Eager_map	Eagerly map the attached data space in.
No_eager_map	Prevent eager mapping of the attached data space.
Attach_mask	Mask of all attach flags.

Definition at line 111 of file [rm](#).

### 16.465.2.2 Region\_flags

```
enum L4Re::Rm::F::Region_flags : 14_uint16_t
```

[Region](#) flags (permissions, cacheability, special).

#### Enumerator

Rights_mask	<a href="#">Region</a> rights.
R	Readable region.
W	Writable region.
X	Executable region.
RW	Readable and writable region.
RX	Readable and executable region.
RWX	Readable, writable and executable region.
Kernel	Kernel-provided memory (KUMEM).
Detach_free	Free the portion of the data space after detach.
Pager	<a href="#">Region</a> has a pager.
Reserved	<a href="#">Region</a> is reserved (blocked).
Caching_mask	Mask of all <a href="#">Rm</a> cache bits.
Cache_normal	Cache bits for normal cacheable memory. This is the default if no other cache-related flag was specified.
Cache_buffered	Cache bits for buffered (write combining) memory.
Cache_uncached	Cache bits for uncached memory.
Ds_map_mask	Mask for all bits for cache options and rights.
Region_flags_mask	Mask of all region flags.

Definition at line 128 of file [rm](#).

The documentation for this struct was generated from the following file:

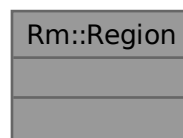
- [l4/re/rm](#)

## 16.466 Rm::Region Struct Reference

A region is a range of virtual addresses which is backed by content.

```
#include <rm>
```

Collaboration diagram for Rm::Region:



### 16.466.1 Detailed Description

A region is a range of virtual addresses which is backed by content.

See also

[Region map API](#)

Definition at line 686 of file [rm](#).

The documentation for this struct was generated from the following file:

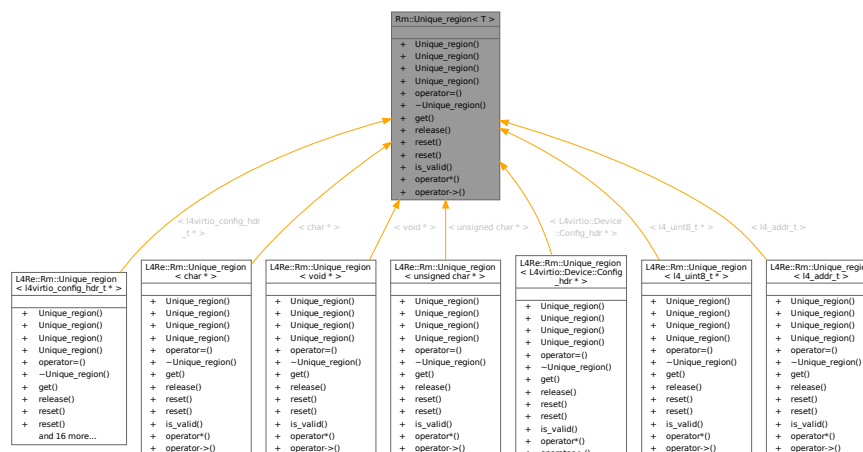
- [l4/re/rm](#)

## 16.467 Rm::Unique\_region< T > Class Template Reference

Unique region.

```
#include <rm>
```

Inheritance diagram for Rm::Unique\_region< T >:



Collaboration diagram for Rm::Unique\_region< T >:

Rm::Unique_region< T >
<ul style="list-style-type: none"> <li>+ Unique_region()</li> <li>+ Unique_region()</li> <li>+ Unique_region()</li> <li>+ Unique_region()</li> <li>+ operator=()</li> <li>+ ~Unique_region()</li> <li>+ get()</li> <li>+ release()</li> <li>+ reset()</li> <li>+ reset()</li> <li>+ is_valid()</li> <li>+ operator*()</li> <li>+ operator-&gt;()</li> </ul>

## Public Member Functions

- **Unique\_region** () noexcept  
*Construct an invalid [Unique\\_region](#).*
- **Unique\_region** (T addr) noexcept  
*Construct a [Unique\\_region](#) from an address.*
- **Unique\_region** (T addr, L4::Cap< Rm > const &rm) noexcept  
*Construct a valid [Unique\\_region](#) from an address and a region manager.*
- **Unique\_region** (Unique\_region &&o) noexcept  
*Move-Construct a [Unique\\_region](#).*
- **Unique\_region** & operator= (Unique\_region &&o) noexcept  
*Move-assign a [Unique\\_region](#).*
- **~Unique\_region** () noexcept  
*Destructor.*
- T **get** () const noexcept  
*Return the address.*
- T **release** () noexcept  
*Return the address and invalidate the [Unique\\_region](#).*
- void **reset** (T addr, L4::Cap< Rm > const &rm) noexcept  
*Set new address and region manager.*
- void **reset** () noexcept  
*Make the [Unique\\_region](#) invalid.*
- bool **is\_valid** () const noexcept  
*Check if the [Unique\\_region](#) is valid.*

- **T operator\*** () const noexcept  
*Dereference the address.*
- **T operator->** () const noexcept  
*Member access for the address.*

### 16.467.1 Detailed Description

```
template<typename T>
class Rm::Unique_region< T >
```

Unique region.

Capture a single region with automatic detach on destruction and unique ownership. Stores the start address and the region-mapper capability internally. A unique region is valid precisely if the internal region-mapper capability is valid. The features for unique ownership and automatic detach are only active for valid unique regions.

Definition at line [435](#) of file [rm](#).

### 16.467.2 Constructor & Destructor Documentation

#### 16.467.2.1 Unique\_region() [1/3]

```
template<typename T>
L4Re::Rm::Unique_region< T >::Unique_region (
    T addr) [inline], [explicit], [noexcept]
```

Construct a [Unique\\_region](#) from an address.

No region manager is set.

#### Parameters

<i>addr</i>	The new address
-------------	-----------------

Definition at line [456](#) of file [rm](#).

#### 16.467.2.2 Unique\_region() [2/3]

```
template<typename T>
L4Re::Rm::Unique_region< T >::Unique_region (
    T addr,
    L4::Cap< Rm > const & rm) [inline], [noexcept]
```

Construct a valid [Unique\\_region](#) from an address and a region manager.

#### Parameters

<i>addr</i>	The address
<i>rm</i>	The region manager

Definition at line 465 of file [rm](#).

### 16.467.2.3 Unique\_region() [3/3]

```
template<typename T>
L4Re::Rm::Unique_region< T >::Unique_region (
    Unique_region< T > && o) [inline], [noexcept]
```

Move-Construct a [Unique\\_region](#).

#### Parameters

<i>o</i>	L-value reference to other region.
----------	------------------------------------

Definition at line 473 of file [rm](#).

### 16.467.2.4 ~Unique\_region()

```
template<typename T>
L4Re::Rm::Unique_region< T >::~~Unique_region () [inline], [noexcept]
```

Destructor.

If the region is valid, call [detach](#).

Definition at line 498 of file [rm](#).

## 16.467.3 Member Function Documentation

### 16.467.3.1 get()

```
template<typename T>
T L4Re::Rm::Unique_region< T >::get () const [inline], [noexcept]
```

Return the address.

#### Returns

the address

Definition at line 509 of file [rm](#).

**16.467.3.2 is\_valid()**

```
template<typename T>
bool L4Re::Rm::Unique_region< T >::is_valid () const [inline], [noexcept]
```

Check if the `Unique_region` is valid.

**Returns**

true iff the `Unique_region` is valid

Definition at line 549 of file `rm`.

**16.467.3.3 operator=()**

```
template<typename T>
Unique_region & L4Re::Rm::Unique_region< T >::operator= (
    Unique_region< T > && o) [inline], [noexcept]
```

Move-assign a `Unique_region`.

**Parameters**

<i>o</i>	L-value reference to region to assign from
----------	--------------------------------------------

Definition at line 481 of file `rm`.

**16.467.3.4 release()**

```
template<typename T>
T L4Re::Rm::Unique_region< T >::release () [inline], [noexcept]
```

Return the address and invalidate the `Unique_region`.

**Returns**

the address

Definition at line 517 of file `rm`.

**16.467.3.5 reset()**

```
template<typename T>
void L4Re::Rm::Unique_region< T >::reset (
    T addr,
    L4::Cap< Rm > const & rm) [inline], [noexcept]
```

Set new address and region manager.

**Parameters**



<i>addr</i>	The new address
<i>rm</i>	The new region manager

Definition at line 529 of file [rm](#).

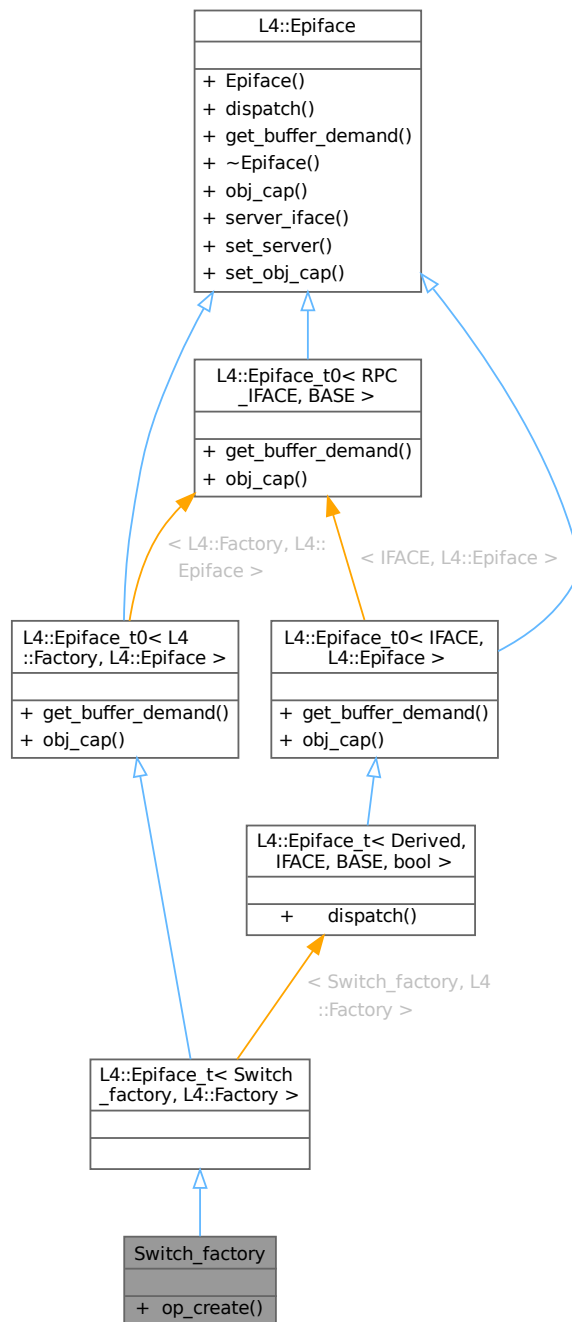
The documentation for this class was generated from the following file:

- [l4/re/rm](#)

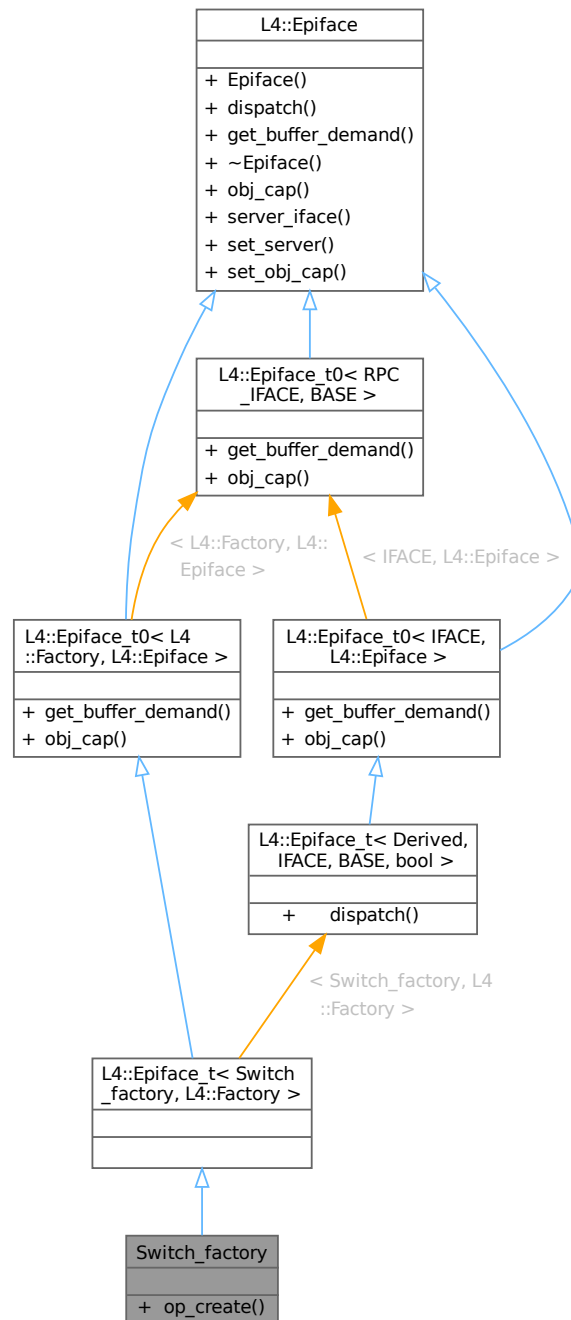
## 16.468 Switch\_factory Class Reference

The IPC interface for creating ports.

Inheritance diagram for Switch\_factory:



Collaboration diagram for Switch\_factory:



### Public Member Functions

- long `op_create` (L4::Factory::Rights, L4::lpc::Cap< void > &res, l4\_umword\_t type, L4::lpc::Varg\_list\_ref va)  
Handle factory protocol.

### Public Member Functions inherited from L4::Epiface\_t0< L4::Factory, L4::Epiface >

- Type\_info::Demand `get_buffer_demand` () const

*Get the server-side buffer demand based in IFACE.*

- `Cap< L4::Factory > obj_cap () const`

*Get the (typed) capability to this object.*

## Public Member Functions inherited from `L4::Epiface`

- `Epiface ()`

*Make a server object.*

- `virtual ~Epiface ()=0`

*Destroy the object.*

- `Stored_cap obj_cap () const`

*Get the capability to the kernel object belonging to this object.*

- `Server_iface * server_iface () const`

*Get pointer to server interface at which the object is currently registered.*

- `int set_server (Server_iface *srv, Cap< void > cap, bool managed=false)`

*Set server registration info for the object.*

- `void set_obj_cap (Cap< void > const &cap)`

*Deprecated server registration function.*

## Additional Inherited Members

## Public Types inherited from `L4::Epiface_t0< L4::Factory, L4::Epiface >`

- `typedef L4::Factory Interface`

*Data type of the IPC interface definition.*

## Public Types inherited from `L4::Epiface`

- `typedef ipc_svr::Server_iface Server_iface`

*Type for abstract server interface.*

- `typedef ipc_svr::Server_iface::Demand Demand`

*Type for server-side receive buffer demand.*

### 16.468.1 Detailed Description

The IPC interface for creating ports.

The Switch factory provides an IPC interface to create ports. Ports are the only option for a client to communicate with the switch and, thus, with other network devices.

The `Switch_factory` gets constructed when the net switch application gets started. It thereafter gets registered on the switch's server to serve IPC `create` calls.

Definition at line 114 of file `main.cc`.

## 16.468.2 Member Function Documentation

### 16.468.2.1 op\_create()

```
long Switch_factory::op_create (
    L4::Factory::Rights ,
    L4::Ipc::Cap< void > & res,
    l4_umword_t type,
    L4::Ipc::Varg_list_ref va) [inline]
```

Handle factory protocol.

This function is invoked after an incoming factory::create request and creates a new port or statistics interface if possible.

Definition at line 528 of file [main.cc](#).

References [L4\\_EINVAL](#).

The documentation for this class was generated from the following file:

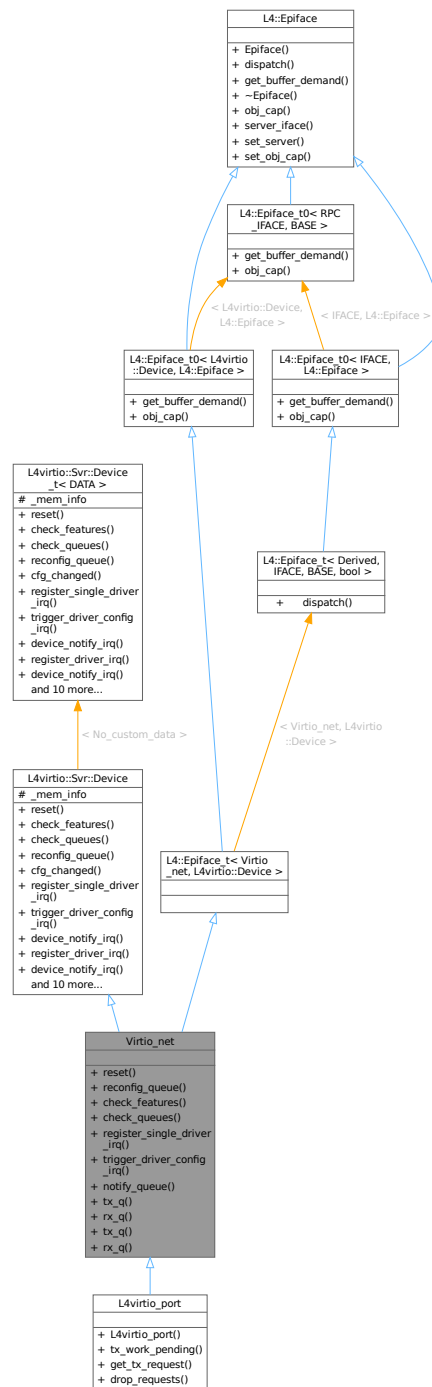
- pkg/virtio-net-switch/server/switch/main.cc

## 16.469 Virtio\_net Class Reference

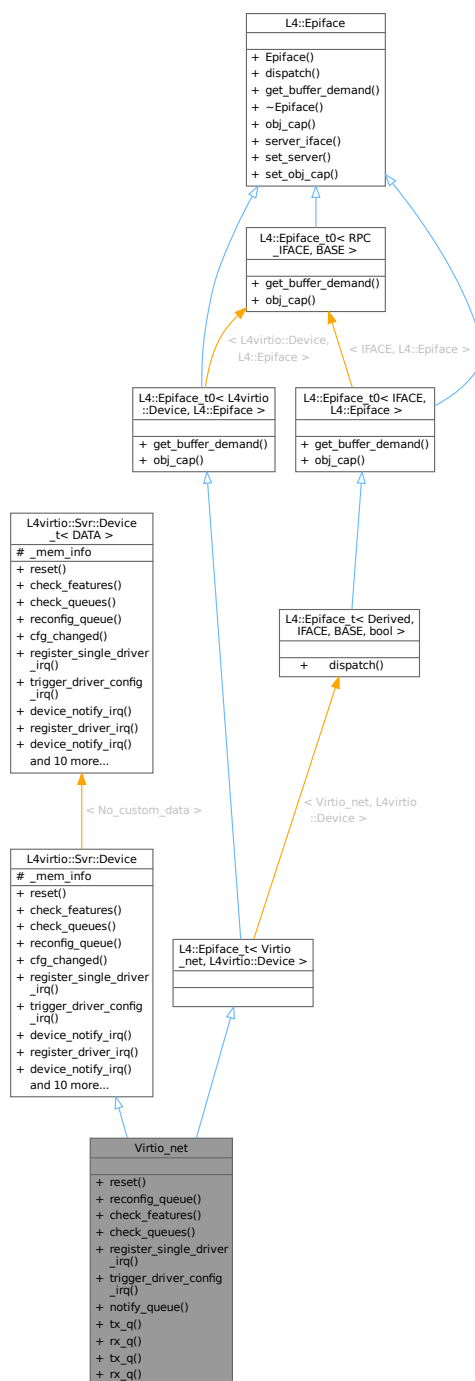
The Base class of a Port.

```
#include <virtio_net.h>
```

Inheritance diagram for Virtio\_net:



Collaboration diagram for Virtio\_net:



## Public Member Functions

- void **reset** () override  
*reset callback, called for doing a device reset*
- int **reconfig\_queue** (unsigned index) override  
*callback for client queue-config request*
- bool **check\_features** () override

- callback for checking the subset of accepted features*
- bool **check\_queues** () override  
*Check whether both virtqueues are ready.*
- void **register\_single\_driver\_irq** () override  
*Save the `_kick_guest_irq` that the client sent via `device_notification_irq()`.*
- void **trigger\_driver\_config\_irq** () override  
*callback for triggering configuration change notification IRQ*
- void **notify\_queue** (L4virtio::Svr::Virtqueue \*queue)  
*Trigger the `_kick_guest_irq` IRQ.*
- Virtqueue \* **tx\_q** ()  
*Getter for the transmission queue.*
- Virtqueue \* **rx\_q** ()  
*Getter for the receive queue.*
- Virtqueue const \* **tx\_q** () const  
*Getter for the transmission queue.*
- Virtqueue const \* **rx\_q** () const  
*Getter for the receive queue.*

## Public Member Functions inherited from L4virtio::Svr::Device\_t< No\_custom\_data >

- virtual void **cfg\_changed** (unsigned)  
*callback for client device configuration changes*
- virtual L4::Cap< L4::Irq > **device\_notify\_irq** () const  
*callback to gather the device notification IRQ (old-style)*
- virtual void **register\_driver\_irq** (unsigned idx)  
*Callback for registering an notification IRQ (multi IRQ).*
- virtual L4::Cap< L4::Irq > **device\_notify\_irq** (unsigned idx)  
*Callback to gather the device notification IRQ (multi IRQ).*
- virtual unsigned **num\_events\_supported** () const  
*Return the highest notification index supported.*
- **Device\_t** (Dev\_config \*dev\_config)  
*Make a device for the given config.*
- **Mem\_list** const \* **mem\_info** () const  
*Get the memory region list used for this device.*
- void **reset\_queue\_config** (unsigned idx, unsigned num\_max, bool inc\_generation=false)  
*Trigger reset for the configuration space for queue idx.*
- void **init\_mem\_info** (unsigned num)  
*Initialize the memory region list to the given maximum.*
- void **device\_error** ()  
*Transition device into `DEVICE_NEEDS_RESET` state.*
- bool **setup\_queue** (Virtqueue \*q, unsigned qn, unsigned num\_max)  
*Enable/disable the specified queue.*
- bool **handle\_mem\_cmd\_write** ()  
*Check for a value in the `cmd` register and handle a write.*
- void **enable\_trusted\_ds\_validation** ()  
*Enable trusted dataspace validation.*
- void **add\_trusted\_dataspaces** (std::shared\_ptr< Ds\_vector const > ds)  
*Provide a list of trusted dataspaces that can be used for validation.*



## Public Member Functions inherited from [L4::Epiface\\_t0](#)< [L4virtio::Device](#), [L4::Epiface](#) >

- [Type\\_info::Demand](#) **get\_buffer\_demand** () const  
*Get the server-side buffer demand based in IFACE.*
- [Cap](#)< [L4virtio::Device](#) > **obj\_cap** () const  
*Get the (typed) capability to this object.*

## Public Member Functions inherited from [L4::Epiface](#)

- **Epiface** ()  
*Make a server object.*
- virtual **~Epiface** ()=0  
*Destroy the object.*
- Stored\_cap [obj\\_cap](#) () const  
*Get the capability to the kernel object belonging to this object.*
- [Server\\_iface](#) \* **server\_iface** () const  
*Get pointer to server interface at which the object is currently registered.*
- int **set\_server** ([Server\\_iface](#) \*srv, [Cap](#)< void > cap, bool managed=false)  
*Set server registration info for the object.*
- void **set\_obj\_cap** ([Cap](#)< void > const &cap)  
*Deprecated server registration function.*

## Additional Inherited Members

## Public Types inherited from [L4::Epiface\\_t0](#)< [L4virtio::Device](#), [L4::Epiface](#) >

- typedef [L4virtio::Device](#) **Interface**  
*Data type of the IPC interface definition.*

## Public Types inherited from [L4::Epiface](#)

- typedef [lpc\\_svr::Server\\_iface](#) **Server\_iface**  
*Type for abstract server interface.*
- typedef [lpc\\_svr::Server\\_iface::Demand](#) **Demand**  
*Type for server-side receive buffer demand.*

## Protected Attributes inherited from [L4virtio::Svr::Device\\_t](#)< [No\\_custom\\_data](#) >

- [Mem\\_list](#) **mem\_info**  
*Memory region list.*

### 16.469.1 Detailed Description

The Base class of a Port.

This class provides the Virtio network protocol specific implementation aspects of a port.

[Virtio\\_net](#) comprises the virtqueues for both, the incoming and the outgoing network requests:

- The transmission queue, containing requests to be transmitted to other ports. The transmission queue is filled by the client, this port relates to.
- The receive queue, containing requests that have been transmitted from other ports. The receive queue is filled by the switch.

Definition at line 71 of file [virtio\\_net.h](#).

### 16.469.2 Member Function Documentation

#### 16.469.2.1 notify\_queue()

```
void Virtio_net::notify_queue (
    L4virtio::Svr::Virtqueue * queue) [inline]
```

Trigger the `_kick_guest_irq` IRQ.

This function gets called on the receiving port, when a request was successfully transmitted by the switch.

Definition at line 269 of file [virtio\\_net.h](#).

References [L4VIRTIO\\_IRQ\\_STATUS\\_VRING](#).

The documentation for this class was generated from the following file:

- `pkg/virtio-net-switch/server/switch/virtio_net.h`

## 16.470 Virtio\_net\_request Class Reference

Abstraction for a network request.

```
#include <request_l4virtio.h>
```

Collaboration diagram for `Virtio_net_request`:

Virtio_net_request
<ul style="list-style-type: none"> <li>+ dst_mac()</li> <li>+ src_mac()</li> <li>+ drop_requests()</li> <li>+ get_request()</li> </ul>

## Public Member Functions

- [Mac\\_addr dst\\_mac](#) () const  
*Get the Mac address of the destination port.*
- [Mac\\_addr src\\_mac](#) () const  
*Get the Mac address of the source port.*

## Static Public Member Functions

- static void [drop\\_requests](#) ([Virtio\\_net](#) \*dev, [L4virtio::Svr::Virtqueue](#) \*queue)  
*Drop all requests of a specific queue.*
- static std::optional< [Virtio\\_net\\_request](#) > [get\\_request](#) ([Virtio\\_net](#) \*dev, [L4virtio::Svr::Virtqueue](#) \*queue)  
*Construct a request from the next entry of a provided queue.*

## 16.470.1 Detailed Description

Abstraction for a network request.

A [Virtio\\_net\\_request](#) is constructed by the source port, using the static function [get\\_request](#) () as part of [Port\\_iface::get\\_tx\\_request](#) ().

On destruction, [finish](#) () will be called, which, will trigger the client IRQ of the source client.

Definition at line 35 of file [request\\_l4virtio.h](#).

## 16.470.2 Member Function Documentation

### 16.470.2.1 drop\_requests()

```
void Virtio_net_request::drop_requests (
    Virtio_net * dev,
    L4virtio::Svr::Virtqueue * queue) [inline], [static]
```

Drop all requests of a specific queue.

This function is used for example to drop all requests in the transmission queue of a monitor port, since monitor ports are not allowed to transmit data.

#### Parameters

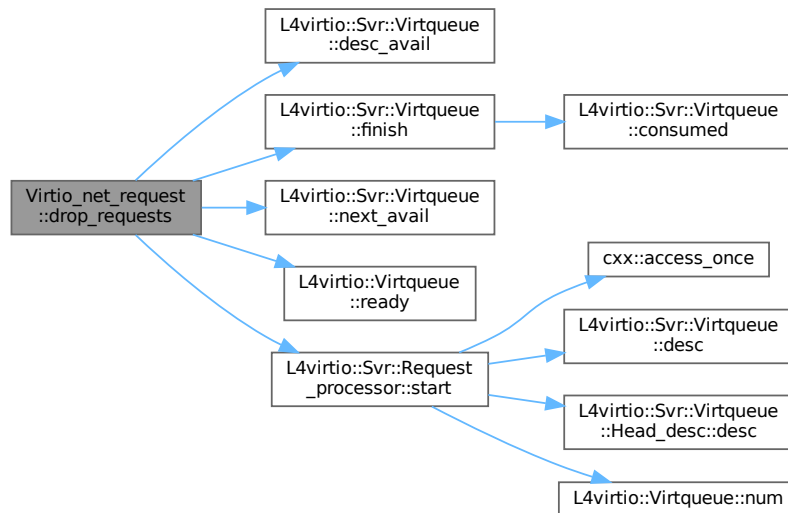
<i>dev</i>	Port of the provided virtqueue.
<i>queue</i>	Virtqueue to drop all requests of.

Definition at line 172 of file [request\\_l4virtio.h](#).

References [L4virtio::Svr::Virtqueue::desc\\_avail\(\)](#), [L4virtio::Svr::Virtqueue::finish\(\)](#), [L4\\_UNLIKELY](#), [L4virtio::Svr::Virtqueue::next\\_avail\(\)](#), [L4virtio::Virtqueue::ready\(\)](#), and [L4virtio::Svr::Request\\_processor::start\(\)](#).

Referenced by [L4virtio\\_port::drop\\_requests\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 16.470.2.2 get\_request()

```

std::optional< Virtio_net_request > Virtio_net_request::get_request (
    Virtio_net * dev,
    L4virtio::Svr::Virtqueue * queue) [inline], [static]
  
```

Construct a request from the next entry of a provided queue.

#### Parameters

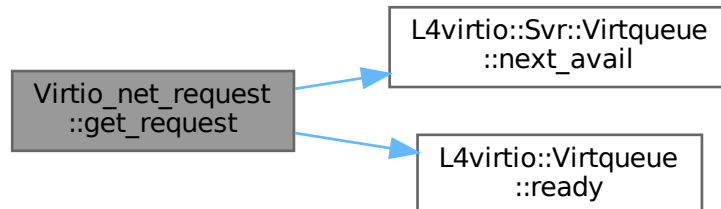
<i>dev</i>	Port of the provided virtqueue.
<i>queue</i>	Virtqueue to extract next entry from.

Definition at line 199 of file [request\\_l4virtio.h](#).

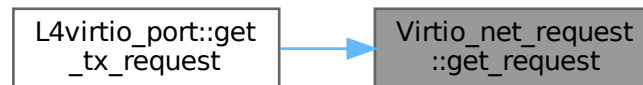
References [L4\\_UNLIKELY](#), [L4virtio::Svr::Virtqueue::next\\_avail\(\)](#), and [L4virtio::Virtqueue::ready\(\)](#).

Referenced by [L4virtio\\_port::get\\_tx\\_request\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

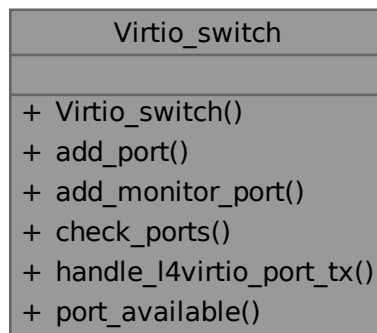
- `pkg/virtio-net-switch/server/switch/request_l4virtio.h`

## 16.471 Virtio\_switch Class Reference

The Virtio switch contains all ports and processes network requests.

```
#include <switch.h>
```

Collaboration diagram for Virtio\_switch:



### Public Member Functions

- [Virtio\\_switch](#) (unsigned max\_ports)  
*Create a switch with n ports.*
- bool [add\\_port](#) (Port\_iface \*port)  
*Add a port to the switch.*
- bool [add\\_monitor\\_port](#) (Port\_iface \*port)  
*Add a monitor port to the switch.*
- void [check\\_ports](#) ()  
*Check validity of ports.*
- bool [handle\\_l4virtio\\_port\\_tx](#) (L4virtio\_port \*port)  
*Handle TX queue of the given port.*
- int [port\\_available](#) (bool monitor)  
*Is there still a free port on this switch available?*

## 16.471.1 Detailed Description

The Virtio switch contains all ports and processes network requests.

A Port on its own is not capable to process an incoming network request because it has no knowledge about other ports. The processing of an incoming request therefore gets delegated to the switch.

The [Virtio\\_switch](#) is constructed at the start of the Virtio Net Switch application. The factory saves a reference to it to pass it to the [Kick\\_irq](#) on port creation.

Definition at line 33 of file [switch.h](#).

## 16.471.2 Constructor & Destructor Documentation

### 16.471.2.1 Virtio\_switch()

```
Virtio_switch::Virtio_switch (
    unsigned max_ports) [explicit]
```

Create a switch with n ports.

### Parameters

<i>max_ports</i>	maximal number of provided ports
------------------	----------------------------------

Definition at line 12 of file [switch.cc](#).

## 16.471.3 Member Function Documentation

### 16.471.3.1 add\_monitor\_port()

```
bool Virtio_switch::add_monitor_port (  
    Port_iface * port)
```

Add a monitor port to the switch.

#### Parameters

<i>port</i>	A pointer to an already constructed Port_iface object.
-------------	--------------------------------------------------------

#### Return values

<i>true</i>	Port was added successfully.
<i>false</i>	Switch was not able to add the port.

Definition at line 54 of file [switch.cc](#).

### 16.471.3.2 add\_port()

```
bool Virtio_switch::add_port (  
    Port_iface * port)
```

Add a port to the switch.

#### Parameters

<i>port</i>	A pointer to an already constructed Port_iface object.
-------------	--------------------------------------------------------

#### Return values

<i>true</i>	Port was added successfully.
<i>false</i>	Switch was not able to add the port.

Definition at line 29 of file [switch.cc](#).

References [Mac\\_addr::is\\_unknown\(\)](#).

Here is the call graph for this function:



### 16.471.3.3 check\_ports()

```
void Virtio_switch::check_ports ()
```

Check validity of ports.

Check whether all ports are still used and remove any unused (unreferenced) ports. Shall be invoked after an incoming cap deletion irq to remove ports without clients.

Definition at line 69 of file [switch.cc](#).

### 16.471.3.4 handle\_l4virtio\_port\_tx()

```
bool Virtio_switch::handle_l4virtio_port_tx (
    L4virtio_port * port)
```

Handle TX queue of the given port.

#### Parameters

<i>port</i>	<a href="#">L4virtio_port</a> to handle pending TX work for.
-------------	--------------------------------------------------------------

#### Return values

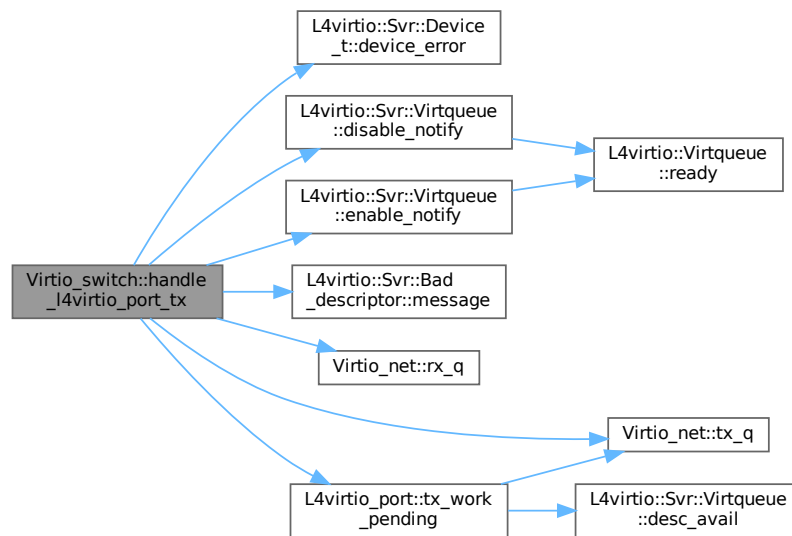
<i>false</i>	Port hit its TX burst limit, and thus a TX pending reschedule notification was queued.
<i>true</i>	Port's entire TX queue was processed.

Definition at line 194 of file [switch.cc](#).

References [L4virtio::Svr::Device\\_t< DATA >::device\\_error\(\)](#), [L4virtio::Svr::Virtqueue::disable\\_notify\(\)](#), [L4virtio::Svr::Virtqueue::enable\\_notify\(\)](#), [L4virtio::Svr::Bad\\_descriptor::message\(\)](#), [Virtio\\_net::rx\\_q\(\)](#), [Virtio\\_net::tx\\_q\(\)](#), and [L4virtio\\_port::tx\\_work\\_pending\(\)](#).



Here is the call graph for this function:



### 16.471.3.5 port\_available()

```
int Virtio_switch::port_available (
    bool monitor) [inline]
```

Is there still a free port on this switch available?

#### Parameters

<i>monitor</i>	True if we look for a monitor slot.
----------------	-------------------------------------

#### Return values

$\geq 0$	The next available port index.
-1	No port available.

Definition at line 148 of file [switch.h](#).

The documentation for this class was generated from the following files:

- `pkg/virtio-net-switch/server/switch/switch.h`
- `pkg/virtio-net-switch/server/switch/switch.cc`

## 16.472 Virtio\_vlan\_mangle Class Reference

Class for VLAN packet rewriting.

```
#include <vlan.h>
```

Collaboration diagram for Virtio\_vlan\_mangle:

Virtio_vlan_mangle
<ul style="list-style-type: none"> <li>+ Virtio_vlan_mangle()</li> <li>+ copy_pkt()</li> <li>+ rewrite_hdr()</li> <li>+ add()</li> <li>+ remove()</li> </ul>

### Public Member Functions

- [Virtio\\_vlan\\_mangle](#) ()  
*Default constructor.*
- [l4\\_uint32\\_t copy\\_pkt](#) ([Buffer](#) &dst, [Buffer](#) &src)  
*Copy packet from src to dst.*
- void [rewrite\\_hdr](#) ([Virtio\\_net::Hdr](#) \*hdr)  
*Rewrite the virtio network header.*

### Static Public Member Functions

- static constexpr [Virtio\\_vlan\\_mangle add](#) ([l4\\_uint16\\_t](#) tci)  
*Construct an object that adds a VLAN tag.*
- static constexpr [Virtio\\_vlan\\_mangle remove](#) ()  
*Construct an object that removes the VLAN tag.*

### 16.472.1 Detailed Description

Class for VLAN packet rewriting.

Definition at line 36 of file [vlan.h](#).

## 16.472.2 Constructor & Destructor Documentation

### 16.472.2.1 Virtio\_vlan\_mangle()

`Virtio_vlan_mangle::Virtio_vlan_mangle () [inline]`

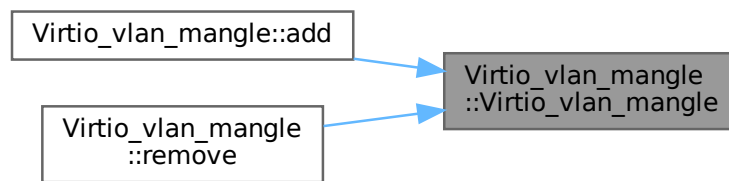
Default constructor.

The packet is not touched in any way.

Definition at line 52 of file [vlan.h](#).

Referenced by [add\(\)](#), and [remove\(\)](#).

Here is the caller graph for this function:



## 16.472.3 Member Function Documentation

### 16.472.3.1 add()

```
constexpr Virtio_vlan_mangle Virtio_vlan_mangle::add (
    14_uint16_t tci) [inline], [static], [constexpr]
```

Construct an object that adds a VLAN tag.

#### Parameters

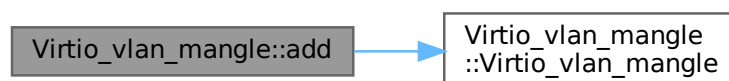
<i>tci</i>	The TCI field of the VLAN tag to add.
------------	---------------------------------------

It is the callers responsibility to ensure that the packet is not already tagged.

Definition at line 64 of file [vlan.h](#).

References [Virtio\\_vlan\\_mangle\(\)](#).

Here is the call graph for this function:



### 16.472.3.2 copy\_pkt()

```
l4_uint32_t Virtio_vlan_mangle::copy_pkt (
    Buffer & dst,
    Buffer & src) [inline]
```

Copy packet from *src* to *dst*.

#### Parameters

<i>src</i>	Source packet buffer
<i>dst</i>	Destination packet buffer

#### Returns

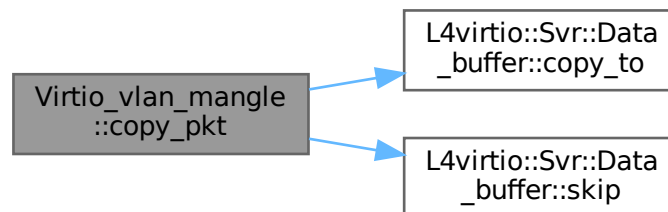
The number of bytes copied

Copy the data from *src* to *dst*, possibly rewriting parts of the packet. The method is expected to be called repeatedly until the source packet is finished. Partial copies are allowed (including reading nothing from the source buffer) as long as progress is made, i.e. repeatedly calling this function eventually consumes the source buffer.

Definition at line 93 of file [vlan.h](#).

References [L4virtio::Svr::Data\\_buffer::copy\\_to\(\)](#), [L4\\_LIKELY](#), [L4virtio::Svr::Data\\_buffer::left](#), [L4virtio::Svr::Data\\_buffer::pos](#), and [L4virtio::Svr::Data\\_buffer::skip\(\)](#).

Here is the call graph for this function:



### 16.472.3.3 remove()

```
constexpr Virtio_vlan_mangle Virtio_vlan_mangle::remove () [inline], [static], [constexpr]
```

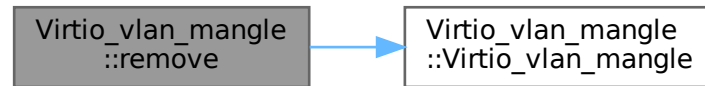
Construct an object that removes the VLAN tag.

This object assumes that the Ethernet packet has a VLAN tag and will slavishly remove the necessary bytes from the packet.

Definition at line 75 of file [vlan.h](#).

References [Virtio\\_vlan\\_mangle\(\)](#).

Here is the call graph for this function:



#### 16.472.3.4 `rewrite_hdr()`

```
void Virtio_vlan_mangle::rewrite_hdr (  
    Virtio_net::Hdr * hdr) [inline]
```

Rewrite the virtio network header.

##### Parameters

<i>hdr</i>	The virtio header of the packet
------------	---------------------------------

This method is called exactly once for every virtio network packet. Any necessary changes to the header are done in-place.

Definition at line 142 of file [vlan.h](#).

References [L4\\_UNLIKELY](#).

The documentation for this class was generated from the following file:

- `pkg/virtio-net-switch/server/switch/vlan.h`



# Chapter 17

## File Documentation

### 17.1 asm\_access.h

```
00001 /*
00002  * Copyright (C) 2021, 2024 Kernkonzept GmbH.
00003  * Author(s): Jakub Jermar <jakub.jermar@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <l4/sys/l4int.h>
00011 #include <x86/l4/drivers/asm_access.h>
00012
00013 namespace Asm_access {
00014
00015     inline
00016     l4_uint64_t
00017     read(l4_uint64_t const *mem)
00018     {
00019         l4_uint64_t val;
00020
00021         asm volatile ("movq %[mem], %[val]" : [val] "=r" (val) : [mem] "m" (*mem));
00022
00023         return val;
00024     }
00025
00026     inline
00027     void
00028     write(l4_uint64_t val, l4_uint64_t *mem)
00029     {
00030         asm volatile ("movq %[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00031     }
00032 }
00033
```

### 17.2 asm\_access.h

```
00001 /*
00002  * Copyright (C) 2021, 2024 Kernkonzept GmbH.
00003  * Author(s): Jakub Jermar <jakub.jermar@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <l4/sys/l4int.h>
00011
00012 namespace Asm_access {
00013
00014     inline
00015     l4_uint8_t
00016     read(l4_uint8_t const *mem)
00017     {
```

```

00018     14_uint8_t val;
00019
00020     asm volatile ("ldrb %[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00021
00022     return val;
00023 }
00024
00025 inline
00026 14_uint16_t
00027 read(14_uint16_t const *mem)
00028 {
00029     14_uint16_t val;
00030
00031     asm volatile ("ldrh %[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00032
00033     return val;
00034 }
00035
00036 inline
00037 14_uint32_t
00038 read(14_uint32_t const *mem)
00039 {
00040     14_uint32_t val;
00041
00042     asm volatile ("ldr %[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00043
00044     return val;
00045 }
00046
00047 inline
00048 void
00049 write(14_uint8_t val, 14_uint8_t *mem)
00050 {
00051     asm volatile ("strb %[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00052 }
00053
00054 inline
00055 void
00056 write(14_uint16_t val, 14_uint16_t *mem)
00057 {
00058     asm volatile ("strh %[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00059 }
00060
00061 inline
00062 void
00063 write(14_uint32_t val, 14_uint32_t *mem)
00064 {
00065     asm volatile ("str %[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00066 }
00067
00068 }

```

## 17.3 asm\_access.h

```

00001 /*
00002  * Copyright (C) 2021, 2024 Kernkonzept GmbH.
00003  * Author(s): Jakub Jermar <jakub.jermar@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <14/sys/l4int.h>
00011
00012 namespace Asm_access {
00013
00014 inline
00015 14_uint8_t
00016 read(14_uint8_t const *mem)
00017 {
00018     14_uint8_t val;
00019
00020     asm volatile ("ldrb %w[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00021
00022     return val;
00023 }
00024
00025 inline
00026 14_uint16_t
00027 read(14_uint16_t const *mem)
00028 {
00029     14_uint16_t val;

```



```

00030
00031     asm volatile ("ldrh %w[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00032
00033     return val;
00034 }
00035
00036 inline
00037 l4_uint32_t
00038 read(l4_uint32_t const *mem)
00039 {
00040     l4_uint32_t val;
00041
00042     asm volatile ("ldr %w[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00043
00044     return val;
00045 }
00046
00047 inline
00048 l4_uint64_t
00049 read(l4_uint64_t const *mem)
00050 {
00051     l4_uint64_t val;
00052
00053     asm volatile ("ldr %[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00054
00055     return val;
00056 }
00057
00058 inline
00059 void
00060 write(l4_uint8_t val, l4_uint8_t *mem)
00061 {
00062     asm volatile ("strb %w[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00063 }
00064
00065 inline
00066 void
00067 write(l4_uint16_t val, l4_uint16_t *mem)
00068 {
00069     asm volatile ("strh %w[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00070 }
00071
00072 inline
00073 void
00074 write(l4_uint32_t val, l4_uint32_t *mem)
00075 {
00076     asm volatile ("str %w[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00077 }
00078
00079 inline
00080 void
00081 write(l4_uint64_t val, l4_uint64_t *mem)
00082 {
00083     asm volatile ("str %[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00084 }
00085
00086 }

```

## 17.4 asm\_access.h

```

00001 /*
00002  * Copyright (C) 2021, 2024 Kernkonzept GmbH.
00003  * Author(s): Jakub Jermar <jakub.jermar@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <l4/drivers/asm_access_gen.h>

```

## 17.5 asm\_access.h

```

00001 /*
00002  * Copyright (C) 2021, 2024 Kernkonzept GmbH.
00003  * Author(s): Jakub Jermar <jakub.jermar@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)

```

```

00006  */
00007
00008 #pragma once
00009
00010 #include <l4/drivers/asm_access_gen.h>

```

## 17.6 asm\_access.h

```

00001 /*
00002  * Copyright (C) 2021, 2024 Kernkonzept GmbH.
00003  * Author(s): Georg Kotheimer <georg.kotheimer@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <l4/sys/l4int.h>
00011
00012 namespace Asm_access {
00013
00014 inline
00015 l4_uint8_t
00016 read(l4_uint8_t const *mem)
00017 {
00018     l4_uint8_t val;
00019
00020     asm volatile ("lb %[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00021
00022     return val;
00023 }
00024
00025 inline
00026 l4_uint16_t
00027 read(l4_uint16_t const *mem)
00028 {
00029     l4_uint16_t val;
00030
00031     asm volatile ("lh %[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00032
00033     return val;
00034 }
00035
00036 inline
00037 l4_uint32_t
00038 read(l4_uint32_t const *mem)
00039 {
00040     l4_uint32_t val;
00041
00042     asm volatile ("lw %[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00043
00044     return val;
00045 }
00046
00047 inline
00048 void
00049 write(l4_uint8_t val, l4_uint8_t *mem)
00050 {
00051     asm volatile ("sb %[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00052 }
00053
00054 inline
00055 void
00056 write(l4_uint16_t val, l4_uint16_t *mem)
00057 {
00058     asm volatile ("sh %[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00059 }
00060
00061 inline
00062 void
00063 write(l4_uint32_t val, l4_uint32_t *mem)
00064 {
00065     asm volatile ("sw %[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00066 }
00067
00068 #if __riscv_xlen == 64
00069
00070 inline
00071 l4_uint64_t
00072 read(l4_uint64_t const *mem)
00073 {
00074     l4_uint64_t val;
00075

```

```

00076     asm volatile ("ld %[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00077
00078     return val;
00079 }
00080
00081 inline
00082 void
00083 write(l4_uint64_t val, l4_uint64_t *mem)
00084 {
00085     asm volatile ("sd %[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00086 }
00087
00088 #endif
00089
00090 }

```

## 17.7 asm\_access.h

```

00001 /*
00002  * Copyright (C) 2021, 2024 Kernkonzept GmbH.
00003  * Author(s): Jakub Jermar <jakub.jermar@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <l4/drivers/asm_access_gen.h>

```

## 17.8 asm\_access.h

```

00001 /*
00002  * Copyright (C) 2021, 2024 Kernkonzept GmbH.
00003  * Author(s): Jakub Jermar <jakub.jermar@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <l4/sys/l4int.h>
00011
00012 namespace Asm_access {
00013
00014     inline
00015     l4_uint8_t
00016     read(l4_uint8_t const *mem)
00017     {
00018         l4_uint8_t val;
00019
00020         asm volatile ("movb %[mem], %[val]" : [val] "=q" (val) : [mem] "m" (*mem));
00021
00022         return val;
00023     }
00024
00025     inline
00026     l4_uint16_t
00027     read(l4_uint16_t const *mem)
00028     {
00029         l4_uint16_t val;
00030
00031         asm volatile ("movw %[mem], %[val]" : [val] "=r" (val) : [mem] "m" (*mem));
00032
00033         return val;
00034     }
00035
00036     inline
00037     l4_uint32_t
00038     read(l4_uint32_t const *mem)
00039     {
00040         l4_uint32_t val;
00041
00042         asm volatile ("movl %[mem], %[val]" : [val] "=r" (val) : [mem] "m" (*mem));
00043
00044         return val;
00045     }
00046
00047     inline

```

```

00048 void
00049 write(l4_uint8_t val, l4_uint8_t *mem)
00050 {
00051     asm volatile ("movb %[val], %[mem]" : [mem] "=m" (*mem) : [val] "qi" (val));
00052 }
00053
00054 inline
00055 void
00056 write(l4_uint16_t val, l4_uint16_t *mem)
00057 {
00058     asm volatile ("movw %[val], %[mem]" : [mem] "=m" (*mem) : [val] "ri" (val));
00059 }
00060
00061 inline
00062 void
00063 write(l4_uint32_t val, l4_uint32_t *mem)
00064 {
00065     asm volatile ("movl %[val], %[mem]" : [mem] "=m" (*mem) : [val] "ri" (val));
00066 }
00067
00068 }

```

## 17.9 asm\_access\_gen.h

```

00001 /*
00002  * Copyright (C) 2021, 2024 Kernkonzept GmbH.
00003  * Author(s): Jakub Jermar <jakub.jermar@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/sys/l4int.h>
00010 #include <l4/cxx/type_traits>
00011
00012 namespace Asm_access {
00013
00014     template <typename T>
00015     struct is_supported_type
00016     {
00017         static const bool value = cxx::is_same<T, l4_uint8_t>::value
00018             || cxx::is_same<T, l4_uint16_t>::value
00019             || cxx::is_same<T, l4_uint32_t>::value
00020             || cxx::is_same<T, l4_uint64_t>::value;
00021     };
00022
00023     template <typename T>
00024     inline
00025     typename cxx::enable_if<is_supported_type<T>::value, T>::type
00026     read(T const *mem)
00027     {
00028         return *reinterpret_cast<volatile T const *>(mem);
00029     }
00030
00031     template <typename T>
00032     inline
00033     typename cxx::enable_if<is_supported_type<T>::value, void>::type
00034     write(T val, T *mem)
00035     {
00036         *reinterpret_cast<volatile T *>(mem) = val;
00037     }
00038
00039 }

```

## 17.10 hw\_mmio\_register\_block

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2014-2021, 2023-2024 Kernkonzept GmbH.
00004  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00005  *             Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include <l4/drivers/hw_register_block>
00012 #include <l4/drivers/asm_access.h>

```

```

00013
00014 namespace L4drivers {
00015
00016 class Mmio_register_block_base
00017 {
00018 protected:
00019     l4_addr_t _base;
00020     l4_addr_t _shift;
00021
00022 public:
00023     explicit Mmio_register_block_base(l4_addr_t base = 0, l4_addr_t shift = 0)
00024         : _base(base), _shift(shift) {}
00025
00026     template< typename T >
00027     T read(l4_addr_t reg) const
00028     { return Asm_access::read(reinterpret_cast<T const *>(_base + (reg « _shift))); }
00029
00030     template< typename T >
00031     void write(T value, l4_addr_t reg) const
00032     { Asm_access::write(value, reinterpret_cast<T *>(_base + (reg « _shift))); }
00033
00034     void set_base(l4_addr_t base) { _base = base; }
00035     void set_shift(l4_addr_t shift) { _shift = shift; }
00036 };
00037
00042 template< unsigned MAX_BITS = 32 >
00043 struct Mmio_register_block
00044 : Register_block_impl<Mmio_register_block<MAX_BITS>, MAX_BITS>,
00045   Mmio_register_block_base
00046 {
00047     explicit Mmio_register_block(l4_addr_t base = 0, l4_addr_t shift = 0)
00048         : Mmio_register_block_base(base, shift) {}
00049 };
00050
00051 }

```

## 17.11 hw\_register\_block

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2014-2021, 2023-2024 Kernkonzept GmbH.
00004  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <l4/sys/types.h>
00011 #include <l4/cxx/type_traits>
00012
00013 namespace L4drivers {
00014
00015
00062
00063
00071 template< unsigned MAX_BITS = 32 >
00072 struct Register_block_base;
00073
00074 template<>
00075 struct Register_block_base<8>
00076 {
00077     virtual l4_uint8_t do_read_8(l4_addr_t reg) const = 0;
00078     virtual void do_write_8(l4_uint8_t value, l4_addr_t reg) = 0;
00079     virtual ~Register_block_base() = 0;
00080 };
00081
00082 inline Register_block_base<8>::~Register_block_base() {}
00083
00084 template<>
00085 struct Register_block_base<16> : Register_block_base<8>
00086 {
00087     virtual l4_uint16_t do_read_16(l4_addr_t reg) const = 0;
00088     virtual void do_write_16(l4_uint16_t value, l4_addr_t reg) = 0;
00089 };
00090
00091 template<>
00092 struct Register_block_base<32> : Register_block_base<16>
00093 {
00094     virtual l4_uint32_t do_read_32(l4_addr_t reg) const = 0;
00095     virtual void do_write_32(l4_uint32_t value, l4_addr_t reg) = 0;
00096 };
00097
00098 template<>

```

```

00099 struct Register_block_base<64> : Register_block_base<32>
00100 {
00101     virtual l4_uint64_t do_read_64(l4_addr_t reg) const = 0;
00102     virtual void do_write_64(l4_uint64_t value, l4_addr_t reg) = 0;
00103 };
00104 #undef REGBLK_READ_TEMPLATE
00105 #undef REGBLK_WRITE_TEMPLATE
00106
00107 template<typename CHILd>
00108 struct Register_block_modify_mixin
00109 {
00110     template< typename T >
00111     T modify(T clear_bits, T set_bits, l4_addr_t reg) const
00112     {
00113         CHILd const *c = static_cast<CHILd const *>(this);
00114         T r = (c->template read<T>(reg) & ~clear_bits) | set_bits;
00115         c->template write<T>(r, reg);
00116         return r;
00117     }
00118
00119     template< typename T >
00120     T set(T set_bits, l4_addr_t reg) const
00121     { return this->template modify<T>(T(0), set_bits, reg); }
00122
00123     template< typename T >
00124     T clear(T clear_bits, l4_addr_t reg) const
00125     { return this->template modify<T>(clear_bits, T(0), reg); }
00126 };
00127
00128 #define REGBLK_READ_TEMPLATE(sZ) \
00129     template< typename T > \
00130     typename cxx::enable_if<sizeof(T) == (sZ / 8), T>::type read(l4_addr_t reg) const \
00131     { \
00132         union X { T t; l4_uint##sZ##_t v; } m; \
00133         m.v = _b->do_read_##sZ (reg); \
00134         return m.t; \
00135     }
00136
00137 #define REGBLK_WRITE_TEMPLATE(sZ) \
00138     template< typename T > \
00139     void write(T value, l4_addr_t reg, typename cxx::enable_if<sizeof(T) == (sZ / 8), T>::type = T()) \
00140     const \
00141     { \
00142         union X { T t; l4_uint##sZ##_t v; } m; \
00143         m.t = value; \
00144         _b->do_write_##sZ(m.v, reg); \
00145     }
00146
00155 template< typename BLOCK >
00156 class Register_block_tmpl
00157 : public Register_block_modify_mixin<Register_block_tmpl<BLOCK> >
00158 {
00159 private:
00160     BLOCK *_b;
00161
00162 public:
00163     Register_block_tmpl(BLOCK *blk) : _b(blk) {}
00164     Register_block_tmpl() = default;
00165
00166     operator BLOCK * () const { return _b; }
00167
00168     REGBLK_READ_TEMPLATE(8)
00169     REGBLK_WRITE_TEMPLATE(8)
00170     REGBLK_READ_TEMPLATE(16)
00171     REGBLK_WRITE_TEMPLATE(16)
00172     REGBLK_READ_TEMPLATE(32)
00173     REGBLK_WRITE_TEMPLATE(32)
00174     REGBLK_READ_TEMPLATE(64)
00175     REGBLK_WRITE_TEMPLATE(64)
00176 };
00177
00178 #undef REGBLK_READ_TEMPLATE
00179 #undef REGBLK_WRITE_TEMPLATE
00180
00181 namespace __Type_helper {
00182     template<unsigned> struct Unsigned;
00183     template<> struct Unsigned<8> { typedef l4_uint8_t type; };
00184     template<> struct Unsigned<16> { typedef l4_uint16_t type; };
00185     template<> struct Unsigned<32> { typedef l4_uint32_t type; };
00186     template<> struct Unsigned<64> { typedef l4_uint64_t type; };
00187 }
00188
00189
00190 template< unsigned BITS, typename BLOCK >
00201 class Ro_register_tmpl

```

```

00202 {
00203 protected:
00204     BLOCK _b;
00205     unsigned _o;
00206
00207 public:
00208     typedef typename __Type_helper::Unsigned<BITS>::type value_type;
00209
00210     Ro_register_tmpl(BLOCK const &blk, unsigned offset) : _b(blk), _o(offset) {}
00211     Ro_register_tmpl() = default;
00212
00217     operator value_type () const
00218     { return _b.template read<value_type>(_o); }
00219
00224     value_type read() const
00225     { return _b.template read<value_type>(_o); }
00226 };
00227
00228
00236 template< unsigned BITS, typename BLOCK >
00237 class Register_tmpl : public Ro_register_tmpl<BITS, BLOCK>
00238 {
00239 public:
00240     typedef typename Ro_register_tmpl<BITS, BLOCK>::value_type value_type;
00241
00242     Register_tmpl(BLOCK const &blk, unsigned offset)
00243     : Ro_register_tmpl<BITS, BLOCK>(blk, offset)
00244     {}
00245
00246     Register_tmpl() = default;
00247
00252     Register_tmpl &operator = (value_type val)
00253     { this->_b.template write<value_type>(val, this->_o); return *this; }
00254
00259     void write(value_type val)
00260     { this->_b.template write<value_type>(val, this->_o); }
00261
00274     value_type set(value_type set_bits)
00275     { return this->_b.template set<value_type>(set_bits, this->_o); }
00276
00290     value_type clear(value_type clear_bits)
00291     { return this->_b.template clear<value_type>(clear_bits, this->_o); }
00292
00308     value_type modify(value_type clear_bits, value_type set_bits)
00309     { return this->_b.template modify<value_type>(clear_bits, set_bits, this->_o); }
00310 };
00311
00312
00324 template<
00325     unsigned MAX_BITS,
00326     typename BLOCK = Register_block_tmpl<
00327         Register_block_base<MAX_BITS>
00328     >
00329 >
00330 class Register_block
00331 {
00332 private:
00333     template< unsigned B, typename BLK > friend class Register_block;
00334     template< unsigned B, typename BLK > friend class Ro_register_block;
00335     typedef BLOCK Block;
00336     Block _b;
00337
00338 public:
00339     Register_block() = default;
00340     Register_block(Block const &blk) : _b(blk) {}
00341     Register_block &operator = (Block const &blk)
00342     { _b = blk; return *this; }
00343
00344     template< unsigned BITS >
00345     Register_block(Register_block<BITS> blk) : _b(blk._b) {}
00346
00347     typedef Register_tmpl<MAX_BITS, Block> Register;
00348     typedef Ro_register_tmpl<MAX_BITS, Block> Ro_register;
00349
00356     template< unsigned BITS >
00357     Ro_register_tmpl<BITS, Block> r(unsigned offset) const
00358     { return Ro_register_tmpl<BITS, Block>(this->_b, offset); }
00359
00365     Ro_register operator [] (unsigned offset) const
00366     { return this->r<MAX_BITS>(offset); }
00367
00368
00375     template< unsigned BITS >
00376     Register_tmpl<BITS, Block> r(unsigned offset)
00377     { return Register_tmpl<BITS, Block>(this->_b, offset); }
00378
00385     Register operator [] (unsigned offset)

```

```

00386 { return this->r<MAX_BITS>(offset); }
00387 };
00388
00398 template<
00399     unsigned MAX_BITS,
00400     typename BLOCK = Register_block_tmpl<
00401         Register_block_base<MAX_BITS> const
00402     >
00403 >
00404 class Ro_register_block
00405 {
00406 private:
00407     template< unsigned B, typename BLK > friend class Ro_register_block;
00408     typedef BLOCK Block;
00409     Block _b;
00410
00411 public:
00412     Ro_register_block() = default;
00413     Ro_register_block(BLOCK const &blk) : _b(blk) {}
00414
00415     template< unsigned BITS >
00416     Ro_register_block(Register_block<BITS> const &blk) : _b(blk._b) {}
00417
00418     typedef Ro_register_tmpl<MAX_BITS, Block> Ro_register;
00419     typedef Ro_register Register;
00420
00426     Ro_register operator [] (unsigned offset) const
00427     { return Ro_register(this->_b, offset); }
00428
00435     template< unsigned BITS >
00436     Ro_register_tmpl<BITS, Block> r(unsigned offset) const
00437     { return Ro_register_tmpl<BITS, Block>(this->_b, offset); }
00438 };
00439
00440
00454 template< typename BASE, unsigned MAX_BITS = 32 >
00455 struct Register_block_impl;
00456
00457 #define REGBLK_IMPL_RW_TEMPLATE(sz, ...) \
00458     l4_uint##sz##_t do_read_##sz(l4_addr_t reg) const override \
00459     { return static_cast<BASE const *>(this)->template read<l4_uint##sz##_t>(reg); } \
00460     \
00461     void do_write_##sz(l4_uint##sz##_t value, l4_addr_t reg) override \
00462     { static_cast<BASE*>(this)->template write<l4_uint##sz##_t>(value, reg); }
00463
00464
00465 template< typename BASE >
00466 struct Register_block_impl<BASE, 8> : public Register_block_base<8>
00467 {
00468     REGBLK_IMPL_RW_TEMPLATE(8);
00469 };
00470
00471 template< typename BASE >
00472 struct Register_block_impl<BASE, 16> : public Register_block_base<16>
00473 {
00474     REGBLK_IMPL_RW_TEMPLATE(8);
00475     REGBLK_IMPL_RW_TEMPLATE(16);
00476 };
00477
00478 template< typename BASE >
00479 struct Register_block_impl<BASE, 32> : public Register_block_base<32>
00480 {
00481     REGBLK_IMPL_RW_TEMPLATE(8);
00482     REGBLK_IMPL_RW_TEMPLATE(16);
00483     REGBLK_IMPL_RW_TEMPLATE(32);
00484 };
00485
00486 template< typename BASE >
00487 struct Register_block_impl<BASE, 64> : public Register_block_base<64>
00488 {
00489     REGBLK_IMPL_RW_TEMPLATE(8);
00490     REGBLK_IMPL_RW_TEMPLATE(16);
00491     REGBLK_IMPL_RW_TEMPLATE(32);
00492     REGBLK_IMPL_RW_TEMPLATE(64);
00493 };
00494
00495 #undef REGBLK_IMPL_RW_TEMPLATE
00496
00497 }

```

## 17.12 io\_regblock.h

```
00001 /*
```



```

00002  * Copyright (C) 2012 Technische Universität Dresden.
00003  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 namespace L4
00010 {
00011     class Io_register_block
00012     {
00013     public:
00017         virtual unsigned char read8(unsigned long reg) const = 0;
00018
00022         virtual unsigned short read16(unsigned long reg) const = 0;
00023
00027         virtual unsigned int read32(unsigned long reg) const = 0;
00028
00029         /*
00030          * \brief Read register with an 8 byte access.
00031          */
00032         //virtual unsigned long long read64(unsigned long reg) const = 0;
00033
00037         virtual void write8(unsigned long reg, unsigned char value) const = 0;
00038
00042         virtual void write16(unsigned long reg, unsigned short value) const = 0;
00043
00047         virtual void write32(unsigned long reg, unsigned int value) const = 0;
00048
00049         /*
00050          * \brief Write register with an 8 byte access.
00051          */
00052         //virtual void write64(unsigned long reg, unsigned long long value) const = 0;
00053
00057         virtual unsigned long addr(unsigned long reg) const = 0;
00058
00064         virtual void delay() const = 0;
00065
00066         virtual ~Io_register_block() = 0;
00067
00075         template< typename R >
00076         R read(unsigned long reg) const
00077         {
00078             static_assert(sizeof(R) == 1 || sizeof(R) == 2 || sizeof(R) == 4,
00079                 "Invalid size");
00080
00081             switch (sizeof(R))
00082             {
00083                 case 1: return read8(reg);
00084                 case 2: return read16(reg);
00085                 case 4: return read32(reg);
00086             };
00087         }
00088
00096         template< typename R >
00097         void write(unsigned long reg, R value) const
00098         {
00099             static_assert(sizeof(R) == 1 || sizeof(R) == 2 || sizeof(R) == 4,
00100                 "Invalid size");
00101
00102             switch (sizeof(R))
00103             {
00104                 case 1: write8(reg, value); return;
00105                 case 2: write16(reg, value); return;
00106                 case 4: write32(reg, value); return;
00107             };
00108         }
00109
00120         template< typename R >
00121         R modify(unsigned long reg, R clear_bits, R set_bits) const
00122         {
00123             R r = (read<R>(reg) & ~clear_bits) | set_bits;
00124             write(reg, r);
00125             return r;
00126         }
00127
00134         template< typename R >
00135         R set(unsigned long reg, R set_bits) const
00136         {
00137             return modify<R>(reg, 0, set_bits);
00138         }
00139
00146         template< typename R >
00147         R clear(unsigned long reg, R clear_bits) const
00148         {
00149             return modify<R>(reg, clear_bits, 0);
00150         }

```

```

00151
00152     };
00153
00154     inline Io_register_block::~Io_register_block() {}
00155
00156
00157     class Io_register_block_mmio : public Io_register_block
00158     {
00159     private:
00160         template< typename R >
00161         R _read(unsigned long reg) const
00162         { return *reinterpret_cast<volatile R *>(_base + (reg « _shift)); }
00163
00164         template< typename R >
00165         void _write(unsigned long reg, R val) const
00166         { *reinterpret_cast<volatile R *>(_base + (reg « _shift)) = val; }
00167
00168     public:
00169         Io_register_block_mmio(unsigned long base, unsigned char shift = 0)
00170         : _base(base), _shift(shift)
00171         {}
00172
00173         unsigned long addr(unsigned long reg) const override
00174         { return _base + (reg « _shift); }
00175
00176         unsigned char read8(unsigned long reg) const override
00177         { return _read<unsigned char>(reg); }
00178         unsigned short read16(unsigned long reg) const override
00179         { return _read<unsigned short>(reg); }
00180         unsigned int read32(unsigned long reg) const override
00181         { return _read<unsigned int>(reg); }
00182
00183         void write8(unsigned long reg, unsigned char val) const override
00184         { _write(reg, val); }
00185         void write16(unsigned long reg, unsigned short val) const override
00186         { _write(reg, val); }
00187         void write32(unsigned long reg, unsigned int val) const override
00188         { _write(reg, val); }
00189
00190         void delay() const override
00191         {}
00192
00193     private:
00194         unsigned long _base;
00195         unsigned char _shift;
00196     };
00197
00198     template<typename ACCESS_TYPE>
00199     class Io_register_block_mmio_fixed_width : public Io_register_block
00200     {
00201     private:
00202         template< typename R >
00203         R _read(unsigned long reg) const
00204         { return *reinterpret_cast<volatile ACCESS_TYPE *>(_base + (reg « _shift)); }
00205
00206         template< typename R >
00207         void _write(unsigned long reg, R val) const
00208         { *reinterpret_cast<volatile ACCESS_TYPE *>(_base + (reg « _shift)) = val; }
00209
00210     public:
00211         Io_register_block_mmio_fixed_width(unsigned long base, unsigned char shift = 0)
00212         : _base(base), _shift(shift)
00213         {}
00214
00215         unsigned long addr(unsigned long reg) const
00216         { return _base + (reg « _shift); }
00217
00218         unsigned char read8(unsigned long reg) const override
00219         { return _read<unsigned char>(reg); }
00220         unsigned short read16(unsigned long reg) const override
00221         { return _read<unsigned short>(reg); }
00222         unsigned int read32(unsigned long reg) const override
00223         { return _read<unsigned int>(reg); }
00224
00225         void write8(unsigned long reg, unsigned char val) const override
00226         { _write(reg, val); }
00227         void write16(unsigned long reg, unsigned short val) const override
00228         { _write(reg, val); }
00229         void write32(unsigned long reg, unsigned int val) const override
00230         { _write(reg, val); }
00231
00232         void delay() const override
00233         {}
00234
00235     private:
00236         unsigned long _base;
00237         unsigned char _shift;

```

```
00238     };
00239 }
```

## 17.13 io\_regblock\_port.h

```
00001 /*
00002  * Copyright (C) 2012 Technische Universität Dresden.
00003  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "io_regblock.h"
00010
00011 namespace L4
00012 {
00013     class Io_register_block_port : public Io_register_block
00014     {
00015     public:
00016         Io_register_block_port(unsigned long base)
00017             : _base(base)
00018         {}
00019
00020         unsigned long addr(unsigned long reg) const override { return _base + reg; }
00021
00022         unsigned char read8(unsigned long reg) const override
00023         {
00024             unsigned char val;
00025             asm volatile("inb %w1, %b0" : "=a" (val) : "Nd" (_base + reg));
00026             return val;
00027         }
00028
00029         unsigned short read16(unsigned long reg) const override
00030         {
00031             unsigned short val;
00032             asm volatile("inw %w1, %w0" : "=a" (val) : "Nd" (_base + reg));
00033             return val;
00034         }
00035
00036         unsigned int read32(unsigned long reg) const override
00037         {
00038             unsigned int val;
00039             asm volatile("in %w1, %0" : "=a" (val) : "Nd" (_base + reg));
00040             return val;
00041         }
00042
00043         void write8(unsigned long reg, unsigned char val) const override
00044         { asm volatile("outb %b0, %w1" : : "a" (val), "Nd" (_base + reg)); }
00045
00046         void write16(unsigned long reg, unsigned short val) const override
00047         { asm volatile("outw %w0, %w1" : : "a" (val), "Nd" (_base + reg)); }
00048
00049         void write32(unsigned long reg, unsigned int val) const override
00050         { asm volatile("out %0, %w1" : : "a" (val), "Nd" (_base + reg)); }
00051
00052         void delay() const override
00053         { asm volatile ("outb %al, $0x80"); }
00054
00055     private:
00056         unsigned long _base;
00057     };
00058 }
```

## 17.14 poll\_timeout\_counter.h

```
00001 /*
00002  * Copyright (C) 2012 Technische Universität Dresden.
00003  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 namespace L4 {
00010
00034 class Poll_timeout_counter
00035 {
```

```

00036 public:
00042 Poll_timeout_counter(unsigned counter_val)
00043 {
00044     set(counter_val);
00045 }
00046
00053 void set(unsigned counter_val)
00054 {
00055     _c = counter_val;
00056 }
00057
00061 bool test(bool expression = true)
00062 {
00063     if (!expression)
00064         return false;
00065
00066     if (_c)
00067     {
00068         --_c;
00069         return true;
00070     }
00071
00072     return false;
00073 }
00074
00081 bool timed_out() const { return _c == 0; }
00082
00083 private:
00084     unsigned _c;
00085 };
00086
00087 }

```

## 17.15 uart\_16550.h

```

00001 /*
00002  * Copyright (C) 2008-2021 Technische Universität Dresden.
00003  * Copyright (C) 2023-2024 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00005  *             Alexander Warg <alexander.warg@os.inf.tu-dresden.de>
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include "uart_base.h"
00012
00013 namespace L4 {
00014
00015 class Uart_16550 : public Uart
00016 {
00017 protected:
00018     enum Registers
00019     {
00020         TRB      = 0x00, // Transmit/Receive Buffer (read/write)
00021         BRD_LOW  = 0x00, // Baud Rate Divisor LSB if bit 7 of LCR is set (read/write)
00022         IER      = 0x01, // Interrupt Enable Register (read/write)
00023         BRD_HIGH = 0x01, // Baud Rate Divisor MSB if bit 7 of LCR is set (read/write)
00024         IIR      = 0x02, // Interrupt Identification Register (read only)
00025         FCR      = 0x02, // 16550 FIFO Control Register (write only)
00026         LCR      = 0x03, // Line Control Register (read/write)
00027         MCR      = 0x04, // Modem Control Register (read/write)
00028         LSR      = 0x05, // Line Status Register (read only)
00029         MSR      = 0x06, // Modem Status Register (read only)
00030         SPR      = 0x07, // Scratch Pad Register (read/write)
00031     };
00032
00033     enum Register_value_iir
00034     {
00035         IIR_BUSY = 7,
00036     };
00037
00038     enum Register_value_lsr
00039     {
00040         LSR_DR      = 0x01, // Receiver data ready
00041         LSR_THRE     = 0x20, // Transmit hold register empty
00042         LSR_TSRE     = 0x40, // Transmitter empty
00043     };
00044
00045     enum Init_values
00046     {
00047 #ifndef UART_16550_INIT_MCR
00048         Init_mcr = UART_16550_INIT_MCR,

```

```

00049 #else
00050     Init_mcr = 0,
00051 #endif
00052 #ifdef UART_16550_INIT_IER
00053     Init_ier = UART_16550_INIT_IER,
00054 #else
00055     Init_ier = 0,
00056 #endif
00057 #ifdef UART_16550_INIT_FCR
00058     Init_fcr = UART_16550_INIT_FCR,
00059 #else
00060     Init_fcr = 0,
00061 #endif
00062 };
00063
00064 public:
00065     enum
00066     {
00067         PAR_NONE = 0x00,
00068         PAR_EVEN = 0x18,
00069         PAR_ODD = 0x08,
00070         DAT_5 = 0x00,
00071         DAT_6 = 0x01,
00072         DAT_7 = 0x02,
00073         DAT_8 = 0x03,
00074         STOP_1 = 0x00,
00075         STOP_2 = 0x04,
00076
00077         MODE_8N1 = PAR_NONE | DAT_8 | STOP_1,
00078         MODE_7E1 = PAR_EVEN | DAT_7 | STOP_1,
00079
00080         // these two values are to leave either mode
00081         // or baud rate unchanged on a call to change_mode
00082         MODE_NC = 0x1000000,
00083         BAUD_NC = 0x1000000,
00084
00085         Base_rate_x86 = 115200,
00086         Base_rate_pxa = 921600,
00087     };
00088
00089     explicit Uart_16550(unsigned long base_rate, unsigned char init_flags = 0,
00090         unsigned char ier_bits = Init_ier,
00091         unsigned char mcr_bits = Init_mcr,
00092         unsigned char fcr_bits = Init_fcr)
00093 : _base_rate(base_rate), _init_flags(init_flags), _mcr_bits(mcr_bits),
00094   _ier_bits(ier_bits), _fcr_bits(fcr_bits)
00095 {}
00096
00097     bool startup(Io_register_block const *regs) override;
00098     void shutdown() override;
00099     bool change_mode(Transfer_mode m, Baud_rate r) override;
00100     int tx_avail() const;
00101     void wait_tx_done() const;
00102     inline void out_char(char c) const;
00103     int write(char const *s, unsigned long count,
00104         bool blocking = true) const override;
00105
00106     bool enable_rx_irq(bool enable = true) override;
00107     int char_avail() const override;
00108     int get_char(bool blocking = true) const override;
00109
00110 private:
00111     unsigned long _base_rate;
00112     unsigned char _init_flags;
00113     unsigned char _mcr_bits;
00114     unsigned char _ier_bits;
00115     unsigned char _fcr_bits;
00116 };
00117
00118 } // namespace L4

```

## 17.16 uart\_16550\_dw.h

```

00001 /*
00002  * Copyright (C) 2015 Technische Universität Dresden.
00003  * Copyright (C) 2023-2024 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@l4re.org>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "uart_16550.h"

```

```

00011
00012 namespace L4 {
00013
00014 class Uart_16550_dw : public Uart_16550
00015 {
00016 public:
00017     explicit Uart_16550_dw(unsigned long base_rate)
00018         : Uart_16550(base_rate)
00019     {}
00020
00021     void irq_ack() override;
00022 };
00023
00024 } // namespace L4

```

## 17.17 uart\_apb.h

```

00001 /*
00002  * Copyright (C) 2017, 2019, 2023-2024 Kernkonzept GmbH.
00003  * Author(s): Georg Kotheimer <georg.kotheimer@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "uart_base.h"
00010
00011 namespace L4 {
00012
00017 class Uart_apb : public Uart
00018 {
00019 public:
00021     Uart_apb(unsigned freq) : _freq(freq) {}
00022     bool startup(Io_register_block const *) override;
00023     void shutdown() override;
00024     bool change_mode(Transfer_mode m, Baud_rate r) override;
00025     int tx_avail() const;
00026     void wait_tx_done() const;
00027     inline void out_char(char c) const;
00028     int write(char const *s, unsigned long count,
00029              bool blocking = true) const override;
00030
00031     bool enable_rx_irq(bool enable) override;
00032     int char_avail() const override;
00033     int get_char(bool blocking = true) const override;
00034
00035 private:
00036     void set_rate(Baud_rate r);
00037     unsigned _freq;
00038 };
00039
00040 } // namespace L4

```

## 17.18 uart\_base.h

```

00001 /*
00002  * Copyright (C) 2009-2012 Technische Universität Dresden.
00003  * Copyright (C) 2023-2024 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <stddef.h>
00011 #include <l4/drivers/io_regblock.h>
00012
00013 #include "poll_timeout_counter.h"
00014
00015 namespace L4 {
00016
00020 class Uart
00021 {
00022 protected:
00023     unsigned _mode;
00024     unsigned _rate;
00025     Io_register_block const *_regs;
00026

```

```

00027 public:
00028     void *operator new (size_t, void* a)
00029     { return a; }
00030
00031 public:
00032     typedef unsigned Transfer_mode;
00033     typedef unsigned Baud_rate;
00034
00035     Uart()
00036     : _mode(~0U), _rate(~0U)
00037     {}
00038
00046     virtual bool startup(Io_register_block const *regs) = 0;
00047
00048     virtual ~Uart() {}
00049
00053     virtual void shutdown() = 0;
00054
00066     virtual bool change_mode(Transfer_mode m, Baud_rate r) = 0;
00067
00078     virtual int write(char const *s, unsigned long count,
00079                      bool blocking = true) const = 0;
00080
00086     Transfer_mode mode() const { return _mode; }
00087
00093     Baud_rate rate() const { return _rate; }
00094
00095
00103     virtual bool enable_rx_irq(bool = true) { return false; }
00104
00108     virtual void irq_ack() {}
00109
00116     virtual int char_avail() const = 0;
00117
00126     virtual int get_char(bool blocking = true) const = 0;
00127
00128 protected:
00140     template <typename Uart_driver, bool Timeout_guard = true>
00141     int generic_write(char const *s, unsigned long count,
00142                      bool blocking = true) const
00143     {
00144         auto *self = static_cast<Uart_driver const*>(this);
00145
00146         unsigned long c;
00147         for (c = 0; c < count; ++c)
00148         {
00149             if (!blocking && !self->tx_avail())
00150                 break;
00151
00152             if constexpr (Timeout_guard)
00153             {
00154                 Poll_timeout_counter i(3000000);
00155                 while (i.test(!self->tx_avail()))
00156                     ;
00157             }
00158             else
00159             {
00160                 while (!self->tx_avail())
00161                     ;
00162             }
00163
00164             self->out_char(*s++);
00165         }
00166
00167         if (blocking)
00168             self->wait_tx_done();
00169
00170         return c;
00171     }
00172 };
00173
00174 } // namespace L4

```

## 17.19 uart\_cadence.h

```

00001 /*
00002  * Copyright (C) 2013 Technische Universität Dresden.
00003  * Copyright (C) 2023-2024 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once

```

```

00009
00010 #include "uart_base.h"
00011
00012 namespace L4 {
00013
00014 class Uart_cadence : public Uart
00015 {
00016 public:
00017     explicit Uart_cadence(unsigned base_rate) : _base_rate(base_rate) {}
00018     bool startup(Io_register_block const *) override;
00019     void shutdown() override;
00020     bool change_mode(Transfer_mode m, Baud_rate r) override;
00021     int tx_avail() const;
00022     void wait_tx_done() const {}
00023     inline void out_char(char c) const;
00024     int write(char const *s, unsigned long count,
00025              bool blocking = true) const override;
00026
00027     bool enable_rx_irq(bool) override;
00028     void irq_ack() override;
00029     int char_avail() const override;
00030     int get_char(bool blocking = true) const override;
00031
00032 private:
00033     unsigned _base_rate;
00034 };
00035
00036 } // namespace L4

```

## 17.20 uart\_dcc-v6.h

```

00001 /*
00002  * Copyright (C) 2009 Technische Universität Dresden.
00003  * Copyright (C) 2023-2024 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "uart_base.h"
00011
00012 namespace L4 {
00013
00014 class Uart_dcc_v6 : public Uart
00015 {
00016 public:
00017     explicit Uart_dcc_v6() {}
00018     explicit Uart_dcc_v6(unsigned /*base_rate*/) {}
00019     bool startup(Io_register_block const *) override;
00020     void shutdown() override;
00021     bool change_mode(Transfer_mode m, Baud_rate r) override;
00022     int tx_avail() const;
00023     void wait_tx_done() const;
00024     inline void out_char(char c) const;
00025     int write(char const *s, unsigned long count,
00026              bool blocking = true) const override;
00027
00028     int char_avail() const override;
00029     int get_char(bool blocking = true) const override;
00030
00031 private:
00032     unsigned get_status() const;
00033 };
00034
00035 } // namespace L4

```

## 17.21 uart\_dm.h

```

00001 /*
00002  * Copyright (C) 2021-2022 Stephan Gerhold <stephan@gerhold.net>
00003  * Copyright (C) 2022-2024 Kernkonzept GmbH.
00004  * Author(s): Stephan Gerhold <stephan@gerhold.net>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009

```



```

00010 #include "uart_base.h"
00011
00012 namespace L4 {
00013
00014 class Uart_dm : public Uart
00015 {
00016 public:
00017     explicit Uart_dm(unsigned /*base_rate*/) {}
00018     bool startup(Io_register_block const *) override;
00019     void shutdown() override;
00020     bool change_mode(Transfer_mode m, Baud_rate r) override;
00021     int tx_avail() const;
00022     void wait_tx_done() const;
00023     inline void out_char(char c) const;
00024     int write(char const *s, unsigned long count,
00025              bool blocking = true) const override;
00026
00027     bool enable_rx_irq(bool enable = true) override;
00028     int char_avail() const override;
00029     int get_char(bool blocking = true) const override;
00030 };
00031
00032 } // namespace L4

```

## 17.22 uart\_dummy.h

```

00001 /*
00002  * Copyright 2009 Technische Universität Dresden.
00003  * Copyright (C) 2023-2024 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "uart_base.h"
00011
00012 namespace L4 {
00013
00014 class Uart_dummy : public Uart
00015 {
00016 public:
00017     explicit Uart_dummy() {}
00018     explicit Uart_dummy(unsigned /*base_rate*/) {}
00019     bool startup(Io_register_block const *) override { return true; }
00020     void shutdown() override {}
00021     bool change_mode(Transfer_mode, Baud_rate) override { return true; }
00022     inline void out_char(char /*ch*/) const {}
00023     int write(char const * /*str*/, unsigned long /*count*/,
00024              bool /*blocking*/ = true) const override
00025     { return 0; }
00026
00027     int char_avail() const override { return false; }
00028     int get_char(bool /*blocking*/ = true) const override { return 0; }
00029 };
00030
00031 } // namespace L4

```

## 17.23 uart\_geni.h

```

00001 /*
00002  * Copyright (C) 2022-2024 Kernkonzept GmbH.
00003  * Author(s): Stephan Gerhold <stephan.gerhold@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "uart_base.h"
00010
00011 namespace L4 {
00012
00013 class Uart_geni : public Uart
00014 {
00015 public:
00016     explicit Uart_geni(unsigned /*base_rate*/) {}
00017     bool startup(Io_register_block const *) override;
00018     void shutdown() override;

```

```

00019 bool change_mode(Transfer_mode m, Baud_rate r) override;
00020 int tx_avail() const;
00021 void wait_tx_done() const;
00022 void out_char(char c) const;
00023 int write(char const *s, unsigned long count,
00024           bool blocking = true) const override;
00025
00026 bool enable_rx_irq(bool enable = true) override;
00027 void irq_ack() override;
00028 int char_avail() const override;
00029 int get_char(bool blocking = true) const override;
00030 };
00031
00032 } // namespace L4

```

## 17.24 uart\_imx.h

```

00001 /*
00002  * Copyright (C) 2008-2009 Technische Universität Dresden.
00003  * Copyright (C) 2023-2024 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "uart_base.h"
00011
00012 namespace L4 {
00013
00014 class Uart_imx : public Uart
00015 {
00016 public:
00017     enum platform_type
00018     {
00019         Type_imx21,
00020         Type_imx35,
00021         Type_imx51,
00022         Type_imx6,
00023         Type_imx7,
00024         Type_imx8,
00025     };
00026     explicit Uart_imx(enum platform_type type) : _type(type) {}
00027     explicit Uart_imx(enum platform_type type, unsigned /*base_rate*/)
00028         : _type(type) {}
00029     bool startup(Io_register_block const *) override;
00030     void shutdown() override;
00031     bool change_mode(Transfer_mode m, Baud_rate r) override;
00032     int tx_avail() const;
00033     void wait_tx_done() const;
00034     inline void out_char(char c) const;
00035     int write(char const *s, unsigned long count,
00036             bool blocking = true) const override;
00037
00038     bool enable_rx_irq(bool enable = true) override;
00039     int char_avail() const override;
00040     int get_char(bool blocking = true) const override;
00041
00042 private:
00043     enum platform_type _type;
00044 };
00045
00046 class Uart_imx21 : public Uart_imx
00047 {
00048 public:
00049     Uart_imx21() : Uart_imx(Type_imx21) {}
00050     explicit Uart_imx21(unsigned base_rate) : Uart_imx(Type_imx21, base_rate) {}
00051 };
00052
00053 class Uart_imx35 : public Uart_imx
00054 {
00055 public:
00056     Uart_imx35() : Uart_imx(Type_imx35) {}
00057     explicit Uart_imx35(unsigned base_rate) : Uart_imx(Type_imx35, base_rate) {}
00058 };
00059
00060 class Uart_imx51 : public Uart_imx
00061 {
00062 public:
00063     Uart_imx51() : Uart_imx(Type_imx51) {}
00064     explicit Uart_imx51(unsigned base_rate) : Uart_imx(Type_imx51, base_rate) {}
00065 };
00066

```

```

00067 class Uart_imx6 : public Uart_imx
00068 {
00069 public:
00070     Uart_imx6() : Uart_imx(Type_imx6) {}
00071     explicit Uart_imx6(unsigned base_rate) : Uart_imx(Type_imx6, base_rate) {}
00072
00073     void irq_ack() override;
00074 };
00075
00076 class Uart_imx7 : public Uart_imx
00077 {
00078 public:
00079     Uart_imx7() : Uart_imx(Type_imx7) {}
00080     explicit Uart_imx7(unsigned base_rate) : Uart_imx(Type_imx7, base_rate) {}
00081 };
00082
00083 class Uart_imx8 : public Uart_imx
00084 {
00085 public:
00086     Uart_imx8() : Uart_imx(Type_imx8) {}
00087     explicit Uart_imx8(unsigned base_rate) : Uart_imx(Type_imx8, base_rate) {}
00088 };
00089
00090 } // namespace L4

```

## 17.25 uart\_leon3.h

```

00001 /*
00002  * Copyright (C) 2011 Technische Universität Dresden.
00003  * Copyright (C) 2023-2024 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00005  *            Björn Döbel <doebel@os.inf.tu-dresden.de>
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include "uart_base.h"
00012
00013 namespace L4 {
00014
00015 class Uart_leon3 : public Uart
00016 {
00017 public:
00018     explicit Uart_leon3() {}
00019     explicit Uart_leon3(unsigned /*base_rate*/) {}
00020     bool startup(Io_register_block const *) override;
00021     void shutdown() override;
00022     bool change_mode(Transfer_mode m, Baud_rate r) override;
00023     int tx_avail() const;
00024     void wait_tx_done() const;
00025     inline void out_char(char c) const;
00026     int write(char const *s, unsigned long count,
00027              bool blocking = true) const override;
00028
00029     bool enable_rx_irq(bool = true) override;
00030     int char_avail() const override;
00031     int get_char(bool blocking = true) const override;
00032 };
00033
00034 } // namespace L4

```

## 17.26 uart\_linflex.h

```

00001 /*
00002  * Copyright (C) 2018 Technische Universität Dresden.
00003  * Copyright (C) 2023-2024 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@l4re.org>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "uart_base.h"
00011
00012 namespace L4 {
00013
00014 class Uart_linflex : public Uart

```

```

00015 {
00016 public:
00017     explicit Uart_linflex(unsigned) {}
00018     bool startup(Io_register_block const *) override;
00019     void shutdown() override;
00020     bool change_mode(Transfer_mode m, Baud_rate r) override;
00021     int tx_avail() const;
00022     void wait_tx_done() const;
00023     inline void out_char(char c) const;
00024     int write(char const *s, unsigned long count,
00025               bool blocking = true) const override;
00026
00027     bool enable_rx_irq(bool enable = true) override;
00028     int char_avail() const override;
00029     int get_char(bool blocking = true) const override;
00030
00031 private:
00032     void set_uartcr(bool fifo);
00033     bool is_tx_fifo_enabled() const;
00034     bool is_rx_fifo_enabled() const;
00035 };
00036
00037 } // namespace L4

```

## 17.27 uart\_lpuart.h

```

00001 /*
00002  * Copyright (C) 2019, 2023-2024 Kernkonzept GmbH.
00003  * Author(s): Adam Lackorzynski <adam@l4re.org>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "uart_base.h"
00010
00011 namespace L4 {
00012
00013 class Uart_lpuart : public Uart
00014 {
00015 public:
00016     Uart_lpuart(unsigned freq = 0) : _freq(freq) {}
00017     bool startup(Io_register_block const *) override;
00018     void shutdown() override;
00019     bool change_mode(Transfer_mode m, Baud_rate r) override;
00020     int tx_avail() const;
00021     void wait_tx_done() const {}
00022     inline void out_char(char c) const;
00023     int write(char const *s, unsigned long count,
00024               bool blocking = true) const override;
00025
00026     bool enable_rx_irq(bool enable = true) override;
00027     int char_avail() const override;
00028     int get_char(bool blocking = true) const override;
00029
00030 private:
00031     unsigned _freq;
00032 };
00033
00034 } // namespace L4

```

## 17.28 uart\_mvebu.h

```

00001 /*
00002  * Copyright (C) 2017, 2023-2024 Kernkonzept GmbH.
00003  * Author(s): Adam Lackorzynski <adam@l4re.org>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "uart_base.h"
00010
00011 namespace L4 {
00012
00013 class Uart_mvebu : public Uart
00014 {
00015 public:

```

```

00016     explicit Uart_mvebu(unsigned baserate) : _baserate(baserate) {}
00017     bool startup(Io_register_block const *) override;
00018     void shutdown() override;
00019     bool change_mode(Transfer_mode m, Baud_rate r) override;
00020     int tx_avail() const;
00021     void wait_tx_done() const {}
00022     inline void out_char(char c) const;
00023     int write(char const *s, unsigned long count,
00024               bool blocking = true) const override;
00025
00026     bool enable_rx_irq(bool enable = true) override;
00027     int char_avail() const override;
00028     int get_char(bool blocking = true) const override;
00029
00030 private:
00031     unsigned _baserate;
00032 };
00033
00034 } // namespace L4

```

## 17.29 uart\_of.h

```

00001 /*
00002  * Copyright (C) 2009 Technische Universität Dresden.
00003  * Copyright (C) 2023-2024 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "uart_base.h"
00011 #include <stdarg.h>
00012 #include <string.h>
00013 #include <l4/drivers/of.h>
00014
00015 namespace L4 {
00016
00017     class Uart_of : public Uart, public L4_drivers::Of
00018     {
00019     private:
00020         ihandle_t _serial;
00021
00022     public:
00023         Uart_of() : Of(), _serial(0) {}
00024         explicit Uart_of(unsigned /*base_rate*/) : Of(), _serial(0) {}
00025         bool startup(Io_register_block const *) override;
00026         void shutdown() override;
00027         bool change_mode(Transfer_mode m, Baud_rate r) override;
00028         int tx_avail() const;
00029         void out_char(char c) const;
00030         int write(char const *s, unsigned long count,
00031                  bool blocking = true) const override;
00032
00033         int char_avail() const override;
00034         int get_char(bool blocking = true) const override;
00035     };
00036
00037 } // namespace L4

```

## 17.30 uart\_omap35x.h

```

00001 /*
00002  * Copyright (C) 2009 Technische Universität Dresden.
00003  * Copyright (C) 2023-2024 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "uart_base.h"
00011
00012 namespace L4 {
00013
00014     class Uart_omap35x : public Uart
00015     {
00016     public:

```

```

00017     explicit Uart_omap35x() {}
00018     explicit Uart_omap35x(unsigned /*base_rate*/) {}
00019     bool startup(Io_register_block const *) override;
00020     void shutdown() override;
00021     bool change_mode(Transfer_mode m, Baud_rate r) override;
00022     int tx_avail() const;
00023     void wait_tx_done() const;
00024     inline void out_char(char c) const;
00025     int write(char const *s, unsigned long count,
00026              bool blocking = true) const override;
00027
00028     bool enable_rx_irq(bool) override;
00029     int char_avail() const override;
00030     int get_char(bool blocking = true) const override;
00031 };
00032
00033 } // namespace L4

```

## 17.31 uart\_pl011.h

```

00001 /*
00002  * Copyright (C) 2009 Technische Universität Dresden.
00003  * Copyright (C) 2023-2024 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "uart_base.h"
00011
00012 namespace L4 {
00013
00014     class Uart_pl011 : public Uart
00015     {
00016     public:
00017         Uart_pl011(unsigned freq) : _freq(freq) {}
00018         bool startup(Io_register_block const *) override;
00019         void shutdown() override;
00020         bool change_mode(Transfer_mode m, Baud_rate r) override;
00021         int tx_avail() const;
00022         void wait_tx_done() const;
00023         inline void out_char(char c) const;
00024         int write(char const *s, unsigned long count,
00025                  bool blocking = true) const override;
00026
00027         bool enable_rx_irq(bool enable) override;
00028         int char_avail() const override;
00029         int get_char(bool blocking = true) const override;
00030
00031     private:
00032         void set_rate(Baud_rate r);
00033         unsigned _freq;
00034     };
00035
00036 } // namespace L4

```

## 17.32 uart\_s3c2410.h

```

00001 /*
00002  * Copyright (C) 2009-2012 Technische Universität Dresden.
00003  * Copyright (C) 2023-2024 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "uart_base.h"
00011
00012 namespace L4 {
00013
00014     class Uart_s3c : public Uart
00015     {
00016     protected:
00017         enum Uart_type
00018         {
00019             Type_24xx, Type_64xx, Type_s5pv210,

```

```

00020     };
00021
00022     Uart_type type() const { return _type; }
00023
00024 public:
00025     explicit Uart_s3c(Uart_type type) : _type(type) {}
00026     explicit Uart_s3c(Uart_type type, unsigned /*base_rate*/) : _type(type) {}
00027     bool startup(Io_register_block const *) override;
00028     void shutdown() override;
00029     bool change_mode(Transfer_mode m, Baud_rate r) override;
00030     int tx_avail() const;
00031     void wait_tx_done() const;
00032     inline void out_char(char c) const;
00033     int write(char const *s, unsigned long count,
00034              bool blocking = true) const override;
00035     void fifo_reset();
00036
00037     int char_avail() const override;
00038     int get_char(bool blocking = true) const override;
00039
00040 protected:
00041     virtual void ack_rx_irq() const = 0;
00042     virtual void wait_for_empty_tx_fifo() const = 0;
00043     virtual unsigned is_rx_fifo_non_empty() const = 0;
00044     virtual unsigned is_tx_fifo_not_full() const = 0;
00045
00046 private:
00047     Uart_type _type;
00048 };
00049
00050 class Uart_s3c2410 : public Uart_s3c
00051 {
00052 public:
00053     Uart_s3c2410() : Uart_s3c(Type_24xx) {}
00054     explicit Uart_s3c2410(unsigned base_rate) : Uart_s3c(Type_24xx, base_rate) {}
00055
00056 protected:
00057     void ack_rx_irq() const override {}
00058     void wait_for_empty_tx_fifo() const override;
00059     unsigned is_rx_fifo_non_empty() const override;
00060     unsigned is_tx_fifo_not_full() const override;
00061
00062     void auto_flow_control(bool on);
00063 };
00064
00065 class Uart_s3c64xx : public Uart_s3c
00066 {
00067 public:
00068     Uart_s3c64xx() : Uart_s3c(Type_64xx) {}
00069     explicit Uart_s3c64xx(unsigned base_rate) : Uart_s3c(Type_64xx, base_rate) {}
00070
00071 protected:
00072     void ack_rx_irq() const override;
00073     void wait_for_empty_tx_fifo() const override;
00074     unsigned is_rx_fifo_non_empty() const override;
00075     unsigned is_tx_fifo_not_full() const override;
00076 };
00077
00078 class Uart_s5pv210 : public Uart_s3c
00079 {
00080 public:
00081     Uart_s5pv210() : Uart_s3c(Type_s5pv210) {}
00082     explicit Uart_s5pv210(unsigned base_rate) : Uart_s3c(Type_s5pv210, base_rate) {}
00083
00084 protected:
00085     void ack_rx_irq() const override;
00086     void wait_for_empty_tx_fifo() const override;
00087     unsigned is_rx_fifo_non_empty() const override;
00088     unsigned is_tx_fifo_not_full() const override;
00089 };
00090
00091 } // namespace L4

```

## 17.33 uart\_sa1000.h

```

00001 /*
00002  * Copyright (C) 2008-2012 Technische Universität Dresden.
00003  * Copyright (C) 2023-2024 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00005  *            Alexander Warg <alexander.warg@os.inf.tu-dresden.de>
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */

```

```

00009 #pragma once
00010
00011 #include "uart_base.h"
00012
00013 namespace L4 {
00014
00015 class Uart_sal000 : public Uart
00016 {
00017 public:
00018     explicit Uart_sal000() {}
00019     explicit Uart_sal000(unsigned /*base_rate*/) {}
00020     bool startup(Io_register_block const *) override;
00021     void shutdown() override;
00022     bool change_mode(Transfer_mode m, Baud_rate r) override;
00023     int tx_avail() const;
00024     void wait_tx_done() const;
00025     inline void out_char(char c) const;
00026     int write(char const *s, unsigned long count,
00027              bool blocking = true) const override;
00028
00029     int char_avail() const override;
00030     int get_char(bool blocking = true) const override;
00031 };
00032
00033 } // namespace L4

```

## 17.34 uart\_sbi.h

```

00001 /*
00002  * Copyright (C) 2021, 2024 Kernkonzept GmbH.
00003  * Author(s): Georg Kotheimer <georg.kotheimer@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "uart_base.h"
00010
00011 namespace L4 {
00012
00013 class Uart_sbi : public Uart
00014 {
00015 public:
00016     Uart_sbi();
00017     explicit Uart_sbi(unsigned /*base_rate*/) : Uart_sbi() {}
00018     bool startup(Io_register_block const *) override;
00019     void shutdown() override;
00020     bool change_mode(Transfer_mode m, Baud_rate r) override;
00021     int tx_avail() const { return true; }
00022     void wait_tx_done() const {}
00023     inline void out_char(char c) const;
00024     int write(char const *s, unsigned long count,
00025              bool blocking = true) const override;
00026
00027     int char_avail() const override;
00028     int get_char(bool blocking = true) const override;
00029
00030 private:
00031     mutable int _bufchar;
00032 };
00033
00034 } // namespace L4

```

## 17.35 uart\_sh.h

```

00001 /*
00002  * Copyright (C) 2016 Technische Universität Dresden.
00003  * Copyright (C) 2023-2024 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@l4re.org>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "uart_base.h"
00011
00012 namespace L4 {
00013

```



```

00014 class Uart_sh : public Uart
00015 {
00016 public:
00017     explicit Uart_sh() {}
00018     explicit Uart_sh(unsigned /*base_rate*/) {}
00019     bool startup(Io_register_block const *) override;
00020     void shutdown() override;
00021     bool change_mode(Transfer_mode m, Baud_rate r) override;
00022     int tx_avail() const;
00023     void wait_tx_done() const {}
00024     inline void out_char(char c) const;
00025     int write(char const *s, unsigned long count,
00026              bool blocking = true) const override;
00027
00028     bool enable_rx_irq(bool enable = true) override;
00029     void irq_ack() override;
00030     int char_avail() const override;
00031     int get_char(bool blocking = true) const override;
00032 };
00033
00034 } // namespace L4

```

## 17.36 uart\_sifive.h

```

00001 /*
00002  * Copyright (C) 2021-2024 Kernkonzept GmbH.
00003  * Author(s): Georg Kotheimer <georg.kotheimer@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "uart_base.h"
00010
00011 namespace L4 {
00012
00013 class Uart_sifive : public Uart
00014 {
00015 public:
00016     Uart_sifive(unsigned freq) : _freq(freq), _bufchar(-1) {}
00017     bool startup(Io_register_block const *) override;
00018     void shutdown() override;
00019     bool change_mode(Transfer_mode m, Baud_rate r) override;
00020     int tx_avail() const;
00021     void wait_tx_done() const;
00022     inline void out_char(char c) const;
00023     int write(char const *s, unsigned long count,
00024              bool blocking = true) const override;
00025
00026     bool enable_rx_irq(bool enable) override;
00027     int char_avail() const override;
00028     int get_char(bool blocking = true) const override;
00029
00030 private:
00031     unsigned _freq;
00032     mutable int _bufchar;
00033 };
00034
00035 } // namespace L4

```

## 17.37 uart\_tegra-tcu.h

```

00001 /*
00002  * Copyright (C) 2025 Kernkonzept GmbH.
00003  * Author(s): Adam Lackorzynski <adam@l4re.org>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "uart_base.h"
00010
00011 namespace L4 {
00012
00013 class Uart_tegra_tcu : public Uart
00014 {
00015 public:
00016     Uart_tegra_tcu() {}

```

```

00017  Uart_tegra_tcu(unsigned long) {}
00018  bool startup(Io_register_block const *tx_mbox) override;
00019  void set_rx_mbox(Io_register_block const *rx_mbox);
00020  void shutdown() override;
00021  bool change_mode(Transfer_mode m, Baud_rate r) override;
00022  int tx_avail() const;
00023  void wait_tx_done() const {}
00024  inline void out_char(char c) const;
00025  int write(char const *s, unsigned long count,
00026           bool blocking = true) const override;
00027
00028  int char_avail() const override;
00029  int get_char(bool blocking = true) const override;
00030
00031 private:
00032     mutable unsigned _current_rx_val = 0u;
00033     Io_register_block const *_rx_mbox = nullptr;
00034 };
00035
00036 } // namespace L4

```

## 17.38 tutorial.lua

```

00001 local _doc = [=[
00002
00003 Tutorial lua script for Ned
00004 =====
00005
00006 Firstly we have a set of definitions available. Some come from 'ned.lua'
00007 embedded script and others from the C++ bindings within Ned. The whole L4
00008 functionality is in the lua module "L4" (use 'local L4 = require("L4");').
00009 The L4 module classes and functions cope with L4 capabilities and
00010 their invocations, provide a set of constants and access to the L4Re environment of
00011 the running program. Finally, of course it can also start L4 applications.
00012
00013 L4 Capabilities
00014 =====
00015
00016 The L4 module defines a user data type for capabilities. A capability in lua
00017 carries a typed L4 capability and is accompanied with a set of type specific
00018 methods that may be called on the object behind the capability. There also
00019 exists a way to cast a capability to a capability to a different type of
00020 object using L4.cast(type, cap).
00021
00022 L4.cast(type, cap)
00023
00024 Returns a cap transformed to a capability of the given type, whereas type
00025 is either the fully qualified C++ name of the class encapsulating the object
00026 or the L4 protocol ID assigned to all L4Re and L4 system objects.
00027 If the type is unknown then nil is returned.
00028
00029 Generic capabilities provide the following methods:
00030
00031 is_valid()
00032
00033 Returns true if the capability is not the invalid cap (L4_INVALID_CAP), and
00034 false if it is the invalid cap.
00035
00036
00037 L4Re::Namespace object
00038 =====
00039
00040 There is a lua type for name spaces that has the following methods:
00041
00042 query(name), or q(name)
00043
00044 Returns a closure (function) that initiates a query for the given name
00045 within the name space. The function takes no arguments and returns
00046 a capability if successful or nil if name is not found.
00047
00048
00049 link(name), or l(name)
00050
00051 Returns a function that can create a link to the given object in the name
00052 space if put into another name space. The function takes two parameters
00053 the target name space and the name in the target name space.
00054 Loader:create_namespace and Loader:fill_namespace calls this function
00055 when things are really put into an L4Re::Namespace.
00056
00057
00058 register(name, cap), or r(name, cap)
00059
00060 Registers the given object capability under the given name. cap can

```

```

00061  be either a real capability (note query returns a function), a string, or
00062  nil. If it is a capability it is just put into the name space.
00063  In the case cap is a string a placeholder will be put into the name space
00064  that will be replaced with a real capability later by some program.
00065  And if nil is use the name will be deleted from the name space.
00066
00067
00068  L4::Factory object
00069  =====
00070
00071  The factory object provides an interface to the generic create method of a
00072  factory.
00073
00074  create(proto, ...)
00075
00076  This method calls the factory to create an object of the given type,
00077  via the L4 protocol number (see L4.Proto table for more) all further
00078  arguments are passed to the factory.
00079
00080
00081  Access to the L4Re Env capabilities
00082  =====
00083
00084  The L4 module defines a table L4.Env that contains the capabilities
00085  of the L4Re::Env::env() environment. Namely:
00086
00087  factory      The kernel factory of Ned
00088  log           The log object of Ned
00089  user_factory The factory provided to Ned, including memory
00090  parent       The parent of Ned
00091  rm           The region map of Ned
00092  scheduler    The scheduler of Ned
00093
00094
00095  Some useful constants
00096  =====
00097
00098  L4.Proto table contains the most important protocol values for
00099  L4 and L4Re objects.
00100
00101  Namespace
00102  Goos
00103  Rm
00104  Irq
00105  Sigma0
00106  Factory
00107  Log
00108  Scheduler
00109
00110  The L4.Info table contains the following functions:
00111
00112  Kip.str()     The banner string found in the kernel info page
00113  arch()        Architecture name, such as: x86, amd64, arm, ppc32
00114  platform()    Platform name, such as: pc, ux, realview, beagleboard
00115  mword_bits() Number of native machine word bits (32, 64)
00116
00117
00118  Support for starting L4 programs
00119  =====
00120
00121  The L4 module defines two classes that are useful for starting L4 applications.
00122  The class L4.Loader that encapsulates a fairly high level policy
00123  that is useful for starting a whole set of processes. And the class L4.App_env
00124  that encapsulates a more fine-grained policy.
00125
00126  L4.Loader
00127  -----
00128
00129  The class L4.Loader encapsulates the policy for starting programs with the
00130  basic building blocks for the application coming from a dedicated loader,
00131  such as Moe or a Loader instance. These building blocks are a region map (Rm),
00132  a scheduler, a memory allocator, and a logging facility.
00133  A L4.Loader object is typically used to start multiple applications. There
00134  is a L4.default_loader instance of L4.Loader that uses the L4.Env.mem_alloc
00135  factory of the current Ned instance to create the objects for a new program.
00136  However you may also use a more restricted factory for applications and
00137  instantiate a loader for them. The L4.Loader objects can already be used
00138  to start a program with L4.Loader:start(app_spec, cmd, ...). Where app_spec
00139  is a table containing parameters for the new application. cmd is the
00140  command to run and the remaining arguments are the command-line options for
00141  the application.
00142
00143  ]==]
00144
00145  L4.default_loader:start({}, "rom/hello");
00146
00147  local _doc = [=[

```

```

00148
00149 This statement does the following:
00150 1. Create a new environment for the application
00151 2. Add the rom name-space into the new environment (thus shares Ned's
00152    'rom' directory with the new program).
00153 3. Creates all the building blocks for the new process and starts the
00154    'l4re' support kernel in the new process which in turn starts 'rom/hello'
00155    in the new process.
00156
00157 Using the app_spec parameter you can modify the behavior in two ways. There are
00158 two supported options 'caps' for providing more capabilities for the
00159 application. And 'log' for modifying the logger tag and color.
00160
00161 ]==]
00162
00163 local my_caps = {
00164     fb = L4.Env.vesa;
00165 };
00166
00167 L4.default_loader:start({caps = my_caps, log = {"APP", "blue"}}, "rom/hello");
00168
00169 local _doc = [=[
00170
00171 This snippet creates a caps template (my_caps) and uses it for the
00172 new process and also sets user-defined log tags. The L4.Loader:start method,
00173 however, automatically adds the 'rom' directory to the caps environment if
00174 not already specified in the template.
00175
00176 Environment variables may be given as a table in the third argument to
00177 start. Argument to the program are given space separated after the program
00178 name within a single string.
00179
00180 ]==]
00181
00182 L4.default_loader:start({}, "rom/program arg1 " .. arg2, { LD_DEBUG = 1 });
00183
00184 local _doc = [=[
00185
00186 L4.default_loader:startv is a variant of the start function that takes the
00187 arguments of the program as a single argument each. If the last argument to
00188 startv is a table it is interpreted as environment variables for the program.
00189 The above example would translate to:
00190
00191 ]==]
00192
00193 L4.default_loader:startv({}, "rom/program", "arg1", arg2, { LD_DEBUG = 1 });
00194
00195 local _doc = [=[
00196
00197 To create a new L4.Loader instance you may use a generic factory for all
00198 building blocks or set individual factories.
00199
00200 ]==]
00201
00202 l = L4.Loader.new({
00203     mem = L4.Env.user_factory:create(L4.Proto.Factory, 512*1024):m("rs")
00204 });
00205
00206 local _doc = [=[
00207
00208 Creates a loader instance that uses the newly created 512 KB factory for
00209 all building blocks. To set individual factories use the options:
00210 'mem'          as memory allocator for the new processes and as
00211                default factory for all objects not explicitly set to a
00212                different factory
00213 'log_fab'      for creating log objects.
00214 'ns_fab'       for creating name-space objects.
00215 'rm_fab'       for creating region-map objects.
00216
00217
00218
00219 L4.App_env
00220 -----
00221
00222 L4.App_env provides a more fine-grained control for a single process or for a
00223 limited number of processes. L4.App_env uses an L4.Loader object as basic
00224 facility. However you can override the memory allocator 'mem' for the new
00225 process as well as the kernel factory 'factory', the log capability etc.
00226
00227 ]==]
00228
00229 local e = L4.App_env.new({
00230     loader = l,
00231     mem = L4.Env.user_factory:create(L4.Proto.Factory, 128*1024):m("rs")
00232 });
00233
00234 e:start("rom/hello");

```

## 17.39 cmd\_control

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00004  * Copyright (C) 2016, 2024 Kernkonzept GmbH.
00005  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include <l4/sys/cxx/ipc_epiface>
00012 #include <l4/sys/cxx/ipc_string>
00013
00014 namespace L4Re { namespace Ned {
00015
00016 class Cmd_control : public L4::Kobject_0t<Cmd_control>
00017 {
00018     L4_INLINE_RPC_NF(long, execute, (L4::Ipc::String<> cmd,
00019                                     L4::Ipc::Array<char> &result));
00020
00021 public:
00022     long execute(L4::Ipc::String<> cmd) noexcept
00023     {
00024         L4::Ipc::Array<char> res(0, NULL);
00025         return execute_t::call(c(), cmd, res);
00026     }
00027
00028     long execute(L4::Ipc::String<> cmd,
00029                 L4::Ipc::String<char> *result) noexcept
00030     {
00031         L4::Ipc::Array<char> res(result->length, result->data);
00032         long r = execute_t::call(c(), cmd, res);
00033         if (r >= 0)
00034             result->length = res.length;
00035         return r;
00036     }
00037
00038 typedef L4::Typeid::Rpc<execute_t> Rpc;
00039 };
00040 } } // namespace

```

## 17.40 debug.h

```

00001 /*
00002  * (c) 2013-2014 Alexander Warg <warg@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/re/util/debug>
00010
00011 struct Err : L4Re::Util::Err
00012 {
00013     Err(Level l = Fatal) : L4Re::Util::Err(l, "VSwitch") {}
00014 };
00015
00016 class Dbg : public L4Re::Util::Dbg
00017 {
00018     enum
00019     {
00020         Verbosity_shift = 4,
00021         Verbosity_mask = (1UL « Verbosity_shift) - 1
00022     };
00023
00024 public:
00025     enum Verbosity : unsigned long
00026     {
00027         Quiet = 0,
00028         Warn = 1,
00029         Info = 2,
00030         Debug = 4,
00031         Trace = 8,
00032         Max_verbosity = 8
00033     };
00034
00035     enum Component
00036     {

```

```

00041     Core = 0,
00042     Virtio,
00043     Port,
00044     Request,
00045     Queue,
00046     Packet,
00047     Max_component
00048 };
00049
00050 #ifndef NDEBUG
00051
00052     static_assert(Max_component * Verbosity_shift <= sizeof(level) * 8,
00053         "Too many components for level mask");
00054     static_assert((Max_verbosity & Verbosity_mask) == Max_verbosity,
00055         "Verbosity_shift too small for verbosity levels");
00056
00062     static void set_verbosity(unsigned mask)
00063     {
00064         for (unsigned i = 0; i < Max_component; ++i)
00065             set_verbosity(i, mask);
00066     }
00067
00074     static void set_verbosity(unsigned c, unsigned mask)
00075     {
00076         level &= ~(Verbosity_mask << (Verbosity_shift * c));
00077         level |= (mask & Verbosity_mask) << (Verbosity_shift * c);
00078     }
00079
00089     static bool is_active(unsigned c, unsigned mask)
00090     { return level & (mask & Verbosity_mask) << (Verbosity_shift * c); }
00091
00098     using L4Re::Util::Dbg::is_active;
00099
00100     Dbg(Component c = Core, Verbosity v = Warn, char const *subsys = "")
00101     : L4Re::Util::Dbg(v << (Verbosity_shift * c), "SWI", subsys)
00102     {}
00103
00104 #else
00105
00106     static void set_verbosity(unsigned) {}
00107     static void set_verbosity(unsigned, unsigned) {}
00108
00109     static bool is_active(unsigned, unsigned) { return false; }
00110     using L4Re::Util::Dbg::is_active;
00111
00112     Dbg(Component c = Core, Verbosity v = Warn, char const *subsys = "")
00113     : L4Re::Util::Dbg(v << (Verbosity_shift * c), "", subsys)
00114     {}
00115
00116 #endif
00117 };

```

## 17.41 debug.h

```

00001 /*
00002  * Copyright (C) 2018, 2024 Kernkonzept GmbH.
00003  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/re/util/debug>
00010
00011 namespace Block_device {
00012
00013     class Err : public L4Re::Util::Err
00014     {
00015     public:
00016         explicit
00017         Err(Level l = Normal) : L4Re::Util::Err(l, "") {}
00018     };
00019
00020     class Dbg : public L4Re::Util::Dbg
00021     {
00022     public:
00023         enum Level
00024         {
00025             Blk_warn = 1,
00026             Blk_info = 2,
00027             Blk_trace = 4,
00028             Blk_steptrace = 8
00029         };

```

```

00030 public:
00031     Dbg(unsigned long l = Blk_info, char const *subsys = "")
00032     : L4Re::Util::Dbg(l, "libblock", subsys) {}
00033
00034     static Dbg warn(char const *subsys = "")
00035     { return Dbg(Blk_warn, subsys); }
00036
00037     static Dbg info(char const *subsys = "")
00038     { return Dbg(Blk_info, subsys); }
00039
00040     static Dbg trace(char const *subsys = "")
00041     { return Dbg(Blk_trace, subsys); }
00042
00043     static Dbg steptrace(char const *subsys = "")
00044     { return Dbg(Blk_steptrace, subsys); }
00045 };
00046
00047 } // name space
00048

```

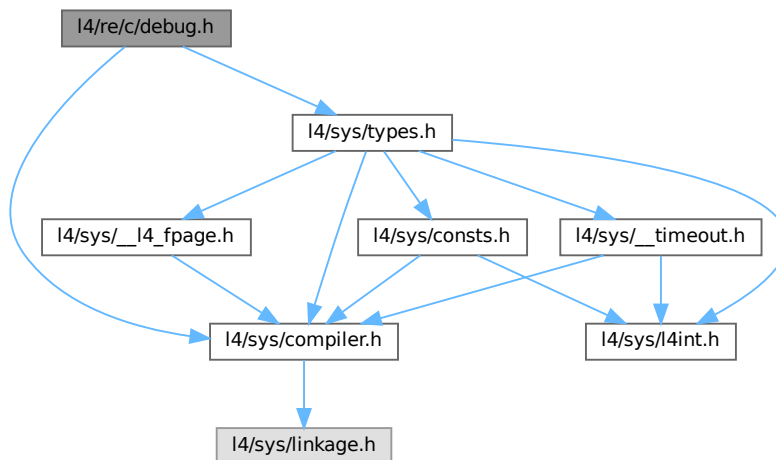
## 17.42 l4/re/c/debug.h File Reference

Debug C interface.

```

#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
Include dependency graph for debug.h:

```



### Functions

- `L4_BEGIN_DECLS` long `l4re_debug_obj_debug` (`l4_cap_idx_t` srv, unsigned long function) `L4_NOTHROW`  
Call debug function of `L4Re` service.

### 17.42.1 Detailed Description

Debug C interface.

Definition in file `debug.h`.

## 17.43 debug.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *      economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00017
00018 #include <14/sys/compiler.h>
00019 #include <14/sys/types.h>
00020
00021 L4_BEGIN_DECLS
00022
00030 L4_CV long
00031 l4re_debug_obj_debug(l4_cap_idx_t srv, unsigned long function) L4_NOTHROW;
00032
00033 L4_END_DECLS

```

## 17.44 filter.cc

```

00001 /*
00002  * Copyright (C) 2016-2017, 2024 Kernkonzept GmbH.
00003  * Author(s): Jean Wolter <jean.wolter@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #include "filter.h"
00008
00009 /* This is an example filter and therefore rather verbose. A real
00010    filter would not produce any output */
00011
00012 bool
00013 filter(const uint8_t *buf, size_t size)
00014 {
00015     // Packet large enough to apply filter condition?
00016     if (size <= 13)
00017         return false;
00018
00019     uint16_t ether_type = (uint16_t)*(buf + 12) << 8
00020                          | (uint16_t)*(buf + 13);
00021     char const *protocol;
00022     switch (ether_type)
00023     {
00024         case 0x0800: protocol = "IPv4"; break;
00025         case 0x0806: protocol = "ARP"; break;
00026         case 0x8100: protocol = "Vlan"; break;
00027         case 0x86dd: protocol = "IPv6"; break;
00028         case 0x8863: protocol = "PPPoE Discovery"; break;
00029         case 0x8864: protocol = "PPPoE Session"; break;
00030         default: protocol = nullptr;
00031     }
00032     if (protocol)
00033         printf("%s\n", protocol);
00034     else
00035         printf("%04x\n", ether_type);
00036
00037     if (ether_type == 0x0806)
00038     {
00039         printf("Do not filter arp\n");
00040         return false;
00041     }
00042
00043     return true;
00044 }

```

## 17.45 filter.h

```

00001 /*
00002  * Copyright (C) 2016-2017, 2023-2024 Kernkonzept GmbH.
00003  * Author(s): Jean Wolter <jean.wolter@kernkonzept.com>

```



```

00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "request.h"
00010 #include <l4/bid_config.h>
00011
00024 #ifndef CONFIG_VNS_PORT_FILTER
00025 bool filter(const uint8_t *buf, size_t size);
00026 #else
00027 inline bool filter(const uint8_t *, size_t)
00028 {
00029     // default implementation filtering out no packets, see filter.cc for
00030     // other examples
00031     return false;
00032 }
00033 #endif
00034
00043 inline bool filter_request(Net_request const &req)
00044 {
00045     size_t size;
00046     const uint8_t *buf = req.buffer(&size);
00047     return filter(buf, size);
00048 }

```

## 17.46 mac\_addr.h

```

00001 /*
00002  * Copyright (C) 2016-2017, 2020, 2022-2024 Kernkonzept GmbH.
00003  * Author(s): Jean Wolter <jean.wolter@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <cstring>
00010 #include <inttypes.h>
00015
00019 class Mac_addr
00020 {
00021 public:
00022     enum
00023     {
00024         Addr_length = 6,
00025         Addr_unknown = 0ULL
00026     };
00027
00028     explicit Mac_addr(char const *_src)
00029     {
00030         /* A mac address is 6 bytes long, it is transmitted in big endian
00031            order over the network. For our internal representation we
00032            focus on easy testability of broadcast/multicast and reorder
00033            the bytes that the most significant byte becomes the least
00034            significant one. */
00035         unsigned char const *src = reinterpret_cast<unsigned char const *>(_src);
00036         _mac = ((uint64_t)src[0]) | (((uint64_t)src[1]) << 8)
00037             | (((uint64_t)src[2]) << 16) | (((uint64_t)src[3]) << 24)
00038             | (((uint64_t)src[4]) << 32) | (((uint64_t)src[5]) << 40);
00039     }
00040
00041     static Mac_addr from_uncached(char volatile const *src)
00042     { return Mac_addr(src); }
00043
00044     explicit Mac_addr(uint64_t mac) : _mac{mac} {}
00045
00046     Mac_addr(Mac_addr const &other) : _mac{other._mac} {}
00047
00049     bool is_broadcast() const
00050     {
00051         /* There are broadcast and multicast addresses, both are supposed
00052            to be delivered to all station and the local network (layer 2).
00053
00054            Broadcast address is FF:FF:FF:FF:FF:FF, multicast addresses have
00055            the LSB of the first octet set. Since this holds for both
00056            broadcast and multicast we test for the multicast bit here.
00057
00058            In our internal representation we store the bytes in reverse
00059            order, so we test the least significant bit of the least
00060            significant byte.
00061         */
00062         return _mac & 1;
00063     }

```

```

00064
00066 bool is_unknown() const
00067 { return _mac == Addr_unknown; }
00068
00069 bool operator == (Mac_addr const &other) const
00070 { return _mac == other._mac; }
00071
00072 bool operator != (Mac_addr const &other) const
00073 { return _mac != other._mac; }
00074
00075 bool operator < (Mac_addr const &other) const
00076 { return _mac < other._mac; }
00077
00078 Mac_addr& operator = (Mac_addr const &other)
00079 { _mac = other._mac; return *this; }
00080
00081 Mac_addr& operator = (uint64_t mac)
00082 { _mac = mac; return *this; }
00083
00084 template<typename T>
00085 void print(T &stream) const
00086 {
00087     stream.cprintf("%02x:%02x:%02x:%02x:%02x",
00088         (int)(_mac & 0xff), (int)((_mac >> 8) & 0xff),
00089         (int)((_mac >> 16) & 0xff), (int)((_mac >> 24) & 0xff),
00090         (int)((_mac >> 32) & 0xff), (int)((_mac >> 40) & 0xff));
00091 }
00092
00093 void to_array(unsigned char mac[6]) const
00094 {
00095     mac[0] = _mac & 0xffU;
00096     mac[1] = (_mac >> 8) & 0xffU;
00097     mac[2] = (_mac >> 16) & 0xffU;
00098     mac[3] = (_mac >> 24) & 0xffU;
00099     mac[4] = (_mac >> 32) & 0xffU;
00100     mac[5] = (_mac >> 40) & 0xffU;
00101 }
00102
00103 private:
00104     explicit Mac_addr(char volatile const *_src)
00105     {
00106         /* A mac address is 6 bytes long, it is transmitted in big endian
00107            order over the network. For our internal representation we
00108            focus on easy testability of broadcast/multicast and reorder
00109            the bytes that the most significant byte becomes the least
00110            significant one. */
00111         volatile unsigned char const *src = reinterpret_cast<volatile unsigned char const *>(_src);
00112         _mac = ((uint64_t)src[0]) | (((uint64_t)src[1]) << 8)
00113             | (((uint64_t)src[2]) << 16) | (((uint64_t)src[3]) << 24)
00114             | (((uint64_t)src[4]) << 32) | (((uint64_t)src[5]) << 40);
00115     }
00116
00117     uint64_t _mac;
00118 };

```

## 17.47 mac\_table.h

```

00001 /*
00002  * Copyright (C) 2016-2017, 2020, 2022-2024 Kernkonzept GmbH.
00003  * Author(s): Jean Wolter <jean.wolter@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "mac_addr.h"
00010 #include "port.h"
00011
00012 #include <array>
00013 #include <map>
00014 #include <tuple>
00015 #include <algorithm>
00020
00039 template<std::size_t Size = 1024U>
00040 class Mac_table
00041 {
00042 public:
00043     Mac_table()
00044     : _mac_table(),
00045       _entries(),
00046       _rr_index(0U)
00047     {}
00048

```

```

00058 Port_iface *lookup(Mac_addr dst, l4_uint16_t vlan_id) const
00059 {
00060     auto entry = _mac_table.find(std::tuple(dst, vlan_id));
00061     return (entry != _mac_table.end()) ? entry->second->port : nullptr;
00062 }
00063
00078 void learn(Mac_addr src, Port_iface *port, l4_uint16_t vlan_id)
00079 {
00080     Dbg info(Dbg::Port, Dbg::Info);
00081
00082     if (L4_UNLIKELY(info.is_active()))
00083     {
00084         // check whether we already know about src mac and vlan_id
00085         auto *p = lookup(src, vlan_id);
00086         if (!p || p != port)
00087         {
00088             info.printf("%s %-20s -> ", !p ? "learned " : "replaced",
00089                 port->get_name());
00090             src.print(info);
00091             info.cprintf("\n");
00092         }
00093     }
00094
00095     auto status = _mac_table.emplace(std::tuple(src, vlan_id),
00096         &_entries[_rr_index]);
00097     if (L4_UNLIKELY(status.second))
00098     {
00099         if (_entries[_rr_index].port)
00100         {
00101             // remove old entry
00102             _mac_table.erase(std::tuple(_entries[_rr_index].addr,
00103                 _entries[_rr_index].vlan_id));
00104         }
00105         // Set/Replace port and mac address
00106         _entries[_rr_index].port = port;
00107         _entries[_rr_index].addr = src;
00108         _entries[_rr_index].vlan_id = vlan_id;
00109         _rr_index = (_rr_index + 1U) % Size;
00110     }
00111     else
00112     {
00113         // Update port to allow for movement of client between ports
00114         status.first->second->port = port;
00115     }
00116 }
00117
00129 void flush(Port_iface *port)
00130 {
00131     typedef std::pair<std::tuple<const Mac_addr, l4_uint16_t>, Entry*> TableEntry;
00132
00133     auto iter = _mac_table.begin();
00134     while ((iter = std::find_if(iter, _mac_table.end(),
00135         [port](TableEntry const &p)
00136             { return p.second->port == port; })))
00137     {
00138         iter = _mac_table.end()
00139     {
00140         iter->second->port = nullptr;
00141         iter->second->addr = Mac_addr::Addr_unknown;
00142         iter->second->vlan_id = 0;
00143         iter = _mac_table.erase(iter);
00144     }
00145
00146     assert(std::find_if(_mac_table.begin(), _mac_table.end(),
00147         [port](TableEntry const &p)
00148             { return p.second->port == port; }) == _mac_table.end());
00149 }
00150 private:
00151 struct Entry {
00152     Port_iface *port;
00153     Mac_addr addr;
00154     l4_uint16_t vlan_id;
00155
00156     Entry()
00157     : port(nullptr),
00158       addr(Mac_addr::Addr_unknown),
00159       vlan_id(0)
00160     {}
00161 };
00162
00163 std::map<std::tuple<Mac_addr, l4_uint16_t>, Entry*> _mac_table;
00164 std::array<Entry, Size> _entries;
00165 size_t _rr_index;
00166 };

```

## 17.48 main.cc

```

00001 /*
00002  * Copyright (C) 2016-2020, 2022-2024 Kernkonzept GmbH.
00003  * Author(s): Jean Wolter <jean.wolter@kernkonzept.com>
00004  *             Manuel von Oltersdorff-Kalettka <manuel.kalettka@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #include <l4/re/util/meta>
00009 #include <l4/re/util/object_registry>
00010 #include <l4/re/util/br_manager>
00011
00012 #include <l4/sys/factory>
00013 #include <l4/sys/task>
00014
00015 #include <l4/sys/cxx/ipc_epiface>
00016 #include <l4/sys/cxx/ipc_varg>
00017 #include <l4/cxx/dlist>
00018 #include <l4/cxx/string>
00019
00020 #include <stdlib.h>
00021 #include <string>
00022 #include <terminate_handler-l4>
00023 #include <vector>
00024
00025 #include "debug.h"
00026 #include "options.h"
00027 #include "switch.h"
00028 #include "vlan.h"
00029 #include <l4/virtio-net-switch/stats.h>
00030
00047
00048
00049 /*
00050  * Registry for our server, used to register
00051  * - factory capability
00052  * - irq object for capability deletion irqs
00053  * - virtio host kick irqs
00054  */
00055 static L4Re::Util::Registry_server<L4Re::Util::Br_manager_hooks> server;
00056
00057 using Ds_vector = std::vector<L4::Cap<L4Re::Dataspace>>;
00058 static std::shared_ptr<Ds_vector> trusted_dataspaces;
00059
00060 static bool
00061 parse_int_param(L4::Ipc::Varg const &param, char const *prefix, int *out)
00062 {
00063     l4_size_t headlen = strlen(prefix);
00064
00065     if (param.length() < headlen)
00066         return false;
00067
00068     char const *pstr = param.value<char const *>();
00069
00070     if (strncmp(pstr, prefix, headlen) != 0)
00071         return false;
00072
00073     std::string tail(pstr + headlen, param.length() - headlen);
00074
00075     if (!parse_int_optstring(tail.c_str(), out))
00076     {
00077         Err(Err::Normal).printf("Bad parameter '%s'. Invalid number specified.\n",
00078                                 prefix);
00079         throw L4::Runtime_error(-L4_EINVAL);
00080     }
00081
00082     return true;
00083 }
00084
00085 static void
00086 assign_random_mac(l4_uint8_t mac[6])
00087 {
00088     static bool initialized = false;
00089
00090     if (!initialized)
00091     {
00092         srandom(l4_kip_clock(l4re_kip()));
00093         initialized = true;
00094     }
00095
00096     for (int i = 0; i < 6; i++)
00097         mac[i] = static_cast<l4_uint8_t>(random());
00098
00099     mac[0] &= ~(1U << 0); // clear multicast bit
00100     mac[0] |= 1U << 1;    // set "locally administered" bit

```

```

00101 }
00102
00114 class Switch_factory : public L4::Epiface_t<Switch_factory, L4::Factory>
00115 {
00119     class Port : public L4virtio_port
00120     {
00121         // Irq used to notify the guest
00122         L4::Cap<L4::Irq> _device_notify_irq;
00123
00124         L4::Cap<L4::Irq> device_notify_irq() const override
00125         { return _device_notify_irq; }
00126
00127     public:
00128         Port(unsigned vq_max, unsigned num_ds, char const *name,
00129              l4_uint8_t const *mac)
00130         : L4virtio_port(vq_max, num_ds, name, mac) {}
00131
00133         void register_end_points(L4Re::Util::Object_registry* registry,
00134                                 L4::Epiface *kick_irq)
00135         {
00136             // register virtio host kick irq
00137             _device_notify_irq = L4Re::chkcap(registry->register_irq_obj(kick_irq));
00138
00139             // register virtio endpoint
00140             L4Re::chkcap(registry->register_obj(this));
00141
00142             // decrement ref counter to get a notification when the last
00143             // external reference vanishes
00144             obj_cap()->dec_refcnt(1);
00145         }
00146
00147         virtual ~Port()
00148         { server.registry()->unregister_obj(this); }
00149     };
00150
00154     class Switch_port : public Port
00155     {
00166         class Kick_irq : public L4::Irqep_t<Kick_irq>
00167         {
00168             Virtio_switch *_switch;
00169             L4virtio_port *_port;
00170
00171         public:
00172             void handle_irq()
00173             { _switch->handle_l4virtio_port_tx(_port); }
00174
00175             Kick_irq(Virtio_switch *virtio_switch, L4virtio_port *port)
00176             : _switch{virtio_switch}, _port{port} {}
00177         };
00178
00185         Kick_irq _kick_irq;
00186         Kick_irq _reschedule_tx_irq;
00187
00188     public:
00189         Switch_port(L4Re::Util::Object_registry *registry,
00190                    Virtio_switch *virtio_switch, unsigned vq_max, unsigned num_ds,
00191                    char const *name, l4_uint8_t const *mac)
00192         : Port(vq_max, num_ds, name, mac),
00193           _kick_irq(virtio_switch, this),
00194           _reschedule_tx_irq(virtio_switch, this)
00195         {
00196             register_end_points(registry, &_kick_irq);
00197
00198             _pending_tx_reschedule =
00199                 L4Re::chkcap(registry->register_irq_obj(&_reschedule_tx_irq),
00200                             "Register TX reschedule IRQ.");
00201             _pending_tx_reschedule->unmask();
00202         }
00203
00204         virtual ~Switch_port()
00205         {
00206             // We need to delete the IRQ object created in register_irq_obj() ourselves
00207             L4::Cap<L4::Task> (L4Re::This_task)
00208                 ->unmap(_kick_irq.obj_cap().fpage(),
00209                        L4_FP_ALL_SPACES | L4_FP_DELETE_OBJ);
00210             server.registry()->unregister_obj(&_kick_irq);
00211
00212             L4::Cap<L4::Task> (L4Re::This_task)
00213                 ->unmap(_pending_tx_reschedule.fpage(),
00214                        L4_FP_ALL_SPACES | L4_FP_DELETE_OBJ);
00215             server.registry()->unregister_obj(&_reschedule_tx_irq);
00216         }
00217     };
00218
00222     class Monitor_port : public Port
00223     {
00229         class Kick_irq : public L4::Irqep_t<Kick_irq>

```

```

00230     {
00231         L4virtio_port *_port;
00232     }
00233 public:
00241     void handle_irq()
00242     {
00243         do
00244         {
00245             _port->tx_q()->disable_notify();
00246             _port->rx_q()->disable_notify();
00247
00248             _port->drop_requests();
00249
00250             _port->tx_q()->enable_notify();
00251             _port->rx_q()->enable_notify();
00252
00253             L4virtio::wmb();
00254             L4virtio::rmb();
00255         }
00256         while (_port->tx_work_pending());
00257     }
00258
00259     Kick_irq(L4virtio_port *port) : _port{port} {}
00260 };
00261
00262     Kick_irq _kick_irq;
00263
00264 public:
00265     Monitor_port(L4Re::Util::Object_registry* registry,
00266                 unsigned vq_max, unsigned num_ds, char const *name,
00267                 l4_uint8_t const *mac)
00268     : Port(vq_max, num_ds, name, mac), _kick_irq(this)
00269     { register_end_points(registry, &_kick_irq); }
00270
00271     virtual ~Monitor_port()
00272     {
00273         // We need to delete the IRQ object created in register_irq_obj() ourselves
00274         L4::Cap<L4::Task> (L4Re::This_task)
00275             ->unmap(_kick_irq.obj_cap().fpage(),
00276                   L4_FP_ALL_SPACES | L4_FP_DELETE_OBJ);
00277         server.registry()->unregister_obj(&_kick_irq);
00278     }
00279 };
00280
00281 class Stats_reader
00282 : public cxx::D_list_item,
00283   public L4::Epiface_t<Stats_reader, Virtio_net_switch::Statistics_if>
00284 {
00285     L4Re::Util::Unique_cap<L4Re::Dataspace> _ds;
00286     l4_addr_t _addr;
00287
00288 public:
00289     Stats_reader()
00290     {
00291         l4_size_t size = Switch_statistics::get_instance().size();
00292         _ds = L4Re::Util::make_unique_cap<L4Re::Dataspace>();
00293         L4Re::chksys(L4Re::Env::env()->mem_alloc()->alloc(size, _ds.get()),
00294                     "Could not allocate shared mem ds.");
00295         L4Re::chksys(L4Re::Env::env()->rm()->attach(&_addr, _ds->size(),
00296                                                     L4Re::Rm::F::Search_addr
00297                                                     | L4Re::Rm::F::RW,
00298                                                     L4::Ipc::make_cap_rw(_ds.get())));
00299
00300         memset(reinterpret_cast<void*>(_addr), 0, _ds->size());
00301     }
00302
00303     ~Stats_reader()
00304     {
00305         L4Re::Env::env()->rm()->detach(reinterpret_cast<l4_addr_t>(_addr), 0);
00306         server.registry()->unregister_obj(this);
00307     }
00308
00309     long op_get_buffer(Virtio_net_switch::Statistics_if::Rights,
00310                      L4::Ipc::Cap<L4Re::Dataspace> &ds)
00311     {
00312         // We hand out the dataspace in a read only manner. Clients must not be
00313         // able to modify information as that would create an unwanted data
00314         // channel.
00315         ds = L4::Ipc::Cap<L4Re::Dataspace>(_ds.get(), L4_CAP_FPAGE_RO);
00316         return L4_EOK;
00317     }
00318
00319     long op_sync(Virtio_net_switch::Statistics_if::Rights)
00320     {
00321         memcpy(reinterpret_cast<void*>(_addr),
00322               reinterpret_cast<void*>(Switch_statistics::get_instance().stats()),
00323               Switch_statistics::get_instance().size());
00324     }

```

```

00327         return L4_EOK;
00328     }
00329
00330     bool is_valid()
00331     { return obj_cap() && obj_cap().validate().label(); }
00332 };
00333
00334 class Stats_reader_list
00335 {
00336     cxx::D_list<Stats_reader> _readers;
00337
00338 public:
00339     void check_readers()
00340     {
00341         auto it = _readers.begin();
00342         while (it != _readers.end())
00343         {
00344             auto *reader = *it;
00345             if (!reader->is_valid())
00346             {
00347                 it = _readers.erase(it);
00348                 delete reader;
00349             }
00350             else
00351                 ++it;
00352         }
00353     }
00354
00355     void push_back(cxx::unique_ptr<Stats_reader> reader)
00356     {
00357         _readers.push_back(reader.release());
00358     }
00359 };
00360
00361 /*
00362  * Handle vanishing caps by telling the switch that a port might have gone
00363  */
00364 struct Del_cap_irq : public L4::Irqep_t<Del_cap_irq>
00365 {
00366 public:
00367     void handle_irq()
00368     {
00369         _switch->check_ports();
00370         _stats_readers->check_readers();
00371     }
00372
00373     Del_cap_irq(Virtio_switch *virtio_switch, Stats_reader_list *stats_readers)
00374     : _switch{virtio_switch},
00375       _stats_readers{stats_readers}
00376     {}
00377
00378 private:
00379     Virtio_switch *_switch;
00380     Stats_reader_list *_stats_readers;
00381 };
00382
00383 Virtio_switch *_virtio_switch;
00384
00385 unsigned _vq_max_num;
00386 Stats_reader_list _stats_readers;
00387 Del_cap_irq _del_cap_irq;
00388
00389 bool handle_opt_arg(L4::Ipc::Varg const &opt, bool &monitor,
00400                    char *name, size_t size,
00401                    l4_uint16_t &vlan_access,
00402                    std::vector<l4_uint16_t> &vlan_trunk,
00403                    bool *vlan_trunk_all,
00404                    l4_uint8_t mac[6], bool &mac_set)
00405 {
00406     assert(opt.is_of<char const *>());
00407     unsigned len = opt.length();
00408     const char *opt_str = opt.data();
00409     Err err(Err::Normal);
00410
00411     if (len > 5)
00412     {
00413         if (!strncmp("type=", opt_str, 5))
00414         {
00415             if (!strncmp("type=monitor", opt_str, len))
00416             {
00417                 monitor = true;
00418                 return true;
00419             }
00420             else if (!strncmp("type=none", opt_str, len))
00421                 return true;
00422         }
00423     }
00424 }

```

```

00430         err.printf("Unknown type '%s'\n", opt.length() - 5, opt.data() + 5);
00431         return false;
00432     }
00433     else if (!strcmp("name=", opt_str, 5))
00434     {
00435         snprintf(name, size, "%s", opt.length() - 5, opt.data() + 5);
00436         return true;
00437     }
00438     else if (!strcmp("vlan=", opt_str, 5))
00439     {
00440         cxx::String str(opt_str + 5, strnlen(opt_str + 5, len - 5));
00441         cxx::String::Index idx;
00442
00443         if ((idx = str.starts_with("access=")))
00444         {
00445             str = str.substr(idx);
00446             l4_uint16_t vid;
00447             int next = str.from_dec(&vid);
00448             if (next && next == str.len() && vlan_valid_id(vid))
00449                 vlan_access = vid;
00450             else
00451             {
00452                 err.printf("Invalid VLAN access port id '%s'\n",
00453                     opt.length(), opt.data());
00454                 return false;
00455             }
00456         }
00457         else if ((idx = str.starts_with("trunk=")))
00458         {
00459             int next;
00460             l4_uint16_t vid;
00461             str = str.substr(idx);
00462             if (str == cxx::String("all"))
00463             {
00464                 *vlan_trunk_all = true;
00465                 return true;
00466             }
00467             while ((next = str.from_dec(&vid)))
00468             {
00469                 if (!vlan_valid_id(vid))
00470                     break;
00471                 vlan_trunk.push_back(vid);
00472                 if (next < str.len() && str[next] != ',')
00473                     break;
00474                 str = str.substr(next+1);
00475             }
00476
00477             if (vlan_trunk.empty() || !str.empty())
00478             {
00479                 err.printf("Invalid VLAN trunk port spec '%s'\n",
00480                     opt.length(), opt.data());
00481                 return false;
00482             }
00483         }
00484         else
00485         {
00486             err.printf("Invalid VLAN specification..\n");
00487             return false;
00488         }
00489     }
00490     return true;
00491 }
00492 else if (!strcmp("mac=", opt_str, 4))
00493 {
00494     size_t const OPT_LEN = 4 /* mac= */ + 6*2 /* digits */ + 5 /* : */;
00495     // expect NUL terminated string for simplicity
00496     if (len > OPT_LEN && opt_str[OPT_LEN] == '\0' &&
00497         sscanf(opt_str+4, "%hhx:%hhx:%hhx:%hhx:%hhx:%hhx", &mac[0],
00498             &mac[1], &mac[2], &mac[3], &mac[4], &mac[5]) == 6)
00499     {
00500         mac_set = true;
00501         return true;
00502     }
00503
00504     err.printf("Invalid mac address '%s'\n", len - 4, opt_str + 4);
00505     return false;
00506 }
00507 }
00508
00509 err.printf("Unknown option '%s'\n", opt.length(), opt.data());
00510 return false;
00511 }
00512
00513 public:
00514     Switch_factory(Virtio_switch *virtio_switch, unsigned vq_max_num)
00515     : _virtio_switch{virtio_switch}, _vq_max_num{vq_max_num},
00516       _del_cap_irq{virtio_switch, &_stats_readers}

```



```

00517 {
00518     auto c = L4Re::chkcapp(server.registry()->register_irq_obj(&_del_cap_irq));
00519     L4Re::chksys(L4Re::Env::env()->main_thread()->register_del_irq(c));
00520 };
00521
00522 long op_create(L4::Factory::Rights, L4::Ipc::Cap<void> &res,
00523               l4_umword_t type, L4::Ipc::Varg_list_ref va)
00524 {
00525     switch (type)
00526     {
00527     case 0:
00528         return create_port(res, va);
00529     case 1:
00530         return create_stats(res);
00531     default:
00532         Dbg(Dbg::Core, Dbg::Warn).printf("op_create: Invalid object type\n");
00533         return -L4_EINVAL;
00534     }
00535 }
00536
00537 long create_port(L4::Ipc::Cap<void> &res, L4::Ipc::Varg_list_ref va)
00538 {
00539     Dbg warn(Dbg::Port, Dbg::Warn, "Port");
00540     Dbg info(Dbg::Port, Dbg::Info, "Port");
00541
00542     info.printf("Incoming port request\n");
00543
00544     bool monitor = false;
00545     char name[20] = "";
00546     unsigned arg_n = 2;
00547     l4_uint16_t vlan_access = 0;
00548     std::vector<l4_uint16_t> vlan_trunk;
00549     bool vlan_trunk_all = false;
00550
00551     l4_uint8_t mac[6];
00552     bool mac_set = false;
00553     int num_ds = 2;
00554
00555     for (L4::Ipc::Varg opt: va)
00556     {
00557         if (!opt.is_of<char const *>())
00558         {
00559             warn.printf("Unexpected type for argument %d\n", arg_n);
00560             return -L4_EINVAL;
00561         }
00562
00563         if (parse_int_param(opt, "ds-max=", &num_ds))
00564         {
00565             if (num_ds <= 0 || num_ds > 80)
00566             {
00567                 Err(Err::Normal).printf("warning: client requested invalid number"
00568                                         " of data spaces: 0 < %d <= 80\n", num_ds);
00569                 return -L4_EINVAL;
00570             }
00571         }
00572         else if (!handle_opt_arg(opt, monitor, name, sizeof(name), vlan_access,
00573                                 vlan_trunk, &vlan_trunk_all, mac, mac_set))
00574             return -L4_EINVAL;
00575
00576         ++arg_n;
00577     }
00578
00579     int port_num = _virtio_switch->port_available(monitor);
00580     if (port_num < 0)
00581     {
00582         warn.printf("No port available\n");
00583         return -L4_ENOMEM;
00584     }
00585
00586     if (vlan_access && (!vlan_trunk.empty() || vlan_trunk_all))
00587     {
00588         warn.printf("VLAN port cannot be access and trunk simultaneously.\n");
00589         return -L4_EINVAL;
00590     }
00591
00592     if (!name[0])
00593         snprintf(name, sizeof(name), "%s[%d]", monitor ? "monitor" : "",
00594                 port_num);
00595
00596     info.printf("    Creating port %s\n", name,
00597                monitor ? " as monitor port" : "");
00598
00599     // Assign a random MAC address if we assign one to our devices but the
00600     // user has not passed an explicit one for a port.
00601     if (!mac_set && Options::get_options()->assign_mac())
00602         assign_random_mac(mac);
00603 }

```

```

00610     l4_uint8_t *mac_ptr = (mac_set || Options::get_options()->assign_mac())
00611                          ? mac : nullptr;
00612
00613     // create port
00614     Port *port;
00615     if (monitor)
00616     {
00617         port = new Monitor_port(server.registry(), _vq_max_num, num_ds, name,
00618                                mac_ptr);
00619         port->set_monitor();
00620
00621         if (vlan_access)
00622             warn.printf("vlan=access=<id> ignored on monitor ports!\n");
00623         if (!vlan_trunk.empty())
00624             warn.printf("vlan=trunk=... ignored on monitor ports!\n");
00625     }
00626     else
00627     {
00628         port = new Switch_port(server.registry(), _virtio_switch, _vq_max_num,
00629                                num_ds, name, mac_ptr);
00630
00631         if (vlan_access)
00632             port->set_vlan_access(vlan_access);
00633         else if (vlan_trunk_all)
00634             port->set_vlan_trunk_all();
00635         else if (!vlan_trunk.empty())
00636             port->set_vlan_trunk(vlan_trunk);
00637     }
00638
00639     port->add_trusted_dataspaces(trusted_dataspaces);
00640     if (!trusted_dataspaces->empty())
00641         port->enable_trusted_ds_validation();
00642
00643     // hand port over to the switch
00644     bool added = monitor ? _virtio_switch->add_monitor_port(port)
00645                          : _virtio_switch->add_port(port);
00646     if (!added)
00647     {
00648         delete port;
00649         return -L4_ENOMEM;
00650     }
00651     res = L4::Ipc::make_cap(port->obj_cap(), L4_CAP_FPAGE_RWS);
00652
00653     info.printf("    Created port %s\n", name);
00654     return L4_EOK;
00655 }
00656
00657 long create_stats(L4::Ipc::Cap<void> &res)
00658 {
00659     // Create a stats reader and throw away our reference to get a notification
00660     // when the external reference vanishes.
00661     auto reader = cxx::make_unique<Stats_reader>();
00662     L4Re::chkcapi(server.registry()->register_obj(reader.get()));
00663     reader->obj_cap()->dec_refcnt(1);
00664     res = L4::Ipc::make_cap(reader->obj_cap(),
00665                             L4_CAP_FPAGE_R | L4_CAP_FPAGE_D);
00666     _stats_readers.push_back(cxx::move(reader));
00667     return L4_EOK;
00668 }
00669 };
00670
00671 #if CONFIG_VNS_IXL
00672 class Ixl_hw_port : public Ixl_port
00673 {
00674     template<typename Derived>
00675     class Port_irq : public L4::Irqp_t<Derived>
00676     {
00677     public:
00678         Port_irq(Virtio_switch *virtio_switch, Ixl_port *port)
00679             : _switch{virtio_switch}, _port{port} {}
00680
00681     protected:
00682         Virtio_switch *_switch;
00683         Ixl_port *_port;
00684     };
00685
00686     class Receive_irq : public Port_irq<Receive_irq>
00687     {
00688     public:
00689         using Port_irq::Port_irq;
00690
00691         void handle_irq()
00692         {
00693             if (!_port->dev()->check_recv_irq(0))
00694                 return;
00695
00696             if (_switch->handle_ixl_port_tx(_port))

```

```

00706     _port->dev()->ack_recv_irq(0);
00707 }
00708 };
00709
00710 class Reschedule_tx_irq : public Port_irq<Reschedule_tx_irq>
00711 {
00712 public:
00713     using Port_irq::Port_irq;
00714
00715     void handle_irq()
00716     {
00717         if (_switch->handle_ixl_port_tx(_port))
00718             // Entire TX queue handled, re-enable the recv IRQ again.
00719             _port->dev()->ack_recv_irq(0);
00720     }
00721 };
00722
00723 Receive_irq _recv_irq;
00724 Reschedule_tx_irq _reschedule_tx_irq;
00725
00726 public:
00727     Ixl_hw_port(L4Re::Util::Object_registry *registry,
00728                 Virtio_switch *virtio_switch, Ixl::Ixl_device *dev)
00729     : Ixl_port(dev),
00730       _recv_irq(virtio_switch, this),
00731       _reschedule_tx_irq(virtio_switch, this)
00732     {
00733         L4::Cap<L4::Irq> recv_irq_cap = L4Re::chkcapi(dev->get_recv_irq(0), "Get receive IRQ");
00734         L4Re::chkcapi(registry->register_obj(&_recv_irq, recv_irq_cap),
00735                       "Register receive IRQ.");
00736         recv_irq_cap->unmask();
00737
00738         _pending_tx_reschedule =
00739             L4Re::chkcapi(registry->register_irq_obj(&_reschedule_tx_irq),
00740                           "Register TX reschedule IRQ.");
00741         _pending_tx_reschedule->unmask();
00742     }
00743
00744     ~Ixl_hw_port() override
00745     {
00746         server.registry()->unregister_obj(&_recv_irq);
00747     }
00748 };
00749
00750 static void
00751 discover_ixl_devices(L4::Cap<L4vbus::Vbus> vbus, Virtio_switch *virtio_switch)
00752 {
00753     struct Ixl::Dev_cfg cfg;
00754     // Configure the device in asynchronous notify mode.
00755     cfg.irq_timeout_ms = -1;
00756
00757     // TODO: Support detecting multiple devices on a Vbus.
00758     // Setup the driver (also resets and initializes the NIC).
00759     Ixl::Ixl_device *dev = Ixl::Ixl_device::ixl_init(vbus, 0, cfg);
00760     if (!dev)
00761         // No Ixl supported device found, Ixl already printed an error message.
00762         return;
00763
00764     Ixl_hw_port *hw_port = new Ixl_hw_port(server.registry(), virtio_switch, dev);
00765     if (!virtio_switch->add_port(hw_port))
00766     {
00767         Err().printf("error adding ixl port\n");
00768         delete hw_port;
00769     }
00770 }
00771 #endif
00772
00773 int main(int argc, char *argv[])
00774 {
00775     trusted_dataspaces = std::make_shared<Ds_vector>();
00776     auto *opts = Options::parse_options(argc, argv, trusted_dataspaces);
00777     if (!opts)
00778     {
00779         Err().printf("Error during command line parsing.\n");
00780         return 1;
00781     }
00782
00783     // Show welcome message if debug level is not set to quiet
00784     if (Dbg(Dbg::Core, Dbg::Warn).is_active())
00785         printf("Hello from l4virtio switch\n");
00786
00787     Virtio_switch *virtio_switch = new Virtio_switch(opts->get_max_ports());
00788
00789     #ifndef CONFIG_VNS_STATS
00790     Switch_statistics::get_instance().initialize(opts->get_max_ports());
00791     #endif
00792

```

```

00793 #if CONFIG_VNS_IXL
00794     auto vbus = L4Re::Env::env()->get_cap<L4vbus::Vbus>("vbus");
00795     if (vbus.is_valid())
00796         discover_ixl_devices(vbus, virtio_switch);
00797 #endif
00798
00799     Switch_factory *factory = new Switch_factory(virtio_switch,
00800                                                  opts->get_virtq_max_num());
00801
00802     L4::Cap<void> cap = server.registry()->register_obj(factory, "svr");
00803     if (!cap.is_valid())
00804     {
00805         Err().printf("error registering switch\n");
00806         return 2;
00807     }
00808
00809     /*
00810     * server loop will handle 4 types of events
00811     * - Switch_factory
00812     *   - factory protocol
00813     *   - capability deletion
00814     *     - delegated to Virtio_switch::check_ports()
00815     * - Switch_factory::Switch_port
00816     *   - irqs triggered by clients
00817     *     - delegated to Virtio_switch::handle_l4virtio_port_tx()
00818     * - Virtio_net_transfer
00819     *   - timeouts for pending transfer requests added by
00820     *     Port_iface::handle_request() via registered via
00821     *     L4::Epiface::server_iface()->add_timeout()
00822     */
00823     server.loop();
00824     return 0;
00825 }
00826

```

## 17.49 Makefile

```

00001 PKGDIR = ..
00002 L4DIR ?= $(PKGDIR)/../..
00003
00004 PKGNAME = drivers
00005 PC_FILENAME = drivers-frst
00006 EXTRA_TARGET += hw_mmio_register_block hw_register_block
00007
00008 include $(L4DIR)/mk/include.mk

```

## 17.50 Makefile

```

00001 PKGDIR = ../..
00002 L4DIR ?= $(PKGDIR)/../..
00003
00004 PKGNAME = drivers
00005
00006 include $(L4DIR)/mk/include.mk

```

## 17.51 Makefile

```

00001 PKGDIR = ../..
00002 L4DIR ?= $(PKGDIR)/../..
00003
00004 EXTRA_TARGET += cmd_control
00005
00006 include $(L4DIR)/mk/include.mk

```

## 17.52 Makefile

```

00001 PKGDIR      ?= ../..
00002 L4DIR      ?= $(PKGDIR)/../..
00003

```

```

00004 TARGET          = l4vio_switch
00005
00006 REQUIRES_LIBS      = libstdc++ l4virtio
00007 REQUIRES_LIBS-$(CONFIG_VNS_IXL) += ixl
00008
00009 SRC_CC-$(CONFIG_VNS_PORT_FILTER) += filter.cc
00010
00011 SRC_CC = main.cc switch.cc options.cc
00012
00013 include $(L4DIR)/mk/prog.mk

```

## 17.53 options.cc

```

00001 /*
00002  * Copyright (C) 2016-2017, 2019, 2022-2024 Kernkonzept GmbH.
00003  * Author(s): Jean Wolter <jean.wolter@kernkonzept.com>
00004  *             Manuel von Oltersdorff-Kalettka <manuel.kalettka@kernkonzept.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #include <getopt.h>
00010 #include <stdlib.h>
00011 #include <cstring>
00012 #include <type_traits>
00013
00014 #include <l4/cxx/exceptions>
00015 #include <l4/re/error_helper>
00016 #include <l4/re/env>
00017
00018 #include "debug.h"
00019 #include "options.h"
00020
00021 bool
00022 parse_int_optstring(char const *optstring, int *out)
00023 {
00024     char *endp;
00025
00026     errno = 0;
00027     long num = strtol(optstring, &endp, 10);
00028
00029     // check that long can be converted to int
00030     if (errno || *endp != '\0' || num < INT_MIN || num > INT_MAX)
00031         return false;
00032
00033     *out = num;
00034
00035     return true;
00036 }
00037
00038 static int
00039 verbosity_mask_from_string(char const *str, unsigned *mask)
00040 {
00041     if (strcmp("quiet", str) == 0)
00042     {
00043         *mask = Dbg::Quiet;
00044         return 0;
00045     }
00046     if (strcmp("warn", str) == 0)
00047     {
00048         *mask = Dbg::Warn;
00049         return 0;
00050     }
00051     if (strcmp("info", str) == 0)
00052     {
00053         *mask = Dbg::Warn | Dbg::Info;
00054         return 0;
00055     }
00056     if (strcmp("debug", str) == 0)
00057     {
00058         *mask = Dbg::Warn | Dbg::Info | Dbg::Debug;
00059         return 0;
00060     }
00061     if (strcmp("trace", str) == 0)
00062     {
00063         *mask = Dbg::Warn | Dbg::Info | Dbg::Debug | Dbg::Trace;
00064         return 0;
00065     }
00066
00067     return -L4_ENOENT;
00068 }
00069
00099 static void

```

```

00100 set_verbosity(char const *str)
00101 {
00102     unsigned mask;
00103     if (verbosity_mask_from_string(str, &mask) == 0)
00104     {
00105         Dbg::set_verbosity(mask);
00106         return;
00107     }
00108
00109     static char const *const components[] =
00110     { "core", "virtio", "port", "request", "queue", "packet" };
00111
00112     static_assert(std::extent<decltype(components)>::value == Dbg::Max_component,
00113         "Component names must match 'enum Component'.");
00114
00115     for (unsigned i = 0; i < Dbg::Max_component; ++i)
00116     {
00117         auto len = strlen(components[i]);
00118         if (strncmp(components[i], str, len) == 0 && str[len] == '='
00119             && verbosity_mask_from_string(str + len + 1, &mask) == 0)
00120         {
00121             Dbg::set_verbosity(i, mask);
00122             return;
00123         }
00124     }
00125 }
00126
00127 int
00128 Options::parse_cmd_line(int argc, char **argv,
00129     std::shared_ptr<Ds_vector> trusted_dataspaces)
00130 {
00131     int opt, index;
00132
00133     struct option options[] =
00134     {
00135         {"size",          1, 0, 's' }, // size of in/out queue == #buffers in queue
00136         {"ports",         1, 0, 'p' }, // number of ports
00137         {"mac",           0, 0, 'm' }, // switch sets MAC address for each client
00138         {"debug",         1, 0, 'D' }, // configure debug levels
00139         {"verbose",       0, 0, 'v' },
00140         {"quiet",         0, 0, 'q' },
00141         {"register-ds",   1, 0, 'd' }, // register a trusted dataspace
00142         {0, 0, 0, 0}
00143     };
00144
00145     unsigned long verbosity = Dbg::Warn;
00146
00147     Dbg info(Dbg::Core, Dbg::Info);
00148
00149     Dbg::set_verbosity(Dbg::Core, Dbg::Info);
00150     info.printf("Arguments:\n");
00151     for (int i = 0; i < argc; ++i)
00152         info.printf("\t%s\n", argv[i]);
00153
00154     Dbg::set_verbosity(verbosity);
00155     while ( (opt = getopt_long(argc, argv, "s:p:mMqVd:d:", options, &index)) != -1)
00156     {
00157         switch (opt)
00158         {
00159             case 's':
00160
00161                 // QueueNumMax must be power of 2 between 1 and 0x8000
00162                 if (!parse_int_optstring(optarg, &_virtq_max_num)
00163                     || _virtq_max_num < 1 || _virtq_max_num > 32768
00164                     || (_virtq_max_num & (_virtq_max_num - 1)))
00165                 {
00166                     Err().printf("Max number of virtqueue buffers must be power of 2 "
00167                         "between 1 and 32768. Invalid value %i or argument "
00168                         "%s\n",
00169                         _virtq_max_num, optarg);
00169                     return -1;
00170                 }
00171                 info.printf("Max number of buffers in virtqueue: %i\n",
00172                     _virtq_max_num);
00173                 break;
00174             case 'p':
00175                 if (parse_int_optstring(optarg, &_max_ports))
00176                     info.printf("Max number of ports: %u\n", _max_ports);
00177                 else
00178                 {
00179                     Err().printf("Invalid number of ports argument: %s\n", optarg);
00180                     return -1;
00181                 }
00182                 break;
00183             case 'q':
00184                 verbosity = Dbg::Quiet;
00185                 Dbg::set_verbosity(verbosity);
00186

```

```

00187         break;
00188     case 'v':
00189         verbosity = (verbosity < 1) | 1;
00190         Dbg::set_verbosity(verbosity);
00191         break;
00192     case 'D':
00193         set_verbosity(optarg);
00194         break;
00195     case 'm':
00196         info.printf("Option -m ignored to compatibility.\n");
00197         break;
00198     case 'M':
00199         _assign_mac = false;
00200         break;
00201     case 'd':
00202     {
00203         L4Re::Cap<L4Re::Dataspace> ds =
00204             L4Re::chkcap(L4Re::Env::env()->get_cap<L4Re::Dataspace>(optarg),
00205                 "Find a dataspace capability.\n");
00206         trusted_dataspaces->push_back(ds);
00207         break;
00208     }
00209     default:
00210         Err().printf("Unknown command line option '%c' (%d)\n", opt, opt);
00211         return -1;
00212     }
00213 }
00214 return 0;
00215 }
00216
00217 static Options options;
00218
00219 Options const *
00220 Options::get_options()
00221 { return &options; }
00222
00223 Options const *
00224 Options::parse_options(int argc, char **argv,
00225     std::shared_ptr<Ds_vector> trusted_dataspaces)
00226 {
00227     if (options.parse_cmd_line(argc, argv, trusted_dataspaces) < 0)
00228         return nullptr;
00229
00230     return &options;
00231 }

```

## 17.54 options.h

```

00001 /*
00002  * Copyright (C) 2016-2017, 2022, 2024 Kernkonzept GmbH.
00003  * Author(s): Jean Wolter <jean.wolter@kernkonzept.com>
00004  *             Manuel von Oltersdorff-Kalettka <manuel.kalettka@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <memory>
00011 #include <vector>
00012 #include <cerrno>
00013 #include <climits>
00014
00015 #include <l4/re/dataspace>
00016
00017 bool
00018 parse_int_optstring(char const *optstring, int *out);
00019
00020 class Options
00021 {
00022     using Ds_vector = std::vector<L4Re::Cap<L4Re::Dataspace>;
00023 public:
00024     int get_max_ports() const
00025     { return _max_ports; }
00026
00027     int get_virtq_max_num() const
00028     { return _virtq_max_num; }
00029
00030     int get_portq_max_num() const
00031     { return _portq_max_num; }
00032
00033     int get_request_timeout() const
00034     { return _request_timeout; }
00035 }

```

```

00036 int assign_mac() const
00037 { return _assign_mac; }
00038
00039 static Options const *
00040 parse_options(int argc, char **argv,
00041               std::shared_ptr<Ds_vector> trusted_dataspaces);
00042 static Options const *get_options();
00043
00044 private:
00045 int _max_ports = 5;
00046 int _virtq_max_num = 0x100; // default value for data queues
00047 int _portq_max_num = 50;    // default value for port queues
00048 int _request_timeout = 1 * 1000 * 1000; // default packet timeout 1 second
00049 bool _assign_mac = true;
00050
00051 int parse_cmd_line(int argc, char **argv,
00052                   std::shared_ptr<Ds_vector> trusted_dataspaces);
00053 };

```

## 17.55 port.h

```

00001 /*
00002  * Copyright (C) 2016-2018, 2020, 2022-2024 Kernkonzept GmbH.
00003  * Author(s): Jean Wolter <jean.wolter@kernkonzept.com>
00004  *            Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "request.h"
00011 #include "mac_addr.h"
00012 #include "vlan.h"
00013 #include "stats.h"
00014
00015 #include <cassert>
00016 #include <set>
00017 #include <vector>
00018
00023
00024 class Port_iface
00025 {
00026 protected:
00027     Virtio_net_switch::Port_statistics *_stats;
00028
00029 public:
00030     Port_iface(char const *name)
00031     {
00032         strncpy(_name, name, sizeof(_name));
00033         _name[sizeof(_name) - 1] = '\0';
00034 #ifdef CONFIG_VNS_STATS
00035         _stats = Switch_statistics::get_instance().allocate_port_statistics(name);
00036         if (!_stats)
00037             throw L4::Runtime_error(-L4_ENOMEM,
00038                                     "Could not allocate port statistics.\n");
00039 #endif
00040     }
00041
00042     virtual ~Port_iface()
00043     {
00044 #ifdef CONFIG_VNS_STATS
00045         _stats->in_use = false;
00046 #endif
00047     }
00048
00049     // delete copy and assignment
00050     Port_iface(Port_iface const &) = delete;
00051     Port_iface &operator = (Port_iface const &) = delete;
00052
00053     char const *get_name() const
00054     { return _name; }
00055
00056     l4_uint16_t get_vlan() const
00057     { return _vlan_id; }
00058
00059     inline bool is_trunk() const
00060     { return _vlan_id == VLAN_ID_TRUNK; }
00061
00062     inline bool is_native() const
00063     { return _vlan_id == VLAN_ID_NATIVE; }
00064
00065     inline bool is_access() const
00066     { return !is_trunk() && !is_native(); }

```



```

00067
00075 void set_vlan_access(l4_uint16_t id)
00076 {
00077     assert(vlan_valid_id(id));
00078     _vlan_id = id;
00079     _vlan_bloom_filter = 0;
00080     _vlan_ids.clear();
00081 }
00082
00092 void set_vlan_trunk(const std::vector<l4_uint16_t> &ids)
00093 {
00094     // bloom filter to quickly reject packets that do not belong to this port
00095     l4_uint32_t filter = 0;
00096
00097     _vlan_ids.clear();
00098     for (const auto id : ids)
00099     {
00100         assert(vlan_valid_id(id));
00101         filter |= vlan_bloom_hash(id);
00102         _vlan_ids.insert(id);
00103     }
00104
00105     _vlan_id = VLAN_ID_TRUNK;
00106     _vlan_bloom_filter = filter;
00107 }
00108
00112 void set_vlan_trunk_all()
00113 {
00114     _vlan_all = true;
00115     _vlan_id = VLAN_ID_TRUNK;
00116     _vlan_bloom_filter = -1;
00117 }
00118
00125 void set_monitor()
00126 {
00127     _vlan_id = VLAN_ID_TRUNK;
00128     _vlan_bloom_filter = 0;
00129 }
00130
00139 bool match_vlan(uint16_t id)
00140 {
00141     // Regular case native/access port
00142     if (id == _vlan_id)
00143         return true;
00144
00145     // This port participates in all VLANs
00146     if (_vlan_all)
00147         return true;
00148
00149     // Quick check: does port probably accept this VLAN?
00150     if ((_vlan_bloom_filter & vlan_bloom_hash(id)) == 0)
00151         return false;
00152
00153     return _vlan_ids.find(id) != _vlan_ids.end();
00154 }
00155
00162 inline Mac_addr mac() const
00163 { return _mac; }
00164
00165 Virtio_vlan_mangle create_vlan_mangle(Port_iface *src_port) const
00166 {
00167     Virtio_vlan_mangle mangle;
00168
00169     if (is_trunk())
00170     {
00171         /*
00172          * Add a VLAN tag only if the packet does not already have one (by
00173          * coming from another trunk port) or if the packet does not belong to
00174          * any VLAN (by coming from a native port). The latter case is only
00175          * relevant if this is a monitor port. Otherwise traffic from native
00176          * ports is never forwarded to trunk ports.
00177          */
00178         if (!src_port->is_trunk() && !src_port->is_native())
00179             mangle = Virtio_vlan_mangle::add(src_port->get_vlan());
00180     }
00181     else
00182     {
00183         /*
00184          * Remove VLAN tag only if the packet actually has one (by coming from a
00185          * trunk port).
00186          */
00187         if (src_port->is_trunk())
00188             mangle = Virtio_vlan_mangle::remove();
00189     }
00190     return mangle;
00191 }
00192 virtual void rx_notify_disable_and_remember() = 0;

```

```

00193     virtual void rx_notify_emit_and_enable() = 0;
00194
00195     virtual bool is_gone() const = 0;
00196
00197     // std::optional<Net_request> get_tx_request() = 0;
00198
00199     enum class Result
00200     {
00201         Delivered, Exception, Dropped,
00202     };
00203
00204     virtual Result handle_request(Port_iface *src_port,
00205                                   Net_transfer &src,
00206                                   l4_uint64_t *bytes_transferred) = 0;
00207
00208     void reschedule_pending_tx()
00209     { _pending_tx_reschedule->trigger(); }
00210
00211 protected:
00212     /*
00213      * VLAN related management information.
00214      *
00215      * A port may either be
00216      * - a native port (_vlan_id == VLAN_ID_NATIVE), or
00217      * - an access port (_vlan_id set accordingly), or
00218      * - a trunk port (_vlan_id == VLAN_ID_TRUNK, _vlan_bloom_filter and
00219      *   _vlan_ids populated accordingly, or _vlan_all == true).
00220      */
00221     l4_uint16_t _vlan_id = VLAN_ID_NATIVE; // VID for native/access port
00222     l4_uint32_t _vlan_bloom_filter = 0; // Bloom filter for trunk ports
00223     std::set<l4_uint16_t> _vlan_ids; // Authoritative list of trunk VLANs
00224     bool _vlan_all; // This port participates in all VLANs (ignoring _vlan_ids)
00225
00226     inline l4_uint32_t vlan_bloom_hash(l4_uint16_t vid)
00227     { return LUL << (vid & 31U); }
00228
00229     L4::Cap<L4::Irq> _pending_tx_reschedule;
00230
00231     Mac_addr _mac = Mac_addr(Mac_addr::Addr_unknown);
00232     char _name[20];
00233
00234 public:
00235 #ifdef CONFIG_VNS_STATS
00236     inline void stat_inc_tx_num()
00237     { _stats->tx_num++; }
00238     inline void stat_inc_tx_dropped()
00239     { _stats->tx_dropped++; }
00240     inline void stat_inc_tx_bytes(l4_uint64_t bytes)
00241     { _stats->tx_bytes += bytes; }
00242     inline void stat_inc_rx_num()
00243     { _stats->rx_num++; }
00244     inline void stat_inc_rx_dropped()
00245     { _stats->rx_dropped++; }
00246     inline void stat_inc_rx_bytes(l4_uint64_t bytes)
00247     { _stats->rx_bytes += bytes; }
00248 #else
00249     inline void stat_inc_tx_num()
00250     {}
00251     inline void stat_inc_tx_dropped()
00252     {}
00253     inline void stat_inc_tx_bytes(l4_uint64_t /*bytes*/)
00254     {}
00255     inline void stat_inc_rx_num()
00256     {}
00257     inline void stat_inc_rx_dropped()
00258     {}
00259     inline void stat_inc_rx_bytes(l4_uint64_t /*bytes*/)
00260     {}
00261 #endif
00262 };
00263

```

## 17.56 port\_ixl.h

```

00001 /*
00002  * Copyright (C) 2024 Kernkonzept GmbH.
00003  * Author(s): Georg Kotheimer <georg.kotheimer@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "port.h"

```

```

00010 #include "request_ixl.h"
00011
00012 #include <l4/ixl/device.h>
00013 #include <l4/ixl/memory.h>
00014
00015 #include <optional>
00016
00021
00022 class Ixl_port : public Port_iface
00023 {
00024 public:
00025     static constexpr unsigned Tx_batch_size = 32;
00026     static constexpr unsigned Num_bufs = 1024;
00027     static constexpr unsigned Buf_size = 2048;
00028     static constexpr l4_uint64_t Max_mem_size = 1ULL << 28;
00029
00030     Ixl_port(Ixl::Ixl_device *dev)
00031     : Port_iface(dev->get_driver_name().c_str()),
00032       _dev(dev),
00033       _mempool(*_dev, Num_bufs, Buf_size, Max_mem_size)
00034     {
00035         Ixl::mac_address mac_addr = _dev->get_mac_addr();
00036         _mac = Mac_addr(reinterpret_cast<char const *>(mac_addr.addr));
00037         #if CONFIG_VNS_STATS
00038         _mac.to_array(_stats->mac);
00039         #endif
00040     }
00041
00042     // OPTIMIZE: Could use this information for rx batching, i.e. collect while
00043     //             rx_notify is disabled, then flush the collected buffers when
00044     //             rx_notify is enabled again.
00045     void rx_notify_disable_and_remember() override {}
00046     void rx_notify_emit_and_enable() override {}
00047     bool is_gone() const override { return false; }
00048
00050     bool tx_work_pending()
00051     {
00052         fetch_tx_requests();
00053         return _tx_batch_idx < _tx_batch_len;
00054     }
00055
00057     std::optional<Ixl_net_request> get_tx_request()
00058     {
00059         fetch_tx_requests();
00060         if (_tx_batch_idx < _tx_batch_len)
00061             return std::make_optional<Ixl_net_request>(_tx_batch[_tx_batch_idx++]);
00062         else
00063             return std::nullopt;
00064     }
00065
00066     Result handle_request(Port_iface *src_port, Net_transfer &src,
00067                          l4_uint64_t *bytes_transferred) override
00068     {
00069         Virtio_vlan_mangle mangle = create_vlan_mangle(src_port);
00070
00071         Dbg trace(Dbg::Request, Dbg::Trace, "REQ-IXL");
00072         trace.printf("%s: Transfer request %p.\n", _name, src.req_id());
00073
00074         struct Ixl::pkt_buf *buf = _mempool.pkt_buf_alloc();
00075         if (!buf)
00076         {
00077             trace.printf("\tTransfer failed, out-of-memory, dropping.\n");
00078             return Result::Dropped;
00079         }
00080
00081         // NOTE: Currently, the switch does not offer checksum or segmentation
00082         //         offloading to its l4virtio clients, so it is fine to simply ignore
00083         //         the Virtio_net::Hdr of the request here.
00084
00085         // Copy the request to the pkt_buf.
00086         Buffer dst_buf(reinterpret_cast<char *>(buf->data),
00087                      Buf_size - offsetof(Ixl::pkt_buf, data));
00088         unsigned max_size = Buf_size - offsetof(Ixl::pkt_buf, data);
00089         for (;;)
00090         {
00091             try
00092             {
00093                 if (src.done())
00094                     // Request completely copied to destination.
00095                     break;
00096             }
00097             catch (L4virtio::Svr::Bad_descriptor &e)
00098             {
00099                 trace.printf("\tTransfer failed, bad descriptor exception, dropping.\n");
00100
00101                 // Handle partial transfers to destination port.
00102                 Ixl::pkt_buf_free(buf);

```

```

00103         throw;
00104     }
00105
00106     if (dst_buf.done())
00107     {
00108         trace.printf(
00109             "\tTransfer failed, exceeds max packet-size, dropping.\n");
00110         Ixl::pkt_buf_free(buf);
00111         return Result::Dropped;
00112     }
00113
00114     auto &src_buf = src.cur_buf();
00115     trace.printf("\tCopying %p#%p:%u (%x) -> %p#%p:%u (%x)\n",
00116         src_port, src_buf.pos, src_buf.left, src_buf.left,
00117         static_cast<Port_iface *>(this),
00118         dst_buf.pos, dst_buf.left, dst_buf.left);
00119
00120     mangle.copy_pkt(dst_buf, src_buf);
00121 }
00122 buf->size = max_size - dst_buf.left;
00123 *bytes_transferred = buf->size;
00124
00125 // Enqueue the pkt_buf at the device.
00126 if (_dev->tx_batch(0, &buf, 1) == 1)
00127 {
00128     trace.printf("\tTransfer queued at device.\n");
00129     return Result::Delivered;
00130 }
00131 else
00132 {
00133     trace.printf("\tTransfer failed, dropping.\n");
00134     Ixl::pkt_buf_free(buf);
00135     return Result::Dropped;
00136 }
00137 }
00138
00139 Ixl::Ixl_device *dev() { return _dev; }
00140
00141 private:
00142 void fetch_tx_requests()
00143 {
00144     if (_tx_batch_idx < _tx_batch_len)
00145         // Previous batch not yet fully processed.
00146         return;
00147
00148     // Batch receive, then cache in member array, to avoid frequent interactions
00149     // with the hardware.
00150     _tx_batch_len = _dev->rx_batch(0, _tx_batch, Tx_batch_size);
00151     _tx_batch_idx = 0;
00152 }
00153
00154 Ixl::Ixl_device *_dev;
00155 Ixl::Mempool _mempool;
00156 Ixl::pkt_buf *_tx_batch[Tx_batch_size];
00157 unsigned _tx_batch_idx = 0;
00158 unsigned _tx_batch_len = 0;
00159 };
00160

```

## 17.57 port\_l4virtio.h

```

00001 /*
00002  * Copyright (C) 2016-2017, 2020, 2022-2024 Kernkonzept GmbH.
00003  * Author(s): Jean Wolter <jean.wolter@kernkonzept.com>
00004  *             Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *             Georg Kotheimer <georg.kotheimer@kernkonzept.com>
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include "port.h"
00012 #include "request_l4virtio.h"
00013 #include "virtio_net.h"
00014
00015 #include <l4/cxx/pair>
00016
00017 #include <vector>
00018
00023
00036 class L4virtio_port : public Port_iface, public Virtio_net
00037 {
00038 public:

```

```

00042 explicit L4virtio_port(unsigned vq_max, unsigned num_ds, char const *name,
00043                          l4_uint8_t const *mac)
00044 : Port_iface(name), Virtio_net(vq_max)
00045 {
00046     init_mem_info(num_ds);
00047
00048     Features hf = _dev_config.host_features(0);
00049     if (mac)
00050     {
00051         _mac = Mac_addr((char const *)mac);
00052         memcpy((void *)_dev_config.priv_config()->mac, mac,
00053               sizeof(_dev_config.priv_config()->mac));
00054
00055         hf.mac() = true;
00056         Dbg d(Dbg::Port, Dbg::Info);
00057         d.cprintf("%s: Adding Mac '", _name);
00058         _mac.print(d);
00059         d.cprintf("' to host features to %x\n", hf.raw);
00060     }
00061     _dev_config.host_features(0) = hf.raw;
00062     _dev_config.reset_hdr();
00063     Dbg(Dbg::Port, Dbg::Info)
00064         .printf("%s: Set host features to %x\n", _name,
00065               _dev_config.host_features(0));
00066 #if CONFIG_VNS_STATS
00067     _mac.to_array(_stats->mac);
00068 #endif
00069 }
00070
00071 void rx_notify_disable_and_remember() override
00072 {
00073     kick_disable_and_remember();
00074 }
00075
00076 void rx_notify_emit_and_enable() override
00077 {
00078     kick_emit_and_enable();
00079 }
00080
00081 bool is_gone() const override
00082 {
00083     return obj_cap() && !obj_cap().validate().label();
00084 }
00085
00086 bool tx_work_pending() const
00087 {
00088     return L4_LIKELY(tx_q()->ready()) && tx_q()->desc_avail();
00089 }
00090
00091
00092 std::optional<Virtio_net_request> get_tx_request()
00093 {
00094     return Virtio_net_request::get_request(this, tx_q());
00095 }
00096
00097
00103 void drop_requests()
00104 { Virtio_net_request::drop_requests(this, tx_q()); }
00105
00106 Result handle_request(Port_iface *src_port, Net_transfer &src,
00107                      l4_uint64_t *bytes_transferred) override
00108 {
00109     Virtio_vlan_mangle mangle = create_vlan_mangle(src_port);
00110
00111     Dbg trace(Dbg::Request, Dbg::Trace, "REQ-VIO");
00112     trace.printf("%s: Transfer request %p.\n", _name, src.req_id());
00113
00114     Buffer dst;
00115     int total = 0;
00116     l4_uint16_t num_merged = 0;
00117     l4_uint64_t total_merged = 0;
00118     typedef cxx::Pair<L4virtio::Svr::Virtqueue::Head_desc, l4_uint32_t> Consumed_entry;
00119     std::vector<Consumed_entry> consumed;
00120
00121     Virtio_net *dst_dev = this;
00122     Virtqueue *dst_queue = rx_q();
00123     L4virtio::Svr::Virtqueue::Head_desc dst_head;
00124     L4virtio::Svr::Request_processor dst_req_proc;
00125     Virtio_net::Hdr *dst_header = nullptr;
00126
00127     for (;;)
00128     {
00129         try
00130         {
00131             if (src.done())
00132                 // Request completely copied to destination.
00133                 break;
00134         }
00135         catch (L4virtio::Svr::Bad_descriptor &e)

```

```

00136     {
00137         trace.printf("\tTransfer failed, bad descriptor exception, dropping.\n");
00138
00139         // Handle partial transfers to destination port.
00140         if (!consumed.empty())
00141             // Partial transfer, rewind to before first descriptor of transfer.
00142             dst_queue->rewind_avail(consumed.at(0).first);
00143         else if (dst_head)
00144             // Partial transfer, still at first _dst_head.
00145             dst_queue->rewind_avail(dst_head);
00146         throw;
00147     }
00148
00149     /* The source data structures are already initialized, the header
00150     is consumed and src stands at the very first real buffer.
00151     Initialize the target data structures if necessary and fill the
00152     header. */
00153     if (!dst_head)
00154     {
00155         if (!dst_queue->ready())
00156             return Result::Dropped;
00157
00158         auto r = dst_queue->next_avail();
00159
00160         if (L4_UNLIKELY(!r))
00161         {
00162             trace.printf("\tTransfer failed, destination queue depleted, dropping.\n");
00163             // Abort incomplete transfer.
00164             if (!consumed.empty())
00165                 dst_queue->rewind_avail(consumed.front().first);
00166             return Result::Dropped;
00167         }
00168
00169         try
00170         {
00171             dst_head = dst_req_proc.start(dst_dev->mem_info(), r, &dst);
00172         }
00173         catch (L4virtio::Svr::Bad_descriptor &e)
00174         {
00175             Dbg(Dbg::Request, Dbg::Warn, "REQ")
00176             .printf("%s: bad descriptor exception: %s - %i"
00177                 " -- signal device error in destination device %p.\n",
00178                 __PRETTY_FUNCTION__, e.message(), e.error, dst_dev);
00179
00180             dst_dev->device_error();
00181             return Result::Exception; // Must not touch the dst queues anymore.
00182         }
00183
00184         if (!dst_header)
00185         {
00186             if (dst.left < sizeof(Virtio_net::Hdr))
00187                 throw L4::Runtime_error(-L4_EINVAL,
00188                     "Target buffer too small for header");
00189             dst_header = reinterpret_cast<Virtio_net::Hdr *>(dst.pos);
00190             trace.printf("\tCopying header to %p (size: %u)\n",
00191                 dst.pos, dst.left);
00192
00193             /*
00194             * Header and csum offloading/general segmentation offloading
00195             *
00196             * We just copy the original header from source to
00197             * destination and have to consider three different
00198             * cases:
00199             * - no flags are set
00200             *   - we got a packet that is completely checksummed
00201             *     and correctly fragmented, there is nothing to
00202             *     do other than copying.
00203             * - virtio_net_hdr_f_needs_csum set
00204             *   - the packet is partially checksummed; if we would
00205             *     send the packet out on the wire we would have
00206             *     to calculate checksums now. But here we rely on
00207             *     the ability of our guest to handle partially
00208             *     checksummed packets and simply delegate the
00209             *     checksum calculation to them.
00210             * - gso_type != gso_none
00211             *   - the packet needs to be segmented; if we would
00212             *     send it out on the wire we would have to
00213             *     segment it now. But again we rely on the
00214             *     ability of our guest to handle gso
00215             *
00216             * We currently assume that our guests negotiated
00217             * virtio_net_f_guest_*, this needs to be checked in
00218             * the future.
00219             *
00220             * We also discussed the usage of
00221             * virtio_net_hdr_f_data_valid to remove the need to
00222             * checksum packets at all. But since our clients send
00223             * partially checksummed packets anyway the only

```

```

00223         * interesting case would be a packet without
00224         * net_hdr_f_needs_checksum set. In that case we would
00225         * signal that we checked the checksum and the
00226         * checksum is actually correct. Since we do not know
00227         * the origin of the packet (it could have been send
00228         * by an external node and could have been routed to
00229         * u) we can not signal this without actually
00230         * verifying the checksum. Otherwise a packet with an
00231         * invalid checksum could be successfully delivered.
00232         */
00233         total = sizeof(Virtio_net::Hdr);
00234         src.copy_header(dst_header);
00235         mangle.rewrite_hdr(dst_header);
00236         dst.skip(total);
00237     }
00238     ++num_merged;
00239 }
00240
00241 bool has_dst_buffer = !dst.done();
00242 if (!has_dst_buffer)
00243     try
00244     {
00245         // The current dst buffer is full, try to get next chained buffer.
00246         has_dst_buffer = dst_req_proc.next(dst_dev->mem_info(), &dst);
00247     }
00248     catch (L4virtio::Svr::Bad_descriptor &e)
00249     {
00250         Dbg(Dbg::Request, Dbg::Warn, "REQ")
00251         .printf("%s: bad descriptor exception: %s - %i"
00252             " -- signal device error in destination device %p.\n",
00253             __PRETTY_FUNCTION__, e.message(), e.error, dst_dev);
00254         dst_dev->device_error();
00255         return Result::Exception; // Must not touch the dst queues anymore.
00256     }
00257
00258 if (has_dst_buffer)
00259 {
00260     auto &src_buf = src.cur_buf();
00261     trace.printf("\tCopying %p#%p:%u (%x) -> %p#%p:%u (%x)\n",
00262         src_port, src_buf.pos, src_buf.left, src_buf.left,
00263         static_cast<Port_iface*>(this),
00264         dst.pos, dst.left, dst.left);
00265
00266     total += mangle.copy_pkt(dst, src_buf);
00267 }
00268 else if (negotiated_features().mrg_rxbuf())
00269 {
00270     // save descriptor information for later
00271     trace.printf("\tSaving descriptor for later\n");
00272     consumed.push_back(Consumed_entry(dst_head, total));
00273     total_merged += total;
00274     total = 0;
00275     dst_head = L4virtio::Svr::Virtqueue::Head_desc();
00276 }
00277 else
00278 {
00279     trace.printf("\tTransfer failed, destination buffer too small, dropping.\n");
00280     // Abort incomplete transfer.
00281     dst_queue->rewind_avail(dst_head);
00282     return Result::Dropped;
00283 }
00284 }
00285
00286 /*
00287 * Finalize the Request delivery. Call `finish()` on the destination
00288 * port's receive queue, which will result in triggering the destination
00289 * client IRQ.
00290 */
00291
00292 if (!dst_header)
00293 {
00294     if (!total)
00295         trace.printf("\tTransfer - not started yet, dropping\n");
00296     return Result::Dropped;
00297 }
00298
00299 if (consumed.empty())
00300 {
00301     assert(dst_head);
00302     assert(num_merged == 1);
00303     trace.printf("\tTransfer - Invoke dst_queue->finish()\n");
00304     dst_header->num_buffers = 1;
00305     dst_queue->finish(dst_head, dst_dev, total);
00306     *bytes_transferred = total;
00307 }
00308 else
00309 {

```

```

00310         assert(dst_head);
00311         dst_header->num_buffers = num_merged;
00312         consumed.push_back(Consumed_entry(dst_head, total));
00313         trace.printf("\tTransfer - Invoke dst_queue->finish(iter)\n");
00314         *bytes_transferred = total + total_merged;
00315         dst_queue->finish(consumed.begin(), consumed.end(), dst_dev);
00316     }
00317     return Result::Delivered;
00318 }
00319 };
00320

```

## 17.58 request.h

```

00001 /*
00002  * Copyright (C) 2016-2017, 2020, 2022, 2024 Kernkonzept GmbH.
00003  * Author(s): Jean Wolter <jean.wolter@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "mac_addr.h"
00010 #include "virtio_net.h"
00011 #include "virtio_net_buffer.h"
00012 #include "vlan.h"
00013
00014 #include <l4/l4virtio/server/virtio>
00015
00016
00021
00033 class Net_transfer
00034 {
00035 public:
00036     virtual ~Net_transfer() = default;
00037
00041     void const *req_id() const { return _req_id; }
00042
00046     virtual void copy_header(Virtio_net::Hdr *dst_header) const = 0;
00047
00054     Buffer &cur_buf() { return _cur_buf; }
00055
00065     virtual bool done() = 0;
00066
00067 protected:
00068     Buffer _cur_buf;
00069     void const *_req_id;
00070 };
00071
00072 class Net_request
00073 {
00074 public:
00075
00076     bool has_vlan() const
00077     {
00078         if (!_pkt.pos || _pkt.left < 14)
00079             return false;
00080
00081         uint8_t *p = reinterpret_cast<uint8_t *>(_pkt.pos);
00082         return p[12] == 0x81U && p[13] == 0x00U;
00083     }
00084
00085     uint16_t vlan_id() const
00086     {
00087         if (!has_vlan() || _pkt.left < 16)
00088             return VLAN_ID_NATIVE;
00089
00090         uint8_t *p = reinterpret_cast<uint8_t *>(_pkt.pos);
00091         return (uint16_t){p[14]} << 8 | p[15] & 0xffffU;
00092     }
00093
00105     uint8_t const *buffer(size_t *size) const
00106     {
00107         *size = _pkt.left;
00108         return reinterpret_cast<uint8_t const *>(_pkt.pos);
00109     }
00110
00111     void dump_pkt() const
00112     {
00113         Dbg pkt_debug(Dbg::Packet, Dbg::Debug, "PKT");
00114         if (pkt_debug.is_active())
00115         {
00116             //pkt_debug.cprintf("\t");

```



```

00117         //src_mac().print(pkt_debug);
00118         //pkt_debug.cprintf(" -> ");
00119         //dst_mac().print(pkt_debug);
00120         //pkt_debug.cprintf("\n");
00121
00122         Dbg pkt_trace(Dbg::Packet, Dbg::Trace, "PKT");
00123         if (pkt_trace.is_active() && _pkt.left >= 14)
00124         {
00125             uint8_t const *packet = reinterpret_cast<uint8_t const *>(_pkt.pos);
00126             pkt_trace.cprintf("\n\tEthertype: ");
00127             uint16_t ether_type = uint16_t{packet[12]} << 8 | packet[13];
00128             char const *protocol;
00129             switch (ether_type)
00130             {
00131                 case 0x0800: protocol = "IPv4"; break;
00132                 case 0x0806: protocol = "ARP"; break;
00133                 case 0x8100: protocol = "Vlan"; break;
00134                 case 0x86dd: protocol = "IPv6"; break;
00135                 case 0x8863: protocol = "PPPoE Discovery"; break;
00136                 case 0x8864: protocol = "PPPoE Session"; break;
00137                 default: protocol = nullptr;
00138             }
00139             if (protocol)
00140                 pkt_trace.cprintf("%s\n", protocol);
00141             else
00142                 pkt_trace.cprintf("%04x\n", ether_type);
00143         }
00144     }
00145 }
00146
00147 protected:
00148     Buffer _pkt;
00149 };
00150

```

## 17.59 request.h

```

00001 /*
00002  * Copyright (C) 2019-2020, 2024 Kernkonzept GmbH.
00003  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 namespace Block_device {
00010
00014 struct Pending_request
00015 {
00016     virtual ~Pending_request() = 0;
00017
00028     virtual int handle_request() = 0;
00029
00036     virtual void fail_request() = 0;
00037 };
00038
00039 inline Pending_request::~Pending_request() = default;
00040
00041 } // namespace

```

## 17.60 request\_ixl.h

```

00001 /*
00002  * Copyright (C) 2024 Kernkonzept GmbH.
00003  * Author(s): Georg Kotheimer <georg.kotheimer@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "port.h"
00010 #include "request.h"
00011
00012 #include <i4/ixl/memory.h>
00013
00014 #include <utility>
00015
00020

```

```

00021 class Ixl_net_request final : public Net_request
00022 {
00023 public:
00024 class Ixl_net_transfer final : public Net_transfer
00025 {
00026 public:
00027 explicit Ixl_net_transfer(Ixl_net_request const &request)
00028 : _request(request)
00029 {
00030     _cur_buf = Buffer(reinterpret_cast<char *>(request.buf()->data),
00031                     request.buf()->size);
00032     _req_id = _request.buf();
00033 }
00034
00035 // delete copy constructor and copy assignment operator
00036 Ixl_net_transfer(Ixl_net_transfer const &) = delete;
00037 Ixl_net_transfer &operator = (Ixl_net_transfer const &) = delete;
00038
00039 void copy_header(Virtio_net::Hdr *dst_header) const override
00040 {
00041     dst_header->flags.data_valid() = 0;
00042     dst_header->flags.need_csum() = 0;
00043     dst_header->gso_type = 0; // GSO_NONE
00044     dst_header->hdr_len = sizeof(Virtio_net::Hdr);
00045     dst_header->gso_size = 0;
00046     dst_header->csum_start = 0;
00047     dst_header->csum_offset = 0;
00048     dst_header->num_buffers = 1;
00049 }
00050
00051 bool done() override { return _cur_buf.done(); }
00052
00053 private:
00054     Ixl_net_request const &_request;
00055 };
00056
00057 void dump_request(Port_iface *port) const
00058 {
00059     Dbg debug(Dbg::Request, Dbg::Debug, "REQ-IXL");
00060     if (debug.is_active())
00061     {
00062         debug.printf("%s: Next packet: %p - %x bytes\n",
00063                     port->get_name(), _pkt.pos, _pkt.left);
00064     }
00065     dump_pkt();
00066 }
00067
00068 explicit Ixl_net_request(Ixl::pkt_buf *buf) : _buf(buf)
00069 {
00070     _pkt = Buffer(reinterpret_cast<char *>(buf->data), buf->size);
00071 }
00072
00073 // delete copy constructor and copy assignment operator
00074 Ixl_net_request(Ixl_net_request const &) = delete;
00075 Ixl_net_request &operator=(Ixl_net_request const &) = delete;
00076
00077 // define move constructor and copy assignment operator
00078 Ixl_net_request(Ixl_net_request &&other)
00079 : _buf(other._buf)
00080 {
00081     _pkt = std::move(other._pkt);
00082
00083     // Invalidate other.
00084     other._buf = nullptr;
00085 }
00086
00087 Ixl_net_request &operator=(Ixl_net_request &&other)
00088 {
00089     // Invalidate self.
00090     if (_buf != nullptr)
00091         Ixl::pkt_buf_free(_buf);
00092
00093     _buf = other._buf;
00094     _pkt = std::move(other._pkt);
00095
00096     // Invalidate other.
00097     other._buf = nullptr;
00098
00099     return *this;
00100 }
00101
00102 ~Ixl_net_request()
00103 {
00104     if (_buf != nullptr)
00105     {
00106         Ixl::pkt_buf_free(_buf);
00107         _buf = nullptr;
00108     }
00109 }

```

```

00108     }
00109 }
00110
00112 Mac_addr dst_mac() const
00113 {
00114     return (_pkt.pos && _pkt.left >= Mac_addr::Addr_length)
00115         ? Mac_addr::from_uncached(_pkt.pos)
00116         : Mac_addr(Mac_addr::Addr_unknown);
00117 }
00118
00120 Mac_addr src_mac() const
00121 {
00122     return (_pkt.pos && _pkt.left >= Mac_addr::Addr_length * 2)
00123         ? Mac_addr::from_uncached(_pkt.pos + Mac_addr::Addr_length)
00124         : Mac_addr(Mac_addr::Addr_unknown);
00125 }
00126
00127 Ixl::pkt_buf *buf() const { return _buf; }
00128
00129 Ixl_net_transfer transfer_src() const
00130 { return Ixl_net_transfer(*this); }
00131
00132 private:
00133     Ixl::pkt_buf *_buf;
00134 };
00135

```

## 17.61 request\_l4virtio.h

```

00001 /*
00002  * Copyright (C) 2016-2017, 2020, 2022, 2024 Kernkonzept GmbH.
00003  * Author(s): Jean Wolter <jean.wolter@kernkonzept.com>
00004  *             Georg Kotheimer <georg.kotheimer@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "debug.h"
00011 #include "port.h"
00012 #include "request.h"
00013 #include "virtio_net.h"
00014
00015 #include <l4/l4virtio/server/virtio>
00016 #include <l4/util/assert.h>
00017
00018 #include <optional>
00019 #include <utility>
00020
00025
00035 class Virtio_net_request final : public Net_request
00036 {
00037 public:
00038     class Virtio_net_transfer final : public Net_transfer
00039     {
00040     public:
00041         explicit Virtio_net_transfer(Virtio_net_request const &request)
00042             : _request(request),
00043               // We already looked at the very first buffer to find the target of the
00044               // packet. The request processor of the "parent request" contains the
00045               // current state of the transaction up to this point. Since there might be
00046               // more than one target for the request we have to keep track of our own
00047               // state and need our own request processor instance, which will be
00048               // initialized using the current state of the "parent request".
00049               _req_proc(_request.get_request_processor())
00050         {
00051             // The buffer descriptors used for this transaction and the amount of bytes
00052             // copied to the current target descriptor.
00053             _cur_buf = request.first_buffer();
00054             _req_id = _request.header();
00055         }
00056
00057         // delete copy constructor and copy assignment operator
00058         Virtio_net_transfer(Virtio_net_transfer const &) = delete;
00059         Virtio_net_transfer &operator = (Virtio_net_transfer const &) = delete;
00060
00061         void copy_header(Virtio_net::Hdr *dst_header) const override
00062         {
00063             memcpy(dst_header, _request.header(), sizeof(Virtio_net::Hdr));
00064         }
00065
00066         bool done() override
00067         {

```

```

00068         return _cur_buf.done() && !_req_proc.next(_request.dev()->mem_info(), &_cur_buf);
00069     }
00070
00071 private:
00072     Virtio_net_request const &_request;
00073     L4virtio::Svr::Request_processor _req_proc;
00074 };
00075
00076 void dump_request(Port_iface *port) const
00077 {
00078     Dbg debug(Dbg::Request, Dbg::Debug, "REQ-VIO");
00079     if (debug.is_active())
00080     {
00081         debug.printf("%s: Next packet: %p:%p - %x bytes\n",
00082             port->get_name(), _header, _pkt.pos, _pkt.left);
00083         if (_header->flags.raw || _header->gso_type)
00084         {
00085             debug.cprintf("flags:\t%x\n\t"
00086                 "gso_type:\t%x\n\t"
00087                 "header len:\t%x\n\t"
00088                 "gso size:\t%x\n\t"
00089                 "csum start:\t%x\n\t"
00090                 "csum offset:\t%x\n\t"
00091                 "\tnum buffer:\t%x\n\t",
00092                 _header->flags.raw,
00093                 _header->gso_type, _header->hdr_len,
00094                 _header->gso_size,
00095                 _header->csum_start, _header->csum_offset,
00096                 _header->num_buffers);
00097         }
00098     }
00099     dump_pkt();
00100 }
00101
00102 // delete copy constructor and copy assignment operator
00103 Virtio_net_request(Virtio_net_request const &) = delete;
00104 Virtio_net_request &operator = (Virtio_net_request const &) = delete;
00105
00106 // define move constructor and copy assignment operator
00107 Virtio_net_request(Virtio_net_request &&other)
00108 : _dev(other._dev),
00109   _queue(other._queue),
00110   _head(std::move(other._head)),
00111   _req_proc(std::move(other._req_proc)),
00112   _header(other._header)
00113 {
00114     _pkt = std::move(other._pkt);
00115
00116     // Invalidate other.
00117     other._queue = nullptr;
00118 }
00119
00120 Virtio_net_request &operator = (Virtio_net_request &&other)
00121 {
00122     // Invalidate self.
00123     finish();
00124
00125     _dev = other._dev;
00126     _queue = other._queue;
00127     _head = std::move(other._head);
00128     _req_proc = std::move(other._req_proc);
00129     _header = other._header;
00130     _pkt = std::move(other._pkt);
00131
00132     // Invalidate other.
00133     other._queue = nullptr;
00134
00135     return *this;
00136 }
00137
00138 Virtio_net_request(Virtio_net *dev, L4virtio::Svr::Virtqueue *queue,
00139     L4virtio::Svr::Virtqueue::Request const &req)
00140 : _dev(dev), _queue(queue)
00141 {
00142     _head = _req_proc.start(_dev->mem_info(), req, &_pkt);
00143
00144     _header = (Virtio_net::Hdr *)_pkt.pos;
00145     L4_uint32_t skipped = _pkt.skip(sizeof(Virtio_net::Hdr));
00146
00147     if (L4_UNLIKELY( (skipped != sizeof(Virtio_net::Hdr))
00148         || (_pkt.done() && !_next_buffer(&_pkt))))
00149     {
00150         _header = 0;
00151         Dbg(Dbg::Queue, Dbg::Warn).printf("Invalid request\n");
00152         return;
00153     }
00154 }

```

```

00155
00156 ~Virtio_net_request()
00157 { finish(); }
00158
00159 bool valid() const
00160 { return _header != 0; }
00161
00172 static void drop_requests(Virtio_net *dev,
00173                           L4virtio::Svr::Virtqueue *queue)
00174 {
00175     if (L4_UNLIKELY(!queue->ready()))
00176         return;
00177
00178     if (queue->desc_avail())
00179         Dbg(Dbg::Request, Dbg::Debug)
00180             .printf("Dropping incoming packets on monitor port\n");
00181
00182     L4virtio::Svr::Request_processor req_proc;
00183     Buffer pkt;
00184
00185     while (auto req = queue->next_avail())
00186     {
00187         auto head = req_proc.start(dev->mem_info(), req, &pkt);
00188         queue->finish(head, dev, 0);
00189     }
00190 }
00191
00198 static std::optional<Virtio_net_request>
00199 get_request(Virtio_net *dev, L4virtio::Svr::Virtqueue *queue)
00200 {
00201     if (L4_UNLIKELY(!queue->ready()))
00202         return std::nullopt;
00203
00204     if (auto r = queue->next_avail())
00205     {
00206         // Virtio_net_request keeps "a lot of internal state",
00207         // therefore we create the object before creating the
00208         // state.
00209         // We might check later on whether it is possible to
00210         // save the state when we actually have to because a
00211         // transfer is blocking on a port.
00212         auto request = Virtio_net_request(dev, queue, r);
00213         if (request.valid())
00214             return request;
00215     }
00216     return std::nullopt;
00217 }
00218
00219 Buffer const &first_buffer() const
00220 { return _pkt; }
00221
00222 Virtio_net::Hdr const *header() const
00223 { return _header; }
00224
00225 L4virtio::Svr::Request_processor const &get_request_processor() const
00226 { return _req_proc; }
00227
00228 Virtio_net const *dev() const
00229 { return _dev; }
00230
00231 Virtio_net_transfer transfer_src() const
00232 { return Virtio_net_transfer(*this); }
00233
00235 Mac_addr dst_mac() const
00236 {
00237     return (_pkt.pos && _pkt.left >= Mac_addr::Addr_length)
00238         ? Mac_addr(_pkt.pos)
00239         : Mac_addr(Mac_addr::Addr_unknown);
00240 }
00241
00243 Mac_addr src_mac() const
00244 {
00245     return (_pkt.pos && _pkt.left >= Mac_addr::Addr_length * 2)
00246         ? Mac_addr(_pkt.pos + Mac_addr::Addr_length)
00247         : Mac_addr(Mac_addr::Addr_unknown);
00248 }
00249
00250 private:
00252 /* needed for Virtqueue::finish() */
00254 Virtio_net *_dev;
00256 L4virtio::Svr::Virtqueue *_queue;
00257 L4virtio::Svr::Virtqueue::Head_desc _head;
00258
00259 /* the actual request processor, encapsulates the decoding of the request */
00260 L4virtio::Svr::Request_processor _req_proc;
00261

```

```

00262  /* A request to the virtio net layer consists of one or more buffers
00263      containing the Virtio_net::Hdr and the actual packet. To make a
00264      switching decision we need to be able to look at the packet while
00265      still being able access the Virtio_net::Hdr for the actual copy
00266      operation. Therefore we keep track of two locations, the header
00267      location and the start of the packet (which might be in a
00268      different buffer) */
00269  Virtio_net::Hdr *_header;
00270
00271  bool _next_buffer(Buffer *buf)
00272  { return _req_proc.next(_dev->mem_info(), buf); }
00273
00274  void finish()
00275  {
00276      if (_queue == nullptr || !_queue->ready())
00277          return;
00278
00279      Dbg(Dbg::Virtio, Dbg::Trace).printf("%s(%p)\n", __PRETTY_FUNCTION__, this);
00280      _queue->finish(_head, _dev, 0);
00281      _queue = nullptr;
00282  }
00283  };
00284
00285
00286
00287
00288
00289
00290

```

## 17.62 stats.h

```

00001 #include <l4/re/env>
00002 #include <l4/re/dataspace>
00003 #include <l4/re/error_helper>
00004 #include <l4/re/util/cap_alloc>
00005 #include <l4/virtio-net-switch/stats.h>
00006
00007 class Switch_statistics
00008 {
00009 private:
00010     L4Re::Util::Ref_cap<L4Re::Dataspace>::Cap_ds;
00011     Virtio_net_switch::Statistics *_stats;
00012     bool _initialized = false;
00013
00014     Switch_statistics() {}
00015
00016     ~Switch_statistics()
00017     {
00018         if (_initialized)
00019             L4Re::Env::env()->rm()->detach(reinterpret_cast<l4_addr_t>(_stats), 0);
00020     }
00021
00022     l4_size_t _size;
00023
00024 public:
00025     Virtio_net_switch::Statistics *stats()
00026     {
00027         if (_initialized)
00028             return _stats;
00029         else
00030             throw L4::Runtime_error(-L4_EAGAIN, "Statistics not set up.");
00031     }
00032
00033     static Switch_statistics& get_instance()
00034     {
00035         static Switch_statistics instance;
00036         return instance;
00037     }
00038
00039     void initialize(l4_uint64_t num_max_ports)
00040     {
00041         _size = l4_round_page(sizeof(Virtio_net_switch::Statistics)
00042                                + sizeof(Virtio_net_switch::Port_statistics) * num_max_ports);
00043         void *addr = malloc(_size);
00044         if (!addr)
00045             throw L4::Runtime_error(-L4_ENOMEM,
00046                                     "Could not allocate statistics memory.");
00047
00048         memset(addr, 0, _size);
00049         _stats = reinterpret_cast<Virtio_net_switch::Statistics *>(addr);
00050         _initialized = true;
00051         _stats->max_ports = num_max_ports;
00052     }
00053
00054     Virtio_net_switch::Port_statistics *
00055     allocate_port_statistics(char const* name)
00056     {
00057         for (unsigned i = 0; i < _stats->max_ports; ++i)

```

```

00058     {
00059         if (!_stats->port_stats[i].in_use)
00060         {
00061             memset(reinterpret_cast<void*>(&_stats->port_stats[i]), 0,
00062                 sizeof(Virtio_net_switch::Port_statistics));
00063             _stats->port_stats[i].in_use = 1;
00064             size_t len = std::min(strlen(name), sizeof(_stats->port_stats[i].name) - 1);
00065             memcpy(_stats->port_stats[i].name, name, len);
00066             _stats->port_stats[i].name[len] = '\\0';
00067             _stats->age++;
00068             return &_stats->port_stats[i];
00069         }
00070     }
00071     return nullptr;
00072 }
00073
00074 inline l4_size_t size()
00075 { return _size; }
00076
00077 Switch_statistics(Switch_statistics const&) = delete;
00078 void operator=(Switch_statistics const &) = delete;
00079 };

```

## 17.63 switch.cc

```

00001 /*
00002  * Copyright (C) 2016-2018, 2020, 2023-2024 Kernkonzept GmbH.
00003  * Author(s): Jean Wolter <jean.wolter@kernkonzept.com>
00004  *             Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #include "debug.h"
00009 #include "switch.h"
00010 #include "filter.h"
00011
00012 Virtio_switch::Virtio_switch(unsigned max_ports)
00013 : _ports(new Port_iface *[max_ports]()),
00014   _max_ports(max_ports)
00015 {
00016 }
00017
00018 int
00019 Virtio_switch::lookup_free_slot()
00020 {
00021     for (unsigned idx = 0; idx < _max_ports; ++idx)
00022         if (!_ports[idx])
00023             return idx;
00024     return -1;
00025 }
00026
00027 bool
00028 Virtio_switch::add_port(Port_iface *port)
00029 {
00030     if (!port->mac().is_unknown())
00031         for (unsigned idx = 0; idx < _max_ports; ++idx)
00032             if (_ports[idx] && _ports[idx]->mac() == port->mac())
00033             {
00034                 Dbg(Dbg::Port, Dbg::Warn)
00035                     .printf("Rejecting port '%s'. MAC address already in use.\n",
00036                         port->get_name());
00037                 return false;
00038             }
00039     int idx = lookup_free_slot();
00040     if (idx < 0)
00041         return false;
00042     unsigned uidx = static_cast<unsigned>(idx);
00043     _ports[uidx] = port;
00044     if (_max_used == uidx)
00045         ++_max_used;
00046     return true;
00047 }
00048
00049 bool
00050 Virtio_switch::add_monitor_port(Port_iface *port)
00051 {
00052     if (!_monitor)
00053         _monitor = port;
00054 }

```

```

00059         return true;
00060     }
00061
00062     Dbg(Dbg::Port, Dbg::Warn).printf("'%' already defined as monitor port,"
00063         " rejecting monitor port '%s'\n",
00064         _monitor->get_name(), port->get_name());
00065     return false;
00066 }
00067
00068 void
00069 Virtio_switch::check_ports()
00070 {
00071     for (unsigned idx = 0; idx < _max_used; ++idx)
00072     {
00073         Port_iface *port = _ports[idx];
00074         if (port && port->is_gone())
00075         {
00076             Dbg(Dbg::Port, Dbg::Info)
00077                 .printf("Client on port %p has gone. Deleting...\n", port);
00078
00079             _ports[idx] = nullptr;
00080             if (idx == _max_used-1)
00081                 --_max_used;
00082
00083             _mac_table.flush(port);
00084             delete(port);
00085         }
00086     }
00087
00088     if (_monitor && _monitor->is_gone())
00089     {
00090         delete(_monitor);
00091         _monitor = nullptr;
00092     }
00093 }
00094
00095 template<typename REQ>
00096 void
00097 Virtio_switch::handle_tx_request(Port_iface *port, REQ const &request)
00098 {
00099     // Trunk ports are required to have a VLAN tag and only accept packets that
00100     // belong to a configured VLAN.
00101     if (port->is_trunk() && !port->match_vlan(request.vlan_id()))
00102     {
00103         // Drop packet.
00104         port->stat_inc_tx_dropped();
00105         return;
00106     }
00107
00108     // Access ports must not be VLAN tagged to prevent double tagging attacks.
00109     if (port->is_access() && request.has_vlan())
00110     {
00111         // Drop packet.
00112         port->stat_inc_tx_dropped();
00113         return;
00114     }
00115
00116     auto handle_request = [](Port_iface *dst_port, Port_iface *src_port,
00117         REQ const &req)
00118     {
00119         auto transfer_src = req.transfer_src();
00120         l4_uint64_t bytes;
00121         auto res = dst_port->handle_request(src_port, transfer_src, &bytes);
00122         switch (res)
00123         {
00124             case Port_iface::Result::Delivered:
00125                 dst_port->stat_inc_tx_num();
00126                 dst_port->stat_inc_tx_bytes(bytes);
00127                 src_port->stat_inc_rx_num();
00128                 src_port->stat_inc_rx_bytes(bytes);
00129                 break;
00130             case Port_iface::Result::Dropped:
00131                 [[fallthrough]];
00132             case Port_iface::Result::Exception:
00133                 [[fallthrough]];
00134             default:
00135                 dst_port->stat_inc_tx_dropped();
00136                 break;
00137         }
00138     };
00139
00140     Mac_addr src = request.src_mac();
00141
00142     auto dst = request.dst_mac();
00143     bool is_broadcast = dst.is_broadcast();
00144     uint16_t vlan = request.has_vlan() ? request.vlan_id() : port->get_vlan();
00145     _mac_table.learn(src, port, vlan);

```



```

00146     if (L4_LIKELY(!is_broadcast))
00147     {
00148         auto *target = _mac_table.lookup(dst, vlan);
00149         if (target)
00150         {
00151             // Do not send packets to the port they came in; they might
00152             // be sent to us by another switch which does not know how
00153             // to reach the target.
00154             if (target != port)
00155             {
00156                 handle_request(target, port, request);
00157                 if (_monitor && !filter_request(request))
00158                     handle_request(_monitor, port, request);
00159             }
00160             return;
00161         }
00162     }
00163
00164     // It is either a broadcast or an unknown destination - send to all
00165     // known ports except the source port
00166     for (unsigned idx = 0; idx < _max_used && _ports[idx]; ++idx)
00167     {
00168         auto *target = _ports[idx];
00169         if (target != port && target->match_vlan(vlan))
00170             handle_request(target, port, request);
00171     }
00172
00173     // Send a copy to the monitor port
00174     if (_monitor && !filter_request(request))
00175         handle_request(_monitor, port, request);
00176 }
00177
00178 template<typename PORT>
00179 void
00180 Virtio_switch::handle_tx_requests(PORT *port, unsigned &num_reqs_handled)
00181 {
00182     while (auto req = port->get_tx_request())
00183     {
00184         req->dump_request(port);
00185         handle_tx_request(port, *req);
00186
00187         if (++num_reqs_handled >= Tx_burst)
00188             // Port has hit its TX burst limit.
00189             break;
00190     }
00191 }
00192
00193 bool
00194 Virtio_switch::handle_l4virtio_port_tx(L4virtio_port *port)
00195 {
00196     /* handle IRQ on one port for the time being */
00197     if (!port->tx_work_pending())
00198         Dbg(Dbg::Port, Dbg::Debug)
00199             .printf("%s: Irq without pending work\n", port->get_name());
00200
00201     unsigned num_reqs_handled = 0;
00202     do
00203     {
00204         port->tx_q()->disable_notify();
00205         port->rx_q()->disable_notify();
00206
00207         if (num_reqs_handled >= Tx_burst)
00208         {
00209             Dbg(Dbg::Port, Dbg::Debug)
00210                 .printf(
00211                     "%s: Tx burst limit hit, reschedule remaining Tx work.\n",
00212                     port->get_name());
00213
00214             // Port has hit its TX burst limit, so for fairness reasons, stop
00215             // processing TX work from this port, and instead reschedule the
00216             // pending work for later.
00217             port->reschedule_pending_tx();
00218             // NOTE: Notifications for this port remain disabled, until eventually
00219             // the reschedule handler calls `handle_l4virtio_port_tx` again.
00220             return false;
00221         }
00222
00223         // Within the loop, to trigger before enabling notifications again.
00224         all_rx_notify_disable_and_remember();
00225
00226         try
00227         {
00228             // throws Bad_descriptor exceptions raised on SRC port
00229             handle_tx_requests(port, num_reqs_handled);
00230         }
00231         catch (L4virtio::Svr::Bad_descriptor &e)
00232         {

```

```

00233         Dbg(Dbg::Port, Dbg::Warn, "REQ")
00234         .printf("%s: caught bad descriptor exception: %s - %i"
00235             " -- Signal device error on device %p.\n",
00236             __PRETTY_FUNCTION__, e.message(), e.error, port);
00237         port->device_error();
00238         all_rx_notify_emit_and_enable();
00239         return false;
00240     }
00241
00242     all_rx_notify_emit_and_enable();
00243
00244     port->tx_q()->enable_notify();
00245     port->rx_q()->enable_notify();
00246
00247     L4virtio::wmb();
00248     L4virtio::rmb();
00249 }
00250 while (port->tx_work_pending());
00251
00252 return true;
00253 }
00254
00255 #if CONFIG_VNS_IXL
00256 bool
00257 Virtio_switch::handle_ixl_port_tx(Ixl_port *port)
00258 {
00259     unsigned num_reqs_handled = 0;
00260
00261     all_rx_notify_disable_and_remember();
00262     handle_tx_requests(port, num_reqs_handled);
00263     all_rx_notify_emit_and_enable();
00264
00265     if (num_reqs_handled >= Tx_burst && port->tx_work_pending())
00266     {
00267         Dbg(Dbg::Port, Dbg::Info)
00268         .printf("%s: Tx burst limit hit, reschedule remaining Tx work.\n",
00269             port->get_name());
00270
00271         // Port has hit its TX burst limit, so for fairness reasons, stop
00272         // processing TX work from this port, and instead reschedule the
00273         // pending work for later.
00274         port->reschedule_pending_tx();
00275         return false;
00276     }
00277
00278     return true;
00279 }
00280 #endif
00281

```

## 17.64 switch.h

```

00001 /*
00002  * Copyright (C) 2016-2017, 2020, 2022-2024 Kernkonzept GmbH.
00003  * Author(s): Jean Wolter <jean.wolter@kernkonzept.com>
00004  *             Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "port.h"
00011 #include "port_l4virtio.h"
00012 #include "mac_table.h"
00013
00014 #if CONFIG_VNS_IXL
00015 #include "port_ixl.h"
00016 #endif
00017
00033 class Virtio_switch
00034 {
00035 private:
00036     Port_iface **_ports;
00037     Port_iface *_monitor = nullptr;
00038
00039     unsigned _max_ports;
00040     unsigned _max_used = 0;
00041     Mac_table<> _mac_table;
00042
00043     // Limits the number of consecutive TX requests a port can process before
00044     // being interrupted to ensure fairness to other ports.
00045     static constexpr unsigned Tx_burst = 128;
00046

```

```

00047 int lookup_free_slot();
00048
00058 template<typename REQ>
00059 void handle_tx_request(Port_iface *port, REQ const &request);
00060
00061 template<typename PORT>
00062 void handle_tx_requests(PORT *port, unsigned &num_reqs_handled);
00063
00064
00065 void all_rx_notify_emit_and_enable()
00066 {
00067     for (unsigned idx = 0; idx < _max_ports; ++idx)
00068         if (_ports[idx])
00069             _ports[idx]->rx_notify_emit_and_enable();
00070 }
00071
00072 void all_rx_notify_disable_and_remember()
00073 {
00074     for (unsigned idx = 0; idx < _max_ports; ++idx)
00075         if (_ports[idx])
00076             _ports[idx]->rx_notify_disable_and_remember();
00077 }
00078
00079 public:
00085     explicit Virtio_switch(unsigned max_ports);
00086
00095     bool add_port(Port_iface *port);
00096
00105     bool add_monitor_port(Port_iface *port);
00106
00114     void check_ports();
00115
00125     bool handle_l4virtio_port_tx(L4virtio_port *port);
00126
00127 #if CONFIG_VNS_IXL
00137     bool handle_ixl_port_tx(Ixl_port *port);
00138 #endif
00139
00148     int port_available(bool monitor)
00149     {
00150         if (monitor)
00151             return !_monitor ? 0 : -1;
00152
00153         return lookup_free_slot();
00154     }
00155 };

```

## 17.65 virtio\_net.h

```

00001 /*
00002  * Copyright (C) 2016-2017, 2019, 2022-2024 Kernkonzept GmbH.
00003  * Author(s): Jean Wolter <jean.wolter@kernkonzept.com>
00004  *             Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <l4/re/dataspace>
00011 #include <l4/re/util/unique_cap>
00012
00013 #include <l4/sys/cxx/ipc_epiface>
00014
00015 #include <l4/l4virtio/server/virtio>
00016 #include <l4/l4virtio/server/l4virtio>
00017 #include <l4/l4virtio/l4virtio>
00018
00019 #include "debug.h"
00024 class Virtqueue : public L4virtio::Svr::Virtqueue
00025 {
00026 public:
00027     bool kick_queue()
00028     {
00029         if (no_notify_guest())
00030             return false;
00031
00032         if (_do_kick)
00033             return true;
00034
00035         _kick_pending = true;
00036         return false;
00037     }
00038 }

```

```

00039 bool kick_enable_get_pending()
00040 {
00041     _do_kick = true;
00042     return _kick_pending;
00043 }
00044
00045 void kick_disable_and_remember()
00046 {
00047     _do_kick = false;
00048     _kick_pending = false;
00049 }
00050
00051 private:
00052     bool _do_kick = true;
00053     bool _kick_pending = false;
00054 };
00055
00071 class Virtio_net :
00072     public L4virtio::Svr::Device,
00073     public L4::Epiface_t<Virtio_net, L4virtio::Device>
00074 {
00075 public:
00076     struct Hdr_flags
00077     {
00078         l4_uint8_t raw;
00079         CXX_BITFIELD_MEMBER( 0, 0, need_csum, raw);
00080         CXX_BITFIELD_MEMBER( 1, 1, data_valid, raw);
00081     };
00082
00083     struct Hdr
00084     {
00085         Hdr_flags flags;
00086         l4_uint8_t gso_type;
00087         l4_uint16_t hdr_len;
00088         l4_uint16_t gso_size;
00089         l4_uint16_t csum_start;
00090         l4_uint16_t csum_offset;
00091         l4_uint16_t num_buffers;
00092     };
00093
00094     struct Features : L4virtio::Svr::Dev_config::Features
00095     {
00096         Features() = default;
00097         Features(l4_uint32_t raw) : L4virtio::Svr::Dev_config::Features(raw) {}
00098
00099         CXX_BITFIELD_MEMBER( 0, 0, csum, raw); // host handles partial csum
00100         CXX_BITFIELD_MEMBER( 1, 1, guest_csum, raw); // guest handles partial csum
00101         CXX_BITFIELD_MEMBER( 5, 5, mac, raw); // host has given mac
00102         CXX_BITFIELD_MEMBER( 6, 6, gso, raw); // host handles packets /w any GSO
00103         CXX_BITFIELD_MEMBER( 7, 7, guest_tso4, raw); // guest handles TSOv4 in
00104         CXX_BITFIELD_MEMBER( 8, 8, guest_tso6, raw); // guest handles TSOv6 in
00105         CXX_BITFIELD_MEMBER( 9, 9, guest_ecn, raw); // guest handles TSO[6] with ECN in
00106         CXX_BITFIELD_MEMBER(10, 10, guest_ufo, raw); // guest handles UFO in
00107         CXX_BITFIELD_MEMBER(11, 11, host_tso4, raw); // host handles TSOv4 in
00108         CXX_BITFIELD_MEMBER(12, 12, host_tso6, raw); // host handles TSOv6 in
00109         CXX_BITFIELD_MEMBER(13, 13, host_ecn, raw); // host handles TSO[6] with ECN in
00110         CXX_BITFIELD_MEMBER(14, 14, host_ufo, raw); // host handles UFO
00111         CXX_BITFIELD_MEMBER(15, 15, mrg_rxbuf, raw); // host can merge receive buffers
00112         CXX_BITFIELD_MEMBER(16, 16, status, raw); // virtio_net_config.status available
00113         CXX_BITFIELD_MEMBER(17, 17, ctrl_vq, raw); // Control channel available
00114         CXX_BITFIELD_MEMBER(18, 18, ctrl_rx, raw); // Control channel RX mode support
00115         CXX_BITFIELD_MEMBER(19, 19, ctrl_vlan, raw); // Control channel VLAN filtering
00116         CXX_BITFIELD_MEMBER(20, 20, ctrl_rx_extra, raw); // Extra RX mode control support
00117         CXX_BITFIELD_MEMBER(21, 21, guest_announce, raw); // Guest can announce device on the network
00118         CXX_BITFIELD_MEMBER(22, 22, mq, raw); // Device supports Receive Flow Steering
00119         CXX_BITFIELD_MEMBER(23, 23, ctrl_mac_addr, raw); // Set MAC address
00120     };
00121
00122     enum
00123     {
00124         Rx = 0,
00125         Tx = 1,
00126     };
00127
00128     struct Net_config_space
00129     {
00130         // The config defining mac address (if VIRTIO_NET_F_MAC aka Features::mac)
00131         l4_uint8_t mac[6];
00132         // currently not used ...
00133         l4_uint16_t status;
00134         l4_uint16_t max_virtqueue_pairs;
00135     };
00136
00137     L4virtio::Svr::Dev_config_t<Net_config_space> _dev_config;
00138
00139     explicit Virtio_net(unsigned vq_max)
00140     : L4virtio::Svr::Device(&_dev_config),

```

```

00141     _dev_config(L4VIRTIO_VENDOR_KK, L4VIRTIO_ID_NET, 2),
00142     _vq_max(vq_max)
00143 {
00144     Features hf(0);
00145     hf.ring_indirect_desc() = true;
00146     hf.mrg_rxbuf() = true;
00147 #if 0
00148     // disable currently unsupported options, but leave them in for
00149     // documentation purposes
00150     hf.csum() = true;
00151     hf.host_tso4() = true;
00152     hf.host_tso6() = true;
00153     hf.host_ufo() = true;
00154     hf.host_ecn() = true;
00155
00156     hf.guest_csum() = true;
00157     hf.guest_tso4() = true;
00158     hf.guest_tso6() = true;
00159     hf.guest_ufo() = true;
00160     hf.guest_ecn() = true;
00161 #endif
00162
00163     _dev_config.host_features(0) = hf.raw;
00164     _dev_config.set_host_feature(L4VIRTIO_FEATURE_VERSION_1);
00165     _dev_config.reset_hdr();
00166
00167     reset_queue_config(Rx, vq_max);
00168     reset_queue_config(Tx, vq_max);
00169 }
00170
00171 void reset() override
00172 {
00173     for (L4virtio::Svr::Virtqueue &q: _q)
00174         q.disable();
00175
00176     reset_queue_config(Rx, _vq_max);
00177     reset_queue_config(Tx, _vq_max);
00178     _dev_config.reset_hdr();
00179 }
00180
00181 template<typename T, unsigned N >
00182 static unsigned array_length(T (&)[N]) { return N; }
00183
00184 int reconfig_queue(unsigned index) override
00185 {
00186     Dbg(Dbg::Virtio, Dbg::Info, "Virtio")
00187         .printf("(%p): Reconfigure queue %d (%p): Status: %02x\n",
00188             this, index, _q + index, _dev_config.status().raw);
00189
00190     if (index >= array_length(_q))
00191         return -L4_ERANGE;
00192
00193     if (setup_queue(_q + index, index, _vq_max))
00194         return 0;
00195
00196     return -L4_EINVAL;
00197 }
00198
00199 void dump_features(Dbg const &dbg, const volatile l4_uint32_t *p)
00200 {
00201     dbg.cprintf("%08x:%08x:%08x:%08x:%08x:%08x:%08x:%08x\n",
00202         p[0], p[1], p[2], p[3], p[4], p[5], p[6], p[17]);
00203 }
00204
00205 void dump_features()
00206 {
00207     Dbg info(Dbg::Virtio, Dbg::Info, "Virtio");
00208     if (!info.is_active())
00209         return;
00210
00211     auto *hdr = _dev_config.hdr();
00212
00213     info.printf("Device %p running (%02x)\n\thost features: ",
00214         this, _dev_config.status().raw);
00215     dump_features(info, hdr->dev_features_map);
00216     info.printf("\tguest features: ");
00217     dump_features(info, hdr->driver_features_map);
00218 }
00219
00220 bool check_features() override
00221 {
00222     _negotiated_features = _dev_config.negotiated_features(0);
00223     return true;
00224 }
00225
00226 bool device_needs_reset() const
00227 { return _dev_config.status().device_needs_reset(); }

```

```

00228
00230 bool check_queues() override
00231 {
00232     for (L4virtio::Svr::Virtqueue &q: _q)
00233         if (!q.ready())
00234         {
00235             reset();
00236             Err().printf("failed to start queues\n");
00237             return false;
00238         }
00239     dump_features();
00240     return true;
00241 }
00242
00243 Server_iface *server_iface() const override
00244 { return L4::Epiface::server_iface(); }
00245
00250 void register_single_driver_irq() override
00251 {
00252     _kick_guest_irq = L4Re::Util::Unique_cap<L4::Irq>(
00253         L4Re::chkcap(server_iface()->template rcv_cap<L4::Irq>(0)));
00254     L4Re::chksys(server_iface()->realloc_rcv_cap(0));
00255 }
00256
00257 void trigger_driver_config_irq() override
00258 {
00259     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_CONFIG);
00260     _kick_guest_irq->trigger();
00261 }
00262
00269 void notify_queue(L4virtio::Svr::Virtqueue *queue)
00270 {
00271     // Downcast to Virtqueue to access kick_queue() - we know that our
00272     // queues have the type Virtqueue.
00273     Virtqueue *q = static_cast<Virtqueue*>(queue);
00274     if (q->kick_queue())
00275     {
00276         _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_VRING);
00277         _kick_guest_irq->trigger();
00278     }
00279 }
00280
00281 void kick_emit_and_enable()
00282 {
00283     bool kick_pending = false;
00284
00285     for (auto &q : _q)
00286         kick_pending |= q.kick_enable_get_pending();
00287
00288     if (kick_pending)
00289     {
00290         _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_VRING);
00291         _kick_guest_irq->trigger();
00292     }
00293 }
00294
00295 void kick_disable_and_remember()
00296 {
00297     for (auto &q : _q)
00298         q.kick_disable_and_remember();
00299 }
00300
00301 Features negotiated_features() const
00302 { return _negotiated_features; }
00303
00305 Virtqueue *tx_q() { return &_q[Tx]; }
00307 Virtqueue *rx_q() { return &_q[Rx]; }
00309 Virtqueue const *tx_q() const { return &_q[Tx]; }
00311 Virtqueue const *rx_q() const { return &_q[Rx]; }
00312
00313 private:
00314     Features _negotiated_features;
00316     unsigned _vq_max;
00318     Virtqueue _q[2];
00323     L4Re::Util::Unique_cap<L4::Irq> _kick_guest_irq;
00324 };

```

## 17.66 virtio\_net.h

```

00001 /* SPDX-License-Identifier: MIT */
00002 /*
00003  * Copyright (C) 2022, 2024 Kernkonzept GmbH.
00004  * Author(s): Stephan Gerhold <stephan.gerhold@kernkonzept.com>

```

```

00005  */
00006
00007 #pragma once
00008
00014
00015 #include <l4/sys/types.h>
00016
00020 typedef struct l4virtio_net_header_t
00021 {
00022     l4_uint8_t flags;
00023     l4_uint8_t gso_type;
00024     l4_uint16_t hdr_len;
00025     l4_uint16_t gso_size;
00026     l4_uint16_t csum_start;
00027     l4_uint16_t csum_offset;
00028     l4_uint16_t num_buffers;
00029 } l4virtio_net_header_t;
00030
00034 typedef struct l4virtio_net_config_t
00035 {
00036     l4_uint8_t mac[6];
00037     l4_uint16_t status;
00038     l4_uint16_t max_virtqueue_pairs;
00039     l4_uint16_t mtu;
00040     l4_uint32_t speed;
00041     l4_uint8_t duplex;
00042 } l4virtio_net_config_t;
00043
00045 enum L4virtio_net_feature_bits
00046 {
00047     L4VIRTIO_NET_F_CSUM = 0,
00048     L4VIRTIO_NET_F_GUEST_CSUM = 1,
00049     L4VIRTIO_NET_F_MTU = 3,
00050     L4VIRTIO_NET_F_MAC = 5,
00051     L4VIRTIO_NET_F_GUEST_TSO4 = 7,
00052     L4VIRTIO_NET_F_GUEST_TSO6 = 8,
00053     L4VIRTIO_NET_F_GUEST_ECN = 9,
00054     L4VIRTIO_NET_F_GUEST_UFO = 10,
00055     L4VIRTIO_NET_F_HOST_TSO4 = 11,
00056     L4VIRTIO_NET_F_HOST_TSO6 = 12,
00057     L4VIRTIO_NET_F_HOST_ECN = 13,
00058     L4VIRTIO_NET_F_HOST_UFO = 14,
00059     L4VIRTIO_NET_F_MRG_RXBUF = 15,
00060     L4VIRTIO_NET_F_STATUS = 16,
00061     L4VIRTIO_NET_F_CTRL_VQ = 17,
00062     L4VIRTIO_NET_F_CTRL_RX = 18,
00063     L4VIRTIO_NET_F_CTRL_VLAN = 19,
00064     L4VIRTIO_NET_F_GUEST_ANNOUNCE = 21,
00065     L4VIRTIO_NET_F_MQ = 22,
00066     L4VIRTIO_NET_F_CTRL_MAC_ADDR = 23,
00067 };
00068

```

## 17.67 virtio\_net\_buffer.h

```

00001 /*
00002  * Copyright (C) 2016-2017, 2022, 2024 Kernkonzept GmbH.
00003  * Author(s): Jean Wolter <jean.wolter@kernkonzept.com>
00004  *           Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <l4/l4virtio/server/l4virtio>
00015
00019 struct Buffer : L4virtio::Svr::Data_buffer
00020 {
00021     Buffer() = default;
00022     Buffer(L4virtio::Svr::Driver_mem_region const *r,
00023           L4virtio::Svr::Virtqueue::Desc const &d,
00024           L4virtio::Svr::Request_processor const *)
00025     {
00026         pos = static_cast<char *>(r->local(d.addr));
00027         left = d.len;
00028     }
00029
00030     Buffer(char *data, l4_uint32_t size)
00031     {
00032         pos = data;
00033         left = size;
00034     }
00035

```

```

00036     template<typename T>
00037     explicit Buffer(T *p) : Data_buffer(p) {};
00038 };
00039

```

## 17.68 vlan.h

```

00001 /*
00002  * Copyright (C) 2020, 2022-2024 Kernkonzept GmbH.
00003  * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/cxx/minmax>
00010 #include <l4/l4virtio/server/virtio>
00011 #include <l4/sys/types.h>
00012 #include <string.h>
00013
00014 #include "virtio_net.h"
00015 #include "virtio_net_buffer.h"
00016
00017 namespace {
00018
00019     const l4_uint16_t VLAN_ID_NATIVE = 0xffffU;
00020     const l4_uint16_t VLAN_ID_TRUNK = 0xfffeU;
00021
00022     inline bool vlan_valid_id(l4_uint16_t id)
00023     {
00024         return id > 0U && id < 0xffffU;
00025     }
00026
00027 }
00028
00029
00030
00031
00032
00033
00034
00035
00036 class Virtio_vlan_mangle
00037 {
00038     l4_uint16_t _tci;
00039     l4_uint8_t _mac_remaining;
00040     l4_int8_t _tag_remaining;
00041
00042     constexpr Virtio_vlan_mangle(l4_uint16_t tci, l4_int8_t tag_remaining)
00043     : _tci{tci}, _mac_remaining{12}, _tag_remaining{tag_remaining}
00044     {}
00045
00046 public:
00047     Virtio_vlan_mangle()
00048     : _tci{0}, _mac_remaining{0}, _tag_remaining{0}
00049     {}
00050
00051
00052     static constexpr Virtio_vlan_mangle add(l4_uint16_t tci)
00053     {
00054         return Virtio_vlan_mangle(tci, 4);
00055     }
00056
00057     static constexpr Virtio_vlan_mangle remove()
00058     {
00059         return Virtio_vlan_mangle(0xffffU, -4);
00060     }
00061
00062
00063     l4_uint32_t copy_pkt(Buffer &dst, Buffer &src)
00064     {
00065         l4_uint32_t ret;
00066
00067         if (L4_LIKELY(_tci == 0))
00068         {
00069             // pass through (no tag or keep tag)
00070             ret = src.copy_to(&dst);
00071         }
00072         else if (_mac_remaining)
00073         {
00074             // copy initial MAC addresses
00075             ret = src.copy_to(&dst, _mac_remaining);
00076             _mac_remaining -= ret;
00077         }
00078         else if (_tag_remaining > 0)
00079         {
00080             // add VLAN tag
00081             l4_uint8_t tag[4] = {
00082                 0x81, 0x00,
00083                 static_cast<l4_uint8_t>(_tci >> 8),
00084                 static_cast<l4_uint8_t>(_tci & 0xffU)
00085             };
00086
00087             ret = src.copy_to(&dst, tag, 4);
00088             _tag_remaining -= 4;
00089         }
00090
00091         return ret;
00092     }
00093
00094
00095
00096
00097
00098
00099
00100
00101
00102
00103
00104
00105
00106
00107
00108
00109
00110
00111
00112
00113
00114
00115

```



```

00116
00117     ret = cxx::min(static_cast<l4_uint32_t>(_tag_remaining), dst.left);
00118     memcpy(dst.pos, &tag[4 - _tag_remaining], ret);
00119     dst.skip(ret);
00120     _tag_remaining -= (int)ret;
00121 }
00122 else if (_tag_remaining < 0)
00123 {
00124     // remove VLAN tag
00125     _tag_remaining += static_cast<int>(src.skip(-_tag_remaining));
00126     ret = 0;
00127 }
00128 else
00129     ret = src.copy_to(&dst);
00130
00131 return ret;
00132 }
00133
00142 void rewrite_hdr(Virtio_net::Hdr *hdr)
00143 {
00144     if (L4_UNLIKELY(!_tci != 0 && hdr->flags.need_csum()))
00145     {
00146         if (_tci == 0xffffU)
00147             hdr->csum_start -= 4U;
00148         else
00149             hdr->csum_start += 4U;
00150     }
00151 }
00152 };
00153

```

## 17.69 amd64/l4/util/perform.h File Reference

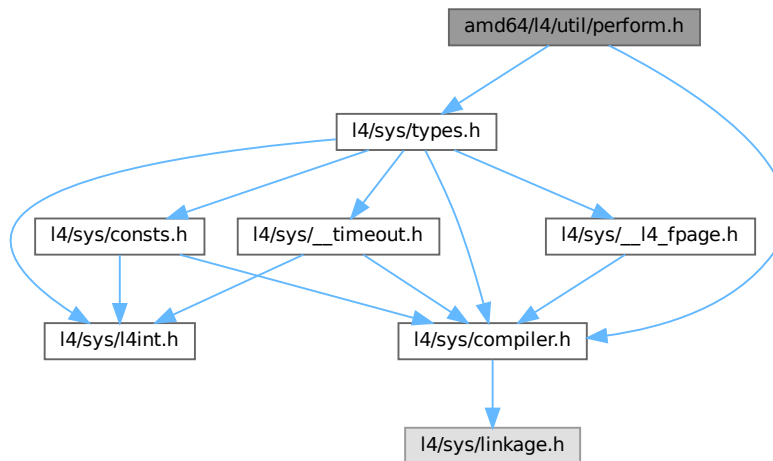
Performance Monitoring using P5/P6 Measurement Counters.

```

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for perform.h:



### 17.69.1 Detailed Description

Performance Monitoring using P5/P6 Measurement Counters.

Define either `CPU_PENTIUM` or `CPU_P6`

Definition in file [perform.h](#).

## 17.70 perform.h

[Go to the documentation of this file.](#)

```

00001
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *          Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *          economic rights: Technische Universität Dresden (Germany)
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #ifndef __L4UTIL_PERFORM_H
00014 #define __L4UTIL_PERFORM_H
00015
00016 #include <l4/sys/types.h>
00017 #include <l4/sys/compiler.h>
00018
00019 L4_BEGIN_DECLS
00020
00021 extern const char*strp6pmc_event(l4_uint32_t event);
00022
00023 #ifndef CONFIG_PERFORM_ONLY_PROTOTYPES
00024
00025 #if ! (defined CPU_PENTIUM ^ defined CPU_P6 ^ defined CPU_K7)
00026
00027 #error You must define your target architecture.
00028 #error Define EITHER CPU_PENTIUM for Intel Pentium or CPU_P6 for Intel PPro/PII/PIII.
00029
00030 #else
00031
00032 /* P5/P6/K7 section */
00033
00034 /* Makros for access to model specific registers (MSR) */
00035
00036 /* Write the 64-Bit Model Specific Register. First argument is the register,
00037  second the 64-Bit value. This can only be called at privilege level 0.
00038  With L4, the kernel emulates the WRMSR when calling in PL 3.
00039  */
00040 static inline void l4_i586_wrmsr(unsigned reg,unsigned long long*val){
00041     unsigned long dummyeax, dummyecx, dummyedx;
00042
00043     asm volatile(
00044         ".byte 0xf; .byte 0x30\n" /* wrmsr */
00045         : "=a" (dummyeax), "=d" (dummyedx), "=c" (dummyecx)
00046         : "2" (reg), "0" (*(unsigned *)val), "1" (*(unsigned *)val+1))
00047     );
00048 }
00049
00050 /* Read the 64-Bit Model Specific Register. First argument is the register,
00051  second the address to a 64-Bit value. This can only be called at
00052  privilege level 0. With L4, the kernel emulates the RDMSR when calling
00053  in PL 3.
00054  */
00055 static inline void l4_i586_rdmsr(unsigned reg,unsigned long long*val){
00056     unsigned dummy;
00057
00058     asm volatile(
00059         ".byte 0xf; .byte 0x32\n" /* rdmsr */
00060         : "=a" (*(unsigned *)val), "=d" (*(unsigned *)val+1), "=c" (dummy)
00061         : "2" (reg)
00062         );
00063 }
00064
00065
00066 #ifndef CPU_PENTIUM
00067 /* Pentium section */
00068
00069 /* functions and events defined here are only usable at Pentium
00070 Processors. P6 architecture does NOT support this kind of measuring and
00071 these events. P6 architecture has its own counters and its own events.
00072 See P6-section for details. */
00073
00074 /* from l4linux/arch/l4-i386/include/perform.h */
00075
00076 static inline void
00077 l4_i586_reset_event_counter(void){
00078     asm volatile("xor %%rax, %%rax\n"
00079         "xor %%rdx, %%rdx\n"
00080         "mov $0x12, %%rcx\n"
00081         ".byte 0x0f, 0x30\n"
00082         "movl $0x13, %%rcx\n"
00083         ".byte 0x0f, 0x30\n"
00084         : : : "cx", "ax", "dx"
00085     );
00086 };
00087

```

```

00088 static inline void
00089 l4_i586_read_event_counter_long(long long *counter0, long long *counter1)
00090 {
00091     asm volatile(
00092         /*      "movl $0, %%eax\n"
00093         "movl $0x11, %%ecx\n"
00094         ".byte 0x0f, 0x30\n" */ /* stop event counting */
00095         "mov $0x12, %%rcx\n"
00096         ".byte 0x0f, 0x32\n"
00097         "mov %%rax, (%%rbx)\n"
00098         "mov %%rdx, 4(%%rbx)\n"
00099         "mov $0x13, %%ecx\n"
00100         ".byte 0x0f, 0x32\n"
00101         "mov %%rax, (%%rsi)\n"
00102         "mov %%rdx, 4(%%rsi)\n"
00103         : /* no output */
00104         : "b" (counter0), "S" (counter1)
00105         : "ax", "cx", "dx"
00106     );
00107 }
00108
00109 static inline void
00110 l4_i586_read_event_counter(int *counter0, int *counter1)
00111 {
00112     asm volatile("push %%rdx\n"
00113         ".byte 0x0f, 0x30\n"
00114         "mov $0x12, %%rcx\n"
00115         ".byte 0x0f, 0x32\n"
00116         "mov %%rax, %%rbx\n"
00117         "movl $0x13, %%rcx\n"
00118         ".byte 0x0f, 0x32\n"
00119         "popl %%edx\n"
00120         : "b" (*counter0), "=a" (*counter1)
00121         : "1" (0), "c" (0x11)
00122         );
00123 }
00124
00125 static inline void
00126 l4_i586_select_event(int event0, int event1)
00127 {
00128     asm volatile(".byte 0x0f, 0x30\n"
00129         :
00130         :
00131         "a" (event0 + (event1 << 16)),
00132         "d" (0),
00133         "c" (0x11)
00134         );
00135 };
00136
00137 #define P5_RD_MISS          0x003 /* 000011B */
00138 #define P5_WR_MISS          0x008 /* 000100B */
00139 #define P5_RW_MISS          0x029 /* 101001B */
00140 #define P5_EX_MISS          0x00e /* 001110B */
00141
00142 #define P5_D_WBACK          0x006 /* 000110B */
00143
00144 #define P5_RW_TLB           0x002 /* 00010B */
00145 #define P5_EX_TLB           0x00d /* 01101B */
00146
00147 #define P5_A_STALL           0x01f /* 11111B */
00148 #define P5_W_STALL           0x019 /* 11001B */
00149 #define P5_R_STALL           0x01a /* 11010B */
00150 #define P5_X_STALL           0x01b /* 11011B */
00151
00152 #define P5_AGI_STALL         0x01f /* 11111B */
00153
00154 #define P5_PIPELINE_FLUSH    0x015 /* 10101B */
00155
00156 #define P5_NON_CACHE_RD      0x01e /* 11110B */
00157 #define P5_NCACHE_REFS       0x01e /* 11110B */
00158 #define P5_LOCKED_BUS        0x01c /* 11100B */
00159
00160 #define P5_MEM2PIPE          0x009 /* 01001B */
00161 #define P5_BANK_CONF         0x00a /* 01010B */
00162
00163
00164 #define P5_INSTRS_EX          0x016 /* 10110B */
00165 #define P5_INSTRS_EX_V       0x017 /* 10111B */
00166
00167
00168 #define P5_CNT_NOTHING        (0x00 << 6) /* 00B << 6 */
00169 #define P5_CNT_EVENT_PL0      (0x01 << 6) /* 01B << 6 */
00170 #define P5_CNT_EVENT_PL3      (0x02 << 6) /* 10B << 6 */
00171 #define P5_CNT_EVENT          (0x03 << 6) /* 11B << 6 */
00172 #define P5_CNT_CLOCKS_PL0     (0x05 << 6) /* 101B << 6 */
00173 #define P5_CNT_CLOCKS_PL3     (0x06 << 6) /* 110B << 6 */
00174 #define P5_CNT_CLOCKS         (0x07 << 6) /* 111B << 6 */

```

```

00175
00176
00177 #else
00178 #if defined CPU_P6
00179 /* PPro/PII/PIII section */
00180
00181 /*-
00182  * Copyright (c) 1997 The President and Fellows of Harvard College.
00183  * All rights reserved.
00184  * Copyright (c) 1997 Aaron B. Brown.
00185  *
00186  * Redistribution and use in source and binary forms, with or without
00187  * modification, are permitted provided that the following conditions
00188  * are met:
00189  * 1. Redistributions of source code must retain the above copyright
00190  *    notice, this list of conditions and the following disclaimer.
00191  * 2. Redistributions in binary form must reproduce the above copyright
00192  *    notice, this list of conditions and the following disclaimer in the
00193  *    documentation and/or other materials provided with the distribution.
00194  * 3. All advertising materials mentioning features or use of this software
00195  *    must display the following acknowledgement:
00196  *       This product includes software developed by Harvard University
00197  *       and its contributors.
00198  * 4. Neither the name of the University nor the names of its contributors
00199  *    may be used to endorse or promote products derived from this software
00200  *    without specific prior written permission.
00201  *
00202  * THIS SOFTWARE IS PROVIDED BY HARVARD AND CONTRIBUTORS ``AS IS'' AND
00203  * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
00204  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
00205  * ARE DISCLAIMED. IN NO EVENT SHALL HARVARD UNIVERSITY OR CONTRIBUTORS BE
00206  * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
00207  * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
00208  * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
00209  * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
00210  * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
00211  * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
00212  * POSSIBILITY OF SUCH DAMAGE.
00213  */
00214
00215 /*****
00216  ** Symbolic names for counter numbers (used in select_p6counter()) **
00217  *****/
00218 *
00219 * These correspond in order to the Pentium Pro counters. Add new counters at
00220 * the end. These agree with the mnemonics in the Pentium Pro Family
00221 * Developer's Manual, vol 3.
00222 *
00223 * Those events marked with a $ require a MESI unit field; those marked with
00224 * a @ require a self/any unit field. Those marked with a 0 are only supported
00225 * in counter 0; those marked with 1 are only supported in counter 1.
00226 */
00227
00228 /* Data cache unit */
00229 #define P6_DATA_MEM_REFS 0x43 /* total memory refs */
00230 #define P6_DCU_LINES_IN 0x45 /* all lines allocated in cache unit */
00231 #define P6_DCU_M_LINES_IN 0x46 /* M lines allocated in cache unit */
00232 #define P6_DCU_M_LINES_OUT 0x47 /* M lines evicted from cache */
00233 #define P6_DCU_MISS_OUTSTANDING 0x48 /* #cycles a miss is outstanding */
00234
00235 /* Instruction fetch unit */
00236 #define P6_IFU_IFETCH 0x80 /* instruction fetches */
00237 #define P6_IFU_IFETCH_MISS 0x81 /* instruction fetch misses */
00238 #define P6_ITLB_MISS 0x85 /* ITLB misses */
00239 #define P6_IFU_MEM_STALL 0x86 /* number of cycles IFU is stalled */
00240 #define P6_ILD_STALL 0x87 /* #stalls in instr length decode */
00241
00242 /* L2 Cache */
00243 #define P6_L2_IFETCH 0x28 /* ($) l2 ifetches */
00244 #define P6_L2_LD 0x29 /* ($) l2 data loads */
00245 #define P6_L2_ST 0x2a /* ($) l2 data stores */
00246 #define P6_L2_LINES_IN 0x24 /* lines allocated in l2 */
00247 #define P6_L2_LINES_OUT 0x26 /* lines removed from l2 */
00248 #define P6_L2_M_LINES_INM 0x25 /* modified lines allocated in l2 */
00249 #define P6_L2_M_LINES_OUTM 0x27 /* modified lines removed from l2 */
00250 #define P6_L2_RQSTS 0x2e /* ($) number of l2 requests */
00251 #define P6_L2_ADS 0x21 /* number of l2 addr strobes */
00252 #define P6_L2_DBUS_BUSY 0x22 /* number of data bus busy cycles */
00253 #define P6_L2_DBUS_BUSY_RD 0x23 /* #bus cycles xferring l2->cpu */
00254
00255 /* External bus logic */
00256 #define P6_BUS_DRDY_CLOCKS 0x62 /* (@) #clocks DRDY is asserted */
00257 #define P6_BUS_LOCK_CLOCKS 0x63 /* (@) #clocks LOCK is asserted */
00258 #define P6_BUS_REQ_OUTSTANDING 0x60 /* #bus requests outstanding */
00259 #define P6_BUS_TRAN_BRD 0x65 /* (@) bus burst read txns */
00260 #define P6_BUS_TRAN_RFO 0x66 /* (@) bus read for ownership txns */
00261 #define P6_BUS_TRAN_WB 0x67 /* (@) bus writeback txns */

```

```

00262 #define P6_BUS_TRAN_IFETCH 0x68 /* (@) bus instr fetch txns */
00263 #define P6_BUS_TRAN_INVALID 0x69 /* (@) bus invalidate txns */
00264 #define P6_BUS_TRAN_PWR 0x6a /* (@) bus partial write txns */
00265 #define P6_BUS_TRANS_P 0x6b /* (@) bus partial txns */
00266 #define P6_BUS_TRANS_IO 0x6c /* (@) bus I/O txns */
00267 #define P6_BUS_TRAN_DEF 0x6d /* (@) bus deferred txns */
00268 #define P6_BUS_TRAN_BURST 0x6e /* (@) bus burst txns */
00269 #define P6_BUS_TRAN_ANY 0x70 /* (@) total bus txns */
00270 #define P6_BUS_TRAN_MEM 0x6f /* (@) total memory txns */
00271 #define P6_BUS_DATA_RCV 0x64 /* #busclocks CPU is receiving data */
00272 #define P6_BUS_BNR_DRV 0x61 /* #busclocks CPU is driving BNR pin */
00273 #define P6_BUS_HIT_DRV 0x7a /* #busclocks CPU is driving HIT pin */
00274 #define P6_BUS_HITM_DRV 0x7b /* #busclocks CPU is driving HITM pin */
00275 #define P6_BUS_SNOOP_STALL 0x7e /* #clkcycles bus is snoop-stalled */
00276
00277 /* FPU */
00278 #define P6_FLOPS 0xc1 /* (0) number of FP ops retired */
00279 #define P6_FP_COMP_OPS 0x10 /* (0) computational FPOPS exec'd */
00280 #define P6_FP_ASSIST 0x11 /* (1) FP excep's handled in ucode */
00281 #define P6_MUL 0x12 /* (1) number of FP multiplies */
00282 #define P6_DIV 0x13 /* (1) number of FP divides */
00283 #define P6_CYCLES_DIV_BUSY 0x14 /* (0) number of cycles divider busy */
00284
00285 /* Memory ordering */
00286 #define P6_LD_BLOCKS 0x03 /* number of store buffer blocks */
00287 #define P6_SB_DRAINS 0x04 /* # of store buffer drain cycles */
00288 #define P6_MISALING_MEM_REF 0x05 /* # misaligned data memory refs */
00289
00290 /* Instruction decoding and retirement */
00291 #define P6_INST_RETIRED 0xc0 /* number of instrs retired */
00292 #define P6_UOPS_RETIRED 0xc2 /* number of micro-ops retired */
00293 #define P6_INST_DECODER 0xd0 /* number of instructions decoded */
00294
00295 /* Interrupts */
00296 #define P6_HW_INT_RX 0xc8 /* number of hardware interrupts */
00297 #define P6_CYCLES_INT_MASKED 0xc6 /* number of cycles hardints masked */
00298 #define P6_CYCLES_INT_PENDING_AND_MASKED 0xc7 /* #cycles masked but pending */
00299
00300 /* Branches */
00301 #define P6_BR_INST_RETIRED 0xc4 /* number of branch instrs retired */
00302 #define P6_BR_MISS_PRED_RETIRED 0xc5 /* number of mispred'd brs retired */
00303 #define P6_BR_TAKEN_RETIRED 0xc9 /* number of taken branches retired */
00304 #define P6_BR_MISS_PRED_TAKEN_RET 0xca /* #taken mispredictions br's retired */
00305 #define P6_BR_INST_DECODED 0xe0 /* number of branch instrs decoded */
00306 #define P6_BTБ_MISSES 0xe2 /* # of branches that missed in BTB */
00307 #define P6_BR_BOGUS 0xe4 /* number of bogus branches */
00308 #define P6_BACLEAR 0xe6 /* # times BACLEAR is asserted */
00309
00310 /* Stalls */
00311 #define P6_RESOURCE_STALLS 0xa2 /* # resource-related stall cycles */
00312 #define P6_PARTIAL_RAT_STALLS 0xd2 /* # cycles/events for partial stalls */
00313
00314 /* Segment register loads */
00315 #define P6_SEGMENT_REG_LOADS 0x06 /* number of segment register loads */
00316
00317 /* Clocks */
00318 #define P6_CPU_CLK_UNHALTED 0x79 /* #clocks CPU is not halted */
00319
00320 /* Unit field tags */
00321 #define P6_UNIT_M 0x0800
00322 #define P6_UNIT_E 0x0400
00323 #define P6_UNIT_S 0x0200
00324 #define P6_UNIT_I 0x0100
00325 #define P6_UNIT_MESI 0x0f00
00326
00327 #define P6_UNIT_SELF 0x0000
00328 #define P6_UNIT_ANY 0x2000
00329
00330 /*****
00331 ** Flag bit definitions (used for the 'flag' field in select_p6counter()) **
00332 *****/
00333 *
00334 * The driver accepts fully-formed counter specifications from user-level.
00335 * The following flags are mnemonics for the bits that get set in the
00336 * PerfEvtSel0 and PerfEvtSel1 MSR's
00337 *
00338 */
00339 #define P6CNT_U 0x010000 /* Monitor user-level events */
00340 #define P6CNT_K 0x020000 /* Monitor kernel-level events */
00341 #define P6CNT_E 0x040000 /* Edge detect: count state transitions */
00342 #define P6CNT_PC 0x080000 /* Pin control: ?? */
00343 #define P6CNT_IE 0x100000 /* Int enable: enable interrupt on overflow */
00344 #define P6CNT_F 0x200000 /* Freeze counter (handled in software) */
00345 #define P6CNT_EN 0x400000 /* enable counters (in PerfEvtSel0) */
00346 #define P6CNT_IV 0x800000 /* Invert counter mask comparison result */
00347
00348 /*****

```



```

00436 #endif
00437
00438 #endif
00439
00440 /* end of P5/P6/K7 section*/
00441 #endif
00442
00443 /* end of not only lib-prototypes section */
00444 #endif
00445
00446 L4_END_DECLS
00447
00448 #endif

```

## 17.71 x86/l4/util/perform.h File Reference

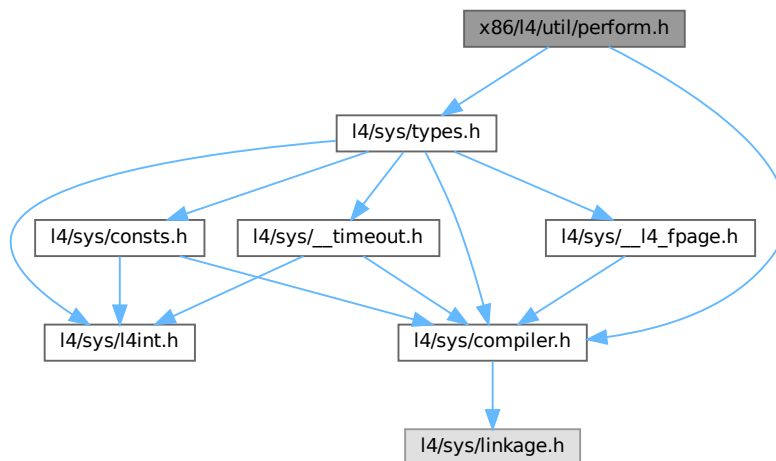
Performance Monitoring using P5/P6 Measurement Counters.

```

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for perform.h:



### 17.71.1 Detailed Description

Performance Monitoring using P5/P6 Measurement Counters.

Define either `CPU_PENTIUM` or `CPU_P6`

Definition in file [perform.h](#).

## 17.72 perform.h

[Go to the documentation of this file.](#)

```

00001
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *      Frank Mehnert <fm3@os.inf.tu-dresden.de>,
00010  *      Lars Reuther <reuther@os.inf.tu-dresden.de>
00011  *      economic rights: Technische Universität Dresden (Germany)
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014
00015 #ifndef __L4UTIL_PERFORM_H
00016 #define __L4UTIL_PERFORM_H
00017
00018 #include <l4/sys/types.h>
00019 #include <l4/sys/compiler.h>
00020
00021 L4_BEGIN_DECLS
00022
00023 extern const char*strp6pmc_event(l4_uint32_t event);
00024
00025 #ifndef CONFIG_PERFORM_ONLY_PROTOTYPES
00026
00027 #if ! (defined CPU_PENTIUM ^ defined CPU_P6 ^ defined CPU_K7)
00028
00029 #error You must define your target architecture.
00030 #error Define EITHER CPU_PENTIUM for Intel Pentium or CPU_P6 for Intel PPro/PII/PIII.
00031
00032 #else
00033
00034 /* P5/P6/K7 section */
00035
00036 /* Makros for access to model specific registers (MSR) */
00037
00038 /* Write the 64-Bit Model Specific Register. First argument is the register,
00039  second the 64-Bit value. This can only be called at privilege level 0.
00040  With L4, the kernel emulates the WRMSR when calling in PL 3.
00041  */
00042 static inline void l4_i586_wrmsr(unsigned reg,unsigned long long*val){
00043     unsigned long dummyeax, dummyecx, dummyedx;
00044
00045     asm volatile(
00046         ".byte 0xf; .byte 0x30\n" /* wrmsr */
00047         : "a" (dummyeax), "d" (dummyedx), "c" (dummyecx)
00048         : "2" (reg), "0" (*(unsigned *)val), "1" (*(unsigned *)val+1))
00049     );
00050 }
00051
00052 /* Read the 64-Bit Model Specific Register. First argument is the register,
00053  second the address to a 64-Bit value. This can only be called at
00054  privilege level 0. With L4, the kernel emulates the RDMSR when calling
00055  in PL 3.
00056  */
00057 static inline void l4_i586_rdmsr(unsigned reg,unsigned long long*val){
00058     unsigned dummy;
00059
00060     asm volatile(
00061         ".byte 0xf; .byte 0x32\n" /* rdmsr */
00062         : "a" (*(unsigned *)val), "d" (*(unsigned *)val+1), "c" (dummy)
00063         : "2" (reg)
00064         );
00065 }
00066
00067
00068 #ifdef CPU_PENTIUM
00069 /* Pentium section */
00070
00071 /* functions and events defined here are only usable at Pentium
00072  Processors. P6 architecture does NOT support this kind of measuring and
00073  these events. P6 architecture has its own counters and its own events.
00074  See P6-section for details. */
00075
00076 /* from l4linux/arch/l4-i386/include/perform.h */
00077
00078 static inline void
00079 l4_i586_reset_event_counter(void){
00080     asm volatile("xor %%eax, %%eax\n"
00081         "xor %%edx, %%edx\n"
00082         "movl $0x12, %%ecx\n"
00083         ".byte 0x0f, 0x30\n"
00084         "movl $0x13, %%ecx\n"
00085         ".byte 0x0f, 0x30\n"
00086         : : "cx", "ax", "dx"
00087     );

```



```

00088 };
00089
00090 static inline void
00091 l4_i586_read_event_counter_long(long long *counter0, long long *counter1)
00092 {
00093     asm volatile(
00094         /*      "movl $0, %%eax\n"
00095         "movl $0x11, %%ecx\n"
00096         ".byte 0x0f, 0x30\n" */ /* stop event counting */
00097         "movl $0x12, %%ecx\n"
00098         ".byte 0x0f, 0x32\n"
00099         "movl %%eax, (%%ebx)\n"
00100         "movl %%edx, 4(%%ebx)\n"
00101         "movl $0x13, %%ecx\n"
00102         ".byte 0x0f, 0x32\n"
00103         "movl %%eax, (%%esi)\n"
00104         "movl %%edx, 4(%%esi)\n"
00105         : /* no output */
00106         : "b" (counter0), "S" (counter1)
00107         : "ax", "cx", "dx"
00108     );
00109 }
00110
00111 static inline void
00112 l4_i586_read_event_counter(int *counter0, int *counter1)
00113 {
00114     asm volatile("pushl %%edx\n"
00115         ".byte 0x0f, 0x30\n"
00116         "movl $0x12, %%ecx\n"
00117         ".byte 0x0f, 0x32\n"
00118         "movl %%eax, %%ebx\n"
00119         "movl $0x13, %%ecx\n"
00120         ".byte 0x0f, 0x32\n"
00121         "popl %%edx\n"
00122         : "=b" (*counter0), "=a" (*counter1)
00123         : "1" (0), "c" (0x11)
00124         );
00125 }
00126
00127 static inline void
00128 l4_i586_select_event(int event0, int event1)
00129 {
00130     asm volatile(".byte 0x0f, 0x30\n"
00131         :
00132         :
00133         "a" (event0 + (event1 << 16)),
00134         "d" (0),
00135         "c" (0x11)
00136     );
00137 };
00138
00139 #define P5_RD_MISS          0x003 /* 000011B */
00140 #define P5_WR_MISS          0x008 /* 000100B */
00141 #define P5_RW_MISS          0x029 /* 101001B */
00142 #define P5_EX_MISS          0x00e /* 001110B */
00143
00144 #define P5_D_WBACK          0x006 /* 000110B */
00145
00146 #define P5_RW_TLB           0x002 /* 00010B */
00147 #define P5_EX_TLB           0x00d /* 01101B */
00148
00149 #define P5_A_STALL           0x01f /* 11111B */
00150 #define P5_W_STALL           0x019 /* 11001B */
00151 #define P5_R_STALL           0x01a /* 11010B */
00152 #define P5_X_STALL           0x01b /* 11011B */
00153
00154 #define P5_AGI_STALL         0x01f /* 11111B */
00155
00156 #define P5_PIPELINE_FLUSH    0x015 /* 10101B */
00157
00158 #define P5_NON_CACHE_RD      0x01e /* 11110B */
00159 #define P5_NCACHE_REFS       0x01e /* 11110B */
00160 #define P5_LOCKED_BUS        0x01c /* 11100B */
00161
00162 #define P5_MEM2PIPE          0x009 /* 01001B */
00163 #define P5_BANK_CONF         0x00a /* 01010B */
00164
00165
00166 #define P5_INSTRS_EX          0x016 /* 10110B */
00167 #define P5_INSTRS_EX_V       0x017 /* 10111B */
00168
00169
00170 #define P5_CNT_NOTHING        (0x00 << 6) /* 00B << 6 */
00171 #define P5_CNT_EVENT_PL0      (0x01 << 6) /* 01B << 6 */
00172 #define P5_CNT_EVENT_PL3      (0x02 << 6) /* 10B << 6 */
00173 #define P5_CNT_EVENT          (0x03 << 6) /* 11B << 6 */
00174 #define P5_CNT_CLOCKS_PL0     (0x05 << 6) /* 101B << 6 */

```

```

00175 #define P5_CNT_CLOCKS_PL3    (0x06 << 6) /* 110B << 6 */
00176 #define P5_CNT_CLOCKS        (0x07 << 6) /* 111B << 6 */
00177
00178
00179 #else
00180 #if defined CPU_P6
00181 /* PPro/PII/PIII section */
00182
00183 /*-
00184  * Copyright (c) 1997 The President and Fellows of Harvard College.
00185  * All rights reserved.
00186  * Copyright (c) 1997 Aaron B. Brown.
00187  *
00188  * Redistribution and use in source and binary forms, with or without
00189  * modification, are permitted provided that the following conditions
00190  * are met:
00191  * 1. Redistributions of source code must retain the above copyright
00192  *    notice, this list of conditions and the following disclaimer.
00193  * 2. Redistributions in binary form must reproduce the above copyright
00194  *    notice, this list of conditions and the following disclaimer in the
00195  *    documentation and/or other materials provided with the distribution.
00196  * 3. All advertising materials mentioning features or use of this software
00197  *    must display the following acknowledgement:
00198  *        This product includes software developed by Harvard University
00199  *        and its contributors.
00200  * 4. Neither the name of the University nor the names of its contributors
00201  *    may be used to endorse or promote products derived from this software
00202  *    without specific prior written permission.
00203  *
00204  * THIS SOFTWARE IS PROVIDED BY HARVARD AND CONTRIBUTORS ``AS IS" AND
00205  * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
00206  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
00207  * ARE DISCLAIMED.  IN NO EVENT SHALL HARVARD UNIVERSITY OR CONTRIBUTORS BE
00208  * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
00209  * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
00210  * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
00211  * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
00212  * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
00213  * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
00214  * POSSIBILITY OF SUCH DAMAGE.
00215  */
00216
00217 /*****
00218  ** Symbolic names for counter numbers (used in select_p6counter()) **
00219  *****/
00220 *
00221 * These correspond in order to the Pentium Pro counters. Add new counters at
00222 * the end. These agree with the mnemonics in the Pentium Pro Family
00223 * Developer's Manual, vol 3.
00224 *
00225 * Those events marked with a $ require a MESI unit field; those marked with
00226 * a @ require a self/any unit field. Those marked with a 0 are only supported
00227 * in counter 0; those marked with 1 are only supported in counter 1.
00228 */
00229
00230 /* Data cache unit */
00231 #define P6_DATA_MEM_REFS    0x43 /* total memory refs */
00232 #define P6_DCU_LINES_IN    0x45 /* all lines allocated in cache unit */
00233 #define P6_DCU_M_LINES_IN  0x46 /* M lines allocated in cache unit */
00234 #define P6_DCU_M_LINES_OUT 0x47 /* M lines evicted from cache */
00235 #define P6_DCU_MISS_OUTSTANDING 0x48 /* #cycles a miss is outstanding */
00236
00237 /* Instruction fetch unit */
00238 #define P6_IFU_IFETCH    0x80 /* instruction fetches */
00239 #define P6_IFU_IFETCH_MISS 0x81 /* instruction fetch misses */
00240 #define P6_ITLB_MISS    0x85 /* ITLB misses */
00241 #define P6_IFU_MEM_STALL 0x86 /* number of cycles IFU is stalled */
00242 #define P6_ILD_STALL    0x87 /* #stalls in instr length decode */
00243
00244 /* L2 Cache */
00245 #define P6_L2_IFETCH    0x28 /* ($) 12 ifetches */
00246 #define P6_L2_LD        0x29 /* ($) 12 data loads */
00247 #define P6_L2_ST        0x2a /* ($) 12 data stores */
00248 #define P6_L2_LINES_IN  0x24 /* lines allocated in L2 */
00249 #define P6_L2_LINES_OUT 0x26 /* lines removed from L2 */
00250 #define P6_L2_M_LINES_INM 0x25 /* modified lines allocated in L2 */
00251 #define P6_L2_M_LINES_OUTM 0x27 /* modified lines removed from L2 */
00252 #define P6_L2_RQSTS    0x2e /* ($) number of l2 requests */
00253 #define P6_L2_ADS      0x21 /* number of l2 addr strobes */
00254 #define P6_L2_DBUS_BUSY 0x22 /* number of data bus busy cycles */
00255 #define P6_L2_DBUS_BUSY_RD 0x23 /* #bus cycles xferring l2->cpu */
00256
00257 /* External bus logic */
00258 #define P6_BUS_DRDY_CLOCKS 0x62 /* (@) #clocks DRDY is asserted */
00259 #define P6_BUS_LOCK_CLOCKS 0x63 /* (@) #clocks LOCK is asserted */
00260 #define P6_BUS_REQ_OUTSTANDING 0x60 /* #bus requests outstanding */
00261 #define P6_BUS_TRAN_BRD    0x65 /* (@) bus burst read txns */

```

```

00262 #define P6_BUS_TRAN_RFO 0x66 /* (0) bus read for ownership txns */
00263 #define P6_BUS_TRAN_WB 0x67 /* (0) bus writeback txns */
00264 #define P6_BUS_TRAN_IFETCH 0x68 /* (0) bus instr fetch txns */
00265 #define P6_BUS_TRAN_INVALID 0x69 /* (0) bus invalidate txns */
00266 #define P6_BUS_TRAN_PWR 0x6a /* (0) bus partial write txns */
00267 #define P6_BUS_TRANS_P 0x6b /* (0) bus partial txns */
00268 #define P6_BUS_TRANS_IO 0x6c /* (0) bus I/O txns */
00269 #define P6_BUS_TRAN_DEF 0x6d /* (0) bus deferred txns */
00270 #define P6_BUS_TRAN_BURST 0x6e /* (0) bus burst txns */
00271 #define P6_BUS_TRAN_ANY 0x70 /* (0) total bus txns */
00272 #define P6_BUS_TRAN_MEM 0x6f /* (0) total memory txns */
00273 #define P6_BUS_DATA_RCV 0x64 /* #busclocks CPU is receiving data */
00274 #define P6_BUS_BNR_DRV 0x61 /* #busclocks CPU is driving BNR pin */
00275 #define P6_BUS_HIT_DRV 0x7a /* #busclocks CPU is driving HIT pin */
00276 #define P6_BUS_HITM_DRV 0x7b /* #busclocks CPU is driving HITM pin */
00277 #define P6_BUS_SNOOP_STALL 0x7e /* #clkcycles bus is snoop-stalled */
00278
00279 /* FPU */
00280 #define P6_FLOPS 0xc1 /* (0) number of FP ops retired */
00281 #define P6_FP_COMP_OPS 0x10 /* (0) computational FPOPS exec'd */
00282 #define P6_FP_ASSIST 0x11 /* (1) FP excep's handled in ucode */
00283 #define P6_MUL 0x12 /* (1) number of FP multiplies */
00284 #define P6_DIV 0x13 /* (1) number of FP divides */
00285 #define P6_CYCLES_DIV_BUSY 0x14 /* (0) number of cycles divider busy */
00286
00287 /* Memory ordering */
00288 #define P6_LD_BLOCKS 0x03 /* number of store buffer blocks */
00289 #define P6_SB_DRAINS 0x04 /* # of store buffer drain cycles */
00290 #define P6_MISALIGN_MEM_REF 0x05 /* # misaligned data memory refs */
00291
00292 /* Instruction decoding and retirement */
00293 #define P6_INST_RETIRED 0xc0 /* number of instrs retired */
00294 #define P6_UOPS_RETIRED 0xc2 /* number of micro-ops retired */
00295 #define P6_INST_DECODER 0xd0 /* number of instructions decoded */
00296
00297 /* Interrupts */
00298 #define P6_HW_INT_RX 0xc8 /* number of hardware interrupts */
00299 #define P6_CYCLES_INT_MASKED 0xc6 /* number of cycles hardints masked */
00300 #define P6_CYCLES_INT_PENDING_AND_MASKED 0xc7 /* #cycles masked but pending */
00301
00302 /* Branches */
00303 #define P6_BR_INST_RETIRED 0xc4 /* number of branch instrs retired */
00304 #define P6_BR_MISS_PRED_RETIRED 0xc5 /* number of mispred'd brs retired */
00305 #define P6_BR_TAKEN_RETIRED 0xc9 /* number of taken branches retired */
00306 #define P6_BR_MISS_PRED_TAKEN_RET 0xca /* #taken mispredictions br's retired */
00307 #define P6_BR_INST_DECODED 0xe0 /* number of branch instrs decoded */
00308 #define P6_BTБ_MISSES 0xe2 /* # of branches that missed in BTБ */
00309 #define P6_BR_BOGUS 0xe4 /* number of bogus branches */
00310 #define P6_BACLEAR 0xe6 /* # times BACLEAR is asserted */
00311
00312 /* Stalls */
00313 #define P6_RESOURCE_STALLS 0xa2 /* # resource-related stall cycles */
00314 #define P6_PARTIAL_RAT_STALLS 0xd2 /* # cycles/events for partial stalls */
00315
00316 /* Segment register loads */
00317 #define P6_SEGMENT_REG_LOADS 0x06 /* number of segment register loads */
00318
00319 /* Clocks */
00320 #define P6_CPU_CLK_UNHALTED 0x79 /* #clocks CPU is not halted */
00321
00322 /* Unit field tags */
00323 #define P6_UNIT_M 0x0800
00324 #define P6_UNIT_E 0x0400
00325 #define P6_UNIT_S 0x0200
00326 #define P6_UNIT_I 0x0100
00327 #define P6_UNIT_MESI 0x0f00
00328
00329 #define P6_UNIT_SELF 0x0000
00330 #define P6_UNIT_ANY 0x2000
00331
00332 /*****
00333  ** Flag bit definitions (used for the 'flag' field in select_p6counter()) **
00334  *****/
00335 *
00336 * The driver accepts fully-formed counter specifications from user-level.
00337 * The following flags are mnemonics for the bits that get set in the
00338 * PerfEvtSel0 and PerfEvtSel1 MSR's
00339 *
00340 */
00341 #define P6CNT_U 0x010000 /* Monitor user-level events */
00342 #define P6CNT_K 0x020000 /* Monitor kernel-level events */
00343 #define P6CNT_E 0x040000 /* Edge detect: count state transitions */
00344 #define P6CNT_PC 0x080000 /* Pin control: ?? */
00345 #define P6CNT_IE 0x100000 /* Int enable: enable interrupt on overflow */
00346 #define P6CNT_F 0x200000 /* Freeze counter (handled in software) */
00347 #define P6CNT_EN 0x400000 /* enable counters (in PerfEvtSel0) */
00348 #define P6CNT_IV 0x800000 /* Invert counter mask comparison result */

```

```

00349
00350 /*****
00351  ** Miscellaneous constants **
00352  *****/
00353  *
00354  * Number of Pentium Pro programable hardware counters.
00355  */
00356 #define NUM_P6HWC 2
00357
00358 /*****
00359  *
00360  * End of Copyright by Harvard College
00361  *
00362  *****/
00363
00364
00365 #define MSR_P6_EVTSEL0 0x186
00366 #define MSR_P6_EVTSEL1 0x187
00367 #define MSR_P6_PERFCTR0 0xc1
00368 #define MSR_P6_PERFCTR1 0xc2
00369
00370 /* P6-specific Makros to manipulate and read counters */
00371
00372 /* Read the 40 bit performance monitoring counter. This requires
00373    the PCE-flag in CR4 to be set. Otherwise GP0 is raised. Works only
00374    at P6.
00375    */
00376 #define l4_i686_rdpmc(cntnr, res_p) \
00377     __asm__ __volatile__( \
00378         "movl %2, %%ecx # put counter number in  \n\
00379         .byte 0xf; .byte 0x33 # RDPMC instruction  \n\
00380         movl %%edx, %1 # High order 32 bits  \n\
00381         movl %%eax, %0 # Low order 32 bits" \
00382         : "=g" (*(int *) (res_p)), "=g" (((int *) res_p)+1)) \
00383         : "g" (cntnr) \
00384         : "ecx", "eax", "edx") \
00385
00386 static inline l4_uint32_t l4_i686_rdpmc_32(int cntnr){
00387     l4_uint32_t x;
00388
00389     __asm__ __volatile__(
00390         ".byte 0xf; .byte 0x33 # RDPMC instruction"
00391         : "=a" (x)
00392         : "c" (cntnr)
00393         : "ecx", "eax", "edx");
00394     return x;
00395 }
00396
00397 static inline void l4_i686_select_perfctr_event(int counter,
00398     unsigned long long val){
00399     l4_i586_wrmsr(MSR_P6_EVTSEL0+counter, &val);
00400 }
00401
00402 static inline void l4_i686_select_perfctr0_event(long long *val){
00403     __asm__ volatile(
00404         "movl $MSR_P6_EVTSEL0, %%ecx\n"
00405         "movl (%ebx), %%eax\n"
00406         "movl 4(%ebx), %%edx\n"
00407         /* ".byte 0xcc, 0xeb, 0x01, 0x21\n"
00408         ".byte 0x0f, 0x30\n" // wrmsr
00409         /* ".byte 0xcc, 0xeb, 0x01, 0x21\n"
00410         : /* no output */
00411         : "b" (val)
00412         : "ax", "cx", "dx", "bx"
00413         );
00414
00415 }
00416
00417 /* end of P6 section */
00418 #else
00419
00420 #define K7CNT_U 0x010000 /* Monitor user-level events */
00421 #define K7CNT_K 0x020000 /* Monitor kernel-level events */
00422 #define K7CNT_E 0x040000 /* Edge detect: count state transitions */
00423 #define K7CNT_PC 0x080000 /* Pin control: ?? */
00424 #define K7CNT_IE 0x100000 /* Int enable: enable interrupt on overflow */
00425 #define K7CNT_F 0x200000 /* Freeze counter (handled in software) */
00426 #define K7CNT_EN 0x400000 /* enable counters (in PerfEvtSel0) */
00427 #define K7CNT_IV 0x800000 /* Invert counter mask comparison result */
00428
00429 #define MSR_K7_EVTSEL0 0xC0010000
00430 #define MSR_K7_EVTSEL1 0xC0010001
00431 #define MSR_K7_EVTSEL2 0xC0010002
00432 #define MSR_K7_EVTSEL3 0xC0010003
00433 #define MSR_K7_PERFCTR0 0xC0010004
00434 #define MSR_K7_PERFCTR1 0xC0010005
00435 #define MSR_K7_PERFCTR2 0xC0010006

```

```

00436 #define MSR_K7_PERFCTR3 0xC0010007
00437
00438 #endif
00439
00440 #endif
00441
00442 /* end of P5/P6/K7 section*/
00443 #endif
00444
00445 /* end of not only lib-prototypes section */
00446 #endif
00447
00448 L4_END_DECLS
00449
00450 #endif

```

## 17.73 amd64/l4/util/rdtsc.h File Reference

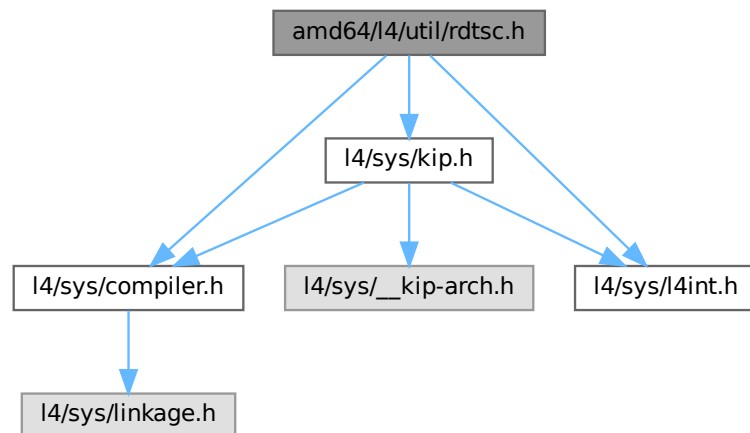
Timestamp counter related functions.

```

#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>
#include <l4/sys/kip.h>

```

Include dependency graph for rdtsc.h:



### Functions

- `l4_cpu_time_t l4_rdtsc` (void)  
*Read current value of CPU-internal timestamp counter.*
- `l4_uint32_t l4_rdtsc_32` (void)  
*Read the lest significant 32 bit of the TSC.*
- `l4_uint64_t l4_rdpmc` (int ecx)  
*Return current value of CPU-internal performance measurement counter.*
- `l4_uint32_t l4_rdpmc_32` (int ecx)  
*Return the least significant 32 bit of a performance counter.*
- `l4_uint64_t l4_tsc_to_ns` (`l4_cpu_time_t` tsc)

- Convert timestamp to ns value.*
- [l4\\_uint64\\_t l4\\_tsc\\_to\\_us \(l4\\_cpu\\_time\\_t tsc\)](#)
- Convert timestamp into micro seconds value.*
- [void l4\\_tsc\\_to\\_s\\_and\\_ns \(l4\\_cpu\\_time\\_t tsc, l4\\_uint32\\_t \\*s, l4\\_uint32\\_t \\*ns\)](#)
- Convert timestamp to s.ns value.*
- [l4\\_cpu\\_time\\_t l4\\_ns\\_to\\_tsc \(l4\\_uint64\\_t ns\)](#)
- Convert nano seconds into CPU ticks.*
- [void l4\\_busy\\_wait\\_ns \(l4\\_uint64\\_t ns\)](#)
- Wait busy for a small amount of time.*
- [void l4\\_busy\\_wait\\_us \(l4\\_uint64\\_t us\)](#)
- Wait busy for a small amount of time.*
- [l4\\_uint32\\_t l4\\_calibrate\\_tsc \(l4\\_kernel\\_info\\_t const \\*kip\)](#)
- Determine scalers for timestamp calculations.*
- [l4\\_uint32\\_t l4\\_tsc\\_init \(l4\\_kernel\\_info\\_t const \\*kip\)](#)
- Initialize scaler for TSC calibrations from the kernel.*
- [l4\\_uint32\\_t l4\\_get\\_hz \(void\)](#)
- Get CPU frequency in Hz.*

### 17.73.1 Detailed Description

Timestamp counter related functions.

#### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [rdtsc.h](#).

## 17.74 rdtsc.h

[Go to the documentation of this file.](#)

```

00001
00008
00009 /*
00010  * (c) 2003-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00011  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00012  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00013  *      economic rights: Technische Universität Dresden (Germany)
00014  * License: see LICENSE.spdx (in this directory or the directories above)
00015  */
00016
00017 #ifndef __l4_rdtsc_h
00018 #define __l4_rdtsc_h
00019
00024
00025 #include <l4/sys/compiler.h>
00026 #include <l4/sys/l4int.h>
00027 #include <l4/sys/kip.h>
00028
00029 L4_BEGIN_DECLS
00030
00031 /* interface */
00036
00037 extern l4_uint32_t l4_scaler_tsc_to_ns;
00038 extern l4_uint32_t l4_scaler_tsc_to_us;
00039 extern l4_uint32_t l4_scaler_ns_to_tsc;
00040 extern l4_uint32_t l4_scaler_tsc_linux;
00041
00046 L4_INLINE l4_cpu_time_t
00047 l4_rdtsc (void);

```

```

00048
00054 L4_INLINE
00055 l4_uint32_t l4_rdtsc_32(void);
00056
00063 L4_INLINE l4_uint64_t
00064 l4_rdpmc (int ecx);
00065
00071 L4_INLINE
00072 l4_uint32_t l4_rdpmc_32(int ecx);
00073
00078 L4_INLINE l4_uint64_t
00079 l4_tsc_to_ns (l4_cpu_time_t tsc);
00080
00085 L4_INLINE l4_uint64_t
00086 l4_tsc_to_us (l4_cpu_time_t tsc);
00087
00093 L4_INLINE void
00094 l4_tsc_to_s_and_ns (l4_cpu_time_t tsc, l4_uint32_t *s, l4_uint32_t *ns);
00095
00101 L4_INLINE l4_cpu_time_t
00102 l4_ns_to_tsc (l4_uint64_t ns);
00103
00109 L4_INLINE void
00110 l4_busy_wait_ns (l4_uint64_t ns);
00111
00117 L4_INLINE void
00118 l4_busy_wait_us (l4_uint64_t us);
00119
00126 L4_INLINE l4_uint32_t
00127 l4_calibrate_tsc(l4_kernel_info_t const *kip);
00128
00142 L4_CV l4_uint32_t
00143 l4_tsc_init(l4_kernel_info_t const *kip);
00144
00149 L4_CV l4_uint32_t
00150 l4_get_hz (void);
00151
00153
00154 L4_END_DECLS
00155
00156 /* implementation */
00157
00158 L4_INLINE l4_uint32_t
00159 l4_calibrate_tsc(l4_kernel_info_t const *kip)
00160 {
00161     return l4_tsc_init(kip);
00162 }
00163
00164 L4_INLINE l4_cpu_time_t
00165 l4_rdtsc (void)
00166 {
00167     l4_umword_t lo, hi;
00168
00169     __asm__ __volatile__ ("rdtsc" : "=a"(lo), "=d"(hi));
00170
00171     return ((l4_cpu_time_t)hi « 32) | lo;
00172 }
00173
00174 L4_INLINE l4_uint64_t
00175 l4_rdpmc (int ecx)
00176 {
00177     l4_umword_t lo, hi;
00178
00179     __asm__ __volatile__ ("rdpmc" : "=a"(lo), "=d"(hi) : "c"(ecx));
00180
00181     return ((l4_cpu_time_t)hi « 32) | lo;
00182 }
00183
00184 L4_INLINE
00185 l4_uint32_t l4_rdtsc_32(void)
00186 {
00187     l4_umword_t lo, hi;
00188
00189     __asm__ __volatile__ ("rdtsc" : "=a"(lo), "=d"(hi));
00190
00191     return lo;
00192 }
00193
00194 L4_INLINE
00195 l4_uint32_t l4_rdpmc_32(int ecx)
00196 {
00197     l4_umword_t lo, hi;
00198
00199     __asm__ __volatile__ ("rdpmc" : "=a"(lo), "=d"(hi) : "c"(ecx));
00200
00201     return lo;
00202 }

```

```

00203
00204 L4_INLINE l4_uint64_t
00205 l4_tsc_to_ns (l4_cpu_time_t tsc)
00206 {
00207     l4_uint64_t ns, dummy;
00208     __asm__
00209     ("
00210      \"mulq  %3      \n\t\"
00211       \"shrd  $27, %%rdx, %%rax      \n\t\"
00212      :\"=a\" (ns), \"=d\" (dummy)
00213      :\"a\" (tsc), \"r\" ((l4_uint64_t)l4_scaler_tsc_to_ns)
00214      );
00215     return ns;
00216 }
00217
00218 L4_INLINE l4_uint64_t
00219 l4_tsc_to_us (l4_cpu_time_t tsc)
00220 {
00221     l4_uint64_t ns, dummy;
00222     __asm__
00223     ("
00224      \"mulq  %3      \n\t\"
00225       \"shrd  $32, %%rdx, %%rax      \n\t\"
00226      :\"=a\" (ns), \"=d\" (dummy)
00227      :\"a\" (tsc), \"r\" ((l4_uint64_t)l4_scaler_tsc_to_us)
00228      );
00229     return ns;
00230 }
00231
00232 L4_INLINE void
00233 l4_tsc_to_s_and_ns (l4_cpu_time_t tsc, l4_uint32_t *s, l4_uint32_t *ns)
00234 {
00235     __asm__
00236     ("
00237      \"mulq  %3      \n\t\"
00238       \"shrd  $27, %%rdx, %%rax      \n\t\"
00239       \"xorq  %%rdx, %%rdx      \n\t\"
00240       \"divq  %4      \n\t\"
00241      :\"=a\" (*s), \"=d\" (*ns)
00242      :\"a\" (tsc), \"r\" ((l4_uint64_t)l4_scaler_tsc_to_ns),
00243       \"rm\" (1000000000ULL)
00244      );
00245 }
00246
00247 L4_INLINE l4_cpu_time_t
00248 l4_ns_to_tsc (l4_uint64_t ns)
00249 {
00250     l4_uint64_t tsc, dummy;
00251     __asm__
00252     ("
00253      \"mulq  %3      \n\t\"
00254       \"shrd  $27, %%rdx, %%rax      \n\t\"
00255      :\"=a\" (tsc), \"=d\" (dummy)
00256      :\"a\" (ns), \"r\" ((l4_uint64_t)l4_scaler_ns_to_tsc)
00257      );
00258     return tsc;
00259 }
00260
00261 L4_INLINE void
00262 l4_busy_wait_ns (l4_uint64_t ns)
00263 {
00264     l4_cpu_time_t stop = l4_rdtsc();
00265     stop += l4_ns_to_tsc(ns);
00266
00267     while (l4_rdtsc() < stop)
00268         ;
00269 }
00270
00271 L4_INLINE void
00272 l4_busy_wait_us (l4_uint64_t us)
00273 {
00274     l4_cpu_time_t stop = l4_rdtsc ();
00275     stop += l4_ns_to_tsc(us*1000ULL);
00276
00277     while (l4_rdtsc() < stop)
00278         ;
00279 }
00280
00281 #endif /* __l4_rdtsc_h */
00282

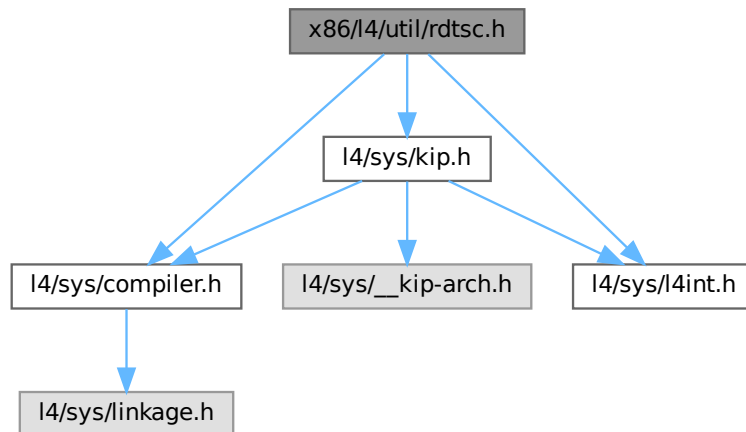
```



## 17.75 x86/l4/util/rdtsc.h File Reference

Timestamp counter related functions.

```
#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>
#include <l4/sys/kip.h>
Include dependency graph for rdtsc.h:
```



### Functions

- [l4\\_cpu\\_time\\_t l4\\_rdtsc](#) (void)  
*Read current value of CPU-internal timestamp counter.*
- [l4\\_uint32\\_t l4\\_rdtsc\\_32](#) (void)  
*Read the least significant 32 bit of the TSC.*
- [l4\\_uint64\\_t l4\\_rdpmc](#) (int ecx)  
*Return current value of CPU-internal performance measurement counter.*
- [l4\\_uint32\\_t l4\\_rdpmc\\_32](#) (int ecx)  
*Return the least significant 32 bit of a performance counter.*
- [l4\\_uint64\\_t l4\\_tsc\\_to\\_ns](#) ([l4\\_cpu\\_time\\_t](#) tsc)  
*Convert timestamp to ns value.*
- [l4\\_uint64\\_t l4\\_tsc\\_to\\_us](#) ([l4\\_cpu\\_time\\_t](#) tsc)  
*Convert timestamp into micro seconds value.*
- void [l4\\_tsc\\_to\\_s\\_and\\_ns](#) ([l4\\_cpu\\_time\\_t](#) tsc, [l4\\_uint32\\_t](#) \*s, [l4\\_uint32\\_t](#) \*ns)  
*Convert timestamp to s.ns value.*
- [l4\\_cpu\\_time\\_t l4\\_ns\\_to\\_tsc](#) ([l4\\_uint64\\_t](#) ns)  
*Convert nano seconds into CPU ticks.*
- void [l4\\_busy\\_wait\\_ns](#) ([l4\\_uint64\\_t](#) ns)  
*Wait busy for a small amount of time.*
- void [l4\\_busy\\_wait\\_us](#) ([l4\\_uint64\\_t](#) us)  
*Wait busy for a small amount of time.*
- [l4\\_uint32\\_t l4\\_calibrate\\_tsc](#) ([l4\\_kernel\\_info\\_t](#) const \*kip)

*Determine scalers for timestamp calculations.*

- [l4\\_uint32\\_t l4\\_tsc\\_init](#) ([l4\\_kernel\\_info\\_t](#) const \*kip)

*Initialize scaler for TSC calibrations from the kernel.*

- [l4\\_uint32\\_t l4\\_get\\_hz](#) (void)

*Get CPU frequency in Hz.*

## 17.75.1 Detailed Description

Timestamp counter related functions.

### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [rdtsc.h](#).

## 17.76 rdtsc.h

[Go to the documentation of this file.](#)

```

00001
00008
00009 /*
00010  * (c) 2003-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00011  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00012  *      Frank Mehnert <fm3@os.inf.tu-dresden.de>,
00013  *      Jork Löser <jork@os.inf.tu-dresden.de>,
00014  *      Martin Pohlack <mp26@os.inf.tu-dresden.de>
00015  *      economic rights: Technische Universität Dresden (Germany)
00016  * License: see LICENSE.spdx (in this directory or the directories above)
00017  */
00018
00019 #ifndef __l4_rdtsc_h
00020 #define __l4_rdtsc_h
00021
00026
00027 #include <l4/sys/compiler.h>
00028 #include <l4/sys/l4int.h>
00029 #include <l4/sys/kip.h>
00030
00031 L4_BEGIN_DECLS
00032
00033 /* interface */
00038
00039 extern l4_uint32_t l4_scaler_tsc_to_ns;
00040 extern l4_uint32_t l4_scaler_tsc_to_us;
00041 extern l4_uint32_t l4_scaler_ns_to_tsc;
00042 extern l4_uint32_t l4_scaler_tsc_linux;
00043
00048 L4_INLINE l4_cpu_time_t
00049 l4_rdtsc (void);
00050
00056 L4_INLINE
00057 l4_uint32_t l4_rdtsc_32(void);
00058
00065 L4_INLINE l4_uint64_t
00066 l4_rdpmc (int ecx);
00067
00073 L4_INLINE
00074 l4_uint32_t l4_rdpmc_32(int ecx);
00075
00080 L4_INLINE l4_uint64_t
00081 l4_tsc_to_ns (l4_cpu_time_t tsc);
00082
00087 L4_INLINE l4_uint64_t
00088 l4_tsc_to_us (l4_cpu_time_t tsc);
00089
00095 L4_INLINE void
00096 l4_tsc_to_s_and_ns (l4_cpu_time_t tsc, l4_uint32_t *s, l4_uint32_t *ns);
00097

```

```

00103 L4_INLINE l4_cpu_time_t
00104 l4_ns_to_tsc (l4_uint64_t ns);
00105
00111 L4_INLINE void
00112 l4_busy_wait_ns (l4_uint64_t ns);
00113
00119 L4_INLINE void
00120 l4_busy_wait_us (l4_uint64_t us);
00121
00128 L4_INLINE l4_uint32_t
00129 l4_calibrate_tsc(l4_kernel_info_t const *kip);
00130
00144 L4_CV l4_uint32_t
00145 l4_tsc_init(l4_kernel_info_t const *kip);
00146
00151 L4_CV l4_uint32_t
00152 l4_get_hz (void);
00153
00155
00156 L4_END_DECLS
00157
00158 /* implementation */
00159
00160 L4_INLINE l4_uint32_t
00161 l4_calibrate_tsc(l4_kernel_info_t const *kip)
00162 {
00163     return l4_tsc_init(kip);
00164 }
00165
00166 L4_INLINE l4_cpu_time_t
00167 l4_rdtsc (void)
00168 {
00169     l4_cpu_time_t v;
00170
00171     __asm__ __volatile__ (
00172         "\n\t"
00173         ".byte 0x0f, 0x31\n\t"
00174         /*"rdtsc\n\t"*/
00175         :
00176         "=A" (v)
00177         : /* no inputs */
00178         );
00179
00180     return v;
00181 }
00182
00183 /* the same, but only 32 bit. Useful for smaller differences,
00184    needs less cycles. */
00185 L4_INLINE
00186 l4_uint32_t l4_rdtsc_32(void)
00187 {
00188     l4_uint32_t x;
00189
00190     __asm__ __volatile__ (
00191         ".byte 0x0f, 0x31\n\t" // rdtsc
00192         : "=a" (x)
00193         :
00194         : "edx");
00195
00196     return x;
00197 }
00198
00199 L4_INLINE l4_uint64_t
00200 l4_rdpmc (int ecx)
00201 {
00202     l4_cpu_time_t v;
00203
00204     __asm__ __volatile__ (
00205         "rdpmc\n\t"
00206         :
00207         "=A" (v)
00208         : "c" (ecx)
00209         );
00210
00211     return v;
00212 }
00213
00214 /* the same, but only 32 bit. Useful for smaller differences,
00215    needs less cycles. */
00216 L4_INLINE
00217 l4_uint32_t l4_rdpmc_32(int ecx)
00218 {
00219     l4_uint32_t x;
00220
00221     __asm__ __volatile__ (
00222         "rdpmc\n\t"
00223         : "=a" (x)

```

```

00224         : "c" (ecx)
00225         : "edx");
00226
00227     return x;
00228 }
00229
00230 L4_INLINE l4_uint64_t
00231 l4_tsc_to_ns (l4_cpu_time_t tsc)
00232 {
00233     l4_uint32_t dummy;
00234     l4_uint64_t ns;
00235     __asm__
00236     ("
00237      \"movl %%edx, %%ecx    \n\t\"
00238      \"mull  %3            \n\t\"
00239      \"movl %%ecx, %%eax    \n\t\"
00240      \"movl %%edx, %%ecx    \n\t\"
00241      \"mull  %3            \n\t\"
00242      \"addl %%ecx, %%eax    \n\t\"
00243      \"adcl  $0, %%edx      \n\t\"
00244      \"shld  $5, %%eax, %%edx \n\t\"
00245      \"shll  $5, %%eax      \n\t\"
00246      :\"=A\" (ns),
00247       \"=&c\" (dummy)
00248      :\"0\" (tsc),
00249       \"g\" (l4_scaler_tsc_to_ns)
00250     );
00251     return ns;
00252 }
00253
00254 L4_INLINE l4_uint64_t
00255 l4_tsc_to_us (l4_cpu_time_t tsc)
00256 {
00257     l4_uint32_t dummy;
00258     l4_uint64_t us;
00259     __asm__
00260     ("
00261      \"movl %%edx, %%ecx    \n\t\"
00262      \"mull  %3            \n\t\"
00263      \"movl %%ecx, %%eax    \n\t\"
00264      \"movl %%edx, %%ecx    \n\t\"
00265      \"mull  %3            \n\t\"
00266      \"addl %%ecx, %%eax    \n\t\"
00267      \"adcl  $0, %%edx      \n\t\"
00268      :\"=A\" (us),
00269       \"=&c\" (dummy)
00270      :\"0\" (tsc),
00271       \"g\" (l4_scaler_tsc_to_us)
00272     );
00273     return us;
00274 }
00275
00276 L4_INLINE void
00277 l4_tsc_to_s_and_ns (l4_cpu_time_t tsc, l4_uint32_t *s, l4_uint32_t *ns)
00278 {
00279     l4_uint32_t dummy;
00280     __asm__
00281     ("
00282      \"movl %%edx, %%ecx    \n\t\"
00283      \"mull  %4            \n\t\"
00284      \"movl %%ecx, %%eax    \n\t\"
00285      \"movl %%edx, %%ecx    \n\t\"
00286      \"mull  %4            \n\t\"
00287      \"addl %%ecx, %%eax    \n\t\"
00288      \"adcl  $0, %%edx      \n\t\"
00289      \"movl $1000000000, %%ecx \n\t\"
00290      \"shld  $5, %%eax, %%edx \n\t\"
00291      \"shll  $5, %%eax      \n\t\"
00292      \"divl  %%ecx          \n\t\"
00293      :\"=a\" (*s), \"=d\" (*ns), \"=&c\" (dummy)
00294      : \"A\" (tsc), \"g\" (l4_scaler_tsc_to_ns)
00295     );
00296 }
00297
00298 L4_INLINE l4_cpu_time_t
00299 l4_ns_to_tsc (l4_uint64_t ns)
00300 {
00301     l4_uint32_t dummy;
00302     l4_cpu_time_t tsc;
00303     __asm__
00304     ("
00305      \"movl %%edx, %%ecx    \n\t\"
00306      \"mull  %3            \n\t\"
00307      \"movl %%ecx, %%eax    \n\t\"
00308      \"movl %%edx, %%ecx    \n\t\"
00309      \"mull  %3            \n\t\"
00310      \"addl %%ecx, %%eax    \n\t\"

```

```

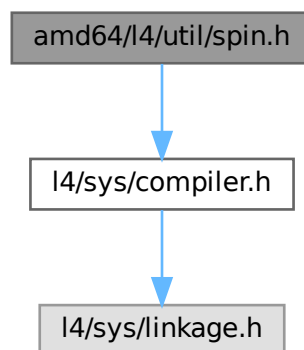
00311     "adcl  $0, %%edx  \n\t"
00312     "shld  $5, %%eax, %%edx  \n\t"
00313     "shll  $5, %%eax  \n\t"
00314     : "=A" (tsc),
00315     "=&c" (dummy)
00316     : "0" (ns),
00317     "g" (l4_scaler_ns_to_tsc)
00318     );
00319     return tsc;
00320 }
00321
00322 L4_INLINE void
00323 l4_busy_wait_ns (l4_uint64_t ns)
00324 {
00325     l4_cpu_time_t stop = l4_rdtsc();
00326     stop += l4_ns_to_tsc(ns);
00327
00328     while (l4_rdtsc() < stop)
00329         ;
00330 }
00331
00332 L4_INLINE void
00333 l4_busy_wait_us (l4_uint64_t us)
00334 {
00335     l4_cpu_time_t stop = l4_rdtsc ();
00336     stop += l4_ns_to_tsc(us*1000ULL);
00337
00338     while (l4_rdtsc() < stop)
00339         ;
00340 }
00341
00342 #endif /* __l4_rdtsc_h */
00343

```

## 17.77 amd64/l4/util/spin.h File Reference

Spinning for amd64.

#include <l4/sys/compiler.h>  
 Include dependency graph for spin.h:



### 17.77.1 Detailed Description

Spinning for amd64.

Definition in file [spin.h](#).

## 17.78 spin.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #ifndef __l4util_spin_h
00012 #define __l4util_spin_h
00013
00014 #include <l4/sys/compiler.h>
00015
00016 L4_BEGIN_DECLS
00017
00018 L4_CV void l4_spin(int x,int y);
00019 L4_CV void l4_spin_vga(int x,int y);
00020 L4_CV void l4_spin_n_text(int x, int y, int len, const char*s);
00021 L4_CV void l4_spin_n_text_vga(int x, int y, int len, const char*s);
00022
00023 /*****
00024  *
00025  * spin_text()      - spinning wheel at the hercules screen. The given text
00026  *                   must be a text constant, no variables or arrays. Its
00027  *                   size is determined with the sizeof operator, it's much
00028  *                   faster than the strlen function.
00029  * spin_text_vga() - same for vga.
00030  *
00031  *****/
00032 #define l4_spin_text(x, y, text) \
00033     l4_spin_n_text((x), (y), sizeof(text)-1, "" text)
00034 #define l4_spin_text_vga(x, y, text) \
00035     l4_spin_n_text_vga((x), (y), sizeof(text)-1, "" text)
00036
00037 L4_END_DECLS
00038
00039 #endif

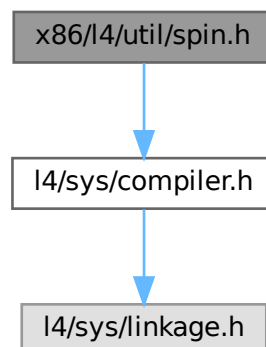
```

## 17.79 x86/l4/util/spin.h File Reference

Spinning for x86.

```
#include <l4/sys/compiler.h>
```

Include dependency graph for spin.h:



## 17.79.1 Detailed Description

Spinning for x86.

Definition in file [spin.h](#).

## 17.80 spin.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Frank Mehnert <fm3@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #ifndef __l4util_spin_h
00012 #define __l4util_spin_h
00013
00014 #include <l4/sys/compiler.h>
00015
00016 L4_BEGIN_DECLS
00017
00018 L4_CV void l4_spin(int x,int y);
00019 L4_CV void l4_spin_vga(int x,int y);
00020 L4_CV void l4_spin_n_text(int x, int y, int len, const char*s);
00021 L4_CV void l4_spin_n_text_vga(int x, int y, int len, const char*s);
00022
00023 /*****
00024  *
00025  * spin_text()      - spinning wheel at the hercules screen. The given text
00026  *                  must be a text constant, no variables or arrays. Its
00027  *                  size is determined with the sizeof operator, it's much
00028  *                  faster than the strlen function.
00029  * spin_text_vga() - same for vga.
00030  *
00031  *****/
00032 #define l4_spin_text(x, y, text) \
00033   l4_spin_n_text((x), (y), sizeof(text)-1, "" text)
00034 #define l4_spin_text_vga(x, y, text) \
00035   l4_spin_n_text_vga((x), (y), sizeof(text)-1, "" text)
00036
00037 L4_END_DECLS
00038
00039 #endif

```

## 17.81 amd64/l4/sys/segment.h File Reference

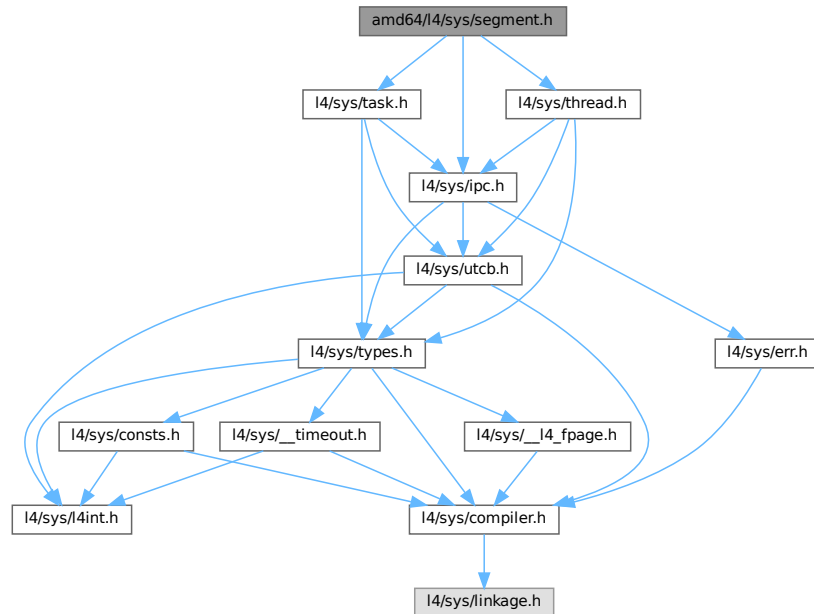
Segment handling (AMD64).

```

#include <l4/sys/ipc.h>
#include <l4/sys/task.h>
#include <l4/sys/thread.h>

```

Include dependency graph for segment.h:



## Enumerations

- enum [L4\\_task\\_ldt\\_x86\\_consts](#) { [L4\\_TASK\\_LDT\\_X86\\_ENTRY\\_SIZE](#) = 8 , [L4\\_TASK\\_LDT\\_X86\\_MAX\\_ENTRIES](#) }
- Constants for LDT handling.*
- enum [L4\\_sys\\_segment](#) { [L4\\_AMD64\\_SEGMENT\\_FS](#) = 0 , [L4\\_AMD64\\_SEGMENT\\_GS](#) = 1 }
- Constants for identifying segments.*

## Functions

- long [fiasco\\_ldt\\_set](#) ([l4\\_cap\\_idx\\_t](#) task, void \*ldt, unsigned int num\_desc, unsigned int entry\_number\_start, [l4\\_utcb\\_t](#) \*utcb)  
*Set LDT segments descriptors.*
- long [fiasco\\_gdt\\_set](#) ([l4\\_cap\\_idx\\_t](#) thread, void \*desc, unsigned int size, unsigned int entry\_number\_start, [l4\\_utcb\\_t](#) \*utcb)  
*Set GDT segment descriptors.*
- unsigned [fiasco\\_gdt\\_get\\_entry\\_offset](#) ([l4\\_cap\\_idx\\_t](#) thread, [l4\\_utcb\\_t](#) \*utcb)  
*Return the offset of the entry in the GDT.*
- long [fiasco\\_amd64\\_set\\_fs](#) ([l4\\_cap\\_idx\\_t](#) thread, [l4\\_umword\\_t](#) base, [l4\\_utcb\\_t](#) \*utcb)  
*Set the base address for the FS segment.*
- long [fiasco\\_amd64\\_set\\_segment\\_base](#) ([l4\\_cap\\_idx\\_t](#) thread, enum [L4\\_sys\\_segment](#) segr, [l4\\_umword\\_t](#) base, [l4\\_utcb\\_t](#) \*utcb)  
*Set the base address for a segment.*
- long [fiasco\\_amd64\\_segment\\_info](#) ([l4\\_cap\\_idx\\_t](#) thread, unsigned \*user\_ds, unsigned \*user\_cs, unsigned \*user32\_cs, [l4\\_utcb\\_t](#) \*utcb)  
*Get segment information.*



## 17.81.1 Detailed Description

Segment handling (AMD64).

Definition in file [segment.h](#).

## 17.81.2 Enumeration Type Documentation

### 17.81.2.1 L4\_sys\_segment

```
enum L4_sys_segment
```

Constants for identifying segments.

#### Enumerator

L4_AMD64_SEGMENT_FS	Constant identifying the FS segment.
L4_AMD64_SEGMENT_GS	Constant identifying the GS segment.

Definition at line 106 of file [segment.h](#).

### 17.81.2.2 L4\_task\_ldt\_x86\_consts

```
enum L4_task_ldt_x86_consts
```

Constants for LDT handling.

#### Enumerator

L4_TASK_LDT_X86_ENTRY_SIZE	Size of an LDT entry.
L4_TASK_LDT_X86_MAX_ENTRIES	Maximum number of LDT entries that can be written with one call.

Definition at line 75 of file [segment.h](#).

## 17.81.3 Function Documentation

### 17.81.3.1 fiasco\_amd64\_segment\_info()

```
long fiasco_amd64_segment_info (  
    l4_cap_idx_t thread,  
    unsigned * user_ds,  
    unsigned * user_cs,  
    unsigned * user32_cs,  
    l4_utcb_t * utcb) [inline]
```

Get segment information.

#### Parameters

---

in	<i>thread</i>	Thread to get info from.
out	<i>user_ds</i>	DS segment selector.
out	<i>user_cs</i>	64-bit CS segment selector.
out	<i>user32_cs</i>	32-bit CS segment selector.
	<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

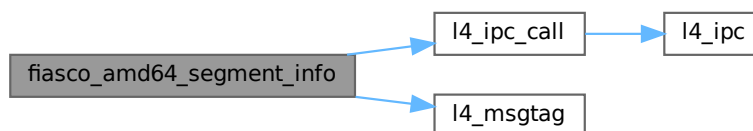
### Returns

System call error

Definition at line 175 of file [segment.h](#).

References [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_THREAD](#), [L4\\_THREAD\\_AMD64\\_GET\\_SEGMENT\\_INFO\\_OP](#), and [l4\\_msg\\_regs\\_t::mr](#).

Here is the call graph for this function:



### 17.81.3.2 fiasco\_amd64\_set\_fs()

```

long fiasco_amd64_set_fs (
    l4_cap_idx_t thread,
    l4_umword_t base,
    l4_utcb_t * utcb) [inline]

```

Set the base address for the FS segment.

### Parameters

<i>thread</i>	Thread for which the FS base address shall be modified.
<i>base</i>	Base address.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

### Return values

<i>L4_EOK</i>	Success.
---------------	----------

<code>-L4_EINVAL</code>	Invalid base address ( <i>base</i> ).
<code>-L4_ENOSYS</code>	Operation not supported with current kernel configuration.

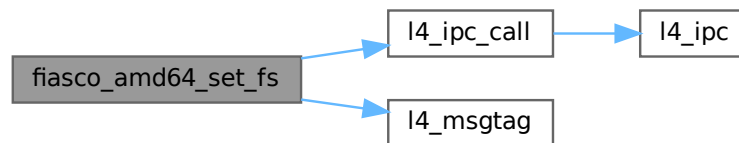
**Note**

Calling this function is equivalent to calling `fiasco_amd64_set_segment_base(thread, L4_↵AMD64_SEGMENT_FS, base, utcb)`.

Definition at line 24 of file [segment.h](#).

References [L4\\_AMD64\\_SEGMENT\\_FS](#), [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_THREAD](#), and [L4\\_THREAD\\_AMD64\\_SET\\_SEGMENT\\_BASE\\_OP](#).

Here is the call graph for this function:

**17.81.3.3 fiasco\_amd64\_set\_segment\_base()**

```

long fiasco_amd64_set_segment_base (
    l4_cap_idx_t thread,
    enum L4_sys_segment segr,
    l4_umword_t base,
    l4_utcb_t * utcb) [inline]

```

Set the base address for a segment.

**Parameters**

<i>thread</i>	Thread for which the base address of the selected segment shall be modified.
<i>segr</i>	Segment to modify (one of <a href="#">L4_sys_segment</a> ).
<i>base</i>	Base address.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

**Return values**

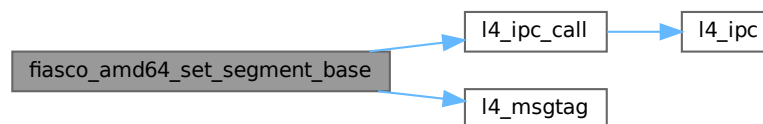
<code>L4_EOK</code>	Success.
---------------------	----------

<code>-L4_EINVAL</code>	Invalid segment ( <code>segr</code> ) or base address ( <code>base</code> ).
<code>-L4_ENOSYS</code>	Operation not supported with current kernel configuration.

Definition at line 32 of file [segment.h](#).

References [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_THREAD](#), and [L4\\_THREAD\\_AMD64\\_SET\\_SEGMENT\\_BASE](#).

Here is the call graph for this function:



## 17.82 segment.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 /*****
00013 #ifndef __L4_SYS_ARCH_X86__SEGMENT_H__
00014 #define __L4_SYS_ARCH_X86__SEGMENT_H__
00015
00016 #ifndef L4API_l4f
00017 #error This header file can only be used with a L4API version!
00018 #endif
00019
00020 #include <l4/sys/ipc.h>
00021
00039 L4_INLINE long
00040 fiasco_ldt_set(l4_cap_idx_t task, void *ldt, unsigned int num_desc,
00041               unsigned int entry_number_start, l4_utcb_t *utcb);
00042
00059 L4_INLINE long
00060 fiasco_gdt_set(l4_cap_idx_t thread, void *desc, unsigned int size,
00061               unsigned int entry_number_start, l4_utcb_t *utcb);
00062
00069 L4_INLINE unsigned
00070 fiasco_gdt_get_entry_offset(l4_cap_idx_t thread, l4_utcb_t *utcb);
00071
00075 enum L4_task_ldt_x86_consts
00076 {
00078     L4_TASK_LDT_X86_ENTRY_SIZE = 8,
00080     L4_TASK_LDT_X86_MAX_ENTRIES
00081         = (L4_UTCB_GENERIC_DATA_SIZE - 2)
00082           / (L4_TASK_LDT_X86_ENTRY_SIZE / (L4_MWORD_BITS / 8)),
00083 };
00084
00100 L4_INLINE long
00101 fiasco_amd64_set_fs(l4_cap_idx_t thread, l4_umword_t base, l4_utcb_t *utcb);
00102
00106 enum L4_sys_segment
00107 {
00109     L4_AMD64_SEGMENT_FS = 0,
00111     L4_AMD64_SEGMENT_GS = 1
00112 };
00113
00127 L4_INLINE long

```

```

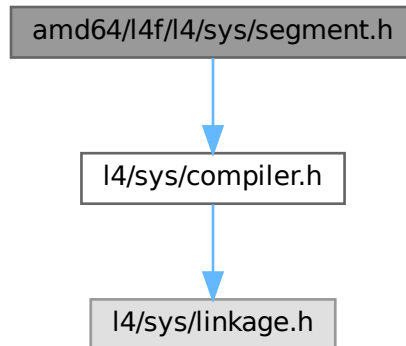
00128 fiasco_amd64_set_segment_base(l4_cap_idx_t thread, enum L4_sys_segment segr,
00129                               l4_umword_t base, l4_utcb_t *utcb);
00130
00140 L4_INLINE long
00141 fiasco_amd64_segment_info(l4_cap_idx_t thread, unsigned *user_ds,
00142                           unsigned *user_cs, unsigned *user32_cs,
00143                           l4_utcb_t *utcb);
00144
00145 /*****
00146 *** Implementation
00147 *****/
00148
00149 #include <l4/sys/task.h>
00150 #include <l4/sys/thread.h>
00151
00152 L4_INLINE long
00153 fiasco_ldt_set(l4_cap_idx_t task, void *ldt, unsigned int num_desc,
00154               unsigned int entry_number_start, l4_utcb_t *utcb)
00155 {
00156     if (num_desc > L4_TASK_LDT_X86_MAX_ENTRIES)
00157         return -L4_EINVAL;
00158     l4_utcb_mr_u(utcb)->mr[0] = L4_TASK_LDT_SET_X86_OP;
00159     l4_utcb_mr_u(utcb)->mr[1] = entry_number_start;
00160     __builtin_memcpy(&l4_utcb_mr_u(utcb)->mr[2], ldt,
00161                     num_desc * L4_TASK_LDT_X86_ENTRY_SIZE);
00162     return l4_error_u(l4_ipc_call(task, utcb, l4_msgtag(L4_PROTO_TASK, 2 + num_desc * 2, 0, 0),
00163                       L4_IPC_NEVER), utcb);
00164 }
00165
00165 L4_INLINE unsigned
00166 fiasco_gdt_get_entry_offset(l4_cap_idx_t thread, l4_utcb_t *utcb)
00167 {
00168     l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_X86_GDT_OP;
00169     if (l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER), utcb))
00170         return -1;
00171     return l4_utcb_mr_u(utcb)->mr[0];
00172 }
00173
00174 L4_INLINE long
00175 fiasco_amd64_segment_info(l4_cap_idx_t thread, unsigned *user_ds,
00176                           unsigned *user_cs, unsigned *user32_cs,
00177                           l4_utcb_t *utcb)
00178 {
00179     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00180     int r;
00181
00182     m->mr[0] = L4_THREAD_AMD64_GET_SEGMENT_INFO_OP;
00183
00184     r = l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0),
00185               L4_IPC_NEVER), utcb);
00186     if (r < 0)
00187         return r;
00188
00189     *user_ds = m->mr[0];
00190     *user_cs = m->mr[1];
00191     *user32_cs = m->mr[2];
00192
00193     return 0;
00194 }
00195
00196 #endif /* ! __L4_SYS_ARCH_X86_SEGMENT_H */

```

## 17.83 amd64/l4f/l4/sys/segment.h File Reference

l4f-specific fs/gs manipulation (AMD64).

```
#include <l4/sys/compiler.h>
Include dependency graph for segment.h:
```



## Functions

- long [fiasco\\_amd64\\_set\\_fs](#) ([l4\\_cap\\_idx\\_t](#) thread, [l4\\_umword\\_t](#) base, [l4\\_utcb\\_t](#) \*utcb)  
*Set the base address for the FS segment.*
- long [fiasco\\_amd64\\_set\\_segment\\_base](#) ([l4\\_cap\\_idx\\_t](#) thread, enum [L4\\_sys\\_segment](#) segr, [l4\\_umword\\_t](#) base, [l4\\_utcb\\_t](#) \*utcb)  
*Set the base address for a segment.*
- long [fiasco\\_gdt\\_set](#) ([l4\\_cap\\_idx\\_t](#) thread, void \*desc, unsigned int size, unsigned int entry\_number\_start, [l4\\_utcb\\_t](#) \*utcb)  
*Set GDT segment descriptors.*

### 17.83.1 Detailed Description

l4f-specific fs/gs manipulation (AMD64).

Definition in file [segment.h](#).

### 17.83.2 Function Documentation

#### 17.83.2.1 fiasco\_amd64\_set\_fs()

```
long fiasco_amd64_set_fs (
    l4\_cap\_idx\_t thread,
    l4\_umword\_t base,
    l4\_utcb\_t * utcb) [inline]
```

Set the base address for the FS segment.

#### Parameters

---

<i>thread</i>	Thread for which the FS base address shall be modified.
<i>base</i>	Base address.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

### Return values

<i>L4_EOK</i>	Success.
<i>-L4_EINVAL</i>	Invalid base address ( <i>base</i> ).
<i>-L4_ENOSYS</i>	Operation not supported with current kernel configuration.

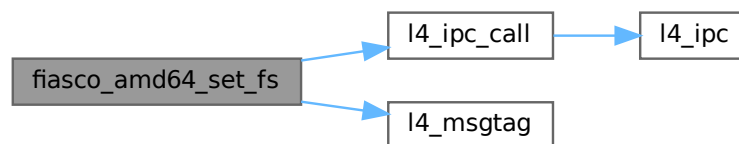
### Note

Calling this function is equivalent to calling `fiasco_amd64_set_segment_base(thread, L4_↔AMD64_SEGMENT_FS, base, utcb)`.

Definition at line 24 of file [segment.h](#).

References [L4\\_AMD64\\_SEGMENT\\_FS](#), [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_THREAD](#), and [L4\\_THREAD\\_AMD64\\_SET\\_SEGMENT\\_BASE\\_OP](#).

Here is the call graph for this function:



### 17.83.2.2 fiasco\_amd64\_set\_segment\_base()

```

long fiasco_amd64_set_segment_base (
    l4_cap_idx_t thread,
    enum L4_sys_segment sgr,
    l4_umword_t base,
    l4_utcb_t * utcb) [inline]

```

Set the base address for a segment.

### Parameters

<i>thread</i>	Thread for which the base address of the selected segment shall be modified.
---------------	------------------------------------------------------------------------------

<i>segr</i>	Segment to modify (one of <a href="#">L4_sys_segment</a> ).
<i>base</i>	Base address.
<i>utcb</i>	UTCB to be used for this operation, shall be the UTCB of the calling thread. See <a href="#">l4_utcb</a> .

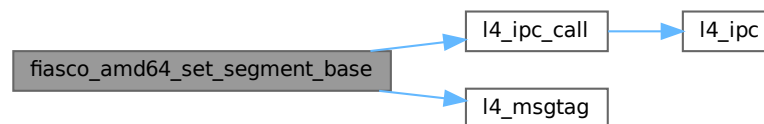
### Return values

<i>L4_EOK</i>	Success.
<i>-L4_EINVAL</i>	Invalid segment ( <i>segr</i> ) or base address ( <i>base</i> ).
<i>-L4_ENOSYS</i>	Operation not supported with current kernel configuration.

Definition at line 32 of file [segment.h](#).

References [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_PROTO\\_THREAD](#), and [L4\\_THREAD\\_AMD64\\_SET\\_SEGMENT\\_BASE](#).

Here is the call graph for this function:



## 17.84 segment.h

[Go to the documentation of this file.](#)

```

00001 #include_next <l4/sys/segment.h>
00002
00008 /*
00009  * (c) 2011 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #ifndef __L4_SYS__ARCH_AMD64__L4API_L4F__SEGMENT_H__
00015 #define __L4_SYS__ARCH_AMD64__L4API_L4F__SEGMENT_H__
00016
00017 #include <l4/sys/compiler.h>
00018
00019 /***** Implementation *****/
00020
00021
00022
00023 L4_INLINE long
00024 fiasco_amd64_set_fs(l4_cap_idx_t thread, l4_umword_t base, l4_utcb_t *utcb)
00025 {
00026     l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_AMD64_SET_SEGMENT_BASE_OP | ((l4_umword_t)L4_AMD64_SEGMENT_FS
00027     << 16);
00027     l4_utcb_mr_u(utcb)->mr[1] = base;
00028     return l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0), L4_IPC_NEVER),
00029     utcb);
00029 }
00030
00031 L4_INLINE long
00032 fiasco_amd64_set_segment_base(l4_cap_idx_t thread, enum L4_sys_segment segr,
00033     l4_umword_t base, l4_utcb_t *utcb)

```



```

00034 {
00035     l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_AMD64_SET_SEGMENT_BASE_OP | ((l4_umword_t)segr << 16);
00036     l4_utcb_mr_u(utcb)->mr[1] = base;
00037     return l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0), L4_IPC_NEVER),
00038         utcb);
00039 }
00040 L4_INLINE long
00041 fiasco_gdt_set(l4_cap_idx_t thread, void *desc, unsigned int size,
00042     unsigned int entry_number_start, l4_utcb_t *utcb)
00043 {
00044     l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_X86_GDT_OP;
00045     l4_utcb_mr_u(utcb)->mr[1] = entry_number_start;
00046     __builtin_memcpy(&l4_utcb_mr_u(utcb)->mr[2], desc, size);
00047     return l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2 + (size / 8), 0, 0),
00048         L4_IPC_NEVER), utcb);
00049 }
00050 #endif /* ! __L4_SYS_ARCH_X86__L4API_L4F__SEGMENT_H__ */

```

## 17.85 x86/l4/sys/segment.h File Reference

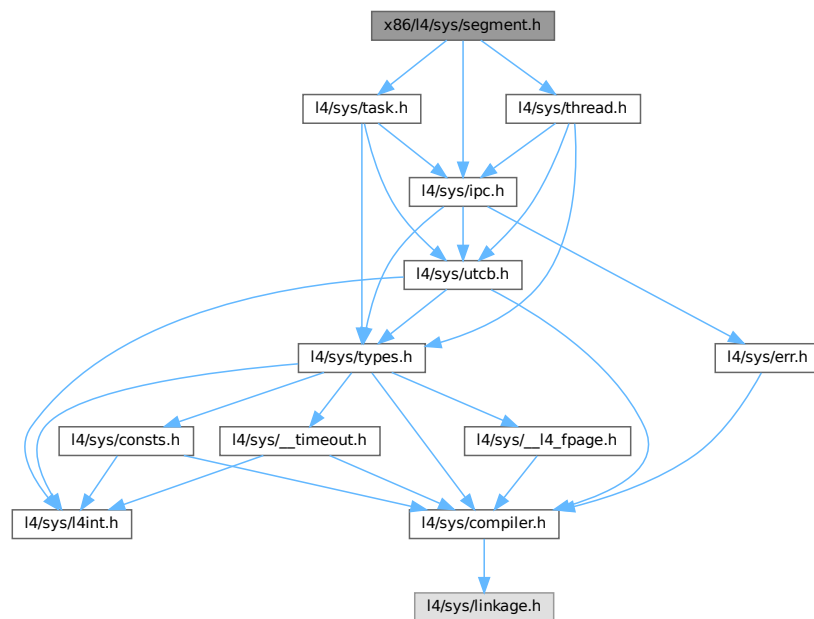
Segment handling (x86).

```

#include <l4/sys/ipc.h>
#include <l4/sys/task.h>
#include <l4/sys/thread.h>

```

Include dependency graph for segment.h:



### Enumerations

- enum [L4\\_task\\_ldt\\_x86\\_consts](#) { [L4\\_TASK\\_LDT\\_X86\\_ENTRY\\_SIZE](#) = 8, [L4\\_TASK\\_LDT\\_X86\\_MAX\\_ENTRIES](#) }

*Contants for LDT handling.*

## Functions

- long [fiasco\\_ldt\\_set](#) ([l4\\_cap\\_idx\\_t](#) task, void \*ldt, unsigned int num\_desc, unsigned int entry\_number\_start, [l4\\_utcb\\_t](#) \*utcb)  
*Set LDT segments descriptors.*
- long [fiasco\\_gdt\\_set](#) ([l4\\_cap\\_idx\\_t](#) thread, void \*desc, unsigned int size, unsigned int entry\_number\_start, [l4\\_utcb\\_t](#) \*utcb)  
*Set GDT segment descriptors.*
- unsigned [fiasco\\_gdt\\_get\\_entry\\_offset](#) ([l4\\_cap\\_idx\\_t](#) thread, [l4\\_utcb\\_t](#) \*utcb)  
*Return the offset of the entry in the GDT.*

## 17.85.1 Detailed Description

Segment handling (x86).

Definition in file [segment.h](#).

## 17.85.2 Enumeration Type Documentation

### 17.85.2.1 L4\_task\_ldt\_x86\_consts

enum [L4\\_task\\_ldt\\_x86\\_consts](#)

Constants for LDT handling.

#### Enumerator

L4_TASK_LDT_X86_ENTRY_SIZE	Size of an LDT entry.
L4_TASK_LDT_X86_MAX_ENTRIES	Maximum number of LDT entries that can be written with one call.

Definition at line 75 of file [segment.h](#).

## 17.86 segment.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 /*****
00013 #ifndef __L4_SYS__ARCH_X86__SEGMENT_H__
00014 #define __L4_SYS__ARCH_X86__SEGMENT_H__
00015
00016 #ifndef L4API_L4f
00017 #error This header file can only be used with a L4API version!
00018 #endif
00019
00020 #include <l4/sys/ipc.h>
00021
00039 L4_INLINE long
00040 fiasco_ldt_set(l4_cap_idx_t task, void *ldt, unsigned int num_desc,
00041               unsigned int entry_number_start, l4_utcb_t *utcb);

```

```

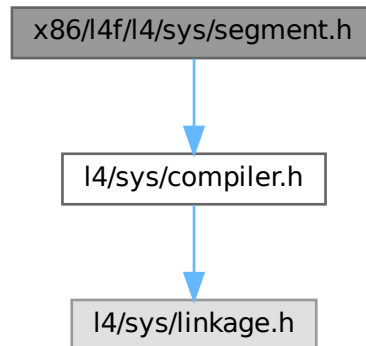
00042
00059 L4_INLINE long
00060 fiasco_gdt_set(l4_cap_idx_t thread, void *desc, unsigned int size,
00061               unsigned int entry_number_start, l4_utcb_t *utcb);
00062
00069 L4_INLINE unsigned
00070 fiasco_gdt_get_entry_offset(l4_cap_idx_t thread, l4_utcb_t *utcb);
00071
00075 enum L4_task_ldt_x86_consts
00076 {
00078     L4_TASK_LDT_X86_ENTRY_SIZE = 8,
00080     L4_TASK_LDT_X86_MAX_ENTRIES
00081     = (L4_UTCB_GENERIC_DATA_SIZE - 2)
00082       / (L4_TASK_LDT_X86_ENTRY_SIZE / (L4_MWORD_BITS / 8)),
00083 };
00084
00085 /*****
00086  *** Implementation
00087  *****/
00088
00089 #include <l4/sys/task.h>
00090 #include <l4/sys/thread.h>
00091
00092 L4_INLINE long
00093 fiasco_ldt_set(l4_cap_idx_t task, void *ldt, unsigned int num_desc,
00094               unsigned int entry_number_start, l4_utcb_t *utcb)
00095 {
00096     if (num_desc > L4_TASK_LDT_X86_MAX_ENTRIES)
00097         return -L4_EINVAL;
00098     l4_utcb_mr_u(utcb)->mr[0] = L4_TASK_LDT_SET_X86_OP;
00099     l4_utcb_mr_u(utcb)->mr[1] = entry_number_start;
00100     __builtin_memcpy(&l4_utcb_mr_u(utcb)->mr[2], ldt,
00101                     num_desc * L4_TASK_LDT_X86_ENTRY_SIZE);
00102     return l4_error_u(l4_ipc_call(task, utcb, l4_msgtag(L4_PROTO_TASK, 2 + num_desc * 2, 0, 0),
00103                     L4_IPC_NEVER), utcb);
00103 }
00104
00105 L4_INLINE unsigned
00106 fiasco_gdt_get_entry_offset(l4_cap_idx_t thread, l4_utcb_t *utcb)
00107 {
00108     l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_X86_GDT_OP;
00109     if (l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER), utcb))
00110         return -1;
00111     return l4_utcb_mr_u(utcb)->mr[0];
00112 }
00113
00114 #endif /* ! __L4_SYS_ARCH_X86_SEGMENT_H */

```

## 17.87 x86/i4f/i4/sys/segment.h File Reference

i4f-specific segment manipulation (x86).

```
#include <l4/sys/compiler.h>
Include dependency graph for segment.h:
```



## Functions

- long [fiasco\\_gdt\\_set](#) ([l4\\_cap\\_idx\\_t](#) thread, void \*desc, unsigned int size, unsigned int entry\_number\_start, [l4\\_utcb\\_t](#) \*utcb)  
Set GDT segment descriptors.

### 17.87.1 Detailed Description

l4f-specific segment manipulation (x86).

Definition in file [segment.h](#).

## 17.88 segment.h

[Go to the documentation of this file.](#)

```

00001 #include_next <l4/sys/segment.h>
00002
00008 /*
00009  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #ifndef __L4_SYS__ARCH_X86__L4API_L4F__SEGMENT_H__
00015 #define __L4_SYS__ARCH_X86__L4API_L4F__SEGMENT_H__
00016
00017 #include <l4/sys/compiler.h>
00018
00019 /***** Implementation *****/
00020
00021
00022
00023 L4_INLINE long
00024 fiasco_gdt_set(l4_cap_idx_t thread, void *desc, unsigned int size,
00025               unsigned int entry_number_start, l4_utcb_t *utcb)
00026 {
00027     l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_X86_GDT_OP;
00028     l4_utcb_mr_u(utcb)->mr[1] = entry_number_start;
00029     __builtin_memcpy(&l4_utcb_mr_u(utcb)->mr[2], desc, size);
00030     return l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2 + (size >> 2), 0, 0),
00031                      L4_IPC_NEVER), utcb);
00031 }
00032
00033 #endif /* ! __L4_SYS__ARCH_X86__L4API_L4F__SEGMENT_H__ */

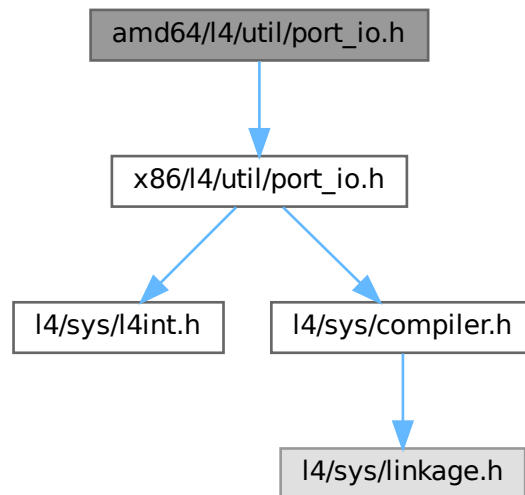
```

## 17.89 amd64/l4/util/port\_io.h File Reference

Port I/O functions.

```
#include <x86/l4/util/port_io.h>
```

Include dependency graph for port\_io.h:



### 17.89.1 Detailed Description

Port I/O functions.

Definition in file [port\\_io.h](#).

## 17.90 port\_io.h

[Go to the documentation of this file.](#)

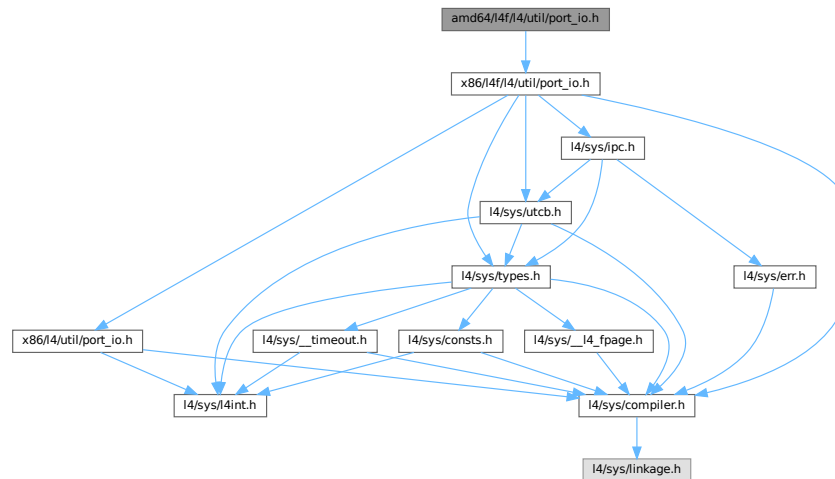
```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010 #include <x86/l4/util/port_io.h>
```

## 17.91 amd64/l4f/l4/util/port\_io.h File Reference

Port I/O functions.

```
#include <x86/l4f/l4/util/port_io.h>
```

Include dependency graph for port\_io.h:



### 17.91.1 Detailed Description

Port I/O functions.

Definition in file [port\\_io.h](#).

## 17.92 port\_io.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  * economic rights: Technische Universität Dresden (Germany)
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010 #include <x86/l4f/l4/util/port_io.h>

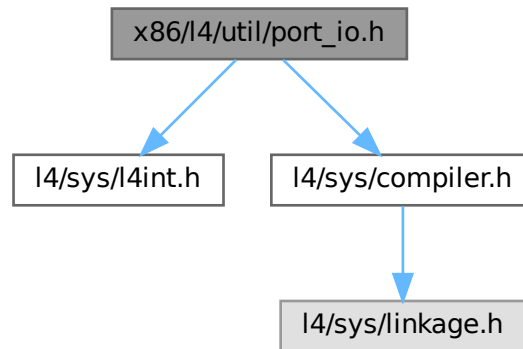
```

## 17.93 x86/l4/util/port\_io.h File Reference

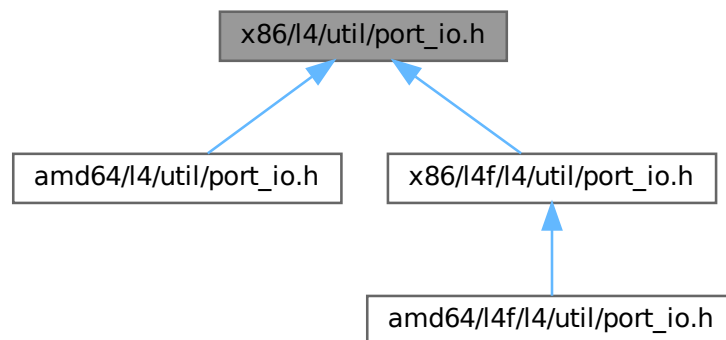
x86 port I/O

```
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>
```

Include dependency graph for port\_io.h:



This graph shows which files directly or indirectly include this file:



## Functions

- `l4_uint8_t l4util_in8 (l4_uint16_t port)`  
*Read byte from I/O port.*
- `l4_uint16_t l4util_in16 (l4_uint16_t port)`  
*Read 16-bit-value from I/O port.*
- `l4_uint32_t l4util_in32 (l4_uint16_t port)`  
*Read 32-bit-value from I/O port.*
- `void l4util_ins8 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)`  
*Read a block of 8-bit-values from I/O ports.*

- void `l4util_ins16` (`l4_uint16_t` port, `l4_umword_t` addr, `l4_umword_t` count)  
*Read a block of 16-bit-values from I/O ports.*
- void `l4util_ins32` (`l4_uint16_t` port, `l4_umword_t` addr, `l4_umword_t` count)  
*Read a block of 32-bit-values from I/O ports.*
- void `l4util_out8` (`l4_uint8_t` value, `l4_uint16_t` port)  
*Write byte to I/O port.*
- void `l4util_out16` (`l4_uint16_t` value, `l4_uint16_t` port)  
*Write 16-bit-value to I/O port.*
- void `l4util_out32` (`l4_uint32_t` value, `l4_uint16_t` port)  
*Write 32-bit-value to I/O port.*
- void `l4util_outs8` (`l4_uint16_t` port, `l4_umword_t` addr, `l4_umword_t` count)  
*Write a block of bytes to I/O port.*
- void `l4util_outs16` (`l4_uint16_t` port, `l4_umword_t` addr, `l4_umword_t` count)  
*Write a block of 16-bit-values to I/O port.*
- void `l4util_outs32` (`l4_uint16_t` port, `l4_umword_t` addr, `l4_umword_t` count)  
*Write block of 32-bit-values to I/O port.*
- void `l4util_iodelay` (void)  
*delay I/O port access by writing to port 0x80*

### 17.93.1 Detailed Description

x86 port I/O

Date

06/2003

Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [port\\_io.h](#).

## 17.94 port\_io.h

[Go to the documentation of this file.](#)

```

00001 /*****
00009 /*****
00010
00011 */
00012 * (c) 2003-2009 Author(s)
00013 *      economic rights: Technische Universität Dresden (Germany)
00014 * License: see LICENSE.spdx (in this directory or the directories above)
00015 */
00016
00017 #ifndef _L4UTIL_PORT_IO_H
00018 #define _L4UTIL_PORT_IO_H
00019
00024
00025 /* L4 includes */
00026 #include <l4/sys/l4int.h>
00027 #include <l4/sys/compiler.h>
00028
00029 /*****
00030 *** Prototypes
00031 *****/

```



```

00032
00033 L4_BEGIN_DECLS
00044 L4_INLINE l4_uint8_t
00045 l4util_in8(l4_uint16_t port);
00046
00053 L4_INLINE l4_uint16_t
00054 l4util_in16(l4_uint16_t port);
00055
00062 L4_INLINE l4_uint32_t
00063 l4util_in32(l4_uint16_t port);
00064
00072 L4_INLINE void
00073 l4util_ins8(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00074
00082 L4_INLINE void
00083 l4util_ins16(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00084
00092 L4_INLINE void
00093 l4util_ins32(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00094
00101 L4_INLINE void
00102 l4util_out8(l4_uint8_t value, l4_uint16_t port);
00103
00110 L4_INLINE void
00111 l4util_out16(l4_uint16_t value, l4_uint16_t port);
00112
00119 L4_INLINE void
00120 l4util_out32(l4_uint32_t value, l4_uint16_t port);
00121
00129 L4_INLINE void
00130 l4util_outs8(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00131
00139 L4_INLINE void
00140 l4util_outs16(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00141
00149 L4_INLINE void
00150 l4util_outs32(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00151
00155 L4_INLINE void
00156 l4util_iodelay(void);
00157
00159
00160 L4_END_DECLS
00161
00162
00163 /*****
00164  *** Implementation
00165  *****/
00166
00167 L4_INLINE l4_uint8_t
00168 l4util_in8(l4_uint16_t port)
00169 {
00170     l4_uint8_t value;
00171     asm volatile ("inb %w1, %b0" : "=a" (value) : "Nd" (port));
00172     return value;
00173 }
00174
00175 L4_INLINE l4_uint16_t
00176 l4util_in16(l4_uint16_t port)
00177 {
00178     l4_uint16_t value;
00179     asm volatile ("inw %w1, %w0" : "=a" (value) : "Nd" (port));
00180     return value;
00181 }
00182
00183 L4_INLINE l4_uint32_t
00184 l4util_in32(l4_uint16_t port)
00185 {
00186     l4_uint32_t value;
00187     asm volatile ("inl %w1, %0" : "=a" (value) : "Nd" (port));
00188     return value;
00189 }
00190
00191 L4_INLINE void
00192 l4util_ins8(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
00193 {
00194     l4_umword_t dummy1, dummy2;
00195     asm volatile ("rep insb" : "=D" (dummy1), "=c" (dummy2)
00196                  : "d" (port), "D" (addr), "c" (count)
00197                  : "memory");
00198 }
00199
00200 L4_INLINE void
00201 l4util_ins16(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
00202 {
00203     l4_umword_t dummy1, dummy2;
00204     asm volatile ("rep insw" : "=D" (dummy1), "=c" (dummy2)

```

```

00205         : "d" (port), "D" (addr), "c"(count)
00206         : "memory");
00207 }
00208
00209 L4_INLINE void
00210 l4util_ins32(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
00211 {
00212     l4_umword_t dummy1, dummy2;
00213     asm volatile ("rep insl" : "=D"(dummy1), "=c"(dummy2)
00214         : "d" (port), "D" (addr), "c"(count)
00215         : "memory");
00216 }
00217
00218 L4_INLINE void
00219 l4util_out8(l4_uint8_t value, l4_uint16_t port)
00220 {
00221     asm volatile ("outb %b0, %w1" : : "a" (value), "Nd" (port));
00222 }
00223
00224 L4_INLINE void
00225 l4util_out16(l4_uint16_t value, l4_uint16_t port)
00226 {
00227     asm volatile ("outw %w0, %w1" : : "a" (value), "Nd" (port));
00228 }
00229
00230 L4_INLINE void
00231 l4util_out32(l4_uint32_t value, l4_uint16_t port)
00232 {
00233     asm volatile ("outl %0, %w1" : : "a" (value), "Nd" (port));
00234 }
00235
00236 L4_INLINE void
00237 l4util_outs8(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
00238 {
00239     l4_umword_t dummy1, dummy2;
00240     asm volatile ("rep outsb" : "=S"(dummy1), "=c"(dummy2)
00241         : "d" (port), "S" (addr), "c"(count)
00242         : "memory");
00243 }
00244
00245 L4_INLINE void
00246 l4util_outs16(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
00247 {
00248     l4_umword_t dummy1, dummy2;
00249     asm volatile ("rep outsw" : "=S"(dummy1), "=c"(dummy2)
00250         : "d" (port), "S" (addr), "c"(count)
00251         : "memory");
00252 }
00253
00254 L4_INLINE void
00255 l4util_outs32(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
00256 {
00257     l4_umword_t dummy1, dummy2;
00258     asm volatile ("rep outsl" : "=S"(dummy1), "=c"(dummy2)
00259         : "d" (port), "S" (addr), "c"(count)
00260         : "memory");
00261 }
00262
00263 L4_INLINE void
00264 l4util_iodelay(void)
00265 {
00266     asm volatile ("outb %al, $0x80");
00267 }
00268
00269 #endif

```

## 17.95 x86/I4f/I4/util/port\_io.h File Reference

port I/O functions

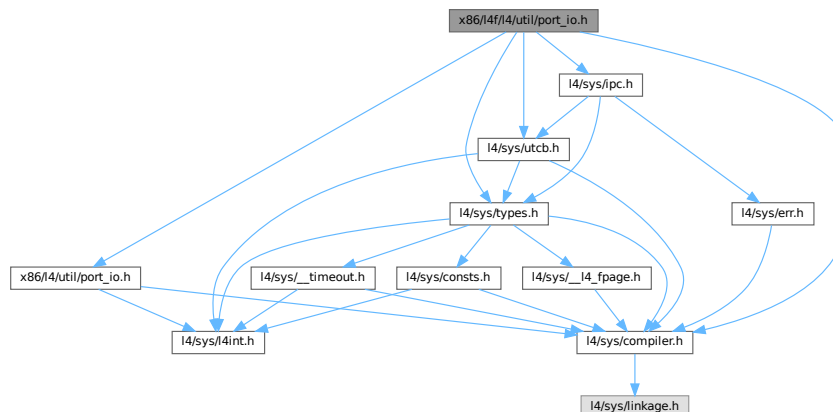
```

#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <x86/l4/util/port_io.h>
#include <l4/sys/utcb.h>

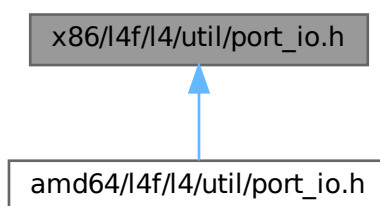
```

```
#include <l4/sys/ipc.h>
```

Include dependency graph for port\_io.h:



This graph shows which files directly or indirectly include this file:



## Functions

- [L4\\_BEGIN\\_DECLS](#) `int l4util_ioport_map (l4_cap_idx_t sigma0id, unsigned port_start, unsigned log2size)`  
Map a range of I/O ports.

## 17.95.1 Detailed Description

port I/O functions

Date

06/2003

Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [port\\_io.h](#).

## 17.96 port\_io.h

[Go to the documentation of this file.](#)

```

00001 /*****
00009 /*****
00010
00011 */
00012 * (c) 2003-2009 Author(s)
00013 *     economic rights: Technische Universität Dresden (Germany)
00014 * License: see LICENSE.spdx (in this directory or the directories above)
00015 */
00016
00017 #ifndef _L4UTIL_PORT_IO_API_H
00018 #define _L4UTIL_PORT_IO_API_H
00019
00020 #include <l4/sys/compiler.h>
00021 #include <l4/sys/types.h>
00022
00023 #include <x86/l4/util/port_io.h>
00024
00025 L4_BEGIN_DECLS
00026
00038 L4_INLINE int
00039 l4util_ioport_map(l4_cap_idx_t sigma0id,
00040                  unsigned port_start, unsigned log2size);
00041
00042 L4_END_DECLS
00043
00044
00045 /*****
00046 *** Implementation
00047 *****/
00048
00049 #include <l4/sys/utcb.h>
00050 #include <l4/sys/ipc.h>
00051
00052
00053 L4_INLINE int
00054 l4util_ioport_map(l4_cap_idx_t sigma0id,
00055                  unsigned port_start, unsigned log2size)
00056 {
00057     l4_fpage_t iofp;
00058     l4_msgtag_t tag;
00059     long err;
00060
00061     iofp = l4_iofpage(port_start, log2size);
00062     l4_utcb_mr()->mr[0] = iofp.raw;
00063     l4_utcb_br()->bdr    = 0;
00064     l4_utcb_br()->br[0] = L4_ITEM_MAP;
00065     l4_utcb_br()->br[1] = iofp.raw;
00066     tag = l4_ipc_call(sigma0id, l4_utcb(),
00067                      l4_msgtag(L4_PROTO_IO_PAGE_FAULT, 1, 0, 0),
00068                      L4_IPC_NEVER);
00069
00070     if ((err = l4_ipc_error(tag, l4_utcb()))
00071         return err;
00072
00073     return l4_msgtag_items(tag) > 0 ? 0 : -L4_ENOENT;
00074 }
00075
00076 #endif
00077

```

## 17.97 \_\_kip-arch.h

```

00001 /*
00002 * (c) 2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003 *     economic rights: Technische Universität Dresden (Germany)
00004 *
00005 * License: see LICENSE.spdx (in this directory or the directories above)
00006 */
00007 #pragma once
00008
00009 struct l4_kip_platform_info_arch
00010 {};

```

## 17.98 \_\_kip-arch.h

```

00001 /*
00002  * (c) 2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00014 struct l4_kip_platform_info_arch
00015 {
00016     struct
00017     {
00018         l4_uint32_t MIDR, CTR, TCMTR, TLBTR, MPIDR, REVIDR;
00019         l4_uint32_t ID_PFR[2], ID_DFR0, ID_AFR0, ID_MMFR[4], ID_ISAR[6];
00020     } cpuinfo;
00021 };

```

## 17.99 \_\_kip-arch.h

```

00001 /*
00002  * (c) 2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00014 struct l4_kip_platform_info_arch
00015 {
00016     struct
00017     {
00018         l4_uint64_t MIDR, MPIDR, REVIDR;
00019         l4_uint64_t ID_PFR[3], ID_DFR0, ID_AFR0, ID_MMFR[4], ID_ISAR[7], ID_MVFR[3];
00020         l4_uint64_t ID_AA64DFR[2], ID_AA64ISAR[3], ID_AA64MMFR[3], ID_AA64PFR[2];
00021     } cpuinfo;
00022 };

```

## 17.100 \_\_kip-arch.h

```

00001 /*
00002  * (c) 2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 struct l4_kip_platform_info_arch
00010 {};

```

## 17.101 amd64/l4/sys/\_\_vcpu-arch.h File Reference

AMD64-specific vCPU interface.



### 17.101.1 Detailed Description

AMD64-specific vCPU interface.

Definition in file [\\_\\_vcpu-arch.h](#).

### 17.101.2 Enumeration Type Documentation

#### 17.101.2.1 anonymous enum

anonymous enum

#### Enumerator

L4_VCPU_STATE_VERSION	Architecture-specific version ID. This ID must match the version field in the <a href="#">l4_vcpu_state_t</a> structure after enabling vCPU mode or extended vCPU mode for a thread.
-----------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Definition at line 16 of file [\\_\\_vcpu-arch.h](#).

## 17.102 \_\_vcpu-arch.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00012 #pragma once
00013
00014 #include <l4/sys/types.h>
00015
00016 enum
00017 {
00024     L4_VCPU_STATE_VERSION = 0x26,
00025
00026     L4_VCPU_STATE_SIZE = 0x200,
00027     L4_VCPU_STATE_EXT_SIZE = L4_PAGESIZE,
00028 };
00029
00034 enum L4_vcpu_state_offset
00035 {
00036     L4_VCPU_OFFSET_EXT_STATE = 0x400,
00037     L4_VCPU_OFFSET_EXT_INFOS = 0x200,
00038 };
00039
00043 typedef struct l4_vcpu_arch_state_t
00044 {
00045     l4_umword_t host_fs_base;
00046     l4_umword_t host_gs_base;
00047     l4_uint16_t host_ds, host_es, host_fs, host_gs;
00048
00049     l4_uint16_t const user_ds32;
00050     l4_uint16_t const user_cs64;
00051     l4_uint16_t const user_cs32;
00052 } l4_vcpu_arch_state_t;
00053
00054
00059 typedef struct l4_vcpu_regs_t
00060 {
00061     l4_umword_t r15;
00062     l4_umword_t r14;
00063     l4_umword_t r13;

```

```

00064  l4_umword_t r12;
00065  l4_umword_t r11;
00066  l4_umword_t r10;
00067  l4_umword_t r9;
00068  l4_umword_t r8;
00069
00070  l4_umword_t di;
00071  l4_umword_t si;
00072  l4_umword_t bp;
00073  l4_umword_t pfa;
00074  l4_umword_t bx;
00075  l4_umword_t dx;
00076  l4_umword_t cx;
00077  l4_umword_t ax;
00078
00079  l4_umword_t trapno;
00080  l4_umword_t err;
00081
00082  l4_umword_t ip;
00083  l4_umword_t cs;
00084  l4_umword_t flags;
00085  l4_umword_t sp;
00086  l4_umword_t ss;
00087  l4_umword_t fs_base;
00088  l4_umword_t gs_base;
00089  l4_uint16_t ds, es, fs, gs;
00090
00091 } l4_vcpu_regs_t;
00092
00097 typedef struct l4_vcpu_ipc_regs_t
00098 {
00099     l4_umword_t _res[1];
00100     l4_umword_t label;
00101     l4_umword_t _res2[5];
00102     l4_msgtag_t tag;
00103 } l4_vcpu_ipc_regs_t;

```

### 17.103 arm/l4/sys/\_\_vcpu-arch.h File Reference

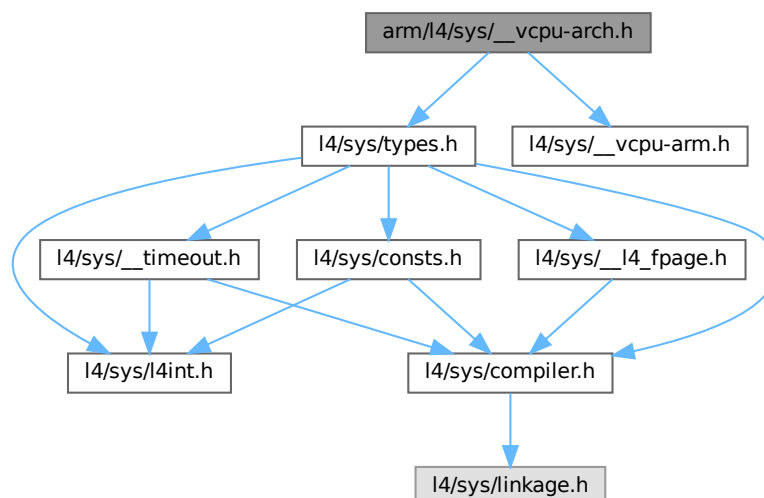
ARM-specific vCPU interface.

```

#include <l4/sys/types.h>
#include <l4/sys/__vcpu-arm.h>

```

Include dependency graph for \_\_vcpu-arch.h:





## Data Structures

- struct [l4\\_vcpu\\_regs\\_t](#)  
*vCPU registers.*
- struct [l4\\_vcpu\\_arch\\_state\\_t](#)  
*Architecture-specific vCPU state.*
- struct [l4\\_vcpu\\_ipc\\_regs\\_t](#)  
*vCPU message registers.*

## Typedefs

- typedef struct [l4\\_vcpu\\_regs\\_t](#) **[l4\\_vcpu\\_regs\\_t](#)**  
*vCPU registers.*
- typedef struct [l4\\_vcpu\\_arch\\_state\\_t](#) **[l4\\_vcpu\\_arch\\_state\\_t](#)**  
*Architecture-specific vCPU state.*
- typedef struct [l4\\_vcpu\\_ipc\\_regs\\_t](#) **[l4\\_vcpu\\_ipc\\_regs\\_t](#)**  
*vCPU message registers.*

## Enumerations

- enum { [L4\\_VCPU\\_STATE\\_VERSION](#) = 0x38 , [L4\\_VCPU\\_STATE\\_SIZE](#) = 0x100 , [L4\\_VCPU\\_STATE\\_EXT\\_SIZE](#) = 0x400 }
  - enum [L4\\_vcpu\\_state\\_offset](#) { [L4\\_VCPU\\_OFFSET\\_EXT\\_STATE](#) = 0x180 , [L4\\_VCPU\\_OFFSET\\_EXT\\_INFOS](#) = 0x100 }
  - enum [L4\\_vcpu\\_e\\_field\\_ids](#) { }
- Offsets for vCPU state layouts.*
- IDs for extended vCPU state fields.*

### 17.103.1 Detailed Description

ARM-specific vCPU interface.

Definition in file [\\_\\_vcpu-arch.h](#).

### 17.103.2 Enumeration Type Documentation

#### 17.103.2.1 anonymous enum

anonymous enum

#### Enumerator

<a href="#">L4_VCPU_STATE_VERSION</a>	Architecture-specific version ID. This ID must match the version field in the <a href="#">l4_vcpu_state_t</a> structure after enabling vCPU mode or extended vCPU mode for a thread.
---------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Definition at line 17 of file [\\_\\_vcpu-arch.h](#).

### 17.103.2.2 L4\_vcpu\_e\_field\_ids

enum `L4_vcpu_e_field_ids`

IDs for extended vCPU state fields.

Bits 14..15: are the field size:

- 0 = 32bit field
- 1 = register width field
- 2 = 64bit field

#### Enumerator

L4_VCPU_E_VTMR_CFG	vtimer irq configuration
--------------------	--------------------------

Definition at line 99 of file `__vcpu-arch.h`.

## 17.104 \_\_vcpu-arch.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00012 #pragma once
00013
00014 #include <l4/sys/types.h>
00015 #include <l4/sys/__vcpu-arm.h>
00016
00017 enum
00018 {
00025     L4_VCPU_STATE_VERSION = 0x38,
00026
00027     L4_VCPU_STATE_SIZE = 0x100,
00028     L4_VCPU_STATE_EXT_SIZE = 0x400,
00029 };
00030
00035 enum L4_vcpu_state_offset
00036 {
00037     L4_VCPU_OFFSET_EXT_STATE = 0x180,
00038     L4_VCPU_OFFSET_EXT_INFOS = 0x100,
00039 };
00040
00041 L4_INLINE l4_arm_vcpu_e_info_t const *
00042 l4_vcpu_e_info(void const *vcpu) L4_NOTHROW
00043 {
00044     return (l4_arm_vcpu_e_info_t const *)((l4_addr_t)vcpu
00045                                           + L4_VCPU_OFFSET_EXT_INFOS);
00046 }
00047
00048 L4_INLINE void *l4_vcpu_e_ptr(void const *vcpu, unsigned id) L4_NOTHROW
00049 { return (void *)((l4_addr_t)vcpu + L4_VCPU_OFFSET_EXT_STATE + (id & 0xfff)); }
00050
00055 typedef struct l4_vcpu_regs_t
00056 {
00057     l4_umword_t pfa;
00058     l4_umword_t err;
00059
00060     l4_umword_t r[13];
00061
00062     l4_umword_t sp;
00063     l4_umword_t lr;

```

```

00064     l4_umword_t _dummy;
00065     l4_umword_t ip;
00066     l4_umword_t flags;
00067     l4_umword_t tpidruro;
00068     l4_umword_t tpidrurw;
00069 } l4_vcpu_regs_t;
00070
00074 typedef struct l4_vcpu_arch_state_t
00075 {
00076     l4_umword_t host_tpidruro;
00077 } l4_vcpu_arch_state_t;
00078
00083 typedef struct l4_vcpu_ipc_regs_t
00084 {
00085     l4_msgtag_t tag;
00086     l4_umword_t _d1[3];
00087     l4_umword_t label;
00088     l4_umword_t _d2[8];
00089 } l4_vcpu_ipc_regs_t;
00090
00099 enum L4_vcpu_e_field_ids
00100 {
00101     L4_VCPU_E_HCR           = 0x8008,
00102     L4_VCPU_E_TTBRO        = 0x8010,
00103     L4_VCPU_E_TTBRI        = 0x8018,
00104     L4_VCPU_E_TTBRI        = 0x0020,
00105     L4_VCPU_E_SCTLR        = 0x0024,
00106     L4_VCPU_E_DACR         = 0x0028,
00107     L4_VCPU_E_FCSEIDR      = 0x002c,
00108
00109     L4_VCPU_E_CNTVCTL       = 0x0030,
00110     L4_VCPU_E_CNTVOFF      = 0x8038,
00111
00112     L4_VCPU_E_VMPIDR        = 0x0040,
00113     L4_VCPU_E_VPIDR         = 0x0044,
00114
00115     L4_VCPU_E_VTMR_CFG      = 0x0048,
00116
00117     L4_VCPU_E_GIC_HCR        = 0x0050,
00118     L4_VCPU_E_GIC_VTR        = 0x0054,
00119     L4_VCPU_E_GIC_VMCR       = 0x0058,
00120     L4_VCPU_E_GIC_MISR       = 0x005c,
00121     L4_VCPU_E_GIC_EISR       = 0x0060,
00122     L4_VCPU_E_GIC_ELSR       = 0x0064,
00123     L4_VCPU_E_GIC_V2_LR0     = 0x0068,
00124     L4_VCPU_E_GIC_V3_LR0     = 0x8068,
00125 };

```

## 17.105 arm64/l4/sys/\_\_vcpu-arch.h File Reference

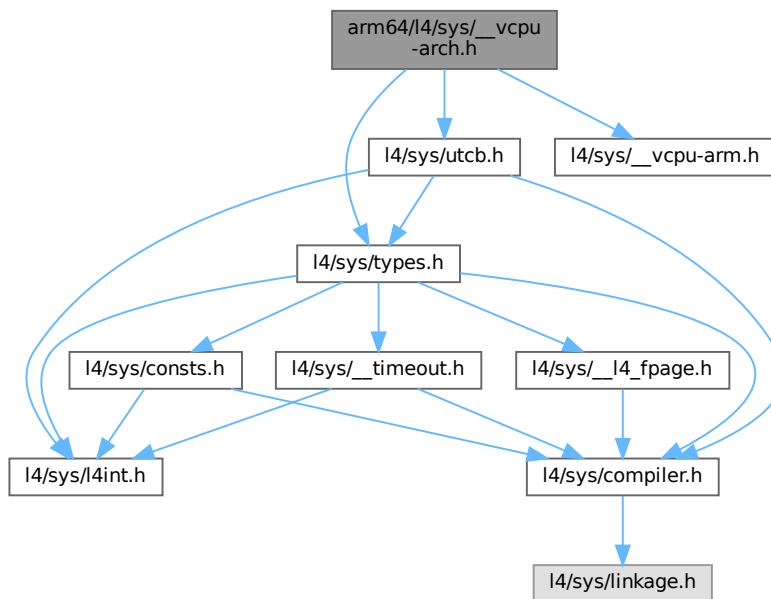
ARM64-specific vCPU interface.

```

#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/__vcpu-arm.h>

```

Include dependency graph for `__vcpu-arch.h`:



## Data Structures

- struct `l4_vcpu_arch_state_t`  
*Architecture-specific vCPU state.*
- struct `l4_vcpu_ipc_regs_t`  
*vCPU message registers.*

## Typedefs

- typedef `l4_exc_regs_t l4_vcpu_regs_t`  
*vCPU registers.*
- typedef struct `l4_vcpu_arch_state_t l4_vcpu_arch_state_t`  
*Architecture-specific vCPU state.*
- typedef struct `l4_vcpu_ipc_regs_t l4_vcpu_ipc_regs_t`  
*vCPU message registers.*

## Enumerations

- enum { `L4_VCPU_STATE_VERSION` = 0x38 , `L4_VCPU_STATE_SIZE` = 0x200 , `L4_VCPU_STATE_EXT_SIZE` = 0x800 }
  - enum `L4_vcpu_state_offset` { `L4_VCPU_OFFSET_EXT_STATE` = 0x280 , `L4_VCPU_OFFSET_EXT_INFOS` = 0x200 }
  - enum `L4_vcpu_e_field_ids` { }
- Offsets for vCPU state layouts.*
- IDs for extended vCPU state fields.*

## 17.105.1 Detailed Description

ARM64-specific vCPU interface.

Definition in file [\\_\\_vcpu-arch.h](#).

## 17.105.2 Enumeration Type Documentation

### 17.105.2.1 anonymous enum

anonymous enum

#### Enumerator

L4_VCPU_STATE_VERSION	Architecture-specific version ID. This ID must match the version field in the <a href="#">l4_vcpu_state_t</a> structure after enabling vCPU mode or extended vCPU mode for a thread.
-----------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Definition at line 18 of file [\\_\\_vcpu-arch.h](#).

### 17.105.2.2 L4\_vcpu\_e\_field\_ids

enum [L4\\_vcpu\\_e\\_field\\_ids](#)

IDs for extended vCPU state fields.

Bits 14..15: are the field size:

- 0 = 32bit field
- 1 = register width field
- 2 = 64bit field

#### Enumerator

L4_VCPU_E_VTMR_CFG	vtimer irq configuration
--------------------	--------------------------

Definition at line 85 of file [\\_\\_vcpu-arch.h](#).

## 17.106 \_\_vcpu-arch.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00012 #pragma once
00013
00014 #include <l4/sys/types.h>
00015 #include <l4/sys/utcb.h>
00016 #include <l4/sys/__vcpu-arm.h>
00017
00018 enum
00019 {
00026     L4_VCPU_STATE_VERSION = 0x38,
00027
00028     L4_VCPU_STATE_SIZE = 0x200,
00029     L4_VCPU_STATE_EXT_SIZE = 0x800,
00030 };
00031
00036 enum L4_vcpu_state_offset
00037 {
00038     L4_VCPU_OFFSET_EXT_STATE = 0x280,
00039     L4_VCPU_OFFSET_EXT_INFOS = 0x200,
00040 };
00041
00042 L4_INLINE l4_arm_vcpu_e_info_t const *
00043 l4_vcpu_e_info(void const *vcpu) L4_NOTHROW
00044 {
00045     return (l4_arm_vcpu_e_info_t const *)((l4_addr_t)vcpu
00046                                           + L4_VCPU_OFFSET_EXT_INFOS);
00047 }
00048
00049 L4_INLINE void *l4_vcpu_e_ptr(void const *vcpu, unsigned id) L4_NOTHROW
00050 { return (void *)((l4_addr_t)vcpu + L4_VCPU_OFFSET_EXT_STATE + (id & 0xfff)); }
00051
00056 typedef l4_exc_regs_t l4_vcpu_regs_t;
00057
00061 typedef struct l4_vcpu_arch_state_t
00062 {
00063     l4_umword_t host_tpidrulo;
00064 } l4_vcpu_arch_state_t;
00065
00070 typedef struct l4_vcpu_ipc_regs_t
00071 {
00072     l4_msgtag_t tag;
00073     l4_umword_t label;
00074     l4_umword_t _d1[3];
00075 } l4_vcpu_ipc_regs_t;
00076
00085 enum L4_vcpu_e_field_ids
00086 {
00087     L4_VCPU_E_HCR          = 0x8008,
00088     L4_VCPU_E_SCTLR       = 0x0010,
00089     L4_VCPU_E_CPACR       = 0x0014,
00090
00091     L4_VCPU_E_CNTVCTL     = 0x0018,
00092     L4_VCPU_E_CNTVOFF     = 0x8020,
00093
00094     L4_VCPU_E_VMPIDR      = 0x8028,
00095     L4_VCPU_E_VPIDR       = 0x0030,
00096
00097     L4_VCPU_E_VTMR_CFG    = 0x0034,
00098     L4_VCPU_E_VTCR        = 0x8038,
00099
00100     L4_VCPU_E_GIC_HCR     = 0x0040,
00101     L4_VCPU_E_GIC_VTR     = 0x0044,
00102     L4_VCPU_E_GIC_VMCR    = 0x0048,
00103     L4_VCPU_E_GIC_MISR    = 0x004c,
00104     L4_VCPU_E_GIC_EISR    = 0x0050,
00105     L4_VCPU_E_GIC_ELSR    = 0x0054,
00106     L4_VCPU_E_GIC_V2_LR0  = 0x0058,
00107     L4_VCPU_E_GIC_V3_LR0  = 0x8058,
00108 };

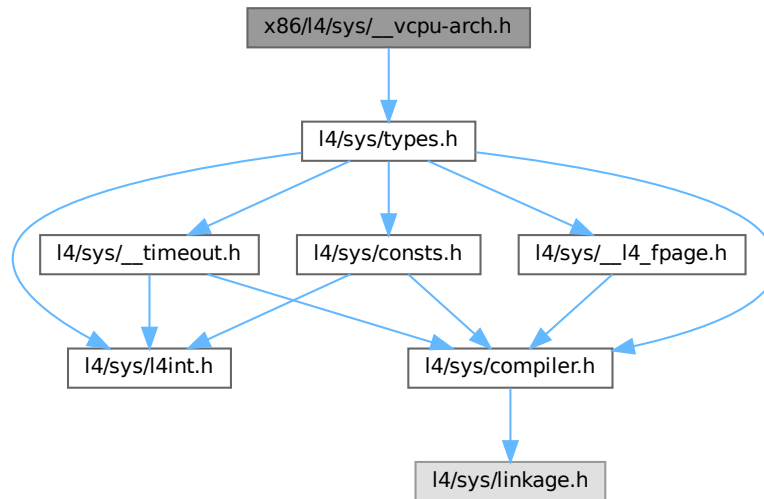
```

## 17.107 x86/l4/sys/\_\_vcpu-arch.h File Reference

x86-specific vCPU interface.

```
#include <l4/sys/types.h>
```

Include dependency graph for \_\_vcpu-arch.h:



### Data Structures

- struct `l4_vcpu_regs_t`  
*vCPU registers.*
- struct `l4_vcpu_arch_state_t`  
*Architecture-specific vCPU state.*
- struct `l4_vcpu_ipc_regs_t`  
*vCPU message registers.*

### Typedefs

- typedef struct `l4_vcpu_regs_t` `l4_vcpu_regs_t`  
*vCPU registers.*
- typedef struct `l4_vcpu_arch_state_t` `l4_vcpu_arch_state_t`  
*Architecture-specific vCPU state.*
- typedef struct `l4_vcpu_ipc_regs_t` `l4_vcpu_ipc_regs_t`  
*vCPU message registers.*

### Enumerations

- enum { `L4_VCPU_STATE_VERSION` = 0x46 , `L4_VCPU_STATE_SIZE` = 0x200 , `L4_VCPU_STATE_EXT_SIZE` = `L4_PAGESIZE` }
  - enum `L4_vcpu_state_offset` { `L4_VCPU_OFFSET_EXT_STATE` = 0x400 , `L4_VCPU_OFFSET_EXT_INFOS` = 0x200 }
- Offsets for vCPU state layouts.*

### 17.107.1 Detailed Description

x86-specific vCPU interface.

Definition in file [\\_\\_vcpu-arch.h](#).

### 17.107.2 Enumeration Type Documentation

#### 17.107.2.1 anonymous enum

anonymous enum

#### Enumerator

L4_VCPU_STATE_VERSION	Architecture-specific version ID. This ID must match the version field in the <a href="#">l4_vcpu_state_t</a> structure after enabling vCPU mode or extended vCPU mode for a thread.
-----------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Definition at line 16 of file [\\_\\_vcpu-arch.h](#).

## 17.108 \_\_vcpu-arch.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00012 #pragma once
00013
00014 #include <l4/sys/types.h>
00015
00016 enum
00017 {
00024     L4_VCPU_STATE_VERSION = 0x46,
00025
00026     L4_VCPU_STATE_SIZE = 0x200,
00027     L4_VCPU_STATE_EXT_SIZE = L4_PAGESIZE,
00028 };
00029
00034 enum L4_vcpu_state_offset
00035 {
00036     L4_VCPU_OFFSET_EXT_STATE = 0x400,
00037     L4_VCPU_OFFSET_EXT_INFOS = 0x200,
00038 };
00039
00044 typedef struct l4_vcpu_regs_t
00045 {
00046     l4_umword_t es;
00047     l4_umword_t ds;
00048     l4_umword_t gs;
00049     l4_umword_t fs;
00050
00051     l4_umword_t di;
00052     l4_umword_t si;
00053     l4_umword_t bp;
00054     l4_umword_t pfa;
00055     l4_umword_t bx;
00056     l4_umword_t dx;
00057     l4_umword_t cx;
00058     l4_umword_t ax;
00059
00060     l4_umword_t trapno;

```



```

00061     l4_umword_t err;
00062
00063     l4_umword_t ip;
00064     l4_umword_t dummy1;
00065     l4_umword_t flags;
00066     l4_umword_t sp;
00067     l4_umword_t ss;
00068 } l4_vcpu_regs_t;
00069
00073 typedef struct l4_vcpu_arch_state_t { } l4_vcpu_arch_state_t;
00074
00079 typedef struct l4_vcpu_ipc_regs_t
00080 {
00081     l4_umword_t _res[2];
00082     l4_umword_t label;
00083     l4_umword_t _res2[3];
00084     l4_msgtag_t tag;
00085 } l4_vcpu_ipc_regs_t;

```

## 17.109 ktrace\_events.h

```

00001 /* Note, automatically generated from Fiasco binary */
00002 #pragma once
00003
00004 enum L4_ktrace_tbuf_entry_fixed
00005 {
00006     l4_ktrace_tbuf_unused = 0,
00007     l4_ktrace_tbuf_pf = 1,
00008     l4_ktrace_tbuf_ipc = 2,
00009     l4_ktrace_tbuf_ipc_res = 3,
00010     l4_ktrace_tbuf_ipc_trace = 4,
00011     l4_ktrace_tbuf_ke = 5,
00012     l4_ktrace_tbuf_ke_reg = 6,
00013     l4_ktrace_tbuf_breakpoint = 7,
00014     l4_ktrace_tbuf_ke_bin = 8,
00015     l4_ktrace_tbuf_dynentries = 9,
00016     l4_ktrace_tbuf_max = 128,
00017     l4_ktrace_tbuf_hidden = 128,
00018 };
00019
00020 typedef unsigned long L4_ktrace_t__Address;
00021 typedef unsigned long L4_ktrace_t__Cap_index;
00022 typedef void L4_ktrace_t__Context;
00023 typedef void L4_ktrace_t__Context__Drq;
00024 typedef unsigned L4_ktrace_t__Context__Drq_log__Type;
00025 typedef unsigned L4_ktrace_t__Cpu_number;
00026 typedef void L4_ktrace_t__Irq_base;
00027 typedef void L4_ktrace_t__Irq_chip;
00028 typedef void L4_ktrace_t__Kobject;
00029 typedef unsigned long L4_ktrace_t__L4_error;
00030 typedef unsigned long L4_ktrace_t__L4_msg_tag;
00031 typedef unsigned long L4_ktrace_t__L4_obj_ref;
00032 typedef unsigned L4_ktrace_t__L4_timeout_pair;
00033 typedef unsigned long L4_ktrace_t__Mword;
00034 typedef void L4_ktrace_t__Rcu_item;
00035 typedef void L4_ktrace_t__Sched_context;
00036 typedef long L4_ktrace_t__Smword;
00037 typedef void L4_ktrace_t__Space;
00038 typedef unsigned short L4_ktrace_t__Unsigned16;
00039 typedef unsigned int L4_ktrace_t__Unsigned32;
00040 typedef unsigned long long L4_ktrace_t__Unsigned64;
00041 typedef unsigned char L4_ktrace_t__Unsigned8;
00042 typedef void L4_ktrace_t__cxx__Type_info;
00043
00044 typedef struct __attribute__((packed))
00045 {
00046     L4_ktrace_t__Mword_number; /* 0+8 */
00047     L4_ktrace_t__Address_ip; /* 8+8 */
00048     L4_ktrace_t__Unsigned64_tsc; /* 16+8 */
00049     L4_ktrace_t__Context *ctx; /* 24+8 */
00050     L4_ktrace_t__Unsigned32_pmc1; /* 32+4 */
00051     L4_ktrace_t__Unsigned32_pmc2; /* 36+4 */
00052     L4_ktrace_t__Unsigned32_kclock; /* 40+4 */
00053     L4_ktrace_t__Unsigned8_type; /* 44+1 */
00054     L4_ktrace_t__Unsigned8_cpu; /* 45+1 */
00055     union __attribute__((packed))
00056     {
00057         struct __attribute__((packed))
00058         {
00059             char __pre_pad[2];
00060             void *func; /* 48+8 */
00061             L4_ktrace_t__Context *thread; /* 56+8 */
00062             L4_ktrace_t__Context__Drq *rq; /* 64+8 */

```

```

00063     L4_ktrace_t__Cpu_number target_cpu; /* 72+4 */
00064     L4_ktrace_t__Context__Drq_log__Type type; /* 76+4 */
00065     char wait; /* 80+1 */
00066 } drq; /* 88 */
00067 struct __attribute__((__packed__))
00068 {
00069     char __pre_pad[2];
00070     L4_ktrace_t__Mword state; /* 48+8 */
00071     L4_ktrace_t__Mword ip; /* 56+8 */
00072     L4_ktrace_t__Mword sp; /* 64+8 */
00073     L4_ktrace_t__Mword space; /* 72+8 */
00074     L4_ktrace_t__Mword err; /* 80+8 */
00075     unsigned char type; /* 88+1 */
00076     unsigned char trap; /* 89+1 */
00077 } vcpu; /* 96 */
00078 struct __attribute__((__packed__))
00079 {
00080     char __pre_pad[2];
00081     L4_ktrace_t__Sword op; /* 48+8 */
00082     L4_ktrace_t__Cap_index buffer; /* 56+8 */
00083     L4_ktrace_t__Mword id; /* 64+8 */
00084     L4_ktrace_t__Mword ram; /* 72+8 */
00085     L4_ktrace_t__Mword newo; /* 80+8 */
00086 } factory; /* 88 */
00087 struct __attribute__((__packed__))
00088 {
00089     char __pre_pad[2];
00090     L4_ktrace_t__Mword gate_dbg_id; /* 48+8 */
00091     L4_ktrace_t__Mword thread_dbg_id; /* 56+8 */
00092     L4_ktrace_t__Mword label; /* 64+8 */
00093 } gate; /* 72 */
00094 struct __attribute__((__packed__))
00095 {
00096     char __pre_pad[2];
00097     L4_ktrace_t__Irq_base *obj; /* 48+8 */
00098     L4_ktrace_t__Irq_chip *chip; /* 56+8 */
00099     L4_ktrace_t__Mword pin; /* 64+8 */
00100 } irq; /* 72 */
00101 struct __attribute__((__packed__))
00102 {
00103     char __pre_pad[2];
00104     L4_ktrace_t__Kobject *obj; /* 48+8 */
00105     L4_ktrace_t__Mword id; /* 56+8 */
00106     L4_ktrace_t__cxx__Type_info *type; /* 64+8 */
00107     L4_ktrace_t__Mword ram; /* 72+8 */
00108 } destroy; /* 80 */
00109 struct __attribute__((__packed__))
00110 {
00111     char __pre_pad[2];
00112     L4_ktrace_t__Cpu_number cpu; /* 48+4 */
00113     char __pad_l[4];
00114     L4_ktrace_t__Rcu_item *item; /* 56+8 */
00115     void *cb; /* 64+8 */
00116     unsigned char event; /* 72+1 */
00117 } rcu; /* 80 */
00118 struct __attribute__((__packed__))
00119 {
00120     char __pre_pad[2];
00121     L4_ktrace_t__Mword id; /* 48+8 */
00122     L4_ktrace_t__Mword mask; /* 56+8 */
00123     L4_ktrace_t__Mword fpage; /* 64+8 */
00124     char map; /* 72+1 */
00125 } tmap; /* 80 */
00126 struct __attribute__((__packed__))
00127 {
00128     char __pre_pad[2];
00129     L4_ktrace_t__Address _address; /* 48+8 */
00130     int _len; /* 56+4 */
00131     char __pad_l[4];
00132     L4_ktrace_t__Mword _value; /* 64+8 */
00133     int _mode; /* 72+4 */
00134 } bp; /* 80 */
00135 struct __attribute__((__packed__))
00136 {
00137     char __pre_pad[2];
00138     L4_ktrace_t__Context *dst; /* 48+8 */
00139     L4_ktrace_t__Context *dst_orig; /* 56+8 */
00140     L4_ktrace_t__Address kernel_ip; /* 64+8 */
00141     L4_ktrace_t__Mword lock_cnt; /* 72+8 */
00142     L4_ktrace_t__Space *from_space; /* 80+8 */
00143     L4_ktrace_t__Sched_context *from_sched; /* 88+8 */
00144     L4_ktrace_t__Mword from_prio; /* 96+8 */
00145 } context_switch; /* 104 */
00146 struct __attribute__((__packed__))
00147 {
00148     } empty; /* 48 */
00149 struct __attribute__((__packed__))

```

```

00150     {
00151         char __pre_pad[2];
00152         L4_ktrace_t__L4_msg_tag_tag; /* 48+8 */
00153         L4_ktrace_t__Mword_dword[2]; /* 56+16 */
00154         L4_ktrace_t__L4_obj_ref_dst; /* 72+8 */
00155         L4_ktrace_t__Mword_dbg_id; /* 80+8 */
00156         L4_ktrace_t__Mword_label; /* 88+8 */
00157         L4_ktrace_t__L4_timeout_pair_timeout; /* 96+4 */
00158         char __pad_1[4];
00159         L4_ktrace_t__Unsigned64_to_abs_rcv; /* 104+8 */
00160     } ipc; /* 112 */
00161     struct __attribute__((__packed__))
00162     {
00163         L4_ktrace_t__Unsigned8_have_snd; /* 46+1 */
00164         L4_ktrace_t__Unsigned8_is_np; /* 47+1 */
00165         L4_ktrace_t__L4_msg_tag_tag; /* 48+8 */
00166         L4_ktrace_t__Mword_dword[2]; /* 56+16 */
00167         L4_ktrace_t__L4_error_result; /* 72+8 */
00168         L4_ktrace_t__Mword_from; /* 80+8 */
00169         L4_ktrace_t__L4_obj_ref_dst; /* 88+8 */
00170         L4_ktrace_t__Mword_pair_event; /* 96+8 */
00171     } ipc_res; /* 104 */
00172     struct __attribute__((__packed__))
00173     {
00174         char __pre_pad[2];
00175         union __attribute__((__packed__)) {
00176             char msg[80]; /* 0+80 */
00177             struct __attribute__((__packed__)) {
00178                 char tag[2]; /* 0+2 */
00179                 char __pad_1[6];
00180                 char *ptr; /* 8+8 */
00181             } mptr; /* 0+16 */
00182         } msg; /* 48+80 */
00183     } ke; /* 128 */
00184     struct __attribute__((__packed__))
00185     {
00186         char _msg[80]; /* 46+80 */
00187     } ke_bin; /* 128 */
00188     struct __attribute__((__packed__))
00189     {
00190         char __pre_pad[2];
00191         L4_ktrace_t__Mword v[3]; /* 48+24 */
00192         union __attribute__((__packed__)) {
00193             char msg[56]; /* 0+56 */
00194             struct __attribute__((__packed__)) {
00195                 char tag[2]; /* 0+2 */
00196                 char __pad_1[6];
00197                 char *ptr; /* 8+8 */
00198             } mptr; /* 0+16 */
00199         } msg; /* 72+56 */
00200     } ke_reg; /* 128 */
00201     struct __attribute__((__packed__))
00202     {
00203         char __pre_pad[2];
00204         L4_ktrace_t__Address_pfa; /* 48+8 */
00205         L4_ktrace_t__Mword_error; /* 56+8 */
00206         L4_ktrace_t__Space *_space; /* 64+8 */
00207     } pf; /* 72 */
00208     struct __attribute__((__packed__))
00209     {
00210         unsigned short mode; /* 46+2 */
00211         L4_ktrace_t__Context *owner; /* 48+8 */
00212         unsigned short id; /* 56+2 */
00213         unsigned short prio; /* 58+2 */
00214         char __pad_1[4];
00215         long left; /* 64+8 */
00216         unsigned long quantum; /* 72+8 */
00217     } sched; /* 80 */
00218     struct __attribute__((__packed__))
00219     {
00220         char _trapno; /* 46+1 */
00221         char __pad_1[1];
00222         L4_ktrace_t__Unsigned16_error; /* 48+2 */
00223         char __pad_2[6];
00224         L4_ktrace_t__Mword_rbp; /* 56+8 */
00225         L4_ktrace_t__Mword_cr2; /* 64+8 */
00226         L4_ktrace_t__Mword_rax; /* 72+8 */
00227         L4_ktrace_t__Mword_rflags; /* 80+8 */
00228         L4_ktrace_t__Mword_rsp; /* 88+8 */
00229         L4_ktrace_t__Unsigned16_cs; /* 96+2 */
00230         L4_ktrace_t__Unsigned16_ds; /* 98+2 */
00231     } trap; /* 104 */
00232     struct __attribute__((__packed__))
00233     {
00234         char _padding[80]; /* 46+80 */
00235         char __post_pad[2]; /* 126+2 */
00236     } fullsize; /* 128 */

```

```

00237     struct __attribute__((__packed__))
00238     {
00239         char __pre_pad[2];
00240         L4_ktrace_t__Cap_index cap_idx; /* 48+8 */
00241     } ieh; /* 56 */
00242     struct __attribute__((__packed__))
00243     {
00244         char __pre_pad[2];
00245         L4_ktrace_t__Mword pfa; /* 48+8 */
00246         L4_ktrace_t__Cap_index cap_idx; /* 56+8 */
00247         L4_ktrace_t__Mword err; /* 64+8 */
00248     } ipfh; /* 72 */
00249     struct __attribute__((__packed__))
00250     {
00251         char __pre_pad[2];
00252         L4_ktrace_t__Mword id; /* 48+8 */
00253         L4_ktrace_t__Mword ip; /* 56+8 */
00254         L4_ktrace_t__Mword sp; /* 64+8 */
00255         L4_ktrace_t__Mword op; /* 72+8 */
00256     } exregs; /* 80 */
00257     struct __attribute__((__packed__))
00258     {
00259         char __pre_pad[2];
00260         L4_ktrace_t__Mword state; /* 48+8 */
00261         L4_ktrace_t__Address user_ip; /* 56+8 */
00262         L4_ktrace_t__Cpu_number src_cpu; /* 64+4 */
00263         L4_ktrace_t__Cpu_number target_cpu; /* 68+4 */
00264     } migration; /* 72 */
00265     struct __attribute__((__packed__))
00266     {
00267         char __pre_pad[2];
00268         L4_ktrace_t__Address user_ip; /* 48+8 */
00269     } timer; /* 56 */
00270     struct __attribute__((__packed__))
00271     {
00272         char __pre_pad[2];
00273         L4_ktrace_t__Mword exitcode; /* 48+8 */
00274         L4_ktrace_t__Mword exitinfo1; /* 56+8 */
00275         L4_ktrace_t__Mword exitinfo2; /* 64+8 */
00276         L4_ktrace_t__Mword rip; /* 72+8 */
00277     } svm; /* 80 */
00278     } m;
00279 } l4_tracebuffer_entry_t;

```

## 17.110 ktrace\_events.h

```

00001 /* Note, automatically generated from Fiasco binary */
00002 #pragma once
00003
00004 enum L4_ktrace_tbuf_entry_fixed
00005 {
00006     l4_ktrace_tbuf_unused = 0,
00007     l4_ktrace_tbuf_pf = 1,
00008     l4_ktrace_tbuf_ipc = 2,
00009     l4_ktrace_tbuf_ipc_res = 3,
00010     l4_ktrace_tbuf_ipc_trace = 4,
00011     l4_ktrace_tbuf_ke = 5,
00012     l4_ktrace_tbuf_ke_reg = 6,
00013     l4_ktrace_tbuf_breakpoint = 7,
00014     l4_ktrace_tbuf_ke_bin = 8,
00015     l4_ktrace_tbuf_dynentries = 9,
00016     l4_ktrace_tbuf_max = 128,
00017     l4_ktrace_tbuf_hidden = 128,
00018 };
00019
00020 typedef unsigned long L4_ktrace_t__Address;
00021 typedef unsigned long L4_ktrace_t__Cap_index;
00022 typedef void L4_ktrace_t__Context;
00023 typedef void L4_ktrace_t__Context__Drq;
00024 typedef unsigned L4_ktrace_t__Context__Drq_log__Type;
00025 typedef unsigned L4_ktrace_t__Cpu_number;
00026 typedef void L4_ktrace_t__Irq_base;
00027 typedef void L4_ktrace_t__Irq_chip;
00028 typedef void L4_ktrace_t__Kobject;
00029 typedef unsigned long L4_ktrace_t__L4_error;
00030 typedef unsigned long L4_ktrace_t__L4_msg_tag;
00031 typedef unsigned long L4_ktrace_t__L4_obj_ref;
00032 typedef unsigned L4_ktrace_t__L4_timeout_pair;
00033 typedef unsigned long L4_ktrace_t__Mword;
00034 typedef void L4_ktrace_t__Rcu_item;
00035 typedef void L4_ktrace_t__Sched_context;
00036 typedef long L4_ktrace_t__Sword;
00037 typedef void L4_ktrace_t__Space;

```

```

00038 typedef unsigned int L4_ktrace_t__Unsigned32;
00039 typedef unsigned long long L4_ktrace_t__Unsigned64;
00040 typedef unsigned char L4_ktrace_t__Unsigned8;
00041 typedef void L4_ktrace_t__cxx_Type_info;
00042
00043 typedef struct __attribute__((packed))
00044 {
00045     L4_ktrace_t__Mword _number; /* 0+4 */
00046     L4_ktrace_t__Address _ip; /* 4+4 */
00047     L4_ktrace_t__Unsigned64 _tsc; /* 8+8 */
00048     L4_ktrace_t__Context *_ctx; /* 16+4 */
00049     L4_ktrace_t__Unsigned32 _pmc1; /* 20+4 */
00050     L4_ktrace_t__Unsigned32 _pmc2; /* 24+4 */
00051     L4_ktrace_t__Unsigned32 _kclock; /* 28+4 */
00052     L4_ktrace_t__Unsigned8 _type; /* 32+1 */
00053     L4_ktrace_t__Unsigned8 _cpu; /* 33+1 */
00054     union __attribute__((__packed__))
00055     {
00056         struct __attribute__((__packed__))
00057         {
00058             char __pre_pad[2];
00059             void *func; /* 36+4 */
00060             L4_ktrace_t__Context *thread; /* 40+4 */
00061             L4_ktrace_t__Context__Drq *rq; /* 44+4 */
00062             L4_ktrace_t__Cpu_number target_cpu; /* 48+4 */
00063             L4_ktrace_t__Context__Drq_log__Type type; /* 52+4 */
00064             char wait; /* 56+1 */
00065         } drq; /* 64 */
00066         struct __attribute__((__packed__))
00067         {
00068             char __pre_pad[2];
00069             L4_ktrace_t__Mword state; /* 36+4 */
00070             L4_ktrace_t__Mword ip; /* 40+4 */
00071             L4_ktrace_t__Mword sp; /* 44+4 */
00072             L4_ktrace_t__Mword space; /* 48+4 */
00073             L4_ktrace_t__Mword err; /* 52+4 */
00074             unsigned char type; /* 56+1 */
00075             unsigned char trap; /* 57+1 */
00076         } vcpu; /* 64 */
00077         struct __attribute__((__packed__))
00078         {
00079             char __pre_pad[2];
00080             L4_ktrace_t__Smword op; /* 36+4 */
00081             L4_ktrace_t__Cap_index buffer; /* 40+4 */
00082             L4_ktrace_t__Mword id; /* 44+4 */
00083             L4_ktrace_t__Mword ram; /* 48+4 */
00084             L4_ktrace_t__Mword newo; /* 52+4 */
00085         } factory; /* 56 */
00086         struct __attribute__((__packed__))
00087         {
00088             char __pre_pad[2];
00089             L4_ktrace_t__Mword gate_dbg_id; /* 36+4 */
00090             L4_ktrace_t__Mword thread_dbg_id; /* 40+4 */
00091             L4_ktrace_t__Mword label; /* 44+4 */
00092         } gate; /* 48 */
00093         struct __attribute__((__packed__))
00094         {
00095             char __pre_pad[2];
00096             L4_ktrace_t__Irq_base *obj; /* 36+4 */
00097             L4_ktrace_t__Irq_chip *chip; /* 40+4 */
00098             L4_ktrace_t__Mword pin; /* 44+4 */
00099         } irq; /* 48 */
00100         struct __attribute__((__packed__))
00101         {
00102             char __pre_pad[2];
00103             L4_ktrace_t__Kobject *obj; /* 36+4 */
00104             L4_ktrace_t__Mword id; /* 40+4 */
00105             L4_ktrace_t__cxx_Type_info *type; /* 44+4 */
00106             L4_ktrace_t__Mword ram; /* 48+4 */
00107         } destroy; /* 56 */
00108         struct __attribute__((__packed__))
00109         {
00110             char __pre_pad[2];
00111             L4_ktrace_t__Cpu_number cpu; /* 36+4 */
00112             L4_ktrace_t__Rcu_item *item; /* 40+4 */
00113             void *cb; /* 44+4 */
00114             unsigned char event; /* 48+1 */
00115         } rcu; /* 56 */
00116         struct __attribute__((__packed__))
00117         {
00118             char __pre_pad[2];
00119             L4_ktrace_t__Mword id; /* 36+4 */
00120             L4_ktrace_t__Mword mask; /* 40+4 */
00121             L4_ktrace_t__Mword fpage; /* 44+4 */
00122             char map; /* 48+1 */
00123         } tmap; /* 56 */
00124         struct __attribute__((__packed__))

```

```

00125     {
00126         char __pre_pad[2];
00127         L4_ktrace_t__Address _address; /* 36+4 */
00128         int _len; /* 40+4 */
00129         L4_ktrace_t__Mword _value; /* 44+4 */
00130         int _mode; /* 48+4 */
00131     } bp; /* 56 */
00132     struct __attribute__((__packed__))
00133     {
00134         char __pre_pad[2];
00135         L4_ktrace_t__Context *dst; /* 36+4 */
00136         L4_ktrace_t__Context *dst_orig; /* 40+4 */
00137         L4_ktrace_t__Address kernel_ip; /* 44+4 */
00138         L4_ktrace_t__Mword lock_cnt; /* 48+4 */
00139         L4_ktrace_t__Space *from_space; /* 52+4 */
00140         L4_ktrace_t__Sched_context *from_sched; /* 56+4 */
00141         L4_ktrace_t__Mword from_prio; /* 60+4 */
00142     } context_switch; /* 64 */
00143     struct __attribute__((__packed__))
00144     {
00145     } empty; /* 40 */
00146     struct __attribute__((__packed__))
00147     {
00148         char __pre_pad[2];
00149         L4_ktrace_t__L4_msg_tag _tag; /* 36+4 */
00150         L4_ktrace_t__Mword _dword[2]; /* 40+8 */
00151         L4_ktrace_t__L4_obj_ref _dst; /* 48+4 */
00152         L4_ktrace_t__Mword _dbg_id; /* 52+4 */
00153         L4_ktrace_t__Mword _label; /* 56+4 */
00154         L4_ktrace_t__L4_timeout_pair _timeout; /* 60+4 */
00155     } ipc; /* 64 */
00156     struct __attribute__((__packed__))
00157     {
00158         L4_ktrace_t__Unsigned8 _have_snd; /* 34+1 */
00159         L4_ktrace_t__Unsigned8 _is_np; /* 35+1 */
00160         L4_ktrace_t__L4_msg_tag _tag; /* 36+4 */
00161         L4_ktrace_t__Mword _dword[2]; /* 40+8 */
00162         L4_ktrace_t__L4_error _result; /* 48+4 */
00163         L4_ktrace_t__Mword _from; /* 52+4 */
00164         L4_ktrace_t__L4_obj_ref _dst; /* 56+4 */
00165         L4_ktrace_t__Mword _pair_event; /* 60+4 */
00166     } ipc_res; /* 64 */
00167     struct __attribute__((__packed__))
00168     {
00169         char __pre_pad[2];
00170         union __attribute__((__packed__)) {
00171             char msg[24]; /* 0+24 */
00172             struct __attribute__((__packed__)) {
00173                 char tag[2]; /* 0+2 */
00174                 char __pad_1[2];
00175                 char *ptr; /* 4+4 */
00176             } mptr; /* 0+8 */
00177             } msg; /* 36+24 */
00178         } ke; /* 64 */
00179     } struct __attribute__((__packed__))
00180     {
00181         char _msg[24]; /* 34+24 */
00182     } ke_bin; /* 64 */
00183     struct __attribute__((__packed__))
00184     {
00185         char __pre_pad[2];
00186         L4_ktrace_t__Mword v[3]; /* 36+12 */
00187         union __attribute__((__packed__)) {
00188             char msg[12]; /* 0+12 */
00189             struct __attribute__((__packed__)) {
00190                 char tag[2]; /* 0+2 */
00191                 char __pad_1[2];
00192                 char *ptr; /* 4+4 */
00193             } mptr; /* 0+8 */
00194             } msg; /* 48+12 */
00195         } ke_reg; /* 64 */
00196     } struct __attribute__((__packed__))
00197     {
00198         char __pre_pad[2];
00199         L4_ktrace_t__Address _pfa; /* 36+4 */
00200         L4_ktrace_t__Mword _error; /* 40+4 */
00201         L4_ktrace_t__Space *_space; /* 44+4 */
00202     } pf; /* 48 */
00203     struct __attribute__((__packed__))
00204     {
00205         unsigned short mode; /* 34+2 */
00206         L4_ktrace_t__Context *owner; /* 36+4 */
00207         unsigned short id; /* 40+2 */
00208         unsigned short prio; /* 42+2 */
00209         long left; /* 44+4 */
00210         unsigned long quantum; /* 48+4 */
00211     } sched; /* 56 */

```

```

00212     struct __attribute__((__packed__))
00213     {
00214         char __pre_pad[2];
00215         L4_ktrace_t__Unsigned32 _error; /* 36+4 */
00216         L4_ktrace_t__Mword _cpsr; /* 40+4 */
00217         L4_ktrace_t__Mword _sp; /* 44+4 */
00218     } trap; /* 48 */
00219     struct __attribute__((__packed__))
00220     {
00221         char _padding[24]; /* 34+24 */
00222         char __post_pad[6]; /* 58+6 */
00223     } fullsize; /* 64 */
00224     struct __attribute__((__packed__))
00225     {
00226         char __pre_pad[2];
00227         L4_ktrace_t__Cap_index cap_idx; /* 36+4 */
00228     } ieh; /* 40 */
00229     struct __attribute__((__packed__))
00230     {
00231         char __pre_pad[2];
00232         L4_ktrace_t__Mword pfa; /* 36+4 */
00233         L4_ktrace_t__Cap_index cap_idx; /* 40+4 */
00234         L4_ktrace_t__Mword err; /* 44+4 */
00235     } ipfh; /* 48 */
00236     struct __attribute__((__packed__))
00237     {
00238         char __pre_pad[2];
00239         L4_ktrace_t__Mword id; /* 36+4 */
00240         L4_ktrace_t__Mword ip; /* 40+4 */
00241         L4_ktrace_t__Mword sp; /* 44+4 */
00242         L4_ktrace_t__Mword op; /* 48+4 */
00243     } exregs; /* 56 */
00244     struct __attribute__((__packed__))
00245     {
00246         char __pre_pad[2];
00247         L4_ktrace_t__Mword state; /* 36+4 */
00248         L4_ktrace_t__Address user_ip; /* 40+4 */
00249         L4_ktrace_t__Cpu_number src_cpu; /* 44+4 */
00250         L4_ktrace_t__Cpu_number target_cpu; /* 48+4 */
00251     } migration; /* 56 */
00252     struct __attribute__((__packed__))
00253     {
00254         char __pre_pad[2];
00255         L4_ktrace_t__Address user_ip; /* 36+4 */
00256     } timer; /* 40 */
00257     } m;
00258 } l4_tracebuffer_entry_t;

```

## 17.111 ktrace\_events.h

```

00001 /* Note, automatically generated from Fiasco binary */
00002 #pragma once
00003
00004 enum L4_ktrace_tbuf_entry_fixed
00005 {
00006     l4_ktrace_tbuf_unused = 0,
00007     l4_ktrace_tbuf_pf = 1,
00008     l4_ktrace_tbuf_ipc = 2,
00009     l4_ktrace_tbuf_ipc_res = 3,
00010     l4_ktrace_tbuf_ipc_trace = 4,
00011     l4_ktrace_tbuf_ke = 5,
00012     l4_ktrace_tbuf_ke_reg = 6,
00013     l4_ktrace_tbuf_breakpoint = 7,
00014     l4_ktrace_tbuf_ke_bin = 8,
00015     l4_ktrace_tbuf_dynentries = 9,
00016     l4_ktrace_tbuf_max = 128,
00017     l4_ktrace_tbuf_hidden = 128,
00018 };
00019
00020 typedef unsigned long L4_ktrace_t__Address;
00021 typedef unsigned long L4_ktrace_t__Cap_index;
00022 typedef void L4_ktrace_t__Context;
00023 typedef void L4_ktrace_t__Context__Drq;
00024 typedef unsigned L4_ktrace_t__Context__Drq_log__Type;
00025 typedef unsigned L4_ktrace_t__Cpu_number;
00026 typedef void L4_ktrace_t__Irq_base;
00027 typedef void L4_ktrace_t__Irq_chip;
00028 typedef void L4_ktrace_t__Kobject;
00029 typedef unsigned long L4_ktrace_t__L4_error;
00030 typedef unsigned long L4_ktrace_t__L4_msg_tag;
00031 typedef unsigned long L4_ktrace_t__L4_obj_ref;
00032 typedef unsigned L4_ktrace_t__L4_timeout_pair;
00033 typedef unsigned long L4_ktrace_t__Mword;

```

```

00034 typedef void L4_ktrace_t__Rcu_item;
00035 typedef void L4_ktrace_t__Sched_context;
00036 typedef long L4_ktrace_t__Smword;
00037 typedef void L4_ktrace_t__Space;
00038 typedef unsigned int L4_ktrace_t__Unsigned32;
00039 typedef unsigned long long L4_ktrace_t__Unsigned64;
00040 typedef unsigned char L4_ktrace_t__Unsigned8;
00041 typedef void L4_ktrace_t__cxx__Type_info;
00042
00043 typedef struct __attribute__((packed))
00044 {
00045     L4_ktrace_t__Mword _number; /* 0+8 */
00046     L4_ktrace_t__Address _ip; /* 8+8 */
00047     L4_ktrace_t__Unsigned64 _tsc; /* 16+8 */
00048     L4_ktrace_t__Context *_ctx; /* 24+8 */
00049     L4_ktrace_t__Unsigned32 _pmc1; /* 32+4 */
00050     L4_ktrace_t__Unsigned32 _pmc2; /* 36+4 */
00051     L4_ktrace_t__Unsigned32 _kclock; /* 40+4 */
00052     L4_ktrace_t__Unsigned8 _type; /* 44+1 */
00053     L4_ktrace_t__Unsigned8 _cpu; /* 45+1 */
00054     union __attribute__((__packed__))
00055     {
00056         struct __attribute__((__packed__))
00057         {
00058             char __pre_pad[2];
00059             void *func; /* 48+8 */
00060             L4_ktrace_t__Context *thread; /* 56+8 */
00061             L4_ktrace_t__Context__Drq *rq; /* 64+8 */
00062             L4_ktrace_t__Cpu_number target_cpu; /* 72+4 */
00063             L4_ktrace_t__Context__Drq_log__Type type; /* 76+4 */
00064             char wait; /* 80+1 */
00065         } drq; /* 88 */
00066         struct __attribute__((__packed__))
00067         {
00068             char __pre_pad[2];
00069             L4_ktrace_t__Mword state; /* 48+8 */
00070             L4_ktrace_t__Mword ip; /* 56+8 */
00071             L4_ktrace_t__Mword sp; /* 64+8 */
00072             L4_ktrace_t__Mword space; /* 72+8 */
00073             L4_ktrace_t__Mword err; /* 80+8 */
00074             unsigned char type; /* 88+1 */
00075             unsigned char trap; /* 89+1 */
00076         } vcpu; /* 96 */
00077         struct __attribute__((__packed__))
00078         {
00079             char __pre_pad[2];
00080             L4_ktrace_t__Smword op; /* 48+8 */
00081             L4_ktrace_t__Cap_index buffer; /* 56+8 */
00082             L4_ktrace_t__Mword id; /* 64+8 */
00083             L4_ktrace_t__Mword ram; /* 72+8 */
00084             L4_ktrace_t__Mword newo; /* 80+8 */
00085         } factory; /* 88 */
00086         struct __attribute__((__packed__))
00087         {
00088             char __pre_pad[2];
00089             L4_ktrace_t__Mword gate_dbg_id; /* 48+8 */
00090             L4_ktrace_t__Mword thread_dbg_id; /* 56+8 */
00091             L4_ktrace_t__Mword label; /* 64+8 */
00092         } gate; /* 72 */
00093         struct __attribute__((__packed__))
00094         {
00095             char __pre_pad[2];
00096             L4_ktrace_t__Irq_base *obj; /* 48+8 */
00097             L4_ktrace_t__Irq_chip *chip; /* 56+8 */
00098             L4_ktrace_t__Mword pin; /* 64+8 */
00099         } irq; /* 72 */
00100         struct __attribute__((__packed__))
00101         {
00102             char __pre_pad[2];
00103             L4_ktrace_t__Kobject *obj; /* 48+8 */
00104             L4_ktrace_t__Mword id; /* 56+8 */
00105             L4_ktrace_t__cxx__Type_info *type; /* 64+8 */
00106             L4_ktrace_t__Mword ram; /* 72+8 */
00107         } destroy; /* 80 */
00108         struct __attribute__((__packed__))
00109         {
00110             char __pre_pad[2];
00111             L4_ktrace_t__Cpu_number cpu; /* 48+4 */
00112             char __pad_l[4];
00113             L4_ktrace_t__Rcu_item *item; /* 56+8 */
00114             void *cb; /* 64+8 */
00115             unsigned char event; /* 72+1 */
00116         } rcu; /* 80 */
00117         struct __attribute__((__packed__))
00118         {
00119             char __pre_pad[2];
00120             L4_ktrace_t__Mword id; /* 48+8 */

```



```

00121     L4_ktrace_t__Mword mask; /* 56+8 */
00122     L4_ktrace_t__Mword fpage; /* 64+8 */
00123     char map; /* 72+1 */
00124 } tmap; /* 80 */
00125 struct __attribute__((__packed__))
00126 {
00127     char __pre_pad[2];
00128     L4_ktrace_t__Address _address; /* 48+8 */
00129     int _len; /* 56+4 */
00130     char __pad_1[4];
00131     L4_ktrace_t__Mword _value; /* 64+8 */
00132     int _mode; /* 72+4 */
00133 } bp; /* 80 */
00134 struct __attribute__((__packed__))
00135 {
00136     char __pre_pad[2];
00137     L4_ktrace_t__Context *dst; /* 48+8 */
00138     L4_ktrace_t__Context *dst_orig; /* 56+8 */
00139     L4_ktrace_t__Address kernel_ip; /* 64+8 */
00140     L4_ktrace_t__Mword lock_cnt; /* 72+8 */
00141     L4_ktrace_t__Space *from_space; /* 80+8 */
00142     L4_ktrace_t__Sched_context *from_sched; /* 88+8 */
00143     L4_ktrace_t__Mword from_prio; /* 96+8 */
00144 } context_switch; /* 104 */
00145 struct __attribute__((__packed__))
00146 {
00147 } empty; /* 48 */
00148 struct __attribute__((__packed__))
00149 {
00150     char __pre_pad[2];
00151     L4_ktrace_t__L4_msg_tag _tag; /* 48+8 */
00152     L4_ktrace_t__Mword _dword[2]; /* 56+16 */
00153     L4_ktrace_t__L4_obj_ref _dst; /* 72+8 */
00154     L4_ktrace_t__Mword _dbg_id; /* 80+8 */
00155     L4_ktrace_t__Mword _label; /* 88+8 */
00156     L4_ktrace_t__L4_timeout_pair _timeout; /* 96+4 */
00157     char __pad_1[4];
00158     L4_ktrace_t__Unsigned64 _to_abs_rcv; /* 104+8 */
00159 } ipc; /* 112 */
00160 struct __attribute__((__packed__))
00161 {
00162     L4_ktrace_t__Unsigned8 _have_snd; /* 46+1 */
00163     L4_ktrace_t__Unsigned8 _is_np; /* 47+1 */
00164     L4_ktrace_t__L4_msg_tag _tag; /* 48+8 */
00165     L4_ktrace_t__Mword _dword[2]; /* 56+16 */
00166     L4_ktrace_t__L4_error_result; /* 72+8 */
00167     L4_ktrace_t__Mword _from; /* 80+8 */
00168     L4_ktrace_t__L4_obj_ref _dst; /* 88+8 */
00169     L4_ktrace_t__Mword _pair_event; /* 96+8 */
00170 } ipc_res; /* 104 */
00171 struct __attribute__((__packed__))
00172 {
00173     char __pre_pad[2];
00174     union __attribute__((__packed__)) {
00175         char msg[80]; /* 0+80 */
00176         struct __attribute__((__packed__)) {
00177             char tag[2]; /* 0+2 */
00178             char __pad_1[6];
00179             char *ptr; /* 8+8 */
00180         } mptr; /* 0+16 */
00181     } msg; /* 48+80 */
00182 } ke; /* 128 */
00183 struct __attribute__((__packed__))
00184 {
00185     char _msg[80]; /* 46+80 */
00186 } ke_bin; /* 128 */
00187 struct __attribute__((__packed__))
00188 {
00189     char __pre_pad[2];
00190     L4_ktrace_t__Mword v[3]; /* 48+24 */
00191     union __attribute__((__packed__)) {
00192         char msg[56]; /* 0+56 */
00193         struct __attribute__((__packed__)) {
00194             char tag[2]; /* 0+2 */
00195             char __pad_1[6];
00196             char *ptr; /* 8+8 */
00197         } mptr; /* 0+16 */
00198     } msg; /* 72+56 */
00199 } ke_reg; /* 128 */
00200 struct __attribute__((__packed__))
00201 {
00202     char __pre_pad[2];
00203     L4_ktrace_t__Address _pfa; /* 48+8 */
00204     L4_ktrace_t__Mword _error; /* 56+8 */
00205     L4_ktrace_t__Space *_space; /* 64+8 */
00206 } pf; /* 72 */
00207 struct __attribute__((__packed__))

```

```

00208     {
00209         unsigned short mode; /* 46+2 */
00210         L4_ktrace_t__Context *owner; /* 48+8 */
00211         unsigned short id; /* 56+2 */
00212         unsigned short prio; /* 58+2 */
00213         char __pad_l[4];
00214         long left; /* 64+8 */
00215         unsigned long quantum; /* 72+8 */
00216     } sched; /* 80 */
00217     struct __attribute__((__packed__))
00218     {
00219         char __pre_pad[2];
00220         L4_ktrace_t__Unsigned32 _error; /* 48+4 */
00221         char __pad_l[4];
00222         L4_ktrace_t__Mword _cpsr; /* 56+8 */
00223         L4_ktrace_t__Mword _sp; /* 64+8 */
00224     } trap; /* 72 */
00225     struct __attribute__((__packed__))
00226     {
00227         char _padding[80]; /* 46+80 */
00228         char __post_pad[2]; /* 126+2 */
00229     } fullsize; /* 128 */
00230     struct __attribute__((__packed__))
00231     {
00232         char __pre_pad[2];
00233         L4_ktrace_t__Cap_index cap_idx; /* 48+8 */
00234     } ieh; /* 56 */
00235     struct __attribute__((__packed__))
00236     {
00237         char __pre_pad[2];
00238         L4_ktrace_t__Mword pfa; /* 48+8 */
00239         L4_ktrace_t__Cap_index cap_idx; /* 56+8 */
00240         L4_ktrace_t__Mword err; /* 64+8 */
00241     } ipfh; /* 72 */
00242     struct __attribute__((__packed__))
00243     {
00244         char __pre_pad[2];
00245         L4_ktrace_t__Mword id; /* 48+8 */
00246         L4_ktrace_t__Mword ip; /* 56+8 */
00247         L4_ktrace_t__Mword sp; /* 64+8 */
00248         L4_ktrace_t__Mword op; /* 72+8 */
00249     } exregs; /* 80 */
00250     struct __attribute__((__packed__))
00251     {
00252         char __pre_pad[2];
00253         L4_ktrace_t__Mword state; /* 48+8 */
00254         L4_ktrace_t__Address user_ip; /* 56+8 */
00255         L4_ktrace_t__Cpu_number src_cpu; /* 64+4 */
00256         L4_ktrace_t__Cpu_number target_cpu; /* 68+4 */
00257     } migration; /* 72 */
00258     struct __attribute__((__packed__))
00259     {
00260         char __pre_pad[2];
00261         L4_ktrace_t__Address user_ip; /* 48+8 */
00262     } timer; /* 56 */
00263     } m;
00264 } l4_tracebuffer_entry_t;

```

## 17.112 ktrace\_events.h

```

00001 /* Note, automatically generated from Fiasco binary */
00002 #pragma once
00003
00004 enum L4_ktrace_tbuf_entry_fixed
00005 {
00006     l4_ktrace_tbuf_unused = 0,
00007     l4_ktrace_tbuf_pf = 1,
00008     l4_ktrace_tbuf_ipc = 2,
00009     l4_ktrace_tbuf_ipc_res = 3,
00010     l4_ktrace_tbuf_ipc_trace = 4,
00011     l4_ktrace_tbuf_ke = 5,
00012     l4_ktrace_tbuf_ke_reg = 6,
00013     l4_ktrace_tbuf_breakpoint = 7,
00014     l4_ktrace_tbuf_ke_bin = 8,
00015     l4_ktrace_tbuf_dynentries = 9,
00016     l4_ktrace_tbuf_max = 128,
00017     l4_ktrace_tbuf_hidden = 128,
00018 };
00019
00020 typedef unsigned long L4_ktrace_t__Address;
00021 typedef unsigned long L4_ktrace_t__Cap_index;
00022 typedef void L4_ktrace_t__Context;
00023 typedef void L4_ktrace_t__Context__Drq;

```

```

00024 typedef unsigned L4_ktrace_t__Context__Drq_log__Type;
00025 typedef unsigned L4_ktrace_t__Cpu_number;
00026 typedef void L4_ktrace_t__Irq_base;
00027 typedef void L4_ktrace_t__Irq_chip;
00028 typedef void L4_ktrace_t__Kobject;
00029 typedef unsigned long L4_ktrace_t__L4_error;
00030 typedef unsigned long L4_ktrace_t__L4_msg_tag;
00031 typedef unsigned long L4_ktrace_t__L4_obj_ref;
00032 typedef unsigned L4_ktrace_t__L4_timeout_pair;
00033 typedef unsigned long L4_ktrace_t__Mword;
00034 typedef void L4_ktrace_t__Rcu_item;
00035 typedef void L4_ktrace_t__Sched_context;
00036 typedef long L4_ktrace_t__Smword;
00037 typedef void L4_ktrace_t__Space;
00038 typedef unsigned short L4_ktrace_t__Unsigned16;
00039 typedef unsigned int L4_ktrace_t__Unsigned32;
00040 typedef unsigned long long L4_ktrace_t__Unsigned64;
00041 typedef unsigned char L4_ktrace_t__Unsigned8;
00042 typedef void L4_ktrace_t__cxx__Type_info;
00043
00044 typedef struct __attribute__((packed))
00045 {
00046     L4_ktrace_t__Mword _number; /* 0+4 */
00047     L4_ktrace_t__Address _ip; /* 4+4 */
00048     L4_ktrace_t__Unsigned64 _tsc; /* 8+8 */
00049     L4_ktrace_t__Context *_ctx; /* 16+4 */
00050     L4_ktrace_t__Unsigned32 _pmc1; /* 20+4 */
00051     L4_ktrace_t__Unsigned32 _pmc2; /* 24+4 */
00052     L4_ktrace_t__Unsigned32 _kclock; /* 28+4 */
00053     L4_ktrace_t__Unsigned8 _type; /* 32+1 */
00054     L4_ktrace_t__Unsigned8 _cpu; /* 33+1 */
00055     union __attribute__((packed))
00056     {
00057         struct __attribute__((packed))
00058         {
00059             char __pre_pad[2];
00060             void *func; /* 36+4 */
00061             L4_ktrace_t__Context *thread; /* 40+4 */
00062             L4_ktrace_t__Context__Drq *rq; /* 44+4 */
00063             L4_ktrace_t__Cpu_number target_cpu; /* 48+4 */
00064             L4_ktrace_t__Context__Drq_log__Type type; /* 52+4 */
00065             char wait; /* 56+1 */
00066         } drq; /* 64 */
00067         struct __attribute__((packed))
00068         {
00069             char __pre_pad[2];
00070             L4_ktrace_t__Mword state; /* 36+4 */
00071             L4_ktrace_t__Mword ip; /* 40+4 */
00072             L4_ktrace_t__Mword sp; /* 44+4 */
00073             L4_ktrace_t__Mword space; /* 48+4 */
00074             L4_ktrace_t__Mword err; /* 52+4 */
00075             unsigned char type; /* 56+1 */
00076             unsigned char trap; /* 57+1 */
00077         } vcpu; /* 64 */
00078         struct __attribute__((packed))
00079         {
00080             char __pre_pad[2];
00081             L4_ktrace_t__Smword op; /* 36+4 */
00082             L4_ktrace_t__Cap_index buffer; /* 40+4 */
00083             L4_ktrace_t__Mword id; /* 44+4 */
00084             L4_ktrace_t__Mword ram; /* 48+4 */
00085             L4_ktrace_t__Mword newo; /* 52+4 */
00086         } factory; /* 56 */
00087         struct __attribute__((packed))
00088         {
00089             char __pre_pad[2];
00090             L4_ktrace_t__Mword gate_dbg_id; /* 36+4 */
00091             L4_ktrace_t__Mword thread_dbg_id; /* 40+4 */
00092             L4_ktrace_t__Mword label; /* 44+4 */
00093         } gate; /* 48 */
00094         struct __attribute__((packed))
00095         {
00096             char __pre_pad[2];
00097             L4_ktrace_t__Irq_base *obj; /* 36+4 */
00098             L4_ktrace_t__Irq_chip *chip; /* 40+4 */
00099             L4_ktrace_t__Mword pin; /* 44+4 */
00100         } irq; /* 48 */
00101         struct __attribute__((packed))
00102         {
00103             char __pre_pad[2];
00104             L4_ktrace_t__Kobject *obj; /* 36+4 */
00105             L4_ktrace_t__Mword id; /* 40+4 */
00106             L4_ktrace_t__cxx__Type_info *type; /* 44+4 */
00107             L4_ktrace_t__Mword ram; /* 48+4 */
00108         } destroy; /* 56 */
00109         struct __attribute__((packed))
00110         {

```

```

00111     char __pre_pad[2];
00112     L4_ktrace_t__Cpu_number cpu; /* 36+4 */
00113     L4_ktrace_t__Rcu_item *item; /* 40+4 */
00114     void *cb; /* 44+4 */
00115     unsigned char event; /* 48+1 */
00116 } rcu; /* 56 */
00117 struct __attribute__((__packed__))
00118 {
00119     char __pre_pad[2];
00120     L4_ktrace_t__Mword id; /* 36+4 */
00121     L4_ktrace_t__Mword mask; /* 40+4 */
00122     L4_ktrace_t__Mword fpage; /* 44+4 */
00123     char map; /* 48+1 */
00124 } tmap; /* 56 */
00125 struct __attribute__((__packed__))
00126 {
00127     char __pre_pad[2];
00128     L4_ktrace_t__Address _address; /* 36+4 */
00129     int _len; /* 40+4 */
00130     L4_ktrace_t__Mword _value; /* 44+4 */
00131     int _mode; /* 48+4 */
00132 } bp; /* 56 */
00133 struct __attribute__((__packed__))
00134 {
00135     char __pre_pad[2];
00136     L4_ktrace_t__Context *dst; /* 36+4 */
00137     L4_ktrace_t__Context *dst_orig; /* 40+4 */
00138     L4_ktrace_t__Address kernel_ip; /* 44+4 */
00139     L4_ktrace_t__Mword lock_cnt; /* 48+4 */
00140     L4_ktrace_t__Space *from_space; /* 52+4 */
00141     L4_ktrace_t__Sched_context *from_sched; /* 56+4 */
00142     L4_ktrace_t__Mword from_prio; /* 60+4 */
00143 } context_switch; /* 64 */
00144 struct __attribute__((__packed__))
00145 {
00146 } empty; /* 40 */
00147 struct __attribute__((__packed__))
00148 {
00149     char __pre_pad[2];
00150     L4_ktrace_t__L4_msg_tag _tag; /* 36+4 */
00151     L4_ktrace_t__Mword _dword[2]; /* 40+8 */
00152     L4_ktrace_t__L4_obj_ref _dst; /* 48+4 */
00153     L4_ktrace_t__Mword _dbg_id; /* 52+4 */
00154     L4_ktrace_t__Mword _label; /* 56+4 */
00155     L4_ktrace_t__L4_timeout_pair _timeout; /* 60+4 */
00156 } ipc; /* 64 */
00157 struct __attribute__((__packed__))
00158 {
00159     L4_ktrace_t__Unsigned8 _have_snd; /* 34+1 */
00160     L4_ktrace_t__Unsigned8 _is_np; /* 35+1 */
00161     L4_ktrace_t__L4_msg_tag _tag; /* 36+4 */
00162     L4_ktrace_t__Mword _dword[2]; /* 40+8 */
00163     L4_ktrace_t__L4_error _result; /* 48+4 */
00164     L4_ktrace_t__Mword _from; /* 52+4 */
00165     L4_ktrace_t__L4_obj_ref _dst; /* 56+4 */
00166     L4_ktrace_t__Mword _pair_event; /* 60+4 */
00167 } ipc_res; /* 64 */
00168 struct __attribute__((__packed__))
00169 {
00170     char __pre_pad[2];
00171     union __attribute__((__packed__)) {
00172         char msg[24]; /* 0+24 */
00173         struct __attribute__((__packed__)) {
00174             char tag[2]; /* 0+2 */
00175             char __pad_1[2];
00176             char *ptr; /* 4+4 */
00177         } mptr; /* 0+8 */
00178     } msg; /* 36+24 */
00179 } ke; /* 64 */
00180 struct __attribute__((__packed__))
00181 {
00182     char _msg[24]; /* 34+24 */
00183 } ke_bin; /* 64 */
00184 struct __attribute__((__packed__))
00185 {
00186     char __pre_pad[2];
00187     L4_ktrace_t__Mword v[3]; /* 36+12 */
00188     union __attribute__((__packed__)) {
00189         char msg[12]; /* 0+12 */
00190         struct __attribute__((__packed__)) {
00191             char tag[2]; /* 0+2 */
00192             char __pad_1[2];
00193             char *ptr; /* 4+4 */
00194         } mptr; /* 0+8 */
00195     } msg; /* 48+12 */
00196 } ke_reg; /* 64 */
00197 struct __attribute__((__packed__))

```

```

00198     {
00199         char __pre_pad[2];
00200         L4_ktrace_t__Address _pfa; /* 36+4 */
00201         L4_ktrace_t__Mword _error; /* 40+4 */
00202         L4_ktrace_t__Space *_space; /* 44+4 */
00203     } pf; /* 48 */
00204     struct __attribute__((__packed__))
00205     {
00206         unsigned short mode; /* 34+2 */
00207         L4_ktrace_t__Context *owner; /* 36+4 */
00208         unsigned short id; /* 40+2 */
00209         unsigned short prio; /* 42+2 */
00210         long left; /* 44+4 */
00211         unsigned long quantum; /* 48+4 */
00212     } sched; /* 56 */
00213     struct __attribute__((__packed__))
00214     {
00215         L4_ktrace_t__Unsigned8 _trapno; /* 34+1 */
00216         char __pad_1[1];
00217         L4_ktrace_t__Unsigned16 _error; /* 36+2 */
00218         char __pad_2[2];
00219         L4_ktrace_t__Mword _ebp; /* 40+4 */
00220         L4_ktrace_t__Mword _cr2; /* 44+4 */
00221         L4_ktrace_t__Mword _eax; /* 48+4 */
00222         L4_ktrace_t__Mword _eflags; /* 52+4 */
00223         L4_ktrace_t__Mword _esp; /* 56+4 */
00224         L4_ktrace_t__Unsigned16 _cs; /* 60+2 */
00225         L4_ktrace_t__Unsigned16 _ds; /* 62+2 */
00226     } trap; /* 64 */
00227     struct __attribute__((__packed__))
00228     {
00229         char __padding[24]; /* 34+24 */
00230         char __post_pad[6]; /* 58+6 */
00231     } fullsize; /* 64 */
00232     struct __attribute__((__packed__))
00233     {
00234         char __pre_pad[2];
00235         L4_ktrace_t__Cap_index cap_idx; /* 36+4 */
00236     } ieh; /* 40 */
00237     struct __attribute__((__packed__))
00238     {
00239         char __pre_pad[2];
00240         L4_ktrace_t__Mword pfa; /* 36+4 */
00241         L4_ktrace_t__Cap_index cap_idx; /* 40+4 */
00242         L4_ktrace_t__Mword err; /* 44+4 */
00243     } ipfh; /* 48 */
00244     struct __attribute__((__packed__))
00245     {
00246         char __pre_pad[2];
00247         L4_ktrace_t__Mword id; /* 36+4 */
00248         L4_ktrace_t__Mword ip; /* 40+4 */
00249         L4_ktrace_t__Mword sp; /* 44+4 */
00250         L4_ktrace_t__Mword op; /* 48+4 */
00251     } exregs; /* 56 */
00252     struct __attribute__((__packed__))
00253     {
00254         char __pre_pad[2];
00255         L4_ktrace_t__Mword state; /* 36+4 */
00256         L4_ktrace_t__Address user_ip; /* 40+4 */
00257         L4_ktrace_t__Cpu_number src_cpu; /* 44+4 */
00258         L4_ktrace_t__Cpu_number target_cpu; /* 48+4 */
00259     } migration; /* 56 */
00260     struct __attribute__((__packed__))
00261     {
00262         char __pre_pad[2];
00263         L4_ktrace_t__Address user_ip; /* 36+4 */
00264     } timer; /* 40 */
00265     struct __attribute__((__packed__))
00266     {
00267         char __pre_pad[2];
00268         L4_ktrace_t__Mword exitcode; /* 36+4 */
00269         L4_ktrace_t__Mword exitinfo1; /* 40+4 */
00270         L4_ktrace_t__Mword exitinfo2; /* 44+4 */
00271         L4_ktrace_t__Mword rip; /* 48+4 */
00272     } svm; /* 56 */
00273     } m;
00274 } l4_tracebuffer_entry_t;

```

## 17.113 amd64/l4/sys/linkage.h File Reference

Linkage.

## Macros

- **#define L4\_CV**  
*Define calling convention.*

### 17.113.1 Detailed Description

Linkage.

Definition in file [linkage.h](#).

## 17.114 linkage.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *                Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *                Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *                economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #ifndef __L4__SYS__ARCH_AMD64__LINKAGE_H__
00015 #define __L4__SYS__ARCH_AMD64__LINKAGE_H__
00016
00017 #ifdef __ASSEMBLY__
00018
00019 #ifndef ENTRY
00020 #define ENTRY(name) \
00021     .globl name; \
00022     .p2align(2); \
00023     name:
00024
00025 #endif /* __ASSEMBLY__ */
00026 #endif /* ! ENTRY */
00027
00028 #define L4_FASTCALL(x)      x
00029 #define l4_fastcall
00030
00036 #define L4_CV
00037
00038 #endif /* ! __L4__SYS__ARCH_AMD64__LINKAGE_H__ */

```

### 17.115 arm/l4/sys/linkage.h File Reference

Linkage.

## Macros

- **#define L4\_CV**  
*Define calling convention.*

### 17.115.1 Detailed Description

Linkage.

Definition in file [linkage.h](#).

## 17.116 linkage.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #ifndef __L4__SYS__ARCH_ARM__LINKAGE_H__
00014 #define __L4__SYS__ARCH_ARM__LINKAGE_H__
00015
00016 #ifdef __ASSEMBLY__
00017 #ifndef ENTRY
00018 #define ENTRY(name) \
00019     .globl name; \
00020     .p2align(2); \
00021     name:
00022 #endif
00023 #endif
00024
00025 #define L4_FASTCALL(x)  x
00026 #define l4_fastcall
00027
00033 #define L4_CV
00034
00035 #ifdef __PIC__
00036 # define L4_LONG_CALL
00037 #else
00038 # define L4_LONG_CALL __attribute__((long_call))
00039 #endif
00040
00041 #endif /* ! __L4__SYS__ARCH_ARM__LINKAGE_H__ */

```

## 17.117 linkage.h

```

00001 #pragma once
00002
00008 #define L4_CV
00009
00010 #define L4_FASTCALL(x)  x
00011 #define l4_fastcall

```

## 17.118 x86/I4/sys/linkage.h File Reference

Linkage.

### Macros

- **#define L4\_CV**  
*Define calling convention.*

### 17.118.1 Detailed Description

Linkage.

Definition in file [linkage.h](#).

## 17.119 linkage.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Frank Mehnert <fm3@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #ifndef __L4__SYS__ARCH_X86__LINKAGE_H__
00015 #define __L4__SYS__ARCH_X86__LINKAGE_H__
00016
00017 #ifdef __ASSEMBLY__
00018
00019 #ifndef ENTRY
00020 #define ENTRY(name) \
00021     .globl name; \
00022     .p2align(2); \
00023     name:
00024
00025 #endif /* ! ENTRY */
00026 #endif /* __ASSEMBLY__ */
00027
00028 #define L4_FASTCALL(x) x __attribute__((regparm(3)))
00029 #define l4_fastcall __attribute__((regparm(3)))
00030
00036 #define L4_CV __attribute__((regparm(0)))
00037
00038 #endif /* ! __L4__SYS__ARCH_X86__LINKAGE_H__ */

```

## 17.120 arm/l4/sys/mem\_op.h File Reference

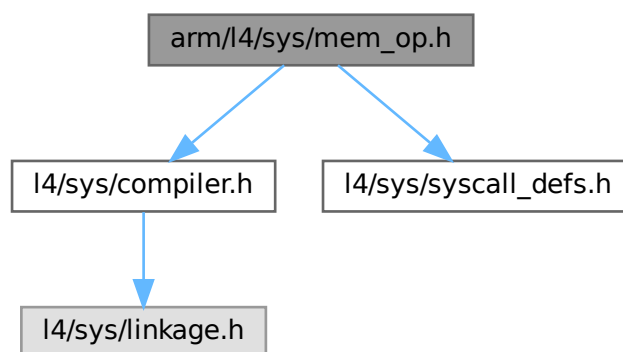
Memory access functions (ARM specific).

```

#include <l4/sys/compiler.h>
#include <l4/sys/syscall_defs.h>

```

Include dependency graph for mem\_op.h:



### Enumerations

- enum `L4_mem_op_widths` { `L4_MEM_WIDTH_1BYTE` = 0, `L4_MEM_WIDTH_2BYTE` = 1, `L4_MEM_WIDTH_4BYTE` = 2 }
- Memory access width definitions.*



## Functions

- unsigned long [l4\\_mem\\_read](#) (unsigned long virtaddress, unsigned width)  
*Read user task memory from kernel privilege level.*
- void [l4\\_mem\\_write](#) (unsigned long virtaddress, unsigned width, unsigned long value)  
*Write user task memory from kernel privilege level.*
- unsigned long [l4\\_mem\\_arm\\_op\\_call](#) (unsigned long op, unsigned long va, unsigned long width, unsigned long value)  
*Implementations.*

### 17.120.1 Detailed Description

Memory access functions (ARM specific).

#### Date

2010-10

#### Author

Adam Lackorzynski [adam@os.inf.tu-dresden.de](mailto:adam@os.inf.tu-dresden.de)

Definition in file [mem\\_op.h](#).

## 17.121 mem\_op.h

[Go to the documentation of this file.](#)

```

00001
00009 /*
00010  * (c) 2010 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * License: see LICENSE.spdx (in this directory or the directories above)
00014  */
00015 #ifndef __L4SYS__INCLUDE__ARCH_ARM__MEM_OP_H__
00016 #define __L4SYS__INCLUDE__ARCH_ARM__MEM_OP_H__
00017
00018 #include <l4/sys/compiler.h>
00019 #include <l4/sys/syscall_defs.h>
00020
00021 L4_BEGIN_DECLS
00022
00035
00040 enum L4_mem_op_widths
00041 {
00042     L4_MEM_WIDTH_1BYTE = 0,
00043     L4_MEM_WIDTH_2BYTE = 1,
00044     L4_MEM_WIDTH_4BYTE = 2,
00045 };
00046
00059 L4_INLINE unsigned long
00060 l4_mem_read(unsigned long virtaddress, unsigned width);
00061
00074 L4_INLINE void
00075 l4_mem_write(unsigned long virtaddress, unsigned width,
00076              unsigned long value);
00077
00078 enum L4_mem_ops
00079 {
00080     L4_MEM_OP_MEM_READ = 0x10,
00081     L4_MEM_OP_MEM_WRITE = 0x11,
00082 };
00083
00087 L4_INLINE unsigned long

```

```

00088 l4_mem_arm_op_call(unsigned long op,
00089                     unsigned long va,
00090                     unsigned long width,
00091                     unsigned long value);
00092
00093
00094
00095 L4_INLINE unsigned long
00096 l4_mem_arm_op_call(unsigned long op,
00097                     unsigned long va,
00098                     unsigned long width,
00099                     unsigned long value)
00100 {
00101     register unsigned long _op    __asm__ ("r0") = op;
00102     register unsigned long _va    __asm__ ("r1") = va;
00103     register unsigned long _width __asm__ ("r2") = width;
00104     register unsigned long _value __asm__ ("r3") = value;
00105
00106     __asm__ __volatile__
00107     ("@ l4_cache_op_arm_call(start) \n\t"
00108      "mov     r5, %[sc]          \n\t"
00109      "blx     __l4_sys_syscall   \n\t"
00110      "@ l4_cache_op_arm_call(end) \n\t"
00111      :
00112      "=r" (_op),
00113      "=r" (_va),
00114      "=r" (_width),
00115      "=r" (_value)
00116      :
00117      [sc] "i" (L4_SYSCALL_MEM_OP),
00118      "0" (_op),
00119      "1" (_va),
00120      "2" (_width),
00121      "3" (_value)
00122      :
00123      "cc", "memory", "r5", "ip", "lr"
00124      );
00125
00126     return _value;
00127 }
00128
00129 L4_INLINE unsigned long
00130 l4_mem_read(unsigned long virtaddress, unsigned width)
00131 {
00132     return l4_mem_arm_op_call(L4_MEM_OP_MEM_READ, virtaddress, width, 0);
00133 }
00134
00135 L4_INLINE void
00136 l4_mem_write(unsigned long virtaddress, unsigned width,
00137              unsigned long value)
00138 {
00139     l4_mem_arm_op_call(L4_MEM_OP_MEM_WRITE, virtaddress, width, value);
00140 }
00141
00142 L4_END_DECLS
00143
00144 #endif /* ! __L4SYS__INCLUDE__ARCH_ARM__MEM_OP_H__ */

```

## 17.122 vm.h

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 #include <l4/sys/__vm-svm.h>
00015 #include <l4/sys/__vm-vmx.h>

```

## 17.123 arm/l4/sys/vm.h File Reference

ARM virtualization interface.

## Data Structures

- struct [l4\\_vm\\_tz\\_state](#)  
*state structure for TrustZone VMs*

### 17.123.1 Detailed Description

ARM virtualization interface.

Definition in file [vm.h](#).

## 17.124 vm.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00019
00024 struct l4_vm_tz_state_mode
00025 {
00026     l4_umword_t sp;
00027     l4_umword_t lr;
00028     l4_umword_t spsr;
00029 };
00030
00031 struct l4_vm_tz_state_irq_inject
00032 {
00033     l4_uint32_t group;
00034     l4_uint32_t irqs[8];
00035 };
00036
00041 struct l4_vm_tz_state
00042 {
00043     l4_umword_t r[13]; // r0 - r12
00044
00045     l4_umword_t sp_usr;
00046     l4_umword_t lr_usr;
00047
00048     struct l4_vm_tz_state_mode irq;
00049
00050     l4_umword_t r_fiq[5]; // r8 - r12
00051     struct l4_vm_tz_state_mode fiq;
00052     struct l4_vm_tz_state_mode abt;
00053     struct l4_vm_tz_state_mode und;
00054     struct l4_vm_tz_state_mode svc;
00055
00056     l4_umword_t pc;
00057     l4_umword_t cpsr;
00058
00059     l4_umword_t pending_events;
00060     l4_uint32_t cpacr;
00061     l4_umword_t cpl0_fpexc;
00062
00063     l4_umword_t pfs;
00064     l4_umword_t pfa;
00065     l4_umword_t exit_reason;
00066
00067     struct l4_vm_tz_state_irq_inject irq_inject;
00068 };
00069
00070 enum L4_vm_exit_reason
00071 {
00072     L4_vm_exit_reason_vmm_call    = 1,
00073     L4_vm_exit_reason_inst_abort  = 2,
00074     L4_vm_exit_reason_data_abort  = 3,
00075     L4_vm_exit_reason_irq        = 4,

```

```

00076     L4_vm_exit_reason_fiq          = 5,
00077     L4_vm_exit_reason_undef        = 6,
00078 };
00079
00080 L4_INLINE int
00081 l4_vm_tz_irq_inject(struct l4_vm_tz_state *state, unsigned irq);
00082
00083 L4_INLINE int
00084 l4_vm_tz_irq_inject(struct l4_vm_tz_state *state, unsigned irq)
00085 {
00086     if (irq > sizeof(state->irq_inject.irqs) * 8)
00087         return -L4_EINVAL;
00088
00089     unsigned g = irq / 32;
00090     state->irq_inject.group |= 1 « g;
00091     state->irq_inject.irqs[g] |= 1 « (irq & 31);
00092
00093     return 0;
00094 }

```

## 17.125 vm.h

```

00001
00002

```

## 17.126 vm.h

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *                Henning Schild <hschild@os.inf.tu-dresden.de>
00009  *                economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00015 #include <l4/sys/__vm-svm.h>
00016 #include <l4/sys/__vm-vmx.h>

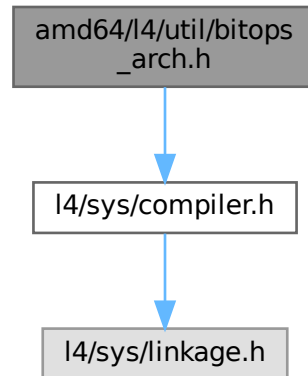
```

## 17.127 amd64/l4/util/bitops\_arch.h File Reference

amd64 bit manipulation functions

```
#include <l4/sys/compiler.h>
```

Include dependency graph for bitops\_arch.h:



### 17.127.1 Detailed Description

amd64 bit manipulation functions

Definition in file [bitops\\_arch.h](#).

## 17.128 bitops\_arch.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (C) 2000-2009 Technische Universität Dresden (Germany)
00003  * Copyright (C) 2016, 2022, 2024 Kernkonzept GmbH. All rights reserved.
00004  * Author(s): Lars Reuther <reuther@os.inf.tu-dresden.de>
00005  *            Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00006  *            Frank Mehnert <frank.mehnert@kernkonzept.com>
00007  *
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010
00016
00017 #pragma once
00018
00019 #include <l4/sys/compiler.h>
00020
00021 /*
00022  * Note: The following Assembler statement may produce wrong code:
00023  *   asm volatile ("btsl %1, %2" : "=ccc"(r) : "Jr"(63), "m"(m) : "memory");
00024  *
00025  * The compiler might chose the first variant because the bit number is smaller
00026  * than 64. However, 'bts' is encoded as 32-bit variant ('btsl') and thus only
00027  * supports immediate bit values up to 31. Some assemblers support immediate
00028  * offsets > 31 by adapting the memory address accordingly but GAS does not.
00029  * With GAS, the instruction will encode an immediate value of 63 but the CPU
00030  * will set bit 31 instead of bit 63!
00031  *
00032  * Therefore, if we would use 'btsl' instead of 'btsq', the correct constraint
00033  * for the bit number parameter would be "Ir" instead of "Jr".
00034  */
00035
00036 L4_BEGIN_DECLS
00037

```

```

00038 /* set bit */
00039 #define __L4UTIL_BITOPS_HAVE_ARCH_SET_BIT
00040 L4_INLINE void
00041 l4util_set_bit(int b, volatile l4_umword_t * dest)
00042 {
00043     __asm__ __volatile__
00044     (
00045         "lock; btsq  %1,%0  \n\t"
00046         :
00047         :
00048         "m"    (*dest), /* 0 mem, destination operand */
00049         "Jr"    ((l4_umword_t)b) /* 1, bit number */
00050         :
00051         "memory", "cc"
00052         );
00053 }
00054
00055 /* clear bit */
00056 #define __L4UTIL_BITOPS_HAVE_ARCH_CLEAR_BIT
00057 L4_INLINE void
00058 l4util_clear_bit(int b, volatile l4_umword_t * dest)
00059 {
00060     __asm__ __volatile__
00061     (
00062         "lock; btrq  %1,%0  \n\t"
00063         :
00064         :
00065         "m"    (*dest), /* 0 mem, destination operand */
00066         "Jr"    ((l4_umword_t)b) /* 1, bit number */
00067         :
00068         "memory", "cc"
00069         );
00070 }
00071
00072 /* change bit */
00073 #define __L4UTIL_BITOPS_HAVE_ARCH_COMPLEMENT_BIT
00074 L4_INLINE void
00075 l4util_complement_bit(int b, volatile l4_umword_t * dest)
00076 {
00077     __asm__ __volatile__
00078     (
00079         "lock; btcq  %1,%0  \n\t"
00080         :
00081         :
00082         "m"    (*dest), /* 0 mem, destination operand */
00083         "Jr"    ((l4_umword_t)b) /* 1, bit number */
00084         :
00085         "memory", "cc"
00086         );
00087 }
00088
00089 /* test bit */
00090 #define __L4UTIL_BITOPS_HAVE_ARCH_TEST_BIT
00091 L4_INLINE int
00092 l4util_test_bit(int b, const volatile l4_umword_t * dest)
00093 {
00094     l4_int8_t bit;
00095
00096     __asm__ __volatile__
00097     (
00098         "btq  %2,%1  \n\t"
00099         :
00100         "=@ccc" (bit) /* 0, old bit value */
00101         :
00102         "m"    (*dest), /* 1 mem, destination operand */
00103         "Jr"    ((l4_umword_t)b) /* 2, bit number */
00104         :
00105         "memory"
00106         );
00107
00108     return bit;
00109 }
00110
00111 /* bit test and set */
00112 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_SET
00113 L4_INLINE int
00114 l4util_bts(int b, volatile l4_umword_t * dest)
00115 {
00116     l4_int8_t bit;
00117
00118     __asm__ __volatile__
00119     (
00120         "lock; btsq  %2,%1  \n\t"
00121         :
00122         "=@ccc" (bit) /* 0, old bit value */
00123         :
00124         "m"    (*dest), /* 1 mem, destination operand */

```

```

00125     "Jr" ((l4_umword_t)b) /* 2, bit number */
00126     :
00127     "memory"
00128     );
00129
00130     return bit;
00131 }
00132
00133 /* bit test and reset */
00134 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_RESET
00135 L4_INLINE int
00136 l4util_btr(int b, volatile l4_umword_t * dest)
00137 {
00138     l4_int8_t bit;
00139
00140     __asm__ __volatile__
00141     (
00142         "lock; btrq %2,%1 \n\t"
00143         :
00144         "=@ccc" (bit) /* 0, old bit value */
00145         :
00146         "m" (*dest), /* 1 mem, destination operand */
00147         "Jr" ((l4_umword_t)b) /* 2, bit number */
00148         :
00149         "memory"
00150         );
00151
00152     return bit;
00153 }
00154
00155 /* bit test and complement */
00156 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_COMPLEMENT
00157 L4_INLINE int
00158 l4util_btc(int b, volatile l4_umword_t * dest)
00159 {
00160     l4_int8_t bit;
00161
00162     __asm__ __volatile__
00163     (
00164         "lock; btcq %2,%1 \n\t"
00165         :
00166         "=@ccc" (bit) /* 0, old bit value */
00167         :
00168         "m" (*dest), /* 1 mem, destination operand */
00169         "Jr" ((l4_umword_t)b) /* 2, bit number */
00170         :
00171         "memory"
00172         );
00173
00174     return bit;
00175 }
00176
00177 /* bit scan reverse */
00178 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_REVERSE
00179 L4_INLINE int
00180 l4util_bsr(l4_umword_t word)
00181 {
00182     l4_umword_t tmp;
00183
00184     if (L4_UNLIKELY(word == 0))
00185         return -1;
00186
00187     __asm__ __volatile__
00188     (
00189         "bsrq %1,%0 \n\t"
00190         :
00191         "=r" (tmp) /* 0, index of most significant set bit */
00192         :
00193         "r" (word) /* 1, argument */
00194         );
00195
00196     return tmp;
00197 }
00198
00199 /* bit scan forward */
00200 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_FORWARD
00201 L4_INLINE int
00202 l4util_bsfl(l4_umword_t word)
00203 {
00204     l4_umword_t tmp;
00205
00206     if (L4_UNLIKELY(word == 0))
00207         return -1;
00208
00209     __asm__ __volatile__
00210     (
00211         "bsfq %1,%0 \n\t"

```

```

00212     :
00213     "=r" (tmp)          /* 0, index of least significant set bit */
00214     :
00215     "r" (word)          /* 1, argument */
00216     );
00217
00218     return tmp;
00219 }
00220
00221 #define __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_SET_BIT
00222 L4_INLINE int
00223 l4util_find_first_set_bit(const void * dest, l4_size_t size)
00224 {
00225     l4_mword_t dummy0, dummy1, res;
00226
00227     __asm__ __volatile__
00228     (
00229         "repe; scasq          \n\t"
00230         "jz     1f            \n\t"
00231         "lea    -8(%%rdi),%%rdi \n\t"
00232         "bsf    (%%rdi),%%rax   \n\t"
00233         "1:      \n\t"
00234         "sub    %%rbx,%%rdi     \n\t"
00235         "shl    $3,%%rdi        \n\t"
00236         "add    %%rdi,%%rax     \n\t"
00237         :
00238         "=a" (res), "=c" (dummy0), "=D" (dummy1)
00239         :
00240         "a" (0), "b" (dest), "c" ((size + 63) >> 6), "D" (dest)
00241         :
00242         "cc", "memory");
00243
00244     return res;
00245 }
00246
00247 #define __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_ZERO_BIT
00248 L4_INLINE int
00249 l4util_find_first_zero_bit(const void * dest, l4_size_t size)
00250 {
00251     l4_mword_t dummy0, dummy1, dummy2, res;
00252
00253     if (!size)
00254         return 0;
00255
00256     __asm__ __volatile__
00257     (
00258         "repe; scasq          \n\t"
00259         "je     1f            \n\t"
00260         "xor    -8(%%rdi),%%rax \n\t"
00261         "sub    $8,%%rdi       \n\t"
00262         "bsf    %%rax,%%rdx     \n\t"
00263         "1:      \n\t"
00264         "sub    %%rsi,%%rdi     \n\t"
00265         "shl    $3,%%rdi        \n\t"
00266         "add    %%rdi,%%rdx     \n\t"
00267         :
00268         "=a" (dummy0), "=c" (dummy1), "=d" (res), "=D" (dummy2)
00269         :
00270         "a" (~0UL), "c" ((size + 63) >> 6), "d" (0), "D" (dest), "S" (dest)
00271         :
00272         "cc", "memory");
00273
00274     return res;
00275 }
00276
00277 L4_END_DECLS

```

## 17.129 arm/l4/util/bitops\_arch.h File Reference

ARM specific implementation of bitops functions.

### 17.129.1 Detailed Description

ARM specific implementation of bitops functions.

Definition in file [bitops\\_arch.h](#).



## 17.130 bitops\_arch.h

[Go to the documentation of this file.](#)

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010 #ifndef __L4UTIL__ARCH_ARM__BITOPS_ARCH_H__
00011 #define __L4UTIL__ARCH_ARM__BITOPS_ARCH_H__
00012
00013
00014 #endif /* ! __L4UTIL__ARCH_ARM__BITOPS_ARCH_H__ */
```

## 17.131 x86/I4/util/bitops\_arch.h File Reference

x86 bit manipulation functions

### 17.131.1 Detailed Description

x86 bit manipulation functions

Definition in file [bitops\\_arch.h](#).

## 17.132 bitops\_arch.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  * Copyright (C) 2000-2009 Technische Universität Dresden (Germany)
00003  * Copyright (C) 2016, 2022, 2024 Kernkonzept GmbH. All rights reserved.
00004  * Author(s): Lars Reuther <reuther@os.inf.tu-dresden.de>
00005  *             Frank Mehnert <frank.mehnert@kernkonzept.com>
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00015
00016 #pragma once
00017
00018 L4_BEGIN_DECLS
00019
00020 /* set bit */
00021 #define __L4UTIL_BITOPS_HAVE_ARCH_SET_BIT
00022 L4_INLINE void
00023 l4util_set_bit(int b, volatile l4_umword_t * dest)
00024 {
00025     __asm__ __volatile__
00026     (
00027         "lock; btsl  %1,%0  \n\t"
00028         :
00029         :
00030         "m"    (*dest),    /* 0 mem, destination operand */
00031         "Ir"   (b)         /* 1,    bit number */
00032         :
00033         "memory", "cc"
00034         );
00035 }
00036
00037 /* clear bit */
00038 #define __L4UTIL_BITOPS_HAVE_ARCH_CLEAR_BIT
00039 L4_INLINE void
00040 l4util_clear_bit(int b, volatile l4_umword_t * dest)
00041 {
00042     __asm__ __volatile__
00043     (
```

```

00044     "lock; btrl %1,%0  \n\t"
00045     :
00046     :
00047     "m"    (*dest),    /* 0 mem, destination operand */
00048     "Ir"   (b)         /* 1,      bit number */
00049     :
00050     "memory", "cc"
00051     );
00052 }
00053
00054 /* change bit */
00055 #define __L4UTIL_BITOPS_HAVE_ARCH_COMPLEMENT_BIT
00056 L4_INLINE void
00057 l4util_complement_bit(int b, volatile l4_umword_t * dest)
00058 {
00059     __asm__ __volatile__
00060     (
00061         "lock; btcl %1,%0  \n\t"
00062         :
00063         :
00064         "m"    (*dest),    /* 0 mem, destination operand */
00065         "Ir"   (b)         /* 1,      bit number */
00066         :
00067         "memory", "cc"
00068         );
00069 }
00070
00071 /* test bit */
00072 #define __L4UTIL_BITOPS_HAVE_ARCH_TEST_BIT
00073 L4_INLINE int
00074 l4util_test_bit(int b, const volatile l4_umword_t * dest)
00075 {
00076     l4_int8_t bit;
00077
00078     __asm__ __volatile__
00079     (
00080         "btl %2,%1  \n\t"
00081         :
00082         "=@ccc" (bit)    /* 0,      old bit value */
00083         :
00084         "m"    (*dest),    /* 1 mem, destination operand */
00085         "Ir"   (b)         /* 2,      bit number */
00086         :
00087         "memory"
00088         );
00089
00090     return bit;
00091 }
00092
00093 /* bit test and set */
00094 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_SET
00095 L4_INLINE int
00096 l4util_bts(int b, volatile l4_umword_t * dest)
00097 {
00098     l4_int8_t bit;
00099
00100     __asm__ __volatile__
00101     (
00102         "lock; btsl %2,%1  \n\t"
00103         :
00104         "=@ccc" (bit)    /* 0,      old bit value */
00105         :
00106         "m"    (*dest),    /* 1 mem, destination operand */
00107         "Ir"   (b)         /* 2,      bit number */
00108         :
00109         "memory"
00110         );
00111
00112     return bit;
00113 }
00114
00115 /* bit test and reset */
00116 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_RESET
00117 L4_INLINE int
00118 l4util_btr(int b, volatile l4_umword_t * dest)
00119 {
00120     l4_int8_t bit;
00121
00122     __asm__ __volatile__
00123     (
00124         "lock; btrl %2,%1  \n\t"
00125         :
00126         "=@ccc" (bit)    /* 0,      old bit value */
00127         :
00128         "m"    (*dest),    /* 1 mem, destination operand */
00129         "Ir"   (b)         /* 2,      bit number */
00130         :

```

```

00131     "memory"
00132     );
00133
00134     return bit;
00135 }
00136
00137 /* bit test and complement */
00138 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_COMPLEMENT
00139 L4_INLINE int
00140 l4util_btc(int b, volatile l4_umword_t * dest)
00141 {
00142     l4_int8_t bit;
00143
00144     __asm__ __volatile__
00145     (
00146         "lock; btl %2,%1 \n\t"
00147         :
00148         "=@ccc" (bit) /* 0, old bit value */
00149         :
00150         "m" (*dest), /* 1 mem, destination operand */
00151         "Ir" (b) /* 2, bit number */
00152         :
00153         "memory"
00154         );
00155
00156     return bit;
00157 }
00158
00159 /* bit scan reverse */
00160 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_REVERSE
00161 L4_INLINE int
00162 l4util_bsr(l4_umword_t word)
00163 {
00164     int tmp;
00165
00166     if (L4_UNLIKELY(word == 0))
00167         return -1;
00168
00169     __asm__ __volatile__
00170     (
00171         "bsrl %1,%0 \n\t"
00172         :
00173         "=r" (tmp) /* 0, index of most significant set bit */
00174         :
00175         "r" (word) /* 1, argument */
00176         );
00177
00178     return tmp;
00179 }
00180
00181 /* bit scan forward */
00182 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_FORWARD
00183 L4_INLINE int
00184 l4util_bsf(l4_umword_t word)
00185 {
00186     int tmp;
00187
00188     if (L4_UNLIKELY(word == 0))
00189         return -1;
00190
00191     __asm__ __volatile__
00192     (
00193         "bsfl %1,%0 \n\t"
00194         :
00195         "=r" (tmp) /* 0, index of least significant set bit */
00196         :
00197         "r" (word) /* 1, argument */
00198         );
00199
00200     return tmp;
00201 }
00202
00203 #define __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_SET_BIT
00204 L4_INLINE int
00205 l4util_find_first_set_bit(const void * dest, l4_size_t size)
00206 {
00207     l4_mword_t dummy0, dummy1, res;
00208
00209     __asm__ __volatile__
00210     (
00211         "repe; scasl \n\t"
00212         "jz 1f \n\t"
00213         "lea -4(%edi),%edi \n\t"
00214         "bsf (%edi),%eax \n\t"
00215         "1: \n\t"
00216         "sub %esi,%edi \n\t"
00217         "shl $3,%edi \n\t"

```

```

00218     "add  %%edi,%%eax          \n\t"
00219     :
00220     "a" (res), "c" (dummy0), "=D" (dummy1)
00221     :
00222     "a" (0), "c" ((size + 31) >> 5), "D" (dest), "S" (dest)
00223     :
00224     "cc", "memory");
00225
00226     return res;
00227 }
00228
00229 #define __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_ZERO_BIT
00230 L4_INLINE int
00231 l4util_find_first_zero_bit(const void * dest, l4_size_t size)
00232 {
00233     l4_mword_t dummy0, dummy1, dummy2, res;
00234
00235     if (!size)
00236         return 0;
00237
00238     __asm__ __volatile__
00239     (
00240         "repe;  scasl          \n\t"
00241         "je     lf            \n\t"
00242         "xor    -4(%%edi),%%eax \n\t"
00243         "sub    $4,%%edi       \n\t"
00244         "bsf    %%eax,%%edx     \n\t"
00245         "l:      \n\t"
00246         "sub    %%esi,%%edi     \n\t"
00247         "shl    $3,%%edi        \n\t"
00248         "add    %%edi,%%edx     \n\t"
00249         :
00250         "a" (dummy0), "c" (dummy1), "=d" (res), "=D" (dummy2)
00251         :
00252         "a" (~0UL), "c" ((size + 31) >> 5), "d" (0), "D" (dest), "S" (dest)
00253         :
00254         "cc", "memory");
00255
00256     return res;
00257 }
00258
00259 L4_END_DECLS

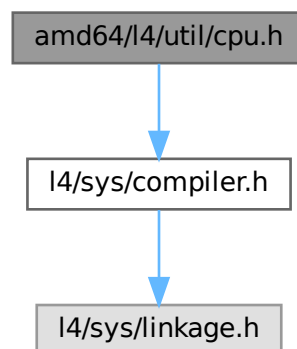
```

## 17.133 amd64/l4/util/cpu.h File Reference

CPU related functions.

```
#include <l4/sys/compiler.h>
```

Include dependency graph for cpu.h:



## Functions

- int [l4util\\_cpu\\_has\\_cpuid](#) (void)  
*Check whether the CPU supports the "cpuid" instruction.*
- unsigned int [l4util\\_cpu\\_capabilities](#) (void)  
*Returns the CPU capabilities if the "cpuid" instruction is available.*
- unsigned int [l4util\\_cpu\\_capabilities\\_nocheck](#) (void)  
*Returns the CPU capabilities.*
- void [l4util\\_cpu\\_cpuid](#) (unsigned long mode, unsigned long \*eax, unsigned long \*ebx, unsigned long \*ecx, unsigned long \*edx)  
*Generic CPUID access function.*

### 17.133.1 Detailed Description

CPU related functions.

#### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [cpu.h](#).

## 17.134 cpu.h

[Go to the documentation of this file.](#)

```

00001
00006
00007 /*
00008  * (c) 2004-2009 Author(s)
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012
00013 #ifndef __L4_UTIL_CPU_H
00014 #define __L4_UTIL_CPU_H
00015
00016 #include <l4/sys/compiler.h>
00017
00018 L4_BEGIN_DECLS
00019
00025
00031 L4_INLINE int      l4util_cpu_has_cpuid(void);
00032
00039 L4_INLINE unsigned int l4util_cpu_capabilities(void);
00040
00046 L4_INLINE unsigned int l4util_cpu_capabilities_nocheck(void);
00047
00051 L4_INLINE void
00052 l4util_cpu_cpuid(unsigned long mode,
00053                 unsigned long *eax, unsigned long *ebx,
00054                 unsigned long *ecx, unsigned long *edx);
00055
00057 static inline void
00058 l4util_cpu_pause(void)
00059 {
00060     __asm__ __volatile__ ("rep; nop");
00061 }
00062
00063 L4_INLINE int
00064 l4util_cpu_has_cpuid(void)
00065 {
00066     return 1;
00067 }
00068
00069 L4_INLINE void

```

```

00070 l4util_cpu_cpuid(unsigned long mode,
00071                  unsigned long *eax, unsigned long *ebx,
00072                  unsigned long *ecx, unsigned long *edx)
00073 {
00074     asm volatile("cpuid"
00075                 : "=a" (*eax),
00076                   "=b" (*ebx),
00077                   "=c" (*ecx),
00078                   "=d" (*edx)
00079                 : "a" (mode)
00080                 );
00081 }
00082
00083 L4_INLINE unsigned int
00084 l4util_cpu_capabilities_nocheck(void)
00085 {
00086     unsigned long dummy, capability;
00087
00088     /* get CPU capabilities */
00089     l4util_cpu_cpuid(1, &dummy, &dummy, &dummy, &capability);
00090
00091     return capability;
00092 }
00093
00094 L4_INLINE unsigned int
00095 l4util_cpu_capabilities(void)
00096 {
00097     if (!l4util_cpu_has_cpuid())
00098         return 0; /* CPU has not cpuid instruction */
00099
00100     return l4util_cpu_capabilities_nocheck();
00101 }
00102
00103 L4_END_DECLS
00104
00105 #endif
00106

```

## 17.135 arm/l4/util/cpu.h File Reference

CPU related functions.

### 17.135.1 Detailed Description

CPU related functions.

Author

Adam Lackorzynski [adam@os.inf.tu-dresden.de](mailto:adam@os.inf.tu-dresden.de)

Definition in file [cpu.h](#).

## 17.136 cpu.h

[Go to the documentation of this file.](#)

```

00001
00002
00003 /*
00004  * (c) 2004-2009 Author(s)
00005  *     economic rights: Technische Universität Dresden (Germany)
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #ifndef __L4_UTIL__ARCH_ARM__CPU_H__
00010 #define __L4_UTIL__ARCH_ARM__CPU_H__
00011
00012 /* Nothing yet */
00013
00014 #endif /* __L4_UTIL__ARCH_ARM__CPU_H__ */

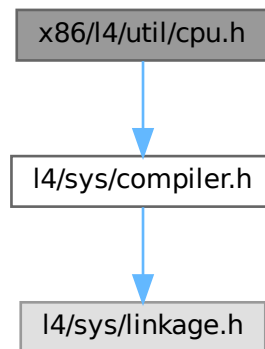
```

## 17.137 x86/l4/util/cpu.h File Reference

CPU related functions.

```
#include <l4/sys/compiler.h>
```

Include dependency graph for cpu.h:



### Functions

- int [l4util\\_cpu\\_has\\_cpuid](#) (void)  
*Check whether the CPU supports the "cpuid" instruction.*
- unsigned int [l4util\\_cpu\\_capabilities](#) (void)  
*Returns the CPU capabilities if the "cpuid" instruction is available.*
- unsigned int [l4util\\_cpu\\_capabilities\\_nocheck](#) (void)  
*Returns the CPU capabilities.*
- void [l4util\\_cpu\\_cpuid](#) (unsigned long mode, unsigned long \*eax, unsigned long \*ebx, unsigned long \*ecx, unsigned long \*edx)  
*Generic CPUID access function.*

### 17.137.1 Detailed Description

CPU related functions.

#### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [cpu.h](#).

## 17.138 cpu.h

[Go to the documentation of this file.](#)

```

00001
00006
00007 /*
00008  * (c) 2004-2009 Author(s)
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012
00013 #ifndef __L4_UTIL_CPU_H
00014 #define __L4_UTIL_CPU_H
00015
00016 #include <l4/sys/compiler.h>
00017
00018 L4_BEGIN_DECLS
00019
00025
00031 L4_INLINE int          l4util_cpu_has_cpuid(void);
00032
00039 L4_INLINE unsigned int l4util_cpu_capabilities(void);
00040
00046 L4_INLINE unsigned int l4util_cpu_capabilities_nocheck(void);
00047
00051 L4_INLINE void
00052 l4util_cpu_cpuid(unsigned long mode,
00053                 unsigned long *eax, unsigned long *ebx,
00054                 unsigned long *ecx, unsigned long *edx);
00055
00057 static inline void
00058 l4util_cpu_pause(void)
00059 {
00060     __asm__ __volatile__ ("rep; nop");
00061 }
00062
00063 L4_INLINE int
00064 l4util_cpu_has_cpuid(void)
00065 {
00066     unsigned long eax;
00067
00068     asm volatile("pushl %%ebx          \t\n"
00069                 "pushfl          \t\n"
00070                 "popl %%eax          \t\n" /* get eflags */
00071                 "movl %%eax, %%ebx \t\n" /* save it */
00072                 "xorl $0x200000, %%eax \t\n" /* toggle ID bit */
00073                 "pushl %%eax          \t\n"
00074                 "popfl          \t\n" /* set again */
00075                 "pushfl          \t\n"
00076                 "popl %%eax          \t\n" /* get it again */
00077                 "xorl %%ebx, %%eax \t\n"
00078                 "pushl %%ebx          \t\n"
00079                 "popfl          \t\n" /* restore saved flags */
00080                 "popl %%ebx          \t\n"
00081                 : "=a" (eax)
00082                 : /* no input */
00083                 );
00084
00085     return eax & 0x200000;
00086 }
00087
00088 L4_INLINE void
00089 l4util_cpu_cpuid(unsigned long mode,
00090                 unsigned long *eax, unsigned long *ebx,
00091                 unsigned long *ecx, unsigned long *edx)
00092 {
00093     asm volatile("pushl %%ebx          \t\n"
00094                 "cpuid          \t\n"
00095                 "mov %%ebx, %%esi \t\n"
00096                 "popl %%ebx          \t\n"
00097                 : "=a" (*eax),
00098                   "=S" (*ebx),
00099                   "=c" (*ecx),
00100                   "=d" (*edx)
00101                 : "a" (mode));
00102 }
00103
00104 L4_INLINE unsigned int
00105 l4util_cpu_capabilities_nocheck(void)
00106 {
00107     unsigned long dummy, capability;
00108
00109     /* get CPU capabilities */
00110     l4util_cpu_cpuid(1, &dummy, &dummy, &dummy, &capability);
00111

```



```

00112     return capability;
00113 }
00114
00115 L4_INLINE unsigned int
00116 l4util_cpu_capabilities(void)
00117 {
00118     if (!l4util_cpu_has_cpuid())
00119         return 0; /* CPU has not cpuid instruction */
00120
00121     return l4util_cpu_capabilities_nocheck();
00122 }
00123
00124 L4_END_DECLS
00125
00126 #endif
00127

```

## 17.139 amd64/l4/util/l4\_macros.h File Reference

Main function.

### 17.139.1 Detailed Description

Main function.

#### Date

08/29/2000

#### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [l4\\_macros.h](#).

## 17.140 l4\_macros.h

[Go to the documentation of this file.](#)

```

00001
00007
00008 /*
00009  * (c) 2006-2009 Author(s)
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013
00014 #ifndef _L4UTIL__ARCH_AMD64__L4_MACROS_H
00015 #define _L4UTIL__ARCH_AMD64__L4_MACROS_H
00016
00017 #include_next <l4/util/l4_macros.h>
00018
00019 #ifndef l4_addr_fmt
00020 #   define l4_addr_fmt    "%016lx"
00021 #endif
00022
00023 #endif /* !_L4UTIL__ARCH_AMD64__L4_MACROS_H */

```

## 17.141 arm/l4/util/l4\_macros.h File Reference

Main function.

### 17.141.1 Detailed Description

Main function.

#### Date

08/29/2000

#### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [l4\\_macros.h](#).

## 17.142 l4\_macros.h

[Go to the documentation of this file.](#)

```
00001
00007
00008 /*
00009  * (c) 2006-2009 Author(s)
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013
00014 #ifndef _L4UTIL__ARCH_ARM__L4_MACROS_H
00015 #define _L4UTIL__ARCH_ARM__L4_MACROS_H
00016
00017 #include_next <l4/util/l4_macros.h>
00018
00019 #ifndef l4_addr_fmt
00020 #   define l4_addr_fmt    "%08lx"
00021 #endif
00022
00023 #endif /* !_L4UTIL__ARCH_ARM__L4_MACROS_H */
```

## 17.143 l4/util/l4\_macros.h File Reference

Some useful generic macros, L4f version.

### 17.143.1 Detailed Description

Some useful generic macros, L4f version.

#### Date

11/12/2002

#### Author

Lars Reuther [reuther@os.inf.tu-dresden.de](mailto:reuther@os.inf.tu-dresden.de)

Definition in file [l4\\_macros.h](#).

## 17.144 l4\_macros.h

[Go to the documentation of this file.](#)

```

00001 /*****
00008 */
00009 * (c) 2000-2009 Author(s)
00010 *     economic rights: Technische Universität Dresden (Germany)
00011 * License: see LICENSE.spdx (in this directory or the directories above)
00012 */
00013
00014 /*****
00015
00016 #pragma once
00017
00018 /*****
00019 *** generic macros
00020 *****/
00021
00022 /* generate L4 thread id printf string */
00023 #ifndef l4util_idstr
00024 #   define l4util_idfmt          "%lx"
00025 #   define l4util_idfmt_adjust   "%04lx"
00026 #   define l4util_idstr(tid)     (tid » L4_CAP_SHIFT)
00027 #endif
00028

```

## 17.145 x86/l4/util/l4\_macros.h File Reference

Main function.

### 17.145.1 Detailed Description

Main function.

#### Date

08/29/2000

#### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [l4\\_macros.h](#).

## 17.146 l4\_macros.h

[Go to the documentation of this file.](#)

```

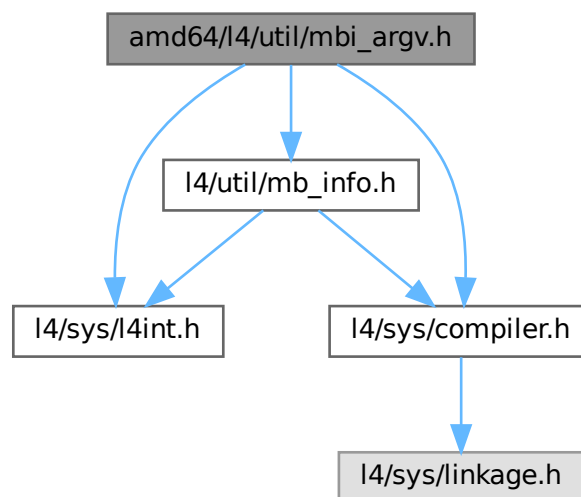
00001
00007
00008 /*
00009 * (c) 2006-2009 Author(s)
00010 *     economic rights: Technische Universität Dresden (Germany)
00011 * License: see LICENSE.spdx (in this directory or the directories above)
00012 */
00013
00014
00015 #ifndef _L4UTIL__ARCH_X86__L4_MACROS_H
00016 #define _L4UTIL__ARCH_X86__L4_MACROS_H
00017
00018 #include_next <l4/util/l4_macros.h>
00019
00020 #ifndef l4_addr_fmt
00021 #   define l4_addr_fmt          "%08lx"
00022 #endif
00023
00024 #endif /* !_L4UTIL__ARCH_X86__L4_MACROS_H */

```

## 17.147 amd64/l4/util/mbi\_argv.h File Reference

command line handling

```
#include <l4/sys/l4int.h>
#include <l4/util/mb_info.h>
#include <l4/sys/compiler.h>
Include dependency graph for mbi_argv.h:
```



### 17.147.1 Detailed Description

command line handling

Date

2003

Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [mbi\\_argv.h](#).

## 17.148 mbi\_argv.h

[Go to the documentation of this file.](#)

```

00001
00007
00008 /*
00009  * (c) 2003-2009 Author(s)
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013
00014 #ifndef L4UTIL_MBI_ARGV
00015 #define L4UTIL_MBI_ARGV
00016
00017 #include <l4/sys/l4int.h>
00018 #include <l4/util/mb_info.h>
00019 #include <l4/sys/compiler.h>
00020
00021 L4_BEGIN_DECLS
00022
00023 L4_CV void l4util_mbi_to_argv(l4_mword_t flag, l4util_mb_info_t *mbi);
00024
00025 extern int  l4util_argc;
00026 extern char *l4util_argv[];
00027
00028 L4_END_DECLS
00029
00030 #endif
00031

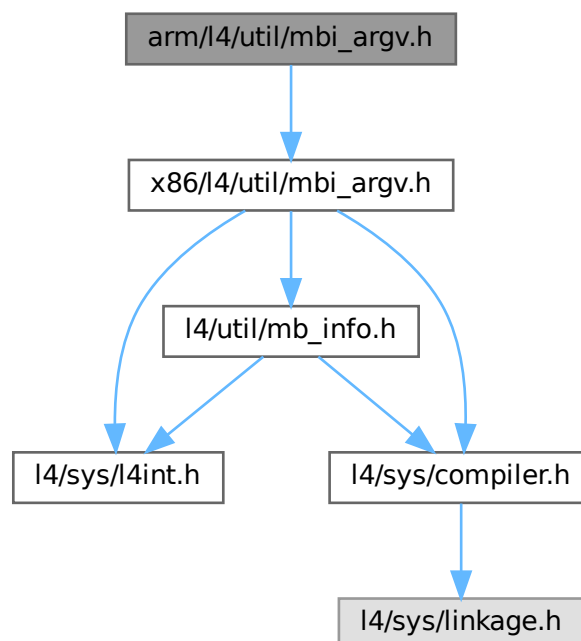
```

## 17.149 arm/l4/util/mbi\_argv.h File Reference

Multiboot.

```
#include <x86/l4/util/mbi_argv.h>
```

Include dependency graph for mbi\_argv.h:



### 17.149.1 Detailed Description

Multiboot.

Definition in file [mbi\\_argv.h](#).

## 17.150 mbi\_argv.h

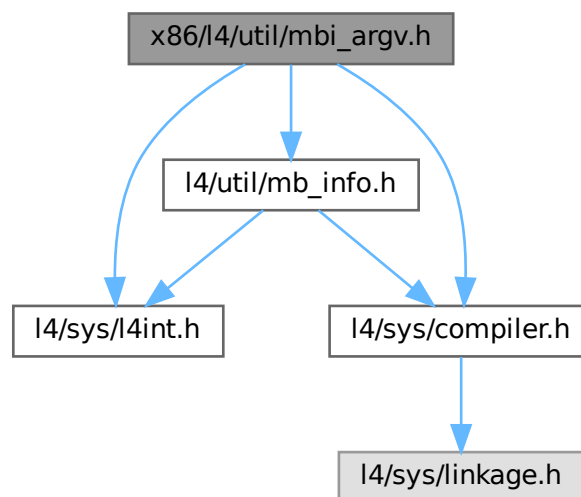
[Go to the documentation of this file.](#)

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010 /* If this persists, move it to a generic place */
00011 #include <x86/l4/util/mbi_argv.h>
```

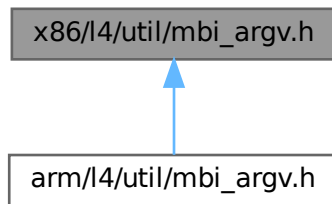
## 17.151 x86/l4/util/mbi\_argv.h File Reference

command line handling

```
#include <l4/sys/l4int.h>
#include <l4/util/mb_info.h>
#include <l4/sys/compiler.h>
Include dependency graph for mbi_argv.h:
```



This graph shows which files directly or indirectly include this file:



### 17.151.1 Detailed Description

command line handling

#### Date

2003

#### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [mbi\\_argv.h](#).

## 17.152 mbi\_argv.h

[Go to the documentation of this file.](#)

```

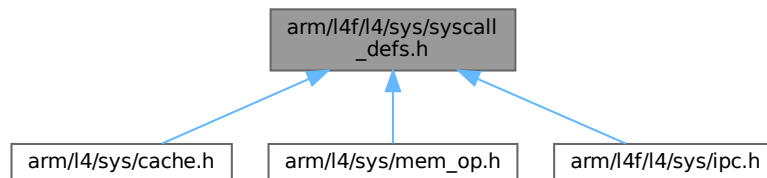
00001
00007
00008 /*
00009  * (c) 2003-2009 Author(s)
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013
00014 #ifndef L4UTIL_MBI_ARGV
00015 #define L4UTIL_MBI_ARGV
00016
00017 #include <l4/sys/l4int.h>
00018 #include <l4/util/mb_info.h>
00019 #include <l4/sys/compiler.h>
00020
00021 L4_BEGIN_DECLS
00022
00023 void l4util_mbi_to_argv(l4_mword_t flag, l4util_mb_info_t *mbi);
00024
00025 extern int l4util_argc;
00026 extern char *l4util_argv[];
00027
00028 L4_END_DECLS
00029
00030 #endif
00031

```

## 17.153 arm/l4f/l4/sys/syscall\_defs.h File Reference

Syscall entry definitions.

This graph shows which files directly or indirectly include this file:



### 17.153.1 Detailed Description

Syscall entry definitions.

Definition in file [syscall\\_defs.h](#).

## 17.154 syscall\_defs.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #ifndef __L4SYS__ARCH_ARM__L4API_L4F__SYSCALLL_DEFS_H__
00012 #define __L4SYS__ARCH_ARM__L4API_L4F__SYSCALLL_DEFS_H__
00013
00014 #define L4_SYSCALLL_INVOKE      (0)
00015 #define L4_SYSCALLL_MEM_OP      (1)
00016
00017 #endif /* __L4SYS__ARCH_ARM__L4API_L4F__SYSCALLL_DEFS_H__ */

```

## 17.155 contrib/libio-io/l4/io/io.h File Reference

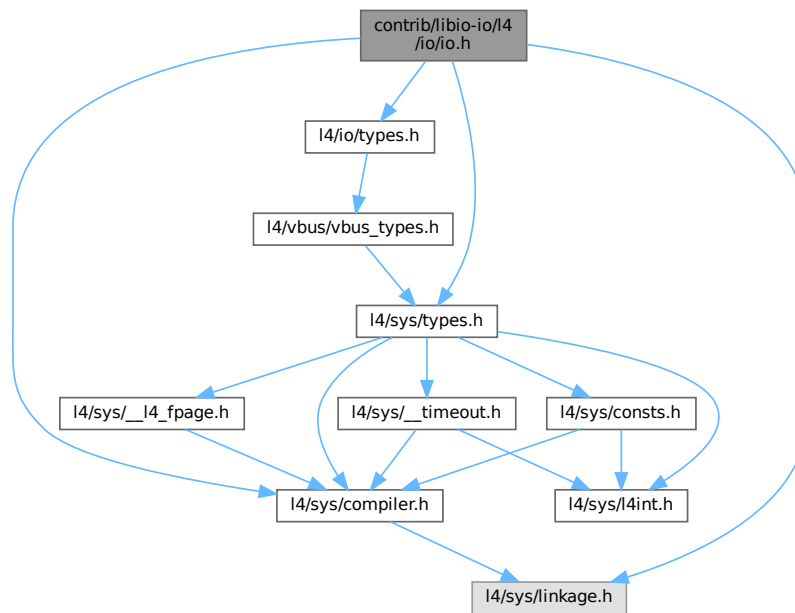
```

#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <l4/sys/linkage.h>

```



```
#include <l4/io/types.h>
Include dependency graph for io.h:
```



## Functions

- [l4\\_cap\\_idx\\_t l4io\\_request\\_icu](#) (void)  
*Request the ICU object of the client.*
- long [l4io\\_request\\_iomem](#) ([l4\\_addr\\_t](#) phys, unsigned long size, int flags, [l4\\_addr\\_t](#) \*virt)  
*Request an IO memory region.*
- long [l4io\\_request\\_iomem\\_region](#) ([l4\\_addr\\_t](#) phys, [l4\\_addr\\_t](#) virt, unsigned long size, int flags)  
*Request an IO memory region and map it to a specified region.*
- long [l4io\\_release\\_iomem](#) ([l4\\_addr\\_t](#) virt, unsigned long size)  
*Release an IO memory region.*
- long [l4io\\_request\\_ioport](#) (unsigned portnum, unsigned len)  
*Request an IO port region.*
- long [l4io\\_release\\_ioport](#) (unsigned portnum, unsigned len)  
*Release an IO port region.*
- [l4io\\_device\\_handle\\_t l4io\\_get\\_root\\_device](#) (void)  
*Get root device handle of the device bus.*
- int [l4io\\_iterate\\_devices](#) ([l4io\\_device\\_handle\\_t](#) \*devhandle, [l4io\\_device\\_t](#) \*dev, [l4io\\_resource\\_handle\\_t](#) \*reshandle)  
*Iterate over the device bus.*
- int [l4io\\_lookup\\_device](#) (const char \*devname, [l4io\\_device\\_handle\\_t](#) \*dev\_handle, [l4io\\_device\\_t](#) \*dev, [l4io\\_resource\\_handle\\_t](#) \*res\_handle)  
*Find a device by name.*
- int [l4io\\_lookup\\_resource](#) ([l4io\\_device\\_handle\\_t](#) devhandle, enum [l4io\\_resource\\_types\\_t](#) type, [l4io\\_resource\\_handle\\_t](#) \*reshandle, [l4io\\_resource\\_t](#) \*res)  
*Request a specific resource from a device description.*

- [l4\\_addr\\_t l4io\\_request\\_resource\\_iomem](#) ([l4io\\_device\\_handle\\_t](#) devhandle, [l4io\\_resource\\_handle\\_t](#) \*reshandle)  
*Request IO memory.*
- void [l4io\\_request\\_all\\_ioports](#) (void(\*res\_cb)([l4vbus\\_resource\\_t](#) const \*res))  
*Request all available IO-port resources.*
- int [l4io\\_has\\_resource](#) (enum [l4io\\_resource\\_types\\_t](#) type, [l4vbus\\_paddr\\_t](#) start, [l4vbus\\_paddr\\_t](#) end)  
*Check if a resource is available.*

## 17.155.1 Function Documentation

### 17.155.1.1 l4io\_get\_root\_device()

```
l4io_device_handle_t l4io_get_root_device (
    void ) [inline]
```

Get root device handle of the device bus.

#### Returns

root device handle

Definition at line 257 of file [io.h](#).

### 17.155.1.2 l4io\_iterate\_devices()

```
int l4io_iterate_devices (
    l4io_device_handle_t * devhandle,
    l4io_device_t * dev,
    l4io_resource_handle_t * reshandle)
```

Iterate over the device bus.

#### Parameters

in, out	<i>devhandle</i>	Device handle to start iterating at. The next device handle is returned here.
out	<i>dev</i>	Device information, may be NULL.
out	<i>reshandle</i>	Resource handle.

#### Returns

0 on success, error code otherwise

References [L4\\_CV](#), and [L4\\_EXPORT](#).

### 17.155.1.3 l4io\_request\_all\_ioports()

```
void l4io_request_all_ioports (
    void(* res_cb )(l4vbus\_resource\_t const *res))
```

Request all available IO-port resources.

#### Parameters

<code>res_cb</code>	Callback function called for every port resource found, give NULL for no callback.
---------------------	------------------------------------------------------------------------------------

References [L4\\_CV](#), and [L4\\_EXPORT](#).

#### 17.155.1.4 l4io\_request\_icu()

```
l4_cap_idx_t l4io_request_icu (
    void )
```

Request the ICU object of the client.

##### Returns

Client ICU object, an invalid capability selector is returned if no ICU is available.

References [L4\\_CV](#), and [L4\\_EXPORT](#).

## 17.156 io.h

[Go to the documentation of this file.](#)

```
00001
00004 /*
00005  * (c) 2008-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00006  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00007  *      economic rights: Technische Universität Dresden (Germany)
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010
00011
00012 #pragma once
00013
00014 #include <l4/sys/compiler.h>
00015 #include <l4/sys/types.h>
00016 #include <l4/sys/linkage.h>
00017 #include <l4/io/types.h>
00018
00019 L4_BEGIN_DECLS
00020
00024
00037 L4_CV long L4_EXPORT
00038 l4io_request_irq(int irqnum, l4_cap_idx_t irqcap);
00039
00046 L4_CV l4_cap_idx_t L4_EXPORT
00047 l4io_request_icu(void);
00048
00060 L4_CV long L4_EXPORT
00061 l4io_release_irq(int irqnum, l4_cap_idx_t irq_cap);
00062
00087 L4_CV long L4_EXPORT
00088 l4io_request_iomem(l4_addr_t phys, unsigned long size, int flags,
00089                  l4_addr_t *virt);
00090
00112 L4_CV long L4_EXPORT
00113 l4io_request_iomem_region(l4_addr_t phys, l4_addr_t virt,
00114                          unsigned long size, int flags);
00115
00123 L4_CV long L4_EXPORT
00124 l4io_release_iomem(l4_addr_t virt, unsigned long size);
00125
00135 L4_CV long L4_EXPORT
00136 l4io_request_ioport(unsigned portnum, unsigned len);
00137
00147 L4_CV long L4_EXPORT
00148 l4io_release_ioport(unsigned portnum, unsigned len);
00149
00150
00151 /* ----- Device handling ----- */
```

```

00152
00158 L4_INLINE
00159 l4io_device_handle_t l4io_get_root_device(void);
00160
00171 L4_CV int L4_EXPORT
00172 l4io_iterate_devices(l4io_device_handle_t *devhandle,
00173                     l4io_device_t *dev, l4io_resource_handle_t *reshandle);
00174
00187 L4_CV int L4_EXPORT
00188 l4io_lookup_device(const char *devname,
00189                  l4io_device_handle_t *dev_handle,
00190                  l4io_device_t *dev, l4io_resource_handle_t *res_handle);
00191
00207 L4_CV int L4_EXPORT
00208 l4io_lookup_resource(l4io_device_handle_t devhandle,
00209                     enum l4io_resource_types_t type,
00210                     l4io_resource_handle_t *reshandle,
00211                     l4io_resource_t *res);
00212
00213
00214 /* ----- Convenience functions ----- */
00215
00228 L4_CV l4_addr_t L4_EXPORT
00229 l4io_request_resource_iomem(l4io_device_handle_t devhandle,
00230                             l4io_resource_handle_t *reshandle);
00231
00238 L4_CV void L4_EXPORT
00239 l4io_request_all_ioports(void (*res_cb)(l4vbus_resource_t const *res));
00240
00249 L4_CV int L4_EXPORT
00250 l4io_has_resource(enum l4io_resource_types_t type,
00251                  l4vbus_paddr_t start, l4vbus_paddr_t end);
00252
00253 /* ----- */
00254 /* Implementations */
00255
00256 L4_INLINE
00257 l4io_device_handle_t l4io_get_root_device(void)
00258 { return 0; }
00259
00260 L4_END_DECLS

```

## 17.157 l4/cxx/alloc.h File Reference

Alloc list.

### Data Structures

- class [L4::Alloc\\_list](#)  
*A simple list-based allocator.*

### Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*

### 17.157.1 Detailed Description

Alloc list.

Definition in file [alloc.h](#).

## 17.158 alloc.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00007  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 namespace L4 {
00015
00020 class Alloc_list
00021 {
00022 public:
00023     Alloc_list() : _free(0) {}
00024     Alloc_list(void *blk, unsigned long size) : _free(0)
00025     { free( blk, size); }
00026
00027     void free(void *blk, unsigned long size);
00028     void *alloc(unsigned long size);
00029
00030 private:
00031     struct Elem
00032     {
00033         Elem *next;
00034         unsigned long size;
00035     };
00036
00037     Elem *_free;
00038 };
00039 }

```

## 17.159 arith

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 namespace cxx { namespace arith {
00012
00027 template<typename N, typename D>
00028 constexpr N
00029 div_ceil(N const &n, D const &d)
00030 {
00031     // Since C++11 the "quotient is truncated towards zero (fractional part is
00032     // discarded)". Thus a negative quotient is already ceiled, whereas a
00033     // positive quotient is floored. Furthermore, since C++11 the sign of the
00034     // % operator is no longer implementation defined, thus we can use n % d to
00035     // detect if the quotient is positive (n % d >= 0) and was truncated (n % d !=
00036     // 0). In that case, we add one to round to the next largest integer.
00037     return n / d + (n % d > 0);
00038 }
00039
00047 template< unsigned long V >
00048 struct Ld
00049 {
00050     enum { value = Ld<V / 2>::value + 1 };
00051 };
00052
00053 template<>
00054 struct Ld<0>
00055 {
00056     enum { value = ~0UL };
00057 };
00058
00059 template<>
00060 struct Ld<1>
00061 {
00062     enum { value = 0 };
00063 };

```

```

00064
00072 constexpr unsigned
00073 log2u(unsigned val)
00074 {
00075     return 8 * sizeof(val) - __builtin_clz(val) - 1;
00076 }
00077
00079 constexpr unsigned
00080 log2u(unsigned long val)
00081 {
00082     return 8 * sizeof(val) - __builtin_clzl(val) - 1;
00083 }
00084
00086 constexpr unsigned
00087 log2u(unsigned long long val)
00088 {
00089     return 8 * sizeof(val) - __builtin_clzll(val) - 1;
00090 }
00091
00100 template<typename T>
00101 constexpr unsigned
00102 log2u_ceil(T val)
00103 {
00104     return val == 1 ? 0 : log2u(val - 1) + 1;
00105 }
00106
00107 }

```

## 17.160 l4/cxx/avl\_map File Reference

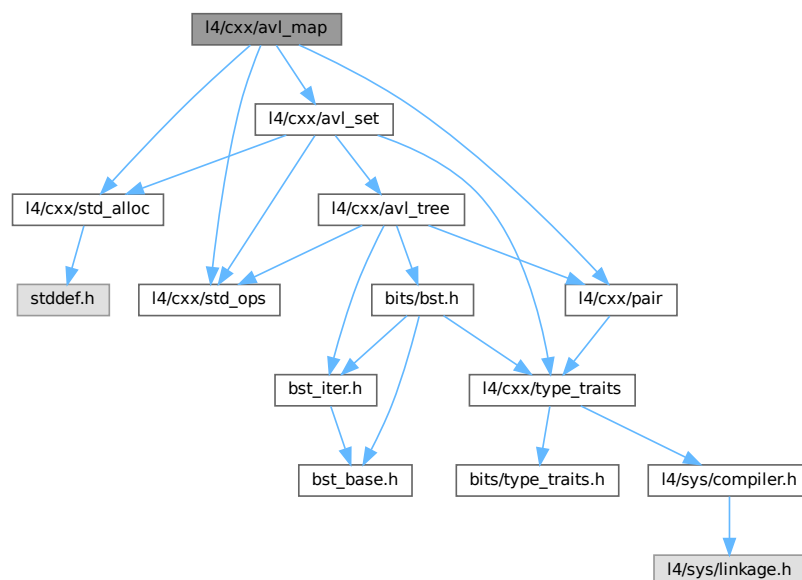
AVL map.

```

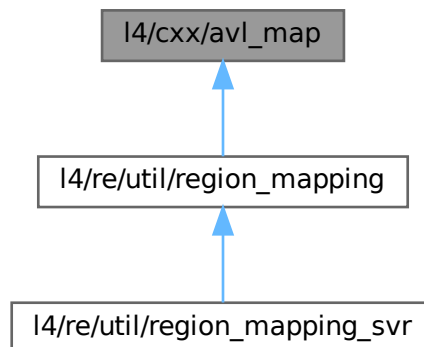
#include <l4/cxx/std_alloc>
#include <l4/cxx/std_ops>
#include <l4/cxx/pair>
#include <l4/cxx/avl_set>

```

Include dependency graph for avl\_map:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [cxx::Bits::Avl\\_map\\_get\\_key< KEY\\_TYPE >](#)  
*Key-getter for [Avl\\_map](#).*
- class [cxx::Avl\\_map< KEY\\_TYPE, DATA\\_TYPE, COMPARE, ALLOC >](#)  
*AVL tree based associative container.*

## Namespaces

- namespace [cxx](#)  
*Our C++ library.*
- namespace [cxx::Bits](#)  
*Internal helpers for the cxx package.*

## 17.160.1 Detailed Description

AVL map.

Definition in file [avl\\_map](#).

## 17.161 avl\_map

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -- Mode: C++ --
00006 /*
00007  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012

```

```

00013 #pragma once
00014
00015 #include <l4/cxx/std_alloc>
00016 #include <l4/cxx/std_ops>
00017 #include <l4/cxx/pair>
00018 #include <l4/cxx/avl_set>
00019
00020 namespace cxx {
00021 namespace Bits {
00022
00024 template<typename KEY_TYPE>
00025 struct Avl_map_get_key
00026 {
00027     typedef KEY_TYPE Key_type;
00028     template<typename NODE>
00029     static Key_type const &key_of(NODE const *n)
00030     { return n->item.first; }
00031 };
00032 }
00033
00042 template< typename KEY_TYPE, typename DATA_TYPE,
00043     template<typename A> class COMPARE = Lt_functor,
00044     template<typename B> class ALLOC = New_allocator >
00045 class Avl_map :
00046     public Bits::Base_avl_set<Pair<KEY_TYPE, DATA_TYPE>,
00047         COMPARE<KEY_TYPE>, ALLOC,
00048         Bits::Avl_map_get_key<KEY_TYPE> >
00049 {
00050 private:
00051     typedef Pair<KEY_TYPE, DATA_TYPE> Local_item_type;
00052     typedef Bits::Base_avl_set<Local_item_type, COMPARE<KEY_TYPE>, ALLOC,
00053         Bits::Avl_map_get_key<KEY_TYPE> > Base_type;
00054
00055 public:
00057     typedef COMPARE<KEY_TYPE> Key_compare;
00059     typedef KEY_TYPE Key_type;
00061     typedef DATA_TYPE Data_type;
00063     typedef typename Base_type::Node Node;
00065     typedef typename Base_type::Node_allocator Node_allocator;
00066
00067     typedef typename Base_type::Iterator Iterator;
00068     typedef typename Base_type::Iterator iterator;
00069     typedef typename Base_type::Const_iterator Const_iterator;
00070     typedef typename Base_type::Const_iterator const_iterator;
00071     typedef typename Base_type::Rev_iterator Rev_iterator;
00072     typedef typename Base_type::Rev_iterator reverse_iterator;
00073     typedef typename Base_type::Const_rev_iterator Const_rev_iterator;
00074     typedef typename Base_type::Const_rev_iterator const_reverse_iterator;
00075
00080     Avl_map(Node_allocator const &alloc = Node_allocator())
00081         : Base_type(alloc)
00082     {}
00083
00099     cxx::Pair<Iterator, int> insert(Key_type const &key, Data_type const &data)
00100     { return Base_type::insert(Pair<Key_type, Data_type>(key, data)); }
00101
00102     template<typename... Args>
00103     cxx::Pair<Iterator, int> emplace(Args &&...args)
00104     { return Base_type::emplace(cxx::forward<Args>(args)...); }
00105
00111     Data_type const &operator [] (Key_type const &key) const
00112     { return this->find_node(key)->second; }
00113
00123     Data_type &operator [] (Key_type const &key)
00124     {
00125         Node n = this->find_node(key);
00126         if (n)
00127             return const_cast<Data_type&>(n->second);
00128         else
00129             return insert(key, Data_type()).first->second;
00130     }
00131 };
00132
00133 }
00134

```

## 17.162 l4/cxx/avl\_set File Reference

AVL set.

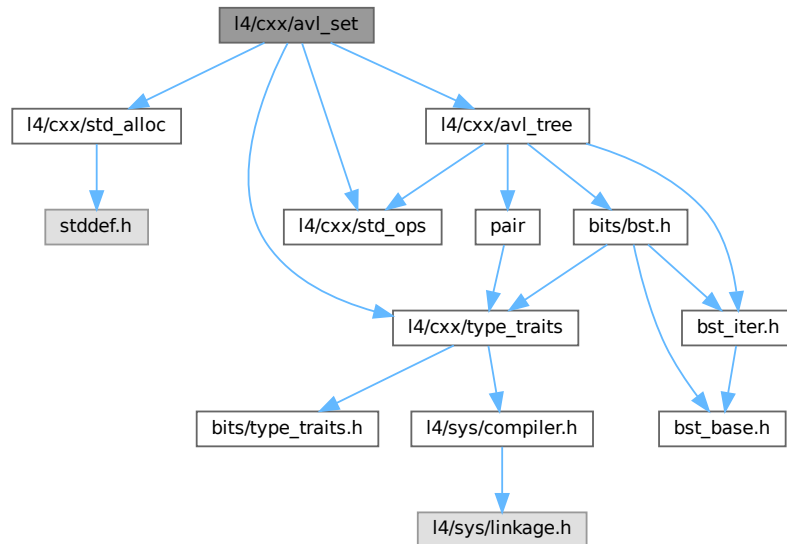
```

#include <l4/cxx/std_alloc>
#include <l4/cxx/std_ops>

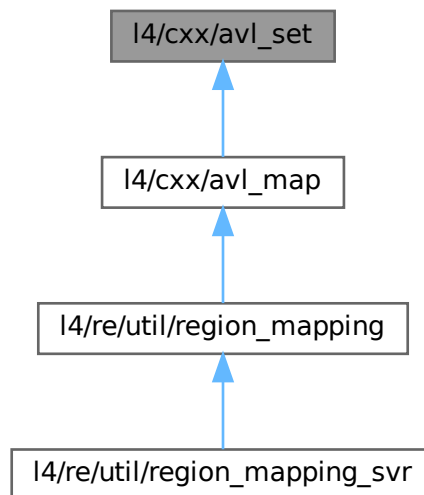
```



```
#include <l4/cxx/type_traits>
#include <l4/cxx/avl_tree>
Include dependency graph for avl_set:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `cxx::Bits::Avl_set_get_key< KEY_TYPE >`

*Internal, key-getter for [Avl\\_set](#) nodes.*

- class [cxx::Bits::Base\\_avl\\_set](#)< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >

*Internal: AVL set with internally managed nodes.*

- class [cxx::Bits::Base\\_avl\\_set](#)< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >::Node

*A smart pointer to a tree item.*

- class [cxx::Avl\\_set](#)< ITEM\_TYPE, COMPARE, ALLOC >

*AVL set for simple comparable items.*

## Namespaces

- namespace [cxx](#)

*Our C++ library.*

- namespace [cxx::Bits](#)

*Internal helpers for the [cxx](#) package.*

## 17.162.1 Detailed Description

AVL set.

Definition in file [avl\\_set](#).

## 17.163 avl\_set

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *      Carsten Weinhold <weinhold@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013
00014 #pragma once
00015
00016 #include <l4/cxx/std_alloc>
00017 #include <l4/cxx/std_ops>
00018 #include <l4/cxx/type_traits>
00019 #include <l4/cxx/avl_tree>
00020
00021 struct Avl_set_tester;
00022
00023 namespace cxx {
00024 namespace Bits {
00025 template< typename Node, typename Key, typename Node_op >
00026 class Avl_set_iter : public __Bst_iter_b<Node, Node_op>
00027 {
00028 private:
00029     typedef __Bst_iter_b<Node, Node_op> Base;
00030
00031     typedef typename Type_traits<Key>::Non_const_type Non_const_key;
00032
00033     typedef Avl_set_iter<Node, Non_const_key, Node_op> Non_const_iter;
00034
00035     using Base::_n;
00036     using Base::_r;
00037     using Base::inc;
00038
00039 public:
00040     Avl_set_iter() = default;
00041
00042     Avl_set_iter(Node const *t) : Base(t) {}
00043
00044     Avl_set_iter(Base const &o) : Base(o) {}
00045 }
```

```

00065
00067   Avl_set_iter(Non_const_iter const &o)
00068   : Base(o) {}
00069
00071   Avl_set_iter &operator = (Non_const_iter const &o)
00072   { Base::operator = (o); return *this; }
00073
00078   Key &operator * () const { return const_cast<Node*>(_n)->item; }
00083   Key *operator -> () const { return &const_cast<Node*>(_n)->item; }
00087   Avl_set_iter &operator ++ () { inc(); return *this; }
00091   Avl_set_iter operator ++ (int)
00092   { Avl_set_iter tmp = *this; inc(); return tmp; }
00093
00094 };
00095
00097 template<typename KEY_TYPE>
00098 struct Avl_set_get_key
00099 {
00100     typedef KEY_TYPE Key_type;
00101     template<typename NODE>
00102     static Key_type const &key_of(NODE const *n)
00103     { return n->item; }
00104 };
00105
00106
00119 template< typename ITEM_TYPE, class COMPARE,
00120           template<typename A> class ALLOC,
00121           typename GET_KEY>
00122 class Base_avl_set
00123 {
00124     friend struct ::Avl_set_tester;
00125
00126 public:
00133     enum
00134     {
00135         E_noent = 2,
00136         E_exist = 17,
00137         E_nomem = 12,
00138         E_inval = 22
00139     };
00141     typedef ITEM_TYPE Item_type;
00143     typedef GET_KEY Get_key;
00145     typedef typename GET_KEY::Key_type Key_type;
00147     typedef typename Type_traits<Item_type>::Const_type Const_item_type;
00149     typedef COMPARE Item_compare;
00150
00151 private:
00153     class _Node : public Avl_tree_node
00154     {
00155     public:
00157         Item_type item;
00158
00159         _Node() = default;
00160
00161         _Node(Item_type const &item) : Avl_tree_node(), item(item) {}
00162
00163         template<typename ...ARGS>
00164         _Node(ARGS &&...args) : Avl_tree_node(), item(cxx::forward<ARGS>(args)...)
00165         {}
00166     };
00167
00168 public:
00172     class Node
00173     {
00174     private:
00175         struct No_type;
00176         friend class Base_avl_set<ITEM_TYPE, COMPARE, ALLOC, GET_KEY>;
00177         _Node const *_n;
00178         explicit Node(_Node const *n) : _n(n) {}
00179
00180     public:
00182         Node() : _n(0) {}
00183
00189         Item_type const &operator * () { return _n->item; }
00195         Item_type const *operator -> () { return &_n->item; }
00196
00201         bool valid() const { return _n; }
00202
00204         operator Item_type const * () { return _n ? &_n->item : 0; }
00205     };
00206
00208     typedef ALLOC<_Node> Node_allocator;
00209
00210 private:
00211     typedef Avl_tree<_Node, GET_KEY, COMPARE> Tree;
00212     Tree _tree;
00214     Node_allocator _alloc;

```

```

00215
00216 Base_avl_set &operator = (Base_avl_set const &) = delete;
00217
00218 typedef typename Tree::Fwd_iter_ops Fwd;
00219 typedef typename Tree::Rev_iter_ops Rev;
00220
00221 public:
00222     typedef typename Type_traits<Item_type>::Param_type Item_param_type;
00223
00225     typedef Avl_set_iter<_Node, Item_type, Fwd> Iterator;
00226     typedef Iterator iterator;
00228     typedef Avl_set_iter<_Node, Const_item_type, Fwd> Const_iterator;
00229     typedef Const_iterator const_iterator;
00231     typedef Avl_set_iter<_Node, Item_type, Rev> Rev_iterator;
00232     typedef Rev_iterator reverse_iterator;
00234     typedef Avl_set_iter<_Node, Const_item_type, Rev> Const_rev_iterator;
00235     typedef Const_rev_iterator const_reverse_iterator;
00236
00243     explicit Base_avl_set(Node_allocator const &alloc = Node_allocator())
00244     : _tree(), _alloc(alloc)
00245     {}
00246
00247     ~Base_avl_set()
00248     {
00249         _tree.remove_all([this](_Node *n)
00250             {
00251                 n->~_Node();
00252                 _alloc.free(n);
00253             });
00254     }
00255
00263     inline Base_avl_set(Base_avl_set const &o);
00264
00282     cxx::Pair<Iterator, int> insert(Item_type const &item);
00283
00284     template<typename... Args>
00285     cxx::Pair<Iterator, int> emplace(Args&&... args);
00286
00295     int remove(Key_type const &item)
00296     {
00297         _Node *n = _tree.remove(item);
00298
00299         if (n)
00300         {
00301             n->~_Node();
00302             _alloc.free(n);
00303             return 0;
00304         }
00305
00306         return -E_noent;
00307     }
00308
00313     int erase(Key_type const &item)
00314     { return remove(item); }
00315
00324     Node find_node(Key_type const &item) const
00325     { return Node(_tree.find_node(item)); }
00326
00335     Node lower_bound_node(Key_type const &key) const
00336     { return Node(_tree.lower_bound_node(key)); }
00337
00338     Node lower_bound_node(Key_type &&key) const
00339     { return Node(_tree.lower_bound_node(key)); }
00340
00345     Const_iterator begin() const { return _tree.begin(); }
00350     Const_iterator end() const { return _tree.end(); }
00351
00356     Iterator begin() { return _tree.begin(); }
00361     Iterator end() { return _tree.end(); }
00362
00367     Const_rev_iterator rbegin() const { return _tree.rbegin(); }
00372     Const_rev_iterator rend() const { return _tree.rend(); }
00373
00378     Rev_iterator rbegin() { return _tree.rbegin(); }
00383     Rev_iterator rend() { return _tree.rend(); }
00384
00385     Const_iterator find(Key_type const &item) const
00386     { return _tree.find(item); }
00387
00388 #ifdef __DEBUG_L4_AVL
00389     bool rec_dump(bool print)
00390     {
00391         return _tree.rec_dump(print);
00392     }
00393 #endif
00394 };
00395

```

```

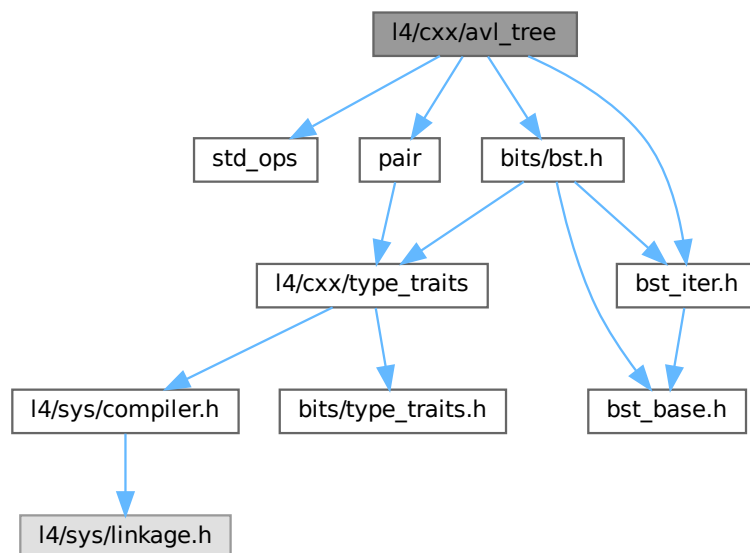
00396
00397 //-----
00398 /* Implementation of AVL Tree */
00399
00400 /* Create a copy */
00401 template< typename Item, class Compare, template<typename A> class Alloc, typename KEY_TYPE>
00402 Base_avl_set<Item,Compare,Alloc,KEY_TYPE>::Base_avl_set(Base_avl_set const &o)
00403 : _tree(), _alloc(o._alloc)
00404 {
00405     for (Const_iterator i = o.begin(); i != o.end(); ++i)
00406         insert(*i);
00407 }
00408
00409 /* Insert new _Node. */
00410 template< typename Item, class Compare, template< typename A > class Alloc, typename KEY_TYPE>
00411 Pair<typename Base_avl_set<Item,Compare,Alloc,KEY_TYPE>::Iterator, int>
00412 Base_avl_set<Item,Compare,Alloc,KEY_TYPE>::insert(Item const &item)
00413 {
00414     _Node *n = _alloc.alloc();
00415     if (!n)
00416         return cxx::pair(end(), -E_nomem);
00417
00418     new (n, Nothrow()) _Node(item);
00419     Pair<_Node *, bool> err = _tree.insert(n);
00420     if (!err.second)
00421     {
00422         n->~_Node();
00423         _alloc.free(n);
00424     }
00425
00426     return cxx::pair(Iterator(typename Tree::Iterator(err.first, err.first)), err.second ? 0 :
-E_exist);
00427 }
00428
00429 /* In-place insert new _Node. */
00430 template< typename Item, class Compare, template< typename A > class Alloc, typename KEY_TYPE>
00431 template<typename... Args>
00432 Pair<typename Base_avl_set<Item,Compare,Alloc,KEY_TYPE>::Iterator, int>
00433 Base_avl_set<Item,Compare,Alloc,KEY_TYPE>::emplace(Args&&... args)
00434 {
00435     _Node *n = _alloc.alloc();
00436     if (!n)
00437         return cxx::pair(end(), -E_nomem);
00438
00439     new (n, Nothrow()) _Node(cxx::forward<Args>(args)...);
00440     Pair<_Node *, bool> err = _tree.insert(n);
00441     if (!err.second)
00442     {
00443         n->~_Node();
00444         _alloc.free(n);
00445     }
00446
00447     return cxx::pair(Iterator(typename Tree::Iterator(err.first, err.first)), err.second ? 0 :
-E_exist);
00448 }
00449
00450 } // namespace Bits
00451
00463 template< typename ITEM_TYPE, class COMPARE = Lt_functor<ITEM_TYPE>,
00464           template<typename A> class ALLOC = New_allocator>
00465 class Avl_set :
00466     public Bits::Base_avl_set<ITEM_TYPE, COMPARE, ALLOC,
00467                               Bits::Avl_set_get_key<ITEM_TYPE> >
00468 {
00469 private:
00470     typedef Bits::Base_avl_set<ITEM_TYPE, COMPARE, ALLOC,
00471                               Bits::Avl_set_get_key<ITEM_TYPE> > Base;
00472
00473 public:
00474     typedef typename Base::Node_allocator Node_allocator;
00475     Avl_set() = default;
00476     Avl_set(Node_allocator const &alloc)
00477         : Base(alloc)
00478     {}
00479 };
00480
00481 } // namespace cxx

```

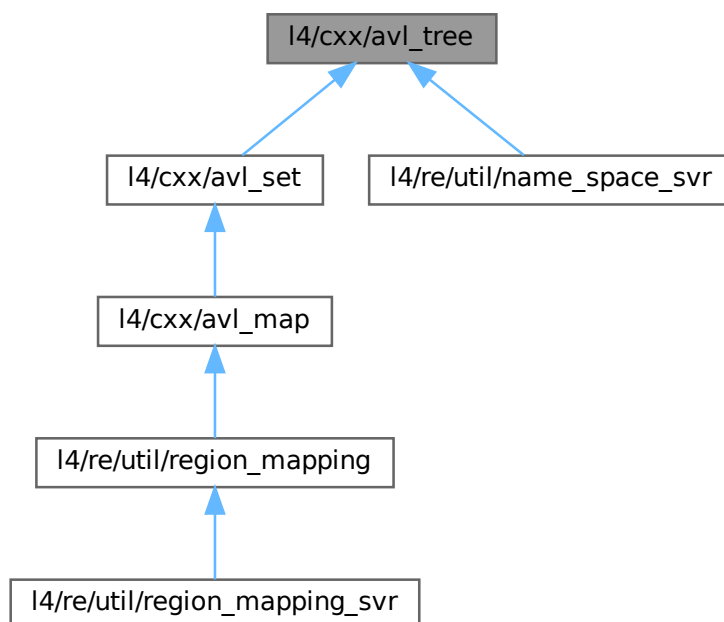
## 17.164 I4/cxx/avl\_tree File Reference

AVL tree.

```
#include "std_ops"
#include "pair"
#include "bits/bst.h"
#include "bits/bst_iter.h"
Include dependency graph for avl_tree:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `cxx::Avl_tree_node`  
*Node of an AVL tree.*
- class `cxx::Avl_tree< Node, Get_key, Compare >`  
*A generic AVL tree.*

## Namespaces

- namespace `cxx`  
*Our C++ library.*

### 17.164.1 Detailed Description

AVL tree.

Definition in file `avl_tree`.

## 17.165 avl\_tree

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *      Carsten Weinhold <weinhold@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013
00014 #pragma once
00015
00016 #include "std_ops"
00017 #include "pair"
00018
00019 #include "bits/bst.h"
00020 #include "bits/bst_iter.h"
00021
00022 struct Avl_set_tester;
00023
00024 namespace cxx {
00025
00026     class Avl_tree_node : public Bits::Bst_node
00027     {
00028     private:
00029         friend struct ::Avl_set_tester;
00030
00031     private:
00032         template< typename Node, typename Get_key, typename Compare >
00033         friend class Avl_tree;
00034
00035         typedef Bits::Direction Bal;
00036         typedef Bits::Direction Dir;
00037
00038         // We are a final BST node, hide interior.
00039         using Bits::Bst_node::next;
00040         using Bits::Bst_node::next_p;
00041         using Bits::Bst_node::rotate;
00042
00043         Bal _balance;
00044
00045     protected:
00046         Avl_tree_node() = default;
00047
00048     private:
00049         Avl_tree_node(Avl_tree_node const &o) = delete;
00050         Avl_tree_node(Avl_tree_node &&o) = delete;
00051
00052         Avl_tree_node &operator = (Avl_tree_node const &o) = default;
00053         Avl_tree_node &operator = (Avl_tree_node &&o) = default;
00054
00055         explicit Avl_tree_node(bool) : Bits::Bst_node(true), _balance(Dir::N) {}
00056
00057         static Bits::Bst_node *rotate2(Bst_node **t, Bal idir, Bal pre);
00058
00059         bool balanced() const { return _balance == Bal::N; }
00060
00061         Bal balance() const { return _balance; }
00062
00063         void balance(Bal b) { _balance = b; }
00064     };
00065
00066     template< typename Node, typename Get_key,
00067             typename Compare = Lt_functor<typename Get_key::Key_type> >
00068     class Avl_tree : public Bits::Bst<Node, Get_key, Compare>
00069     {
00070     private:
00071         typedef Bits::Bst<Node, Get_key, Compare> Bst;
00072
00073         using Bst::_head;
00074
00075         using Bst::k;
00076
00077         typedef typename Avl_tree_node::Bal Bal;
00078         typedef typename Avl_tree_node::Dir Dir;
00079
00080         Avl_tree(Avl_tree const &o) = delete;
00081         Avl_tree &operator = (Avl_tree const &o) = delete;
00082         Avl_tree(Avl_tree &&o) = delete;
00083         Avl_tree &operator = (Avl_tree &&o) = delete;
00084
00085     public:
00086         typedef typename Bst::Key_type Key_type;

```



```

00124     typedef typename Bst::Key_param_type Key_param_type;
00126
00127     // Grab iterator types from Bst
00130     typedef typename Bst::Iterator Iterator;
00132     typedef typename Bst::Const_iterator Const_iterator;
00134     typedef typename Bst::Rev_iterator Rev_iterator;
00136     typedef typename Bst::Const_rev_iterator Const_rev_iterator;
00138
00146     Pair<Node *, bool> insert(Node *new_node);
00147
00154     Node *remove(Key_param_type key);
00158     Node *erase(Key_param_type key) { return remove(key); }
00159
00161     Avl_tree() = default;
00162
00164     ~Avl_tree() noexcept
00165     {
00166         this->remove_all([](Node *){});
00167     }
00168
00169 #ifdef __DEBUG_L4_AVL
00170     bool rec_dump(Avl_tree_node *n, int depth, int *dp, bool print, char pfx);
00171     bool rec_dump(bool print)
00172     {
00173         int dp=0;
00174         return rec_dump(static_cast<Avl_tree_node *>(_head), 0, &dp, print, '+');
00175     }
00176 #endif
00177 };
00178
00179
00180 //-----
00181 /* IMPLEMENTATION: Bits::__Bst_iter_b */
00182
00183
00184 inline
00185 Bits::Bst_node *
00186 Avl_tree_node::rotate2(Bst_node **t, Bal idir, Bal pre)
00187 {
00188     typedef Bits::Bst_node N;
00189     typedef Avl_tree_node A;
00190     N *tmp[2] = { *t, N::next(*t, idir) };
00191     *t = N::next(tmp[1], !idir);
00192     A *n = static_cast<A*>(*t);
00193
00194     N::next(tmp[0], idir, N::next(n, !idir));
00195     N::next(tmp[1], !idir, N::next(n, idir));
00196     N::next(n, !idir, tmp[0]);
00197     N::next(n, idir, tmp[1]);
00198
00199     n->balance(Bal::N);
00200
00201     if (pre == Bal::N)
00202     {
00203         static_cast<A*>(tmp[0])->balance(Bal::N);
00204         static_cast<A*>(tmp[1])->balance(Bal::N);
00205         return 0;
00206     }
00207
00208     static_cast<A*>(tmp[pre != idir])->balance(!pre);
00209     static_cast<A*>(tmp[pre == idir])->balance(Bal::N);
00210
00211     return N::next(tmp[pre == idir], !pre);
00212 }
00213
00214 //-----
00215 /* Implementation of AVL Tree */
00216
00217 /* Insert new _Node. */
00218 template< typename Node, typename Get_key, class Compare>
00219 Pair<Node *, bool>
00220 Avl_tree<Node, Get_key, Compare>::insert(Node *new_node)
00221 {
00222     typedef Avl_tree_node A;
00223     typedef Bits::Bst_node N;
00224     N **t = &_head; /* search variable */
00225     N **s = &_head; /* node where rebalancing may occur */
00226     Key_param_type new_key = Get_key::key_of(new_node);
00227
00228     // search insertion point
00229     for (N *p; (p = *t);)
00230     {
00231         Dir b = this->dir(new_key, p);
00232         if (b == Dir::N)
00233             return pair(static_cast<Node*>(p), false);
00234
00235         if (!static_cast<A const *>(p)->balanced())

```

```

00236     s = t;
00237
00238     t = A::next_p(p, b);
00239 }
00240
00241 *static_cast<A*>(new_node) = A(true);
00242 *t = new_node;
00243
00244 N *n = *s;
00245 A *a = static_cast<A*>(n);
00246 if (!a->balanced())
00247 {
00248     A::Bal b(this->greater(new_key, n));
00249     if (a->balance() != b)
00250     {
00251         // ok we got in balance the shorter subtree go higher
00252         a->balance(Bal::N);
00253         // propagate the new balance down to the new node
00254         n = A::next(n, b);
00255     }
00256     else if (b == Bal(this->greater(new_key, A::next(n, b))))
00257     {
00258         // left-left or right-right case -> single rotation
00259         A::rotate(s, b);
00260         a->balance(Bal::N);
00261         static_cast<A*>(*s)->balance(Bal::N);
00262         n = A::next(*s, b);
00263     }
00264     else
00265     {
00266         // need a double rotation
00267         n = A::next(A::next(n, b), !b);
00268         n = A::rotate2(s, b, n == new_node ? Bal::N : Bal(this->greater(new_key, n)));
00269     }
00270 }
00271
00272 for (A::Bal b; n && n != new_node; static_cast<A*>(n)->balance(b), n = A::next(n, b))
00273     b = Bal(this->greater(new_key, n));
00274
00275 return pair(new_node, true);
00276 }
00277
00278
00279 /* remove an element */
00280 template< typename Node, typename Get_key, class Compare>
00281 inline
00282 Node *Avl_tree<Node, Get_key, Compare>::remove(Key_param_type key)
00283 {
00284     typedef Avl_tree_node A;
00285     typedef Bits::Bst_node N;
00286     N **q = &_head; /* search variable */
00287     N **s = &_head; /* last ('deepest') node on the search path to q
00288                      * with balance 0, at this place the rebalancing
00289                      * stops in any case */
00290     N **t = 0;
00291     Dir dir;
00292
00293     // find target node and rebalancing entry
00294     for (N *n; (n = *q); q = A::next_p(n, dir))
00295     {
00296         dir = Dir(this->greater(key, n));
00297         if (dir == Dir::L && !this->greater(k(n), key))
00298             /* found node */
00299             t = q;
00300
00301         if (!A::next(n, dir))
00302             break;
00303
00304         A const *a = static_cast<A const *>(n);
00305         if (a->balanced() || (a->balance() == !dir && A::next<A>(n, !dir)->balanced()))
00306             s = q;
00307     }
00308
00309     // nothing found
00310     if (!t)
00311         return 0;
00312
00313     A *i = static_cast<A*>(*t);
00314
00315     for (N *n; (n = *s); s = A::next_p(n, dir))
00316     {
00317         dir = Dir(this->greater(key, n));
00318
00319         if (!A::next(n, dir))
00320             break;
00321
00322         A *a = static_cast<A*>(n);

```

```

00323         // got one out of balance
00324         if (a->balanced())
00325             a->balance(!dir);
00326         else if (a->balance() == dir)
00327             a->balance(Bal::N);
00328         else
00329         {
00330             // we need rotations to get in balance
00331             Bal b = A::next<A>(n, !dir)->balance();
00332             if (b == dir)
00333                 A::rotate2(s, !dir, A::next<A>(A::next(n, !dir), dir)->balance());
00334             else
00335             {
00336                 A::rotate(s, !dir);
00337                 if (b != Bal::N)
00338                 {
00339                     a->balance(Bal::N);
00340                     static_cast<A*>(*s)->balance(Bal::N);
00341                 }
00342                 else
00343                 {
00344                     a->balance(!dir);
00345                     static_cast<A*>(*s)->balance(dir);
00346                 }
00347             }
00348             if (n == i)
00349                 t = A::next_p(*s, dir);
00350         }
00351     }
00352
00353     A *n = static_cast<A*>(*q);
00354     *t = n;
00355     *q = A::next(n, !dir);
00356     *n = *i;
00357
00358     return static_cast<Node*>(i);
00359 }
00360
00361 #ifdef __DEBUG_L4_AVL
00362 template< typename Node, typename Get_key, class Compare>
00363 bool Avl_tree<Node, Get_key, Compare>::rec_dump(Avl_tree_node *n, int depth, int *dp, bool print, char
pfx)
00364 {
00365     typedef Avl_tree_node A;
00366
00367     if (!n)
00368         return true;
00369
00370     int dpx[2] = {depth, depth};
00371     bool res = true;
00372
00373     res = rec_dump(A::next<A>(n, Dir::R), depth + 1, dpx + 1, print, '/');
00374
00375     if (print)
00376     {
00377         fprintf(stderr, "%2d: [%8p] b=%1d: ", depth, n, (int)n->balance().d);
00378
00379         for (int i = 0; i < depth; ++i)
00380             std::cerr << "    ";
00381
00382         std::cerr << pfx << (static_cast<Node*>(n)->item) << std::endl;
00383     }
00384
00385     res = res & rec_dump(A::next<A>(n, Dir::L), depth + 1, dpx, print, '\\');
00386
00387     int b = dpx[1] - dpx[0];
00388
00389     if (b < 0)
00390         *dp = dpx[0];
00391     else
00392         *dp = dpx[1];
00393
00394     Bal x = n->balance();
00395     if ((b < -1 || b > 1) ||
00396         (b == 0 && x != Bal::N) ||
00397         (b == -1 && x != Bal::L) ||
00398         (b == 1 && x != Bal::R))
00399     {
00400         if (print)
00401             fprintf(stderr, "%2d: [%8p] b=%1d: balance error %d\n", depth, n, (int)n->balance().d, b);
00402         return false;
00403     }
00404     return res;
00405 }
00406 #endif
00407
00408 }

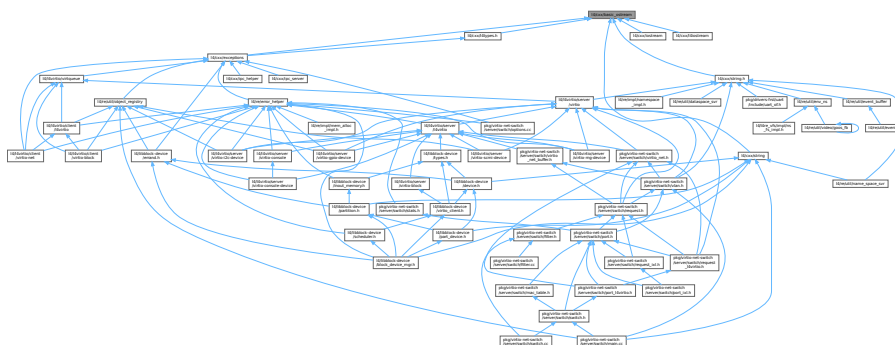
```

00409

## 17.166 l4/cxx/basic\_ostream File Reference

Basic IO stream.

This graph shows which files directly or indirectly include this file:



### Data Structures

- class [L4::IOModifier](#)  
*Modifier class for the IO stream.*

### Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*

### Variables

- [IOModifier](#) const [L4::hex](#)  
*Modifies the stream to print numbers as hexadecimal values.*
- [IOModifier](#) const [L4::dec](#)  
*Modifies the stream to print numbers as decimal values.*

### 17.166.1 Detailed Description

Basic IO stream.

Definition in file [basic\\_ostream](#).

## 17.167 basic\_ostream

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 namespace L4 {
00015
00022     class IOModifier
00023     {
00024     public:
00025         IOModifier(int x) : mod(x) {}
00026         bool operator == (IOModifier o) { return mod == o.mod; }
00027         bool operator != (IOModifier o) { return mod != o.mod; }
00028         int mod;
00029     };
00030
00035     class IOBackend
00036     {
00037     public:
00038         typedef int Mode;
00039
00040     protected:
00041         friend class BasicOStream;
00042
00043         IOBackend()
00044         : int_mode(10)
00045         {}
00046
00047         virtual ~IOBackend() {}
00048
00049         virtual void write(char const *str, unsigned len) = 0;
00050
00051     private:
00052         void write(IOModifier m);
00053         void write(long long int c, int len);
00054         void write(long long unsigned c, int len);
00055         void write(long long unsigned c, unsigned char base = 10,
00056                   unsigned char len = 0, char pad = ' ');
00057
00058         Mode mode() const
00059         { return int_mode; }
00060
00061         void mode(Mode m)
00062         { int_mode = m; }
00063
00064         int int_mode;
00065     };
00066
00071     class BasicOStream
00072     {
00073     public:
00074         BasicOStream(IOBackend *b)
00075         : iob(b)
00076         {}
00077
00078         void write(char const *str, unsigned len)
00079         {
00080             if (iob)
00081                 iob->write(str, len);
00082         }
00083
00084         void write(long long int c, int len)
00085         {
00086             if (iob)
00087                 iob->write(c, len);
00088         }
00089
00090         void write(long long unsigned c, unsigned char base = 10,
00091                   unsigned char len = 0, char pad = ' ')
00092         {
00093             if (iob)
00094                 iob->write(c, base, len, pad);
00095         }
00096
00097         void write(long long unsigned c, int len)
00098         {
00099             if (iob)
00100                 iob->write(c, len);

```

```

00101     }
00102
00103     void write(IOModifier m)
00104     {
00105         if (iob)
00106             iob->write(m);
00107     }
00108
00109     IOBackend::Mode be_mode() const
00110     {
00111         if (iob)
00112             return iob->mode();
00113         return 0;
00114     }
00115
00116     void be_mode(IOBackend::Mode m)
00117     {
00118         if (iob)
00119             iob->mode(m);
00120     }
00121
00122 private:
00123     IOBackend *iob;
00124 };
00125
00130 class IONumFmt
00131 {
00132 public:
00133     IONumFmt(unsigned long long n, unsigned char base = 10,
00134             unsigned char len = 0, char pad = ' ')
00135         : n(n), base(base), len(len), pad(pad)
00136     {}
00137
00138     BasicOStream &print(BasicOStream &o) const;
00139
00140 private:
00141     unsigned long long n;
00142     unsigned char base, len;
00143     char pad;
00144 };
00145
00146 inline IONumFmt n_hex(unsigned long long n) { return IONumFmt(n, 16); }
00147
00151 extern IOModifier const hex;
00152
00156 extern IOModifier const dec;
00157
00158 inline
00159 BasicOStream &IONumFmt::print(BasicOStream &o) const
00160 {
00161     o.write(n, base, len, pad);
00162     return o;
00163 }
00164 }
00165
00166 // Implementation
00167
00169 inline
00170 L4::BasicOStream &
00171 operator « (L4::BasicOStream &s, char const * const str)
00172 {
00173     if (!str)
00174     {
00175         s.write("(NULL)", 6);
00176         return s;
00177     }
00178
00179     unsigned l = 0;
00180     for (; str[l] != 0; l++)
00181         ;
00182     s.write(str, l);
00183     return s;
00184 }
00185
00186 inline
00187 L4::BasicOStream &
00188 operator « (L4::BasicOStream &s, signed short u)
00189 {
00190     s.write(static_cast<long long signed>(u), -1);
00191     return s;
00192 }
00193
00194 inline
00195 L4::BasicOStream &
00196 operator « (L4::BasicOStream &s, signed u)
00197 {

```

```

00198     s.write(static_cast<long long signed>(u), -1);
00199     return s;
00200 }
00201
00202 inline
00203 L4::BasicOStream &
00204 operator « (L4::BasicOStream &s, signed long u)
00205 {
00206     s.write(static_cast<long long signed>(u), -1);
00207     return s;
00208 }
00209
00210 inline
00211 L4::BasicOStream &
00212 operator « (L4::BasicOStream &s, signed long long u)
00213 {
00214     s.write(u, -1);
00215     return s;
00216 }
00217
00218 inline
00219 L4::BasicOStream &
00220 operator « (L4::BasicOStream &s, unsigned short u)
00221 {
00222     s.write(static_cast<long long unsigned>(u), -1);
00223     return s;
00224 }
00225
00226 inline
00227 L4::BasicOStream &
00228 operator « (L4::BasicOStream &s, unsigned u)
00229 {
00230     s.write(static_cast<long long unsigned>(u), -1);
00231     return s;
00232 }
00233
00234 inline
00235 L4::BasicOStream &
00236 operator « (L4::BasicOStream &s, unsigned long u)
00237 {
00238     s.write(static_cast<long long unsigned>(u), -1);
00239     return s;
00240 }
00241
00242 inline
00243 L4::BasicOStream &
00244 operator « (L4::BasicOStream &s, unsigned long long u)
00245 {
00246     s.write(u, -1);
00247     return s;
00248 }
00249
00250 inline
00251 L4::BasicOStream &
00252 operator « (L4::BasicOStream &s, void const *u)
00253 {
00254     long unsigned x = reinterpret_cast<long unsigned>(u);
00255     L4::IOBackend::Mode mode = s.be_mode();
00256     s.write(L4::hex);
00257     s.write(static_cast<long long unsigned>(x), -1);
00258     s.be_mode(mode);
00259     return s;
00260 }
00261
00262 inline
00263 L4::BasicOStream &
00264 operator « (L4::BasicOStream &s, L4::IOModifier m)
00265 {
00266     s.write(m);
00267     return s;
00268 }
00269
00270 inline
00271 L4::BasicOStream &
00272 operator « (L4::BasicOStream &s, char c)
00273 {
00274     s.write(&c, 1);
00275     return s;
00276 }
00277
00278 inline
00279 L4::BasicOStream &
00280 operator « (L4::BasicOStream &o, L4::IONumFmt const &n)
00281 { return n.print(o); }

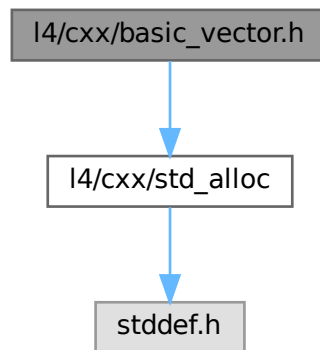
```

## 17.168 l4/cxx/basic\_vector.h File Reference

Basic vector.

```
#include <l4/cxx/std_alloc>
```

Include dependency graph for basic\_vector.h:



### Namespaces

- namespace `cxx`  
*Our C++ library.*

### 17.168.1 Detailed Description

Basic vector.

Definition in file [basic\\_vector.h](#).

## 17.169 basic\_vector.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00013 #include <l4/cxx/std_alloc>
00014
00015 namespace cxx {
00016
00017 template< typename T >
00018 class Basic_vector
00019 {

```



```

00020 public:
00021     Basic_vector(T *array, unsigned long capacity)
00022     : _array(array), _capacity(capacity)
00023     {
00024         for (unsigned long i = 0; i < capacity; ++i)
00025             new (&_array[i]) T();
00026     }
00027 private:
00028     T *_array;
00029     unsigned long _capacity;
00031 };
00032
00033 };

```

## 17.170 bitfield

```

00001 // vi:set ft=cpp: -- Mode: C++ --
00002 /*
00003  * (c) 2012 Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 #include "type_list"
00012
00014 namespace cxx {
00015
00023 template<typename T, unsigned LSB, unsigned MSB>
00024 class Bitfield
00025 {
00026 private:
00027     typedef remove_reference_t<T> Base_type;
00028
00029     static_assert(MSB >= LSB, "boundary mismatch in bit-field definition");
00030     static_assert(MSB < sizeof(Base_type) * 8, "MSB outside of bit-field type");
00031     static_assert(LSB < sizeof(Base_type) * 8, "LSB outside of bit-field type");
00032
00038     template<unsigned BITS> struct Best_type
00039     {
00040         template< typename TY > struct Cmp { enum { value = (BITS <= sizeof(TY)*8) }; };
00041         typedef cxx::type_list<
00042             unsigned char,
00043             unsigned short,
00044             unsigned int,
00045             unsigned long,
00046             unsigned long long
00047             > Unsigned_types;
00048         typedef cxx::find_type_t<Unsigned_types, Cmp> Type;
00049     };
00050
00051 public:
00052     enum
00053     {
00054         Bits = MSB + 1 - LSB,
00055         Lsb = LSB,
00056         Msb = MSB,
00057     };
00058
00060     enum Masks : Base_type
00061     {
00062         Low_mask = static_cast<Base_type>(~0ULL) >> (sizeof(Base_type)*8 - Bits),
00063         Mask = Low_mask << Lsb,
00064     };
00067
00074     typedef typename Best_type<Bits>::Type Bits_type;
00075
00082     typedef typename Best_type<Bits + Lsb>::Type Shift_type;
00083
00084 private:
00085     static_assert(sizeof(Bits_type)*8 >= Bits, "error finding the type to store the bits");
00086     static_assert(sizeof(Shift_type)*8 >= Bits + Lsb, "error finding the type to keep the shifted
bits");
00087     static_assert(sizeof(Bits_type) <= sizeof(Base_type), "size mismatch for Bits_type");
00088     static_assert(sizeof(Shift_type) <= sizeof(Base_type), "size mismatch for Shift_type");
00089     static_assert(sizeof(Bits_type) <= sizeof(Shift_type), "size mismatch for Shift_type and
Bits_type");
00090
00091 public:
00099     static constexpr Bits_type get(Shift_type val)

```

```

00100 { return (val >> Lsb) & Low_mask; }
00101
00112 static constexpr Base_type get_unshifted(Shift_type val)
00113 { return val & Mask; }
00114
00127 static constexpr Base_type set_dirty(Base_type dest, Shift_type val)
00128 {
00129     //assert (!(val & ~Low_mask));
00130     return (dest & ~Mask) | (val << Lsb);
00131 }
00132
00147 static constexpr Base_type set_unshifted_dirty(Base_type dest, Shift_type val)
00148 {
00149     //assert (!(val & ~Mask));
00150     return (dest & ~Mask) | val;
00151 }
00152
00161 static Base_type set(Base_type dest, Bits_type val)
00162 { return set_dirty(dest, val & Low_mask); }
00163
00173 static Base_type set_unshifted(Base_type dest, Shift_type val)
00174 { return set_unshifted_dirty(dest, val & Mask); }
00175
00187 static constexpr Base_type val_dirty(Shift_type val) { return val << Lsb; }
00188
00196 static constexpr Base_type val(Bits_type val) { return val_dirty(val & Low_mask); }
00197
00206 static constexpr Base_type val_unshifted(Shift_type val) { return val & Mask; }
00207
00209 template< typename TT >
00210 class Value_base
00211 {
00212 private:
00213     TT v;
00214
00215 public:
00216     constexpr Value_base(TT t) : v(t) {}
00217     constexpr Bits_type get() const { return Bitfield::get(v); }
00218     constexpr Base_type get_unshifted() const { return Bitfield::get_unshifted(v); }
00219
00220     void set(Bits_type val) { v = Bitfield::set(v, val); }
00221     void set_dirty(Bits_type val) { v = Bitfield::set_dirty(v, val); }
00222     void set_unshifted(Shift_type val) { v = Bitfield::set_unshifted(v, val); }
00223     void set_unshifted_dirty(Shift_type val) { v = Bitfield::set_unshifted_dirty(v, val); }
00224 };
00225
00227 template< typename TT >
00228 class Value : public Value_base<TT>
00229 {
00230 public:
00231     constexpr Value(TT t) : Value_base<TT>(t) {}
00232     constexpr operator Bits_type () const { return this->get(); }
00233     constexpr Value &operator = (Bits_type val) { this->set(val); return *this; }
00234     constexpr Value &operator = (Value const &val)
00235     { this->set(val.get()); return *this; }
00236     Value(Value const &) = default;
00237 };
00238
00240 template< typename TT >
00241 class Value_unshifted : public Value_base<TT>
00242 {
00243 public:
00244     constexpr Value_unshifted(TT t) : Value_base<TT>(t) {}
00245     constexpr operator Shift_type () const { return this->get_unshifted(); }
00246     constexpr Value_unshifted &operator = (Shift_type val) { this->set_unshifted(val); return *this; }
00247     constexpr Value_unshifted &operator = (Value_unshifted const &val)
00248     { this->set_unshifted(val.get_unshifted()); return *this; }
00249     Value_unshifted(Value_unshifted const &) = default;
00250 };
00251
00253 typedef Value<Base_type &> Ref;
00255 typedef Value<Base_type volatile &> Ref_volatile;
00257 typedef Value<Base_type const> Val;
00258
00260 typedef Value_unshifted<Base_type &> Ref_unshifted;
00262 typedef Value_unshifted<Base_type volatile &> Ref_unshifted_volatile;
00264 typedef Value_unshifted<Base_type const> Val_unshifted;
00265 };
00266
00267 #define CXX_BITFIELD_MEMBER(LSB, MSB, name, data_member) \
00268 \
00269 \
00270 typedef cxx::Bitfield<decltype(data_member), LSB, MSB> name ## _bfm_t; \
00271 \
00272 constexpr typename name ## _bfm_t::Val name() const { return data_member; } \
00273 typename name ## _bfm_t::Val name() const volatile { return data_member; } \
00274 \

```

```

00275     constexpr typename name ## _bfm_t::Ref name() { return data_member; } \
00276     typename name ## _bfm_t::Ref_volatile name() volatile { return data_member; } \
00277
00278
00279 #define CXX_BITFIELD_MEMBER_RO(LSB, MSB, name, data_member) \
00280 \
00281 \
00282     typedef cxx::Bitfield<decltype(data_member), LSB, MSB> name ## _bfm_t; \
00283 \
00284     constexpr typename name ## _bfm_t::Val name() const { return data_member; } \
00285     typename name ## _bfm_t::Val name() const volatile { return data_member; } \
00286
00287
00288 #define CXX_BITFIELD_MEMBER_UNSHIFTED(LSB, MSB, name, data_member) \
00289 \
00290 \
00291     typedef cxx::Bitfield<decltype(data_member), LSB, MSB> name ## _bfm_t; \
00292 \
00293     constexpr typename name ## _bfm_t::Val_unshifted name() const { return data_member; } \
00294     typename name ## _bfm_t::Val_unshifted name() const volatile { return data_member; } \
00295 \
00296     constexpr typename name ## _bfm_t::Ref_unshifted name() { return data_member; } \
00297     typename name ## _bfm_t::Ref_unshifted_volatile name() volatile { return data_member; } \
00298
00299
00300 #define CXX_BITFIELD_MEMBER_UNSHIFTED_RO(LSB, MSB, name, data_member) \
00301 \
00302 \
00303     typedef cxx::Bitfield<decltype(data_member), LSB, MSB> name ## _bfm_t; \
00304 \
00305     constexpr typename name ## _bfm_t::Val_unshifted name() const { return data_member; } \
00306     typename name ## _bfm_t::Val_unshifted name() const volatile { return data_member; } \
00307
00308 }

```

## 17.171 bitmap

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2014 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 namespace cxx {
00012
00013     class Bitmap_base
00014     {
00015     protected:
00016         typedef unsigned long word_type;
00017
00018         enum
00019         {
00020             W_bits = sizeof(word_type) * 8,
00021             C_bits = 8,
00022         };
00023
00024         word_type *_bits;
00025
00026         static unsigned word_index(unsigned bit) { return bit / W_bits; }
00027
00028         static unsigned bit_index(unsigned bit) { return bit % W_bits; }
00029
00030         class Bit
00031         {
00032         {
00033             Bitmap_base *_bm;
00034             long _bit;
00035
00036         public:
00037             Bit(Bitmap_base *bm, long bit) : _bm(bm), _bit(bit) {}
00038             Bit &operator = (bool val) { _bm->bit(_bit, val); return *this; }
00039             operator bool () const { return _bm->bit(_bit); }
00040         };
00041
00042     public:
00043         explicit Bitmap_base(void *bits) noexcept : _bits(reinterpret_cast<word_type *>(bits)) {}
00044
00045         static long words(long bits) noexcept { return (bits + W_bits - 1) / W_bits; }
00046         static long bit_buffer_bytes(long bits) noexcept
00047         { return words(bits) * W_bits / 8; }
00048     }
00049 }

```

```

00076
00077 template< long BITS >
00078 class Word
00079 {
00080 public:
00081     typedef unsigned long Type;
00082     enum
00083     {
00084         Size = (BITS + W_bits - 1) / W_bits
00085     };
00086 };
00087
00088
00089 static long chars(long bits) noexcept
00090 { return (bits + C_bits - 1) / C_bits; }
00091
00092
00093 template< long BITS >
00094 class Char
00095 {
00096 public:
00097     typedef unsigned char Type;
00098     enum
00099     {
00100         Size = (BITS + C_bits - 1) / C_bits
00101     };
00102 };
00103
00104
00105 void bit(long bit, bool on) noexcept;
00106
00107 void clear_bit(long bit) noexcept;
00108
00109 void atomic_clear_bit(long bit) noexcept;
00110
00111 word_type atomic_get_and_clear(long bit) noexcept;
00112
00113 void set_bit(long bit) noexcept;
00114
00115 void atomic_set_bit(long bit) noexcept;
00116
00117 word_type atomic_get_and_set(long bit) noexcept;
00118
00119 word_type bit(long bit) const noexcept;
00120
00121
00122 word_type operator [] (long bit) const noexcept
00123 { return this->bit(bit); }
00124
00125
00126 Bit operator [] (long bit) noexcept
00127 { return Bit(this, bit); }
00128
00129
00130 long scan_zero(long max_bit, long start_bit = 0) const noexcept;
00131
00132 void *bit_buffer() const noexcept { return _bits; }
00133
00134 protected:
00135     static int _bz1(unsigned long w) noexcept;
00136 };
00137
00138 template<int BITS>
00139 class Bitmap : public Bitmap_base
00140 {
00141 private:
00142     char _bits[Bitmap_base::Char<BITS>::Size];
00143 public:
00144     Bitmap() noexcept : Bitmap_base(_bits) {}
00145     Bitmap(Bitmap<BITS> const &o) noexcept : Bitmap_base(_bits)
00146     { __builtin_memcpy(_bits, o._bits, sizeof(_bits)); }
00147     long scan_zero(long start_bit = 0) const noexcept;
00148     void clear_all()
00149     { __builtin_memset(_bits, 0, sizeof(_bits)); }
00150 };
00151
00152 inline
00153 void
00154 Bitmap_base::bit(long bit, bool on) noexcept
00155 {
00156     long idx = word_index(bit);
00157     long b = bit_index(bit);
00158     _bits[idx] = (_bits[idx] & ~(1UL << b)) | (static_cast<unsigned long>(on) << b);
00159 }
00160
00161 inline
00162 void
00163 Bitmap_base::clear_bit(long bit) noexcept
00164 {

```

```

00262     long idx = word_index(bit);
00263     long b   = bit_index(bit);
00264     _bits[idx] &= ~(1UL << b);
00265 }
00266
00267 inline
00268 void
00269 Bitmap_base::atomic_clear_bit(long bit) noexcept
00270 {
00271     long idx = word_index(bit);
00272     long b   = bit_index(bit);
00273     word_type mask = 1UL << b;
00274     __atomic_and_fetch(&_bits[idx], ~mask, __ATOMIC_RELAXED);
00275 }
00276
00277 inline
00278 Bitmap_base::word_type
00279 Bitmap_base::atomic_get_and_clear(long bit) noexcept
00280 {
00281     long idx = word_index(bit);
00282     long b   = bit_index(bit);
00283     word_type mask = 1UL << b;
00284     return __atomic_fetch_and(&_bits[idx], ~mask, __ATOMIC_RELAXED) & mask;
00285 }
00286
00287 inline
00288 void
00289 Bitmap_base::set_bit(long bit) noexcept
00290 {
00291     long idx = word_index(bit);
00292     long b   = bit_index(bit);
00293     _bits[idx] |= (1UL << b);
00294 }
00295
00296 inline
00297 void
00298 Bitmap_base::atomic_set_bit(long bit) noexcept
00299 {
00300     long idx = word_index(bit);
00301     long b   = bit_index(bit);
00302     word_type mask = 1UL << b;
00303     __atomic_or_fetch(&_bits[idx], mask, __ATOMIC_RELAXED);
00304 }
00305
00306 inline
00307 Bitmap_base::word_type
00308 Bitmap_base::atomic_get_and_set(long bit) noexcept
00309 {
00310     long idx = word_index(bit);
00311     long b   = bit_index(bit);
00312     word_type mask = 1UL << b;
00313     return __atomic_fetch_or(&_bits[idx], mask, __ATOMIC_RELAXED) & mask;
00314 }
00315
00316 inline
00317 Bitmap_base::word_type
00318 Bitmap_base::bit(long bit) const noexcept
00319 {
00320     long idx = word_index(bit);
00321     long b   = bit_index(bit);
00322     return _bits[idx] & (1UL << b);
00323 }
00324
00325 inline
00326 int
00327 Bitmap_base::_bzl(unsigned long w) noexcept
00328 {
00329     for (int i = 0; i < W_bits; ++i, w >>= 1)
00330     {
00331         if ((w & 1) == 0)
00332             return i;
00333     }
00334     return -1;
00335 }
00336
00337 inline
00338 long
00339 Bitmap_base::scan_zero(long max_bit, long start_bit) const noexcept
00340 {
00341     if (!(operator [] (start_bit)))
00342         return start_bit;
00343
00344     long idx = word_index(start_bit);
00345     max_bit -= start_bit & ~(W_bits - 1);
00346
00347     for (; max_bit > 0; max_bit -= W_bits, ++idx)

```

```

00349     {
00350         if (_bits[idx] == 0)
00351             return idx * W_bits;
00352
00353         if (_bits[idx] != ~OUL)
00354         {
00355             long zbit = _bz1(_bits[idx]);
00356             return zbit < max_bit ? idx * W_bits + zbit : -1;
00357         }
00358     }
00359
00360     return -1;
00361 }
00362
00363 template<int BITS> inline
00364 long
00365 Bitmap<BITS>::scan_zero(long start_bit) const noexcept
00366 {
00367     return Bitmap_base::scan_zero(BITS, start_bit);
00368 }
00369
00370 };

```

## 17.172 I4/cxx/bits/bst.h File Reference

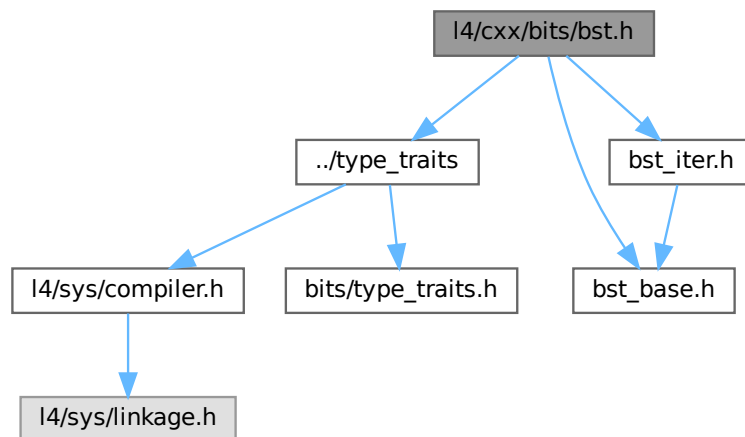
AVL tree.

```

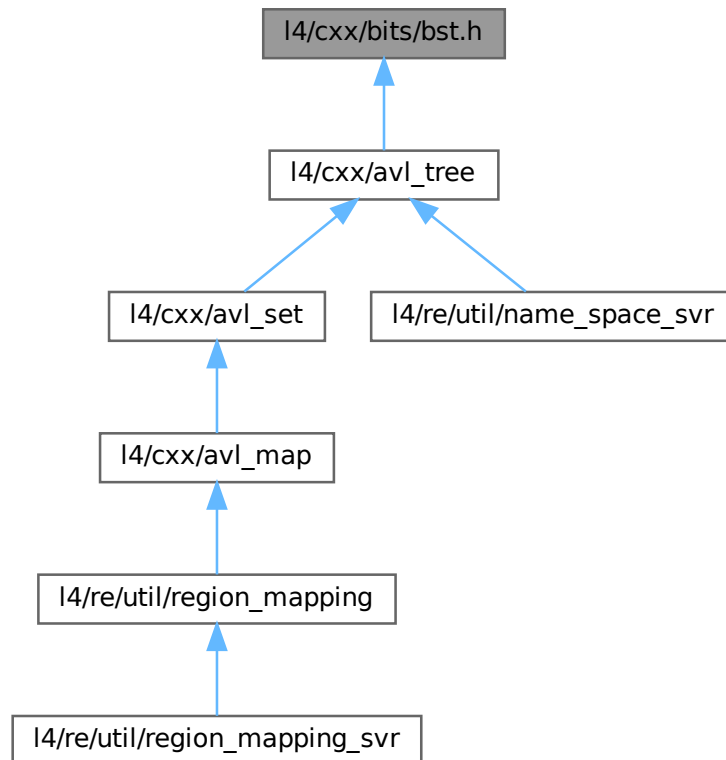
#include "../type_traits"
#include "bst_base.h"
#include "bst_iter.h"

```

Include dependency graph for bst.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [cxx::Bits::Bst< Node, Get\\_key, Compare >](#)  
*Basic binary search tree (BST).*

## Namespaces

- namespace [cxx](#)  
*Our C++ library.*
- namespace [cxx::Bits](#)  
*Internal helpers for the cxx package.*

### 17.172.1 Detailed Description

AVL tree.

Definition in file [bst.h](#).

## 17.173 bst.h

[Go to the documentation of this file.](#)

```

00001 // vi:ft=cpp
00006 /*
00007  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *      Carsten Weinhold <weinhold@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00015 #include "../type_traits"
00016 #include "bst_base.h"
00017 #include "bst_iter.h"
00018
00019 struct Avl_set_tester;
00020
00021 namespace cxx { namespace Bits {
00022
00023     template< typename Node, typename Get_key, typename Compare >
00031     class Bst
00032     {
00033     friend struct ::Avl_set_tester;
00034
00035     private:
00036         typedef Direction Dir;
00037         struct Fwd
00038         {
00039             static Node *child(Node const *n, Direction d)
00040             { return Bst_node::next<Node>(n, d); }
00041
00042             static bool cmp(Node const *l, Node const *r)
00043             { return Compare() (Get_key::key_of(l), Get_key::key_of(r)); }
00044         };
00045
00046         struct Rev
00047         {
00048             static Node *child(Node const *n, Direction d)
00049             { return Bst_node::next<Node>(n, !d); }
00050
00051             static bool cmp(Node const *l, Node const *r)
00052             { return Compare() (Get_key::key_of(r), Get_key::key_of(l)); }
00053         };
00054
00055     public:
00056         typedef typename Get_key::Key_type Key_type;
00057         typedef typename Type_traits<Key_type>::Param_type Key_param_type;
00058
00059         typedef Fwd Fwd_iter_ops;
00060         typedef Rev Rev_iter_ops;
00061
00062         typedef __Bst_iter<Node, Node, Fwd> Iterator;
00063         typedef __Bst_iter<Node, Node const, Fwd> Const_iterator;
00064         typedef __Bst_iter<Node, Node, Rev> Rev_iterator;
00065         typedef __Bst_iter<Node, Node const, Rev> Const_rev_iterator;
00066
00067     protected:
00068
00069         Bst_node *_head;
00070
00071         Bst() : _head(0) {}
00072
00073         Node *head() const { return static_cast<Node*>(_head); }
00074
00075         static Key_type k(Bst_node const *n)
00076         { return Get_key::key_of(static_cast<Node const *>(n)); }
00077
00078         static Dir dir(Key_param_type l, Key_param_type r)
00079         {
00080             Compare cmp;
00081             Dir d(cmp(r, l));
00082             if (d == Direction::L && !cmp(l, r))
00083                 return Direction::N;
00084             return d;
00085         }
00086
00087         static Dir dir(Key_param_type l, Bst_node const *r)
00088         { return dir(l, k(r)); }
00089
00090         static bool greater(Key_param_type l, Key_param_type r)
00091         { return Compare() (r, l); }
00092     };
00093
00094     } }
00095
00096     }
00097
00098     }
00099
00100     }
00101
00102     }
00103
00104     }
00105
00106     }
00107
00108     }
00109
00110     }
00111
00112     }
00113
00114     }
00115
00116     }
00117
00118     }
00119
00120     }
00121
00122     }
00123
00124     }
00125
00126     }
00127
00128     }
00129
00130     }
00131
00132     }
00133
00134     }
00135
00136     }
00137
00138     }
00139
00140     }
00141
00142     }
00143
00144     }
00145
00146     }
00147
00148     }
00149
00150     }
00151
00152     }
00153
00154     }
00155
00156     }
00157
00158     }
00159
00160     }
00161
00162     }
00163
00164     }
00165
00166     }
00167
00168     }
00169
00170     }
00171
00172     }
00173
00174     }
00175
00176     }
00177
00178     }
00179
00180     }
00181
00182     }
00183
00184     }
00185
00186     }
00187
00188     }
00189
00190     }
00191
00192     }
00193
00194     }
00195
00196     }
00197
00198     }
00199
00200     }
00201
00202     }
00203
00204     }
00205
00206     }
00207
00208     }
00209
00210     }
00211
00212     }
00213
00214     }
00215
00216     }
00217
00218     }
00219
00220     }
00221
00222     }
00223
00224     }
00225
00226     }
00227
00228     }
00229
00230     }
00231
00232     }
00233
00234     }
00235
00236     }
00237
00238     }
00239
00240     }
00241
00242     }
00243
00244     }
00245
00246     }
00247
00248     }
00249
00250     }
00251
00252     }
00253
00254     }
00255
00256     }
00257
00258     }
00259
00260     }
00261
00262     }
00263
00264     }
00265
00266     }
00267
00268     }
00269
00270     }
00271
00272     }
00273
00274     }
00275
00276     }
00277
00278     }
00279
00280     }
00281
00282     }
00283
00284     }
00285
00286     }
00287
00288     }
00289
00290     }
00291
00292     }
00293
00294     }
00295
00296     }
00297
00298     }
00299
00300     }
00301
00302     }
00303
00304     }
00305
00306     }
00307
00308     }
00309
00310     }
00311
00312     }
00313
00314     }
00315
00316     }
00317
00318     }
00319
00320     }
00321
00322     }
00323
00324     }
00325
00326     }
00327
00328     }
00329
00330     }
00331
00332     }
00333
00334     }
00335
00336     }
00337
00338     }
00339
00340     }
00341
00342     }
00343
00344     }
00345
00346     }
00347
00348     }
00349
00350     }
00351
00352     }
00353
00354     }
00355
00356     }
00357
00358     }
00359
00360     }
00361
00362     }
00363
00364     }
00365
00366     }
00367
00368     }
00369
00370     }
00371
00372     }
00373
00374     }
00375
00376     }
00377
00378     }
00379
00380     }
00381
00382     }
00383
00384     }
00385
00386     }
00387
00388     }
00389
00390     }
00391
00392     }
00393
00394     }
00395
00396     }
00397
00398     }
00399
00400     }
00401
00402     }
00403
00404     }
00405
00406     }
00407
00408     }
00409
00410     }
00411
00412     }
00413
00414     }
00415
00416     }
00417
00418     }
00419
00420     }
00421
00422     }
00423
00424     }
00425
00426     }
00427
00428     }
00429
00430     }
00431
00432     }
00433
00434     }
00435
00436     }
00437
00438     }
00439
00440     }
00441
00442     }
00443
00444     }
00445
00446     }
00447
00448     }
00449
00450     }
00451
00452     }
00453
00454     }
00455
00456     }
00457
00458     }
00459
00460     }
00461
00462     }
00463
00464     }
00465
00466     }
00467
00468     }
00469
00470     }
00471
00472     }
00473
00474     }
00475
00476     }
00477
00478     }
00479
00480     }
00481
00482     }
00483
00484     }
00485
00486     }
00487
00488     }
00489
00490     }
00491
00492     }
00493
00494     }
00495
00496     }
00497
00498     }
00499
00500     }
00501
00502     }
00503
00504     }
00505
00506     }
00507
00508     }
00509
00510     }
00511
00512     }
00513
00514     }
00515
00516     }
00517
00518     }
00519
00520     }
00521
00522     }
00523
00524     }
00525
00526     }
00527
00528     }
00529
00530     }
00531
00532     }
00533
00534     }
00535
00536     }
00537
00538     }
00539
00540     }
00541
00542     }
00543
00544     }
00545
00546     }
00547
00548     }
00549
00550     }
00551
00552     }
00553
00554     }
00555
00556     }
00557
00558     }
00559
00560     }
00561
00562     }
00563
00564     }
00565
00566     }
00567
00568     }
00569
00570     }
00571
00572     }
00573
00574     }
00575
00576     }
00577
00578     }
00579
00580     }
00581
00582     }
00583
00584     }
00585
00586     }
00587
00588     }
00589
00590     }
00591
00592     }
00593
00594     }
00595
00596     }
00597
00598     }
00599
00600     }
00601
00602     }
00603
00604     }
00605
00606     }
00607
00608     }
00609
00610     }
00611
00612     }
00613
00614     }
00615
00616     }
00617
00618     }
00619
00620     }
00621
00622     }
00623
00624     }
00625
00626     }
00627
00628     }
00629
00630     }
00631
00632     }
00633
00634     }
00635
00636     }
00637
00638     }
00639
00640     }
00641
00642     }
00643
00644     }
00645
00646     }
00647
00648     }
00649
00650     }
00651
00652     }
00653
00654     }
00655
00656     }
00657
00658     }
00659
00660     }
00661
00662     }
00663
00664     }
00665
00666     }
00667
00668     }
00669
00670     }
00671
00672     }
00673
00674     }
00675
00676     }
00677
00678     }
00679
00680     }
00681
00682     }
00683
00684     }
00685
00686     }
00687
00688     }
00689
00690     }
00691
00692     }
00693
00694     }
00695
00696     }
00697
00698     }
00699
00700     }
00701
00702     }
00703
00704     }
00705
00706     }
00707
00708     }
00709
00710     }
00711
00712     }
00713
00714     }
00715
00716     }
00717
00718     }
00719
00720     }
00721
00722     }
00723
00724     }
00725
00726     }
00727
00728     }
00729
00730     }
00731
00732     }
00733
00734     }
00735
00736     }
00737
00738     }
00739
00740     }
00741
00742     }
00743
00744     }
00745
00746     }
00747
00748     }
00749
00750     }
00751
00752     }
00753
00754     }
00755
00756     }
00757
00758     }
00759
00760     }
00761
00762     }
00763
00764     }
00765
00766     }
00767
00768     }
00769
00770     }
00771
00772     }
00773
00774     }
00775
00776     }
00777
00778     }
00779
00780     }
00781
00782     }
00783
00784     }
00785
00786     }
00787
00788     }
00789
00790     }
00791
00792     }
00793
00794     }
00795
00796     }
00797
00798     }
00799
00800     }
00801
00802     }
00803
00804     }
00805
00806     }
00807
00808     }
00809
00810     }
00811
00812     }
00813
00814     }
00815
00816     }
00817
00818     }
00819
00820     }
00821
00822     }
00823
00824     }
00825
00826     }
00827
00828     }
00829
00830     }
00831
00832     }
00833
00834     }
00835
00836     }
00837
00838     }
00839
00840     }
00841
00842     }
00843
00844     }
00845
00846     }
00847
00848     }
00849
00850     }
00851
00852     }
00853
00854     }
00855
00856     }
00857
00858     }
00859
00860     }
00861
00862     }
00863
00864     }
00865
00866     }
00867
00868     }
00869
00870     }
00871
00872     }
00873
00874     }
00875
00876     }
00877
00878     }
00879
00880     }
00881
00882     }
00883
00884     }
00885
00886     }
00887
00888     }
00889
00890     }
00891
00892     }
00893
00894     }
00895
00896     }
00897
00898     }
00899
00900     }
00901
00902     }
00903
00904     }
00905
00906     }
00907
00908     }
00909
00910     }
00911
00912     }
00913
00914     }
00915
00916     }
00917
00918     }
00919
00920     }
00921
00922     }
00923
00924     }
00925
00926     }
00927
00928     }
00929
00930     }
00931
00932     }
00933
00934     }
00935
00936     }
00937
00938     }
00939
00940     }
00941
00942     }
00943
00944     }
00945
00946     }
00947
00948     }
00949
00950     }
00951
00952     }
00953
00954     }
00955
00956     }
00957
00958     }
00959
00960     }
00961
00962     }
00963
00964     }
00965
00966     }
00967
00968     }
00969
00970     }
00971
00972     }
00973
00974     }
00975
00976     }
00977
00978     }
00979
00980     }
00981
00982     }
00983
00984     }
00985
00986     }
00987
00988     }
00989
00990     }
00991
00992     }
00993
00994     }
00995
00996     }
00997
00998     }
00999
01000     }
01001
01002     }
01003
01004     }
01005
01006     }
01007
01008     }
01009
01010     }
01011
01012     }
01013
01014     }
01015
01016     }
01017
01018     }
01019
01020     }
01021
01022     }
01023
01024     }
01025
01026     }
01027
01028     }
01029
01030     }
01031
01032     }
01033
01034     }
01035
01036     }
01037
01038     }
01039
01040     }
01041
01042     }
01043
01044     }
01045
01046     }
01047
01048     }
01049
01050     }
01051
01052     }
01053
01054     }
01055
01056     }
01057
01058     }
01059
01060     }
01061
01062     }
01063
01064     }
01065
01066     }
01067
01068     }
01069
01070     }
01071
01072     }
01073
01074     }
01075
01076     }
01077
01078     }
01079
01080     }
01081
01082     }
01083
01084     }
01085
01086     }
01087
01088     }
01089
01090     }
01091
01092     }
01093
01094     }
01095
01096     }
01097
01098     }
01099
01100     }
01101
01102     }
01103
01104     }
01105
01106     }
01107
01108     }
01109
01110     }
01111
01112     }
01113
01114     }
01115
01116     }
01117
01118     }
01119
01120     }
01121
01122     }
01123
01124     }
01125
01126     }
01127
01128     }
01129
01130     }
01131
01132     }
01133
01134     }
01135
01136     }
01137
01138     }
01139
01140     }
01141
01142     }
01143
01144     }
01145
01146     }
01147
01148     }
01149
01150     }
01151
01152     }
01153
01154     }
01155
01156     }
01157
01158     }
01159
01160     }
01161
01162     }
01163
01164     }
01165
01166     }
01167
01168     }
01169
01170     }
01171
01172     }
01173
01174     }
01175
01176     }
01177
01178     }
01179
01180     }
01181
01182     }
01183
01184     }
01185
01186     }
01187
01188     }
01189
01190     }
01191
01192     }
01193
01194     }
01195
01196     }
01197
01198     }
01199
01200     }
01201
01202     }
01203
01204     }
01205
01206     }
01207
01208     }
01209
01210     }
01211
01212     }
01213
01214     }
01215
01216     }
01217
01218     }
01219
01220     }
01221
01222     }
01223
01224     }
01225
01226     }
01227
01228     }
01229
01230     }
01231
01232     }
01233
01234     }
01235
01236     }
01237
01238     }
01239
01240     }
01241
01242     }
01243
01244     }
01245
01246     }
01247
01248     }
01249
01250     }
01251
01252     }
01253
01254     }
01255
01256     }
01257
01258     }
01259
01260     }
01261
01262     }
01263
01264     }
01265
01266     }
01267
01268     }
01269
01270     }
01271
01272     }
01273
01274     }
01275
01276     }
01277
01278     }
01279
01280     }
01281
01282     }
01283
01284     }
01285
01286     }
01287
01288     }
01289
01290     }
01291
01292     }
01293
01294     }
01295
01296     }
01297
01298     }
01299
01300     }
01301
01302     }
01303
01304     }
01305
01306     }
01307
01308     }
01309
01310     }
01311
01312     }
01313
01314     }
01315
01316     }
01317
01318     }
01319
01320     }
01321
01322     }
01323
01324     }
01325
01326     }
01327
01328     }
01329
01330     }
01331
01332     }
01333
01334     }
01335
01336     }
01337
01338     }
01339
01340     }
01341
01342     }
01343
01344     }
01345
01346     }
01347
01348     }
01349
01350     }
01351
01352     }
01353
01354     }
01355
01356     }
01357
01358     }
01359
01360     }
01361
01362     }
01363
01364     }
01365
01366     }
01367
01368     }
01369
01370     }
01371
01372     }
01373
01374     }
01375
01376     }
01377
01378     }
01379
01380     }
01381
01382     }
01383
01384     }
01385
01386     }
01387
01388     }
01389
01390     }
01391
01392     }
01393
01394     }
01395
01396     }
01397
01398     }
01399
01400     }
01401
01402     }
01403
01404     }
01405
01406     }
01407
01408     }
01409
01410     }
01411
01412     }
01413
01414     }
01415
01416     }
01417
01418     }
01419
01420     }
01421
01422     }
01423
01424     }
01425
01426     }
01427
01428     }
01429
01430     }
01431
01432     }
01433
01434     }
01435
01436     }
01437
01438     }
01439
01440     }
01441
01442     }
01443
01444     }
01445
01446     }
01447
01448     }
01449
01450     }
01451
01452     }
01453
01454     }
01455
01456     }
01457
01458     }
01459
01460     }
01461
01462     }
01463
01464     }
01465
01466     }
01467
01468     }
01469
01470     }
01471
01472     }
01473
01474     }
01475
01476     }
01477
01478     }
01479
01480     }
01481
01482     }
01483
01484     }
01485
01486     }
01487
01488     }
01489
01490     }
01491
01492     }
01493
01494     }
01495
01496     }
01497
01498     }
01499
01500     }
01501
01502     }
01503
01504     }
01505
01506     }
01507
01508     }
01509
01510     }
01511
01512     }
01513
01514     }
01515
01516     }
01517
01518     }
01519
01520     }
01521
01522     }
01523
01524     }
01525
01526     }
01527
01528     }
01529
01530     }
01531
01532     }
01533
01534     }
01535
01536     }
01537
01538     }
01539
01540     }
01541
01542     }
01543
01544     }
01545
01546     }
01547
01548     }
01549
01550     }
01551
01552     }
01553
01554     }
01555
01556     }
01557
01558     }
01559
01560     }
01561
01562     }
01563
01564     }
01565
01566     }
01567
01568     }
01569
01570     }
01571
01572     }
01573
01574     }
01575
01576     }
01577
01578     }
01579
01580     }
01581
01582     }
01583
01584     }
01585
01586     }
01587
01588     }
01589
01590     }
01591
01592     }
01593
01594     }
01595
01596     }
01597
01598     }
01599
01600     }
01601
01602     }
01603
01604     }
01605
01606     }
01607
01608     }
01609
01610     }
01611
01612     }
01613
01614     }
01615
01616     }
01617
01618     }
01619
01620     }
01621
01622     }
01623
01624     }
01625
01626     }
01627
01628     }
01629
01630     }
01631
01632     }
01633
01634     }
01635
01636     }
01637
01638     }
01639
01640     }
01641
01642     }
01643
01644     }
01645
01646     }
01647
01648     }
01649
01650     }
01651
01652     }
01653
01654     }
01655
01656     }
01657
01658     }
01659
01660     }
01661
01662     }
01663
01664     }
01665
01666     }
01667
01668     }
01669
01670     }
01671
01672     }
01673
01674     }
01
```



```

00136 static bool greater(Key_param_type l, Bst_node const *r)
00137 { return greater(l, k(r)); }
00138
00140 static bool greater(Bst_node const *l, Bst_node const *r)
00141 { return greater(k(l), k(r)); }
00143
00150 template<typename FUNC>
00151 static void remove_tree(Bst_node *head, FUNC &&callback)
00152 {
00153     if (Bst_node *n = Bst_node::next(head, Dir::L))
00154         remove_tree(n, callback);
00155
00156     if (Bst_node *n = Bst_node::next(head, Dir::R))
00157         remove_tree(n, callback);
00158
00159     callback(static_cast<Node *>(head));
00160 }
00161
00162 public:
00163
00172 Const_iterator begin() const { return Const_iterator(head()); }
00177 Const_iterator end() const { return Const_iterator(); }
00178
00183 Iterator begin() { return Iterator(head()); }
00188 Iterator end() { return Iterator(); }
00189
00194 Const_rev_iterator rbegin() const { return Const_rev_iterator(head()); }
00199 Const_rev_iterator rend() const { return Const_rev_iterator(); }
00200
00205 Rev_iterator rbegin() { return Rev_iterator(head()); }
00210 Rev_iterator rend() { return Rev_iterator(); }
00212
00213
00218
00224 Node *find_node(Key_param_type key) const;
00225
00232 Node *lower_bound_node(Key_param_type key) const;
00233
00240 Const_iterator find(Key_param_type key) const;
00241
00250 template<typename FUNC>
00251 void remove_all(FUNC &&callback)
00252 {
00253     if (!_head)
00254         return;
00255
00256     Bst_node *head = _head;
00257     _head = 0;
00258     remove_tree(head, cxx::forward<FUNC>(callback));
00259 }
00260
00261
00263 };
00264
00265 /* find an element */
00266 template< typename Node, typename Get_key, class Compare>
00267 inline
00268 Node *
00269 Bst<Node, Get_key, Compare>::find_node(Key_param_type key) const
00270 {
00271     Dir d;
00272
00273     for (Bst_node *q = _head; q; q = Bst_node::next(q, d))
00274     {
00275         d = dir(key, q);
00276         if (d == Dir::N)
00277             return static_cast<Node*>(q);
00278     }
00279     return 0;
00280 }
00281
00282 template< typename Node, typename Get_key, class Compare>
00283 inline
00284 Node *
00285 Bst<Node, Get_key, Compare>::lower_bound_node(Key_param_type key) const
00286 {
00287     Dir d;
00288     Bst_node *r = 0;
00289
00290     for (Bst_node *q = _head; q; q = Bst_node::next(q, d))
00291     {
00292         d = dir(key, q);
00293         if (d == Dir::L)
00294             r = q; // found a node greater than key
00295         else if (d == Dir::N)
00296             return static_cast<Node*>(q);
00297     }

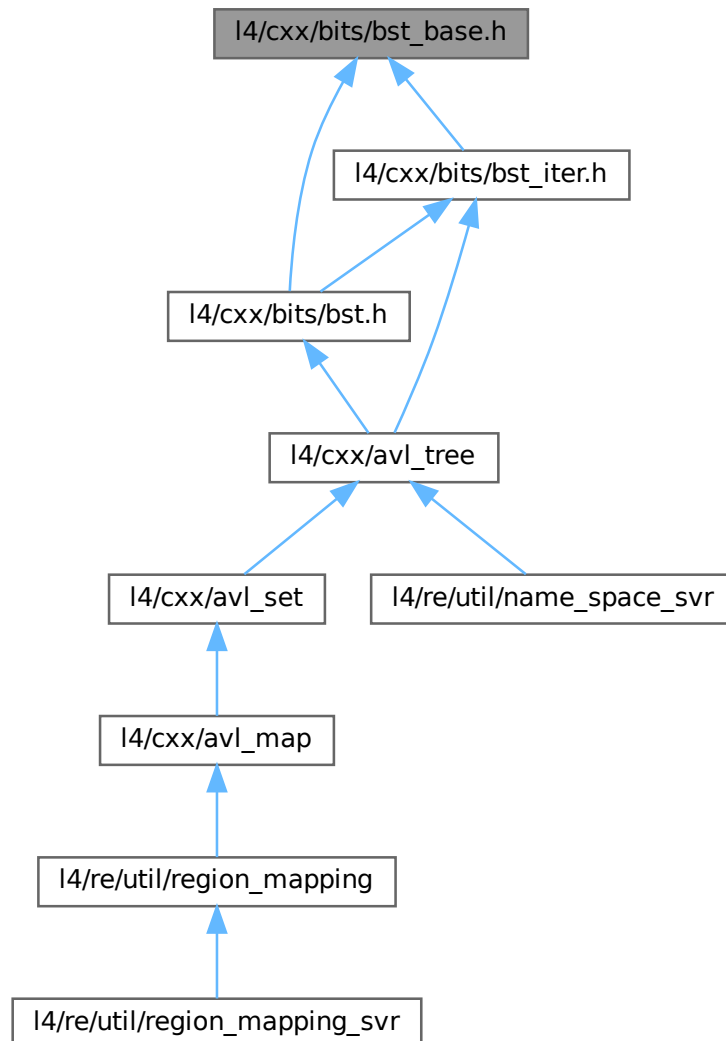
```

```
00298     return static_cast<Node*>(r);
00299 }
00300
00301 /* find an element */
00302 template< typename Node, typename Get_key, class Compare>
00303 inline
00304 typename Bst<Node, Get_key, Compare>::Const_iterator
00305 Bst<Node, Get_key, Compare>::find(Key_param_type key) const
00306 {
00307     Bst_node *q = _head;
00308     Bst_node *r = q;
00309
00310     for (Dir d; q; q = Bst_node::next(q, d))
00311     {
00312         d = dir(key, q);
00313         if (d == Dir::N)
00314             return Iterator(static_cast<Node*>(q), static_cast<Node *>(r));
00315
00316         if (d != Dir::L && q == r)
00317             r = Bst_node::next(q, d);
00318     }
00319     return Iterator();
00320 }
00321
00322 }
```

## 17.174 I4/cxx/bits/bst\_base.h File Reference

AVL tree.

This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [cxx::Bits::Direction](#)  
*The direction to go in a binary search tree.*
- class [cxx::Bits::Bst\\_node](#)  
*Basic type of a node in a binary search tree (BST).*

## Namespaces

- namespace [cxx](#)  
*Our C++ library.*
- namespace [cxx::Bits](#)  
*Internal helpers for the cxx package.*

## 17.174.1 Detailed Description

AVL tree.

Definition in file [bst\\_base.h](#).

## 17.175 bst\_base.h

[Go to the documentation of this file.](#)

```

00001 // vi:ft=cpp
00006 /*
00007  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *          Carsten Weinhold <weinhold@os.inf.tu-dresden.de>
00009  *          economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013
00014 #pragma once
00015
00016 /*
00017  * This file contains very basic bits for implementing binary search trees
00018  */
00019 namespace cxx {
00023 namespace Bits {
00024
00028 struct Direction
00029 {
00031     enum Direction_e
00032     {
00033         L = 0,
00034         R = 1,
00035         N = 2
00036     };
00037     unsigned char d;
00038
00040     Direction() = default;
00041
00043     Direction(Direction_e d) : d(d) {}
00044
00046     explicit Direction(bool b) : d(Direction_e(b)) /*d(b ? R : L)*/ {}
00047
00052     Direction operator ! () const { return Direction(!d); }
00053
00055
00057     bool operator == (Direction_e o) const { return d == o; }
00059     bool operator != (Direction_e o) const { return d != o; }
00061     bool operator == (Direction o) const { return d == o.d; }
00063     bool operator != (Direction o) const { return d != o.d; }
00065 };
00066
00070 class Bst_node
00071 {
00072     // all BSTs are friends
00073     template< typename Node, typename Get_key, typename Compare >
00074     friend class Bst;
00075
00076 protected:
00085
00087     static Bst_node *next(Bst_node const *p, Direction d)
00088     { return p->_c[d.d]; }
00089
00091     static void next(Bst_node *p, Direction d, Bst_node *n)
00092     { p->_c[d.d] = n; }
00093
00095     static Bst_node **next_p(Bst_node *p, Direction d)
00096     { return &p->_c[d.d]; }
00097
00099     template< typename Node > static
00100     Node *next(Bst_node const *p, Direction d)
00101     { return static_cast<Node *>(p->_c[d.d]); }
00102
00104     static void rotate(Bst_node **t, Direction idir);
00106
00107 private:
00108     Bst_node *_c[2];
00109
00110 protected:

```

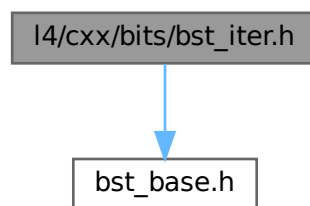
```
00112     Bst_node() {}
00113
00115     explicit Bst_node(bool) { _c[0] = _c[1] = 0; }
00116 };
00117
00118 inline
00119 void
00120 Bst_node::rotate(Bst_node **t, Direction idir)
00121 {
00122     Bst_node *tmp = *t;
00123     *t = next(tmp, idir);
00124     next(tmp, idir, next(*t, !idir));
00125     next(*t, !idir, tmp);
00126 }
00127
00128 }
```

## 17.176 I4/cxx/bits/bst\_iter.h File Reference

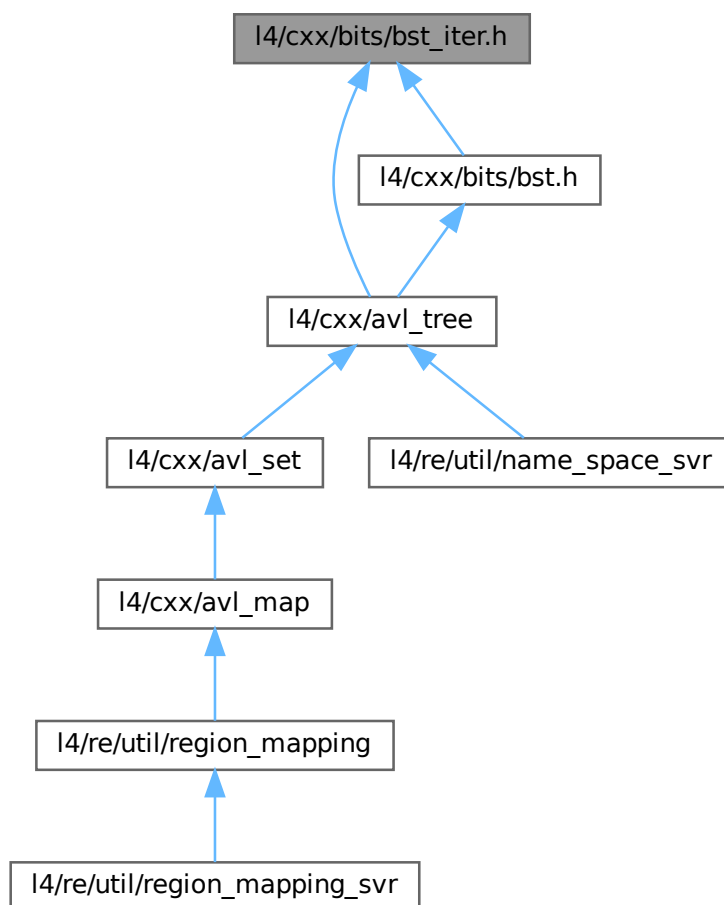
AVL tree.

```
#include "bst_base.h"
```

Include dependency graph for bst\_iter.h:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace `cxx`  
*Our C++ library.*
- namespace `cxx::Bits`  
*Internal helpers for the cxx package.*

### 17.176.1 Detailed Description

AVL tree.

Definition in file [bst\\_iter.h](#).

## 17.177 bst\_iter.h

[Go to the documentation of this file.](#)

```

00001 // vi:ft=cpp
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *      Carsten Weinhold <weinhold@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 #include "bst_base.h"
00012
00013 namespace cxx { namespace Bits {
00014
00015     template< typename Node, typename Node_op >
00016     class __Bst_iter_b
00017     {
00018     protected:
00019         typedef Direction Dir;
00020         Node const *_n;
00021         Node const *_r;
00022
00023         __Bst_iter_b() : _n(0), _r(0) {}
00024
00025         __Bst_iter_b(Node const *t)
00026             : _n(t), _r(_n)
00027         { _downmost(); }
00028
00029         __Bst_iter_b(Node const *t, Node const *r)
00030             : _n(t), _r(r)
00031         {}
00032
00033         inline void _downmost();
00034
00035         inline void inc();
00036
00037     public:
00038         bool operator == (__Bst_iter_b const &o) const { return _n == o._n; }
00039         bool operator != (__Bst_iter_b const &o) const { return _n != o._n; }
00040     };
00041
00042     template< typename Node, typename Node_type, typename Node_op >
00043     class __Bst_iter : public __Bst_iter_b<Node, Node_op>
00044     {
00045     private:
00046         typedef __Bst_iter_b<Node, Node_op> Base;
00047         using Base::_n;
00048         using Base::_r;
00049         using Base::inc;
00050
00051     public:
00052         __Bst_iter() {}
00053
00054         __Bst_iter(Node const *t) : Base(t) {}
00055         __Bst_iter(Node const *t, Node const *r) : Base(t, r) {}
00056
00057         // template<typename Key2>
00058         __Bst_iter(Base const &o) : Base(o) {}
00059
00060         Node_type &operator * () const { return *const_cast<Node *>(_n); }
00061         Node_type *operator -> () const { return const_cast<Node *>(_n); }
00062         __Bst_iter &operator ++ () { inc(); return *this; }
00063         __Bst_iter &operator ++ (int)
00064         { __Bst_iter tmp = *this; inc(); return tmp; }
00065     };
00066
00067 //-----
00068 /* IMPLEMENTATION: __Bst_iter_b */
00069
00070 template< typename Node, typename Node_op>
00071 inline
00072 void __Bst_iter_b<Node, Node_op>::_downmost()
00073 {
00074     while (_n)
00075     {
00076         Node *n = Node_op::child(_n, Dir::L);
00077         if (n)
00078             _n = n;
00079         else

```

```

00134     return;
00135     }
00136 }
00137
00138 template< typename Node, typename Node_op>
00139 void __Bst_iter_b<Node, Node_op>::inc()
00140 {
00141     if (!_n)
00142         return;
00143
00144     if (_n == _r)
00145     {
00146         _r = _n = Node_op::child(_r, Dir::R);
00147         _downmost();
00148         return;
00149     }
00150
00151     if (Node_op::child(_n, Dir::R))
00152     {
00153         _n = Node_op::child(_n, Dir::R);
00154         _downmost();
00155         return;
00156     }
00157
00158     Node const *q = _r;
00159     Node const *p = _r;
00160     while (1)
00161     {
00162         if (Node_op::cmp(_n, q))
00163         {
00164             p = q;
00165             q = Node_op::child(q, Dir::L);
00166         }
00167         else if (_n == q || Node_op::child(q, Dir::R) == _n)
00168         {
00169             _n = p;
00170             return;
00171         }
00172         else
00173             q = Node_op::child(q, Dir::R);
00174     }
00175 }
00176
00177 }}

```

## 17.178 list\_basics.h

```

00001 /*
00002  * (c) 2011 Alexander Warg <warg@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 namespace cxx { namespace Bits {
00011
00012     template< typename T >
00013     class List_iterator_end_ptr
00014     {
00015     private:
00016         template< typename U > friend class Basic_list;
00017         static void *_end;
00018     };
00019
00020     template< typename T >
00021     void *List_iterator_end_ptr<T>::_end;
00022
00023     template< typename VALUE_T, typename TYPE >
00024     struct Basic_list_policy
00025     {
00026         typedef VALUE_T *Value_type;
00027         typedef VALUE_T const *Const_value_type;
00028         typedef TYPE **Type;
00029         typedef TYPE *Const_type;
00030         typedef TYPE *Head_type;
00031         typedef TYPE Item_type;
00032
00033         static Type next(Type c) { return &(*c)->_n; }
00034         static Const_type next(Const_type c) { return c->_n; }
00035     };
00036

```



```

00038 template< typename POLICY >
00039 class Basic_list
00040 {
00041     Basic_list(Basic_list const &) = delete;
00042     void operator = (Basic_list const &) = delete;
00043
00044 public:
00045     typedef typename POLICY::Value_type Value_type;
00046     typedef typename POLICY::Const_value_type Const_value_type;
00047
00048     class Iterator
00049     {
00050     private:
00051         typedef typename POLICY::Type Internal_type;
00052
00053     public:
00054         typedef typename POLICY::Value_type value_type;
00055         typedef typename POLICY::Value_type Value_type;
00056
00057         Value_type operator * () const { return static_cast<Value_type>(*_c); }
00058         Value_type operator -> () const { return static_cast<Value_type>(*_c); }
00059         Iterator operator ++ () { _c = POLICY::next(_c); return *this; }
00060
00061         bool operator == (Iterator const &o) const { return *_c == *o._c; }
00062         bool operator != (Iterator const &o) const { return !operator == (o); }
00063
00064         Iterator() : _c(__end()) {}
00065
00066     private:
00067         friend class Basic_list;
00068         static Internal_type __end()
00069         {
00070             union X { Internal_type l; void **v; } z;
00071             z.v = &Bits::List_iterator_end_ptr<void>::_end;
00072             return z.l;
00073         }
00074
00075         explicit Iterator(Internal_type i) : _c(i) {}
00076
00077         Internal_type _c;
00078     };
00079
00080     class Const_iterator
00081     {
00082     private:
00083         typedef typename POLICY::Const_type Internal_type;
00084
00085     public:
00086         typedef typename POLICY::Value_type value_type;
00087         typedef typename POLICY::Value_type Value_type;
00088
00089         Value_type operator * () const { return static_cast<Value_type>(_c); }
00090         Value_type operator -> () const { return static_cast<Value_type>(_c); }
00091         Const_iterator operator ++ () { _c = POLICY::next(_c); return *this; }
00092
00093         friend bool operator == (Const_iterator const &lhs, Const_iterator const &rhs)
00094         { return lhs._c == rhs._c; }
00095         friend bool operator != (Const_iterator const &lhs, Const_iterator const &rhs)
00096         { return lhs._c != rhs._c; }
00097
00098         Const_iterator() {}
00099         Const_iterator(Iterator const &o) : _c(*o) {}
00100
00101     private:
00102         friend class Basic_list;
00103
00104         explicit Const_iterator(Internal_type i) : _c(i) {}
00105
00106         Internal_type _c;
00107     };
00108
00109     // BSS allocation
00110     explicit Basic_list(bool) {}
00111     Basic_list() : _f(0) {}
00112
00113     Basic_list(Basic_list &&o) : _f(o._f)
00114     {
00115         o.clear();
00116     }
00117
00118     Basic_list &operator = (Basic_list &&o)
00119     {
00120         _f = o._f;
00121         o.clear();
00122         return *this;
00123     }
00124

```

```

00126 bool empty() const { return !_f; }
00128 Value_type front() const { return static_cast<Value_type>(_f); }
00129
00135 void clear() { _f = 0; }
00136
00138 Iterator begin() { return Iterator(&_f); }
00140 Const_iterator begin() const { return Const_iterator(_f); }
00148 static Const_iterator iter(Const_value_type c) { return Const_iterator(c); }
00150 Const_iterator end() const { return Const_iterator(nullptr); }
00152 Iterator end() { return Iterator(); }
00153
00154 protected:
00155 static typename POLICY::Type __get_internal(Iterator const &i) { return i._c; }
00156 static Iterator __iter(typename POLICY::Type c) { return Iterator(c); }
00157
00159 typename POLICY::Head_type _f;
00160 };
00161
00162 }}
00163

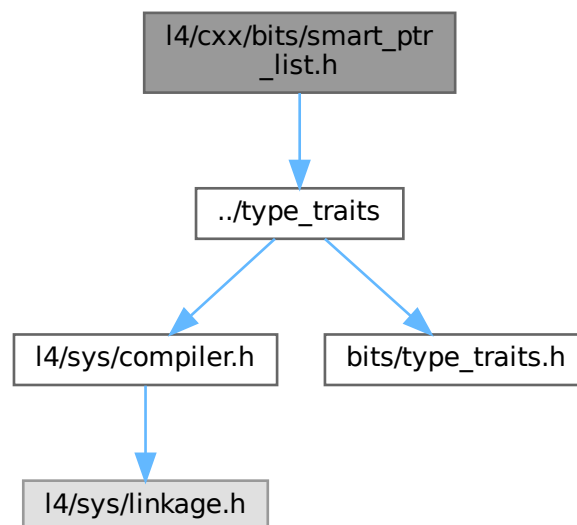
```

## 17.179 l4/cxx/bits/smart\_ptr\_list.h File Reference

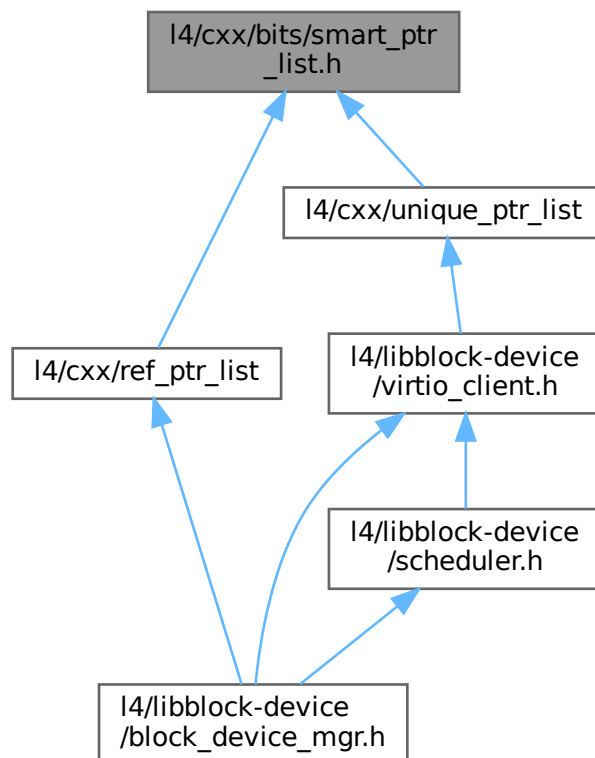
Implementation of a list of smart-pointer-managed objects.

```
#include "../type_traits"
```

Include dependency graph for smart\_ptr\_list.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [cxx::Bits::Smart\\_ptr\\_list\\_item< T, STORE\\_T >](#)  
*List item for an arbitrary item in a [Smart\\_ptr\\_list](#).*
- class [cxx::Bits::Smart\\_ptr\\_list< ITEM >](#)  
*List of smart-pointer-managed objects.*

## Namespaces

- namespace [cxx](#)  
*Our C++ library.*
- namespace [cxx::Bits](#)  
*Internal helpers for the cxx package.*

### 17.179.1 Detailed Description

Implementation of a list of smart-pointer-managed objects.

Definition in file [smart\\_ptr\\_list.h](#).

## 17.180 smart\_ptr\_list.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * Copyright (C) 2018, 2022, 2024 Kernkonzept GmbH.
00007  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00013 #include "../type_traits"
00014
00015 namespace cxx { namespace Bits {
00016
00026 template <typename T, typename STORE_T>
00027 class Smart_ptr_list_item
00028 {
00029     using Value_type = T;
00030     using Storage_type = STORE_T;
00031
00032     template<typename U> friend class Smart_ptr_list;
00033     Storage_type _n;
00034 };
00035
00045 template <typename ITEM>
00046 class Smart_ptr_list
00047 {
00048     using Value_type = typename ITEM::Value_type;
00049     using Next_type = typename ITEM::Storage_type;
00050
00051 public:
00052     class Iterator
00053     {
00054     public:
00055         Iterator() : _c(nullptr) {}
00056
00057         Value_type *operator * () const { return _c; }
00058         Value_type *operator -> () const { return _c; }
00059
00060         Iterator operator ++ ()
00061         {
00062             _c = _c->_n.get();
00063             return *this;
00064         }
00065
00066         bool operator == (Iterator const &o) const { return _c == o._c; }
00067         bool operator != (Iterator const &o) const { return !operator == (o); }
00068
00069     private:
00070         friend class Smart_ptr_list;
00071
00072         explicit Iterator(Value_type *i) : _c(i) {}
00073
00074         Value_type *_c;
00075     };
00076
00077     class Const_iterator
00078     {
00079     public:
00080         Const_iterator() : _c(nullptr) {}
00081
00082         Value_type const *operator * () const { return _c; }
00083         Value_type const *operator -> () const { return _c; }
00084
00085         Const_iterator operator ++ ()
00086         {
00087             _c = _c->_n.get();
00088             return *this;
00089         }
00090
00091         bool operator == (Const_iterator const &o) const { return _c == o._c; }
00092         bool operator != (Const_iterator const &o) const { return !operator == (o); }
00093
00094     private:
00095         friend class Smart_ptr_list;
00096
00097         explicit Const_iterator(Value_type const *i) : _c(i) {}
00098
00099         Value_type const *_c;
00100     };
00101
00102     Smart_ptr_list() : _b(&_f) {}
00103

```

```

00105 void push_front(Next_type &&e)
00106 {
00107     e->_n = cxx::move(this->_f);
00108     this->_f = cxx::move(e);
00109
00110     if (_b == &_f)
00111         _b = &(_f->_n);
00112 }
00113
00115 void push_front(Next_type const &e)
00116 {
00117     e->_n = cxx::move(this->_f);
00118     this->_f = e;
00119
00120     if (_b == &_f)
00121         _b = &(_f->_n);
00122 }
00123
00125 void push_back(Next_type &&e)
00126 {
00127     *_b = cxx::move(e);
00128     _b = &(*_b->_n);
00129 }
00130
00132 void push_back(Next_type const &e)
00133 {
00134     *_b = e;
00135     _b = &(*_b->_n);
00136 }
00137
00139 Value_type *front() const
00140 { return _f.get(); }
00141
00149 Next_type pop_front()
00150 {
00151     Next_type ret = cxx::move(_f);
00152
00153     if (ret)
00154         _f = cxx::move(ret->_n);
00155
00156     if (!_f)
00157         _b = &_f;
00158
00159     return ret;
00160 }
00161
00163 bool empty() const
00164 { return !_f; }
00165
00166 Iterator begin() { return Iterator(_f.get()); }
00167 Iterator end() { return Iterator(); }
00168
00169 Const_iterator begin() const { return Const_iterator(_f.get()); }
00170 Const_iterator end() const { return Const_iterator(); }
00171
00172 Const_iterator cbegin() const { return const_iterator(_f.get()); }
00173 Const_iterator cend() const { return Const_iterator(); }
00174
00175 private:
00176     Next_type _f;
00177     Next_type *_b;
00178 };
00179
00180 }

```

## 17.181 type\_traits.h

```

00001 // vi:ft=cpp
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 namespace cxx {
00013
00014     class Null_type;
00015
00016     template< bool flag, typename T, typename F >

```

```

00017 class Select
00018 {
00019 public:
00020     typedef T Type;
00021 };
00022
00023 template< typename T, typename F >
00024 class Select< false, T, F >
00025 {
00026 public:
00027     typedef F Type;
00028 };
00029
00030
00031
00032 template< typename T, typename U >
00033 class Conversion
00034 {
00035     typedef char S;
00036     class B { char dummy[2]; };
00037     static S test(U);
00038     static B test(...);
00039     static T make_T();
00040 public:
00041     enum
00042     {
00043         exists = sizeof(test(make_T())) == sizeof(S),
00044         two_way = exists && Conversion<U,T>::exists,
00045         exists_2_way = two_way,
00046         same_type = false
00047     };
00048 };
00049
00050 template< >
00051 class Conversion<void, void>
00052 {
00053 public:
00054     enum { exists = 1, two_way = 1, exists_2_way = two_way, same_type = 1 };
00055 };
00056
00057 template< typename T >
00058 class Conversion<T, T>
00059 {
00060 public:
00061     enum { exists = 1, two_way = 1, exists_2_way = two_way, same_type = 1 };
00062 };
00063
00064 template< typename T >
00065 class Conversion<void, T>
00066 {
00067 public:
00068     enum { exists = 0, two_way = 0, exists_2_way = two_way, same_type = 0 };
00069 };
00070
00071 template< typename T >
00072 class Conversion<T, void>
00073 {
00074 public:
00075     enum { exists = 0, two_way = 0, exists_2_way = two_way, same_type = 0 };
00076 };
00077
00078 template< int I >
00079 class Int_to_type
00080 {
00081 public:
00082     enum { i = I };
00083 };
00084
00085 namespace TT
00086 {
00087     template< typename U > class Pointer_traits
00088     {
00089     public:
00090         typedef Null_type Pointee;
00091         enum { value = false };
00092     };
00093
00094     template< typename U > class Pointer_traits< U* >
00095     {
00096     public:
00097         typedef U Pointee;
00098         enum { value = true };
00099     };
00100
00101     template< typename U > struct Ref_traits
00102     {
00103         enum { value = false };

```

```

00104     typedef U Referee;
00105 };
00106
00107 template< typename U > struct Ref_traits<U&>
00108 {
00109     enum { value = true };
00110     typedef U Referee;
00111 };
00112
00113
00114 template< typename U > struct Add_ref { typedef U &Type; };
00115 template< typename U > struct Add_ref<U&> { typedef U Type; };
00116
00117 template< typename U > struct PMF_traits { enum { value = false }; };
00118 template< typename U, typename F > struct PMF_traits<U F::*>
00119 { enum { value = true }; };
00120
00121
00122 template< typename U > class Is_unsigned { public: enum { value = false }; };
00123 template<> class Is_unsigned<unsigned> { public: enum { value = true }; };
00124 template<> class Is_unsigned<unsigned char> {
00125     public: enum { value = true };
00126 };
00127 template<> class Is_unsigned<unsigned short> {
00128     public: enum { value = true };
00129 };
00130 template<> class Is_unsigned<unsigned long> {
00131     public: enum { value = true };
00132 };
00133 template<> class Is_unsigned<unsigned long long> {
00134     public: enum { value = true };
00135 };
00136
00137 template< typename U > class Is_signed { public: enum { value = false }; };
00138 template<> class Is_signed<signed char> { public: enum { value = true }; };
00139 template<> class Is_signed<signed short> { public: enum { value = true }; };
00140 template<> class Is_signed<signed> { public: enum { value = true }; };
00141 template<> class Is_signed<signed long> { public: enum { value = true }; };
00142 template<> class Is_signed<signed long long> {
00143     public: enum { value = true };
00144 };
00145
00146 template< typename U > class Is_int { public: enum { value = false }; };
00147 template<> class Is_int< char > { public: enum { value = true }; };
00148 template<> class Is_int< bool > { public: enum { value = true }; };
00149 template<> class Is_int< wchar_t > { public: enum { value = true }; };
00150
00151 template< typename U > class Is_float { public: enum { value = false }; };
00152 template<> class Is_float< float > { public: enum { value = true }; };
00153 template<> class Is_float< double > { public: enum { value = true }; };
00154 template<> class Is_float< long double > { public: enum { value = true }; };
00155
00156 template<typename T> class Const_traits
00157 {
00158 public:
00159     enum { value = false };
00160     typedef T Type;
00161     typedef const T Const_type;
00162 };
00163
00164 template<typename T> class Const_traits<const T>
00165 {
00166 public:
00167     enum { value = true };
00168     typedef T Type;
00169     typedef const T Const_type;
00170 };
00171 };
00172
00173 template< typename T >
00174 class Type_traits
00175 {
00176 public:
00177
00178     enum
00179     {
00180         is_unsigned = TT::Is_unsigned<T>::value,
00181         is_signed   = TT::Is_signed<T>::value,
00182         is_int      = TT::Is_int<T>::value,
00183         is_float    = TT::Is_float<T>::value,
00184         is_pointer  = TT::Pointer_traits<T>::value,
00185         is_pointer_to_member = TT::PMF_traits<T>::value,
00186         is_reference = TT::Ref_traits<T>::value,
00187         is_scalar   = is_unsigned || is_signed || is_int || is_pointer
00188                     || is_pointer_to_member || is_reference,
00189         is_fundamental = is_unsigned || is_signed || is_float
00190                     || Conversion<T, void>::same_type,

```

```

00191     is_const      = TT::Const_traits<T>::value,
00192
00193     alignment =
00194     (sizeof(T) >= sizeof(unsigned long)
00195      ? sizeof(unsigned long)
00196      : (sizeof(T) >= sizeof(unsigned)
00197        ? sizeof(unsigned)
00198        : (sizeof(T) >= sizeof(short)
00199          ? sizeof(short)
00200          : 1)))
00201 };
00202
00203 typedef typename Select<is_scalar, T, typename TT::Add_ref<typename
TT::Const_traits<T>::Const_type>::Type>::Type Param_type;
00204 typedef typename TT::Pointer_traits<T>::Pointee Pointee_type;
00205 typedef typename TT::Ref_traits<T>::Referee Referee_type;
00206 typedef typename TT::Const_traits<T>::Type Non_const_type;
00207 typedef typename TT::Const_traits<T>::Const_type Const_type;
00208
00209 static unsigned long align(unsigned long a)
00210 {
00211     return (a + static_cast<unsigned long>(alignment) - 1UL)
00212           & ~(static_cast<unsigned long>(alignment) - 1UL);
00213 }
00214 };
00215
00216
00217 };
00218
00219
00220

```

## 17.182 dlist

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2011 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 namespace cxx {
00012
00013 class D_list_item
00014 {
00015 public:
00016     constexpr D_list_item() : _dli_next(nullptr), _dli_prev(nullptr) {}
00017
00018     D_list_item(D_list_item const &) = delete;
00019     void operator = (D_list_item const &) = delete;
00020
00021 private:
00022     friend struct D_list_item_policy;
00023
00024     D_list_item *_dli_next, *_dli_prev;
00025 };
00026
00027 struct D_list_item_policy
00028 {
00029     typedef D_list_item Item;
00030     static D_list_item *&prev(D_list_item *e) { return e->_dli_prev; }
00031     static D_list_item *&next(D_list_item *e) { return e->_dli_next; }
00032     static D_list_item *&prev(D_list_item const *e) { return e->_dli_prev; }
00033     static D_list_item *&next(D_list_item const *e) { return e->_dli_next; }
00034 };
00035
00036 template< typename T >
00037 struct Sd_list_head_policy
00038 {
00039     typedef T *Head_type;
00040     static T *head(Head_type h) { return h; }
00041     static void set_head(Head_type &h, T *v) { h = v; }
00042 };
00043
00044 template<
00045     typename T,
00046     typename C = D_list_item_policy
00047 >
00048 class D_list_cyclic
00049 {

```



```

00050 protected:
00051     template< typename VALUE, typename ITEM >
00052     class __Iterator
00053     {
00054     public:
00055         typedef VALUE *Value_type;
00056         typedef VALUE *value_type;
00057
00058         __Iterator() {}
00059
00060         bool operator == (__Iterator const &o) const
00061         { return _c == o._c; }
00062
00063         bool operator != (__Iterator const &o) const
00064         { return _c != o._c; }
00065
00066         __Iterator &operator ++ ()
00067         {
00068             _c = C::next(_c);
00069             return *this;
00070         }
00071
00072         __Iterator &operator -- ()
00073         {
00074             _c = C::prev(_c);
00075             return *this;
00076         }
00077
00078         Value_type operator * () const { return static_cast<Value_type>(_c); }
00079         Value_type operator -> () const { return static_cast<Value_type>(_c); }
00080
00081     protected:
00082         friend class D_list_cyclic;
00083
00084         explicit __Iterator(ITEM *s) : _c(s) {}
00085
00086         ITEM *_c;
00087     };
00088
00089 public:
00090     typedef T *Value_type;
00091     typedef T *value_type;
00092     typedef __Iterator<T, typename C::Item> Iterator;
00093     typedef Iterator Const_iterator;
00094
00095     static void remove(T *e)
00096     {
00097         C::next(C::prev(e)) = C::next(e);
00098         C::prev(C::next(e)) = C::prev(e);
00099         C::next(e) = nullptr;
00100     }
00101
00102     static Iterator erase(Iterator const &e)
00103     {
00104         typename C::Item *n = C::next(*e);
00105         remove(*e);
00106         return __iter(n);
00107     }
00108
00109     static Iterator iter(T const *e) { return Iterator(const_cast<T*>(e)); }
00110
00111     static bool in_list(T const *e) { return C::next(const_cast<T*>(e)); }
00112     static bool has_sibling(T const *e) { return C::next(const_cast<T*>(e)) != e; }
00113
00114     static Iterator insert_after(T *e, Iterator const &pos)
00115     {
00116         C::prev(e) = pos._c;
00117         C::next(e) = C::next(pos._c);
00118         C::prev(C::next(pos._c)) = e;
00119         C::next(pos._c) = e;
00120         return pos;
00121     }
00122
00123     static Iterator insert_before(T *e, Iterator const &pos)
00124     {
00125         C::next(e) = pos._c;
00126         C::prev(e) = C::prev(pos._c);
00127         C::next(C::prev(pos._c)) = e;
00128         C::prev(pos._c) = e;
00129         return pos;
00130     }
00131
00132 protected:
00133     static void self_insert(typename C::Item *e)
00134     { C::next(e) = C::prev(e) = e; }
00135
00136     static void remove_last(T *e)

```

```

00137     { C::next(e) = nullptr; }
00138
00145     static void splice_heads(Const_iterator pos, typename C::Item *other_list)
00146     {
00147         typename C::Item *ins_next = pos._C;
00148         typename C::Item *ins_prev = C::prev(pos._C);
00149         typename C::Item *other_head = C::next(other_list);
00150         typename C::Item *other_tail = C::prev(other_list);
00151
00152         C::next(ins_prev) = other_head;
00153         C::prev(other_head) = ins_prev;
00154         C::prev(ins_next) = other_tail;
00155         C::next(other_tail) = ins_next;
00156     }
00157
00158     static Iterator __iter(typename C::Item *e) { return Iterator(e); }
00159 };
00160
00161 template<
00162     typename T,
00163     typename C = D_list_item_policy,
00164     typename H = Sd_list_head_policy<T>,
00165     bool BSS = false
00166 >
00167 class Sd_list : public D_list_cyclic<T, C>
00168 {
00169 private:
00170     typedef D_list_cyclic<T, C> Base;
00171
00172 public:
00173     class Iterator : public Base::Iterator
00174     {
00175     public:
00176         Iterator &operator ++ ()
00177         {
00178             if (this->_c)
00179                 Base::Iterator::operator ++ ();
00180
00181             if (this->_c == _h)
00182                 this->_c = nullptr;
00183
00184             return *this;
00185         }
00186
00187         Iterator &operator -- () = delete;
00188
00189 private:
00190     friend class Sd_list;
00191
00192     explicit Iterator(T *h) : Base::Iterator(h), _h(h) {}
00193     typename C::Item *_h;
00194 };
00195
00196 class R_iterator : public Base::Iterator
00197 {
00198 public:
00199     R_iterator &operator ++ ()
00200     {
00201         if (this->_c)
00202             Base::Iterator::operator -- ();
00203
00204         if (this->_c == _h)
00205             this->_c = nullptr;
00206
00207         return *this;
00208     }
00209
00210     R_iterator &operator -- () = delete;
00211
00212 private:
00213     friend class Sd_list;
00214
00215     explicit R_iterator(T *h) : Base::Iterator(h), _h(h) {}
00216     typename C::Item *_h;
00217 };
00218
00219 //typedef typename Base::Iterator Iterator;
00220 enum Pos
00221 { Back, Front };
00222
00223 Sd_list()
00224 {
00225     if (!BSS)
00226         H::set_head(_f, nullptr);
00227 }
00228
00229 bool empty() const { return !H::head(_f); }

```

```

00230 T *front() const { return H::head(_f); }
00231
00232 void remove(T *e)
00233 {
00234     T *h = H::head(_f);
00235     if (e == C::next(e)) // must be the last
00236     {
00237         Base::remove_last(e);
00238         H::set_head(_f, nullptr);
00239         return;
00240     }
00241
00242     if (e == H::head(_f))
00243         H::set_head(_f, static_cast<T*>(C::next(h)));
00244
00245     Base::remove(e);
00246 }
00247
00248 Iterator erase(Iterator const &e)
00249 {
00250     Iterator next = e;
00251     ++next;
00252
00253     remove(*e);
00254     return next;
00255 }
00256
00257 void push(T *e, Pos pos)
00258 {
00259     T *h = H::head(_f);
00260     if (!h)
00261     {
00262         Base::self_insert(e);
00263         H::set_head(_f, e);
00264     }
00265     else
00266     {
00267         Base::insert_before(e, this->iter(h));
00268         if (pos == Front)
00269             H::set_head(_f, e);
00270     }
00271 }
00272
00273 void push_back(T *e) { push(e, Back); }
00274 void push_front(T *e) { push(e, Front); }
00275 void rotate_to(T *h) { H::set_head(_f, h); }
00276
00277 typename H::Head_type const &head() const { return _f; }
00278 typename H::Head_type &head() { return _f; }
00279
00280 Iterator begin() { return Iterator(H::head(_f)); }
00281 Iterator end() { return Iterator(nullptr); }
00282
00283 R_iterator rbegin()
00284 {
00285     if (head())
00286         return R_iterator(static_cast<T*>(C::prev(H::head(_f))));
00287     return R_iterator(nullptr);
00288 }
00289 R_iterator rend() { return R_iterator(nullptr); }
00290
00291 private:
00292     Sd_list(Sd_list const &);
00293     void operator = (Sd_list const &);
00294
00295     typename H::Head_type _f;
00296 };
00297
00298 template<
00299     typename T,
00300     typename C = D_list_item_policy,
00301     bool BSS = false
00302 >
00303 class D_list : public D_list_cyclic<T, C>
00304 {
00305 private:
00306     typedef D_list_cyclic<T, C> Base;
00307     typedef typename C::Item Internal_type;
00308
00309 public:
00310     enum Pos
00311     { Back, Front };
00312
00313     typedef typename Base::Iterator Iterator;
00314     typedef typename Base::Const_iterator Const_iterator;
00315     typedef T* value_type;
00316     typedef T* Value_type;

```

```

00317
00318 D_list() { this->self_insert(&_h); }
00319 ~D_list() { clear(); }
00320
00321 D_list(D_list &o)
00322 {
00323     if (o.empty())
00324     {
00325         this->self_insert(&_h);
00326     }
00327     else
00328     {
00329         Internal_type *p = C::prev(&o._h);
00330         Internal_type *n = C::next(&o._h);
00331         C::prev(&_h) = p;
00332         C::next(&_h) = n;
00333         C::next(p) = &_h;
00334         C::prev(n) = &_h;
00335         o.self_insert(&o._h);
00336     }
00337 }
00338
00339 D_list &operator=(D_list &o)
00340 {
00341     if (&o == this)
00342         return *this;
00343
00344     clear();
00345
00346     if (!o.empty())
00347     {
00348         Internal_type *p = C::prev(&o._h);
00349         Internal_type *n = C::next(&o._h);
00350         C::prev(&_h) = p;
00351         C::next(&_h) = n;
00352         C::next(p) = &_h;
00353         C::prev(n) = &_h;
00354         o.self_insert(&o._h);
00355     }
00356 }
00357
00358 D_list(D_list const &) = delete;
00359 void operator = (D_list const &) = delete;
00360
00361 void splice(Const_iterator pos, D_list &&other)
00362 {
00363     if (other.empty())
00364         return;
00365
00366     Base::splice_heads(pos, &other._h);
00367     other.self_insert(&other._h);
00368 }
00369
00370 bool empty() const { return C::next(&_h) == &_h; }
00371
00372 static void remove(T *e) { Base::remove(e); }
00373 Iterator erase(Iterator const &e) { return Base::erase(e); }
00374
00375 void clear()
00376 {
00377     // Just clear the _dli_next pointers of all elements. It is the indicator
00378     // that an element is not on a list.
00379     Internal_type *i = C::next(&_h);
00380     while (i != &_h)
00381     {
00382         Internal_type *d = i;
00383         i = C::next(i);
00384         C::next(d) = nullptr;
00385     }
00386
00387     this->self_insert(&_h);
00388 }
00389
00390 void push(T *e, Pos pos)
00391 {
00392     if (pos == Front)
00393         Base::insert_after(e, end());
00394     else
00395         Base::insert_before(e, end());
00396 }
00397
00398 void push_back(T *e) { push(e, Back); }
00399 void push_front(T *e) { push(e, Front); }
00400
00401 T *pop_back()
00402 {
00403     T *ret = *(end()--);

```

```

00409     remove(ret);
00410     return ret;
00411 }
00412
00418 T *pop_front()
00419 {
00420     T *ret = *begin();
00421     remove(ret);
00422     return ret;
00423 }
00424
00425 Iterator begin() const { return this->__iter(C::next(const_cast<Internal_type *>(&_h))); }
00426 Iterator end() const { return this->__iter(const_cast<Internal_type *>(&_h)); }
00427
00428 bool has_sibling(T const *e)
00429 {
00430     return C::next(const_cast<T*>(e)) != &_h
00431         || C::prev(const_cast<T*>(e)) != &_h;
00432 }
00433
00434 private:
00435     Internal_type _h;
00436 };
00437
00438 }
00439

```

## 17.183 l4/cxx/exceptions File Reference

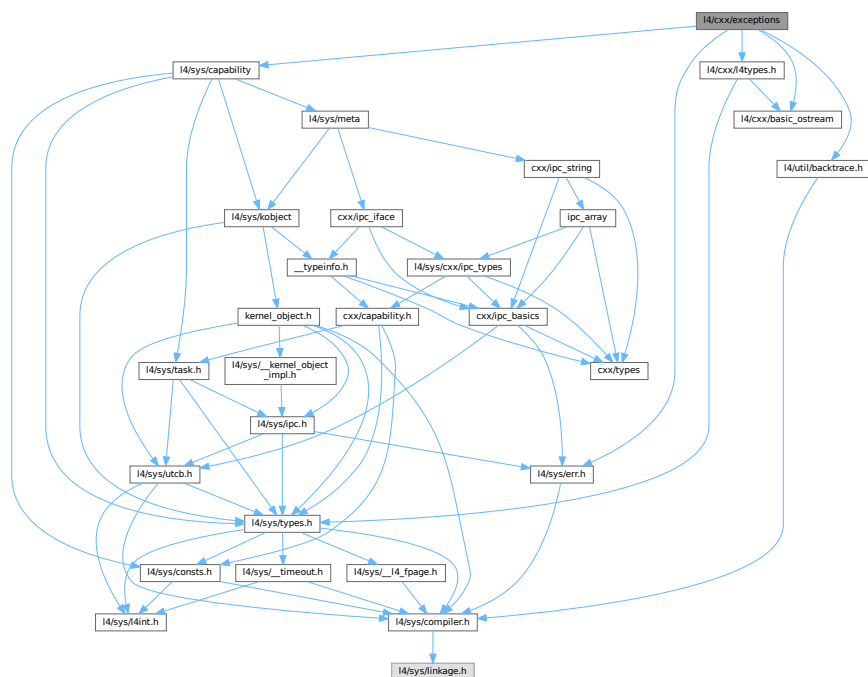
Base exceptions.

```

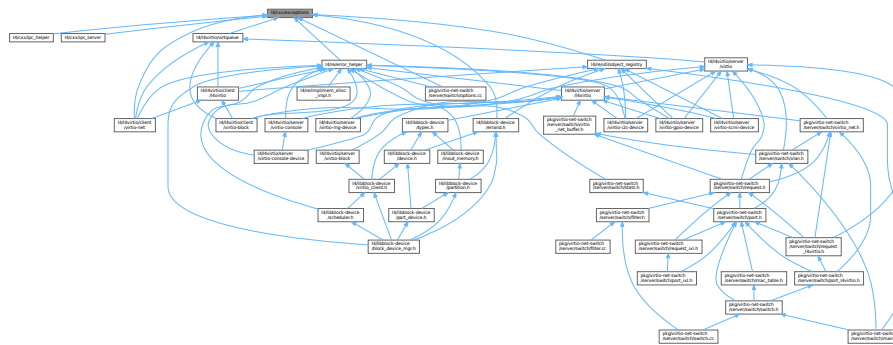
#include <l4/cxx/l4types.h>
#include <l4/cxx/basic_ostream>
#include <l4/sys/err.h>
#include <l4/sys/capability>
#include <l4/util/backtrace.h>

```

Include dependency graph for exceptions:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4::Exception\\_tracer](#)  
*Back-trace support for exceptions.*
- class [L4::Base\\_exception](#)  
*Base class for all exceptions, thrown by the [L4Re](#) framework.*
- class [L4::Runtime\\_error](#)  
*Exception for an abstract runtime error.*
- class [L4::Out\\_of\\_memory](#)  
*Exception signalling insufficient memory.*
- class [L4::Element\\_already\\_exists](#)  
*Exception for duplicate element insertions.*
- class [L4::Unknown\\_error](#)  
*Exception for an unknown condition.*
- class [L4::Element\\_not\\_found](#)  
*Exception for a failed lookup (element not found).*
- class [L4::Invalid\\_capability](#)  
*Indicates that an invalid object was invoked.*
- class [L4::Com\\_error](#)  
*Error conditions during IPC.*
- class [L4::Bounds\\_error](#)  
*Access out of bounds.*

## Namespaces

- namespace [L4](#)  
*[L4](#) low-level kernel interface.*

## Macros

- `#define L4_CXX_EXCEPTION_BACKTRACE 20`  
*Number of instruction pointers in backtrace.*

## 17.183.1 Detailed Description

Base exceptions.

Definition in file [exceptions](#).

## 17.184 exceptions

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #pragma once
00015
00016 #include <l4/cxx/l4types.h>
00017 #include <l4/cxx/basic_ostream>
00018 #include <l4/sys/err.h>
00019 #include <l4/sys/capability>
00020
00021
00027
00028 #ifndef L4_CXX_NO_EXCEPTION_BACKTRACE
00029 # define L4_CXX_EXCEPTION_BACKTRACE 20
00030 #endif
00031
00032 #if defined(L4_CXX_EXCEPTION_BACKTRACE)
00033 #include <l4/util/backtrace.h>
00034 #endif
00035
00037 namespace L4
00038 {
00051     class Exception_tracer
00052     {
00053     #if defined(L4_CXX_EXCEPTION_BACKTRACE)
00054     private:
00055         void *_pc_array[L4_CXX_EXCEPTION_BACKTRACE];
00056         int _frame_cnt;
00057
00058     protected:
00062     #if defined(__PIC__)
00063         Exception_tracer() noexcept : _frame_cnt(0) {}
00064     #else
00065         Exception_tracer() noexcept
00066             : _frame_cnt(l4util_backtrace(_pc_array, L4_CXX_EXCEPTION_BACKTRACE)) {}
00067     #endif
00068
00069     public:
00073         void const *const *pc_array() const noexcept { return _pc_array; }
00077         int frame_count() const noexcept { return _frame_cnt; }
00078     #else
00079     protected:
00083         Exception_tracer() noexcept {}
00084
00085     public:
00089         void const *const *pc_array() const noexcept { return 0; }
00093         int frame_count() const noexcept { return 0; }
00094     #endif
00095     };
00096
00105     class Base_exception : public Exception_tracer
00106     {
00107     protected:
00109         Base_exception() noexcept {}
00110
00111     public:
00115         virtual char const *str() const noexcept = 0;
00116
00118         virtual ~Base_exception() noexcept {}
00119     };
00120
00128     class Runtime_error : public Base_exception
00129     {
00130     private:

```

```

00131     long _errno;
00132     char _extra[80];
00133
00134 public:
00141     explicit Runtime_error(long err_no, char const *extra = 0) noexcept
00142     : _errno(err_no)
00143     {
00144         if (!extra)
00145             _extra[0] = 0;
00146         else
00147         {
00148             unsigned i = 0;
00149             for (; i < sizeof(_extra) && extra[i]; ++i)
00150                 _extra[i] = extra[i];
00151             _extra[i < sizeof(_extra) ? i : sizeof(_extra) - 1] = 0;
00152         }
00153     }
00154     char const *str() const noexcept override
00155     { return l4sys_errtostr(_errno); }
00156
00162     char const *extra_str() const { return _extra; }
00163     ~Runtime_error() noexcept {}
00164
00170     long err_no() const noexcept { return _errno; }
00171 };
00172
00177 class Out_of_memory : public Runtime_error
00178 {
00179 public:
00181     explicit Out_of_memory(char const *extra = "") noexcept
00182     : Runtime_error(-L4_ENOMEM, extra) {}
00184     ~Out_of_memory() noexcept {}
00185 };
00186
00187
00192 class Element_already_exists : public Runtime_error
00193 {
00194 public:
00195     explicit Element_already_exists(char const *e = "") noexcept
00196     : Runtime_error(-L4_EEXIST, e) {}
00197     ~Element_already_exists() noexcept {}
00198 };
00199
00208 class Unknown_error : public Base_exception
00209 {
00210 public:
00211     Unknown_error() noexcept {}
00212     char const *str() const noexcept override { return "unknown error"; }
00213     ~Unknown_error() noexcept {}
00214 };
00215
00220 class Element_not_found : public Runtime_error
00221 {
00222 public:
00223     explicit Element_not_found(char const *e = "") noexcept
00224     : Runtime_error(-L4_ENOENT, e) {}
00225 };
00226
00234 class Invalid_capability : public Base_exception
00235 {
00236 private:
00237     Cap<void> const _o;
00238
00239 public:
00244     explicit Invalid_capability(Cap<void> const &o) noexcept : _o(o) {}
00245     template< typename T>
00246     explicit Invalid_capability(Cap<T> const &o) noexcept : _o(o.cap()) {}
00247     char const *str() const noexcept override { return "invalid object"; }
00248
00253     Cap<void> const &cap() const noexcept { return _o; }
00254     ~Invalid_capability() noexcept {}
00255 };
00256
00263 class Com_error : public Runtime_error
00264 {
00265 public:
00270     explicit Com_error(long err) noexcept : Runtime_error(err) {}
00271
00272     ~Com_error() noexcept {}
00273 };
00274
00278 class Bounds_error : public Runtime_error
00279 {
00280 public:
00281     explicit Bounds_error(char const *e = "") noexcept
00282     : Runtime_error(-L4_ERANGE, e) {}
00283     ~Bounds_error() noexcept {}

```



```

00284     };
00286 };
00287
00288 inline
00289 L4::BasicOStream &
00290 operator « (L4::BasicOStream &o, L4::Base_exception const &e)
00291 {
00292     o « "Exception: " « e.str() « ".\n";
00293     o « "Backtrace:\n";
00294     for (int i = 0; i < e.frame_count(); ++i)
00295         o « "    " « L4::n_hex(l4_addr_t(e.pc_array()[i])) « '\n';
00296
00297     return o;
00298 }
00299
00300 inline
00301 L4::BasicOStream &
00302 operator « (L4::BasicOStream &o, L4::Runtime_error const &e)
00303 {
00304     o « "Exception: " « e.str();
00305     if (e.extra_str())
00306         o « ": " « e.extra_str();
00307     o « "\n" « "Backtrace:\n";
00308     for (int i = 0; i < e.frame_count(); ++i)
00309         o « "    " « L4::n_hex(l4_addr_t(e.pc_array()[i])) « '\n';
00310
00311     return o;
00312 }

```

## 17.185 hlist

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2011 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 #include "bits/list_basics.h"
00012 #include "type_traits"
00013
00014 namespace cxx {
00015
00021 template<typename ELEM_TYPE>
00022 class H_list_item_t
00023 {
00024 public:
00030     H_list_item_t() : _n(0), _pn(0) {}
00037     ~H_list_item_t() noexcept { l_remove(); }
00038
00039 private:
00040     H_list_item_t(H_list_item_t const &) = delete;
00041
00042     template<typename T, typename P> friend class H_list;
00043     template<typename T, typename X> friend struct Bits::Basic_list_policy;
00044
00045     void l_remove() noexcept
00046     {
00047         if (!_pn)
00048             return;
00049
00050         *_pn = _n;
00051         if (_n)
00052             _n->_pn = _pn;
00053
00054         _pn = 0;
00055     }
00056
00057     H_list_item_t *_n, **_pn;
00058 };
00059
00061 typedef H_list_item_t<void> H_list_item;
00062
00068 template< typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item> >
00069 class H_list : public Bits::Basic_list<POLICY>
00070 {
00071 private:
00072     typedef typename POLICY::Item_type Item;
00073     typedef Bits::Basic_list<POLICY> Base;
00074     H_list(H_list const &);

```

```

00075     void operator = (H_list const &);
00076
00077 public:
00078     typedef typename Base::Iterator Iterator;
00079
00080     // BSS allocation
00081     explicit H_list(bool x) : Base(x) {}
00082     H_list() : Base() {}
00083
00093     static Iterator iter(T *c) { return Base::__iter(c->Item::_pn); }
00094
00096     static bool in_list(T const *e) { return e->Item::_pn; }
00097
00099     void add(T *e)
00100     {
00101         if (this->_f)
00102             this->_f->_pn = &e->Item::_n;
00103         e->Item::_n = this->_f;
00104         e->Item::_pn = &this->_f;
00105         this->_f = static_cast<Item*>(e);
00106     }
00107
00109     void push_front(T *e) { add(e); }
00110
00116     T *pop_front()
00117     {
00118         T *r = this->front();
00119         remove(r);
00120         return r;
00121     }
00122
00133     Iterator insert(T *e, Iterator const &pred)
00134     {
00135         Item **x = &this->_f;
00136         if (pred != Base::end())
00137             x = &(*pred)->_n;
00138
00139         e->Item::_n = *x;
00140
00141         if (*x)
00142             (*x)->_pn = &(e->Item::_n);
00143
00144         e->Item::_pn = x;
00145         *x = static_cast<Item*>(e);
00146         return iter(e);
00147     }
00148
00160     static Iterator insert_after(T *e, Iterator const &pred)
00161     {
00162         Item **x = &(*pred)->_n;
00163         e->Item::_n = *x;
00164
00165         if (*x)
00166             (*x)->_pn = &(e->Item::_n);
00167
00168         e->Item::_pn = x;
00169         *x = static_cast<Item*>(e);
00170         return iter(e);
00171     }
00172
00180     static void insert_before(T *e, Iterator const &succ)
00181     {
00182         Item **x = Base::__get_internal(succ);
00183
00184         e->Item::_n = *x;
00185         e->Item::_pn = x;
00186
00187         if (*x)
00188             (*x)->_pn = &e->Item::_n;
00189
00190         *x = static_cast<Item*>(e);
00191     }
00192
00204     static void replace(T *p, T *e)
00205     {
00206         e->Item::_n = p->Item::_n;
00207         e->Item::_pn = p->Item::_pn;
00208         *(p->Item::_pn) = static_cast<Item*>(e);
00209         if (e->Item::_n)
00210             e->Item::_n->_pn = &(e->Item::_n);
00211
00212         p->Item::_pn = 0;
00213     }
00214
00220     static void remove(T *e)
00221     { e->Item::_l_remove(); }
00222

```

```

00236 static Iterator erase(Iterator const &e)
00237 { e->Item::l_remove(); return e; }
00238 };
00239
00247 template< typename T >
00248 struct H_list_t : H_list<T, Bits::Basic_list_policy< T, H_list_item_t<T> > >
00249 {
00250     H_list_t() = default;
00251     H_list_t(bool b)
00252     : H_list<T, Bits::Basic_list_policy< T, H_list_item_t<T> > >(b)
00253     {};
00254 };
00255
00256 template< typename T >
00257 class H_list_bss : public H_list<T>
00258 {
00259 public:
00260     H_list_bss() : H_list<T>(true) {}
00261 };
00262
00263 template< typename T >
00264 class H_list_t_bss : public H_list_t<T>
00265 {
00266 public:
00267     H_list_t_bss() : H_list_t<T>(true) {}
00268 };
00269
00270
00271 }

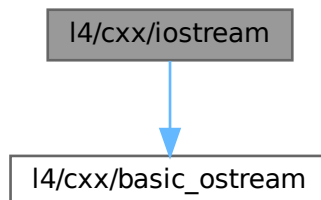
```

## 17.186 l4/cxx/iostream File Reference

IO Stream.

```
#include <l4/cxx/basic_ostream>
```

Include dependency graph for iostream:



### Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*

### Variables

- BasicOStream **L4::cout**  
*Standard output stream.*
- BasicOStream **L4::cerr**  
*Standard error stream.*

## 17.186.1 Detailed Description

IO Stream.

Definition in file [iostream](#).

## 17.187 iostream

[Go to the documentation of this file.](#)

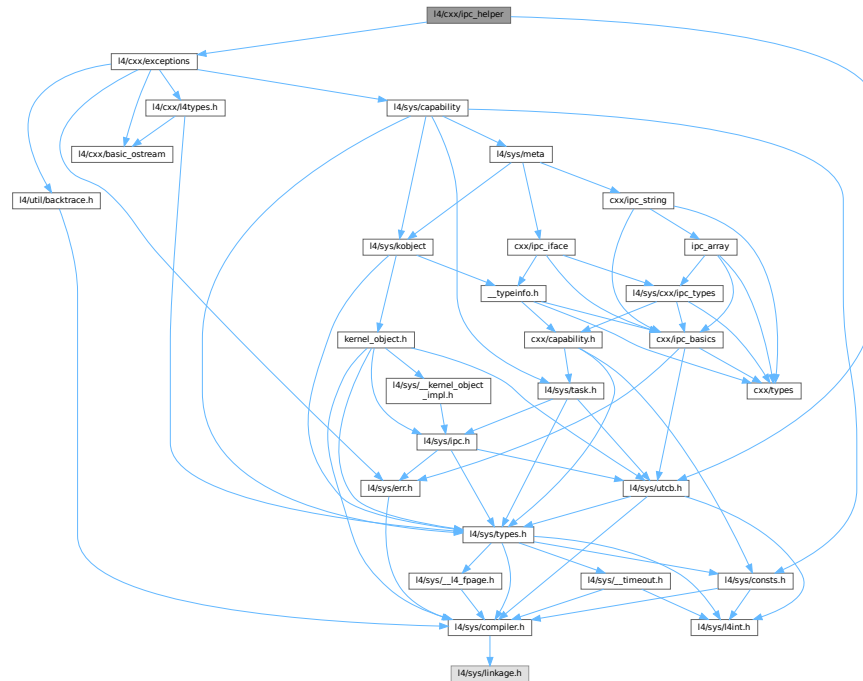
```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *                      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #pragma once
00015
00016 #include <l4/cxx/basic_ostream>
00017
00018 namespace L4 {
00019
00023     extern BasicOStream cout;
00024
00028     extern BasicOStream cerr;
00029
00030     extern void iostream_init();
00031
00032     static void __attribute__((used, constructor)) __iostream_init()
00033     { iostream_init(); }
00034 };
```

## 17.188 l4/cxx/ipc\_helper File Reference

IPC helper.

```
#include <l4/cxx/exceptions>
#include <l4/sys/utcb.h>
```

Include dependency graph for ipc\_helper:



## Namespaces

- namespace [L4](#)  
*[L4](#) low-level kernel interface.*

## Functions

- void [L4::throw\\_ipc\\_exception](#) ([L4::Cap](#)< void > const &o, [l4\\_msgtag\\_t](#) const &err, [l4\\_utcb\\_t](#) \*utcb)  
*Throw an [L4](#) IPC error as exception.*
- void [L4::throw\\_ipc\\_exception](#) (void const \*o, [l4\\_msgtag\\_t](#) const &err, [l4\\_utcb\\_t](#) \*utcb)  
*Throw an [L4](#) IPC error as exception.*

### 17.188.1 Detailed Description

IPC helper.

Definition in file [ipc\\_helper](#).

## 17.189 ipc\_helper

[Go to the documentation of this file.](#)

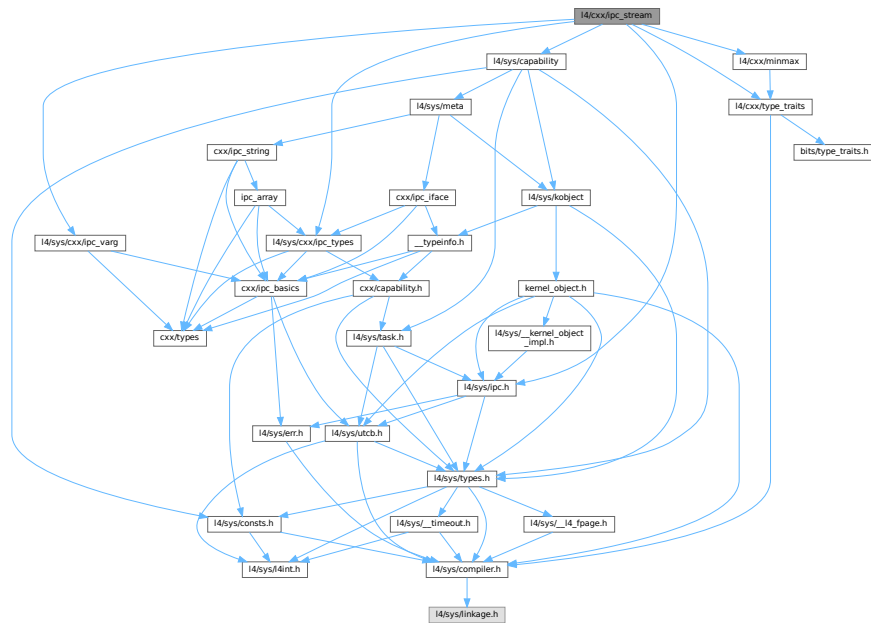
```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include <l4/cxx/exceptions>
00012 #include <l4/sys/utcb.h>
00013
00014 namespace L4
00015 {
00016 #ifdef __EXCEPTIONS
00017     inline void
00018     throw_ipc_exception([[maybe_unused]] L4::Cap<void> const &o,
00019                         l4_msgtag_t const &err, l4_utcb_t *utcb)
00020     {
00021         if (err.has_error())
00022             throw (L4::Com_error(l4_error_u(err, utcb)));
00023     }
00024
00025     inline void
00026     throw_ipc_exception(void const *o, l4_msgtag_t const &err,
00027                         l4_utcb_t *utcb)
00028     {
00029         throw_ipc_exception(L4::Cap<void>(o), err, utcb);
00030     }
00031 #endif
00032 }
```

## 17.190 l4/cxx/ipc\_stream File Reference

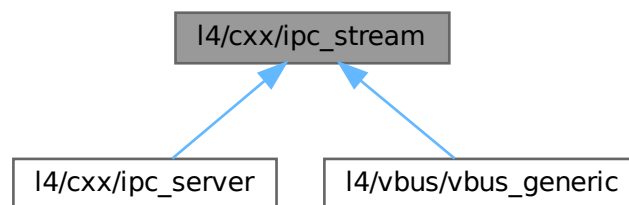
IPC stream.

```
#include <l4/sys/ipc.h>
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_varg>
#include <l4/cxx/type_traits>
#include <l4/cxx/minmax>
```

Include dependency graph for ipc\_stream:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4::ipc::Str\\_cp\\_in< T >](#)  
Abstraction for extracting a zero-terminated string from an [ipc::lstream](#).
- class [L4::ipc::Msg\\_ptr< T >](#)  
Pointer to an element of type *T* in an [ipc::lstream](#).
- class [L4::ipc::lstream](#)  
Input stream for IPC unmarshalling.
- class [L4::ipc::Ostream](#)  
Output stream for IPC marshalling.
- class [L4::ipc::lostream](#)  
Input/Output stream for IPC [un]marshalling.

## Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*
- namespace [L4::lpc](#)  
*IPC related functionality.*

## Functions

- `template<typename T>`  
`Internal::Buf_cp_out< T > L4::lpc::buf\_cp\_out (T const *v, unsigned long size)`  
*Insert an array into an [lpc::Ostream](#).*
- `template<typename T>`  
`Internal::Buf_cp_in< T > L4::lpc::buf\_cp\_in (T *v, unsigned long &size)`  
*Extract an array from an [lpc::Istream](#).*
- `template<typename T>`  
`Str_cp_in< T > L4::lpc::str\_cp\_in (T *v, unsigned long &size)`  
*Create a [Str\\_cp\\_in](#) for the given values.*
- `template<typename T>`  
`Msg_ptr< T > L4::lpc::msg\_ptr (T *&p)`  
*Create an [Msg\\_ptr](#) to adjust the given pointer.*
- `template<typename T>`  
`Internal::Buf_in< T > L4::lpc::buf\_in (T *&v, unsigned long &size)`  
*Return a pointer to stream array data.*
- `L4::lpc::Istream & operator>> (L4::lpc::Istream &s, bool &v)`  
*Extract one element of type *T* from the stream *s*.*
- `L4::lpc::Istream & operator>> (L4::lpc::Istream &s, l4\_msgtag\_t &v)`  
*Extract the *L4* message tag from the stream *s*.*
- `template<typename T>`  
`L4::lpc::Istream & operator>> (L4::lpc::Istream &s, L4::lpc::Internal::Buf\_in< T > const &v)`  
*Extract an array of *T* elements from the stream *s*.*
- `template<typename T>`  
`L4::lpc::Istream & operator>> (L4::lpc::Istream &s, L4::lpc::Msg\_ptr< T > const &v)`  
*Extract an element of type *T* from the stream *s*.*
- `template<typename T>`  
`L4::lpc::Istream & operator>> (L4::lpc::Istream &s, L4::lpc::Str\_cp\_in< T > const &v)`  
*Extract a zero-terminated string from the stream.*
- `L4::lpc::Ostream & operator<< (L4::lpc::Ostream &s, bool v)`  
*Insert an element to type *T* into the stream *s*.*
- `L4::lpc::Ostream & operator<< (L4::lpc::Ostream &s, l4\_msgtag\_t const &v)`  
*Insert the *L4* message tag into the stream *s*.*
- `template<typename T>`  
`L4::lpc::Ostream & operator<< (L4::lpc::Ostream &s, L4::lpc::Internal::Buf\_cp\_out< T > const &v)`  
*Insert an array with elements of type *T* into the stream *s*.*
- `L4::lpc::Ostream & operator<< (L4::lpc::Ostream &s, char const *v)`  
*Insert a zero terminated character string into the stream *s*.*
- `template<typename T>`  
`T L4::lpc::read (Istream &s)`  
*Read a value out of a stream.*



## 17.190.1 Detailed Description

IPC stream.

Definition in file [ipc\\_stream](#).

## 17.190.2 Function Documentation

### 17.190.2.1 operator<<() [1/4]

```
L4::Ipc::Ostream & operator<< (  
    L4::Ipc::Ostream & s,  
    bool v) [inline]
```

Insert an element to type T into the stream s.

#### Parameters

s	The stream to insert the element v.
v	The element to insert.

#### Returns

The stream s.

Definition at line 1197 of file [ipc\\_stream](#).

References [L4::Ipc::Ostream::put\(\)](#).

Here is the call graph for this function:



### 17.190.2.2 operator<<() [2/4]

```
L4::Ipc::Ostream & operator<< (  
    L4::Ipc::Ostream & s,  
    char const * v) [inline]
```

Insert a zero terminated character string into the stream s.

#### Parameters

<i>s</i>	The stream to insert the string <i>v</i> .
<i>v</i>	The string to insert.

**Returns**

The stream *s*.

This operator produces basically the same content as the array insertion, however the length of the array is calculated using `strlen(v) + 1`. The string is copied into the message including the trailing zero.

Definition at line 1269 of file `ipc_stream`.

References [L4::Ipc::Ostream::put\(\)](#).

Here is the call graph for this function:

**17.190.2.3 operator<<() [3/4]**

```

template<typename T>
L4::Ipc::Ostream & operator<< (
    L4::Ipc::Ostream & s,
    L4::Ipc::Internal::Buf_cp_out< T > const & v) [inline]
  
```

Insert an array with elements of type *T* into the stream *s*.

**Parameters**

<i>s</i>	The stream to insert the array <i>v</i> .
<i>v</i>	The array to insert (see <code>Ipc::Buf_cp_out()</code> ).

**Returns**

The stream *s*.

Definition at line 1248 of file [ipc\\_stream](#).

References [L4::Ipc::Ostream::put\(\)](#).

Here is the call graph for this function:

**17.190.2.4 operator<<() [4/4]**

```
L4::Ipc::Ostream & operator<< (  
    L4::Ipc::Ostream & s,  
    l4_msgtag_t const & v) [inline]
```

Insert the [L4](#) message tag into the stream *s*.

**Parameters**

<i>s</i>	The stream to insert the tag <i>v</i> .
<i>v</i>	The <a href="#">L4</a> message tag to insert.

**Returns**

The stream *s*.

**Note**

Only one message tag can be inserted into a stream. Multiple insertions simply overwrite previous insertions.

Definition at line 1232 of file [ipc\\_stream](#).

References [L4::Ipc::Ostream::tag\(\)](#).

Here is the call graph for this function:



**17.190.2.5 operator>>() [1/6]**

```
L4::Ipc::Istream & operator>> (
    L4::Ipc::Istream & s,
    bool & v) [inline]
```

Extract one element of type T from the stream s.

**Parameters**

	s	The stream to extract from.
out	v	Extracted value.

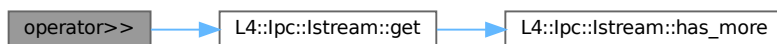
**Returns**

The stream s.

Definition at line 1044 of file [ipc\\_stream](#).

References [L4::Ipc::Istream::get\(\)](#).

Here is the call graph for this function:

**17.190.2.6 operator>>() [2/6]**

```
template<typename T>
L4::Ipc::Istream & operator>> (
    L4::Ipc::Istream & s,
    L4::Ipc::Internal::Buf_cp_in< T > const & v) [inline]
```

Extract an array of T elements from the stream s.

**Parameters**

	s	The stream to extract from.
out	v	<a href="#">Buffer</a> description to copy the array to ( <a href="#">Ipc::Buf_cp_out()</a> ).

**Returns**

The stream `s`.

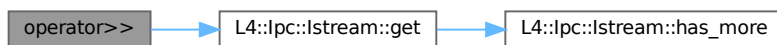
This operator does a copy out of the data into the given buffer.

See `lpc::Buf_in`, `lpc::Buf_cp_in`, and `lpc::Buf_cp_out`.

Definition at line 1151 of file `ipc_stream`.

References [L4::lpc::Istream::get\(\)](#).

Here is the call graph for this function:

**17.190.2.7 operator>>() [3/6]**

```

template<typename T>
L4::Ipc::Istream & operator>> (
    L4::Ipc::Istream & s,
    L4::Ipc::Internal::Buf_in< T > const & v) [inline]
  
```

Extract an array of `T` elements from the stream `s`.

**Parameters**

	<code>s</code>	The stream to extract from.
<code>out</code>	<code>v</code>	Pointer to the extracted array ( <code>ipc_buf_in()</code> ).

**Returns**

The stream `s`.

This operator actually does not copy out the data in the array, but returns a pointer into the message buffer itself. This means that the data is only valid as long as there is no new data inserted into the stream.

**Note**

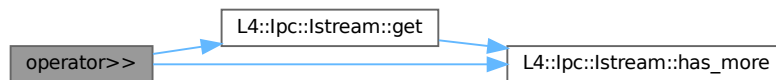
If array does not fit into transmitted words size will be set to zero. Client has to implement check against zero.

See `lpc::Buf_in`, `lpc::Buf_cp_in`, and `lpc::Buf_cp_out`.

Definition at line 1103 of file `ipc_stream`.

References `L4::lpc::Istream::get()`, and `L4::lpc::Istream::has_more()`.

Here is the call graph for this function:

**17.190.2.8 operator>>() [4/6]**

```

template<typename T>
L4::Ipc::Istream & operator>> (
    L4::Ipc::Istream & s,
    L4::Ipc::Msg_ptr< T > const & v) [inline]
  
```

Extract an element of type `T` from the stream `s`.

**Parameters**

	<code>s</code>	The stream to extract from.
<code>out</code>	<code>v</code>	Pointer to the extracted element.

**Returns**

The stream `s`.

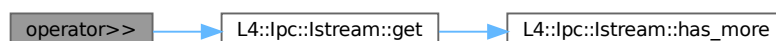
This operator actually does not copy out the data, but returns a pointer into the message buffer itself. This means that the data is only valid as long as there is no new data inserted into the stream.

See `Msg_ptr`.

Definition at line 1130 of file `ipc_stream`.

References `L4::lpc::Istream::get()`.

Here is the call graph for this function:



**17.190.2.9 operator>>() [5/6]**

```
template<typename T>
L4::Ipc::Istream & operator>> (
    L4::Ipc::Istream & s,
    L4::Ipc::Str_cp_in< T > const & v) [inline]
```

Extract a zero-terminated string from the stream.

**Parameters**

	<i>s</i>	The stream to extract from.
out	<i>v</i>	Buffer description to copy the array to (lpc::Str_cp_out()).

**Returns**

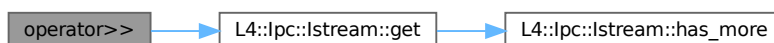
The stream *s*.

This operator does a copy out of the data into the given buffer.

Definition at line 1172 of file `lpc_stream`.

References `L4::lpc::Istream::get()`.

Here is the call graph for this function:

**17.190.2.10 operator>>() [6/6]**

```
L4::Ipc::Istream & operator>> (
    L4::Ipc::Istream & s,
    l4_msgtag_t & v) [inline]
```

Extract the `L4` message tag from the stream *s*.

**Parameters**

	<i>s</i>	The stream to extract from.
out	<i>v</i>	The extracted tag.

**Returns**

The stream `s`.

Definition at line 1078 of file `ipc_stream`.

References `L4::ipc::Istream::tag()`.

Here is the call graph for this function:



## 17.191 ipc\_stream

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *                Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *                Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *                economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #pragma once
00015
00016 #include <l4/sys/ipc.h>
00017 #include <l4/sys/capability>
00018 #include <l4/sys/cxx/ipc_types>
00019 #include <l4/sys/cxx/ipc_varg>
00020 #include <l4/cxx/type_traits>
00021 #include <l4/cxx/minmax>
00022
00023 namespace L4 {
00024 namespace Ipc {
00025
00026 class Ostream;
00027 class Istream;
00028
00029 namespace Internal {
00047 template< typename T >
00048 class Buf_cp_out
00049 {
00050 public:
00057   Buf_cp_out(T const *v, unsigned long size) : _v(v), _s(size) {}
00058
00066   unsigned long size() const { return _s; }
00067
00075   T const *buf() const { return _v; }
00076
00077 private:
00078   friend class Ostream;
00079   T const *_v;
00080   unsigned long _s;
00081 };
00082 }
00083
00099 template< typename T >
00100 Internal::Buf_cp_out<T> buf_cp_out(T const *v, unsigned long size)
00101 { return Internal::Buf_cp_out<T>(v, size); }
00102
00103
00104 namespace Internal {

```



```

00117 template< typename T >
00118 class Buf_cp_in
00119 {
00120 public:
00129   Buf_cp_in(T *v, unsigned long &size) : _v(v), _s(&size) {}
00130
00131   unsigned long &size() const { return *_s; }
00132   T *buf() const { return _v; }
00133
00134 private:
00135   friend class Istream;
00136   T *_v;
00137   unsigned long *_s;
00138 };
00139 }
00140
00158 template< typename T >
00159 Internal::Buf_cp_in<T> buf_cp_in(T *v, unsigned long &size)
00160 { return Internal::Buf_cp_in<T>(v, size); }
00161
00177 template< typename T >
00178 class Str_cp_in
00179 {
00180 public:
00189   Str_cp_in(T *v, unsigned long &size) : _v(v), _s(&size) {}
00190
00191   unsigned long &size() const { return *_s; }
00192   T *buf() const { return _v; }
00193
00194 private:
00195   friend class Istream;
00196   T *_v;
00197   unsigned long *_s;
00198 };
00199
00212 template< typename T >
00213 Str_cp_in<T> str_cp_in(T *v, unsigned long &size)
00214 { return Str_cp_in<T>(v, size); }
00215
00228 template< typename T >
00229 class Msg_ptr
00230 {
00231 private:
00232   T **_p;
00233 public:
00240   explicit Msg_ptr(T *&p) : _p(&p) {}
00241   void set(T *p) const { *_p = p; }
00242 };
00243
00251 template< typename T >
00252 Msg_ptr<T> msg_ptr(T *&p)
00253 { return Msg_ptr<T>(p); }
00254
00255 namespace Internal {
00270 template< typename T >
00271 class Buf_in
00272 {
00273 public:
00280   Buf_in(T *&v, unsigned long &size) : _v(&v), _s(&size) {}
00281
00282   void set_size(unsigned long s) const { *_s = s; }
00283   T *&buf() const { return *_v; }
00284
00285 private:
00286   friend class Istream;
00287   T **_v;
00288   unsigned long *_s;
00289 };
00290 }
00291
00309 template< typename T >
00310 Internal::Buf_in<T> buf_in(T *&v, unsigned long &size)
00311 { return Internal::Buf_in<T>(v, size); }
00312
00313 namespace Utc_b_stream_check
00314 {
00315   static bool check_utcb_data_offset(unsigned sz)
00316   { return sz > sizeof(l4_umword_t) * L4_UTCB_GENERIC_DATA_SIZE; }
00317 }
00318
00319
00334 class Istream
00335 {
00336 public:
00348   Istream(l4_utcb_t *utcb)
00349   : _tag(), _utcb(utcb),

```

```

00350     _current_msg(reinterpret_cast<char*>(l4_utcb_mr_u(utcb)->mr)),
00351     _pos(0), _current_buf(0)
00352 {}
00353
00358 void reset()
00359 {
00360     _pos = 0;
00361     _current_buf = 0;
00362     _current_msg = reinterpret_cast<char*>(l4_utcb_mr_u(utcb)->mr);
00363 }
00364
00368 template< typename T >
00369 bool has_more(unsigned long count = 1)
00370 {
00371     auto const max_bytes = L4_UTCB_GENERIC_DATA_SIZE * sizeof(l4_umword_t);
00372     unsigned apos = cxx::Type_traits<T>::align(_pos);
00373     return (count <= max_bytes / sizeof(T))
00374         && (apos + (sizeof(T) * count)
00375             <= _tag.words() * sizeof(l4_umword_t));
00376 }
00377
00382
00393 template< typename T >
00394 unsigned long get(T *buf, unsigned long elems)
00395 {
00396     if (L4_UNLIKELY(!has_more<T>(elems)))
00397         return 0;
00398
00399     unsigned long size = elems * sizeof(T);
00400     _pos = cxx::Type_traits<T>::align(_pos);
00401
00402     __builtin_memcpy(buf, _current_msg + _pos, size);
00403     _pos += size;
00404     return elems;
00405 }
00406
00407
00413 template< typename T >
00414 void skip(unsigned long elems)
00415 {
00416     if (L4_UNLIKELY(!has_more<T>(elems)))
00417         return;
00418
00419     unsigned long size = elems * sizeof(T);
00420     _pos = cxx::Type_traits<T>::align(_pos);
00421     _pos += size;
00422 }
00423
00438 template< typename T >
00439 unsigned long get(Msg_ptr<T> const &buf, unsigned long elems = 1)
00440 {
00441     if (L4_UNLIKELY(!has_more<T>(elems)))
00442         return 0;
00443
00444     unsigned long size = elems * sizeof(T);
00445     _pos = cxx::Type_traits<T>::align(_pos);
00446
00447     buf.set(reinterpret_cast<T*>(_current_msg + _pos));
00448     _pos += size;
00449     return elems;
00450 }
00451
00452
00463 template< typename T >
00464 bool get(T &v)
00465 {
00466     if (L4_UNLIKELY(!has_more<T>()))
00467     {
00468         v = T();
00469         return false;
00470     }
00471
00472     _pos = cxx::Type_traits<T>::align(_pos);
00473     v = *(reinterpret_cast<T*>(_current_msg + _pos));
00474     _pos += sizeof(T);
00475     return true;
00476 }
00477
00478
00479 bool get(Ipc::Varg *va)
00480 {
00481     Ipc::Varg::Tag t;
00482     if (!has_more<Ipc::Varg::Tag>())
00483     {
00484         va->tag(0);
00485         return 0;
00486     }

```

```

00487     get(t);
00488     va->tag(t);
00489     char const *d;
00490     get(msg_ptr(d), va->length());
00491     va->data(d);
00492
00493     return 1;
00494 }
00495
00505 l4_msgtag_t tag() const { return _tag; }
00506
00507
00517 l4_msgtag_t &tag() { return _tag; }
00518
00520
00525 inline bool put(Rcv_fpage const &);
00526
00531 inline bool put(Small_buf const &);
00532
00533
00538
00548 inline l4_msgtag_t wait(l4_umword_t *src)
00549 { return wait(src, L4_IPC_NEVER); }
00550
00561 inline l4_msgtag_t wait(l4_umword_t *src, l4_timeout_t timeout);
00562
00572 inline l4_msgtag_t receive(l4_cap_idx_t src)
00573 { return receive(src, L4_IPC_NEVER); }
00574
00575 inline l4_msgtag_t receive(l4_cap_idx_t src, l4_timeout_t timeout);
00576
00577
00581 inline l4_utcb_t *utcb() const { return _utcb; }
00582
00583 protected:
00584     l4_msgtag_t _tag;
00585     l4_utcb_t *_utcb;
00586     char *_current_msg;
00587     unsigned _pos;
00588     unsigned char _current_buf;
00589 };
00590
00591 class Istream_copy : public Istream
00592 {
00593 private:
00594     l4_msg_regs_t _mrs;
00595
00596 public:
00597     Istream_copy(Istream const &o) : Istream(o), _mrs(*l4_utcb_mr_u(o.utcb()))
00598     {
00599         // do some reverse mr to utcb trickery
00600         _utcb = reinterpret_cast<l4_utcb_t *>
00601             (reinterpret_cast<l4_addr_t>(&_mrs)
00602              - reinterpret_cast<l4_addr_t>(l4_utcb_mr_u(nullptr)));
00603         _current_msg = reinterpret_cast<char*>(l4_utcb_mr_u(_utcb)->mr);
00604     }
00605
00606 };
00607
00623 class Ostream
00624 {
00625 public:
00629     Ostream(l4_utcb_t *utcb)
00630     : _tag(), _utcb(utcb),
00631       _current_msg(reinterpret_cast<char *>(l4_utcb_mr_u(_utcb)->mr)),
00632       _pos(0), _current_item(0)
00633     {}
00634
00638     void reset()
00639     {
00640         _pos = 0;
00641         _current_item = 0;
00642         _current_msg = reinterpret_cast<char*>(l4_utcb_mr_u(_utcb)->mr);
00643     }
00644
00652
00659 template< typename T >
00660 bool put(T *buf, unsigned long size)
00661 {
00662     size *= sizeof(T);
00663     _pos = cxx::Type_traits<T>::align(_pos);
00664     if (Utc_stream_check::check_utcb_data_offset(_pos + size))
00665         return false;
00666
00667     __builtin_memcpy(_current_msg + _pos, buf, size);
00668     _pos += size;
00669     return true;
00670 }

```

```

00671
00672 template< typename T >
00673 bool put(T const &v)
00674 {
00675     _pos = cxx::Type_traits<T>::align(_pos);
00676     if (Utcstream_check::check_utcb_data_offset(_pos + sizeof(T)))
00677         return false;
00678
00679     *(reinterpret_cast<T*>(_current_msg + _pos)) = v;
00680     _pos += sizeof(T);
00681     return true;
00682 }
00683
00684 int put(Varg const &va)
00685 {
00686     put(va.tag());
00687     put(va.data(), va.length());
00688
00689     return 0;
00690 }
00691
00692 template< typename T >
00693 int put(Varg_t<T> const &va)
00694 { return put(static_cast<Varg const &>(va)); }
00695
00696 l4_msgtag_t tag() const { return _tag; }
00697
00698 l4_msgtag_t &tag() { return _tag; }
00699
00700 inline bool put_snd_item(Snd_fpage const &);
00701
00702 inline l4_msgtag_t send(l4_cap_idx_t dst, long proto = 0, unsigned flags = 0);
00703
00704 inline l4_utcb_t *utcb() const { return _utcb; }
00705 #if 0
00706 unsigned long tell() const
00707 {
00708     unsigned w = l4_bytes_to_mwords(_pos) - _current_item * 2;
00709     _tag = l4_msgtag(0, w, _current_item, 0);
00710 }
00711 #endif
00712 public:
00713     l4_msgtag_t prepare_ipc(long proto = 0, unsigned flags = 0)
00714     {
00715         unsigned w = l4_bytes_to_mwords(_pos) - _current_item * 2;
00716         return l4_msgtag(proto, w, _current_item, flags);
00717     }
00718
00719     // XXX: this is a hack for <l4/sys/cxx/ipc_server> adaption
00720     void set_ipc_params(l4_msgtag_t tag)
00721     {
00722         _pos = (tag.words() + tag.items() * 2) * sizeof(l4_umword_t);
00723         _current_item = tag.items();
00724     }
00725 protected:
00726     l4_msgtag_t _tag;
00727     l4_utcb_t *_utcb;
00728     char *_current_msg;
00729     unsigned _pos;
00730     unsigned char _current_item;
00731 };
00732
00733 class Iostream : public Istream, public Ostream
00734 {
00735 public:
00736     explicit Iostream(l4_utcb_t *utcb)
00737     : Istream(utcb), Ostream(utcb)
00738     {}
00739
00740     // disambiguate those functions
00741     l4_msgtag_t tag() const { return Istream::tag(); }
00742     l4_msgtag_t &tag() { return Istream::tag(); }
00743     l4_utcb_t *utcb() const { return Istream::utcb(); }
00744
00745     void reset()
00746     {
00747         Istream::reset();
00748         Ostream::reset();
00749     }
00750
00751
00752
00753

```

```

00829
00830 using Istream::get;
00831 using Istream::put;
00832 using Ostream::put;
00833
00835
00840
00856 inline l4_msgtag_t call(l4_cap_idx_t dst, l4_timeout_t timeout, long proto = 0);
00857 inline l4_msgtag_t call(l4_cap_idx_t dst, long proto = 0);
00858
00874 inline l4_msgtag_t reply_and_wait(l4_umword_t *src_dst, long proto = 0)
00875 { return reply_and_wait(src_dst, L4_IPC_SEND_TIMEOUT_0, proto); }
00876
00877 inline l4_msgtag_t send_and_wait(l4_cap_idx_t dest, l4_umword_t *src,
00878                                 long proto = 0)
00879 { return send_and_wait(dest, src, L4_IPC_SEND_TIMEOUT_0, proto); }
00880
00897 inline l4_msgtag_t reply_and_wait(l4_umword_t *src_dst,
00898                                 l4_timeout_t timeout, long proto = 0);
00899 inline l4_msgtag_t send_and_wait(l4_cap_idx_t dest, l4_umword_t *src,
00900                                 l4_timeout_t timeout, long proto = 0);
00901 inline l4_msgtag_t reply(l4_timeout_t timeout, long proto = 0);
00902 inline l4_msgtag_t reply(long proto = 0)
00903 { return reply(L4_IPC_SEND_TIMEOUT_0, proto); }
00904
00906 };
00907
00908
00909 inline bool
00910 Ostream::put_snd_item(Snd_fpage const &v)
00911 {
00912     typedef Snd_fpage T;
00913     _pos = cxx::Type_traits<Snd_fpage>::align(_pos);
00914     if (Utcb_stream_check::check_utcb_data_offset(_pos + sizeof(T))
00915         return false;
00916
00917     *(reinterpret_cast<T*>(_current_msg + _pos)) = v;
00918     _pos += sizeof(T);
00919     ++_current_item;
00920     return true;
00921 }
00922
00923
00924 inline bool
00925 Istream::put(Rcv_fpage const &item)
00926 {
00927     unsigned words = item.forward_mappings() ? 3 : 2;
00928     if (_current_buf >= L4_UTCB_GENERIC_BUFFERS_SIZE - words - 1)
00929         return false;
00930
00931     l4_utcb_br_u(_utcb)->bdr &= ~L4_BDR_OFFSET_MASK;
00932
00933     l4_umword_t *buf
00934         = reinterpret_cast<l4_umword_t *>(&l4_utcb_br_u(_utcb)->br[_current_buf]);
00935     *buf++ = item.base_x();
00936     *buf++ = item.data();
00937     if (item.forward_mappings())
00938         *buf++ = item.rcv_task();
00939     _current_buf += words;
00940     return true;
00941 }
00942
00943
00944 inline bool
00945 Istream::put(Small_buf const &item)
00946 {
00947     if (_current_buf >= L4_UTCB_GENERIC_BUFFERS_SIZE - 2)
00948         return false;
00949
00950     l4_utcb_br_u(_utcb)->bdr &= ~L4_BDR_OFFSET_MASK;
00951
00952     reinterpret_cast<Small_buf*>(&l4_utcb_br_u(_utcb)->br[_current_buf]) = item;
00953     _current_buf += 1;
00954     return true;
00955 }
00956
00957
00958 inline l4_msgtag_t
00959 Ostream::send(l4_cap_idx_t dst, long proto, unsigned flags)
00960 {
00961     l4_msgtag_t tag = prepare_ipc(proto, L4_MSGTAG_FLAGS & flags);
00962     return l4_ipc_send(dst, _utcb, tag, L4_IPC_NEVER);
00963 }
00964
00965 inline l4_msgtag_t
00966 Iostream::call(l4_cap_idx_t dst, l4_timeout_t timeout, long label)
00967 {

```

```

00968     l4_msgtag_t tag = prepare_ipc(label);
00969     tag = l4_ipc_call(dst, Ostream::_utcb, tag, timeout);
00970     Istream::tag() = tag;
00971     Istream::_pos = 0;
00972     return tag;
00973 }
00974
00975 inline l4_msgtag_t
00976 Iostream::call(l4_cap_idx_t dst, long label)
00977 { return call(dst, L4_IPC_NEVER, label); }
00978
00979
00980 inline l4_msgtag_t
00981 Iostream::reply_and_wait(l4_umword_t *src_dst, l4_timeout_t timeout, long proto)
00982 {
00983     l4_msgtag_t tag = prepare_ipc(proto);
00984     tag = l4_ipc_reply_and_wait(Ostream::_utcb, tag, src_dst, timeout);
00985     Istream::tag() = tag;
00986     Istream::_pos = 0;
00987     return tag;
00988 }
00989
00990
00991 inline l4_msgtag_t
00992 Iostream::send_and_wait(l4_cap_idx_t dest, l4_umword_t *src,
00993                        l4_timeout_t timeout, long proto)
00994 {
00995     l4_msgtag_t tag = prepare_ipc(proto);
00996     tag = l4_ipc_send_and_wait(dest, Ostream::_utcb, tag, src, timeout);
00997     Istream::tag() = tag;
00998     Istream::_pos = 0;
00999     return tag;
01000 }
01001
01002 inline l4_msgtag_t
01003 Iostream::reply(l4_timeout_t timeout, long proto)
01004 {
01005     l4_msgtag_t tag = prepare_ipc(proto);
01006     tag = l4_ipc_send(L4_INVALID_CAP | L4_SYSF_REPLY, Ostream::_utcb, tag, timeout);
01007     Istream::tag() = tag;
01008     Istream::_pos = 0;
01009     return tag;
01010 }
01011
01012 inline l4_msgtag_t
01013 Istream::wait(l4_umword_t *src, l4_timeout_t timeout)
01014 {
01015     l4_msgtag_t res;
01016     res = l4_ipc_wait(_utcb, src, timeout);
01017     tag() = res;
01018     _pos = 0;
01019     return res;
01020 }
01021
01022
01023 inline l4_msgtag_t
01024 Istream::receive(l4_cap_idx_t src, l4_timeout_t timeout)
01025 {
01026     l4_msgtag_t res;
01027     res = l4_ipc_receive(src, _utcb, timeout);
01028     tag() = res;
01029     _pos = 0;
01030     return res;
01031 }
01032
01033 } // namespace Ipc
01034 } // namespace L4
01035
01044 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, bool &v) { s.get(v); return s; }
01045 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, int &v) { s.get(v); return s; }
01046 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, long int &v) { s.get(v); return s; }
01047 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, long long int &v) { s.get(v); return s; }
01048 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned int &v) { s.get(v); return s; }
01049 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned long int &v) { s.get(v); return s; }
01050 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned long long int &v) { s.get(v);
    return s; }
01051 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, short int &v) { s.get(v); return s; }
01052 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned short int &v) { s.get(v); return s; }
01053 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, char &v) { s.get(v); return s; }
01054 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned char &v) { s.get(v); return s; }
01055 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, signed char &v) { s.get(v); return s; }
01056 inline L4::Ipc::Istream &operator « (L4::Ipc::Istream &s, L4::Ipc::Rcv_fpage const &v) { s.put(v);
    return s; }
01057 inline L4::Ipc::Istream &operator « (L4::Ipc::Istream &s, L4::Ipc::Small_buf const &v) { s.put(v);
    return s; }

```

```

01058 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, L4::Ipc::Snd_fpage &v)
01059 {
01060     l4_umword_t b, d;
01061     s » b » d;
01062     v = L4::Ipc::Snd_fpage(b, d);
01063     return s;
01064 }
01065 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, L4::Ipc::Varg &v)
01066 { s.get(&v); return s; }
01067
01068
01077 inline
01078 L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, l4_msgtag_t &v)
01079 {
01080     v = s.tag();
01081     return s;
01082 }
01083
01101 template< typename T >
01102 inline
01103 L4::Ipc::Istream &operator » (L4::Ipc::Istream &s,
01104                               L4::Ipc::Internal::Buf_in<T> const &v)
01105 {
01106     unsigned long si;
01107     if (s.get(si) && s.has_more<T>(si))
01108         v.set_size(s.get(L4::Ipc::Msg_ptr<T>(v.buf()), si));
01109     else
01110         v.set_size(0);
01111     return s;
01112 }
01113
01128 template< typename T >
01129 inline
01130 L4::Ipc::Istream &operator » (L4::Ipc::Istream &s,
01131                               L4::Ipc::Msg_ptr<T> const &v)
01132 {
01133     s.get(v);
01134     return s;
01135 }
01136
01149 template< typename T >
01150 inline
01151 L4::Ipc::Istream &operator » (L4::Ipc::Istream &s,
01152                               L4::Ipc::Internal::Buf_cp_in<T> const &v)
01153 {
01154     unsigned long sz;
01155     s.get(sz);
01156     v.size() = s.get(v.buf(), cxx::min(v.size(), sz));
01157     return s;
01158 }
01159
01170 template< typename T >
01171 inline
01172 L4::Ipc::Istream &operator » (L4::Ipc::Istream &s,
01173                               L4::Ipc::Str_cp_in<T> const &v)
01174 {
01175     unsigned long sz;
01176     s.get(sz);
01177     unsigned long rsz = s.get(v.buf(), cxx::min(v.size(), sz));
01178     if (rsz < v.size() && v.buf()[rsz - 1])
01179         ++rsz; // add the zero termination behind the received data
01180
01181     if (rsz != 0)
01182         v.buf()[rsz - 1] = 0;
01183
01184     v.size() = rsz;
01185     return s;
01186 }
01187
01188
01197 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, bool v) { s.put(v); return s; }
01198 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, int v) { s.put(v); return s; }
01199 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, long int v) { s.put(v); return s; }
01200 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, long long int v) { s.put(v); return s; }
01201 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned int v) { s.put(v); return s; }
01202 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned long int v) { s.put(v); return s; }
01203 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned long long int v) { s.put(v); return
s; }
01204 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, short int v) { s.put(v); return s; }
01205 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned short int v) { s.put(v); return s;
}
01206 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, char v) { s.put(v); return s; }
01207 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned char v) { s.put(v); return s; }
01208 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, signed char v) { s.put(v); return s; }
01209 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, L4::Ipc::Snd_fpage const &v) {
s.put_snd_item(v); return s; }
01210 template< typename T >

```

```

01211 inline L4::Ipc::Ostream &operator << (L4::Ipc::Ostream &s, L4::Cap<T> const &v)
01212 { s << L4::Ipc::Snd_fpage(v.fpage()); return s; }
01213
01214 inline L4::Ipc::Ostream &operator << (L4::Ipc::Ostream &s, L4::Ipc::Varg const &v)
01215 { s.put(v); return s; }
01216 template< typename T >
01217 inline L4::Ipc::Ostream &operator << (L4::Ipc::Ostream &s, L4::Ipc::Varg_t<T> const &v)
01218 { s.put(v); return s; }
01219
01231 inline
01232 L4::Ipc::Ostream &operator << (L4::Ipc::Ostream &s, l4_msgtag_t const &v)
01233 {
01234     s.tag() = v;
01235     return s;
01236 }
01237
01246 template< typename T >
01247 inline
01248 L4::Ipc::Ostream &operator << (L4::Ipc::Ostream &s,
01249                               L4::Ipc::Internal::Buf_cp_out<T> const &v)
01250 {
01251     s.put(v.size());
01252     s.put(v.buf(), v.size());
01253     return s;
01254 }
01255
01268 inline
01269 L4::Ipc::Ostream &operator << (L4::Ipc::Ostream &s, char const *v)
01270 {
01271     unsigned long l = __builtin_strlen(v) + 1;
01272     s.put(l);
01273     s.put(v, l);
01274     return s;
01275 }
01276
01277 namespace L4 { namespace Ipc {
01287 template< typename T >
01288 inline
01289 T read(Istream &s) { T t; s >> t; return t; }
01290
01291 } // namespace Ipc
01292 } // namespace L4

```

## 17.192 ipc\_timeout\_queue

```

00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Steffen Liebergeld <steffen.liebergeld@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/cxx/hlist>
00010 #include <l4/sys/cxx/ipc_server_loop>
00011
00012 namespace L4 { namespace Ipc_svr {
00013
00018 class Timeout : public cxx::H_list_item
00019 {
00020     friend class Timeout_queue;
00021 public:
00023     Timeout() : _timeout(0) {}
00024
00026     virtual ~Timeout() = 0;
00027
00034     virtual void expired() = 0;
00035
00042     l4_kernel_clock_t timeout() const
00043     { return _timeout; }
00044
00045 private:
00046     l4_kernel_clock_t _timeout;
00047 };
00048
00049 inline Timeout::~~Timeout() {}
00050
00055 class Timeout_queue
00056 {
00057 public:
00059     typedef L4::Ipc_svr::Timeout Timeout;
00060
00065     l4_kernel_clock_t next_timeout() const

```



```

00066 {
00067     if (auto e = _timeouts.front())
00068         return e->timeout();
00069     return 0;
00070 }
00071
00072
00081 bool timeout_expired(l4_kernel_clock_t now) const
00082 {
00083     l4_kernel_clock_t next = next_timeout();
00084     return (next != 0) && (next <= now);
00085 }
00086
00091 void handle_expired_timeouts(l4_kernel_clock_t now)
00092 {
00093     while (!_timeouts.empty())
00094     {
00095         Queue::Iterator top = _timeouts.begin();
00096         if ((*top)->_timeout > now)
00097             return;
00098
00099         Timeout *t = *top;
00100         top = _timeouts.erase(top);
00101         t->expired();
00102     }
00103 }
00104
00111 void add(Timeout *timeout, l4_kernel_clock_t time)
00112 {
00113     timeout->_timeout = time;
00114     Queue::Iterator i = _timeouts.begin();
00115     while (i != _timeouts.end() && (*i)->timeout() < time)
00116         ++i;
00117     _timeouts.insert_before(timeout, i);
00118 }
00119
00126 void remove(Timeout *timeout)
00127 {
00128     _timeouts.remove(timeout);
00129 }
00130
00131 private:
00132     typedef cxx::H_list<Timeout> Queue;
00133     Queue _timeouts;
00134 };
00135
00149 template< typename HOOKS, typename BR_MAN = Br_manager_no_buffers >
00150 class Timeout_queue_hooks : public BR_MAN
00151 {
00152     l4_kernel_clock_t _now()
00153     { return static_cast<HOOKS*>(this)->now(); }
00154
00155     unsigned _timeout_br()
00156     { return this->first_free_br(); }
00157
00158 public:
00160     l4_timeout_t timeout()
00161     {
00162         l4_kernel_clock_t t = queue.next_timeout();
00163         if (t)
00164             return l4_timeout(L4_IPC_TIMEOUT_0, l4_timeout_abs(t, _timeout_br()));
00165         return L4_IPC_SEND_TIMEOUT_0;
00166     }
00167
00169     void setup_wait(l4_utcb_t *utcb, L4::Ipc_svr::Reply_mode mode)
00170     {
00171         // we must handle the timer only when called after a possible reply
00172         // otherwise we probably destroy the reply message.
00173         if (mode == L4::Ipc_svr::Reply_separate)
00174         {
00175             l4_kernel_clock_t now = _now();
00176             if (queue.timeout_expired(now))
00177                 queue.handle_expired_timeouts(now);
00178         }
00179
00180         BR_MAN::setup_wait(utcb, mode);
00181     }
00182
00184     L4::Ipc_svr::Reply_mode before_reply(l4_msgtag_t, l4_utcb_t *)
00185     {
00186         // split up reply and wait when a timeout has expired
00187         if (queue.timeout_expired(_now()))
00188             return L4::Ipc_svr::Reply_separate;
00189         return L4::Ipc_svr::Reply_compound;
00190     }
00191

```

```

00202 int add_timeout(Timeout *timeout, l4_kernel_clock_t time) override
00203 {
00204     queue.add(timeout, time);
00205     return 0;
00206 }
00207
00215 int remove_timeout(Timeout *timeout) override
00216 {
00217     queue.remove(timeout);
00218     return 0;
00219 }
00220
00221 Timeout_queue queue;
00222 };
00223
00224 }}

```

## 17.193 l4/cxx/l4iostream File Reference

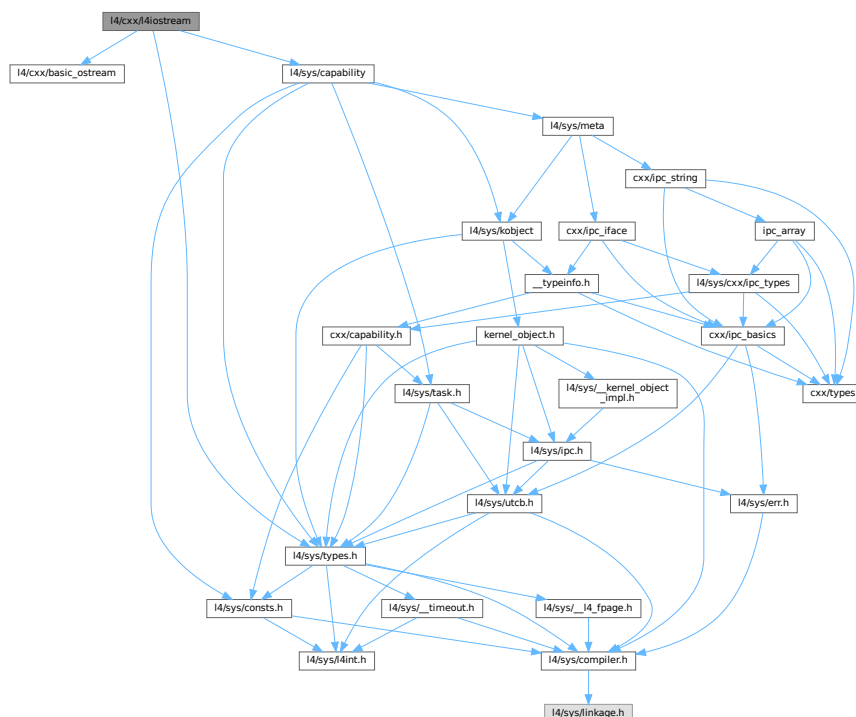
[L4](#) IO stream.

```

#include <l4/cxx/basic_ostream>
#include <l4/sys/types.h>
#include <l4/sys/capability>

```

Include dependency graph for l4iostream:



### 17.193.1 Detailed Description

[L4](#) IO stream.

Definition in file [l4iostream](#).

## 17.194 l4iostream

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00004  * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *     economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include <l4/cxx/basic_ostream>
00012 #include <l4/sys/types.h>
00013 #include <l4/sys/capability>
00014
00015 inline
00016 L4::BasicOStream &operator << (L4::BasicOStream &o, l4_msgtag_t const &tag)
00017 {
00018     L4::IOBackend::Mode m = o.be_mode();
00019     o << "[l=" << L4::dec << tag.label() << "; w=" << tag.words() << "; i="
00020         << tag.items() << "];";
00021     o.be_mode(m);
00022     return o;
00023 }
00024
00025 template<typename T>
00026 inline
00027 L4::BasicOStream &operator << (L4::BasicOStream &o, L4::Cap<T> const &cap)
00028 {
00029     o << "[C:" << L4::n_hex(cap.cap()) << "];";
00030     return o;
00031 }
00032
00033
00034
00035

```

## 17.195 l4/cxx/l4types.h File Reference

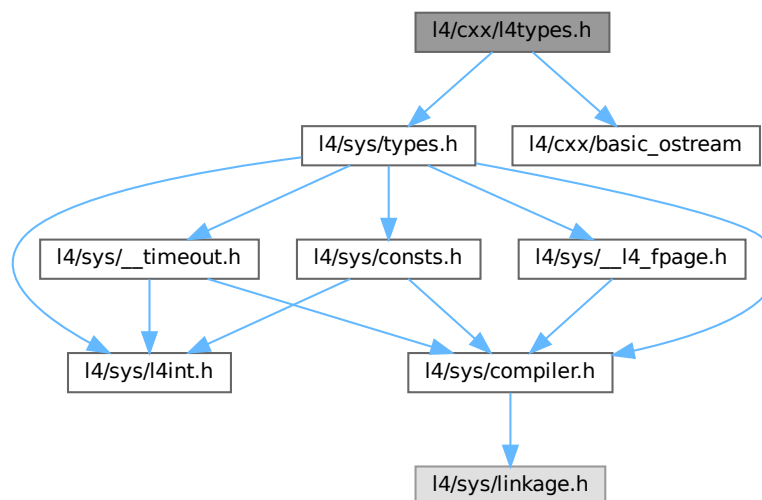
L4 Types.

```

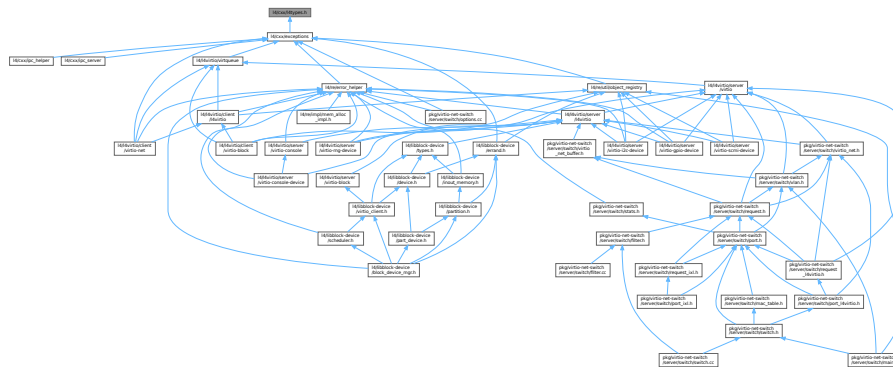
#include <l4/sys/types.h>
#include <l4/cxx/basic_ostream>

```

Include dependency graph for l4types.h:



This graph shows which files directly or indirectly include this file:



## 17.195.1 Detailed Description

[L4](#) Types.

Definition in file [l4types.h](#).

## 17.196 l4types.h

[Go to the documentation of this file.](#)

```
00001
00005 /*
00006  * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00013 #include <l4/sys/types.h>
00014 #include <l4/cxx/basic_ostream>
```

## 17.197 list

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 #include <l4/cxx/type_traits>
00012 #include <l4/cxx/std_alloc>
00013 #include <l4/cxx/std_ops>
00014
00015 namespace cxx {
00016 /*
00017  * Classes: List_item, List<D, Alloc>
00018  */
00019
00020 class List_item
00021 {
00022 public:
00023     class Iter
00024     {
```

```

00036 public:
00037     Iter(List_item *c, List_item *f) noexcept : _c(c), _f(f) {}
00038     Iter(List_item *f = 0) noexcept : _c(f), _f(f) {}
00039
00040     List_item *operator * () const noexcept { return _c; }
00041     List_item *operator -> () const noexcept { return _c; }
00042     Iter &operator ++ () noexcept
00043     {
00044         if (!_f)
00045             _c = 0;
00046         else
00047             _c = _c->get_next_item();
00048
00049         if (_c == _f)
00050             _c = 0;
00051
00052         return *this;
00053     }
00054
00055     Iter operator ++ (int) noexcept
00056     { Iter o = *this; operator ++ (); return o; }
00057
00058     Iter &operator -- () noexcept
00059     {
00060         if (!_f)
00061             _c = 0;
00062         else
00063             _c = _c->get_prev_item();
00064
00065         if (_c == _f)
00066             _c = 0;
00067
00068         return *this;
00069     }
00070
00071     Iter operator -- (int) noexcept
00072     { Iter o = *this; operator -- (); return o; }
00073
00075     List_item *remove_me() noexcept
00076     {
00077         if (!_c)
00078             return 0;
00079
00080         List_item *l = _c;
00081         operator ++ ();
00082         l->remove_me();
00083
00084         if (_f == l)
00085             _f = _c;
00086
00087         return l;
00088     }
00089
00090 private:
00091     List_item *_c, *_f;
00092 };
00093
00107 template< typename T, bool Poly = false>
00108 class T_iter : public Iter
00109 {
00110 private:
00111     static bool const P = !Conversion<const T*, const List_item *>::exists
00112         || Poly;
00113
00114     static List_item *cast_to_li(T *i, Int_to_type<true>) noexcept
00115     { return dynamic_cast<List_item*>(i); }
00116
00117     static List_item *cast_to_li(T *i, Int_to_type<false>) noexcept
00118     { return i; }
00119
00120     static T *cast_to_type(List_item *i, Int_to_type<true>) noexcept
00121     { return dynamic_cast<T*>(i); }
00122
00123     static T *cast_to_type(List_item *i, Int_to_type<false>) noexcept
00124     { return static_cast<T*>(i); }
00125
00126 public:
00127
00128     template< typename O >
00129     explicit T_iter(T_iter<O> const &o) noexcept
00130         : Iter(o) { dynamic_cast<T*>(&o); }
00131
00132     //T_iter(CListItem *f) : Iter(f) {}
00133     T_iter(T *f = 0) noexcept : Iter(cast_to_li(f, Int_to_type<P>())) {}
00134     T_iter(T *c, T *f) noexcept
00135         : Iter(cast_to_li(c, Int_to_type<P>()),
00136             cast_to_li(f, Int_to_type<P>()))

```

```

00137     {}
00138
00139     inline T *operator * () const noexcept
00140     { return cast_to_type(Iter::operator * (), Int_to_type<P>()); }
00141     inline T *operator -> () const noexcept
00142     { return operator * (); }
00143
00144     T_iter<T, Poly> operator ++ (int) noexcept
00145     { T_iter<T, Poly> o = *this; Iter::operator ++ (); return o; }
00146     T_iter<T, Poly> operator -- (int) noexcept
00147     { T_iter<T, Poly> o = *this; Iter::operator -- (); return o; }
00148     T_iter<T, Poly> &operator ++ () noexcept
00149     { Iter::operator ++ (); return *this; }
00150     T_iter<T, Poly> &operator -- () noexcept
00151     { Iter::operator -- (); return *this; }
00152     inline T *remove_me() noexcept;
00153 };
00154
00155 List_item() noexcept : _n(this), _p(this) {}
00156
00157 protected:
00158     List_item(List_item const &) noexcept : _n(this), _p(this) {}
00159
00160 public:
00162     List_item *get_prev_item() const noexcept { return _p; }
00163
00165     List_item *get_next_item() const noexcept { return _n; }
00166
00168     void insert_prev_item(List_item *p) noexcept
00169     {
00170         p->_p->_n = this;
00171         List_item *pr = p->_p;
00172         p->_p = _p;
00173         _p->_n = p;
00174         _p = pr;
00175     }
00176
00178     void insert_next_item(List_item *p) noexcept
00179     {
00180         p->_p->_n = _n;
00181         p->_p = this;
00182         _n->_p = p;
00183         _n = p;
00184     }
00185
00187     void remove_me() noexcept
00188     {
00189         if (_p != this)
00190         {
00191             _p->_n = _n;
00192             _n->_p = _p;
00193         }
00194         _p = _n = this;
00195     }
00196
00205     template< typename C, typename N >
00206     static inline C *push_back(C *head, N *p) noexcept;
00207
00216     template< typename C, typename N >
00217     static inline C *push_front(C *head, N *p) noexcept;
00218
00227     template< typename C, typename N >
00228     static inline C *remove(C *head, N *p) noexcept;
00229
00230 private:
00231     List_item *_n, *_p;
00232 };
00233
00234
00235 /* IMPLEMENTATION -----*/
00236 template< typename C, typename N >
00237 C *List_item::push_back(C *h, N *p) noexcept
00238 {
00239     if (!p)
00240         return h;
00241     if (!h)
00242         return p;
00243     h->insert_prev_item(p);
00244     return h;
00245 }
00246
00247 template< typename C, typename N >
00248 C *List_item::push_front(C *h, N *p) noexcept
00249 {
00250     if (!p)
00251         return h;
00252     if (h)

```

```

00253     h->insert_prev_item(p);
00254     return p;
00255 }
00256
00257 template< typename C, typename N >
00258 C *List_item::remove(C *h, N *p) noexcept
00259 {
00260     if (!p)
00261         return h;
00262     if (!h)
00263         return 0;
00264     if (h == p)
00265     {
00266         if (p == p->_n)
00267             h = 0;
00268         else
00269             h = static_cast<C*>(p->_n);
00270     }
00271     p->remove_me();
00272     return h;
00273 }
00274
00275 template< typename T, bool Poly >
00276 inline
00277 T *List_item::T_iter<T, Poly>::remove_me() noexcept
00278 { return cast_to_type(Iter::remove_me(), Int_to_type<P>()); }
00279
00280 template< typename T >
00281 class T_list_item : public List_item
00282 {
00283 public:
00284     T *next() const { return static_cast<T*>(List_item::get_next_item()); }
00285     T *prev() const { return static_cast<T*>(List_item::get_prev_item()); }
00286 };
00287
00288 template< typename LI >
00289 class L_list
00290 {
00291 private:
00292     LI *_h;
00293 public:
00294     L_list() : _h(0) {}
00295
00296     void push_front(LI *e) { _h = LI::push_front(_h, e); }
00297     void push_back(LI *e) { _h = LI::push_back(_h, e); }
00298     void insert_before(LI *e, LI *p)
00299     {
00300         p->insert_prev_item(e);
00301         if (_h == p)
00302             _h = e;
00303     }
00304     void insert_after(LI *e, LI *p) { p->insert_next_item(e); }
00305
00306     void remove(LI *e)
00307     { _h = LI::remove(_h, e); }
00308
00309     LI *head() const { return _h; }
00310 };
00311
00312 template< typename D, template<typename A> class Alloc = New_allocator >
00313 class List
00314 {
00315 private:
00316     class E : public List_item
00317     {
00318     public:
00319         E(D const &d) noexcept : data(d) {}
00320         D data;
00321     };
00322
00323     class Node : private E
00324     {};
00325
00326     typedef Alloc<Node> Node_alloc;
00327
00328     class Iter
00329     {
00330     private:
00331         List_item::T_iter<E> _i;
00332     public:

```

```

00349     Iter(E *e) noexcept : _i(e) {}
00350
00351     D &operator * () const noexcept { return (*_i)->data; }
00352     D &operator -> () const noexcept { return (*_i)->data; }
00353
00354     Iter operator ++ (int) noexcept
00355     { Iter o = *this; operator ++ (); return o; }
00356     Iter operator -- (int) noexcept
00357     { Iter o = *this; operator -- (); return o; }
00358     Iter &operator ++ () noexcept { ++_i; return *this; }
00359     Iter &operator -- () noexcept { --_i; return *this; }
00360
00362     operator E* () const noexcept { return *_i; }
00363 };
00364
00365     List(Alloc<Node> const &a = Alloc<Node>()) noexcept : _h(0), _l(0), _a(a) {}
00366
00368     void push_back(D const &d) noexcept
00369     {
00370         void *n = _a.alloc();
00371         if (!n) return;
00372         _h = E::push_back(_h, new (n) E(d));
00373         ++_l;
00374     }
00375
00377     void push_front(D const &d) noexcept
00378     {
00379         void *n = _a.alloc();
00380         if (!n) return;
00381         _h = E::push_front(_h, new (n) E(d));
00382         ++_l;
00383     }
00384
00386     void remove(Iter const &i) noexcept
00387     { E *e = i; _h = E::remove(_h, e); --_l; _a.free(e); }
00388
00390     unsigned long size() const noexcept { return _l; }
00391
00393     D const &operator [] (unsigned long idx) const noexcept
00394     { Iter i = _h; for (; idx && *i; ++i, --idx) { } return *i; }
00395
00397     D &operator [] (unsigned long idx) noexcept
00398     { Iter i = _h; for (; idx && *i; ++i, --idx) { } return *i; }
00399
00401     Iter items() noexcept { return Iter(_h); }
00402
00403 private:
00404     E *_h;
00405     unsigned _l;
00406     Alloc<Node> _a;
00407 };
00408
00409
00410 };
00411

```

## 17.198 list\_alloc

```

00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 #include <l4/cxx/arith>
00013 #include <l4/cxx/minmax>
00014 #include <l4/sys/consts.h>
00015
00016 namespace cxx {
00017
00021 class List_alloc
00022 {
00023 private:
00024     friend class List_alloc_sanity_guard;
00025
00026     struct Mem_block
00027     {
00028         Mem_block *next;

```



```

00029     unsigned long size;
00030 };
00031
00032 Mem_block *_first;
00033
00034 inline void check_overlap(void *, unsigned long );
00035 inline void sanity_check_list(char const *, char const *);
00036 inline void merge();
00037
00038 public:
00039
00046 List_alloc() : _first(0) {}
00047
00060 inline void free(void *block, unsigned long size, bool initial_free = false);
00061
00076 inline void *alloc(unsigned long size, unsigned long align,
00077                   unsigned long lower = 0, unsigned long upper = ~0UL);
00078
00099 inline void *alloc_max(unsigned long min, unsigned long *max,
00100                       unsigned long align, unsigned granularity,
00101                       unsigned long lower = 0, unsigned long upper = ~0UL);
00102
00108 inline unsigned long avail();
00109
00110 template <typename DBG>
00111 void dump_free_list(DBG &out);
00112 };
00113
00114 #if !defined (CXX_LIST_ALLOC_SANITY)
00115 class List_alloc_sanity_guard
00116 {
00117 public:
00118     List_alloc_sanity_guard(List_alloc *, char const *)
00119     {}
00120 };
00121
00122
00123 void
00124 List_alloc::check_overlap(void *, unsigned long )
00125 {}
00126
00127 void
00128 List_alloc::sanity_check_list(char const *, char const *)
00129 {}
00130
00131
00132 #else
00133
00134 class List_alloc_sanity_guard
00135 {
00136 private:
00137     List_alloc *a;
00138     char const *func;
00139
00140 public:
00141     List_alloc_sanity_guard(List_alloc *a, char const *func)
00142     : a(a), func(func)
00143     { a->sanity_check_list(func, "entry"); }
00144
00145     ~List_alloc_sanity_guard()
00146     { a->sanity_check_list(func, "exit"); }
00147 };
00148
00149 void
00150 List_alloc::check_overlap(void *b, unsigned long s)
00151 {
00152     unsigned long const mb_align = (1UL << arith::Ld<sizeof(Mem_block)>::value) - 1;
00153     if ((unsigned long)b & mb_align)
00154     {
00155         L4::cerr << "List_alloc(FATAL): trying to free unaligned memory: "
00156                 << b << " align=" << arith::Ld<sizeof(Mem_block)>::value << "\n";
00157     }
00158
00159     Mem_block *c = _first;
00160     for (; c ; c = c->next)
00161     {
00162         unsigned long x_s = (unsigned long)b;
00163         unsigned long x_e = x_s + s;
00164         unsigned long b_s = (unsigned long)c;
00165         unsigned long b_e = b_s + c->size;
00166
00167         if ((x_s >= b_s && x_s < b_e)
00168             || (x_e > b_s && x_e <= b_e)
00169             || (b_s >= x_s && b_s < x_e)
00170             || (b_e > x_s && b_e <= x_e))
00171         {
00172             L4::cerr << "List_alloc(FATAL): trying to free memory that "

```

```

00173         "is already free: \n ["
00174         << (void*)x_s << '-' << (void*)x_e << ") overlaps ["
00175         << (void*)b_s << '-' << (void*)b_e << ")\n";
00176     }
00177 }
00178 }
00179
00180 void
00181 List_alloc::sanity_check_list(char const *func, char const *info)
00182 {
00183     Mem_block *c = _first;
00184     for (; c ; c = c->next)
00185     {
00186         if (c->next)
00187         {
00188             if (c >= c->next)
00189             {
00190                 L4::cerr << "List_alloc(FATAL): " << func << '(' << info
00191                 << "): list order violation\n";
00192             }
00193
00194             if (((unsigned long)c) + c->size > (unsigned long)c->next)
00195             {
00196                 L4::cerr << "List_alloc(FATAL): " << func << '(' << info
00197                 << "): list order violation\n";
00198             }
00199         }
00200     }
00201 }
00202
00203 #endif
00204
00205 void
00206 List_alloc::merge()
00207 {
00208     List_alloc_sanity_guard __attribute__((unused)) guard(this, __func__);
00209     Mem_block *c = _first;
00210     while (c && c->next)
00211     {
00212         unsigned long f_start = reinterpret_cast<unsigned long>(c);
00213         unsigned long f_end   = f_start + c->size;
00214         unsigned long n_start = reinterpret_cast<unsigned long>(c->next);
00215
00216         if (f_end == n_start)
00217         {
00218             c->size += c->next->size;
00219             c->next = c->next->next;
00220             continue;
00221         }
00222
00223         c = c->next;
00224     }
00225 }
00226
00227 void
00228 List_alloc::free(void *block, unsigned long size, bool initial_free)
00229 {
00230     List_alloc_sanity_guard __attribute__((unused)) guard(this, __func__);
00231
00232     unsigned long const mb_align = (1UL << arith::Ld<sizeof(Mem_block)>::value) - 1;
00233
00234     if (initial_free)
00235     {
00236         // enforce alignment constraint on initial memory
00237         unsigned long nblock = (reinterpret_cast<unsigned long>(block) + mb_align)
00238                                & ~mb_align;
00239         size = (size - (nblock - reinterpret_cast<unsigned long>(block)))
00240                & ~mb_align;
00241         block = reinterpret_cast<void*>(nblock);
00242     }
00243     else
00244         // blow up size to the minimum aligned size
00245         size = (size + mb_align) & ~mb_align;
00246
00247     check_overlap(block, size);
00248
00249     Mem_block **c = &_first;
00250     Mem_block *next = 0;
00251
00252     if (*c)
00253     {
00254         while (*c && *c < block)
00255             c = &(*c)->next;
00256
00257         next = *c;
00258     }
00259 }

```

```

00260  *c = reinterpret_cast<Mem_block*>(block);
00261
00262  (*c)->next = next;
00263  (*c)->size = size;
00264
00265  merge();
00266 }
00267
00268 void *
00269 List_alloc::alloc_max(unsigned long min, unsigned long *max, unsigned long align,
00270                      unsigned granularity, unsigned long lower,
00271                      unsigned long upper)
00272 {
00273   List_alloc_sanity_guard __attribute__((unused)) guard(this, __func__);
00274
00275   unsigned char const mb_bits = arith::Ld<sizeof(Mem_block)>::value;
00276   unsigned long const mb_align = (1UL << mb_bits) - 1;
00277
00278   // blow minimum up to at least the minimum aligned size of a Mem_block
00279   min = 14_round_size(min, mb_bits);
00280   // truncate maximum to at least the size of a Mem_block
00281   *max = 14_trunc_size(*max, mb_bits);
00282   // truncate maximum size according to granularity
00283   *max = *max & ~(granularity - 1UL);
00284
00285   if (min > *max)
00286     return 0;
00287
00288   unsigned long almask = align ? (align - 1UL) : 0;
00289
00290   // minimum alignment is given by the size of a Mem_block
00291   if (almask < mb_align)
00292     almask = mb_align;
00293
00294   Mem_block **c = &_first;
00295   Mem_block **fit = 0;
00296   unsigned long max_fit = 0;
00297   unsigned long a_lower = (lower + almask) & ~almask;
00298
00299   for (; *c; c = &(*c)->next)
00300   {
00301     // address of free memory block
00302     unsigned long n_start = reinterpret_cast<unsigned long>(*c);
00303
00304     // block too small, next
00305     // XXX: maybe we can skip this and just do the test below
00306     if ((*c)->size < min)
00307       continue;
00308
00309     // block outside region, next
00310     if (upper < n_start || a_lower > n_start + (*c)->size)
00311       continue;
00312
00313     // aligned start address within the free block
00314     unsigned long a_start = (n_start + almask) & ~almask;
00315
00316     // check if aligned start address is behind the block, next
00317     if (a_start - n_start >= (*c)->size)
00318       continue;
00319
00320     a_start = a_start < a_lower ? a_lower : a_start;
00321
00322     // end address would overflow, next
00323     if (min > ~0UL - a_start)
00324       continue;
00325
00326     // block outside region, next
00327     if (a_start + min - 1UL > upper)
00328       continue;
00329
00330     // remaining size after subtracting the padding for the alignment
00331     unsigned long r_size = (*c)->size - a_start + n_start;
00332
00333     // upper limit can limit maximum size
00334     if (a_start + r_size - 1UL > upper)
00335       r_size = upper - a_start + 1UL;
00336
00337     // round down according to granularity
00338     r_size &= ~(granularity - 1UL);
00339
00340     // block too small
00341     if (r_size < min)
00342       continue;
00343
00344     if (r_size >= *max)
00345     {
00346       fit = c;

```

```

00347         max_fit = *max;
00348         break;
00349     }
00350
00351     if (r_size > max_fit)
00352     {
00353         max_fit = r_size;
00354         fit = c;
00355     }
00356 }
00357
00358 if (fit)
00359 {
00360     unsigned long n_start = reinterpret_cast<unsigned long>(*fit);
00361     unsigned long a_lower = (lower + almask) & ~almask;
00362     unsigned long a_start = (n_start + almask) & ~almask;
00363     a_start = a_start < a_lower ? a_lower : a_start;
00364     unsigned long r_size = (*fit)->size - a_start + n_start;
00365
00366     if (a_start > n_start)
00367     {
00368         (*fit)->size -= r_size;
00369         fit = &(*fit)->next;
00370     }
00371     else
00372         *fit = (*fit)->next;
00373
00374     *max = max_fit;
00375     if (r_size == max_fit)
00376         return reinterpret_cast<void *>(a_start);
00377
00378     Mem_block *m = reinterpret_cast<Mem_block*>(a_start + max_fit);
00379     m->next = *fit;
00380     m->size = r_size - max_fit;
00381     *fit = m;
00382     return reinterpret_cast<void *>(a_start);
00383 }
00384
00385 return 0;
00386 }
00387
00388 void *
00389 List_alloc::alloc(unsigned long size, unsigned long align, unsigned long lower,
00390                  unsigned long upper)
00391 {
00392     List_alloc_sanity_guard __attribute__((unused)) guard(this, __func__);
00393
00394     unsigned long const mb_align
00395         = (1UL << arith::Ld<sizeof(Mem_block)>::value) - 1;
00396
00397     // blow up size to the minimum aligned size
00398     size = (size + mb_align) & ~mb_align;
00399
00400     unsigned long almask = align ? (align - 1UL) : 0;
00401
00402     // minimum alignment is given by the size of a Mem_block
00403     if (almask < mb_align)
00404         almask = mb_align;
00405
00406     Mem_block **c = &_first;
00407     unsigned long a_lower = (lower + almask) & ~almask;
00408
00409     for (; *c; c=&(*c)->next)
00410     {
00411         // address of free memory block
00412         unsigned long n_start = reinterpret_cast<unsigned long>(*c);
00413
00414         // block too small, next
00415         // XXX: maybe we can skip this and just do the test below
00416         if ((*c)->size < size)
00417             continue;
00418
00419         // block outside region, next
00420         if (upper < n_start || a_lower > n_start + (*c)->size)
00421             continue;
00422
00423         // aligned start address within the free block
00424         unsigned long a_start = (n_start + almask) & ~almask;
00425
00426         // block too small after alignment, next
00427         if (a_start - n_start >= (*c)->size)
00428             continue;
00429
00430         a_start = a_start < a_lower ? a_lower : a_start;
00431
00432         // end address would overflow, next
00433         if (size > ~0UL - a_start)

```

```

00434         continue;
00435
00436         // block outside region, next
00437         if (a_start + size - 1UL > upper)
00438             continue;
00439
00440         // remaining size after subtracting the padding
00441         // for the alignment
00442         unsigned long r_size = (*c)->size - a_start + n_start;
00443
00444         // block too small
00445         if (r_size < size)
00446             continue;
00447
00448         if (a_start > n_start)
00449         {
00450             // have free space before the allocated block
00451             // shrink the block and set c to the next pointer of that
00452             // block
00453             (*c)->size -= r_size;
00454             c = &(*c)->next;
00455         }
00456         else
00457             // drop the block, c remains the next pointer of the
00458             // previous block
00459             *c = (*c)->next;
00460
00461         // allocated the whole remaining space
00462         if (r_size == size)
00463             return reinterpret_cast<void*>(a_start);
00464
00465         // add a new free block behind the allocated block
00466         Mem_block *m = reinterpret_cast<Mem_block*>(a_start + size);
00467         m->next = *c;
00468         m->size = r_size - size;
00469         *c = m;
00470         return reinterpret_cast<void *>(a_start);
00471     }
00472
00473     return 0;
00474 }
00475
00476 unsigned long
00477 List_alloc::avail()
00478 {
00479     List_alloc_sanity_guard __attribute__((unused)) guard(this, __FUNCTION__);
00480     Mem_block *c = _first;
00481     unsigned long a = 0;
00482     while (c)
00483     {
00484         a += c->size;
00485         c = c->next;
00486     }
00487
00488     return a;
00489 }
00490
00491 template <typename DBG>
00492 void
00493 List_alloc::dump_free_list(DBG &out)
00494 {
00495     Mem_block *c = _first;
00496     while (c)
00497     {
00498         unsigned sz;
00499         const char *unit;
00500
00501         if (c->size < 1024)
00502         {
00503             sz = c->size;
00504             unit = "Byte";
00505         }
00506         else if (c->size < 1 « 20)
00507         {
00508             sz = c->size » 10;
00509             unit = "KB";
00510         }
00511         else
00512         {
00513             sz = c->size » 20;
00514             unit = "MB";
00515         }
00516
00517         out.printf("%12p - %12p (%u %s)\n", c,
00518             reinterpret_cast<char *>(c) + c->size - 1, sz, unit);
00519
00520         c = c->next;

```

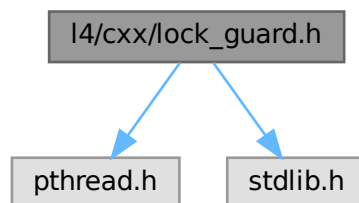
```
00521     }  
00522 }  
00523  
00524 }
```

## 17.199 I4/cxx/lock\_guard.h File Reference

Lock guard implementation.

```
#include <pthread.h>  
#include <stdlib.h>
```

Include dependency graph for lock\_guard.h:



### Data Structures

- class [L4::Lock\\_guard](#)

*Basic lock guard implementation that prevents forgotten unlocks on exit paths from a method or a block of code.*

### Namespaces

- namespace [L4](#)

[L4](#) *low-level kernel interface.*

### 17.199.1 Detailed Description

Lock guard implementation.

Definition in file [lock\\_guard.h](#).

## 17.200 lock\_guard.h

[Go to the documentation of this file.](#)

```

00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2025 Kernkonzept GmbH.
00004  * Author(s): Martin Decky <martin.decky@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00013
00014 #pragma once
00015
00016 #include <pthread.h>
00017 #include <stdlib.h>
00018
00019 namespace L4 {
00020
00044 class Lock_guard
00045 {
00046 public:
00047     Lock_guard() = delete;
00048     Lock_guard(const Lock_guard &) = delete;
00049     Lock_guard &operator=(const Lock_guard &) = delete;
00050
00059     explicit Lock_guard(pthread_mutex_t &lock) : _lock(&lock)
00060     {
00061         _status = pthread_mutex_lock(_lock);
00062     }
00063
00071     Lock_guard(Lock_guard &&guard) : _lock(guard._lock), _status(guard._status)
00072     {
00073         guard.release();
00074     }
00075
00090     Lock_guard &operator=(Lock_guard &&guard)
00091     {
00092         // Unlock the currently associated mutex (if any).
00093         reset();
00094
00095         // Move the state from the other guard.
00096         _lock = guard._lock;
00097         _status = guard._status;
00098
00099         // Release the mutex from the other guard.
00100         guard.release();
00101
00102         return *this;
00103     }
00104
00110     int status() const
00111     {
00112         return _status;
00113     }
00114
00126     ~Lock_guard()
00127     {
00128         reset();
00129     }
00130
00131 private:
00137     void release()
00138     {
00139         _lock = nullptr;
00140     }
00141
00150     void reset()
00151     {
00152         // No mutex might be associated with this lock guard only if the mutex has
00153         // been moved to a different lock guard.
00154         if (_lock)
00155         {
00156             _status = pthread_mutex_unlock(_lock);
00157             release();
00158         }
00159     }
00160
00161     pthread_mutex_t *_lock;
00162     int _status;
00163 };
00164
00165 } // namespace L4

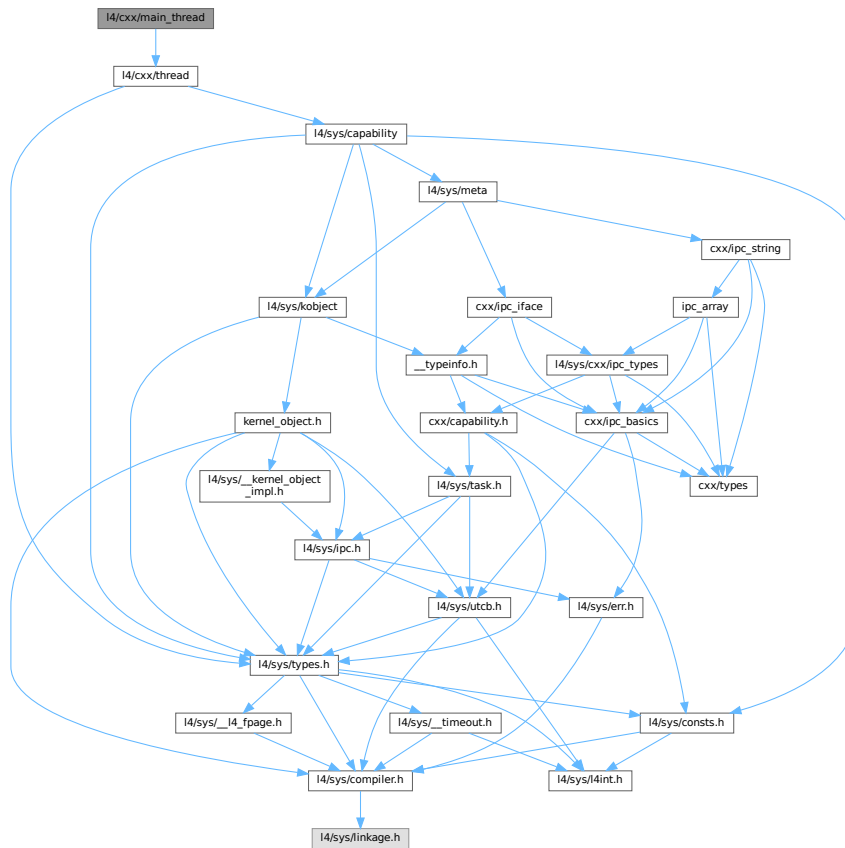
```

## 17.201 I4/cxx/main\_thread File Reference

Main thread.

```
#include <l4/cxx/thread>
```

Include dependency graph for main\_thread:



### Namespaces

- namespace `cxx`  
*Our C++ library.*

### 17.201.1 Detailed Description

Main thread.

Definition in file [main\\_thread](#).



## 17.202 main\_thread

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2004-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  *
00014  * As a special exception, you may use this file as part of a free software
00015  * library without restriction. Specifically, if other files instantiate
00016  * templates or use macros or inline functions from this file, or you compile
00017  * this file and link it with other files to produce an executable, this
00018  * file does not by itself cause the resulting executable to be covered by
00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023
00024 #ifndef L4_CXX_MAIN_THREAD_H__
00025 #define L4_CXX_MAIN_THREAD_H__
00026
00027 #include <l4/cxx/thread>
00028
00029 namespace cxx {
00030     class MainThread : public Thread
00031     {
00032     public:
00033         MainThread() : Thread(true)
00034         {}
00035     };
00036 };
00037
00038 #endif /* L4_CXX_MAIN_THREAD_H__ */

```

## 17.203 minmax

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "type_traits"
00011
00016 namespace cxx
00017 {
00018     // trivial, used to terminate the variadic recursion
00019     template<typename A>
00020     constexpr A const &
00021     min(A const &a)
00022     { return a; }
00023
00024     template<typename A, typename ...ARGS>
00025     constexpr A const &
00026     min(A const &a1, A const &a2, ARGS const &...a)
00027     {
00028         return min((a1 <= a2) ? a1 : a2, a...);
00029     }
00030
00031     template<typename A, typename ...ARGS>
00032     constexpr A const &
00033     min(cxx::identity_t<A> const &a1,
00034         cxx::identity_t<A> const &a2,
00035         ARGS const &...a)
00036     {
00037         return min<A>((a1 <= a2) ? a1 : a2, a...);
00038     }
00039
00040     // trivial, used to terminate the variadic recursion
00041     template<typename A>
00042     constexpr A const &
00043     max(A const &a)

```

```

00064 { return a; }
00065
00076 template<typename A, typename ...ARGS>
00077 constexpr A const &
00078 max(A const &a1, A const &a2, ARGS const &...a)
00079 { return max((a1 >= a2) ? a1 : a2, a...); }
00080
00091 template<typename A, typename ...ARGS>
00092 constexpr A const &
00093 max(cxx::identity_t<A> const &a1,
00094     cxx::identity_t<A> const &a2,
00095     ARGS const &...a)
00096 {
00097     return max<A>((a1 >= a2) ? a1 : a2, a...);
00098 }
00099
00107 template< typename T1 >
00108 inline
00109 T1 clamp(T1 v, T1 lo, T1 hi)
00110 { return min(hi, max(lo, v)); }
00111 };

```

## 17.204 numeric

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2025 Kernkonzept GmbH.
00004  * Author(s): Martin Decky <martin.decky@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 #include <l4/cxx/type_traits>
00012
00013 namespace cxx {
00014
00033 template <typename T>
00034 constexpr T gcd(T a, T b)
00035 {
00036     static_assert(Type_traits<T>::is_unsigned, "Type must be unsigned");
00037
00038     while (b != 0)
00039     {
00040         T remainder = a % b;
00041         a = b;
00042         b = remainder;
00043     }
00044
00045     return a;
00046 }
00047
00067 template <typename T>
00068 constexpr T lcm(T a, T b)
00069 {
00070     static_assert(Type_traits<T>::is_unsigned, "Type must be unsigned");
00071
00072     if (a == 0 || b == 0)
00073         return 0;
00074
00075     return (a / gcd(a, b)) * b;
00076 }
00077
00078 }

```

## 17.205 observer

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <l4/cxx/hlist>

```

```

00011
00012 namespace cxx {
00013
00014 class Observer : public H_list_item
00015 {
00016 public:
00017     virtual void notify() = 0;
00018 };
00019
00020 class Notifier : public H_list<Observer>
00021 {
00022 public:
00023     void notify()
00024     {
00025         for (Iterator i = begin(); i != end(); ++i)
00026             i->notify();
00027     }
00028 };
00029
00030 }
00031
00032

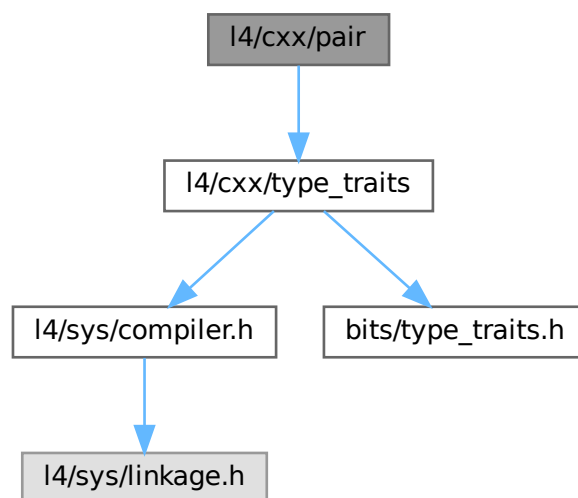
```

## 17.206 l4/cxx/pair File Reference

Pair implementation.

```
#include <l4/cxx/type_traits>
```

Include dependency graph for pair:





```

00046 : first(cxx::forward<A1>(first)), second(cxx::forward<A2>(second)) {}
00047
00052 template<typename A1>
00053 Pair(A1 &&first)
00054 : first(cxx::forward<A1>(first)), second() {}
00055
00057 Pair() = default;
00058 };
00059
00060 template< typename F, typename S >
00061 Pair<F,S> pair(F const &f, S const &s)
00062 { return cxx::Pair<F,S>(f,s); }
00063
00064
00073 template< typename Cmp, typename Typ >
00074 class Pair_first_compare
00075 {
00076 private:
00077     Cmp const &_cmp;
00078
00079 public:
00084     Pair_first_compare(Cmp const &cmp = Cmp()) : _cmp(cmp) {}
00085
00091     bool operator () (Typ const &l, Typ const &r) const
00092     { return _cmp(l.first,r.first); }
00093 };
00094
00095 }
00096
00097 template< typename OS, typename A, typename B >
00098 inline
00099 OS &operator << (OS &os, cxx::Pair<A,B> const &p)
00100 {
00101     os << p.first << ' ' << p.second;
00102     return os;
00103 }
00104

```

## 17.208 ref\_ptr

```

00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 #include "type_traits"
00012 #include <stddef.h>
00013 #include <l4/sys/compiler.h>
00014
00015 namespace cxx {
00016
00017 template< typename T >
00018 struct Default_ref_counter
00019 {
00020     void h_drop_ref(T *p) noexcept
00021     {
00022         if (p->remove_ref() == 0)
00023             delete p;
00024     }
00025
00026     void h_take_ref(T *p) noexcept
00027     {
00028         p->add_ref();
00029     }
00030 };
00031
00032 struct Ref_ptr_base
00033 {
00034     enum Default_value
00035     { Nil = 0 };
00036 };
00037
00038 template<typename T, template< typename X > class CNT = Default_ref_counter>
00039 class Weak_ptr;
00040
00041
00042 template <
00043     typename T = void,
00044     template< typename X > class CNT = Default_ref_counter

```

```

00069 >
00070 class Ref_ptr : public Ref_ptr_base, private CNT<T>
00071 {
00072 private:
00073     typedef decltype(nullptr) Null_type;
00074     typedef Weak_ptr<T, CNT> Wp;
00075 public:
00076     Ref_ptr() noexcept : _p(0) {}
00077     Ref_ptr(Ref_ptr_base::Default_value v)
00078     : _p(reinterpret_cast<T*>(static_cast<unsigned long>(v))) {}
00079     Ref_ptr(Wp const &o) noexcept : _p(o.ptr())
00080     { __take_ref(); }
00081     Ref_ptr(decltype(nullptr) n) noexcept : _p(n) {}
00082     template<typename X>
00083     explicit Ref_ptr(X *o) noexcept : _p(o)
00084     { __take_ref(); }
00085     Ref_ptr(T *o, [[maybe_unused]] bool d) noexcept : _p(o) { }
00086     T *get() const noexcept
00087     {
00088         return _p;
00089     }
00090     T *ptr() const noexcept
00091     {
00092         return _p;
00093     }
00094     T *release() noexcept
00095     {
00096         T *p = _p;
00097         _p = 0;
00098         return p;
00099     }
00100     ~Ref_ptr() noexcept
00101     { __drop_ref(); }
00102     template<typename OT>
00103     Ref_ptr(Ref_ptr<OT, CNT> const &o) noexcept
00104     {
00105         _p = o.ptr();
00106         __take_ref();
00107     }
00108     Ref_ptr(Ref_ptr<T> const &o) noexcept
00109     {
00110         _p = o._p;
00111         __take_ref();
00112     }
00113     template< typename OT >
00114     void operator = (Ref_ptr<OT> const &o) noexcept
00115     {
00116         __drop_ref();
00117         _p = o.ptr();
00118         __take_ref();
00119     }
00120     void operator = (Ref_ptr<T> const &o) noexcept
00121     {
00122         if (&o == this)
00123             return;
00124         __drop_ref();
00125         _p = o._p;
00126         __take_ref();
00127     }
00128     void operator = (Null_type) noexcept
00129     {
00130         __drop_ref();
00131         _p = 0;
00132     }
00133     template<typename OT>
00134     Ref_ptr(Ref_ptr<OT, CNT> &&o) noexcept
00135     { _p = o.release(); }
00136     Ref_ptr(Ref_ptr<T> &&o) noexcept
00137     { _p = o.release(); }

```

```

00191
00192     template< typename OT >
00193     void operator = (Ref_ptr<OT> &o) noexcept
00194     {
00195         __drop_ref();
00196         _p = o.release();
00197     }
00198
00199     void operator = (Ref_ptr<T> &o) noexcept
00200     {
00201         if (&o == this)
00202             return;
00203         __drop_ref();
00204         _p = o.release();
00205     }
00206 }
00207
00208 [[nodiscard]] explicit operator bool () const noexcept { return _p; }
00209
00210 T *operator -> () const noexcept
00211 { return _p; }
00212
00213 [[nodiscard]] bool operator == (Ref_ptr const &o) const noexcept
00214 { return _p == o._p; }
00215
00216 [[nodiscard]] bool operator != (Ref_ptr const &o) const noexcept
00217 { return _p != o._p; }
00218
00219 [[nodiscard]] bool operator < (Ref_ptr const &o) const noexcept
00220 { return _p < o._p; }
00221
00222 [[nodiscard]] bool operator <= (Ref_ptr const &o) const noexcept
00223 { return _p <= o._p; }
00224
00225 [[nodiscard]] bool operator > (Ref_ptr const &o) const noexcept
00226 { return _p > o._p; }
00227
00228 [[nodiscard]] bool operator >= (Ref_ptr const &o) const noexcept
00229 { return _p >= o._p; }
00230
00231 [[nodiscard]] bool operator == (T const *o) const noexcept
00232 { return _p == o; }
00233
00234 [[nodiscard]] bool operator < (T const *o) const noexcept
00235 { return _p < o; }
00236
00237 [[nodiscard]] bool operator <= (T const *o) const noexcept
00238 { return _p <= o; }
00239
00240 [[nodiscard]] bool operator > (T const *o) const noexcept
00241 { return _p > o; }
00242
00243 [[nodiscard]] bool operator >= (T const *o) const noexcept
00244 { return _p >= o; }
00245
00246 private:
00247     void __drop_ref() noexcept
00248     {
00249         if (_p)
00250             static_cast<CNT<T*>>(this)->h_drop_ref(_p);
00251     }
00252
00253     void __take_ref() noexcept
00254     {
00255         if (_p)
00256             static_cast<CNT<T*>>(this)->h_take_ref(_p);
00257     }
00258
00259     T *_p;
00260 };
00261
00262
00263 template<typename T, template< typename X > class CNT>
00264 class Weak_ptr
00265 {
00266 private:
00267     struct Null_type;
00268     typedef Ref_ptr<T, CNT> Rp;
00269
00270 public:
00271     Weak_ptr() = default;
00272     Weak_ptr(decltype(nullptr)) : _p(nullptr) {}
00273     // backwards 0 ctor
00274     explicit Weak_ptr(int x) noexcept
00275     L4_DEPRECATED("Use initialization from 'nullptr'")
00276     : _p(nullptr)
00277     { if (x != 0) __builtin_trap(); }

```

```

00278
00279 Weak_ptr(Rp const &o) noexcept : _p(o.ptr()) {}
00280 explicit Weak_ptr(T *o) noexcept : _p(o) {}
00281
00282 template<typename OT>
00283 Weak_ptr(Weak_ptr<OT, CNT> const &o) noexcept : _p(o.ptr()) {}
00284
00285 Weak_ptr(Weak_ptr<T, CNT> const &o) noexcept : _p(o._p) {}
00286
00287 Weak_ptr<T, CNT> &operator = (const Weak_ptr<T, CNT> &o) = default;
00288
00289 T *get() const noexcept { return _p; }
00290 T *ptr() const noexcept { return _p; }
00291
00292 T *operator -> () const noexcept { return _p; }
00293 operator Null_type const * () const noexcept
00294 { return reinterpret_cast<Null_type const*>(_p); }
00295
00296 private:
00297     T *_p;
00298 };
00299
00300 template<typename OT, typename T> inline
00301 Ref_ptr<OT> ref_ptr_static_cast(Ref_ptr<T> const &o)
00302 { return ref_ptr(static_cast<OT*>(o.ptr())); }
00303
00304 template< typename T >
00305 inline Ref_ptr<T> ref_ptr(T *t)
00306 { return Ref_ptr<T>(t); }
00307
00308 template< typename T >
00309 inline Weak_ptr<T> weak_ptr(T *t)
00310 { return Weak_ptr<T>(t); }
00311
00312
00313 class Ref_obj
00314 {
00315 private:
00316     mutable int _ref_cnt;
00317
00318 public:
00319     Ref_obj() : _ref_cnt(0) {}
00320     void add_ref() const noexcept { ++_ref_cnt; }
00321     int remove_ref() const noexcept { return --_ref_cnt; }
00322 };
00323
00324 template< typename T, typename... Args >
00325 Ref_ptr<T>
00326 make_ref_obj(Args &&... args)
00327 { return cxx::Ref_ptr<T>(new T(cxx::forward<Args>(args)...)); }
00328
00329 template<typename T, typename U>
00330 Ref_ptr<T>
00331 dynamic_pointer_cast(Ref_ptr<U> const &p) noexcept
00332 {
00333     // our constructor from a naked pointer increments the counter
00334     return Ref_ptr<T>(dynamic_cast<T *>(p.get()));
00335 }
00336
00337 template<typename T, typename U>
00338 Ref_ptr<T>
00339 static_pointer_cast(Ref_ptr<U> const &p) noexcept
00340 {
00341     // our constructor from a naked pointer increments the counter
00342     return Ref_ptr<T>(static_cast<T *>(p.get()));
00343 }
00344
00345 }

```

## 17.209 l4/cxx/ref\_ptr\_list File Reference

Implementation of a list of ref-ptr-managed objects.

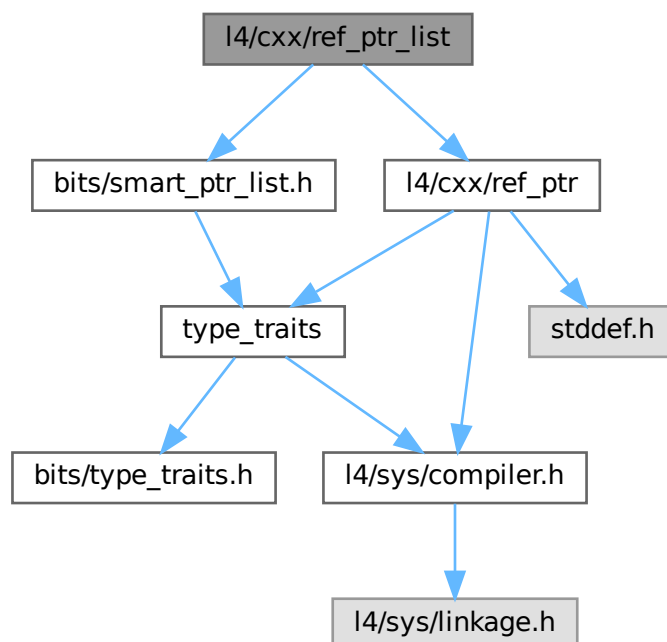
```

#include <l4/cxx/ref_ptr>
#include "bits/smart_ptr_list.h"

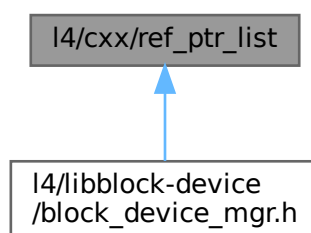
```



Include dependency graph for `ref_ptr_list`:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `cxx::Ref_obj_list_item< T >`  
Item for list linked via `cxx::Ref_ptr` with default reference counting.

## Namespaces

- namespace `cxx`  
Our C++ library.

## Typedefs

- `template<typename T>`  
`using cxx::Ref_ptr_list_item = Bits::Smart_ptr_list_item<T, cxx::Ref_ptr<T> >`  
*Item for list linked with **cxx::Ref\_ptr**.*
- `template<typename T>`  
`using cxx::Ref_ptr_list = Bits::Smart_ptr_list<Ref_ptr_list_item<T> >`  
*Single-linked list where elements are connected via a **cxx::Ref\_ptr**.*

### 17.209.1 Detailed Description

Implementation of a list of ref-ptr-managed objects.

Definition in file [ref\\_ptr\\_list](#).

## 17.210 ref\_ptr\_list

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * Copyright (C) 2018, 2022, 2024 Kernkonzept GmbH.
00008  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 #include <l4/cxx/ref_ptr>
00015
00016 #include "bits/smart_ptr_list.h"
00017
00018 namespace cxx {
00019
00021 template <typename T>
00022 using Ref_ptr_list_item = Bits::Smart_ptr_list_item<T, cxx::Ref_ptr<T> >;
00023
00025 template <typename T>
00026 struct Ref_obj_list_item : public Ref_ptr_list_item<T>, public cxx::Ref_obj {};
00027
00031 template <typename T>
00032 using Ref_ptr_list = Bits::Smart_ptr_list<Ref_ptr_list_item<T> >;
00033
00034 }
```

## 17.211 slab\_alloc

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 #include <l4/cxx/std_alloc>
00013 #include <l4/cxx/hlist>
00014 #include <l4/sys/consts.h>
00015
00016 namespace cxx {
00017
00029 template< int Obj_size, int Slab_size = L4_PAGESIZE,
00030          int Max_free = 2, template<typename A> class Alloc = New_allocator >
00031 class Base_slab
```

```

00032 {
00033 private:
00034     struct Free_o
00035     {
00036         Free_o *next;
00037     };
00038
00039 protected:
00040     struct Slab_i;
00041
00042 private:
00043     struct Slab_head : public H_list_item
00044     {
00045         unsigned num_free;
00046         Free_o *free;
00047         Base_slab<Obj_size, Slab_size, Max_free, Alloc> *cache;
00048
00049         inline Slab_head() noexcept : num_free(0), free(0), cache(0)
00050         {}
00051     };
00052
00053 // In an empty or partially filled slab, each free object stores a pointer to
00054 // the next free object. Thus, the size of an object needs to be at least the
00055 // size of a pointer.
00056 static_assert(Obj_size >= sizeof(void *),
00057               "Object size must be at least the size of a pointer.");
00058 static_assert(Obj_size <= Slab_size - sizeof(Slab_head),
00059               "Object_size exceeds slab capability.");
00060
00061 public:
00062     enum
00063     {
00064         object_size      = Obj_size,
00065         slab_size        = Slab_size,
00066         objects_per_slab = (Slab_size - sizeof(Slab_head)) / object_size,
00067         max_free_slabs   = Max_free,
00068     };
00069
00070 protected:
00071     struct Slab_store
00072     {
00073         char _o[slab_size - sizeof(Slab_head)];
00074         Free_o *object(unsigned obj) noexcept
00075         { return reinterpret_cast<Free_o*>(_o + object_size * obj); }
00076     };
00077
00078     struct Slab_i : public Slab_store, public Slab_head
00079     {};
00080
00081 public:
00082     typedef Alloc<Slab_i> Slab_alloc;
00083
00084     typedef void Obj_type;
00085
00086 private:
00087     Slab_alloc _alloc;
00088     unsigned _num_free;
00089     unsigned _num_slabs;
00090     H_list<Slab_i> _full_slabs;
00091     H_list<Slab_i> _partial_slabs;
00092     H_list<Slab_i> _empty_slabs;
00093
00094     void add_slab(Slab_i *s) noexcept
00095     {
00096         s->num_free = objects_per_slab;
00097         s->cache = this;
00098
00099         //L4::cerr << "Slab: " << this << "->add_slab(" << s << ", size="
00100         // << slab_size << "):" << " f=" << s->object(0) << '\n';
00101
00102         // initialize free list
00103         Free_o *f = s->free = s->object(0);
00104         for (unsigned i = 1; i < objects_per_slab; ++i)
00105         {
00106             f->next = s->object(i);
00107             f = f->next;
00108         }
00109         f->next = 0;
00110
00111         // insert slab into cache's list
00112         _empty_slabs.push_front(s);
00113         ++_num_slabs;
00114         ++_num_free;
00115     }
00116
00117     bool grow() noexcept
00118     {

```

```

00136     Slab_i *s = _alloc.alloc();
00137     if (!s)
00138         return false;
00139
00140     new (s, cxx::Nothrow()) Slab_i();
00141
00142     add_slab(s);
00143     return true;
00144 }
00145
00155 void shrink() noexcept
00156 {
00157     if (!_alloc.can_free)
00158         return;
00159
00160     while (!_empty_slabs.empty() && _num_free > max_free_slabs)
00161     {
00162         Slab_i *s = _empty_slabs.front();
00163         _empty_slabs.remove(s);
00164         --_num_free;
00165         --_num_slabs;
00166         _alloc.free(s);
00167     }
00168 }
00169
00170 public:
00171 Base_slab(Slab_alloc const &alloc = Slab_alloc()) noexcept
00172     : _alloc(alloc), _num_free(0), _num_slabs(0), _full_slabs(),
00173       _partial_slabs(), _empty_slabs()
00174 {}
00175
00176 ~Base_slab() noexcept
00177 {
00178     while (!_empty_slabs.empty())
00179     {
00180         Slab_i *o = _empty_slabs.front();
00181         _empty_slabs.remove(o);
00182         _alloc.free(o);
00183     }
00184     while (!_partial_slabs.empty())
00185     {
00186         Slab_i *o = _partial_slabs.front();
00187         _partial_slabs.remove(o);
00188         _alloc.free(o);
00189     }
00190     while (!_full_slabs.empty())
00191     {
00192         Slab_i *o = _full_slabs.front();
00193         _full_slabs.remove(o);
00194         _alloc.free(o);
00195     }
00196 }
00197
00207 void *alloc() noexcept
00208 {
00209     H_list<Slab_i> *free = &_partial_slabs;
00210     if (free->empty())
00211         free = &_empty_slabs;
00212
00213     if (free->empty() && !grow())
00214         return 0;
00215
00216     Slab_i *s = free->front();
00217     Free_o *o = s->free;
00218     s->free = o->next;
00219
00220     if (free == &_empty_slabs)
00221     {
00222         _empty_slabs.remove(s);
00223         --_num_free;
00224     }
00225
00226     --(s->num_free);
00227
00228     if (!s->free)
00229     {
00230         _partial_slabs.remove(s);
00231         _full_slabs.push_front(s);
00232     }
00233     else if (free == &_empty_slabs)
00234         _partial_slabs.push_front(s);
00235
00236     //L4::cerr << this << "->alloc(): " << o << ", of " << s << '\n';
00237
00238     return o;
00239 }
00240

```

```

00246 void free(void *_o) noexcept
00247 {
00248     if (!_o)
00249         return;
00250
00251     unsigned long addr = reinterpret_cast<unsigned long>(_o);
00252
00253     // find out the slab the object is in
00254     addr = (addr / slab_size) * slab_size;
00255     Slab_i *s = reinterpret_cast<Slab_i*>(addr);
00256
00257     if (s->cache != this)
00258         return;
00259
00260     Free_o *o = reinterpret_cast<Free_o*>(_o);
00261
00262     o->next = s->free;
00263     s->free = o;
00264
00265     bool was_full = false;
00266
00267     if (!s->num_free)
00268     {
00269         _full_slabs.remove(s);
00270         was_full = true;
00271     }
00272
00273     ++(s->num_free);
00274
00275     if (s->num_free == objects_per_slab)
00276     {
00277         if (!was_full)
00278             _partial_slabs.remove(s);
00279
00280         _empty_slabs.push_front(s);
00281         ++_num_free;
00282         if (_num_free > max_free_slabs)
00283             shrink();
00284
00285         was_full = false;
00286     }
00287     else if (was_full)
00288         _partial_slabs.push_front(s);
00289
00290     //L4::cerr << this << "->free(" << _o << "): of " << s << '\n';
00291 }
00292
00293 unsigned total_objects() const noexcept
00294 { return _num_slabs * objects_per_slab; }
00295
00296 unsigned free_objects() const noexcept
00297 {
00298     unsigned count = 0;
00299
00300     /* count partial slabs first */
00301     for (typename H_list<Slab_i>::Const_iterator s = _partial_slabs.begin();
00302          s != _partial_slabs.end(); ++s)
00303         count += s->num_free;
00304
00305     /* add empty slabs */
00306     count += _num_free * objects_per_slab;
00307
00308     return count;
00309 }
00310
00311 template<typename Type, int Slab_size = L4_PAGESIZE,
00312         int Max_free = 2, template<typename A> class Alloc = New_allocator >
00313 class Slab : public Base_slab<sizeof(Type), Slab_size, Max_free, Alloc>
00314 {
00315 private:
00316     typedef Base_slab<sizeof(Type), Slab_size, Max_free, Alloc> Base_type;
00317 public:
00318     typedef Type Obj_type;
00319
00320     Slab(typename Base_type::Slab_alloc const &alloc
00321          = typename Base_type::Slab_alloc()) noexcept
00322         : Base_slab<sizeof(Type), Slab_size, Max_free, Alloc>(alloc) {}
00323
00324     Type *alloc() noexcept
00325     {
00326         return reinterpret_cast<Type *>(Base_type::alloc());
00327     }
00328
00329     void free(Type *o) noexcept

```

```

00367 { Base_slab<sizeof(Type), Slab_size, Max_free, Alloc>::free(o); }
00368 };
00369
00370
00386 template< int Obj_size, int Slab_size = L4_PAGESIZE,
00387 int Max_free = 2, template<typename A> class Alloc = New_allocator >
00388 class Base_slab_static
00389 {
00390 private:
00391     typedef Base_slab<Obj_size, Slab_size, Max_free, Alloc> _A;
00392     static _A _a;
00393 public:
00394     typedef void Obj_type;
00395     enum
00396     {
00398         object_size      = Obj_size,
00400         slab_size        = Slab_size,
00402         objects_per_slab = _A::objects_per_slab,
00404         max_free_slabs   = Max_free,
00405     };
00406
00412 void *alloc() noexcept { return _a.alloc(); }
00413
00420 void free(void *p) noexcept { _a.free(p); }
00421
00430 unsigned total_objects() const noexcept { return _a.total_objects(); }
00431
00440 unsigned free_objects() const noexcept { return _a.free_objects(); }
00441 };
00442
00443
00444 template< int _O, int _S, int _M, template<typename A> class Alloc >
00445 typename Base_slab_static<_O,_S,_M,Alloc>::_A
00446 Base_slab_static<_O,_S,_M,Alloc>::_a;
00447
00463 template<typename Type, int Slab_size = L4_PAGESIZE,
00464 int Max_free = 2, template<typename A> class Alloc = New_allocator >
00465 class Slab_static
00466 : public Base_slab_static<sizeof(Type), Slab_size, Max_free, Alloc>
00467 {
00468 public:
00469
00470     typedef Type Obj_type;
00471     Type *alloc() noexcept
00472     {
00480         return reinterpret_cast<Type *>(
00481             Base_slab_static<sizeof(Type), Slab_size, Max_free, Alloc>::alloc());
00482     }
00483 };
00484
00485 }

```

## 17.212 slist

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2011 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 #include "bits/list_basics.h"
00012
00013 namespace cxx {
00014
00015 class S_list_item
00016 {
00017 public:
00018     S_list_item() : _n(0) {}
00019     // BSS allocation
00020     explicit S_list_item(bool) {}
00021
00022 private:
00023     template<typename T, typename P> friend class S_list;
00024     template<typename T, typename P> friend class S_list_tail;
00025     template<typename T, typename X> friend struct Bits::Basic_list_policy;
00026
00027     S_list_item(S_list_item const &);
00028     void operator = (S_list_item const &);
00029 }

```

```

00030     S_list_item *_n;
00031 };
00032
00033 template< typename T, typename POLICY = Bits::Basic_list_policy< T, S_list_item > >
00040 class S_list : public Bits::Basic_list<POLICY>
00041 {
00042     S_list(S_list const &) = delete;
00043     void operator = (S_list const &) = delete;
00044
00045 private:
00046     typedef typename Bits::Basic_list<POLICY> Base;
00047
00048 public:
00049     typedef typename Base::Iterator Iterator;
00050
00051     S_list(S_list &&o) : Base(static_cast<Base&&>(o)) {}
00052
00053     S_list &operator = (S_list &&o)
00054     {
00055         Base::operator = (static_cast<Base&&>(o));
00056         return *this;
00057     }
00058
00059     // BSS allocation
00060     explicit S_list(bool x) : Base(x) {}
00061
00062     S_list() : Base() {}
00063
00065     void add(T *e)
00066     {
00067         e->_n = this->_f;
00068         this->_f = e;
00069     }
00070
00071     template< typename CAS >
00072     void add(T *e, CAS const &c)
00073     {
00074         do
00075         {
00076             e->_n = this->_f;
00077         }
00078         while (!c(&this->_f, e->_n, e));
00079     }
00080
00082     void push_front(T *e) { add(e); }
00083
00089     T *pop_front()
00090     {
00091         T *r = this->front();
00092         if (this->_f)
00093             this->_f = this->_f->_n;
00094         return r;
00095     }
00096
00097     void insert(T *e, Iterator const &pred)
00098     {
00099         S_list_item *p = *pred;
00100         e->_n = p->_n;
00101         p->_n = e;
00102     }
00103
00104     static void insert_before(T *e, Iterator const &succ)
00105     {
00106         S_list_item **x = Base::__get_internal(succ);
00107
00108         e->_n = *x;
00109         *x = e;
00110     }
00111
00112     static void replace(Iterator const &p, T*e)
00113     {
00114         S_list_item **x = Base::__get_internal(p);
00115         e->_n = (*x)->_n;
00116         *x = e;
00117     }
00118
00119     static Iterator erase(Iterator const &e)
00120     {
00121         S_list_item **x = Base::__get_internal(e);
00122         *x = (*x)->_n;
00123         return e;
00124     }
00125
00126 };
00127
00128
00129 template< typename T >

```

```

00130 class S_list_bss : public S_list<T>
00131 {
00132 public:
00133     S_list_bss() : S_list<T>(true) {}
00134 };
00135
00136 template< typename T, typename POLICY = Bits::Basic_list_policy< T, S_list_item > >
00137 class S_list_tail : public S_list<T, POLICY>
00138 {
00139 private:
00140     typedef S_list<T, POLICY> Base;
00141     void add(T *e) = delete;
00142
00143 public:
00144     using Iterator = typename Base::Iterator;
00145     S_list_tail() : Base(), _tail(&this->_f) {}
00146
00147     S_list_tail(S_list_tail &t)
00148     : Base(static_cast<Base&&>(t)), _tail(t.empty() ? &this->_f : t._tail)
00149     {
00150         t._tail = &t._f;
00151     }
00152
00153     S_list_tail &operator = (S_list_tail &t)
00154     {
00155         if (&t == this)
00156             return *this;
00157
00158         Base::operator = (static_cast<Base &&>(t));
00159         _tail = t.empty() ? &this->_f : t._tail;
00160         t._tail = &t._f;
00161         return *this;
00162     }
00163
00164     void push_front(T *e)
00165     {
00166         if (Base::empty())
00167             _tail = &e->_n;
00168
00169         Base::push_front(e);
00170     }
00171
00172     void push_back(T *e)
00173     {
00174         e->_n = 0;
00175         *_tail = e;
00176         _tail = &e->_n;
00177     }
00178
00179     void clear()
00180     {
00181         Base::clear();
00182         _tail = &this->_f;
00183     }
00184
00185     void append(S_list_tail &o)
00186     {
00187         T *x = o.front();
00188         *_tail = x;
00189         if (x)
00190             _tail = o._tail;
00191         o.clear();
00192     }
00193
00194     T *pop_front()
00195     {
00196         T *t = Base::pop_front();
00197         if (t && Base::empty())
00198             _tail = &this->_f;
00199         return t;
00200     }
00201
00202 private:
00203     static void insert(T *e, Iterator const &pred);
00204     static void insert_before(T *e, Iterator const &succ);
00205     static void replace(Iterator const &p, T *e);
00206     static Iterator erase(Iterator const &e);
00207
00208 private:
00209     S_list_item **_tail;
00210 };
00211
00212 }

```



## 17.213 static\_container

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2012-2013 Technische Universität Dresden.
00004  * Copyright (C) 2016-2017, 2020, 2023-2024 Kernkonzept GmbH.
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 #include <l4/cxx/type_traits>
00012 #include <stddef.h>
00013
00014 namespace cxx {
00015
00016 template< typename T >
00017 class Static_container
00018 {
00019 private:
00020     struct X : T
00021     {
00022         void *operator new (size_t, void *p) noexcept { return p; }
00023         void operator delete (void *) {}
00024         X() = default;
00025         template<typename ...Args>
00026         X(Args && ...a) : T(cxx::forward<Args>(a)...) {}
00027     };
00028
00029 public:
00030     void operator = (Static_container const &) = delete;
00031     Static_container(Static_container const &) = delete;
00032     Static_container() = default;
00033
00034     T *get() { return reinterpret_cast<X*>(_s); }
00035     T *operator -> () { return get(); }
00036     T &operator * () { return *get(); }
00037     operator T* () { return get(); }
00038
00039     void construct()
00040     { new (reinterpret_cast<void*>(_s)) X; }
00041
00042     template< typename ...Args >
00043     void construct(Args && ...args)
00044     { new (reinterpret_cast<void*>(_s)) X(cxx::forward<Args>(args)...) };
00045
00046 private:
00047     char _s[sizeof(X)] __attribute__((aligned(__alignof(X)))));
00048 };
00049
00050 }
00051
00052

```

## 17.214 static\_vector

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002
00003 #pragma once
00004
00005 #include "type_traits"
00006
00007 namespace cxx {
00008
00015 template<typename T, typename IDX = unsigned>
00016 class static_vector
00017 {
00018 private:
00019     template<typename X, typename IDX2> friend class static_vector;
00020     T *_v;
00021     IDX _l;
00022
00023 public:
00024     typedef T value_type;
00025     typedef IDX index_type;
00026
00027     static_vector() = default;
00028     static_vector(value_type *v, index_type length) : _v(v), _l(length) {}
00029
00030     template<typename Z,
00031              typename = enable_if_t<is_same<remove_extent_t<Z>, T>::value>

```

```

00032 constexpr static_vector(Z &v) : _v(v), _l(array_size(v))
00033 {}
00034
00036 template<typename X,
00037         typename = enable_if_t<is_convertible<X, T>::value>
00038 static_vector(static_vector<X, IDX> const &o) : _v(o._v), _l(o._l) {}
00039
00040 index_type size() const { return _l; }
00041 bool empty() const { return _l == 0; }
00042
00043 value_type &operator [] (index_type idx) { return _v[idx]; }
00044 value_type const &operator [] (index_type idx) const { return _v[idx]; }
00045
00046 value_type *begin() { return _v; }
00047 value_type *end() { return _v + _l; }
00048 value_type const *begin() const { return _v; }
00049 value_type const *end() const { return _v + _l; }
00050 value_type const *cbegin() const { return _v; }
00051 value_type const *cend() const { return _v + _l; }
00052
00054 index_type index(value_type const *o) const { return o - _v; }
00055 index_type index(value_type const &o) const { return &o - _v; }
00056 };
00057
00058 }

```

## 17.215 std\_alloc

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 #include <stddef.h>
00013 namespace cxx {
00019 class Nothrow {};
00020 }
00021
00028 inline void *operator new (size_t, void *mem, cxx::Nothrow const &) noexcept
00029 { return mem; }
00030
00035 void *operator new (size_t, cxx::Nothrow const &) noexcept;
00036
00042 void operator delete (void *, cxx::Nothrow const &) noexcept;
00043
00044 namespace cxx {
00045
00046 template< typename _Type >
00056 class New_allocator
00057 {
00058 public:
00059     enum { can_free = true };
00060
00061     New_allocator() noexcept {}
00062     New_allocator(New_allocator const &) noexcept {}
00063
00064     ~New_allocator() noexcept {}
00065
00066     _Type *alloc() noexcept
00067     { return static_cast<_Type*> (::operator new(sizeof (_Type), cxx::Nothrow())); }
00068
00069     void free(_Type *t) noexcept
00070     { ::operator delete(t, cxx::Nothrow()); }
00071 };
00072
00073 }
00074

```

## 17.216 std\_ops

```

00001 // vim:set ft=cpp: -*- Mode: C++ -*-

```

```

00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 namespace cxx {
00013
00017 template< typename Obj >
00018 struct Lt_functor
00019 {
00020     bool operator () (Obj const &l, Obj const &r) const
00021     { return l < r; }
00022 };
00023
00024 };
00025

```

## 17.217 string

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00012 #pragma once
00013
00014 #include <l4/cxx/minmax>
00015 #include <l4/cxx/basic_ostream>
00016
00017 namespace cxx {
00018
00030 class String
00031 {
00032 public:
00033
00035     typedef char const *Index;
00036
00038     String(char const *s) noexcept : _start(s), _len(__builtin_strlen(s)) {}
00040     String(char const *s, unsigned long len) noexcept : _start(s), _len(len) {}
00041
00048     String(char const *s, char const *e) noexcept : _start(s), _len(e - s) {}
00049
00051     String() : _start(0), _len(0) {}
00052
00054     Index start() const { return _start; }
00056     Index end() const { return _start + _len; }
00058     int len() const { return _len; }
00059
00061     void start(char const *s) { _start = s; }
00063     void len(unsigned long len) { _len = len; }
00065     bool empty() const { return !_len; }
00066
00068     String head(Index end) const
00069     {
00070         if (end < _start)
00071             return String();
00072
00073         if (eof(end))
00074             return *this;
00075
00076         return String(_start, end - _start);
00077     }
00078
00080     String head(unsigned long end) const
00081     { return head(start() + end); }
00082
00084     String substr(unsigned long idx, unsigned long len = ~0UL) const
00085     {
00086         if (idx >= _len)
00087             return String(end(), 0UL);
00088
00089         return String(_start + idx, cxx::min(len, _len - idx));
00090     }
00091

```

```

00093 String substr(char const *start, unsigned long len = 0) const
00094 {
00095     if (start >= _start && !eof(start))
00096     {
00097         unsigned long nlen = _start + _len - start;
00098         if (len != 0)
00099             nlen = cxx::min(nlen, len);
00100         return String(start, nlen);
00101     }
00102
00103     return String(end(), 0UL);
00104 }
00105
00107 template< typename F >
00108 char const *find_match(F &&match) const
00109 {
00110     String::Index s = _start;
00111     while (1)
00112     {
00113         if (eof(s))
00114             return s;
00115
00116         if (match(*s))
00117             return s;
00118
00119         ++s;
00120     }
00121 }
00122
00124 char const *find(char const *c) const
00125 { return find(c, start()); }
00126
00128 char const *find(int c) const
00129 { return find(c, start()); }
00130
00132 char const *rfind(char const *c) const
00133 {
00134     if (!_len)
00135         return end();
00136
00137     char const *p = end();
00138     --p;
00139     while (p >= _start)
00140     {
00141         if (*p == *c)
00142             return p;
00143         --p;
00144     }
00145     return end();
00146 }
00147
00148
00155 Index starts_with(cxx::String const &c) const
00156 {
00157     unsigned long i;
00158     for (i = 0; i < c._len && i < _len; ++i)
00159         if (_start[i] != c[i])
00160             return 0;
00161     return i == c._len ? start() + i : 0;
00162 }
00163
00165 char const *find(int c, char const *s) const
00166 {
00167     if (s < _start)
00168         return end();
00169
00170     while (1)
00171     {
00172         if (eof(s))
00173             return s;
00174
00175         if (*s == c)
00176             return s;
00177
00178         ++s;
00179     }
00180 }
00181
00191 char const *find(char const *c, char const *s) const
00192 {
00193     if (s < _start)
00194         return end();
00195
00196     while (1)
00197     {
00198         if (eof(s))
00199             return s;

```

```

00200
00201     for (char const *x = c; *x; ++x)
00202         if (*s == *x)
00203             return s;
00204
00205     ++s;
00206 }
00207 }
00208
00210 char const &operator [] (unsigned long idx) const { return _start[idx]; }
00212 char const &operator [] (int idx) const { return _start[idx]; }
00214 char const &operator [] (Index idx) const { return *idx; }
00215
00217 bool eof(char const *s) const { return s >= _start + _len || !*s; }
00218
00227 template<typename INT>
00228 int from_dec(INT *v) const
00229 {
00230     *v = 0;
00231     Index c;
00232     for (c = start(); !eof(c); ++c)
00233     {
00234         unsigned char n;
00235         if (*c >= '0' && *c <= '9')
00236             n = *c - '0';
00237         else
00238             return c - start();
00239
00240         *v *= 10;
00241         *v += n;
00242     }
00243     return c - start();
00244 }
00245
00256 template<typename INT>
00257 int from_hex(INT *v) const
00258 {
00259     *v = 0;
00260     unsigned shift = 0;
00261     Index c;
00262     for (c = start(); !eof(c); ++c)
00263     {
00264         shift += 4;
00265         if (shift > sizeof(INT) * 8)
00266             return -1;
00267         unsigned char n;
00268         if (*c >= '0' && *c <= '9')
00269             n = *c - '0';
00270         else if (*c >= 'A' && *c <= 'F')
00271             n = *c - 'A' + 10;
00272         else if (*c >= 'a' && *c <= 'f')
00273             n = *c - 'a' + 10;
00274         else
00275             return c - start();
00276
00277         *v <<= 4;
00278         *v |= n;
00279     }
00280     return c - start();
00281 }
00282
00284 bool operator == (String const &o) const
00285 {
00286     if (len() != o.len())
00287         return false;
00288
00289     for (unsigned long i = 0; i < _len; ++i)
00290         if (_start[i] != o._start[i])
00291             return false;
00292
00293     return true;
00294 }
00295
00297 bool operator != (String const &o) const
00298 { return ! (operator == (o)); }
00299
00300 private:
00301     char const *_start;
00302     unsigned long _len;
00303 };
00304
00305 }
00306
00308 inline
00309 L4::BasicOStream &operator << (L4::BasicOStream &s, cxx::String const &str)
00310 {
00311     s.write(str.start(), str.len());

```

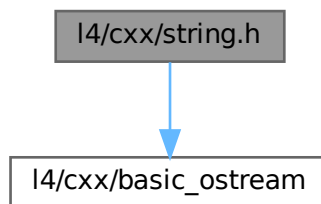
```
00312     return s;
00313 }
```

## 17.218 l4/cxx/string.h File Reference

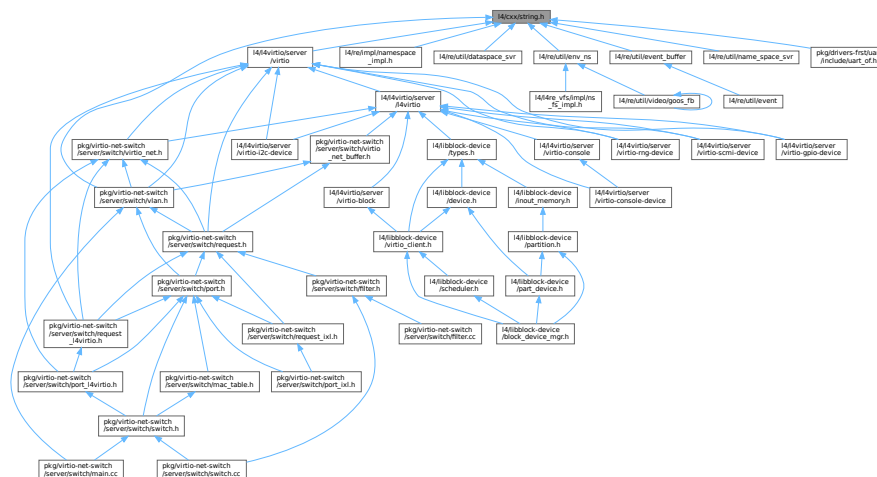
String.

```
#include <l4/cxx/basic_ostream>
```

Include dependency graph for string.h:



This graph shows which files directly or indirectly include this file:



### Data Structures

- class [L4::String](#)  
*A null-terminated string container class.*

### Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*

## 17.218.1 Detailed Description

String.

Definition in file [string.h](#).

## 17.219 string.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00007  *          Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00008  *          economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 #include <l4/cxx/basic_ostream>
00015
00016 namespace L4 {
00017
00022     class String
00023     {
00024     public:
00025         String(char const *str = "") : _str(str)
00026         {}
00027
00028         unsigned length() const
00029         {
00030             unsigned l;
00031             for (l = 0; _str[l]; l++)
00032                 ;
00033             return l;
00034         }
00035
00036         char const *p_str() const { return _str; }
00037
00038     private:
00039         char const *_str;
00040     };
00041
00042     inline
00044     L4::BasicOStream &operator « (L4::BasicOStream &o, L4::String const &s)
00045     {
00046         o « s.p_str();
00047         return o;
00048     }

```

## 17.220 type\_list

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 #pragma once
00003
00004 /*
00005  * (c) 2012 Alexander Warg <warg@os.inf.tu-dresden.de>,
00006  *          economic rights: Technische Universität Dresden (Germany)
00007  *
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010
00011 #include "type_traits"
00012
00014 namespace cxx {
00015
00016     template< typename ...T >
00017     struct type_list;
00018
00019     template<>
00020     struct type_list<>

```

```

00021 {
00022     typedef false_type head;
00023     typedef false_type tail;
00024 };
00025
00026 template<typename HEAD, typename ...TAIL>
00027 struct type_list<HEAD, TAIL...>
00028 {
00029     typedef HEAD head;
00030     typedef type_list<TAIL...> tail;
00031 };
00032
00033 template<typename TYPELIST, template <typename T> class PREDICATE>
00034 struct find_type;
00035
00036 template<template <typename T> class PREDICATE>
00037 struct find_type<type_list<>, PREDICATE>
00038 {
00039     typedef false_type type;
00040 };
00041
00042 template<typename TYPELIST, template <typename T> class PREDICATE>
00043 struct find_type
00044 {
00045     typedef typename conditional<PREDICATE<typename TYPELIST::head>::value,
00046                                 typename TYPELIST::head,
00047                                 typename find_type<typename TYPELIST::tail, PREDICATE>::type>::type
00048     type;
00049 };
00050
00051 template<typename TYPELIST, template <typename T> class PREDICATE>
00052 using find_type_t = typename find_type<TYPELIST, PREDICATE>::type;
00053 }
00054

```

## 17.221 type\_traits

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002
00003 /*
00004  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00005  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00006  *      economic rights: Technische Universität Dresden (Germany)
00007  *
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010
00011
00012 #pragma once
00013
00014 #pragma GCC system_header
00015
00016 #include <14/sys/compiler.h>
00017 #include "bits/type_traits.h"
00018
00019 namespace cxx {
00020
00021 template< typename T, T V >
00022 struct integral_constant
00023 {
00024     static T const value = V;
00025     typedef T value_type;
00026     typedef integral_constant<T, V> type;
00027 };
00028
00029 typedef integral_constant<bool, true> true_type;
00030 typedef integral_constant<bool, false> false_type;
00031
00032 template< typename T > struct remove_reference;
00033
00034 template< typename T > struct identity { typedef T type; };
00035 template< typename T > using identity_t = typename identity<T>::type;
00036
00037 template< typename T1, typename T2 > struct is_same;
00038
00039 template< typename T > struct remove_const;
00040
00041 template< typename T > struct remove_volatile;
00042
00043 template< typename T > struct remove_cv;
00044
00045 template< typename T > struct remove_pointer;

```



```

00046
00047 template< typename T > struct remove_extent;
00048
00049 template< typename T > struct remove_all_extents;
00050
00051
00052
00053 template< typename, typename >
00054 struct is_same : false_type {};
00055
00056 template< typename T >
00057 struct is_same<T, T> : true_type {};
00058
00059 template< typename T1, typename T2 >
00060 inline constexpr bool is_same_v = is_same<T1, T2>::value;
00061
00062 template< typename T >
00063 struct remove_reference { typedef T type; };
00064
00065 template< typename T >
00066 struct remove_reference<T &> { typedef T type; };
00067
00068 template< typename T >
00069 struct remove_reference<T &&> { typedef T type; };
00070
00071 template< typename T >
00072 using remove_reference_t = typename remove_reference<T>::type;
00073
00074 template< typename T > struct remove_const { typedef T type; };
00075 template< typename T > struct remove_const<T const> { typedef T type; };
00076 template< typename T > using remove_const_t = typename remove_const<T>::type;
00077
00078 template< typename T > struct remove_volatile { typedef T type; };
00079 template< typename T > struct remove_volatile<T volatile> { typedef T type; };
00080 template< typename T > using remove_volatile_t = typename remove_volatile<T>::type;
00081
00082 template< typename T >
00083 struct remove_cv { typedef remove_const_t<remove_volatile_t<T>> type; };
00084
00085 template< typename T >
00086 using remove_cv_t = typename remove_cv<T>::type;
00087
00088 template<class T>
00089 struct remove_cvref { using type = remove_cv_t<remove_reference_t<T>>; };
00090
00091 template< typename T >
00092 using remove_cvref_t = typename remove_cvref<T>::type;
00093
00094 template< typename T, typename >
00095 struct __remove_pointer_h { typedef T type; };
00096
00097 template< typename T, typename I >
00098 struct __remove_pointer_h<T, I*> { typedef I type; };
00099
00100 template< typename T >
00101 struct remove_pointer : __remove_pointer_h<T, remove_cv_t<T>> {};
00102
00103 template< typename T >
00104 using remove_pointer_t = typename remove_pointer<T>::type;
00105
00106
00107 template< typename T >
00108 struct remove_extent { typedef T type; };
00109
00110 template< typename T >
00111 struct remove_extent<T[]> { typedef T type; };
00112
00113 template< typename T, unsigned long N >
00114 struct remove_extent<T[N]> { typedef T type; };
00115
00116 template< typename T >
00117 using remove_extent_t = typename remove_extent<T>::type;
00118
00119
00120 template< typename T >
00121 struct remove_all_extents { typedef T type; };
00122
00123 template< typename T >
00124 struct remove_all_extents<T[]> { typedef typename remove_all_extents<T>::type type; };
00125
00126 template< typename T, unsigned long N >
00127 struct remove_all_extents<T[N]> { typedef typename remove_all_extents<T>::type type; };
00128
00129 template< typename T >
00130 using remove_all_extents_t = typename remove_all_extents<T>::type;
00131
00132 template< typename T >

```

```

00133 constexpr T &&
00134 forward(cxx::remove_reference_t<T> &t)
00135 { return static_cast<T &&>(t); }
00136
00137 template< typename T >
00138 constexpr T &&
00139 forward(cxx::remove_reference_t<T> &&t)
00140 { return static_cast<T &&>(t); }
00141
00142 template< typename T >
00143 constexpr cxx::remove_reference_t<T> &&
00144 move(T &&t) { return static_cast< cxx::remove_reference_t<T> &&>(t); }
00145
00146 template< bool, typename T = void >
00147 struct enable_if {};
00148
00149 template< typename T >
00150 struct enable_if<true, T> { typedef T type; };
00151
00152 template< bool C, typename T = void >
00153 using enable_if_t = typename enable_if<C, T>::type;
00154
00155 template< typename T >
00156 struct is_const : false_type {};
00157
00158 template< typename T >
00159 struct is_const<T const> : true_type {};
00160
00161 template< typename T >
00162 inline constexpr bool is_const_v = is_const<T>::value;
00163
00164 template< typename T >
00165 struct is_volatile : false_type {};
00166
00167 template< typename T >
00168 struct is_volatile<T volatile> : true_type {};
00169
00170 template< typename T >
00171 inline constexpr bool is_volatile_v = is_volatile<T>::value;
00172
00173 template< typename T >
00174 struct is_pointer : false_type {};
00175
00176 template< typename T >
00177 struct is_pointer<T *> : true_type {};
00178
00179 template< typename T >
00180 inline constexpr bool is_pointer_v = is_pointer<T>::value;
00181
00182 template<class T>
00183 inline constexpr bool is_null_pointer_v = is_same_v<decltype(nullptr), remove_cv_t<T>;
00184
00185 template< typename T >
00186 struct is_reference : false_type {};
00187
00188 template< typename T >
00189 struct is_reference<T &> : true_type {};
00190
00191 template< typename T >
00192 struct is_reference<T &&> : true_type {};
00193
00194 template< typename T >
00195 inline constexpr bool is_reference_v = is_reference<T>::value;
00196
00197 template< bool, typename, typename >
00198 struct conditional;
00199
00200 template< bool C, typename T_TRUE, typename T_FALSE >
00201 struct conditional { typedef T_TRUE type; };
00202
00203 template< typename T_TRUE, typename T_FALSE >
00204 struct conditional< false, T_TRUE, T_FALSE > { typedef T_FALSE type; };
00205
00206 template< bool C, typename T_TRUE, typename T_FALSE >
00207 using conditional_t = typename conditional<C, T_TRUE, T_FALSE>::type;
00208
00209 template<typename T>
00210 struct is_enum : integral_constant<bool, __is_enum(T)> {};
00211
00212 template< typename T >
00213 inline constexpr bool is_enum_v = is_enum<T>::value;
00214
00215 template<typename T>
00216 struct is_polymorphic : cxx::integral_constant<bool, __is_polymorphic(T)> {};
00217
00218 template< typename T > struct is_integral : false_type {};
00219

```

```

00220 template<> struct is_integral<bool> : true_type {};
00221
00222 template<> struct is_integral<char> : true_type {};
00223 template<> struct is_integral<signed char> : true_type {};
00224 template<> struct is_integral<unsigned char> : true_type {};
00225 template<> struct is_integral<short> : true_type {};
00226 template<> struct is_integral<unsigned short> : true_type {};
00227 template<> struct is_integral<int> : true_type {};
00228 template<> struct is_integral<unsigned int> : true_type {};
00229 template<> struct is_integral<long> : true_type {};
00230 template<> struct is_integral<unsigned long> : true_type {};
00231 template<> struct is_integral<long long> : true_type {};
00232 template<> struct is_integral<unsigned long long> : true_type {};
00233
00234 template< typename T >
00235 inline constexpr bool is_integral_v = is_integral<T>::value;
00236
00237 template< typename T, bool = is_integral_v<T> || is_enum_v<T> >
00238 struct __is_signed_helper : integral_constant<bool, static_cast<bool>(T(-1) < T(0))> {};;
00239
00240 template< typename T >
00241 struct __is_signed_helper<T, false> : integral_constant<bool, false> {};;
00242
00243 template< typename T >
00244 struct is_signed : __is_signed_helper<T> {};;
00245
00246 template< typename T >
00247 inline constexpr bool is_signed_v = is_signed<T>::value;
00248
00249
00250 template< typename >
00251 struct is_array : false_type {};;
00252
00253 template< typename T >
00254 struct is_array<T[]> : true_type {};;
00255
00256 template< typename T, unsigned long N >
00257 struct is_array<T[N]> : true_type {};;
00258
00259 template< typename T >
00260 inline constexpr bool is_array_v = is_array<T>::value;
00261
00262 template< typename T, unsigned N >
00263 constexpr unsigned array_size(T const (&)[N]) { return N; }
00264
00265 template< int SIZE, bool SIGN = false, bool = true > struct int_type_for_size;
00266
00267 template<> struct int_type_for_size<sizeof(char), true, true>
00268 { typedef signed char type; };
00269
00270 template<> struct int_type_for_size<sizeof(char), false, true>
00271 { typedef unsigned char type; };
00272
00273 template<> struct int_type_for_size<sizeof(short), true, (sizeof(short) > sizeof(char))>
00274 { typedef short type; };
00275
00276 template<> struct int_type_for_size<sizeof(short), false, (sizeof(short) > sizeof(char))>
00277 { typedef unsigned short type; };
00278
00279 template<> struct int_type_for_size<sizeof(int), true, (sizeof(int) > sizeof(short))>
00280 { typedef int type; };
00281
00282 template<> struct int_type_for_size<sizeof(int), false, (sizeof(int) > sizeof(short))>
00283 { typedef unsigned int type; };
00284
00285 template<> struct int_type_for_size<sizeof(long), true, (sizeof(long) > sizeof(int))>
00286 { typedef long type; };
00287
00288 template<> struct int_type_for_size<sizeof(long), false, (sizeof(long) > sizeof(int))>
00289 { typedef unsigned long type; };
00290
00291 template<> struct int_type_for_size<sizeof(long long), true, (sizeof(long long) > sizeof(long))>
00292 { typedef long long type; };
00293
00294 template<> struct int_type_for_size<sizeof(long long), false, (sizeof(long long) > sizeof(long))>
00295 { typedef unsigned long long type; };
00296
00297 template< int SIZE, bool SIGN = false>
00298 using int_type_for_size_t = typename int_type_for_size<SIZE, SIGN>::type;
00299
00300 template< typename T, class Enable = void > struct underlying_type {};;
00301
00302 template< typename T >
00303 struct underlying_type<T, typename enable_if<is_enum_v<T>::type >
00304 {
00305     typedef int_type_for_size_t<sizeof(T), is_signed_v<T> type;
00306 };

```

```

00307
00308 template< typename T >
00309 using underlying_type_t = typename underlying_type<T>::type;
00310
00311 template< typename T > struct make_signed;
00312 template<> struct make_signed<char> { typedef signed char type; };
00313 template<> struct make_signed<unsigned char> { typedef signed char type; };
00314 template<> struct make_signed<signed char> { typedef signed char type; };
00315 template<> struct make_signed<unsigned int> { typedef signed int type; };
00316 template<> struct make_signed<signed int> { typedef signed int type; };
00317 template<> struct make_signed<unsigned long int> { typedef signed long int type; };
00318 template<> struct make_signed<signed long int> { typedef signed long int type; };
00319 template<> struct make_signed<unsigned long long int> { typedef signed long long int type; };
00320 template<> struct make_signed<signed long long int> { typedef signed long long int type; };
00321 template< typename T > using make_signed_t = typename make_signed<T>::type;
00322
00323 template< typename T > struct make_unsigned;
00324 template<> struct make_unsigned<char> { typedef unsigned char type; };
00325 template<> struct make_unsigned<unsigned char> { typedef unsigned char type; };
00326 template<> struct make_unsigned<signed char> { typedef unsigned char type; };
00327 template<> struct make_unsigned<unsigned int> { typedef unsigned int type; };
00328 template<> struct make_unsigned<signed int> { typedef unsigned int type; };
00329 template<> struct make_unsigned<unsigned long int> { typedef unsigned long int type; };
00330 template<> struct make_unsigned<signed long int> { typedef unsigned long int type; };
00331 template<> struct make_unsigned<unsigned long long int> { typedef unsigned long long int type; };
00332 template<> struct make_unsigned<signed long long int> { typedef unsigned long long int type; };
00333 template< typename T > using make_unsigned_t = typename make_unsigned<T>::type;
00334
00335
00336 template<typename From, typename To>
00337 struct is_convertible
00338 {
00339 private:
00340     struct _true { char x[2]; };
00341     struct _false {};
00342
00343     static _true _helper(To const *);
00344     static _false _helper(...);
00345 public:
00346     enum
00347     {
00348         value = sizeof(_true) == sizeof(_helper(static_cast<From*>(0)))
00349             ? true : false
00350     };
00351
00352     typedef bool value_type;
00353 };
00354
00355 template<typename From, typename To>
00356 inline constexpr bool is_convertible_v = is_convertible<From, To>::value;
00357
00358 template< typename T >
00359 struct is_empty : integral_constant<bool, __is_empty(T)> {};
00360
00361 template< typename T >
00362 inline constexpr bool is_empty_v = is_empty<T>::value;
00363
00364
00365 #if L4_HAS_BUILTIN(__is_function)
00366 template < typename T >
00367 struct is_function : integral_constant<bool, __is_function(T)> {};
00368 #else
00369 template < typename T >
00370 struct is_function : integral_constant<bool, !is_reference_v<T>
00371                                     && !is_const_v<const T> > {};
00372 #endif
00373
00374 template< typename T >
00375 inline constexpr bool is_function_v = is_function<T>::value;
00376
00377 }
00378

```

## 17.222 unique\_ptr

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2013 Technische Universität Dresden.
00004  * Copyright (C) 2014-2017, 2020, 2023-2024 Kernkonzept GmbH.
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008

```

```

00009 #pragma once
00010
00011 #include "type_traits"
00012
00013 namespace cxx
00014 {
00015
00016 template< typename T >
00017 struct default_delete
00018 {
00019     default_delete() {}
00020
00021     template< typename U >
00022     default_delete(default_delete<U> const &) {}
00023
00024     void operator () (T *p) const
00025     { delete p; }
00026 };
00027
00028 template< typename T >
00029 struct default_delete<T[]>
00030 {
00031     default_delete() {}
00032
00033     void operator () (T *p)
00034     { delete [] p; }
00035 };
00036
00037 template< typename T, typename C >
00038 struct unique_ptr_index_op {};
00039
00040 template< typename T, typename C >
00041 struct unique_ptr_index_op<T[], C>
00042 {
00043     typedef T &reference;
00044     reference operator [] (int idx) const
00045     { return static_cast<C const *>(this)->get()[idx]; }
00046 };
00047
00048 template< typename T, typename T_Del = default_delete<T> >
00049 class unique_ptr : public unique_ptr_index_op<T, unique_ptr<T, T_Del> >
00050 {
00051 private:
00052     struct _unspec;
00053     typedef _unspec* _unspec_ptr_type;
00054
00055 public:
00056     typedef cxx::remove_extent_t<T> element_type;
00057     typedef element_type *pointer;
00058     typedef element_type &reference;
00059     typedef T_Del deleter_type;
00060
00061     unique_ptr() : _ptr(pointer()) {}
00062
00063     explicit unique_ptr(pointer p) : _ptr(p) {}
00064
00065     unique_ptr(unique_ptr &&o) : _ptr(o.release()) {}
00066
00067     ~unique_ptr() { reset(); }
00068
00069     unique_ptr &operator = (unique_ptr &&o)
00070     {
00071         reset(o.release());
00072         return *this;
00073     }
00074
00075     unique_ptr &operator = (_unspec_ptr_type)
00076     {
00077         reset();
00078         return *this;
00079     }
00080
00081     element_type &operator * () const { return *get(); }
00082     pointer operator -> () const { return get(); }
00083
00084     pointer get() const { return _ptr; }
00085
00086     operator _unspec_ptr_type () const
00087     { return reinterpret_cast<_unspec_ptr_type>(get()); }
00088
00089     pointer release()
00090     {
00091         pointer r = _ptr;
00092         _ptr = 0;
00093         return r;
00094     }
00095

```

```

00096 void reset(pointer p = pointer())
00097 {
00098     if (p != get())
00099     {
00100         deleter_type()(get());
00101         _ptr = p;
00102     }
00103 }
00104
00105 unique_ptr(unique_ptr const &) = delete;
00106 unique_ptr &operator = (unique_ptr const &) = delete;
00107
00108 private:
00109     pointer _ptr;
00110 };
00111
00112 template< typename T >
00113 unique_ptr<T>
00114 make_unique_ptr(T *p)
00115 { return unique_ptr<T>(p); }
00116
00117 template< typename T >
00118 cxx::enable_if_t<cxx::is_array<T>::value, unique_ptr<T>»
00119 make_unique(unsigned long size)
00120 { return cxx::unique_ptr<T>(new cxx::remove_extent_t<T>[size]()); }
00121
00122 template< typename T, typename... Args >
00123 cxx::enable_if_t<!cxx::is_array<T>::value, unique_ptr<T>»
00124 make_unique(Args &&... args)
00125 { return cxx::unique_ptr<T>(new T(cxx::forward<Args>(args)...)); }
00126
00127 }

```

## 17.223 l4/cxx/unique\_ptr\_list File Reference

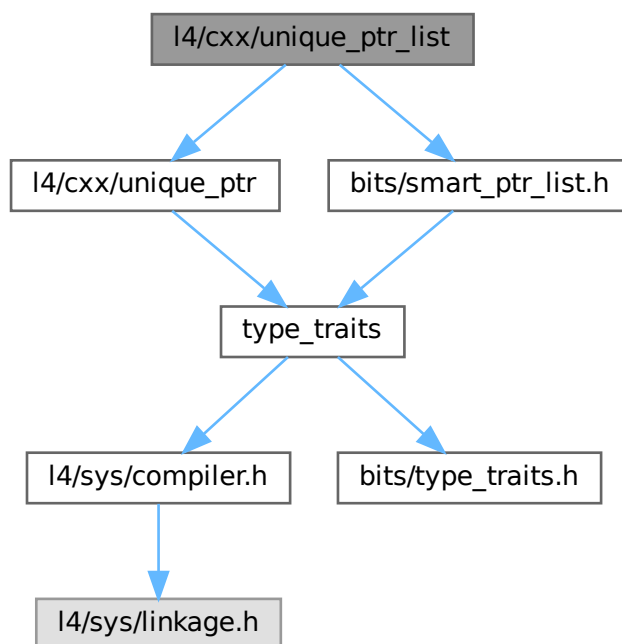
Implementation of a list of unique-ptr-managed objects.

```

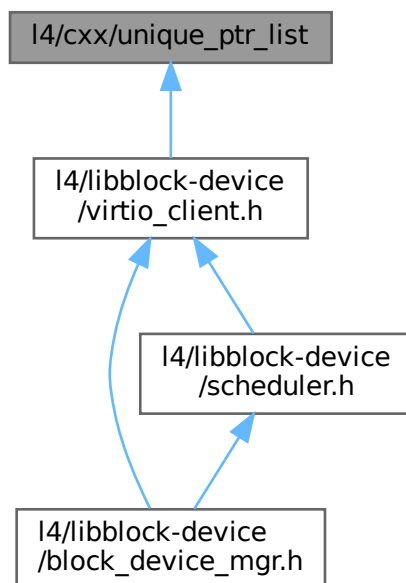
#include <l4/cxx/unique_ptr>
#include "bits/smart_ptr_list.h"

```

Include dependency graph for unique\_ptr\_list:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace `cxx`  
*Our C++ library.*

## Typedefs

- `template<typename T>`  
`using cxx::Unique_ptr_list_item = Bits::Smart_ptr_list_item<T, cxx::unique_ptr<T> >`  
*Item for list linked with `cxx::unique_ptr`.*
- `template<typename T>`  
`using cxx::Unique_ptr_list = Bits::Smart_ptr_list<Unique_ptr_list_item<T> >`  
*Single-linked list where elements are connected with a `cxx::unique_ptr`.*

### 17.223.1 Detailed Description

Implementation of a list of unique-ptr-managed objects.

Definition in file `unique_ptr_list`.

## 17.224 `unique_ptr_list`

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * Copyright (C) 2018-2019, 2022, 2024 Kernkonzept GmbH.
00008  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 #include <l4/cxx/unique_ptr>
00015
00016 #include "bits/smart_ptr_list.h"
00017
00018 namespace cxx {
00019
00021 template <typename T>
00022 using Unique_ptr_list_item = Bits::Smart_ptr_list_item<T, cxx::unique_ptr<T> >;
00023
00027 template <typename T>
00028 using Unique_ptr_list = Bits::Smart_ptr_list<Unique_ptr_list_item<T> >;
00029
00030 }
```

### 17.225 `utils`

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2013 Technische Universität Dresden.
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 namespace cxx {
00011
00039 template< typename T > inline
00040 T access_once(T const *a)
```



```

00041 {
00042 #if 1
00043     __asm__ __volatile__ ( "" : "=m"(*const_cast<T*>(a));
00044         T tmp = *a;
00045     __asm__ __volatile__ ( "" : "=m"(*const_cast<T*>(a));
00046         return tmp;
00047 #else
00048     return *static_cast<T const volatile *>(a);
00049 #endif
00050 }
00051
00070 template< typename T, typename VAL > inline
00071 void write_now(T *a, VAL &&val)
00072 {
00073     __asm__ __volatile__ ( "" : "=m"(*a));
00074     *a = val;
00075     __asm__ __volatile__ ( "" : : "m"(*a));
00076 }
00077
00078
00079 }
00080

```

## 17.226 weak\_ref

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2015, 2017, 2024 Kernkonzept GmbH.
00004  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00005  *             Alexander Warg <alexander.warg@kernkonzept.com>
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include "hlist"
00012
00013 namespace cxx {
00014
00024 class Weak_ref_base : public H_list_item_t<Weak_ref_base>
00025 {
00026 protected:
00027     Weak_ref_base(void const *ptr = nullptr) : _obj(ptr) {}
00028     void reset_hard() { _obj = nullptr; }
00029     void const *_obj;
00030
00031 public:
00038     struct List : H_list_t<Weak_ref_base>
00039     {
00040         void reset()
00041         {
00042             while (!empty())
00043                 pop_front()->reset_hard();
00044         }
00045
00046         ~List()
00047         { reset(); }
00048     };
00049
00050     explicit operator bool () const
00051     { return _obj ? true : false; }
00052 };
00053
00054
00094 template <typename T>
00095 class Weak_ref : public Weak_ref_base
00096 {
00097 public:
00098     T *get() const
00099     { return reinterpret_cast<T*>(const_cast<void *>(_obj)); }
00100
00101     T *reset(T *n)
00102     {
00103         T *r = get();
00104         if (r)
00105             r->remove_weak_ref(this);
00106
00107         _obj = n;
00108         if (n)
00109             n->add_weak_ref(this);
00110
00111         return r;
00112     }

```

```

00113
00114 Weak_ref(T *s = nullptr) : Weak_ref_base(s)
00115 {
00116     if (s)
00117         s->add_weak_ref(this);
00118 }
00119
00120 ~Weak_ref() { reset(0); }
00121
00122 void operator = (T *n)
00123 { reset(n); }
00124
00125 Weak_ref(Weak_ref const &o) : Weak_ref_base(o._obj)
00126 {
00127     if (T *x = get())
00128         x->add_weak_ref(this);
00129 }
00130
00131 Weak_ref &operator = (Weak_ref const &o)
00132 {
00133     if (&o == this)
00134         return *this;
00135
00136     reset(o.get());
00137     return *this;
00138 }
00139
00140 T &operator * () const { return get(); }
00141 T *operator -> () const { return get(); }
00142 };
00143
00144 class Weak_ref_obj
00145 {
00146 protected:
00147     template <typename T> friend class Weak_ref;
00148     mutable Weak_ref_base::List weak_references;
00149
00150     void add_weak_ref(Weak_ref_base *ref) const
00151     { weak_references.push_front(ref); }
00152
00153     void remove_weak_ref(Weak_ref_base *ref) const
00154     { weak_references.remove(ref); }
00155 };
00156
00157 }

```

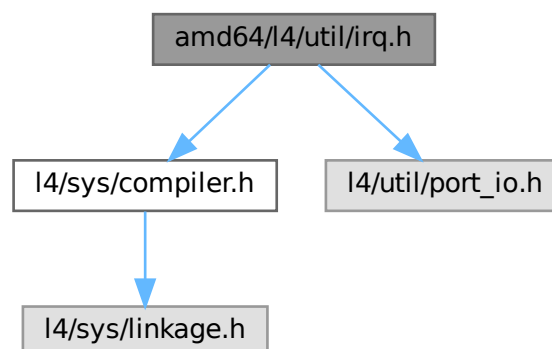
## 17.227 amd64/l4/util/irq.h File Reference

some PIC and hardware interrupt related functions

```
#include <l4/sys/compiler.h>
```

```
#include <l4/util/port_io.h>
```

Include dependency graph for irq.h:



## 17.227.1 Detailed Description

some PIC and hardware interrupt related functions

### Date

2003

### Author

Jork Loeser [jork.loeser@inf.tu-dresden.de](mailto:jork.loeser@inf.tu-dresden.de) Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [irq.h](#).

## 17.228 irq.h

[Go to the documentation of this file.](#)

```

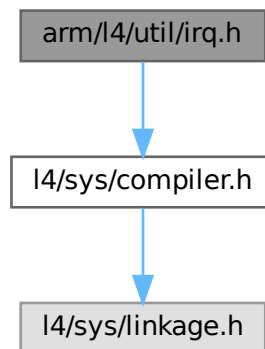
00001
00008
00009 /*
00010  * (c) 2003-2009 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014
00015 #ifndef __L4_IRQ_H__
00016 #define __L4_IRQ_H__
00017
00018 #include <l4/sys/compiler.h>
00019 #include <l4/util/port_io.h>
00020
00021 L4_BEGIN_DECLS
00022
00027
00030 static inline void
00031 l4util_cli (void)
00032 {
00033     __asm__ __volatile__ ("cli" : : : "memory");
00034 }
00035
00038 static inline void
00039 l4util_sti (void)
00040 {
00041     __asm__ __volatile__ ("sti" : : : "memory");
00042 }
00043
00047 static inline void
00048 l4util_flags_save (l4_umword_t *flags)
00049 {
00050     __asm__ __volatile__ ("pushf ; popq %0" : "=g" (*flags) : : "memory");
00051 }
00052
00055 static inline void
00056 l4util_flags_restore (l4_umword_t *flags)
00057 {
00058     __asm__ __volatile__ ("pushq %0 ; popf" : : "g" (*flags) : "memory");
00059 }
00061
00062 L4_END_DECLS
00063
00064 #endif

```

## 17.229 arm/l4/util/irq.h File Reference

ARM specific implementation of irq functions.

#include <l4/sys/compiler.h>  
Include dependency graph for irq.h:



### 17.229.1 Detailed Description

ARM specific implementation of irq functions.

Do not use.

Definition in file [irq.h](#).

## 17.230 irq.h

[Go to the documentation of this file.](#)

```

00001
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00005  *      Frank Mehnert <fm3@os.inf.tu-dresden.de>
00006  *      economic rights: Technische Universität Dresden (Germany)
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #ifndef __L4UTIL__ARCH_ARCH__IRQ_H__
00010 #define __L4UTIL__ARCH_ARCH__IRQ_H__
00011
00012 #ifdef __GNUC__
00013 #include <l4/sys/compiler.h>
00014
00015 L4_BEGIN_DECLS
00016
00017 L4_INLINE void l4util_cli (void);
00018 L4_INLINE void l4util_sti (void);
00019 L4_INLINE void l4util_flags_save(l4_umword_t *flags);
00020 L4_INLINE void l4util_flags_restore(l4_umword_t *flags);
00021
00022 L4_INLINE
  
```

```

00029 void
00030 l4util_cli(void)
00031 {
00032     extern void __do_not_use_l4util_cli(void);
00033     __do_not_use_l4util_cli();
00034 }
00035
00036
00037 L4_INLINE
00038 void
00039 l4util_sti(void)
00040 {
00041     extern void __do_not_use_l4util_sti(void);
00042     __do_not_use_l4util_sti();
00043 }
00044
00045
00046 L4_INLINE
00047 void
00048 l4util_flags_save(l4_umword_t *flags)
00049 {
00050     (void) flags;
00051     extern void __do_not_use_l4util_flags_save(void);
00052     __do_not_use_l4util_flags_save();
00053 }
00054
00055 L4_INLINE
00056 void
00057 l4util_flags_restore(l4_umword_t *flags)
00058 {
00059     (void) flags;
00060     extern void __do_not_use_l4util_flags_restore(void);
00061     __do_not_use_l4util_flags_restore();
00062 }
00063
00064 L4_END_DECLS
00065
00066 #endif // __GNUC__
00067
00068 #endif /* ! __L4UTIL__ARCH_ARCH__IRQ_H__ */

```

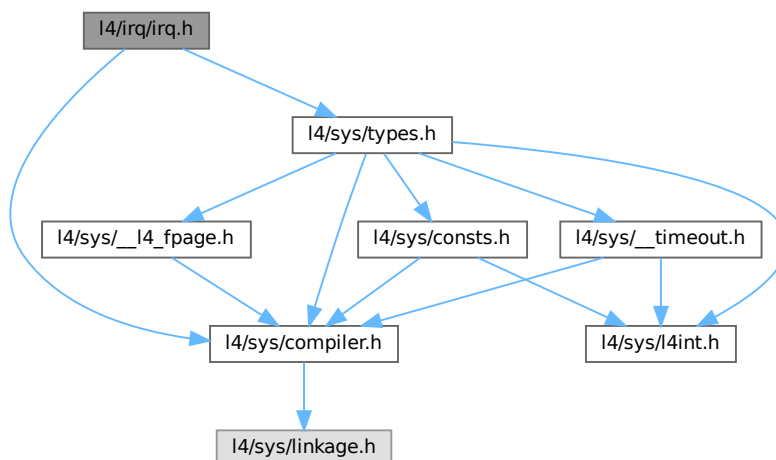
## 17.231 l4/irq/irq.h File Reference

IRQ handling routines.

```
#include <l4/sys/compiler.h>
```

```
#include <l4/sys/types.h>
```

Include dependency graph for irq.h:



## Functions

- `l4irq_t * l4irq_attach` (int irqnum)  
*Attach/connect to IRQ.*
- `l4irq_t * l4irq_attach_ft` (int irqnum, unsigned mode)  
*Attach/connect to IRQ using given type.*
- `l4irq_t * l4irq_attach_thread` (int irqnum, `l4_cap_idx_t` to\_thread)  
*Attach/connect to IRQ.*
- `l4irq_t * l4irq_attach_thread_ft` (int irqnum, `l4_cap_idx_t` to\_thread, unsigned mode)  
*Attach/connect to IRQ using given type.*
- `long l4irq_wait` (`l4irq_t` \*irq)  
*Wait for specified IRQ.*
- `long l4irq_unmask_and_wait_any` (`l4irq_t` \*unmask\_irq, `l4irq_t` \*\*ret\_irq)  
*Unmask a specific IRQ and wait for any attached IRQ.*
- `long l4irq_wait_any` (`l4irq_t` \*\*irq)  
*Wait for any attached IRQ.*
- `long l4irq_unmask` (`l4irq_t` \*irq)  
*Unmask a specific IRQ.*
- `long l4irq_detach` (`l4irq_t` \*irq)  
*Detach from IRQ.*
- `l4irq_t * l4irq_request` (int irqnum, void(\*isr\_handler)(void \*), void \*isr\_data, int irq\_thread\_prio, unsigned mode)  
*Attach asynchronous ISR handler to IRQ.*
- `long l4irq_release` (`l4irq_t` \*irq)  
*Release asynchronous ISR handler and free resources.*
- `l4irq_t * l4irq_attach_cap` (`l4_cap_idx_t` irqcap)  
*Attach/connect to IRQ.*
- `l4irq_t * l4irq_attach_cap_ft` (`l4_cap_idx_t` irqcap, unsigned mode)  
*Attach/connect to IRQ using given type.*
- `l4irq_t * l4irq_attach_thread_cap` (`l4_cap_idx_t` irqcap, `l4_cap_idx_t` to\_thread)  
*Attach/connect to IRQ.*
- `l4irq_t * l4irq_attach_thread_cap_ft` (`l4_cap_idx_t` irqcap, `l4_cap_idx_t` to\_thread, unsigned mode)  
*Attach/connect to IRQ using given type.*
- `l4irq_t * l4irq_request_cap` (`l4_cap_idx_t` irqcap, void(\*isr\_handler)(void \*), void \*isr\_data, int irq\_thread\_prio, unsigned mode)  
*Attach asynchronous ISR handler to IRQ.*

### 17.231.1 Detailed Description

IRQ handling routines.

Definition in file [irq.h](#).

## 17.232 irq.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Henning Schild <hschild@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  */
00014 #pragma once
00015
00016 #include <l4/sys/compiler.h>
00017 #include <l4/sys/types.h>
00018
00019 __BEGIN_DECLS
00020
00027
00028 struct l4irq_t;
00029 typedef struct l4irq_t l4irq_t;
00030
00041 L4_CV l4irq_t *
00042 l4irq_attach(int irqnum);
00043
00055 L4_CV l4irq_t *
00056 l4irq_attach_ft(int irqnum, unsigned mode);
00057
00068 L4_CV l4irq_t *
00069 l4irq_attach_thread(int irqnum, l4_cap_idx_t to_thread);
00070
00082 L4_CV l4irq_t *
00083 l4irq_attach_thread_ft(int irqnum, l4_cap_idx_t to_thread,
00084                        unsigned mode);
00085
00093 L4_CV long
00094 l4irq_wait(l4irq_t *irq);
00095
00105 L4_CV long
00106 l4irq_unmask_and_wait_any(l4irq_t *unmask_irq, l4irq_t **ret_irq);
00107
00115 L4_CV long
00116 l4irq_wait_any(l4irq_t **irq);
00117
00128 L4_CV long
00129 l4irq_unmask(l4irq_t *irq);
00130
00138 L4_CV long
00139 l4irq_detach(l4irq_t *irq);
00140
00141
00142
00143 /*****
00152
00165 L4_CV l4irq_t *
00166 l4irq_request(int irqnum, void (*isr_handler)(void *), void *isr_data,
00167              int irq_thread_prio, unsigned mode);
00168
00176 L4_CV long
00177 l4irq_release(l4irq_t *irq);
00178
00179
00180
00181 /*****
00182 /*****
00183
00188
00199 L4_CV l4irq_t *
00200 l4irq_attach_cap(l4_cap_idx_t irqcap);
00201
00213 L4_CV l4irq_t *
00214 l4irq_attach_cap_ft(l4_cap_idx_t irqcap, unsigned mode);
00215
00226 L4_CV l4irq_t *
00227 l4irq_attach_thread_cap(l4_cap_idx_t irqcap, l4_cap_idx_t to_thread);
00228
00240 L4_CV l4irq_t *
00241 l4irq_attach_thread_cap_ft(l4_cap_idx_t irqcap, l4_cap_idx_t to_thread,
00242                             unsigned mode);
00243
00244 /*****
00252
00265 L4_CV l4irq_t *

```

```

00266 l4irq_request_cap(l4_cap_idx_t irqcap,
00267                     void (*isr_handler)(void *), void *isr_data,
00268                     int irq_thread_prio, unsigned mode);
00269
00270 __END_DECLS

```

## 17.233 l4/sys/irq.h File Reference

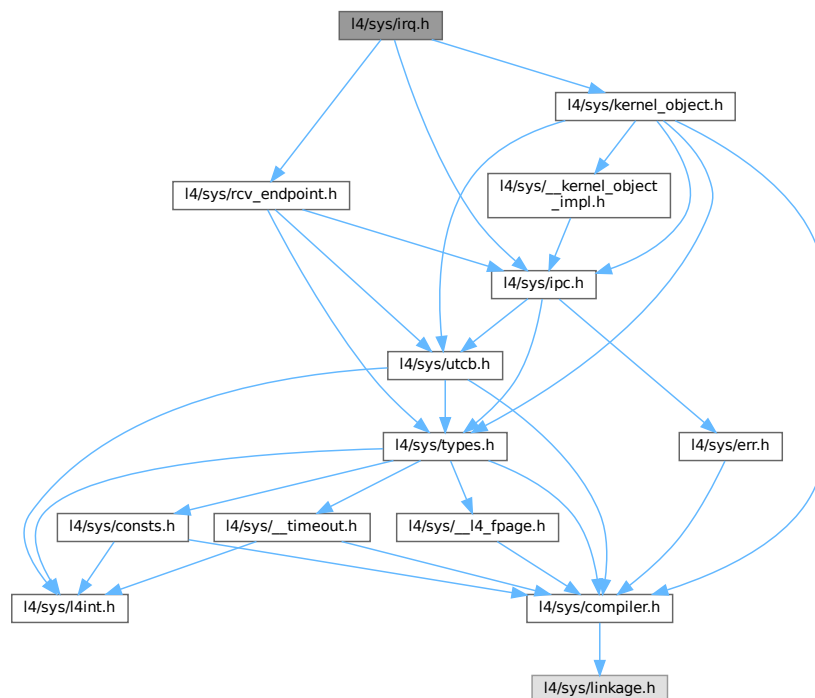
C Irq interface.

```

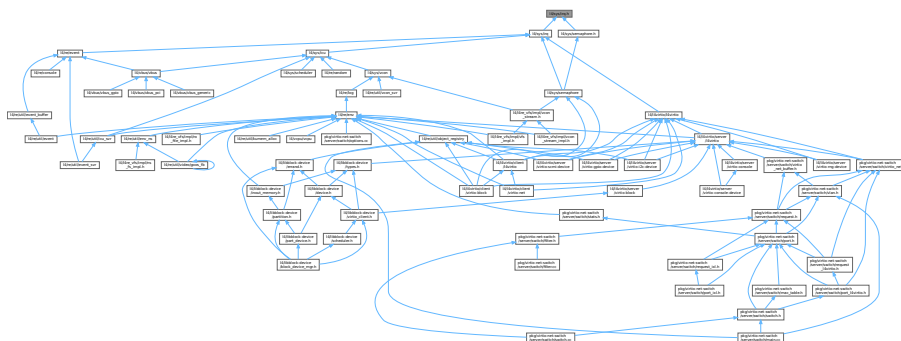
#include <l4/sys/kernel_object.h>
#include <l4/sys/ipc.h>
#include <l4/sys/rcv_endpoint.h>

```

Include dependency graph for irq.h:



This graph shows which files directly or indirectly include this file:





## Functions

- [l4\\_msgtag\\_t l4\\_irq\\_detach \(l4\\_cap\\_idx\\_t irq\) L4\\_NOTHROW](#)  
*Detach from an interrupt source.*
- [l4\\_msgtag\\_t l4\\_irq\\_detach\\_u \(l4\\_cap\\_idx\\_t irq, l4\\_utcb\\_t \\*utcb\) L4\\_NOTHROW](#)  
*Detach from this interrupt.*
- [l4\\_msgtag\\_t l4\\_irq\\_bind\\_vcpu \(l4\\_cap\\_idx\\_t irq, l4\\_cap\\_idx\\_t thread, l4\\_umword\\_t cfg\) L4\\_NOTHROW](#)  
*Bind a thread to this Irq for vCPU interrupt forwarding.*
- [l4\\_msgtag\\_t l4\\_irq\\_bind\\_vcpu\\_u \(l4\\_cap\\_idx\\_t irq, l4\\_cap\\_idx\\_t thread, l4\\_umword\\_t cfg, l4\\_utcb\\_t \\*utcb\) L4\\_NOTHROW](#)  
*Bind a thread to this Irq for vCPU interrupt forwarding.*
- [l4\\_msgtag\\_t l4\\_irq\\_trigger \(l4\\_cap\\_idx\\_t irq\) L4\\_NOTHROW](#)  
*Trigger an IRQ.*
- [l4\\_msgtag\\_t l4\\_irq\\_trigger\\_u \(l4\\_cap\\_idx\\_t irq, l4\\_utcb\\_t \\*utcb\) L4\\_NOTHROW](#)  
*Trigger the object.*
- [l4\\_msgtag\\_t l4\\_irq\\_receive \(l4\\_cap\\_idx\\_t irq, l4\\_timeout\\_t to\) L4\\_NOTHROW](#)  
*Unmask and wait for specified IRQ.*
- [l4\\_msgtag\\_t l4\\_irq\\_receive\\_u \(l4\\_cap\\_idx\\_t irq, l4\\_timeout\\_t timeout, l4\\_utcb\\_t \\*utcb\) L4\\_NOTHROW](#)  
*Unmask and wait for this IRQ.*
- [l4\\_msgtag\\_t l4\\_irq\\_wait \(l4\\_cap\\_idx\\_t irq, l4\\_umword\\_t \\*label, l4\\_timeout\\_t to\) L4\\_NOTHROW](#)  
*Unmask IRQ and wait for any message.*
- [l4\\_msgtag\\_t l4\\_irq\\_wait\\_u \(l4\\_cap\\_idx\\_t irq, l4\\_umword\\_t \\*label, l4\\_timeout\\_t timeout, l4\\_utcb\\_t \\*utcb\) L4\\_NOTHROW](#)  
*Unmask IRQ and (open) wait for any message.*
- [l4\\_msgtag\\_t l4\\_irq\\_unmask \(l4\\_cap\\_idx\\_t irq\) L4\\_NOTHROW](#)  
*Unmask IRQ.*
- [l4\\_msgtag\\_t l4\\_irq\\_unmask\\_u \(l4\\_cap\\_idx\\_t irq, l4\\_utcb\\_t \\*utcb\) L4\\_NOTHROW](#)  
*Unmask this IRQ.*

### 17.233.1 Detailed Description

C Irq interface.

Definition in file [irq.h](#).

## 17.234 irq.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Björn Döbel <doebel@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #pragma once
00015
00016 #include <l4/sys/kernel_object.h>
00017 #include <l4/sys/ipc.h>
00018 #include <l4/sys/rcv_endpoint.h>
00019
00046
00062 L4_INLINE l4_msgtag_t
00063 l4_irq_detach(l4_cap_idx_t irq) L4_NOTHROW;
```

```

00064
00071 L4_INLINE l4_msgtag_t
00072 l4_irq_detach_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW;
00073
00074
00109 L4_INLINE l4_msgtag_t
00110 l4_irq_bind_vcpu(l4_cap_idx_t irq, l4_cap_idx_t thread,
00111                 l4_umword_t cfg) L4_NOTHROW;
00112
00119 L4_INLINE l4_msgtag_t
00120 l4_irq_bind_vcpu_u(l4_cap_idx_t irq, l4_cap_idx_t thread, l4_umword_t cfg,
00121                   l4_utcb_t *utcb) L4_NOTHROW;
00122
00123
00139 L4_INLINE l4_msgtag_t
00140 l4_irq_trigger(l4_cap_idx_t irq) L4_NOTHROW;
00141
00148 L4_INLINE l4_msgtag_t
00149 l4_irq_trigger_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW;
00150
00160 L4_INLINE l4_msgtag_t
00161 l4_irq_receive(l4_cap_idx_t irq, l4_timeout_t to) L4_NOTHROW;
00162
00169 L4_INLINE l4_msgtag_t
00170 l4_irq_receive_u(l4_cap_idx_t irq, l4_timeout_t timeout, l4_utcb_t *utcb) L4_NOTHROW;
00171
00182 L4_INLINE l4_msgtag_t
00183 l4_irq_wait(l4_cap_idx_t irq, l4_umword_t *label,
00184            l4_timeout_t to) L4_NOTHROW;
00185
00192 L4_INLINE l4_msgtag_t
00193 l4_irq_wait_u(l4_cap_idx_t irq, l4_umword_t *label,
00194              l4_timeout_t timeout, l4_utcb_t *utcb) L4_NOTHROW;
00195
00206 L4_INLINE l4_msgtag_t
00207 l4_irq_unmask(l4_cap_idx_t irq) L4_NOTHROW;
00208
00215 L4_INLINE l4_msgtag_t
00216 l4_irq_unmask_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW;
00217
00221 enum L4_irq_sender_op
00222 {
00223     L4_IRQ_SENDER_OP_RESERVED1 = 0, // Ex ATTACH
00224     L4_IRQ_SENDER_OP_DETACH     = 1,
00225     L4_IRQ_SENDER_OP_BIND_VCPU  = 2,
00226 };
00227
00231 enum L4_irq_op
00232 {
00233     L4_IRQ_OP_TRIGGER    = 2,
00234     L4_IRQ_OP_EOI       = 4
00235 };
00236
00237 /*****
00238  * Implementations
00239  */
00240
00241 L4_INLINE l4_msgtag_t
00242 l4_irq_detach_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW
00243 {
00244     l4_utcb_mr_u(utcb)->mr[0] = L4_IRQ_SENDER_OP_DETACH;
00245     return l4_ipc_call(irq, utcb, l4_msgtag(L4_PROTO_IRQ_SENDER, 1, 0, 0),
00246                      L4_IPC_NEVER);
00247 }
00248
00249 L4_INLINE l4_msgtag_t
00250 l4_irq_bind_vcpu_u(l4_cap_idx_t irq, l4_cap_idx_t thread, l4_umword_t cfg,
00251                   l4_utcb_t *utcb) L4_NOTHROW
00252 {
00253     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00254     m->mr[0] = L4_IRQ_SENDER_OP_BIND_VCPU;
00255     m->mr[1] = cfg;
00256     m->mr[2] = l4_map_obj_control(0, 0);
00257     m->mr[3] = l4_obj_fpage(thread, 0, L4_CAP_FPAGE_RWS).raw;
00258     return l4_ipc_call(irq, utcb, l4_msgtag(L4_PROTO_IRQ_SENDER, 2, 1, 0),
00259                      L4_IPC_NEVER);
00260 }
00261
00262 L4_INLINE l4_msgtag_t
00263 l4_irq_trigger_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW
00264 {
00265     return l4_ipc_send(irq, utcb, l4_msgtag(L4_PROTO_IRQ, 0, 0, 0),
00266                      L4_IPC_BOTH_TIMEOUT_0);
00267 }
00268
00269 L4_INLINE l4_msgtag_t
00270 l4_irq_receive_u(l4_cap_idx_t irq, l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW

```

```

00271 {
00272     l4_utcb_mr_u(utcb)->mr[0] = L4_IRQ_OP_EOI;
00273     return l4_ipc_call(irq, utcb, l4_msgtag(L4_PROTO_IRQ, 1, 0, 0), to);
00274 }
00275
00276 L4_INLINE l4_msgtag_t
00277 l4_irq_wait_u(l4_cap_idx_t irq, l4_umword_t *label,
00278               l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW
00279 {
00280     l4_utcb_mr_u(utcb)->mr[0] = L4_IRQ_OP_EOI;
00281     return l4_ipc_send_and_wait(irq, utcb, l4_msgtag(L4_PROTO_IRQ, 1, 0, 0),
00282                                 label, to);
00283 }
00284
00285 L4_INLINE l4_msgtag_t
00286 l4_irq_unmask_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW
00287 {
00288     l4_utcb_mr_u(utcb)->mr[0] = L4_IRQ_OP_EOI;
00289     return l4_ipc_send(irq, utcb, l4_msgtag(L4_PROTO_IRQ, 1, 0, 0), L4_IPC_NEVER);
00290 }
00291
00292
00293 L4_INLINE l4_msgtag_t
00294 l4_irq_detach(l4_cap_idx_t irq) L4_NOTHROW
00295 {
00296     return l4_irq_detach_u(irq, l4_utcb());
00297 }
00298
00299 L4_INLINE l4_msgtag_t
00300 l4_irq_bind_vcpu(l4_cap_idx_t irq, l4_cap_idx_t thread,
00301                  l4_umword_t cfg) L4_NOTHROW
00302 {
00303     return l4_irq_bind_vcpu_u(irq, thread, cfg, l4_utcb());
00304 }
00305
00306 L4_INLINE l4_msgtag_t
00307 l4_irq_trigger(l4_cap_idx_t irq) L4_NOTHROW
00308 {
00309     return l4_irq_trigger_u(irq, l4_utcb());
00310 }
00311
00312 L4_INLINE l4_msgtag_t
00313 l4_irq_receive(l4_cap_idx_t irq, l4_timeout_t to) L4_NOTHROW
00314 {
00315     return l4_irq_receive_u(irq, to, l4_utcb());
00316 }
00317
00318 L4_INLINE l4_msgtag_t
00319 l4_irq_wait(l4_cap_idx_t irq, l4_umword_t *label,
00320             l4_timeout_t to) L4_NOTHROW
00321 {
00322     return l4_irq_wait_u(irq, label, to, l4_utcb());
00323 }
00324
00325 L4_INLINE l4_msgtag_t
00326 l4_irq_unmask(l4_cap_idx_t irq) L4_NOTHROW
00327 {
00328     return l4_irq_unmask_u(irq, l4_utcb());
00329 }
00330

```

## 17.235 x86/I4/util/irq.h File Reference

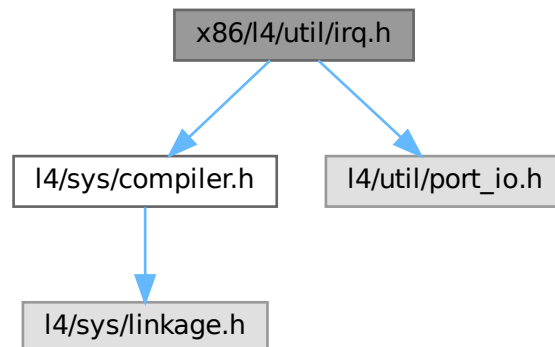
some PIC and hardware interrupt related functions

```

#include <l4/sys/compiler.h>
#include <l4/util/port_io.h>

```

Include dependency graph for irq.h:



### 17.235.1 Detailed Description

some PIC and hardware interrupt related functions

Date

2003

Author

Jork Loeser [jork.loeser@inf.tu-dresden.de](mailto:jork.loeser@inf.tu-dresden.de) Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [irq.h](#).

## 17.236 irq.h

[Go to the documentation of this file.](#)

```

00001
00008
00009 /*
00010  * (c) 2003-2009 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014
00015 #ifndef __L4_IRQ_H__
00016 #define __L4_IRQ_H__
00017
00018 #include <l4/sys/compiler.h>
00019 #include <l4/util/port_io.h>
00020
00021 L4_BEGIN_DECLS
00022
00027
00030 static inline void
00031 l4util_cli (void)
00032 {
00033     __asm__ __volatile__ ("cli" : : : "memory");

```

```

00034 }
00035
00038 static inline void
00039 l4util_sti (void)
00040 {
00041     __asm__ __volatile__ ("sti" : : "memory");
00042 }
00043
00047 static inline void
00048 l4util_flags_save (l4_umword_t *flags)
00049 {
00050     __asm__ __volatile__ ("pushfl ; popl %0 " : "=g" (*flags) : "memory");
00051 }
00052
00055 static inline void
00056 l4util_flags_restore (l4_umword_t *flags)
00057 {
00058     __asm__ __volatile__ ("pushl %0 ; popfl " : : "g" (*flags) : "memory");
00059 }
00061
00062 L4_END_DECLS
00063
00064 #endif

```

## 17.237 backend

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include <l4/l4re_vfs/vfs.h>
00012 #include <l4/crtn/initpriorities.h>
00013
00014 extern "C" void l4re_vfs_select_poll_notify(void);
00015
00016 namespace L4Re { namespace Vfs {
00017
00019 extern L4Re::Vfs::Ops *vfs_ops asm ("l4re_env_posix_vfs_ops");
00020
00021 class Mount_tree;
00022
00030 class Be_file : public File
00031 {
00032 public:
00033     void *operator new (size_t size) noexcept
00034     { return vfs_ops->malloc(size); }
00035
00036     void *operator new (size_t, void *m) noexcept
00037     { return m; }
00038
00039     void operator delete (void *m)
00040     { vfs_ops->free(m); }
00041
00042     // used in close, to unlock all locks of a file (as POSIX says)
00043     int unlock_all_locks() noexcept override
00044     { return 0; }
00045
00046     // for mmap
00047     L4::Cap<L4Re::Dataspace> data_space() noexcept override
00048     { return L4::Cap<L4Re::Dataspace>::Invalid; }
00049
00051     ssize_t readv(const struct iovec*, int) noexcept override
00052     { return -EINVAL; }
00053
00055     ssize_t writev(const struct iovec*, int) noexcept override
00056     { return -EINVAL; }
00057
00059     ssize_t pwritev(const struct iovec*, int, off64_t) noexcept override
00060     { return -EINVAL; }
00061
00063     ssize_t preadv(const struct iovec*, int, off64_t) noexcept override
00064     { return -EINVAL; }
00065
00067     off64_t lseek(off64_t, int) noexcept override
00068     { return -ESPIPE; }
00069
00071     int ftruncate(off64_t) noexcept override

```

```

00072     { return -EINVAL; }
00073
00075 int fsync() const noexcept override
00076 { return -EINVAL; }
00077
00079 int fdatasync() const noexcept override
00080 { return -EINVAL; }
00081
00083 int ioctl(unsigned long, va_list) noexcept override
00084 { return -EINVAL; }
00085
00086 int fstat(struct stat64 *) const noexcept override
00087 { return -EINVAL; }
00088
00090 int fchmod(mode_t) noexcept override
00091 { return -EINVAL; }
00092
00094 int get_status_flags() const noexcept override
00095 { return 0; }
00096
00098 int set_status_flags(long) noexcept override
00099 { return 0; }
00100
00102 int get_lock(struct flock64 *) noexcept override
00103 { return -EOLCK; }
00104
00106 int set_lock(struct flock64 *, bool) noexcept override
00107 { return -EOLCK; }
00108
00110 int faccessat(const char *, int, int) noexcept override
00111 { return -ENOTDIR; }
00112
00114 int fchmodat(const char *, mode_t, int) noexcept override
00115 { return -ENOTDIR; }
00116
00118 int utime(const struct utimbuf *) noexcept override
00119 { return -EROFS; }
00120
00122 int utimes(const struct timeval [2]) noexcept override
00123 { return -EROFS; }
00124
00126 int utimensat(const char *, const struct timespec [2], int) noexcept override
00127 { return -EROFS; }
00128
00130 int mkdir(const char *, mode_t) noexcept override
00131 { return -ENOTDIR; }
00132
00134 int unlink(const char *) noexcept override
00135 { return -ENOTDIR; }
00136
00138 int rename(const char *, const char *) noexcept override
00139 { return -ENOTDIR; }
00140
00142 int link(const char *, const char *) noexcept override
00143 { return -ENOTDIR; }
00144
00146 int symlink(const char *, const char *) noexcept override
00147 { return -EPERM; }
00148
00150 int rmdir(const char *) noexcept override
00151 { return -ENOTDIR; }
00152
00154 ssize_t readlink(char *, size_t) override
00155 { return -EINVAL; }
00156
00157 ssize_t getdents(char *, size_t) noexcept override
00158 { return -ENOTDIR; }
00159
00160
00161
00162 // Socket interface
00163 int bind(sockaddr const *, socklen_t) noexcept override
00164 { return -ENOTSOCK; }
00165
00166 int connect(sockaddr const *, socklen_t) noexcept override
00167 { return -ENOTSOCK; }
00168
00169 ssize_t send(void const *, size_t, int) noexcept override
00170 { return -ENOTSOCK; }
00171
00172 ssize_t recv(void *, size_t, int) noexcept override
00173 { return -ENOTSOCK; }
00174
00175 ssize_t sendto(void const *, size_t, int, sockaddr const *, socklen_t) noexcept
00176     override
00177 { return -ENOTSOCK; }
00178

```

```

00179     ssize_t recvfrom(void *, size_t, int, sockaddr *, socklen_t *) noexcept override
00180     { return -ENOTSOCK; }
00181
00182     ssize_t sendmsg(msghdr const *, int) noexcept override
00183     { return -ENOTSOCK; }
00184
00185     ssize_t recvmsg(msghdr *, int) noexcept override
00186     { return -ENOTSOCK; }
00187
00188     int getsockopt(int, int, void *, socklen_t *) noexcept override
00189     { return -ENOTSOCK; }
00190
00191     int setsockopt(int, int, void const *, socklen_t) noexcept override
00192     { return -ENOTSOCK; }
00193
00194     int listen(int) noexcept override
00195     { return -ENOTSOCK; }
00196
00197     int accept(sockaddr *, socklen_t *) noexcept override
00198     { return -ENOTSOCK; }
00199
00200     int shutdown(int) noexcept override
00201     { return -ENOTSOCK; }
00202
00203     int getsockname(sockaddr *, socklen_t *) noexcept override
00204     { return -ENOTSOCK; }
00205
00206     int getpeername(sockaddr *, socklen_t *) noexcept override
00207     { return -ENOTSOCK; }
00208
00217     bool check_ready(Ready_type) noexcept override
00218     { return false; }
00219
00220     ~Be_file() noexcept = 0;
00221
00222 private:
00224     int get_entry(const char *, int, mode_t, cxx::Ref_ptr<File> *) noexcept override
00225     { return -ENOTDIR; }
00226
00227 protected:
00228     const char *get_mount(const char *path, cxx::Ref_ptr<File> *dir) noexcept;
00229 };
00230
00231 inline
00232 Be_file::~~Be_file() noexcept {}
00233
00234 class Be_file_pos : public Be_file
00235 {
00236 public:
00237     Be_file_pos() noexcept : Be_file(), _pos(0) {}
00238
00239     virtual off64_t size() const noexcept = 0;
00240
00241     ssize_t readv(const struct iovec *v, int iovcnt) noexcept override
00242     {
00243         ssize_t r = preadv(v, iovcnt, _pos);
00244         if (r > 0)
00245             _pos += r;
00246         return r;
00247     }
00248
00249     ssize_t writev(const struct iovec *v, int iovcnt) noexcept override
00250     {
00251         ssize_t r = pwritev(v, iovcnt, _pos);
00252         if (r > 0)
00253             _pos += r;
00254         return r;
00255     }
00256
00257     ssize_t preadv(const struct iovec *v, int iovcnt, off64_t offset) noexcept override = 0;
00258     ssize_t pwritev(const struct iovec *v, int iovcnt, off64_t offset) noexcept override = 0;
00259
00260     off64_t lseek(off64_t offset, int whence) noexcept override
00261     {
00262         off64_t r;
00263         switch (whence)
00264         {
00265             case SEEK_SET: r = offset; break;
00266             case SEEK_CUR: r = _pos + offset; break;
00267             case SEEK_END: r = size() + offset; break;
00268             default: return -EINVAL;
00269         };
00270
00271         if (r < 0)
00272             return -EINVAL;
00273
00274         _pos = r;

```

```

00275     return _pos;
00276 }
00277
00278 ~Be_file_pos() noexcept = 0;
00279
00280 protected:
00281     off64_t pos() const noexcept { return _pos; }
00282
00283 private:
00284     off64_t _pos;
00285 };
00286
00287 inline Be_file_pos::~Be_file_pos() noexcept {}
00288
00289 class Be_file_stream : public Be_file
00290 {
00291 public:
00292     ssize_t preadv(const struct iovec *v, int iovcnt, off64_t) noexcept override
00293     { return readv(v, iovcnt); }
00294
00295     ssize_t pwritev(const struct iovec *v, int iovcnt, off64_t) noexcept override
00296     { return writev(v, iovcnt); }
00297
00298     ~Be_file_stream() noexcept = 0;
00299 };
00300
00301 inline Be_file_stream::~Be_file_stream() noexcept {}
00302
00303 class Be_file_system : public File_system
00304 {
00305 private:
00306     char const *const _fstype;
00307
00308 public:
00309     explicit Be_file_system(char const *fstype) noexcept
00310     : File_system(), _fstype(fstype)
00311     {
00312         vfs_ops->register_file_system(this);
00313     }
00314
00315     ~Be_file_system() noexcept
00316     {
00317         vfs_ops->unregister_file_system(this);
00318     }
00319
00320     char const *type() const noexcept override { return _fstype; }
00321 };
00322
00323 /* Make sure filesystems can register before the constructor of libmount
00324  * runs */
00325 #define L4RE_VFS_FILE_SYSTEM_ATTRIBUTE \
00326     __attribute__((init_priority(INIT_PRIO_LATE)))
00327
00328 }

```

## 17.238 default\_ops\_impl.h

```

00001 // vi:ft=cpp
00002 /*
00003  * Copyright (C) 2016, 2023-2024 Kernkonzept GmbH.
00004  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #include <l4/cxx/static_container>
00009
00010 namespace {
00011 struct Vfs_init
00012 {
00013     // The Static_containers are used to prevent automatic destruction during
00014     // program shutdown. At least the `vfs` object must never be destructed
00015     // because any later attempt to do any kind of file-descriptor access in
00016     // the program would crash, and we could not be sure that the destructor
00017     // would really be executed after each possible operation using files or file
00018     // descriptors.
00019     cxx::Static_container<Vfs> vfs;
00020
00021     // The Static_containers below are just for providing ordering. The factories
00022     // must be initialized after the `vfs` object.
00023     cxx::Static_container<L4Re::Vfs::File_factory_t<L4Re::Dataspace, L4Re::Core::Ro_file> > ro_file;
00024     cxx::Static_container<L4Re::Vfs::File_factory_t<L4Re::Namespace, L4Re::Core::Ns_dir> > ns_dir;

```



```

00025     cxx::Static_container<L4Re::Vfs::File_factory_t<L4::Vcon, L4Re::Core::Vcon_stream> > vcon_stream;
00026
00027     Vfs_init()
00028     {
00029         vfs.construct();
00030         __rtld_l4re_env_posix_vfs_ops = vfs;
00031         ns_dir.construct();
00032         auto ns_ptr = cxx::ref_ptr(ns_dir.get());
00033         vfs->register_file_factory(ns_ptr);
00034         ns_ptr.release(); // prevent deletion of static object
00035
00036         ro_file.construct();
00037         auto ro_ptr = cxx::ref_ptr(ro_file.get());
00038         vfs->register_file_factory(ro_ptr);
00039         ro_ptr.release(); // prevent deletion of static object
00040
00041         vcon_stream.construct();
00042         auto vcon_ptr = cxx::ref_ptr(vcon_stream.get());
00043         vfs->register_file_factory(vcon_ptr);
00044         vcon_ptr.release(); // prevent deletion of static object
00045     }
00046 };
00047
00048 static Vfs_init __vfs_init __attribute__((init_priority(INIT_PRIO_VFS_INIT)));
00049
00050 };

```

## 17.239 fd\_store.h

```

00001 /*
00002  * (c) 2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/l4re_vfs/vfs.h>
00010
00011 namespace L4Re { namespace Core {
00012
00013     using cxx::Ref_ptr;
00014
00015     class Fd_store
00016     {
00017     public:
00018         enum { MAX_FILES = 50 };
00019
00020         Fd_store() noexcept : _fd_hint(0) {}
00021
00022         int alloc() noexcept;
00023         void free(int fd) noexcept;
00024         bool check_fd(int fd) noexcept;
00025         Ref_ptr<L4Re::Vfs::File> get(int fd) noexcept;
00026         void set(int fd, Ref_ptr<L4Re::Vfs::File> const &f) noexcept;
00027
00028     private:
00029         int _fd_hint;
00030         Ref_ptr<L4Re::Vfs::File> _files[MAX_FILES];
00031     };
00032
00033     inline
00034     bool
00035     Fd_store::check_fd(int fd) noexcept
00036     {
00037         return fd >= 0 && fd < MAX_FILES;
00038     }
00039
00040     inline
00041     Ref_ptr<L4Re::Vfs::File>
00042     Fd_store::get(int fd) noexcept
00043     {
00044         if (check_fd(fd))
00045             return _files[fd];
00046         return Ref_ptr<>::Nil;
00047     }
00048
00049     inline
00050     void
00051     Fd_store::set(int fd, Ref_ptr<L4Re::Vfs::File> const &f) noexcept
00052     {
00053         _files[fd] = f;
00054     }

```

```

00055 }
00056
00057 }}

```

## 17.240 fd\_store\_impl.h

```

00001 /*
00002  * (c) 2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00003  *      economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #include "fd_store.h"
00008
00009 namespace L4Re { namespace Core {
00010
00011     int
00012     Fd_store::alloc() noexcept
00013     {
00014         for (int i = _fd_hint; i < MAX_FILES; ++i)
00015         {
00016             if (!_files[i])
00017             {
00018                 _fd_hint = i + 1;
00019                 return i;
00020             }
00021         }
00022         return -1;
00023     }
00024 }
00025
00026 void
00027 Fd_store::free(int fd) noexcept
00028 {
00029     _files[fd] = 0;
00030     if (fd < _fd_hint)
00031         _fd_hint = fd;
00032 }
00033
00034 }}
00035

```

## 17.241 ns\_fs.h

```

00001 /*
00002  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <l4/l4re_vfs/backend>
00011 #include <l4/sys/capability>
00012 #include <l4/re/namespace>
00013 #include <l4/re/unique_cap>
00014
00015 namespace L4Re { namespace Core {
00016
00017     using cxx::Ref_ptr;
00018
00019     class Env_dir : public L4Re::Vfs::Be_file
00020     {
00021     public:
00022         explicit Env_dir(L4Re::Env const *env)
00023             : _env(env), _current_cap_entry(env->initial_caps())
00024         {}
00025
00026         ssize_t readv(const struct iovec*, int) noexcept override { return -EISDIR; }
00027         ssize_t writev(const struct iovec*, int) noexcept override { return -EISDIR; }
00028         ssize_t preadv(const struct iovec*, int, off64_t) noexcept override { return -EISDIR; }
00029         ssize_t pwritev(const struct iovec*, int, off64_t) noexcept override { return -EISDIR; }
00030         int fstat(struct stat64 *) const noexcept override;
00031         int faccessat(const char *path, int mode, int flags) noexcept override;
00032         int get_entry(const char *path, int flags, mode_t mode,
00033                     Ref_ptr<L4Re::Vfs::File> *) noexcept override;
00034         ssize_t getdents(char *, size_t) noexcept override;
00035
00036     };
00037
00038 } }

```

```

00035
00036 ~Env_dir() noexcept {}
00037
00038 private:
00039     int get_ds(const char *path, L4Re::Unique_cap<L4Re::Dataspace> *ds) noexcept;
00040     bool check_type(Env::Cap_entry const *e, long protocol) noexcept;
00041
00042     L4Re::Env const *_env;
00043     Env::Cap_entry const *_current_cap_entry;
00044 };
00045
00046 class Ns_dir : public L4Re::Vfs::Be_file
00047 {
00048 public:
00049     explicit Ns_dir(L4::Cap<L4Re::Namespace> ns)
00050         : _ns(ns), _current_dir_pos(0)
00051     {}
00052
00053     ssize_t readv(const struct iovec*, int) noexcept override { return -EISDIR; }
00054     ssize_t writev(const struct iovec*, int) noexcept override { return -EISDIR; }
00055     ssize_t preadv(const struct iovec*, int, off64_t) noexcept override { return -EISDIR; }
00056     ssize_t pwritev(const struct iovec*, int, off64_t) noexcept override { return -EISDIR; }
00057     int fstat(struct stat64 *) const noexcept override;
00058     int faccessat(const char *path, int mode, int flags) noexcept override;
00059     int get_entry(const char *path, int flags, mode_t mode,
00060                  Ref_ptr<L4Re::Vfs::File> *) noexcept override;
00061     ssize_t getdents(char *, size_t) noexcept override;
00062
00063 ~Ns_dir() noexcept {}
00064
00065 private:
00066     int get_ds(const char *path, L4Re::Unique_cap<L4Re::Dataspace> *ds) noexcept;
00067
00068     L4::Cap<L4Re::Namespace> _ns;
00069     size_t _current_dir_pos;
00070 };
00071
00072 }

```

## 17.242 ns\_fs\_impl.h

```

00001 /*
00002  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #include "ns_fs.h"
00009
00010 #include <l4/re/dataspace>
00011 #include <l4/re/util/env_ns>
00012 #include <l4/re/unique_cap>
00013 #include <dirent.h>
00014
00015 namespace L4Re { namespace Core {
00016
00017     static
00018     Ref_ptr<L4Re::Vfs::File>
00019     cap_to_vfs_object(L4::Cap<void> o, int *err)
00020     {
00021         L4::Cap<L4::Meta> m = L4::cap_reinterpret_cast<L4::Meta>(o);
00022         long proto = 0;
00023         char name_buf[256];
00024         L4::Ipc::String<char> name(sizeof(name_buf), name_buf);
00025         int r = l4_error(m->interface(0, &proto, &name));
00026         *err = -ENOPROTOPT;
00027         if (r < 0)
00028             // could not get type of object so bail out
00029             return Ref_ptr<L4Re::Vfs::File>();
00030
00031         *err = -EPROTO;
00032         Ref_ptr<L4Re::Vfs::File_factory> factory;
00033
00034         if (proto != 0)
00035             factory = L4Re::Vfs::vfs_ops->get_file_factory(proto);
00036
00037         if (!factory)
00038             factory = L4Re::Vfs::vfs_ops->get_file_factory(name.data());
00039
00040         if (!factory)
00041             return Ref_ptr<L4Re::Vfs::File>();
00042

```

```

00043     *err = -ENOMEM;
00044     return factory->create(o);
00045 }
00046
00047
00048 int
00049 Ns_dir::get_ds(const char *path, L4Re::Unique_cap<L4Re::Dataspace> *ds) noexcept
00050 {
00051     auto file = L4Re::make_unique_cap<L4Re::Dataspace>(L4Re::virt_cap_alloc);
00052     if (!file.is_valid())
00053         return -ENOMEM;
00054     int err = _ns->query(path, file.get());
00055     if (err < 0)
00056         return -ENOENT;
00057     *ds = cxx::move(file);
00058     return err;
00059 }
00060
00061 int
00062 Ns_dir::get_entry(const char *path, int /*flags*/, mode_t /*mode*/,
00063                  Ref_ptr<L4Re::Vfs::File> *f) noexcept
00064 {
00065     if (!*path)
00066     {
00067         *f = cxx::ref_ptr(this);
00068         return 0;
00069     }
00070     L4Re::Unique_cap<Dataspace> file;
00071     int err = get_ds(path, &file);
00072     if (err < 0)
00073         return -ENOENT;
00074     cxx::Ref_ptr<L4Re::Vfs::File> fi = cap_to_vfs_object(file.get(), &err);
00075     if (!fi)
00076         return err;
00077     file.release();
00078     *f = cxx::move(fi);
00079     return 0;
00080 }
00081
00082 int
00083 Ns_dir::faccessat(const char *path, int mode, int /*flags*/) noexcept
00084 {
00085     auto tmpcap = L4Re::make_unique_cap<void>(L4Re::virt_cap_alloc);
00086     if (!tmpcap.is_valid())
00087         return -ENOMEM;
00088     if (_ns->query(path, tmpcap.get()))
00089         return -ENOENT;
00090     if (mode & W_OK)
00091         return -EACCES;
00092     return 0;
00093 }
00094
00095 int
00096 Ns_dir::fstat(struct stat64 *b) const noexcept
00097 {
00098     b->st_dev = 1;
00099     b->st_ino = 1;
00100     b->st_mode = S_IRWXU | S_IFDIR;
00101     b->st_nlink = 0;
00102     b->st_uid = 0;
00103     b->st_gid = 0;
00104     b->st_rdev = 0;
00105     b->st_size = 0;
00106     b->st_blksize = 0;
00107     b->st_blocks = 0;
00108     b->st_atime = 0;
00109     b->st_mtime = 0;
00110     b->st_ctime = 0;
00111     return 0;
00112 }
00113
00114 ssize_t
00115 Ns_dir::getdents(char *buf, size_t dest_sz) noexcept
00116 {
00117     struct dirent64 *dest = reinterpret_cast<struct dirent64 *>(buf);

```

```

00130     ssize_t ret = 0;
00131     l4_addr_t infoaddr;
00132     size_t infosz;
00133
00134     L4Re::Unique_cap<Dataspace> dirinfofile;
00135     int err = get_ds(".dirinfo", &dirinfofile);
00136     if (err)
00137         return 0;
00138
00139     infosz = dirinfofile->size();
00140     if (infosz <= 0)
00141         return 0;
00142
00143     infoaddr = L4_PAGESIZE;
00144     err = L4Re::Env::env()->rm()->attach(&infoaddr, infosz,
00145                                         Rm::F::Search_addr | Rm::F::R,
00146                                         dirinfofile.get(), 0);
00147     if (err < 0)
00148         return 0;
00149
00150     char *p = reinterpret_cast<char *>(infoaddr) + _current_dir_pos;
00151     char *end = reinterpret_cast<char *>(infoaddr) + infosz;
00152
00153     char *current_dirinfo_entry = p;
00154     while (dest && p < end)
00155     {
00156         // parse lines of dirinfofile
00157         long len = 0;
00158         for (; p < end && *p >= '0' && *p <= '9'; ++p)
00159         {
00160             len *= 10;
00161             len += *p - '0';
00162         }
00163
00164         if (len == 0)
00165             break;
00166
00167         if (p == end)
00168             break;
00169
00170         if (*p != ':')
00171             break;
00172         p++; // skip colon
00173
00174         if (p + len >= end)
00175             break;
00176
00177         unsigned l = len + 1;
00178         if (l > sizeof(dest->d_name))
00179             l = sizeof(dest->d_name);
00180
00181         unsigned n = offsetof(struct dirent64, d_name) + l;
00182         n = (n + sizeof(long) - 1) & ~(sizeof(long) - 1);
00183
00184         if (n > dest_sz)
00185             break;
00186
00187         dest->d_ino = 1;
00188         dest->d_off = 0;
00189         memcpy(dest->d_name, p, l - 1);
00190         dest->d_name[l - 1] = 0;
00191         dest->d_reclen = n;
00192         dest->d_type = DT_UNKNOWN;
00193         ret += n;
00194         dest_sz -= n;
00195
00196         // next entry
00197         dest = reinterpret_cast<struct dirent64 *>
00198             (reinterpret_cast<unsigned long>(dest) + n);
00199
00200         // next infodirfile line
00201         p += len;
00202         while (p < end && *p && (*p == '\n' || *p == '\r'))
00203             p++;
00204
00205         current_dirinfo_entry = p;
00206     }
00207
00208     _current_dir_pos = current_dirinfo_entry - reinterpret_cast<char *>(infoaddr);
00209
00210     if (!ret) // hack since we should only reset this at open times
00211         _current_dir_pos = 0;
00212
00213     L4Re::Env::env()->rm()->detach(infoaddr, 0);
00214
00215     return ret;
00216 }

```

```

00217
00218 int
00219 Env_dir::get_ds(const char *path, L4Re::Unique_cap<L4Re::Dataspace> *ds) noexcept
00220 {
00221     Vfs::Path p(path);
00222     Vfs::Path first = p.strip_first();
00223
00224     if (first.empty())
00225         return -ENOENT;
00226
00227     L4::Cap<L4Re::Namespace>
00228     c = _env->get_cap<L4Re::Namespace>(first.path(), first.length());
00229
00230     if (!c.is_valid())
00231         return -ENOENT;
00232
00233     if (p.empty())
00234     {
00235         *ds = L4Re::Unique_cap<L4Re::Dataspace>(L4::cap_reinterpret_cast<L4Re::Dataspace>(c));
00236         return 0;
00237     }
00238
00239     auto file = L4Re::make_unique_cap<L4Re::Dataspace>(L4Re::virt_cap_alloc);
00240
00241     if (!file.is_valid())
00242         return -ENOMEM;
00243
00244     int err = c->query(p.path(), p.length(), file.get());
00245
00246     if (err < 0)
00247         return -ENOENT;
00248
00249     *ds = cxx::move(file);
00250     return err;
00251 }
00252
00253 int
00254 Env_dir::get_entry(const char *path, int /*flags*/, mode_t /*mode*/,
00255                   Ref_ptr<L4Re::Vfs::File> *f) noexcept
00256 {
00257     if (!*path)
00258     {
00259         *f = cxx::ref_ptr(this);
00260         return 0;
00261     }
00262
00263     L4Re::Unique_cap<Dataspace> file;
00264     int err = get_ds(path, &file);
00265
00266     if (err < 0)
00267         return -ENOENT;
00268
00269     cxx::Ref_ptr<L4Re::Vfs::File> fi = cap_to_vfs_object(file.get(), &err);
00270     if (!fi)
00271         return err;
00272
00273     file.release();
00274     *f = cxx::move(fi);
00275     return 0;
00276 }
00277
00278 int
00279 Env_dir::faccessat(const char *path, int mode, int /*flags*/) noexcept
00280 {
00281     Vfs::Path p(path);
00282     Vfs::Path first = p.strip_first();
00283
00284     if (first.empty())
00285         return -ENOENT;
00286
00287     L4::Cap<L4Re::Namespace>
00288     c = _env->get_cap<L4Re::Namespace>(first.path(), first.length());
00289
00290     if (!c.is_valid())
00291         return -ENOENT;
00292
00293     if (p.empty())
00294     {
00295         if (mode & W_OK)
00296             return -EACCES;
00297
00298         return 0;
00299     }
00300
00301     auto tmpcap = L4Re::make_unique_cap<void>(L4Re::virt_cap_alloc);
00302
00303     if (!tmpcap.is_valid())

```

```

00304     return -ENOMEM;
00305
00306     if (c->query(p.path(), p.length(), tmpcap.get()))
00307         return -ENOENT;
00308
00309     if (mode & W_OK)
00310         return -EACCES;
00311
00312     return 0;
00313 }
00314
00315 bool
00316 Env_dir::check_type(Env::Cap_entry const *e, long protocol) noexcept
00317 {
00318     L4::Cap<L4::Meta> m(e->cap);
00319     return m->supports(protocol).label();
00320 }
00321
00322 int
00323 Env_dir::fstat(struct stat64 *b) const noexcept
00324 {
00325     b->st_dev = 1;
00326     b->st_ino = 1;
00327     b->st_mode = S_IRWXU | S_IFDIR;
00328     b->st_nlink = 0;
00329     b->st_uid = 0;
00330     b->st_gid = 0;
00331     b->st_rdev = 0;
00332     b->st_size = 0;
00333     b->st_blksize = 0;
00334     b->st_blocks = 0;
00335     b->st_atime = 0;
00336     b->st_mtime = 0;
00337     b->st_ctime = 0;
00338     return 0;
00339 }
00340
00341 ssize_t
00342 Env_dir::getdents(char *buf, size_t sz) noexcept
00343 {
00344     struct dirent64 *d = reinterpret_cast<struct dirent64 *>(buf);
00345     ssize_t ret = 0;
00346
00347     while (d
00348         && _current_cap_entry
00349         && _current_cap_entry->flags != ~0UL)
00350     {
00351         unsigned l = strlen(_current_cap_entry->name) + 1;
00352         if (l > sizeof(d->d_name))
00353             l = sizeof(d->d_name);
00354
00355         unsigned n = offsetof(struct dirent64, d_name) + 1;
00356         n = (n + sizeof(long) - 1) & ~(sizeof(long) - 1);
00357
00358         if (n <= sz)
00359         {
00360             d->d_ino = 1;
00361             d->d_off = 0;
00362             memcpy(d->d_name, _current_cap_entry->name, l);
00363             d->d_name[l - 1] = 0;
00364             d->d_reclen = n;
00365             if (check_type(_current_cap_entry, L4Re::Namespace::Protocol))
00366                 d->d_type = DT_DIR;
00367             else if (check_type(_current_cap_entry, L4Re::Dataspace::Protocol))
00368                 d->d_type = DT_REG;
00369             else
00370                 d->d_type = DT_UNKNOWN;
00371             ret += n;
00372             sz -= n;
00373             d = reinterpret_cast<struct dirent64 *>
00374                 (reinterpret_cast<unsigned long*>(d) + n);
00375             _current_cap_entry++;
00376         }
00377         else
00378             return ret;
00379     }
00380
00381     // bit of a hack because we should only (re)set this when opening the dir
00382     if (!ret)
00383         _current_cap_entry = _env->initial_caps();
00384
00385     return ret;
00386 }
00387
00388 }}

```

## 17.243 ro\_file.h

```

00001 /*
00002  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <l4/l4re_vfs/backend>
00011
00012 namespace L4Re { namespace Core {
00013
00014 class Ro_file : public L4Re::Vfs::Be_file_pos
00015 {
00016 private:
00017     L4::Cap<L4Re::Dataspace> _ds;
00018     off64_t _size;
00019     char const *_addr;
00020
00021 public:
00022     explicit Ro_file(L4::Cap<L4Re::Dataspace> ds) noexcept
00023         : Be_file_pos(), _ds(ds), _addr(0)
00024     {
00025         _size = _ds->size();
00026     }
00027
00028     L4::Cap<L4Re::Dataspace> data_space() noexcept override { return _ds; }
00029
00030     int fstat(struct stat64 *buf) const noexcept override;
00031
00032     int ioctl(unsigned long, va_list) noexcept override;
00033
00034     off64_t size() const noexcept override { return _size; }
00035
00036     int get_status_flags() const noexcept override
00037     { return O_RDONLY; }
00038
00039     int set_status_flags(long) noexcept override
00040     { return 0; }
00041
00042     bool check_ready(Ready_type rt) noexcept override
00043     { return rt == Read; }
00044
00045     ~Ro_file() noexcept;
00046
00047 private:
00048     ssize_t read_single(const struct iovec*, off64_t) noexcept;
00049     ssize_t preadv(const struct iovec *, int, off64_t) noexcept override;
00050     ssize_t pwritev(const struct iovec *, int , off64_t) noexcept override;
00051 };
00052
00053
00054
00055
00056
00057
00058
00059
00060
00061
00062
00063
00064 }

```

## 17.244 ro\_file\_impl.h

```

00001 /*
00002  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #include "ro_file.h"
00010
00011 #include <sys/ioctl.h>
00012
00013 #include <l4/re/env>
00014
00015 namespace L4Re { namespace Core {
00016
00017 Ro_file::~Ro_file() noexcept
00018 {
00019     if (_addr)
00020         L4Re::Env::env()->rm()->detach(l4_addr_t(_addr), 0);
00021
00022     L4Re::virt_cap_alloc->release(_ds);
00023 }

```



```

00024
00025 int
00026 Ro_file::fstat(struct stat64 *buf) const noexcept
00027 {
00028     static int fake = 0;
00029
00030     memset(buf, 0, sizeof(*buf));
00031     buf->st_size = _size;
00032     buf->st_mode = S_IFREG | 0644;
00033     buf->st_dev = _ds.cap();
00034     buf->st_ino = ++fake;
00035     buf->st_blksize = L4_PAGESIZE;
00036     buf->st_blocks = 14_round_page(_size);
00037     return 0;
00038 }
00039
00040 ssize_t
00041 Ro_file::read_single(const struct iovec *vec, off64_t pos) noexcept
00042 {
00043     off64_t l = vec->iiov_len;
00044     if (_size - pos < l)
00045         l = _size - pos;
00046
00047     if (l > 0)
00048     {
00049         Vfs_config::memcpy(vec->iiov_base, _addr + pos, l);
00050         return l;
00051     }
00052
00053     return 0;
00054 }
00055
00056 ssize_t
00057 Ro_file::preadv(const struct iovec *vec, int cnt, off64_t offset) noexcept
00058 {
00059     if (!_addr)
00060     {
00061         void const *file = reinterpret_cast<void*>(L4_PAGESIZE);
00062         long err = L4Re::Env::env()->rm()->attach(&file, _size,
00063                                                  Rm::F::Search_addr | Rm::F::R,
00064                                                  _ds, 0);
00065
00066         if (err < 0)
00067             return err;
00068
00069         _addr = static_cast<char const *>(file);
00070     }
00071
00072     ssize_t l = 0;
00073
00074     while (cnt > 0)
00075     {
00076         ssize_t r = read_single(vec, offset);
00077         offset += r;
00078         l += r;
00079
00080         if (static_cast<size_t>(r) < vec->iiov_len)
00081             return l;
00082
00083         ++vec;
00084         --cnt;
00085     }
00086     return l;
00087 }
00088
00089 ssize_t
00090 Ro_file::pwritev(const struct iovec *, int, off64_t) noexcept
00091 {
00092     return -EROFS;
00093 }
00094
00095 int
00096 Ro_file::ioctl(unsigned long v, va_list args) noexcept
00097 {
00098     switch (v)
00099     {
00100     case FIONREAD: // return amount of data still available
00101         int *available = va_arg(args, int *);
00102         *available = _size - pos();
00103         return 0;
00104     };
00105     return -ENOTTY;
00106 }
00107
00108 }

```

## 17.245 vcon\_stream.h

```

00001 /*
00002  * (c) 2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/sys/capability>
00010 #include <l4/sys/vcon>
00011 #include <l4/sys/semaphore>
00012
00013 #include <l4/l4re_vfs/backend>
00014
00015 namespace L4Re { namespace Core {
00016
00017 class Vcon_stream : public L4Re::Vfs::Be_file_stream
00018 {
00019 private:
00020     L4::Cap<L4::Vcon> _s;
00021     L4::Cap<L4::Semaphore> _irq;
00022     unsigned _irq_bound;
00023
00024 public:
00025     explicit Vcon_stream(L4::Cap<L4::Vcon> s) noexcept;
00026
00027     ssize_t readv(const struct iovec*, int iovcnt) noexcept override;
00028     ssize_t writev(const struct iovec*, int iovcnt) noexcept override;
00029     int fstat(struct stat64 *buf) const noexcept override;
00030     int get_status_flags() const noexcept override { return O_RDWR; }
00031     int set_status_flags(long) noexcept override { return 0; }
00032     int ioctl(unsigned long request, va_list args) noexcept override;
00033
00034     ~Vcon_stream() noexcept {}
00035     void operator delete (void *) {}
00036 };
00037
00038 }}
```

## 17.246 vcon\_stream\_impl.h

```

00001 /*
00002  * (c) 2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #include <l4/re/env>
00009 #include <l4/sys/factory>
00010 #include <l4/cxx/minmax>
00011
00012 #include "vcon_stream.h"
00013
00014 #include <limits.h>
00015 #include <termios.h>
00016 #include <unistd.h>
00017 #include <sys/ioctl.h>
00018 #include <sys/ttydefaults.h>
00019
00020 namespace L4Re { namespace Core {
00021 Vcon_stream::Vcon_stream(L4::Cap<L4::Vcon> s) noexcept
00022 : Be_file_stream(),
00023   _s(s), _irq(L4Re::virt_cap_alloc->alloc<L4::Semaphore>()), _irq_bound(false)
00024 {
00025     // [[maybe_unused]] int res =
00026     l4_error(L4Re::Env::env()->factory()->create(_irq));
00027     // (void)res; // handle errors!
00028 }
00029
00030 ssize_t
00031 Vcon_stream::readv(const struct iovec *iovec, int iovcnt) noexcept
00032 {
00033     if (iovcnt < 0)
00034         return -EINVAL;
00035
00036     if (!_irq_bound)
00037     {
00038         bool was_bound = __atomic_exchange_n(&_irq_bound, true, __ATOMIC_SEQ_CST);
00039         if (!was_bound)
```

```

00040         if (l4_error(_s->bind(0, _irq)) < 0)
00041             return -EIO;
00042     }
00043
00044     ssize_t bytes = 0;
00045     for (; iovcnt > 0; --iovcnt, ++iovec)
00046     {
00047         size_t len = cxx::min<size_t>(iovec->iov_len, SSIZE_MAX - bytes);
00048         if (len == 0)
00049             continue;
00050
00051         char *buf = static_cast<char *>(iovec->iov_base);
00052
00053         while (1)
00054         {
00055             size_t l = cxx::min<size_t>(L4_VCON_READ_SIZE, len);
00056             int ret = _s->read(buf, l);
00057
00058             if (ret > static_cast<int>(1))
00059                 ret = 1;
00060
00061             if (ret < 0)
00062                 return ret;
00063             else if (ret == 0)
00064             {
00065                 if (bytes)
00066                     return bytes;
00067
00068                 ret = _s->read(buf, l);
00069                 if (ret < 0)
00070                     return ret;
00071                 else if (ret == 0)
00072                 {
00073                     _irq->down();
00074                     continue;
00075                 }
00076             }
00077
00078             bytes += ret;
00079             len -= ret;
00080             buf += ret;
00081
00082             if (len == 0)
00083                 break;
00084         }
00085     }
00086
00087     return bytes;
00088 }
00089
00090 ssize_t
00091 Vcon_stream::writev(const struct iovec *iovec, int iovcnt) noexcept
00092 {
00093     l4_msg_regs_t store;
00094     l4_msg_regs_t *mr = l4_utcb_mr();
00095
00096     if (iovcnt < 0)
00097         return -EINVAL;
00098
00099     Vfs_config::memcpy(&store, mr, sizeof(store));
00100
00101     ssize_t written = 0;
00102     while (iovcnt)
00103     {
00104         size_t sl = cxx::min<size_t>(iovec->iov_len, SSIZE_MAX - written);
00105         char const *b = static_cast<char const *>(iovec->iov_base);
00106
00107         for (; sl > L4_VCON_WRITE_SIZE;
00108             ; sl -= L4_VCON_WRITE_SIZE, b += L4_VCON_WRITE_SIZE,
00109             written += L4_VCON_WRITE_SIZE)
00110             _s->send(b, L4_VCON_WRITE_SIZE);
00111
00112         _s->send(b, sl);
00113
00114         written += sl;
00115
00116         ++iovec;
00117         --iovcnt;
00118     }
00119     Vfs_config::memcpy(mr, &store, sizeof(store));
00120     return written;
00121 }
00122
00123 int
00124 Vcon_stream::fstat(struct stat64 *buf) const noexcept
00125 {
00126     buf->st_size = 0;

```

```

00127     buf->st_mode = 0666;
00128     buf->st_dev = _s.cap();
00129     buf->st_ino = 0;
00130     return 0;
00131 }
00132
00133 int
00134 Vcon_stream::ioctl(unsigned long request, va_list args) noexcept
00135 {
00136     switch (request) {
00137         case TCGETS:
00138             {
00139                 //vt100_tcgetattr(term, (struct termios *)argp);
00140
00141                 struct termios *t = va_arg(args, struct termios *);
00142
00143                 l4_vcon_attr_t l4a;
00144                 if (!l4_error(_s->get_attr(&l4a)))
00145                 {
00146                     t->c_iflag = l4a.i_flags;
00147                     t->c_oflag = l4a.o_flags; // output flags
00148                     t->c_cflag = 0; // control flags
00149                     t->c_lflag = l4a.l_flags; // local flags
00150                 }
00151                 else
00152                     t->c_iflag = t->c_oflag = t->c_cflag = t->c_lflag = 0;
00153             }
00154             //if 0
00155             //t->c_lflag |= ECHO; // if term->echo
00156             //t->c_lflag |= ICANON; // if term->term_mode == VT100MODE_COOKED
00157         #endif
00158
00159         t->c_cc[VEOF] = CEOF;
00160         t->c_cc[VEOL] = _POSIX_VDISABLE;
00161         t->c_cc[VEOL2] = _POSIX_VDISABLE;
00162         t->c_cc[VERASE] = CERASE;
00163         t->c_cc[VWERASE] = CWERASE;
00164         t->c_cc[VKILL] = CKILL;
00165         t->c_cc[VREPRINT] = CREPRINT;
00166         t->c_cc[VINTR] = CINTR;
00167         t->c_cc[VQUIT] = _POSIX_VDISABLE;
00168         t->c_cc[VSUSP] = CSUSP;
00169         t->c_cc[VSTART] = CSTART;
00170         t->c_cc[VSTOP] = CSTOP;
00171         t->c_cc[VLNEXT] = CLNEXT;
00172         t->c_cc[VDISCARD] = CDISCARD;
00173         t->c_cc[VMIN] = CMIN;
00174         t->c_cc[VTIME] = 0;
00175     }
00176
00177     return 0;
00178
00179     case TCSETS:
00180     case TCSETSW:
00181     case TCSETSF:
00182     {
00183         //vt100_tcsetattr(term, (struct termios *)argp);
00184         struct termios const *t = va_arg(args, struct termios const *);
00185
00186         // XXX: well, we're cheating, get this from the other side!
00187
00188         l4_vcon_attr_t l4a;
00189         l4a.i_flags = t->c_iflag;
00190         l4a.o_flags = t->c_oflag; // output flags
00191         l4a.l_flags = t->c_lflag; // local flags
00192         _s->set_attr(&l4a);
00193     }
00194     return 0;
00195
00196     default:
00197         break;
00198 };
00199 return -ENOTTY;
00200 }
00201
00202 }}

```

## 17.247 vfs\_impl.h

```

00001 /*
00002  * (c) 2008-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  * Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  * Björn Döbel <doebel@os.inf.tu-dresden.de>

```

```

00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #include "fd_store.h"
00011 #include "vcon_stream.h"
00012 #include "ns_fs.h"
00013
00014 #include <l4/bid_config.h>
00015 #include <l4/re/env>
00016 #include <l4/re/rm>
00017 #include <l4/re/dataspace>
00018 #include <l4/sys/assert.h>
00019 #include <l4/cxx/hlist>
00020 #include <l4/cxx/pair>
00021 #include <l4/cxx/std_alloc>
00022
00023 #include <l4/l4re_vfs/backend>
00024 #include <l4/re/shared_cap>
00025
00026 #include <unistd.h>
00027 #include <stdarg.h>
00028 #include <errno.h>
00029 #include <sys/uio.h>
00030
00031 #if 0
00032 #include <l4/sys/kdebug.h>
00033 static int debug_mmap = 1;
00034 #define DEBUG_LOG(level, dbg...) do { if (level) dbg } while (0)
00035 #else
00036 #define DEBUG_LOG(level, dbg...) do { } while (0)
00037 #endif
00038
00044 #define USE_BIG_ANON_DS
00045
00046 using L4Re::Rm;
00047
00048 namespace {
00049
00050 using cxx::Ref_ptr;
00051
00052 class Fd_store : public L4Re::Core::Fd_store
00053 {
00054 public:
00055     Fd_store() noexcept;
00056 };
00057
00058 // for internal Vcon_streams we want to have a placement new operator, so
00059 // inherit and add one
00060 class Std_stream : public L4Re::Core::Vcon_stream
00061 {
00062 public:
00063     Std_stream(L4::Cap<L4::Vcon> c) : L4Re::Core::Vcon_stream(c) {}
00064 };
00065
00066 Fd_store::Fd_store() noexcept
00067 {
00068     // use this strange way to prevent deletion of the stdio object
00069     // this depends on Fd_store to being a singleton !!!
00070     static char m[sizeof(Std_stream)] __attribute__((aligned(sizeof(long))));
00071     Std_stream *s = new (m) Std_stream(L4Re::Env::env()->log());
00072     // make sure that we never delete the static io stream thing
00073     s->add_ref();
00074     set(0, cxx::ref_ptr(s)); // stdin
00075     set(1, cxx::ref_ptr(s)); // stdout
00076     set(2, cxx::ref_ptr(s)); // stderr
00077 }
00078
00079 class Root_mount_tree : public L4Re::Vfs::Mount_tree
00080 {
00081 public:
00082     Root_mount_tree() : L4Re::Vfs::Mount_tree(0) {}
00083     void operator delete (void *) {}
00084 };
00085
00086 class Vfs : public L4Re::Vfs::Ops
00087 {
00088 private:
00089     bool _early_oom;
00090
00091 public:
00092     Vfs()
00093     : _early_oom(true), _root_mount(), _root(L4Re::Env::env())
00094     {
00095         _root_mount.add_ref();
00096         _root.add_ref();
00097     }

```

```

00097     _root_mount.mount(cxx::ref_ptr(&_root));
00098     _cwd = cxx::ref_ptr(&_root);
00099
00100 #if 0
00101     Ref_ptr<L4Re::Vfs::File> rom;
00102     _root.openat("rom", 0, 0, &rom);
00103
00104     _root_mount.create_tree("lib/foo", rom);
00105
00106     _root.openat("lib", 0, 0, &_cwd);
00107
00108 #endif
00109 }
00110
00111 int alloc_fd(Ref_ptr<L4Re::Vfs::File> const &f) noexcept override;
00112 Ref_ptr<L4Re::Vfs::File> free_fd(int fd) noexcept override;
00113 Ref_ptr<L4Re::Vfs::File> get_root() noexcept override;
00114 Ref_ptr<L4Re::Vfs::File> get_cwd() noexcept override;
00115 void set_cwd(Ref_ptr<L4Re::Vfs::File> const &dir) noexcept override;
00116 Ref_ptr<L4Re::Vfs::File> get_file(int fd) noexcept override;
00117 cxx::Pair<Ref_ptr<L4Re::Vfs::File>, int>
00118     set_fd(int fd, Ref_ptr<L4Re::Vfs::File> const &f = Ref_ptr<>::Nil) noexcept
00119         override;
00120
00121 int mmap2(void *start, size_t len, int prot, int flags, int fd,
00122           off_t offset, void **ptr) noexcept override;
00123
00124 int munmap(void *start, size_t len) noexcept override;
00125 int mremap(void *old, size_t old_sz, size_t new_sz, int flags,
00126            void **new_addr) noexcept override;
00127 int mprotect(const void *a, size_t sz, int prot) noexcept override;
00128 int msync(void *addr, size_t len, int flags) noexcept override;
00129 int madvise(void *addr, size_t len, int advice) noexcept override;
00130
00131 int register_file_system(L4Re::Vfs::File_system *f) noexcept override;
00132 int unregister_file_system(L4Re::Vfs::File_system *f) noexcept override;
00133 L4Re::Vfs::File_system *get_file_system(char const *fstype) noexcept override;
00134 L4Re::Vfs::File_system_list file_system_list() noexcept override;
00135
00136 int register_file_factory(cxx::Ref_ptr<L4Re::Vfs::File_factory> f) noexcept override;
00137 int unregister_file_factory(cxx::Ref_ptr<L4Re::Vfs::File_factory> f) noexcept override;
00138 Ref_ptr<L4Re::Vfs::File_factory> get_file_factory(int proto) noexcept override;
00139 Ref_ptr<L4Re::Vfs::File_factory> get_file_factory(char const *proto_name) noexcept override;
00140 int mount(char const *path, cxx::Ref_ptr<L4Re::Vfs::File> const &dir) noexcept override;
00141
00142 void operator delete (void *) {}
00143
00144 void *malloc(size_t size) noexcept override { return Vfs_config::malloc(size); }
00145 void free(void *m) noexcept override { Vfs_config::free(m); }
00146
00147 private:
00148     Root_mount_tree _root_mount;
00149     L4Re::Core::Env_dir _root;
00150     Ref_ptr<L4Re::Vfs::File> _cwd;
00151     Fd_store fds;
00152
00153     L4Re::Vfs::File_system *_fs_registry;
00154
00155     struct File_factory_item : cxx::H_list_item_t<File_factory_item>
00156     {
00157         cxx::Ref_ptr<L4Re::Vfs::File_factory> f;
00158         explicit File_factory_item(cxx::Ref_ptr<L4Re::Vfs::File_factory> const &f)
00159             : f(f) {};
00160
00161         File_factory_item() = default;
00162         File_factory_item(File_factory_item const &) = delete;
00163         File_factory_item &operator = (File_factory_item const &) = delete;
00164     };
00165
00166     cxx::H_list_t<File_factory_item> _file_factories;
00167
00168     l4_addr_t _anon_offset;
00169     L4Re::Shared_cap<L4Re::Dataspace> _anon_ds;
00170
00171     int alloc_ds(unsigned long size, L4Re::Shared_cap<L4Re::Dataspace> *ds);
00172     int alloc_anon_mem(l4_umword_t size, L4Re::Shared_cap<L4Re::Dataspace> *ds,
00173                       l4_addr_t *offset);
00174
00175     void align_mmap_start_and_length(void **start, size_t *length);
00176     int munmap_regions(void *start, size_t len);
00177
00178     L4Re::Vfs::File_system *find_fs_from_type(char const *fstype) noexcept;
00179 };
00180
00181 static inline bool strequal(char const *a, char const *b)
00182 {
00183     for (;*a && *a == *b; ++a, ++b)

```

```

00184     ;
00185     return *a == *b;
00186 }
00187
00188 int
00189 Vfs::register_file_system(L4Re::Vfs::File_system *f) noexcept
00190 {
00191     using L4Re::Vfs::File_system;
00192
00193     if (!f)
00194         return -EINVAL;
00195
00196     for (File_system *c = _fs_registry; c; c = c->next())
00197         if (strequal(c->type(), f->type()))
00198             return -EEXIST;
00199
00200     f->next(_fs_registry);
00201     _fs_registry = f;
00202
00203     return 0;
00204 }
00205
00206 int
00207 Vfs::unregister_file_system(L4Re::Vfs::File_system *f) noexcept
00208 {
00209     using L4Re::Vfs::File_system;
00210
00211     if (!f)
00212         return -EINVAL;
00213
00214     File_system **p = &_fs_registry;
00215
00216     for (; *p; p = &(*p)->next())
00217         if (*p == f)
00218         {
00219             *p = f->next();
00220             f->next() = 0;
00221             return 0;
00222         }
00223
00224     return -ENOENT;
00225 }
00226
00227 L4Re::Vfs::File_system *
00228 Vfs::find_fs_from_type(char const *fstype) noexcept
00229 {
00230     L4Re::Vfs::File_system_list fsl(_fs_registry);
00231     for (L4Re::Vfs::File_system_list::Iterator c = fsl.begin();
00232          c != fsl.end(); ++c)
00233         if (strequal(c->type(), fstype))
00234             return *c;
00235     return 0;
00236 }
00237
00238 L4Re::Vfs::File_system_list
00239 Vfs::file_system_list() noexcept
00240 {
00241     return L4Re::Vfs::File_system_list(_fs_registry);
00242 }
00243
00244 L4Re::Vfs::File_system *
00245 Vfs::get_file_system(char const *fstype) noexcept
00246 {
00247     L4Re::Vfs::File_system *fs;
00248     if ((fs = find_fs_from_type(fstype)))
00249         return fs;
00250
00251     // Try to load a file system module dynamically
00252     int res = Vfs_config::load_module(fstype);
00253     if (res < 0)
00254         return 0;
00255
00256     // Try again
00257     return find_fs_from_type(fstype);
00258 }
00259
00260 int
00261 Vfs::register_file_factory(cxx::Ref_ptr<L4Re::Vfs::File_factory> f) noexcept
00262 {
00263     if (!f)
00264         return -EINVAL;
00265
00266     void *x = this->m_malloc(sizeof(File_factory_item));
00267     if (!x)
00268         return -ENOMEM;
00269
00270     auto ff = new (x, cxx::Nothrow()) File_factory_item(f);

```

```

00271  _file_factories.push_front(ff);
00272  return 0;
00273 }
00274
00275 int
00276 Vfs::unregister_file_factory(cxx::Ref_ptr<L4Re::Vfs::File_factory> f) noexcept
00277 {
00278     for (auto p: _file_factories)
00279     {
00280         if (p->f == f)
00281         {
00282             _file_factories.remove(p);
00283             p->~File_factory_item();
00284             this->free(p);
00285             return 0;
00286         }
00287     }
00288     return -ENOENT;
00289 }
00290
00291 Ref_ptr<L4Re::Vfs::File_factory>
00292 Vfs::get_file_factory(int proto) noexcept
00293 {
00294     for (auto p: _file_factories)
00295         if (p->f->proto() == proto)
00296             return p->f;
00297
00298     return Ref_ptr<L4Re::Vfs::File_factory>();
00299 }
00300
00301 Ref_ptr<L4Re::Vfs::File_factory>
00302 Vfs::get_file_factory(char const *proto_name) noexcept
00303 {
00304     for (auto p: _file_factories)
00305     {
00306         auto n = p->f->proto_name();
00307         if (n)
00308         {
00309             char const *a = n;
00310             char const *b = proto_name;
00311             for (; *a && *b && *a == *b; ++a, ++b)
00312                 ;
00313
00314             if ((*a == 0) && (*b == 0))
00315                 return p->f;
00316         }
00317     }
00318
00319     return Ref_ptr<L4Re::Vfs::File_factory>();
00320 }
00321
00322 int
00323 Vfs::alloc_fd(Ref_ptr<L4Re::Vfs::File> const &f) noexcept
00324 {
00325     int fd = fds.alloc();
00326     if (fd < 0)
00327         return -EMFILE;
00328
00329     if (f)
00330         fds.set(fd, f);
00331
00332     return fd;
00333 }
00334
00335 Ref_ptr<L4Re::Vfs::File>
00336 Vfs::free_fd(int fd) noexcept
00337 {
00338     Ref_ptr<L4Re::Vfs::File> f = fds.get(fd);
00339
00340     if (!f)
00341         return Ref_ptr<>::Nil;
00342
00343     fds.free(fd);
00344     return f;
00345 }
00346
00347
00348 Ref_ptr<L4Re::Vfs::File>
00349 Vfs::get_root() noexcept
00350 {
00351     return cxx::ref_ptr(&_root);
00352 }
00353
00354 Ref_ptr<L4Re::Vfs::File>
00355 Vfs::get_cwd() noexcept
00356 {
00357     return _cwd;

```



```

00358 }
00359
00360 void
00361 Vfs::set_cwd(Ref_ptr<L4Re::Vfs::File> const &dir) noexcept
00362 {
00363     // FIXME: check for is dir
00364     if (dir)
00365         _cwd = dir;
00366 }
00367
00368 Ref_ptr<L4Re::Vfs::File>
00369 Vfs::get_file(int fd) noexcept
00370 {
00371     return fds.get(fd);
00372 }
00373
00374 cxx::Pair<Ref_ptr<L4Re::Vfs::File>, int>
00375 Vfs::set_fd(int fd, Ref_ptr<L4Re::Vfs::File> const &f) noexcept
00376 {
00377     if (!fds.check_fd(fd))
00378         return cxx::pair(Ref_ptr<L4Re::Vfs::File>(Ref_ptr<>::Nil), EBADF);
00379
00380     Ref_ptr<L4Re::Vfs::File> old = fds.get(fd);
00381     fds.set(fd, f);
00382     return cxx::pair(old, 0);
00383 }
00384
00385 #define GET_FILE_DBG(fd, err) \
00386     Ref_ptr<L4Re::Vfs::File> fi = fds.get(fd); \
00387     if (!fi) \
00388     { \
00389         return -err; \
00390     }
00391
00392 #define GET_FILE(fd, err) \
00393     Ref_ptr<L4Re::Vfs::File> fi = fds.get(fd); \
00394     if (!fi) \
00395         return -err;
00396
00397 void
00398 Vfs::align_mmap_start_and_length(void **start, size_t *length)
00399 {
00400     l4_addr_t const s = reinterpret_cast<l4_addr_t>(*start);
00401     size_t const o = s & (L4_PAGESIZE - 1);
00402
00403     *start = reinterpret_cast<void*>(l4_trunc_page(s));
00404     *length = l4_round_page(*length + o);
00405 }
00406
00407 int
00408 Vfs::munmap_regions(void *start, size_t len)
00409 {
00410     using namespace L4;
00411     using namespace L4Re;
00412
00413     int err;
00414     Cap<Dataspace> ds;
00415     Cap<Rm> r = Env::env()->rm();
00416
00417     if (l4_addr_t(start) & (L4_PAGESIZE - 1))
00418         return -EINVAL;
00419
00420     align_mmap_start_and_length(&start, &len);
00421
00422     while (1)
00423     {
00424         DEBUG_LOG(debug_mmap, {
00425             outstring("DETACH: start = 0x");
00426             outhex32(l4_addr_t(start));
00427             outstring(" len = 0x");
00428             outhex32(len);
00429             outstring("\n");
00430         });
00431         err = r->detach(l4_addr_t(start), len, &ds, This_task);
00432         if (err < 0)
00433             return err;
00434
00435         switch (err & Rm::Detach_result_mask)
00436         {
00437             case Rm::Split_ds:
00438                 if (ds.is_valid())
00439                     L4Re::virt_cap_alloc->take(ds);
00440                 return 0;
00441             case Rm::Detached_ds:
00442                 if (ds.is_valid())
00443                     L4Re::virt_cap_alloc->release(ds);
00444

```

```

00445         break;
00446     default:
00447         break;
00448     }
00449
00450     if (!(err & Rm::Detach_again))
00451         return 0;
00452 }
00453 }
00454
00455 int
00456 Vfs::munmap(void *start, size_t len) L4_NOTHROW
00457 {
00458     using namespace L4;
00459     using namespace L4Re;
00460
00461     int err = 0;
00462     Cap<Rm> r = Env::env()->rm();
00463
00464     // Fields for obtaining a list of areas for the calling process
00465     long area_cnt = -1;           // No. of areas in this process
00466     Rm::Area const *area_array;
00467     bool matches_area = false;    // true if unmap parameters match an area
00468
00469     // First check if there are any areas matching the munmap request. Those
00470     // might have been created by an mmap call using PROT_NONE as protection
00471     // modifier.
00472
00473     area_cnt = r->get_areas((l4_addr_t) start, &area_array);
00474
00475     // It is enough to check for the very first entry, since get_areas will
00476     // only return areas with a starting address equal or greater to <start>.
00477     // However, we intend to unmap at most the area starting exactly at
00478     // <start>.
00479     if (area_cnt > 0)
00480     {
00481         size_t area_size = area_array[0].end - area_array[0].start + 1;
00482
00483         // Only free the area if the munmap parameters describe it exactly.
00484         if (area_array[0].start == (l4_addr_t) start && area_size == len)
00485         {
00486             r->free_area((l4_addr_t) start);
00487             matches_area = true;
00488         }
00489     }
00490
00491     // After clearing possible area reservations from PROT_NONE mappings, clear
00492     // any regions in the address range specified. Note that errors shall be
00493     // suppressed if an area was freed but no regions were found.
00494     err = munmap_regions(start, len);
00495     if (err == -ENOENT && matches_area)
00496         return 0;
00497
00498     return err;
00499 }
00500
00501 int
00502 Vfs::alloc_ds(unsigned long size, L4Re::Shared_cap<L4Re::Dataspace> *ds)
00503 {
00504     *ds = L4Re::make_shared_cap<L4Re::Dataspace>(L4Re::virt_cap_alloc);
00505
00506     if (!ds->is_valid())
00507         return -ENOMEM;
00508
00509     int err;
00510     if ((err = Vfs_config::allocator()->alloc(size, ds->get())) < 0)
00511         return err;
00512
00513     DEBUG_LOG(debug_mmap, {
00514         outstring("ANON DS ALLOCATED: size=");
00515         outhex32(size);
00516         outstring(" cap = 0x");
00517         outhex32(ds->cap());
00518         outstring("\n");
00519     });
00520
00521     return 0;
00522 }
00523
00524 int
00525 Vfs::alloc_anon_mem(l4_umword_t size, L4Re::Shared_cap<L4Re::Dataspace> *ds,
00526                    l4_addr_t *offset)
00527 {
00528     #if !defined(CONFIG_MMU)
00529         // Small values for !MMU systems. These platforms do not have much memory
00530         // typically and the memory must be instantly allocated.
00531         enum

```

```

00532 {
00533     ANON_MEM_DS_POOL_SIZE = 256UL « 10, // size of a pool dataspace used for anon memory
00534     ANON_MEM_MAX_SIZE     = 32UL « 10, // chunk size that will be allocate a dataspace
00535 };
00536 #elif defined(USE_BIG_ANON_DS)
00537 enum
00538 {
00539     ANON_MEM_DS_POOL_SIZE = 256UL « 20, // size of a pool dataspace used for anon memory
00540     ANON_MEM_MAX_SIZE     = 32UL « 20, // chunk size that will be allocate a dataspace
00541 };
00542 #else
00543 enum
00544 {
00545     ANON_MEM_DS_POOL_SIZE = 256UL « 20, // size of a pool dataspace used for anon memory
00546     ANON_MEM_MAX_SIZE     = 0UL « 20,  // chunk size that will be allocate a dataspace
00547 };
00548 #endif
00549
00550 if (size >= ANON_MEM_MAX_SIZE)
00551 {
00552     int err;
00553     if ((err = alloc_ds(size, ds)) < 0)
00554         return err;
00555
00556     *offset = 0;
00557
00558     if (!_early_oom)
00559         return err;
00560
00561     return (*ds)->allocate(0, size);
00562 }
00563
00564 if (!_anon_ds.is_valid() || _anon_offset + size >= ANON_MEM_DS_POOL_SIZE)
00565 {
00566     int err;
00567     if ((err = alloc_ds(ANON_MEM_DS_POOL_SIZE, ds)) < 0)
00568         return err;
00569
00570     _anon_offset = 0;
00571     _anon_ds = *ds;
00572 }
00573 else
00574     *ds = _anon_ds;
00575
00576 if (_early_oom)
00577 {
00578     if (int err = (*ds)->allocate(_anon_offset, size))
00579         return err;
00580 }
00581
00582 *offset = _anon_offset;
00583 _anon_offset += size;
00584 return 0;
00585 }
00586
00587 int
00588 Vfs::mmap2(void *start, size_t len, int prot, int flags, int fd, off_t page4k_offset,
00589            void **resptr) L4_NOTHROW
00590 {
00591     DEBUG_LOG(debug_mmap, {
00592         outstring("MMAP params: ");
00593         outstring("start = 0x");
00594         outhex32(l4_addr_t(start));
00595         outstring(", len = 0x");
00596         outhex32(len);
00597         outstring(", prot = 0x");
00598         outhex32(prot);
00599         outstring(", flags = 0x");
00600         outhex32(flags);
00601         outstring(", offset = 0x");
00602         outhex32(page4k_offset);
00603         outstring("\n");
00604     });
00605
00606     using namespace L4Re;
00607     off64_t offset = l4_trunc_page(page4k_offset « 12);
00608
00609     if (flags & MAP_FIXED)
00610         if (l4_addr_t(start) & (L4_PAGESIZE - 1))
00611             return -EINVAL;
00612
00613     align_mmap_start_and_length(&start, &len);
00614
00615     // special code to just reserve an area of the virtual address space
00616     // Same behavior should be exposed when mapping with PROT_NONE. Mind that
00617     // PROT_NONE can only be specified exclusively, since it is defined to 0x0.
00618     if ((flags & 0x1000000) || (prot == PROT_NONE))

```

```

00619     {
00620         int err;
00621         L4::Cap<Rm> r = Env::env()->rm();
00622         L4_addr_t area = reinterpret_cast<L4_addr_t>(start);
00623         err = r->reserve_area(&area, len, L4Re::Rm::F::Search_addr);
00624         if (err < 0)
00625             return err;
00626
00627         *resp_ptr = reinterpret_cast<void*>(area);
00628
00629         DEBUG_LOG(debug_mmap, {
00630             outstring(" MMAP reserved area: 0x");
00631             outhex32(area);
00632             outstring(" length= 0x");
00633             outhex32(len);
00634             outstring("\n");
00635         });
00636
00637         return 0;
00638     }
00639
00640     L4Re::Shared_cap<L4Re::Dataspace> ds;
00641     L4_addr_t anon_offset = 0;
00642     L4Re::Rm::Flags rm_flags(0);
00643
00644     if (flags & (MAP_ANONYMOUS | MAP_PRIVATE))
00645     {
00646         rm_flags |= L4Re::Rm::F::Detach_free;
00647
00648         int err = alloc_anon_mem(len, &ds, &anon_offset);
00649         if (err)
00650             return err;
00651
00652         DEBUG_LOG(debug_mmap, {
00653             outstring(" USE ANON MEM: 0x");
00654             outhex32(ds.cap());
00655             outstring(" offs = 0x");
00656             outhex32(anon_offset);
00657             outstring("\n");
00658         });
00659     }
00660
00661     char const *region_name = "[unknown]";
00662     L4_addr_t file_offset = 0;
00663     if (!(flags & MAP_ANONYMOUS))
00664     {
00665         Ref_ptr<L4Re::Vfs::File> fi = fds.get(fd);
00666         if (!fi)
00667             return -EBADF;
00668
00669         region_name = fi->path();
00670
00671         L4::Cap<L4Re::Dataspace> fds = fi->data_space();
00672
00673         if (!fds.is_valid())
00674             return -EINVAL;
00675
00676         if (len + offset > L4_round_page(fds->size()))
00677             return -EINVAL;
00678
00679         if (flags & MAP_PRIVATE)
00680         {
00681             DEBUG_LOG(debug_mmap, outstring("COW\n"));
00682             int err = ds->copy_in(anon_offset, fds, offset, len);
00683             file_offset = offset;
00684             if (err == -L4_EINVAL)
00685             {
00686                 L4::Cap<Rm> r = Env::env()->rm();
00687                 Rm::Unique_region<char*> src;
00688                 Rm::Unique_region<char*> dst;
00689                 err = r->attach(&src, len,
00690                             L4Re::Rm::F::Search_addr | L4Re::Rm::F::R,
00691                             fds, offset);
00692                 if (err < 0)
00693                     return err;
00694
00695                 err = r->attach(&dst, len,
00696                             L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00697                             ds.get(), anon_offset);
00698                 if (err < 0)
00699                     return err;
00700
00701                 memcpy(dst.get(), src.get(), len);
00702
00703                 region_name = "[mmap-private]";
00704                 file_offset = (unsigned long)dst.get();
00705             }

```

```

00706         else if (err)
00707             return err;
00708
00709         offset = anon_offset;
00710     }
00711     else
00712     {
00713         L4Re::virt_cap_alloc->take(fds);
00714         ds = L4Re::Shared_cap<L4Re::Dataspace>(fds, L4Re::virt_cap_alloc);
00715     }
00716 }
00717 else
00718 {
00719     offset = anon_offset;
00720     region_name = "[anon]";
00721     file_offset = offset;
00722 }
00723
00724
00725 if (!(flags & MAP_FIXED) && start == 0)
00726     start = reinterpret_cast<void*>(L4_PAGESIZE);
00727
00728 char *data = static_cast<char *>(start);
00729 L4::Cap<Rm> r = Env::env()->rm();
00730 l4_addr_t overmap_area = L4_INVALID_ADDR;
00731
00732 int err;
00733 if (flags & MAP_FIXED)
00734 {
00735     overmap_area = l4_addr_t(start);
00736
00737     err = r->reserve_area(&overmap_area, len);
00738     if (err < 0)
00739         overmap_area = L4_INVALID_ADDR;
00740
00741     rm_flags |= Rm::F::In_area;
00742
00743     // Make sure to remove old mappings residing at the respective address
00744     // range. If none exists, we are fine as well, allowing us to ignore
00745     // ENOENT here.
00746     err = munmap_regions(start, len);
00747     if (err && err != -ENOENT)
00748         return err;
00749 }
00750
00751 if (!(flags & MAP_FIXED))
00752     rm_flags |= Rm::F::Search_addr;
00753 if (prot & PROT_READ)
00754     rm_flags |= Rm::F::R;
00755 if (prot & PROT_WRITE)
00756     rm_flags |= Rm::F::W;
00757 if (prot & PROT_EXEC)
00758     rm_flags |= Rm::F::X;
00759
00760 err = r->attach(&data, len, rm_flags,
00761               L4::Ipc::make_cap(ds.get(), (prot & PROT_WRITE)
00762                                ? L4_CAP_FPAGE_RW
00763                                : L4_CAP_FPAGE_RO),
00764               offset, L4_PAGESHIFT, L4::Cap<L4::Task>::Invalid,
00765               region_name, file_offset);
00766
00767 DEBUG_LOG(debug_mmap, {
00768     outstring("  MAPPED: 0x");
00769     outhex32(ds.cap());
00770     outstring("  addr: 0x");
00771     outhex32(l4_addr_t(data));
00772     outstring("  bytes: 0x");
00773     outhex32(len);
00774     outstring("  offset: 0x");
00775     outhex32(offset);
00776     outstring("  err = ");
00777     outdec(err);
00778     outstring("\n");
00779 });
00780
00781
00782 if (overmap_area != L4_INVALID_ADDR)
00783     r->free_area(overmap_area);
00784
00785 if (err < 0)
00786     return err;
00787
00788 l4_assert (!(start && !data));
00789
00790 // release ownership of the attached DS
00791 ds.release();
00792 *resptr = data;

```

```

00793
00794     return 0;
00795 }
00796
00797 namespace {
00798     class Auto_area
00799     {
00800     public:
00801         L4::Cap<L4Re::Rm> r;
00802         l4_addr_t a;
00803
00804         explicit Auto_area(L4::Cap<L4Re::Rm> r, l4_addr_t a = L4_INVALID_ADDR)
00805             : r(r), a(a) {}
00806
00807         int reserve(l4_addr_t _a, l4_size_t sz, L4Re::Rm::Flags flags)
00808         {
00809             free();
00810             a = _a;
00811             int e = r->reserve_area(&a, sz, flags);
00812             if (e)
00813                 a = L4_INVALID_ADDR;
00814             return e;
00815         }
00816
00817         void free()
00818         {
00819             if (is_valid())
00820             {
00821                 r->free_area(a);
00822                 a = L4_INVALID_ADDR;
00823             }
00824         }
00825
00826         bool is_valid() const { return a != L4_INVALID_ADDR; }
00827
00828         ~Auto_area() { free(); }
00829     };
00830 }
00831
00832 int
00833 Vfs::mremap(void *old_addr, size_t old_size, size_t new_size, int flags,
00834             void **new_addr) L4_NOTHROW
00835 {
00836     using namespace L4Re;
00837
00838     DEBUG_LOG(debug_mmap, {
00839         outstring("Mremap: addr = 0x");
00840         outhex32((l4_umword_t)old_addr);
00841         outstring(" old_size = 0x");
00842         outhex32(old_size);
00843         outstring(" new_size = 0x");
00844         outhex32(new_size);
00845         outstring("\n");
00846     });
00847
00848     if (flags & MREMAP_FIXED && !(flags & MREMAP_MAYMOVE))
00849         return -EINVAL;
00850
00851     l4_addr_t oa = l4_trunc_page(reinterpret_cast<l4_addr_t>(old_addr));
00852     if (oa != reinterpret_cast<l4_addr_t>(old_addr))
00853         return -EINVAL;
00854
00855     bool const fixed = flags & MREMAP_FIXED;
00856     bool const maymove = flags & MREMAP_MAYMOVE;
00857
00858     L4::Cap<Rm> r = Env::env()->rm();
00859
00860     // sanitize input parameters to multiples of pages
00861     old_size = l4_round_page(old_size);
00862     new_size = l4_round_page(new_size);
00863
00864     if (!fixed)
00865     {
00866         if (new_size < old_size)
00867         {
00868             *new_addr = old_addr;
00869             return munmap(reinterpret_cast<void*>(oa + new_size),
00870                           old_size - new_size);
00871         }
00872
00873         if (new_size == old_size)
00874         {
00875             *new_addr = old_addr;
00876             return 0;
00877         }
00878     }
00879

```

```

00880 Auto_area old_area(r);
00881 int err = old_area.reserve(oa, old_size, L4Re::Rm::Flags(0));
00882 if (err < 0)
00883     return -EINVAL;
00884
00885 l4_addr_t pad_addr;
00886 Auto_area new_area(r);
00887 if (fixed)
00888 {
00889     l4_addr_t na = l4_trunc_page(reinterpret_cast<l4_addr_t>(*new_addr));
00890     if (na != reinterpret_cast<l4_addr_t>(*new_addr))
00891         return -EINVAL;
00892
00893     // check if the current virtual memory area can be expanded
00894     int err = new_area.reserve(na, new_size, L4Re::Rm::Flags(0));
00895     if (err < 0)
00896         return err;
00897
00898     pad_addr = na;
00899     // unmap all stuff and remap ours ....
00900 }
00901 else
00902 {
00903     l4_addr_t ta = oa + old_size;
00904     unsigned long ts = new_size - old_size;
00905     // check if the current virtual memory area can be expanded
00906     long err = new_area.reserve(ta, ts, L4Re::Rm::Flags(0));
00907     if (!maymove && err)
00908         return -ENOMEM;
00909
00910     L4Re::Rm::Offset toffs;
00911     L4Re::Rm::Flags tflags;
00912     L4::Cap<L4Re::Dataspace> tds;
00913
00914     err = r->find(&ta, &ts, &toffs, &tflags, &tds);
00915
00916     // there is enough space to expand the mapping in place
00917     if (err == -ENOENT || (err == 0 && (tflags & Rm::F::In_area)))
00918     {
00919         old_area.free(); // pad at the original address
00920         pad_addr = oa + old_size;
00921         *new_addr = old_addr;
00922     }
00923     else if (!maymove)
00924         return -ENOMEM;
00925     else
00926     {
00927         // search for a new area to remap
00928         err = new_area.reserve(0, new_size, Rm::F::Search_addr);
00929         if (err < 0)
00930             return -ENOMEM;
00931
00932         pad_addr = new_area.a + old_size;
00933         *new_addr = reinterpret_cast<void *>(new_area.a);
00934     }
00935 }
00936
00937 if (old_area.is_valid())
00938 {
00939     unsigned long size = old_size;
00940
00941     l4_addr_t a = old_area.a;
00942     unsigned long s = 1;
00943     L4Re::Rm::Offset o;
00944     L4Re::Rm::Flags f;
00945     L4::Cap<L4Re::Dataspace> ds;
00946
00947     while (r->find(&a, &s, &o, &f, &ds) >= 0 && !(f & Rm::F::In_area))
00948     {
00949         if (a < old_area.a)
00950         {
00951             auto d = old_area.a - a;
00952             a = old_area.a;
00953             s -= d;
00954             o += d;
00955         }
00956
00957         if (a + s > old_area.a + old_size)
00958             s = old_area.a + old_size - a;
00959
00960         l4_addr_t x = a - old_area.a + new_area.a;
00961
00962         int err = r->attach(&x, s, Rm::F::In_area | f,
00963             L4::Ipc::make_cap(ds, f.cap_rights()), o);
00964         if (err < 0)
00965             return err;
00966     }

```

```

00967         // count the new attached ds reference
00968         L4Re::virt_cap_alloc->take(ds);
00969
00970         err = r->detach(a, s, &ds, This_task,
00971                       Rm::Detach_exact | Rm::Detach_keep);
00972         if (err < 0)
00973             return err;
00974
00975         switch (err & Rm::Detach_result_mask)
00976         {
00977             case Rm::Split_ds:
00978                 // add a reference as we split up a mapping
00979                 if (ds.is_valid())
00980                     L4Re::virt_cap_alloc->take(ds);
00981                 break;
00982             case Rm::Detached_ds:
00983                 if (ds.is_valid())
00984                     L4Re::virt_cap_alloc->release(ds);
00985                 break;
00986             default:
00987                 break;
00988         }
00989
00990         if (size <= s)
00991             break;
00992         a += s;
00993         size -= s;
00994         s = 1;
00995     }
00996
00997     old_area.free();
00998 }
00999
01000 if (old_size < new_size)
01001 {
01002     l4_addr_t const pad_sz = new_size - old_size;
01003     l4_addr_t toffs;
01004     L4Re::Shared_cap<L4Re::Dataspace> tds;
01005     int err = alloc_anon_mem(pad_sz, &tds, &toffs);
01006     if (err)
01007         return err;
01008
01009     // FIXME: must get the protection rights from the old
01010     // mapping and use the same here, for now just use RWX
01011     err = r->attach(&pad_addr, pad_sz,
01012                  Rm::F::In_area | Rm::F::Detach_free | Rm::F::RWX,
01013                  L4::Ipc::make_cap_rw(tds.get()), toffs);
01014     if (err < 0)
01015         return err;
01016
01017     // release ownership of tds, the region map is now the new owner
01018     tds.release();
01019 }
01020
01021 return 0;
01022 }
01023
01024 int
01025 Vfs::mprotect(const void * /* a */, size_t /* sz */, int prot) L4_NOTHROW
01026 {
01027     return (prot & PROT_WRITE) ? -ENOSYS : 0;
01028 }
01029
01030 int
01031 Vfs::msync(void *, size_t, int) L4_NOTHROW
01032 { return 0; }
01033
01034 int
01035 Vfs::madvise(void *, size_t, int) L4_NOTHROW
01036 { return 0; }
01037
01038 }
01039
01040 L4Re::Vfs::Ops *__rtld_l4re_env_posix_vfs_ops;
01041 extern void *l4re_env_posix_vfs_ops __attribute__((alias("__rtld_l4re_env_posix_vfs_ops"),
01042 visibility("default")));
01043
01044 namespace {
01045     class Real_mount_tree : public L4Re::Vfs::Mount_tree
01046     {
01047     public:
01048         explicit Real_mount_tree(char *n) : Mount_tree(n) {}
01049
01050         void *operator new (size_t size)
01051         { return __rtld_l4re_env_posix_vfs_ops->malloc(size); }
01052
01053         void operator delete (void *mem)

```



```

01053     { __rtld_l4re_env_posix_vfs_ops->free(mem); }
01054 };
01055 }
01056
01057 int
01058 Vfs::mount(char const *path, cxx::Ref_ptr<L4Re::Vfs::File> const &dir) noexcept
01059 {
01060     using L4Re::Vfs::File;
01061     using L4Re::Vfs::Mount_tree;
01062     using L4Re::Vfs::Path;
01063
01064     cxx::Ref_ptr<Mount_tree> root = get_root()->mount_tree();
01065     if (!root)
01066         return -EINVAL;
01067
01068     cxx::Ref_ptr<Mount_tree> base;
01069     Path p = root->lookup(Path(path), &base);
01070
01071     while (!p.empty())
01072     {
01073         Path f = p.strip_first();
01074
01075         if (f.empty())
01076             return -EEXIST;
01077
01078         char *name = __rtld_l4re_env_posix_vfs_ops->strndup(f.path(), f.length());
01079         if (!name)
01080             return -ENOMEM;
01081
01082         auto nt = cxx::make_ref_obj<Real_mount_tree>(name);
01083         if (!nt)
01084         {
01085             __rtld_l4re_env_posix_vfs_ops->free(name);
01086             return -ENOMEM;
01087         }
01088
01089         base->add_child_node(nt);
01090         base = nt;
01091
01092         if (p.empty())
01093         {
01094             nt->mount(dir);
01095             return 0;
01096         }
01097     }
01098
01099     return -EINVAL;
01100 }
01101
01102 #undef DEBUG_LOG
01103 #undef GET_FILE_DBG
01104 #undef GET_FILE

```

## 17.248 vfs.h

```

00001 /*
00002  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <14/sys/compiler.h>
00011
00012 #include <unistd.h>
00013 #include <stdarg.h>
00014 #include <fcntl.h>
00015 #include <sys/stat.h>
00016 #include <sys/time.h>
00017 #include <sys/mman.h>
00018 #include <sys/socket.h>
00019 #include <utime.h>
00020 #include <errno.h>
00021
00022 #ifndef AT_FDCWD
00023 # define AT_FDCWD -100
00024 #endif
00025
00026 #ifdef __cplusplus
00027 #include <14/sys/capability>

```

```

00029 #include <l4/re/cap_alloc>
00030 #include <l4/re/dataspace>
00031 #include <l4/cxx/pair>
00032 #include <l4/cxx/ref_ptr>
00033
00034 namespace L4Re {
00038 namespace Vfs {
00039
00040 class Mount_tree;
00041 class File;
00042
00052 class Generic_file
00053 {
00054 public:
00060 enum Ready_type : unsigned
00061 {
00062     Read = 0,
00063     Write,
00064     Exception
00065 };
00066
00067 virtual ~Generic_file() noexcept = 0;
00068
00080 virtual int unlock_all_locks() noexcept = 0;
00081
00090 virtual int fstat(struct stat64 *buf) const noexcept = 0;
00091
00097 virtual int fchmod(mode_t) noexcept = 0;
00098
00108 virtual int get_status_flags() const noexcept = 0;
00109
00125 virtual int set_status_flags(long flags) noexcept = 0;
00126
00127 virtual int utime(const struct utimbuf *) noexcept = 0;
00128 virtual int utimes(const struct timeval [2]) noexcept = 0;
00129 virtual ssize_t readlink(char *, size_t) = 0;
00130
00146 virtual bool check_ready(Ready_type rt) noexcept = 0;
00147 };
00148
00149 inline
00150 Generic_file::~Generic_file() noexcept
00151 {}
00152
00160 class Directory
00161 {
00162 public:
00163     virtual ~Directory() noexcept = 0;
00164
00178     virtual int faccessat(const char *path, int mode, int flags) noexcept = 0;
00179
00192     virtual int mkdir(const char *path, mode_t mode) noexcept = 0;
00193
00204     virtual int unlink(const char *path) noexcept = 0;
00205
00219     virtual int rename(const char *src_path, const char *dst_path) noexcept = 0;
00220
00234     virtual int link(const char *src_path, const char *dst_path) noexcept = 0;
00235
00248     virtual int symlink(const char *src_path, const char *dst_path) noexcept = 0;
00249
00260     virtual int rmdir(const char *path) noexcept = 0;
00261     virtual int openat(const char *path, int flags, mode_t mode,
00262                       cxx::Ref_ptr<File> *f) noexcept = 0;
00263
00264     virtual ssize_t getdents(char *buf, size_t sizebytes) noexcept = 0;
00265
00266     virtual int fchmodat(const char *pathname,
00267                         mode_t mode, int flags) noexcept = 0;
00268
00269     virtual int utimensat(const char *pathname,
00270                          const struct timespec times[2], int flags) noexcept = 0;
00271
00275     virtual int get_entry(const char *, int, mode_t, cxx::Ref_ptr<File> *) noexcept = 0;
00276 };
00277
00278 inline
00279 Directory::~Directory() noexcept
00280 {}
00281
00287 class Regular_file
00288 {
00289 public:
00290     virtual ~Regular_file() noexcept = 0;
00291
00302     virtual L4::Cap<L4Re::Dataspace> data_space() noexcept = 0;
00303

```

```

00313 virtual ssize_t readv(const struct iovec*, int iovcnt) noexcept = 0;
00314
00325 virtual ssize_t writev(const struct iovec*, int iovcnt) noexcept = 0;
00326
00327 virtual ssize_t preadv(const struct iovec *iov, int iovcnt, off64_t offset) noexcept = 0;
00328 virtual ssize_t pwritev(const struct iovec *iov, int iovcnt, off64_t offset) noexcept = 0;
00329
00337 virtual off64_t lseek(off64_t, int) noexcept = 0;
00338
00339
00347 virtual int ftruncate(off64_t pos) noexcept = 0;
00348
00354 virtual int fsync() const noexcept = 0;
00355
00361 virtual int fdatasync() const noexcept = 0;
00362
00372 virtual int get_lock(struct flock64 *lock) noexcept = 0;
00373
00382 virtual int set_lock(struct flock64 *lock, bool wait) noexcept = 0;
00383 };
00384
00385 inline
00386 Regular_file::~Regular_file() noexcept
00387 {}
00388
00389 class Socket
00390 {
00391 public:
00392     virtual ~Socket() noexcept = 0;
00393     virtual int bind(sockaddr const *, socklen_t) noexcept = 0;
00394     virtual int connect(sockaddr const *, socklen_t) noexcept = 0;
00395     virtual ssize_t send(void const *, size_t, int) noexcept = 0;
00396     virtual ssize_t recv(void *, size_t, int) noexcept = 0;
00397     virtual ssize_t sendto(void const *, size_t, int, sockaddr const *, socklen_t) noexcept = 0;
00398     virtual ssize_t recvfrom(void *, size_t, int, sockaddr *, socklen_t *) noexcept = 0;
00399     virtual ssize_t sendmsg(msghdr const *, int) noexcept = 0;
00400     virtual ssize_t recvmsg(msghdr *, int) noexcept = 0;
00401     virtual int getsockopt(int level, int opt, void *, socklen_t *) noexcept = 0;
00402     virtual int setsockopt(int level, int opt, void const *, socklen_t) noexcept = 0;
00403     virtual int listen(int) noexcept = 0;
00404     virtual int accept(sockaddr *addr, socklen_t *) noexcept = 0;
00405     virtual int shutdown(int) noexcept = 0;
00406
00407     virtual int getsockname(sockaddr *, socklen_t *) noexcept = 0;
00408     virtual int getpeername(sockaddr *, socklen_t *) noexcept = 0;
00409 };
00410
00411 inline
00412 Socket::~Socket() noexcept
00413 {}
00414
00420 class Special_file
00421 {
00422 public:
00423     virtual ~Special_file() noexcept = 0;
00424
00435     virtual int ioctl(unsigned long cmd, va_list args) noexcept = 0;
00436 };
00437
00438 inline
00439 Special_file::~Special_file() noexcept
00440 {}
00441
00455 class File :
00456     public Generic_file,
00457     public Regular_file,
00458     public Directory,
00459     public Special_file,
00460     public Socket
00461 {
00462     friend class Mount_tree;
00463
00464 private:
00465     void operator = (File const &);
00466
00467 protected:
00468     File() noexcept : _ref_cnt(0) {}
00469     File(File const &)
00470     : Generic_file(), Regular_file(), Directory(), Special_file(), _ref_cnt(0)
00471     {}
00472
00473 public:
00474
00475     const char *get_mount(const char *path, cxx::Ref_ptr<File> *dir,
00476                           cxx::Ref_ptr<Mount_tree> *mt = 0) noexcept;
00477
00478     int openat(const char *path, int flags, mode_t mode,

```

```

00479         cxx::Ref_ptr<File> *f) noexcept override;
00480
00481     void add_ref() noexcept { ++_ref_cnt; }
00482     int remove_ref() noexcept { return --_ref_cnt; }
00483
00484     virtual ~File() noexcept = 0;
00485
00486     cxx::Ref_ptr<Mount_tree> mount_tree() const noexcept
00487     { return _mount_tree; }
00488
00489     char const *path() const noexcept { return _path; }
00490
00491 private:
00492     int _ref_cnt;
00493     cxx::Ref_ptr<Mount_tree> _mount_tree;
00494     char _path[80] = "";
00495 };
00496
00497 inline
00498 File::~File() noexcept
00499 {}
00500
00501 class Path
00502 {
00503 private:
00504     char const *_p;
00505     unsigned _l;
00506
00507 public:
00508     Path() noexcept : _p(0), _l(0) {}
00509
00510     explicit Path(char const *p) noexcept : _p(p)
00511     { for (_l = 0; *p; ++p, ++_l) ; }
00512
00513     Path(char const *p, unsigned l) noexcept : _p(p), _l(l)
00514     {}
00515
00516     static bool __is_sep(char s) noexcept;
00517
00518     Path cmp_path(char const *prefix) const noexcept;
00519
00520     struct Invalid_ptr;
00521     operator Invalid_ptr const * () const
00522     { return reinterpret_cast<Invalid_ptr const *>(_p); }
00523
00524     unsigned length() const { return _l; }
00525     char const *path() const { return _p; }
00526
00527     bool empty() const { return _l == 0; }
00528
00529     bool is_sep(unsigned offset) const { return __is_sep(_p[offset]); }
00530
00531     bool strip_sep()
00532     {
00533         bool s = false;
00534         for (; __is_sep(*_p) && _l; ++_p, --_l)
00535             s = true;
00536         return s;
00537     }
00538
00539     Path first() const
00540     {
00541         unsigned i;
00542         for (i = 0; i < _l && !is_sep(i); ++i)
00543             ;
00544         return Path(_p, i);
00545     }
00546
00547     Path strip_first()
00548     {
00549         Path r = first();
00550         _p += r.length();
00551         _l -= r.length();
00552         strip_sep();
00553         return r;
00554     }
00555 };
00556
00557 };
00558
00559 class Mount_tree
00560 {
00561 public:
00562     explicit Mount_tree(char *n) noexcept;
00571

```

```

00572 Path lookup(Path const &path, cxx::Ref_ptr<Mount_tree> *mt,
00573             cxx::Ref_ptr<Mount_tree> *mp = 0) noexcept;
00574
00575 Path find(Path const &p, cxx::Ref_ptr<Mount_tree> *t) noexcept;
00576
00577 cxx::Ref_ptr<File> mount() const
00578 { return _mount; }
00579
00580 void mount(cxx::Ref_ptr<File> const &m)
00581 {
00582     m->_mount_tree = cxx::ref_ptr(this);
00583     _mount = m;
00584 }
00585
00586 static int create_tree(cxx::Ref_ptr<Mount_tree> const &root,
00587                       char const *path,
00588                       cxx::Ref_ptr<File> const &dir) noexcept;
00589
00590 void add_child_node(cxx::Ref_ptr<Mount_tree> const &cld);
00591
00592 virtual ~Mount_tree() noexcept = 0;
00593
00594 void add_ref() noexcept { ++_ref_cnt; }
00595 int remove_ref() noexcept { return --_ref_cnt; }
00596
00597 private:
00598 friend class Real_mount_tree;
00599
00600 int _ref_cnt;
00601 char *_name;
00602 cxx::Ref_ptr<Mount_tree> _cld;
00603 cxx::Ref_ptr<Mount_tree> _sib;
00604 cxx::Ref_ptr<File> _mount;
00605 };
00606
00607 inline
00608 Mount_tree::~Mount_tree() noexcept
00609 {}
00610
00611 inline bool
00612 Path::__is_sep(char s) noexcept
00613 { return s == '/'; }
00614
00615 inline Path
00616 Path::cmp_path(char const *n) const noexcept
00617 {
00618     char const *p = _p;
00619     for (; *p && !__is_sep(*p) && *n; ++p, ++n)
00620         if (*p != *n)
00621             return Path();
00622     if (*n || (*p && !__is_sep(*p)))
00623         return Path();
00624     return Path(p, _l - (p - _p));
00625 }
00626
00627 inline
00628 Mount_tree::Mount_tree(char *n) noexcept
00629 : _ref_cnt(0), _name(n)
00630 {}
00631
00632 inline Path
00633 Mount_tree::find(Path const &p, cxx::Ref_ptr<Mount_tree> *t) noexcept
00634 {
00635     if (!_cld)
00636         return Path();
00637     for (cxx::Ref_ptr<Mount_tree> x = _cld; x; x = x->_sib)
00638     {
00639         Path const r = p.cmp_path(x->_name);
00640         if (r)
00641         {
00642             *t = x;
00643             return r;
00644         }
00645     }
00646     return Path();
00647 }
00648
00649 inline Path
00650 Mount_tree::lookup(Path const &path, cxx::Ref_ptr<Mount_tree> *mt,
00651                   cxx::Ref_ptr<Mount_tree> *mp) noexcept
00652 {
00653     cxx::Ref_ptr<Mount_tree> x(this);
00654     Path p = path;

```

```

00659
00660     if (p.first().cmp_path("."))
00661         p.strip_first();
00662     Path last_mp = p;
00663
00664     if (mp)
00665         *mp = x;;
00666
00667     while (1)
00668     {
00669         Path r = x->find(p, &x);
00670
00671         if (!r)
00672         {
00673             if (mp)
00674                 return last_mp;
00675
00676             if (mt)
00677                 *mt = x;
00678
00679             return p;
00680         }
00681
00682         r.strip_sep();
00683
00684         if (mp && x->_mount)
00685         {
00686             last_mp = r;
00687             *mp = x;
00688         }
00689
00690         if (r.empty())
00691         {
00692             if (mt)
00693                 *mt = x;
00694
00695             if (mp)
00696                 return last_mp;
00697             else
00698                 return r;
00699         }
00700
00701         p = r;
00702     }
00703 }
00704
00705 inline
00706 void
00707 Mount_tree::add_child_node(cxx::Ref_ptr<Mount_tree> const &cld)
00708 {
00709     cld->_sib = _cld;
00710     _cld = cld;
00711 }
00712
00713 inline
00714 const char *
00715 File::get_mount(const char *path, cxx::Ref_ptr<File> *dir,
00716                 cxx::Ref_ptr<Mount_tree> *mt) noexcept
00717 {
00718     if (!_mount_tree)
00719     {
00720         *dir = cxx::ref_ptr(this);
00721         return path;
00722     }
00723
00724     cxx::Ref_ptr<Mount_tree> mp;
00725     Path p = _mount_tree->lookup(Path(path), mt, &mp);
00726     if (mp->mount())
00727     {
00728         *dir = mp->mount();
00729         return p.path();
00730     }
00731     else
00732     {
00733         *dir = cxx::ref_ptr(this);
00734         return path;
00735     }
00736 }
00737
00738 inline int
00739 File::openat(const char *path, int flags, mode_t mode,
00740              cxx::Ref_ptr<File> *f) noexcept
00741 {
00742     cxx::Ref_ptr<File> dir;
00743     cxx::Ref_ptr<Mount_tree> mt;
00744     char const *m_path = get_mount(path, &dir, &mt);

```

```

00746
00747     int res = dir->get_entry(m_path, flags, mode, f);
00748
00749     if (res < 0)
00750         return res;
00751
00752     if (!(*f)->_mount_tree && mt)
00753         (*f)->_mount_tree = mt;
00754
00755     // Debugging {
00756     size_t i = 0;
00757     for (; i < sizeof((*f)->_path) - 1 && path[i]; ++i)
00758         (*f)->_path[i] = path[i];
00759     (*f)->_path[i] = '\0';
00760     // } Debugging
00761
00762     return res;
00763 }
00764
00773 class Mman
00774 {
00775 public:
00777     virtual int mmap2(void *start, size_t len, int prot, int flags, int fd,
00778                     off_t offset, void **ptr) noexcept = 0;
00779
00781     virtual int munmap(void *start, size_t len) noexcept = 0;
00782
00784     virtual int mremap(void *old, size_t old_sz, size_t new_sz, int flags,
00785                     void **new_addr) noexcept = 0;
00786
00788     virtual int mprotect(const void *a, size_t sz, int prot) noexcept = 0;
00789
00791     virtual int msync(void *addr, size_t len, int flags) noexcept = 0;
00792
00794     virtual int madvise(void *addr, size_t len, int advice) noexcept = 0;
00795
00796     virtual ~Mman() noexcept = 0;
00797 };
00798
00799 inline
00800 Mman::~Mman() noexcept {}
00801
00802 class File_factory
00803 {
00804 private:
00805     int _ref_cnt = 0;
00806     int _proto = 0;
00807     char const *_proto_name = 0;
00808
00809     template<typename T> friend struct cxx::Default_ref_counter;
00810     void add_ref() noexcept { ++_ref_cnt; }
00811     int remove_ref() noexcept { return --_ref_cnt; }
00812
00813 public:
00814     explicit File_factory(int proto) : _proto(proto) {}
00815     explicit File_factory(char const *proto_name) : _proto_name(proto_name) {}
00816     File_factory(int proto, char const *proto_name)
00817         : _proto(proto), _proto_name(proto_name)
00818     {}
00819
00820     File_factory(File_factory const &) = delete;
00821     File_factory &operator = (File_factory const &) = delete;
00822
00823     char const *proto_name() const { return _proto_name; }
00824     int proto() const { return _proto; }
00825
00826     virtual ~File_factory() noexcept = 0;
00827     virtual cxx::Ref_ptr<File> create(L4::Cap<void> file) = 0;
00828 };
00829
00830 inline File_factory::~File_factory() noexcept {}
00831
00832 template<typename IFACE, typename IMPL>
00833 class File_factory_t : public File_factory
00834 {
00835 public:
00836     File_factory_t()
00837         : File_factory(IFACE::Protocol, L4::kobject_typeid<IFACE>()->name())
00838     {}
00839
00840     cxx::Ref_ptr<File> create(L4::Cap<void> file) override
00841     { return cxx::make_ref_obj<IMPL>(L4::cap_cast<IFACE>(file)); }
00842 };
00843
00857 class File_system
00858 {
00859 protected:

```

```

00860     File_system *_next;
00861
00862 public:
00863     File_system() noexcept : _next(0) {}
00864     virtual char const *type() const noexcept = 0;
00870
00887     virtual int mount(char const *source, unsigned long mountflags,
00888                     void const *data, cxx::Ref_ptr<File> *dir) noexcept = 0;
00889
00890     virtual ~File_system() noexcept = 0;
00891
00896     File_system *next() const noexcept { return _next; }
00897     File_system *&next() noexcept { return _next; }
00898     void next(File_system *n) noexcept { _next = n; }
00899 };
00900
00901 inline
00902 File_system::~File_system() noexcept
00903 {}
00904
00905 class File_system_list
00906 {
00907 public:
00908     class Iterator
00909     {
00910     public:
00911         explicit constexpr Iterator(File_system *c = nullptr) : _c(c) {}
00912
00913         Iterator &operator++()
00914         {
00915             if (_c)
00916                 _c = _c->next();
00917             return *this;
00918         }
00919
00920         bool operator==(Iterator const &other) const { return _c == other._c; }
00921         bool operator!=(Iterator const &other) const { return _c != other._c; }
00922         File_system *operator*() const { return _c; }
00923         File_system *operator->() const { return _c; }
00924
00925     private:
00926         File_system *_c;
00927     };
00928
00929     File_system_list(File_system *head) : _head(head) {}
00930
00931     constexpr Iterator begin() const
00932     { return Iterator(_head); }
00933
00934     constexpr Iterator end() const
00935     { return Iterator(); }
00936
00937 private:
00938     File_system *_head;
00939 };
00940
00946 class Fs
00947 {
00948 public:
00954     virtual cxx::Ref_ptr<File> get_file(int fd) noexcept = 0;
00955
00957     virtual cxx::Ref_ptr<File> get_root() noexcept = 0;
00958
00960     virtual cxx::Ref_ptr<File> get_cwd() noexcept { return get_root(); }
00961
00963     virtual void set_cwd(cxx::Ref_ptr<File> const &) noexcept {}
00964
00970     virtual int alloc_fd(cxx::Ref_ptr<File> const &f = cxx::Ref_ptr<>::Nil) noexcept = 0;
00971
00982     virtual cxx::Pair<cxx::Ref_ptr<File>, int>
00983         set_fd(int fd, cxx::Ref_ptr<File> const &f = cxx::Ref_ptr<>::Nil) noexcept = 0;
00984
00990     virtual cxx::Ref_ptr<File> free_fd(int fd) noexcept = 0;
00991
00999     virtual int mount(char const *path, cxx::Ref_ptr<File> const &dir) noexcept = 0;
01000
01008     virtual int register_file_system(File_system *f) noexcept = 0;
01009
01017     virtual int unregister_file_system(File_system *f) noexcept = 0;
01018
01026     virtual File_system *get_file_system(char const *fstype) noexcept = 0;
01027
01036     virtual File_system_list file_system_list() noexcept = 0;
01037
01041     int mount(char const *source, char const *target,
01042             char const *fstype, unsigned long mountflags,
01043             void const *data) noexcept;

```



```

01044
01045 virtual int register_file_factory(cxx::Ref_ptr<File_factory> f) noexcept = 0;
01046 virtual int unregister_file_factory(cxx::Ref_ptr<File_factory> f) noexcept = 0;
01047 virtual cxx::Ref_ptr<File_factory> get_file_factory(int proto) noexcept = 0;
01048 virtual cxx::Ref_ptr<File_factory> get_file_factory(char const *proto_name) noexcept = 0;
01049
01050 virtual ~Fs() = 0;
01051
01052 private:
01053     int mount_one(char const *source, char const *target,
01054                  File_system *fs, unsigned long mountflags,
01055                  void const *data) noexcept;
01056 };
01057
01058 inline int
01059 Fs::mount_one(char const *source, char const *target,
01060               File_system *fs, unsigned long mountflags,
01061               void const *data) noexcept
01062 {
01063     cxx::Ref_ptr<File> dir;
01064     int res = fs->mount(source, mountflags, data, &dir);
01065
01066     if (res < 0)
01067         return res;
01068
01069     return mount(target, dir);
01070 }
01071
01072 inline int
01073 Fs::mount(char const *source, char const *target,
01074           char const *fstype, unsigned long mountflags,
01075           void const *data) noexcept
01076 {
01077     if (fstype[0] == 'a'
01078         && fstype[1] == 'u'
01079         && fstype[2] == 't'
01080         && fstype[3] == 'o'
01081         && fstype[4] == 0)
01082     {
01083         File_system_list fsl = file_system_list();
01084         for (File_system_list::Iterator c = fsl.begin(); c != fsl.end(); ++c)
01085             if (mount_one(source, target, *c, mountflags, data) == 0)
01086                 return 0;
01087
01088         return -ENODEV;
01089     }
01090
01091     File_system *fs = get_file_system(fstype);
01092
01093     if (!fs)
01094         return -ENODEV;
01095
01096     return mount_one(source, target, fs, mountflags, data);
01097 }
01098
01099 inline
01100 Fs::~Fs()
01101 {}
01102
01103 class Ops : public Mman, public Fs
01104 {
01105 public:
01106     virtual void *malloc(size_t bytes) noexcept = 0;
01107     virtual void free(void *mem) noexcept = 0;
01108     virtual ~Ops() noexcept = 0;
01109
01110     char *strndup(char const *str, unsigned l) noexcept
01111     {
01112         unsigned len;
01113         for (len = 0; str[len] && len < l; ++len)
01114             ;
01115
01116         if (len == 0)
01117             return nullptr;
01118
01119         ++len;
01120
01121         char *b = static_cast<char *>(this->malloc(len));
01122         if (b == nullptr)
01123             return nullptr;
01124
01125         char *r = b;
01126         for (; len - 1 > 0 && *str; --len, ++b, ++str)
01127             *b = *str;
01128
01129         *b = 0;
01130
01131         return r;
01132     }
01133
01134     return r;
01135 }

```

```

01137     }
01138
01139 };
01140
01141 inline
01142 Ops::~Ops() noexcept
01143 {}
01144
01145 }}
01146
01147 #endif
01148

```

## 17.249 virtio-net

```

00001 // vi:ft=cpp
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2022, 2024 Kernkonzept GmbH.
00005  * Author(s): Stephan Gerhold <stephan.gerhold@kernkonzept.com>
00006  */
00007 #pragma once
00008
00009 #include <cstring>
00010 #include <functional>
00011
00012 #include <l4/cxx/exceptions>
00013 #include <l4/cxx/minmax>
00014 #include <l4/re/dataspace>
00015 #include <l4/re/env>
00016 #include <l4/re/error_helper>
00017 #include <l4/re/util/unique_cap>
00018 #include <l4/sys/consts.h>
00019
00020 #include <l4/l4virtio/client/l4virtio>
00021 #include <l4/l4virtio/l4virtio>
00022 #include <l4/l4virtio/virtio_net.h>
00023 #include <l4/l4virtio/virtqueue>
00024
00025 namespace L4virtio { namespace Driver {
00026
00030 class Virtio_net_device : public L4virtio::Driver::Device
00031 {
00032 public:
00037     struct Packet
00038     {
00039         l4virtio_net_header_t hdr;
00040         l4_uint8_t data[1500 + 14]; /* MTU + Ethernet header */
00041     };
00042
00047     int rx_queue_size() const
00048     { return max_queue_size(0); }
00049
00054     int tx_queue_size() const
00055     { return max_queue_size(1); }
00056
00066     void setup_device(L4::Cap<L4virtio::Device> srvcap)
00067     {
00068         // Contact device.
00069         driver_connect(srvcap, false);
00070
00071         if (_config->device != L4VIRTIO_ID_NET)
00072             L4Re::chksys(-L4_ENODEV, "Device is not a network device.");
00073
00074         if (_config->num_queues < 2)
00075             L4Re::chksys(-L4_EINVAL, "Invalid number of queues reported.");
00076
00077         auto rxqsz = rx_queue_size();
00078         auto txqsz = tx_queue_size();
00079
00080         // Allocate memory for RX/TX queue and RX/TX packet buffers
00081         auto rxqoff = 0;
00082         auto txqoff = l4_round_size(rxqoff + rxqsz * _rxq.total_size(rxqsz),
00083                                     L4virtio::Virtqueue::Desc_align);
00084         auto rxpktoff = l4_round_size(txqoff + txqsz * _txq.total_size(txqsz),
00085                                     L4virtio::Virtqueue::Desc_align);
00086         auto txpktoff = rxpktoff + rxqsz * sizeof(Packet);
00087         auto totalsz = txpktoff + txqsz * sizeof(Packet);
00088
00089         _queue_ds = L4Re::chkcapi(L4Re::Util::make_unique_cap<L4Re::Dataspace>(),
00090                                   "Allocate queue dataspace capability");
00091         auto *e = L4Re::Env::env();
00092         L4Re::chksys(e->mem_alloc()->alloc(totalsz, _queue_ds.get()),

```

```

00093                                     L4Re::Mem_alloc::Continuous
00094                                     | L4Re::Mem_alloc::Pinned),
00095     "Allocate memory for virtio structures");
00096
00097     L4Re::chksys(e->rm()->attach(&_queue_region, totalsz,
00098                                     L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00099                                     L4::Ipc::make_cap_rw(_queue_ds.get()), 0,
00100                                     L4_PAGESHIFT),
00101     "Attach dataspace for virtio structures");
00102
00103     l4_uint64_t devaddr;
00104     L4Re::chksys(register_ds(_queue_ds.get(), 0, totalsz, &devaddr),
00105     "Register queue dataspace with device");
00106
00107     _rxq.init_queue(rxqsz, _queue_region.get() + rxqoff);
00108     _txq.init_queue(txqsz, _queue_region.get() + txqoff);
00109
00110     config_queue(0, rxqsz, devaddr + rxqoff,
00111                 devaddr + rxqoff + _rxq.avail_offset(),
00112                 devaddr + rxqoff + _rxq.used_offset());
00113     config_queue(1, txqsz, devaddr + txqoff,
00114                 devaddr + txqoff + _txq.avail_offset(),
00115                 devaddr + txqoff + _txq.used_offset());
00116
00117     _rxpkts = reinterpret_cast<Packet*>(_queue_region.get() + rxpktoff);
00118     _txpkts = reinterpret_cast<Packet*>(_queue_region.get() + txpktoff);
00119
00120     // Prepare descriptors to save work later
00121     for (l4_uint16_t descno = 0; descno < rxqsz; ++descno)
00122     {
00123         auto &desc = _rxq.desc(descno);
00124         desc.addr = L4virtio::Ptr<void>(devaddr + rxpktoff +
00125                                         descno * sizeof(Packet));
00126         desc.len = sizeof(Packet);
00127         desc.flags.write() = 1;
00128     }
00129     for (l4_uint16_t descno = 0; descno < txqsz; ++descno)
00130     {
00131         auto &desc = _txq.desc(descno);
00132         desc.addr = L4virtio::Ptr<void>(devaddr + txpktoff +
00133                                         descno * sizeof(Packet));
00134         desc.len = sizeof(Packet);
00135     }
00136
00137     // Setup notification IRQ
00138     _driver_notification_irq =
00139         L4Re::chkcap(L4Re::Util::make_unique_cap<L4::Irq>()),
00140         "Allocate notification capability");
00141
00142     L4Re::chksys(l4_error(e->factory()->create(_driver_notification_irq.get())),
00143     "Create irq for notifications from device");
00144
00145     L4Re::chksys(_device->bind(0, _driver_notification_irq.get()),
00146     "Bind driver notification interrupt");
00147
00148     // Finish handshake with device
00149     l4virtio_set_feature(_config->driver_features_map,
00150                         L4VIRTIO_FEATURE_VERSION_1);
00151     l4virtio_set_feature(_config->driver_features_map, L4VIRTIO_NET_F_MAC);
00152     driver_acknowledge();
00153 }
00154
00158 l4virtio_net_config_t const &device_config() const
00159 {
00160     return *_config->device_config<l4virtio_net_config_t>();
00161 }
00162
00169 int bind_rx_notification_irq(L4::Cap<L4::Thread> thread, l4_umword_t label)
00170 {
00171     return l4_error(_driver_notification_irq->bind_thread(thread, label));
00172 }
00173
00180 Packet &rx_pkt(l4_uint16_t descno)
00181 {
00182     if (descno >= _rxq.num())
00183         throw L4::Bounds_error("Invalid used descriptor number in RX queue");
00184     return _rxpkts[descno];
00185 }
00186
00202 l4_uint16_t wait_rx(l4_uint32_t *len = nullptr)
00203 {
00204     l4_uint16_t descno;
00205     // Wait until used descriptor becomes available.
00206     for (;;)
00207     {
00208         descno = _rxq.find_next_used(len);
00209         if (descno != Virtqueue::Eoq)

```

```

00210         break;
00211
00212         L4Re::chksys(_driver_notification_irq->receive(), "Wait for RX");
00213     }
00214
00215     if (len)
00216         // Ensure that the length provided by the device in wait_for_next_used()
00217         // is not larger than the buffer and subtract the length of the header.
00218         *len = cxx::min(*len - sizeof(_rxpkts[0].hdr), sizeof(_rxpkts[0].data));
00219     return descno;
00220 }
00221
00230 void finish_rx(l4_uint16_t descno)
00231 {
00232     _rxq.free_descriptor(descno, descno);
00233 }
00234
00238 void queue_rx()
00239 {
00240     l4_uint16_t descno;
00241     while ((descno = _rxq.alloc_descriptor()) != Virtqueue::Eoq)
00242         _rxq.enqueue_descriptor(descno);
00243     notify(_rxq);
00244 }
00245
00260 bool tx(std::function<l4_uint32_t(Packet&)> prepare)
00261 {
00262     auto descno = _txq.alloc_descriptor();
00263     if (descno == Virtqueue::Eoq)
00264     {
00265         // Try again after cleaning old descriptors that have already been used
00266         free_used_tx_descriptors();
00267         descno = _txq.alloc_descriptor();
00268         if (descno == Virtqueue::Eoq)
00269             return false;
00270     }
00271
00272     auto &pkt = _txpkts[descno];
00273     auto &desc = _txq.desc(descno);
00274     desc.len = sizeof(pkt.hdr) + prepare(pkt);
00275     send(_txq, descno);
00276     return true;
00277 }
00278
00279 private:
00280 void free_used_tx_descriptors()
00281 {
00282     l4_uint16_t used;
00283     while ((used = _txq.find_next_used()) != Virtqueue::Eoq)
00284     {
00285         if (used >= _txq.num())
00286             throw L4::Bounds_error("Invalid used descriptor number in TX queue");
00287         _txq.free_descriptor(used, used);
00288     }
00289 }
00290
00291 private:
00292     L4Re::Util::Unique_cap<L4Re::Dataspace> _queue_ds;
00293     L4Re::Rm::Unique_region<l4_uint8_t *> _queue_region;
00294     L4Re::Util::Unique_cap<L4::Irq> _driver_notification_irq;
00295     L4virtio::Driver::Virtqueue _rxq, _txq;
00296     Packet *_rxpkts, *_txpkts;
00297 };
00298
00299 } }

```

## 17.250 l4virtio

```

00001 // vi:ft=cpp
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2015-2020, 2022, 2024 Kernkonzept GmbH.
00005  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00006  *
00007  */
00008 #pragma once
00009
00010 #include <l4/sys/factory>
00011 #include <l4/sys/semaphore>
00012 #include <l4/re/dataspace>
00013 #include <l4/re/env>
00014 #include <l4/re/util/unique_cap>
00015 #include <l4/re/util/object_registry>

```

```

00016 #include <l4/re/error_helper>
00017
00018 #include <l4/util/atomic.h>
00019 #include <l4/util/bitops.h>
00020 #include <l4/l4virtio/l4virtio>
00021 #include <l4/l4virtio/virtqueue>
00022 #include <l4/sys/consts.h>
00023
00024 #include <cstring>
00025
00026 namespace L4virtio { namespace Driver {
00027
00031 class Device
00032 {
00033 public:
00056 void driver_connect(L4::Cap<L4virtio::Device> srvcap, bool manage_notify = true)
00057 {
00058     _device = srvcap;
00059
00060     _next_devaddr = L4_SUPERPAGESIZE;
00061
00062     auto *e = L4Re::Env::env();
00063
00064     // Set up the virtio configuration page.
00065
00066     _config_cap = L4Re::chkcapi(L4Re::Util::make_unique_cap<L4Re::Dataspace>(),
00067     "Allocate config dataspace capability");
00068
00069     l4_addr_t ds_offset;
00070     L4Re::chksys(_device->device_config(_config_cap.get(), &ds_offset),
00071     "Request virtio config page");
00072
00073     if (ds_offset & ~L4_PAGEMASK)
00074         L4Re::chksys(-L4_EINVAL, "Virtio config page is page aligned.");
00075
00076     L4Re::chksys(e->rm()->attach(&_config, L4_PAGESIZE,
00077     L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00078     L4::Ipc::make_cap_rw(_config_cap.get(), ds_offset,
00079     L4_PAGESHIFT),
00080     "Attach config dataspace");
00081
00082     if (memcmp(&_config->magic, "virt", 4) != 0)
00083         L4Re::chksys(-L4_ENODEV, "Device config has wrong magic value");
00084
00085     if (_config->version != 2)
00086         L4Re::chksys(-L4_ENODEV, "Invalid virtio version, must be 2");
00087
00088     _device->set_status(0); // reset
00089     int status = L4VIRTIO_STATUS_ACKNOWLEDGE;
00090     _device->set_status(status);
00091
00092     status |= L4VIRTIO_STATUS_DRIVER;
00093     _device->set_status(status);
00094
00095     if (_config->fail_state())
00096         L4Re::chksys(-L4_EIO, "Device failure during initialisation.");
00097
00098     // Set up the interrupt used to notify the device about events.
00099     // (only supporting one interrupt with index 0 at the moment)
00100
00101     _host_irq = L4Re::chkcapi(L4Re::Util::make_unique_cap<L4::Irq>(),
00102     "Allocate host IRQ capability");
00103
00104     L4Re::chksys(_device->device_notification_irq(0, _host_irq.get()),
00105     "Request device notification interrupt.");
00106
00107     // Set up the interrupt to get notifications from the device.
00108     // (only supporting one interrupt with index 0 at the moment)
00109     if (manage_notify)
00110     {
00111         _driver_notification =
00112             L4Re::chkcapi(L4Re::Util::make_unique_cap<L4::Semaphore>(),
00113             "Allocate notification capability");
00114
00115         L4Re::chksys(l4_error(e->factory()->create(_driver_notification.get())),
00116         "Create semaphore for notifications from device");
00117
00118         L4Re::chksys(_device->bind(0, _driver_notification.get()),
00119         "Bind driver notification interrupt");
00120     }
00121 }
00122
00129 int bind_notification_irq(unsigned index, L4::Cap<L4::Triggerable> irq) const
00130 { return l4_error(_device->bind(index, irq)); }
00131
00133 bool fail_state() const { return _config->fail_state(); }
00134

```

```

00145 bool feature_negotiated(unsigned int feat) const
00146 { return l4virtio_get_feature(_config->driver_features_map, feat); }
00147
00156 int driver_acknowledge()
00157 {
00158     if (!l4virtio_get_feature(_config->dev_features_map,
00159                             L4VIRTIO_FEATURE_VERSION_1))
00160         L4Re::chksys(-L4_ENODEV,
00161                     "Require Virtio 1.0 device; Legacy device not supported.");
00162
00163     _config->driver_features_map[0] &= _config->dev_features_map[0];
00164     _config->driver_features_map[1] &= _config->dev_features_map[1];
00165
00166     _device->set_status(_config->status | L4VIRTIO_STATUS_FEATURES_OK);
00167
00168     if (!(_config->status & L4VIRTIO_STATUS_FEATURES_OK))
00169         L4Re::chksys(-L4_EINVAL, "Negotiation of device features.");
00170
00171     _device->set_status(_config->status | L4VIRTIO_STATUS_DRIVER_OK);
00172
00173     if (_config->fail_state())
00174         return -L4_EIO;
00175
00176     return L4_EOK;
00177 }
00178
00196 int register_ds(L4::Cap<L4Re::Dataspace> ds, l4_umword_t offset,
00197               l4_umword_t size, l4_uint64_t *devaddr)
00198 {
00199     *devaddr = next_device_address(size);
00200     return _device->register_ds(L4::Ipc::make_cap_rw(ds), *devaddr, offset, size);
00201 }
00202
00212 int config_queue(int num, unsigned size, l4_uint64_t desc_addr,
00213                 l4_uint64_t avail_addr, l4_uint64_t used_addr)
00214 {
00215     auto *queueconf = &_amp;config->queues()[num];
00216     queueconf->num = size;
00217     queueconf->desc_addr = desc_addr;
00218     queueconf->avail_addr = avail_addr;
00219     queueconf->used_addr = used_addr;
00220     queueconf->ready = 1;
00221
00222     return _device->config_queue(num);
00223 }
00224
00230 int max_queue_size(int num) const
00231 {
00232     return _config->queues()[num].num_max;
00233 }
00234
00247 int send_and_wait(Virtqueue &queue, l4_uint16_t descno)
00248 {
00249     send(queue, descno);
00250
00251     // wait for a reply, we assume that no other
00252     // request will get in the way.
00253     auto head = wait_for_next_used(queue);
00254
00255     if (head < 0)
00256         return head;
00257
00258     return (head == descno) ? L4_EOK : -L4_EINVAL;
00259 }
00260
00268 int wait(int index) const
00269 {
00270     if (index != 0)
00271         return -L4_EEXIST;
00272
00273     return l4_ipc_error(_driver_notification->down(), l4_utcb());
00274 }
00275
00291 int wait_for_next_used(Virtqueue &queue, l4_uint32_t *len = nullptr) const
00292 {
00293     while (true)
00294     {
00295         int err = wait(0);
00296
00297         if (err < 0)
00298             return err;
00299
00300         auto head = queue.find_next_used(len);
00301         if (head != Virtqueue::Eoq) // spurious interrupt?
00302             return head;
00303     }
00304 }

```

```

00305
00312 void send(Virtqueue &queue, l4_uint16_t descno)
00313 {
00314     queue.enqueue_descriptor(descno);
00315     notify(queue);
00316 }
00317
00318 void notify(Virtqueue &queue)
00319 {
00320     if (!queue.no_notify_host())
00321         _host_irq->trigger();
00322 }
00323
00324 private:
00325 l4_uint64_t next_device_address(l4_umword_t size)
00326 {
00327     l4_umword_t ret;
00328     size = l4_round_page(size);
00329     do
00330     {
00331         ret = _next_devaddr;
00332         if (l4_umword_t(~0) - ret < size)
00333             L4Re::chksys(-L4_ENOMEM, "Out of device address space.");
00334     }
00335     while (!l4util_cmpxchg(&_next_devaddr, ret, ret + size));
00336     return ret;
00337 }
00338
00339 protected:
00340 L4::Cap<L4virtio::Device> _device;
00341 L4Re::Rm::Unique_region<L4virtio::Device::Config_hdr *> _config;
00342 l4_umword_t _next_devaddr;
00343 L4Re::Util::Unique_cap<L4::Semaphore> _driver_notification;
00344
00345 private:
00346 L4Re::Util::Unique_cap<L4::Irq> _host_irq;
00347 L4Re::Util::Unique_cap<L4Re::Dataspace> _config_cap;
00348 };
00349
00350 }

```

## 17.251 l4virtio

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2013-2024 Kernkonzept GmbH.
00005  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00006  *             Matthias Lange <matthias.lange@kernkonzept.com>
00007  */
00008
00009 #pragma once
00010
00011 #include "virtio.h"
00012 #include <l4/sys/capability>
00013 #include <l4/sys/cxx/ipc_client>
00014 #include <l4/re/dataspace>
00015 #include <l4/sys/irq>
00016 #include <l4/cxx/utis>
00017
00018 namespace L4virtio {
00019 class Device :
00020     public L4::Kobject_t<Device, L4::Icu, L4VIRTIO_PROTOCOL,
00021         L4::Type_info::Demand_t<1> >
00022 {
00023 public:
00024     typedef l4virtio_config_queue_t Config_queue;
00025     struct Config_hdr : l4virtio_config_hdr_t
00026     {
00027         Config_queue *queues() const
00028         { return l4virtio_config_queues(this); }
00029
00030         template <typename T>
00031         T *device_config() const
00032         {
00033             return static_cast<T*>(l4virtio_device_config(this));
00034         }
00035     };
00036
00037     int config_queue(unsigned num, L4::Cap<L4::Triggerable> out_notify,
00038         L4::Cap<L4::Triggerable> in_notify,
00039         l4_timeout_s to = L4_IPC_TIMEOUT_NEVER)
00040     {

```

```

00060         return send_cmd(L4VIRTIO_CMD_CFG_QUEUE | num,
00061                         out_notify, in_notify, to);
00062     }
00063
00064     int notify_queue(unsigned num, L4::Cap<L4::Triggerable> out_notify,
00065                     L4::Cap<L4::Triggerable> in_notify,
00066                     l4_timeout_s to = L4_IPC_TIMEOUT_NEVER)
00067     {
00068         return send_cmd(L4VIRTIO_CMD_NOTIFY_QUEUE | num,
00069                         out_notify, in_notify, to);
00070     }
00071
00080     bool fail_state() const
00081     {
00082         auto cfg_status = cxx::access_once(&status);
00083         return cfg_status
00084             & (L4VIRTIO_STATUS_FAILED | L4VIRTIO_STATUS_DEVICE_NEEDS_RESET);
00085     }
00086
00087     int set_status(unsigned new_status, L4::Cap<L4::Triggerable> out_notify,
00088                  L4::Cap<L4::Triggerable> in_notify,
00089                  l4_timeout_s to = L4_IPC_TIMEOUT_NEVER)
00090     {
00091         return send_cmd(L4VIRTIO_CMD_SET_STATUS | new_status,
00092                         out_notify, in_notify, to);
00093     }
00094
00095     int cfg_changed(unsigned reg, L4::Cap<L4::Triggerable> out_notify,
00096                   L4::Cap<L4::Triggerable> in_notify,
00097                   l4_timeout_s to = L4_IPC_TIMEOUT_NEVER)
00098     {
00099         return send_cmd(L4VIRTIO_CMD_CFG_CHANGED | reg,
00100                         out_notify, in_notify, to);
00101     }
00102
00103     int send_cmd(unsigned command, L4::Cap<L4::Triggerable> out_notify,
00104                  L4::Cap<L4::Triggerable> in_notify,
00105                  l4_timeout_s to = L4_IPC_TIMEOUT_NEVER)
00106     {
00107         cxx::write_now(&cmd, command);
00108
00109         if (out_notify)
00110             out_notify->trigger();
00111
00112         auto utcb = l4_utcb();
00113         auto ipc_to = l4_timeout(L4_IPC_TIMEOUT_0, to);
00114
00115         do
00116         {
00117             if (in_notify)
00118                 if (l4_ipc_error(l4_ipc_receive(in_notify.cap(), utcb, ipc_to),
00119                                utcb) == L4_IPC_RETIMEOUT)
00120                     break;
00121         }
00122         while (cxx::access_once(&cmd));
00123
00124         return cxx::access_once(&cmd) ? -L4_EBUSY : L4_EOK;
00125     }
00126 };
00127
00137 L4_INLINE_RPC_OP(L4VIRTIO_OP_SET_STATUS, long,
00138                 set_status, (unsigned status));
00139
00155 L4_INLINE_RPC_OP(L4VIRTIO_OP_CONFIG_QUEUE, long,
00156                 config_queue, (unsigned queue));
00157
00181 L4_INLINE_RPC_OP(L4VIRTIO_OP_REGISTER_DS, long,
00182                 register_ds, (L4::Ipc::Cap<L4Re::Dataspace> ds_cap,
00183                               l4_uint64_t base, l4_umword_t offset,
00184                               l4_umword_t size));
00185
00194 L4_INLINE_RPC_OP(L4VIRTIO_OP_DEVICE_CONFIG, long, device_config,
00195                 (L4::Ipc::Out<L4::Cap<L4Re::Dataspace> > config_ds,
00196                  l4_addr_t *ds_offset));
00197
00216 L4_INLINE_RPC_OP(L4VIRTIO_OP_GET_DEVICE_IRQ, long, device_notification_irq,
00217                 (unsigned index, L4::Ipc::Out<L4::Cap<L4::Triggerable> > irq));
00218
00219 typedef L4::Typeid::Rpcs<set_status_t, config_queue_t, register_ds_t,
00220                         device_config_t, device_notification_irq_t>
00221     Rpcs;
00222 };
00223 };
00224
00225 }

```



## 17.252 l4virtio

```

00001 // vi:ft=cpp
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2014-2024 Kernkonzept GmbH.
00005  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00006  *           Manuel von Oltersdorff-Kalettkka <manuel.kalettkka@kernkonzept.com>
00007  */
00008 #pragma once
00009
00010 #include <algorithm>
00011 #include <limits.h>
00012 #include <memory>
00013 #include <vector>
00014
00015 #include <l4/re/dataspace>
00016 #include <l4/re/util/debug>
00017 #include <l4/re/env>
00018 #include <l4/re/error_helper>
00019 #include <l4/re/rm>
00020 #include <l4/re/util/cap_alloc>
00021 #include <l4/re/util/shared_cap>
00022 #include <l4/re/util/unique_cap>
00023
00024 #include <l4/sys/types.h>
00025 #include <l4/re/util/meta>
00026
00027 #include <l4/cxx/bitfield>
00028 #include <l4/cxx/utills>
00029 #include <l4/cxx/unique_ptr>
00030
00031 #include <l4/sys/cxx/ipc_legacy>
00032
00033 #include "../l4virtio"
00034 #include "virtio"
00035
00036 namespace L4virtio {
00037 namespace Svr {
00038
00039 class Dev_config
00040 {
00041 public:
00042     typedef Dev_status Status;
00043     typedef Dev_features Features;
00044 private:
00045     typedef L4Re::Rm::Unique_region<l4virtio_config_hdr_t*> Cfg_region;
00046     typedef L4Re::Util::Shared_cap<L4Re::Dataspace> Cfg_cap;
00047
00048     l4_uint32_t _vendor, _device, _qoffset, _nqueues;
00049     l4_uint32_t _host_features[sizeof(l4virtio_config_hdr_t::dev_features_map)
00050                               / sizeof(l4_uint32_t)];
00051
00052     Cfg_cap _ds;
00053     Cfg_region _config;
00054     l4_addr_t _ds_offset = 0;
00055
00056     Status _status{0}; // status shadow, can be trusted by the device model
00057
00058     static l4_uint32_t align(l4_uint32_t x)
00059     { return (x + 0xfU) & ~0xfU; }
00060
00061     void attach_n_init_cfg(Cfg_cap const &cfg, l4_addr_t offset)
00062     {
00063         L4Re::chksys(L4Re::Env::env()->rm()->attach(&_config, L4_PAGESIZE,
00064             L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00065             L4::Ipc::make_cap_rw(cfg.get()),
00066             offset),
00067             "Attach config space to local address space.");
00068
00069         _config->generation = 0;
00070         memset(_config->driver_features_map, 0, sizeof(_config->driver_features_map));
00071         memset(_host_features, 0, sizeof(_host_features));
00072         set_host_feature(L4VIRTIO_FEATURE_VERSION_1);
00073         reset_hdr();
00074
00075         _ds = cfg;
00076         _ds_offset = offset;
00077     }
00078
00079 protected:
00080     void volatile *get_priv_config() const
00081     {
00082         return l4virtio_device_config(_config.get());
00083     }
00084 }

```

```

00097
00098 public:
00099
00112 Dev_config(l4_uint32_t vendor, l4_uint32_t device,
00113            unsigned cfg_size, l4_uint32_t num_queues = 0)
00114 : _vendor(vendor), _device(device),
00115   _qoffset(0x100 + align(cfg_size)),
00116   _nqueues(num_queues)
00117 {
00118     using L4Re::Dataspace;
00119     using L4Re::chkcapi;
00120     using L4Re::chksys;
00121
00122     if (sizeof(l4virtio_config_queue_t) * _nqueues + _qoffset > L4_PAGESIZE)
00123     {
00124         // too many queues does not fit into our page
00125         _qoffset = 0;
00126         _nqueues = 0;
00127     }
00128
00129     auto cfg = chkcapi(L4Re::Util::make_shared_cap<Dataspace>());
00130     chksys(L4Re::Env::env()->mem_alloc()->alloc(L4_PAGESIZE, cfg.get()));
00131
00132     attach_n_init_cfg(cfg, 0);
00133 }
00134
00146 Dev_config(Cfg_cap const &cfg, l4_addr_t cfg_offset,
00147            l4_uint32_t vendor, l4_uint32_t device,
00148            unsigned cfg_size, l4_uint32_t num_queues = 0)
00149 : _vendor(vendor), _device(device),
00150   _qoffset(0x100 + align(cfg_size)),
00151   _nqueues(num_queues)
00152 {
00153     if (sizeof(l4virtio_config_queue_t) * _nqueues + _qoffset > L4_PAGESIZE)
00154     {
00155         // too many queues does not fit into our page
00156         _qoffset = 0;
00157         _nqueues = 0;
00158     }
00159
00160     attach_n_init_cfg(cfg, cfg_offset);
00161 }
00162
00163 void set_host_feature(unsigned feature)
00164 { l4virtio_set_feature(_host_features, feature); }
00165
00166 void clear_host_feature(unsigned feature)
00167 { l4virtio_clear_feature(_host_features, feature); }
00168
00169 bool get_host_feature(unsigned feature)
00170 { return l4virtio_get_feature(_host_features, feature); }
00171
00172 bool get_guest_feature(unsigned feature)
00173 { return l4virtio_get_feature(_config->driver_features_map, feature); }
00174
00175 l4_uint32_t &host_features(unsigned idx)
00176 { return _host_features[idx]; }
00177
00178 l4_uint32_t host_features(unsigned idx) const
00179 { return _host_features[idx]; }
00180
00184 l4_uint32_t num_queues() const
00185 { return _nqueues; }
00186
00198 l4_uint32_t guest_features(unsigned idx) const
00199 { return _config->driver_features_map[idx]; }
00200
00212 l4_uint32_t negotiated_features(unsigned idx) const
00213 { return _config->driver_features_map[idx] & _host_features[idx]; }
00214
00222 Status status() const { return _status; }
00223
00230 l4_uint32_t get_cmd() const
00231 {
00232     return hdr()->cmd;
00233 }
00234
00241 void reset_cmd()
00242 {
00243     const_cast<l4_uint32_t volatile &>(hdr()->cmd) = 0;
00244 }
00245
00253 void set_status(Status status)
00254 {
00255     _status = status;
00256     const_cast<l4_uint32_t volatile &>(hdr()->status) = status.raw;
00257 }

```

```

00258
00265 void add_irq_status(l4_uint32_t status)
00266 {
00267     const_cast<l4_uint32_t volatile &>(hdr()->irq_status) |= status;
00268 }
00269
00276 void set_device_needs_reset()
00277 {
00278     _status.device_needs_reset() = 1;
00279     const_cast<l4_uint32_t volatile &>(hdr()->status) = _status.raw;
00280 }
00281
00286 bool change_queue_config(l4_uint32_t num_queues)
00287 {
00288     if (sizeof(l4virtio_config_queue_t) * num_queues + _qoffset > L4_PAGESIZE)
00289         // too many queues does not fit into our page
00290         return false;
00291
00292     _nqueues = num_queues;
00293     reset_hdr(true);
00294     return true;
00295 }
00296
00303 l4virtio_config_queue_t volatile const *qconfig(unsigned index) const
00304 {
00305     if (L4_UNLIKELY(_qoffset < sizeof (l4virtio_config_hdr_t)))
00306         return 0;
00307
00308     if (L4_UNLIKELY(index >= _nqueues))
00309         return 0;
00310
00311     return reinterpret_cast<l4virtio_config_queue_t const *>
00312         (reinterpret_cast<char *>(_config.get()) + _qoffset) + index;
00313 }
00314
00318 void reset_hdr(bool inc_generation = false) const
00319 {
00320     _config->magic = L4VIRTIO_MAGIC;
00321     _config->version = 2;
00322     _config->device = _device;
00323     _config->vendor = _vendor;
00324     _config->status = 0;
00325     _config->irq_status = 0;
00326     _config->num_queues = _nqueues;
00327     _config->queues_offset = _qoffset;
00328
00329     memcpy(_config->dev_features_map, _host_features,
00330           sizeof(_config->dev_features_map));
00331     wmb();
00332     if (inc_generation)
00333         ++_config->generation;
00334 }
00335
00345 bool reset_queue(unsigned index, unsigned num_max,
00346                 bool inc_generation = false) const
00347 {
00348     l4virtio_config_queue_t volatile *qc;
00349     // this function is allowed to write to the device config
00350     qc = const_cast<l4virtio_config_queue_t volatile *>(qconfig(index));
00351     if (L4_UNLIKELY(qc == 0))
00352         return false;
00353
00354     qc->num_max = num_max;
00355     qc->num = 0;
00356     qc->ready = 0;
00357     wmb();
00358     if (inc_generation)
00359         ++_config->generation;
00360
00361     return true;
00362 }
00363
00368 l4virtio_config_hdr_t const volatile *hdr() const
00369 { return _config.get(); }
00370
00375 L4::Cap<L4Re::Dataspace> ds() const { return _ds.get(); }
00376
00381 l4_addr_t ds_offset() const
00382 { return _ds_offset; }
00383 };
00384
00385
00386 template<typename PRIV_CONFIG>
00387 class Dev_config_t : public Dev_config
00388 {
00389 public:

```

```

00391     typedef PRIV_CONFIG Priv_config;
00392
00404     Dev_config_t(l4_uint32_t vendor, l4_uint32_t device,
00405                 l4_uint32_t num_queues = 0)
00406     : Dev_config(vendor, device, sizeof(PRIV_CONFIG), num_queues)
00407     {}
00408
00419     Dev_config_t(L4Re::Util::Shared_cap<L4Re::Dataspace> const &cfg,
00420                 l4_addr_t cfg_offset, l4_uint32_t vendor, l4_uint32_t device,
00421                 l4_uint32_t num_queues = 0)
00422     : Dev_config(cfg, cfg_offset, vendor, device, sizeof(PRIV_CONFIG),
00423                 num_queues)
00424     {}
00425
00435     Priv_config volatile *priv_config() const
00436     {
00437         return static_cast<Priv_config volatile *>(get_priv_config());
00438     }
00439
00440 };
00441
00442 struct No_custom_data {};
00443
00449 template <typename DATA>
00450 class Driver_mem_region_t : public DATA
00451 {
00452 public:
00453     struct Flags
00454     {
00455         Flags() = default;
00456         explicit Flags(l4_uint32_t raw) : raw(raw) {}
00457
00458         l4_uint32_t raw;
00459         CXX_BITFIELD_MEMBER(0, 0, rw, raw);
00460     };
00461
00462 private:
00464     typedef L4Re::Util::Unique_cap<L4Re::Dataspace> Ds_cap;
00465
00466     l4_uint64_t _drv_base;
00467     l4_uint64_t _trans_offset;
00468     l4_umword_t _size;
00469     Flags      _flags;
00470
00471     Ds_cap      _ds;
00472     l4_addr_t   _ds_offset;
00473
00475     L4Re::Rm::Unique_region<l4_addr_t> _local_base;
00476
00477     template<typename T>
00478     T _local(l4_uint64_t addr) const
00479     {
00480         return reinterpret_cast<T>(addr - _trans_offset);
00481     }
00482
00483 public:
00485     Driver_mem_region_t() : _size(0) {}
00486
00498     Driver_mem_region_t(l4_uint64_t drv_base, l4_umword_t size,
00499                         l4_addr_t offset, Ds_cap &ds)
00500     : _drv_base(l4_trunc_page(drv_base)), _size(0), _flags(0),
00501       _ds_offset(l4_trunc_page(offset))
00502     {
00503         using L4Re::chksys;
00504         using L4Re::Env;
00505
00506         L4Re::Dataspace::Stats ds_info = L4Re::Dataspace::Stats();
00507         // Sometimes we work with dataspace that do not implement all dataspace
00508         // methods and return an error instead. An example of such a dataspace is
00509         // io's Vi::System_bus. We detect this case when the info method returns
00510         // -L4_ENOSYS and simply assume the dataspace is good for us.
00511         long err = ds->info(&ds_info);
00512         if (err >= 0)
00513         {
00514             l4_addr_t ds_size = l4_round_page(ds_info.size);
00515
00516             if (ds_size < L4_PAGESIZE)
00517                 chksys(-L4_EINVAL, "DS too small");
00518
00519             if (_ds_offset >= ds_size)
00520                 chksys(-L4_ERANGE, "offset larger than DS size");
00521
00522             size = l4_round_page(size);
00523             if (size > ds_size)
00524                 chksys(-L4_EINVAL, "size larger than DS size");
00525
00526             if (_ds_offset > ds_size - size)

```

```

00527         chksys(-L4_EINVAL, "invalid offset or size");
00528
00529         // overflow check
00530         if ((ULONG_MAX - size) < _drv_base)
00531             chksys(-L4_EINVAL, "invalid size");
00532
00533         _flags.rw() = (ds_info.flags & L4Re::Dataspace::F::W).raw != 0;
00534     }
00535     else if (err == -L4_ENOSYS)
00536     {
00537         _flags.rw() = true;
00538     }
00539     else
00540     {
00541         chksys(err, "getting data-space infos");
00542     }
00543
00544     auto f = L4Re::Rm::F::Search_addr | L4Re::Rm::F::R;
00545     if (_flags.rw())
00546         f |= L4Re::Rm::F::W;
00547
00548     // use a big alignment to save PT/TLB entries and kernel memory resources!
00549     chksys(Env::env()->rm()->attach(&_local_base, size, f,
00550                                     L4::Ipc::make_cap(ds.get(), _flags.rw()
00551                                                         ? L4_CAP_FPAGE_RW
00552                                                         : L4_CAP_FPAGE_RO),
00553                                     _ds_offset, L4_SUPERPAGESHIFT));
00554
00555     _size = size;
00556     _ds = cxx::move(ds);
00557     _trans_offset = _drv_base - _local_base.get();
00558 }
00559
00560 bool is_writable() const { return _flags.rw(); }
00561
00562 Flags flags() const { return _flags; }
00563
00564 bool empty() const
00565 { return _size == 0; }
00566
00567 l4_uint64_t drv_base() const { return _drv_base; }
00568
00569 l4_addr_t local_base() const { return _local_base.get(); }
00570
00571 l4_umword_t size() const { return _size; }
00572
00573 l4_addr_t ds_offset() const { return _ds_offset; }
00574
00575 L4::Cap<L4Re::Dataspace> ds() const { return _ds.get(); }
00576
00577 bool contains(l4_uint64_t base, l4_umword_t size) const
00578 {
00579     if (base < _drv_base)
00580         return false;
00581
00582     if (base > _drv_base + _size - 1)
00583         return false;
00584
00585     if (size > _size)
00586         return false;
00587
00588     if (base - _drv_base > _size - size)
00589         return false;
00590
00591     return true;
00592 }
00593
00594 template<typename T>
00595 T *local(Ptr<T> p) const
00596 { return _local<T*>(p.get()); }
00597 };
00598
00599 typedef Driver_mem_region_t<No_custom_data> Driver_mem_region;
00600
00601 template <typename DATA>
00602 class Driver_mem_list_t
00603 {
00604 public:
00605     typedef Driver_mem_region_t<DATA> Mem_region;
00606
00607 private:
00608     cxx::unique_ptr<Mem_region[]> _l;
00609     unsigned _max;
00610     unsigned _free;
00611
00612 public:
00613     typedef L4Re::Util::Unique_cap<L4Re::Dataspace> Ds_cap;

```

```

00642
00644 Driver_mem_list_t() : _max(0), _free(0) {}
00645
00650 void init(unsigned max)
00651 {
00652     _l = cxx::make_unique<Driver_mem_region_t<DATA>[]>(max);
00653     _max = max;
00654     _free = 0;
00655 }
00656
00658 bool full() const
00659 { return _free == _max; }
00660
00669 Mem_region const *add(l4_uint64_t drv_base, l4_umword_t size,
00670                       l4_addr_t offset, Ds_cap &&ds)
00671 {
00672     if (full())
00673         L4Re::chksys(-L4_ENOMEM);
00674
00675     _l[_free++] = Mem_region(drv_base, size, offset, cxx::move(ds));
00676     return &_l[_free - 1];
00677 }
00678
00683 void remove(Mem_region const *r)
00684 {
00685     if (r < &_l[0] || r >= &_l[_free])
00686         L4Re::chksys(-L4_ERANGE);
00687
00688     unsigned idx = r - &_l[0];
00689
00690     for (unsigned i = idx + 1; i < _free - 1; ++i)
00691         _l[i] = cxx::move(_l[i + 1]);
00692
00693     _l[--_free] = Mem_region();
00694 }
00695
00703 Mem_region *find(l4_uint64_t base, l4_umword_t size) const
00704 {
00705     return _find(base, size);
00706 }
00707
00717 void load_desc(Virtqueue::Desc const &desc, Request_processor const *p,
00718               Virtqueue::Desc const **table) const
00719 {
00720     Mem_region const *r = find(desc.addr.get(), desc.len);
00721     if (L4_UNLIKELY(!r))
00722         throw Bad_descriptor(p, Bad_descriptor::Bad_address);
00723
00724     *table = static_cast<Virtqueue::Desc const *>(r->local(desc.addr));
00725 }
00726
00737 void load_desc(Virtqueue::Desc const &desc, Request_processor const *p,
00738               Mem_region const **data) const
00739 {
00740     Mem_region const *r = find(desc.addr.get(), desc.len);
00741     if (L4_UNLIKELY(!r))
00742         throw Bad_descriptor(p, Bad_descriptor::Bad_address);
00743
00744     *data = r;
00745 }
00746
00763 template<typename ARG>
00764 void load_desc(Virtqueue::Desc const &desc, Request_processor const *p,
00765               ARG *data) const
00766 {
00767     Mem_region *r = find(desc.addr.get(), desc.len);
00768     if (L4_UNLIKELY(!r))
00769         throw Bad_descriptor(p, Bad_descriptor::Bad_address);
00770
00771     *data = ARG(r, desc, p);
00772 }
00773
00774 Mem_region *begin() { return &_l[0]; }
00775 Mem_region const *begin() const { return &_l[0]; }
00776
00777 Mem_region *end() { return &_l[_free]; }
00778 Mem_region const *end() const { return &_l[_free]; }
00779
00780 private:
00781 Mem_region *_find(l4_uint64_t base, l4_umword_t size) const
00782 {
00783     for (unsigned i = 0; i < _free; ++i)
00784         if (_l[i].contains(base, size))
00785             return &_l[i];
00786     return 0;
00787 }
00788

```

```

00789
00790 };
00791
00792 typedef Driver_mem_list_t<No_custom_data> Driver_mem_list;
00793
00800 template<typename DATA>
00801 class Device_t
00802 {
00803 public:
00804     typedef Driver_mem_list_t<DATA> Mem_list;
00805
00806 protected:
00807     Mem_list _mem_info;
00808
00809 private:
00810     Dev_config *_device_config;
00811
00812     using Ds_vector = std::vector<L4::Cap<L4Re::Dataspace>;
00814     std::shared_ptr<Ds_vector const> _trusted_ds_caps;
00815
00817     bool _trusted_ds_validation_enabled = false;
00818
00819 public:
00820     L4_RPC_LEGACY_DISPATCH(L4virtio::Device);
00821     template<typename IOS> int virtio_dispatch(unsigned r, IOS &ios)
00822     { return dispatch(r, ios); }
00823
00825     virtual void reset() = 0;
00826
00828     virtual bool check_features()
00829     { return true; }
00830
00832     virtual bool check_queues() = 0;
00833
00835     virtual int reconfig_queue(unsigned idx) = 0;
00836
00838     virtual void cfg_changed(unsigned /* reg */) {};
00839
00841     virtual void register_single_driver_irq()
00842     { L4Re::chksys(-L4_ENOSYS, "Legacy single IRQ interface not implemented."); }
00843
00845     virtual void trigger_driver_config_irq() = 0;
00846
00848     virtual L4::Cap<L4::Irq> device_notify_irq() const
00849     {
00850         L4Re::chksys(-L4_ENOSYS, "Legacy single IRQ interface not implemented.");
00851         return L4::Cap<L4::Irq>();
00852     }
00853
00860     virtual void register_driver_irq(unsigned idx)
00861     {
00862         if (idx != 0)
00863             L4Re::chksys(-L4_ENOSYS, "Multi IRQ interface not implemented.");
00864
00865         register_single_driver_irq();
00866     }
00867
00874     virtual L4::Cap<L4::Irq> device_notify_irq(unsigned idx)
00875     {
00876         if (idx != 0)
00877             L4Re::chksys(-L4_ENOSYS, "Multi IRQ interface not implemented.");
00878
00879         return device_notify_irq();
00880     }
00881
00883     virtual unsigned num_events_supported() const
00884     { return 1; }
00885
00886     virtual L4::Ipc_svr::Server_iface *server_iface() const = 0;
00887
00891     Device_t(Dev_config *dev_config)
00892     : _device_config(dev_config)
00893     {}
00894
00898     Mem_list const *mem_info() const
00899     { return &_mem_info; };
00900
00901     long op_set_status(L4virtio::Device::Rights, unsigned status)
00902     { return _set_status(status); }
00903
00904     long op_config_queue(L4virtio::Device::Rights, unsigned queue)
00905     {
00906         Dev_config::Status status = _device_config->status();
00907         if (status.fail_state() || !status.acked() || !status.driver())
00908             return -L4_EIO;
00909
00910         return reconfig_queue(queue);

```

```

00911     }
00912
00913     long op_register_ds(L4virtio::Device::Rights,
00914                        L4::Ipc::Snd_fpage ds_cap_fp, l4_uint64_t ds_base,
00915                        l4_umword_t offset, l4_umword_t sz)
00916     {
00917         L4Re::Util::Dbg()
00918             .printf("Registering dataspace from 0x%llx with %lu KiB, offset 0x%lx\n",
00919                    ds_base, sz » 10, offset);
00920
00921         _check_n_init_shm(ds_cap_fp, ds_base, sz, offset);
00922
00923         return 0;
00924     }
00925
00926     long op_device_config(L4virtio::Device::Rights,
00927                          L4::Ipc::Cap<L4Re::Dataspace> &config_ds,
00928                          l4_addr_t &ds_offset)
00929     {
00930         L4Re::Util::Dbg()
00931             .printf("register client: host IRQ: %lx config DS: %lx\n",
00932                    device_notify_irq().cap(), _device_config->ds().cap());
00933
00934         config_ds = L4::Ipc::make_cap(_device_config->ds(), L4_CAP_FPAGE_RW);
00935         ds_offset = _device_config->ds_offset();
00936         return 0;
00937     }
00938
00939     long op_device_notification_irq(L4virtio::Device::Rights,
00940                                     unsigned idx,
00941                                     L4::Ipc::Cap<L4::Triggerable> &irq)
00942     {
00943         auto cap = device_notify_irq(idx);
00944
00945         if (!cap.is_valid())
00946             return -L4_EINVAL;
00947
00948         irq = L4::Ipc::make_cap(cap, L4_CAP_FPAGE_RO);
00949         return L4_EOK;
00950     }
00951
00952     int op_bind(L4::Icu::Rights, l4_umword_t idx, L4::Ipc::Snd_fpage irq_cap_fp)
00953     {
00954         if (idx >= num_events_supported())
00955             return -L4_ERANGE;
00956
00957         if (!irq_cap_fp.cap_received())
00958             return -L4_EINVAL;
00959
00960         register_driver_irq(idx);
00961
00962         return L4_EOK;
00963     }
00964
00965     int op_unbind(L4::Icu::Rights, l4_umword_t, L4::Ipc::Snd_fpage)
00966     {
00967         return -L4_ENOSYS;
00968     }
00969
00970     int op_info(L4::Icu::Rights, L4::Icu::_Info &info)
00971     {
00972         info.features = 0;
00973         info.nr_irqs = num_events_supported();
00974         info.nr_msis = 0;
00975
00976         return L4_EOK;
00977     }
00978
00979     int op_msi_info(L4::Icu::Rights, l4_umword_t, l4_uint64_t, l4_icu_msi_info_t &)
00980     { return -L4_ENOSYS; }
00981
00982     int op_mask(L4::Icu::Rights, l4_umword_t)
00983     { return -L4_ENOSYS; }
00984
00985     int op_unmask(L4::Icu::Rights, l4_umword_t)
00986     { return -L4_ENOREPLY; }
00987
00988     int op_set_mode(L4::Icu::Rights, l4_umword_t, l4_umword_t)
00989     { return -L4_ENOSYS; }
00990
01002     void reset_queue_config(unsigned idx, unsigned num_max,
01003                             bool inc_generation = false)
01004     {
01005         _device_config->reset_queue(idx, num_max, inc_generation);
01006     }
01007
01012     void init_mem_info(unsigned num)

```



```

01013 {
01014     _mem_info.init(num);
01015 }
01016
01024 void device_error()
01025 {
01026     reset();
01027     _device_config->set_device_needs_reset();
01028
01029     // the device MUST NOT notify the driver before DRIVER_OK.
01030     if (_device_config->status().driver_ok())
01031         trigger_driver_config_irq();
01032 }
01033
01047 bool setup_queue(Virtqueue *q, unsigned qn, unsigned num_max)
01048 {
01049     l4virtio_config_queue_t volatile const *qc;
01050     qc = _device_config->qconfig(qn);
01051     if (L4_UNLIKELY(qc == 0))
01052         return false;
01053
01054     if (!qc->ready)
01055     {
01056         q->disable();
01057         return true;
01058     }
01059
01060     // read to local variables before check
01061     l4_uint32_t num = qc->num;
01062     l4_uint64_t desc = qc->desc_addr;
01063     l4_uint64_t avail = qc->avail_addr;
01064     l4_uint64_t used = qc->used_addr;
01065
01066     if (0)
01067         printf("%p: setup queue: num=0x%x max_num=0x%x desc=0x%llx avail=0x%llx used=0x%llx\n",
01068             this, num, num_max, desc, avail, used);
01069
01070     if (!num || num > num_max)
01071         return false;
01072
01073     // num must be power of two
01074     if (num & (num - 1))
01075         return false;
01076
01077     if (desc & 0xf)
01078         return false;
01079
01080     if (avail & 0x1)
01081         return false;
01082
01083     if (used & 0x3)
01084         return false;
01085
01086     auto const *desc_info = _mem_info.find(desc, Virtqueue::desc_size(num));
01087     if (L4_UNLIKELY(!desc_info))
01088         return false;
01089
01090     auto const *avail_info = _mem_info.find(avail, Virtqueue::avail_size(num));
01091     if (L4_UNLIKELY(!avail_info))
01092         return false;
01093
01094     auto const *used_info = _mem_info.find(used, Virtqueue::used_size(num));
01095     if (L4_UNLIKELY(!used_info || !used_info->is_writable()))
01096         return false;
01097
01098     L4Re::Util::Dbg()
01099         .printf("shm=[%llx-%llx] local=[%lx-%lx] desc=[%llx-%llx] (%p-%p)\n",
01100             desc_info->drv_base(), desc_info->drv_base() + desc_info->size() - 1,
01101             desc_info->local_base(),
01102             desc_info->local_base() + desc_info->size() - 1,
01103             desc, desc + Virtqueue::desc_size(num),
01104             desc_info->local(Ptr<char>(desc)),
01105             desc_info->local(Ptr<char>(desc)) + Virtqueue::desc_size(num));
01106
01107     L4Re::Util::Dbg()
01108         .printf("shm=[%llx-%llx] local=[%lx-%lx] avail=[%llx-%llx] (%p-%p)\n",
01109             avail_info->drv_base(), avail_info->drv_base() + avail_info->size() - 1,
01110             avail_info->local_base(),
01111             avail_info->local_base() + avail_info->size() - 1,
01112             avail, avail + Virtqueue::avail_size(num),
01113             avail_info->local(Ptr<char>(avail)),
01114             avail_info->local(Ptr<char>(avail)) + Virtqueue::avail_size(num));
01115
01116     L4Re::Util::Dbg()
01117         .printf("shm=[%llx-%llx] local=[%lx-%lx] used=[%llx-%llx] (%p-%p)\n",
01118             used_info->drv_base(), used_info->drv_base() + used_info->size() - 1,
01119             used_info->local_base(),

```

```

01120         used_info->local_base() + used_info->size() - 1,
01121         used, used + Virtqueue::used_size(num),
01122         used_info->local(Ptr<char>(used)),
01123         used_info->local(Ptr<char>(used)) + Virtqueue::used_size(num));
01124
01125     q->setup(num, desc_info->local(Ptr<void>(desc)),
01126             avail_info->local(Ptr<void>(avail)),
01127             used_info->local(Ptr<void>(used)));
01128     return true;
01129 }
01130
01131 void check_n_init_shm(L4Re::Util::Unique_cap<L4Re::Dataspace> &&shm,
01132                      l4_uint64_t base, l4_umword_t size, l4_addr_t offset)
01133 {
01134     if (_mem_info.full())
01135         L4Re::chksys(-L4_ENOMEM);
01136
01137     auto const *i = _mem_info.add(base, size, offset, cxx::move(shm));
01138     L4Re::Util::Dbg()
01139         .printf("PORT[%p]: DMA guest [%llx-%llx] local [%lx-%lx] offset %lx\n",
01140                this, i->drv_base(), i->drv_base() + i->size() - 1,
01141                i->local_base(),
01142                i->local_base() + i->size() - 1,
01143                i->ds_offset());
01144 }
01145
01154 bool handle_mem_cmd_write()
01155 {
01156     l4_uint32_t cmd = _device_config->get_cmd();
01157     if (L4_LIKELY(!(cmd & L4VIRTIO_CMD_MASK)))
01158         return false;
01159
01160     switch (cmd & L4VIRTIO_CMD_MASK)
01161     {
01162     case L4VIRTIO_CMD_SET_STATUS:
01163         _set_status(cmd & ~L4VIRTIO_CMD_MASK);
01164         break;
01165
01166     case L4VIRTIO_CMD_CFG_QUEUE:
01167         reconfig_queue(cmd & ~L4VIRTIO_CMD_MASK);
01168         break;
01169
01170     case L4VIRTIO_CMD_CFG_CHANGED:
01171         cfg_changed(cmd & ~L4VIRTIO_CMD_MASK);
01172         break;
01173
01174     default:
01175         // unknown command
01176         break;
01177     }
01178
01179     _device_config->reset_cmd();
01180
01181     return true;
01182 }
01183
01187 void enable_trusted_ds_validation()
01188 {
01189     _trusted_ds_validation_enabled = true;
01190 }
01191
01197 void
01198 add_trusted_dataspaces(std::shared_ptr<Ds_vector const> ds)
01199 {
01200     _trusted_ds_caps = ds;
01201 }
01202
01203
01204 private:
01216 long validate_ds(L4::Cap<L4Re::Dataspace> ds)
01217 {
01218     if (!_trusted_ds_caps)
01219         return -L4_EINVAL;
01220     if (std::any_of(_trusted_ds_caps->cbegin(), _trusted_ds_caps->cend(),
01221                    [&ds] (L4::Cap<L4Re::Dataspace> cap)
01222                    {
01223                        return L4Re::Env::env()->task()
01224                            ->cap_equal(ds, cap).label() == 1;
01225                    })
01226     )
01227     {
01228         return L4_EOK;
01229     }
01230     return -L4_EINVAL;
01231 }
01232
01233 void _check_n_init_shm(L4::Ipc::Snd_fpage shm_cap_fp,

```

```

01234         l4_uint64_t base, l4_umword_t size, l4_addr_t offset)
01235     {
01236         if (!shm_cap_fp.cap_received())
01237             L4Re::chksys(-L4_EINVAL);
01238
01239         L4Re::Util::Unique_cap<L4Re::Dataspace> ds(
01240             L4Re::chkcapi(server_iface()->template rcv_cap<L4Re::Dataspace>(0)));
01241         L4Re::chksys(server_iface()->realloc_rcv_cap(0));
01242
01243         if (_trusted_ds_validation_enabled)
01244             L4Re::chksys(validate_ds(ds.get()), "Validating the dataspace.");
01245
01246         check_n_init_shm(cxx::move(ds), base, size, offset);
01247     }
01248
01249     bool check_features_internal()
01250     {
01251         static_assert(sizeof(l4virtio_config_hdr_t::driver_features_map)
01252             == sizeof(l4virtio_config_hdr_t::dev_features_map),
01253             "Driver and device feature maps must be of the same size");
01254
01255         // From the Virtio 1.0 specification 6.1 Driver Requirements and 6.2 Device
01256         // Requirements: A driver MUST accept VIRTIO_F_VERSION_1 if it is offered.
01257         // A device MUST offer VIRTIO_F_VERSION_1. A device MAY fail to operate
01258         // further if VIRTIO_F_VERSION_1 is not accepted.
01259         //
01260         // The L4virtio implementation does not support legacy interfaces so we
01261         // fail here if the Virtio 1.0 feature was not accepted.
01262         if (!_device_config->get_guest_feature(L4VIRTIO_FEATURE_VERSION_1))
01263             return false;
01264
01265         for (auto i = 0u;
01266             i < sizeof(l4virtio_config_hdr_t::driver_features_map)
01267                 / sizeof(l4virtio_config_hdr_t::driver_features_map[0]);
01268             i++)
01269         {
01270             // Driver must not accept features that were not offered by device
01271             if (_device_config->guest_features(i)
01272                 & ~_device_config->host_features(i))
01273                 return false;
01274         }
01275         return check_features();
01276     }
01277
01299     void check_and_update_status(Dev_config::Status status)
01300     {
01301         // snapshot of current status
01302         Dev_config::Status current_status = _device_config->status();
01303
01304         // handle reset
01305         if (!status.raw)
01306         {
01307             _device_config->set_status(status);
01308             return;
01309         }
01310
01311         // Do no further processing in case of driver or device failure. If FAILED
01312         // or DEVICE_NEEDS_RESET are set only these fail_state bits will be set in
01313         // addition to the current status bits already set.
01314         if (current_status.fail_state() || status.fail_state())
01315         {
01316             if (current_status.fail_state() != status.fail_state())
01317             {
01318                 current_status.fail_state() =
01319                     current_status.fail_state() | status.fail_state();
01320                 _device_config->set_status(current_status);
01321             }
01322             return;
01323         }
01324
01325         // Enforce init sequence ACKNOWLEDGE, DRIVER, FEATURES_OK, DRIVER_OK.
01326         // We do not enforce that only one additional new bit is set per call.
01327         if ((!status.acked() && status.driver())
01328             || (!status.driver() && status.features_ok())
01329             || (!status.features_ok() && status.driver_ok()))
01330         {
01331             current_status.device_needs_reset() = 1;
01332             _device_config->set_status(current_status);
01333             return;
01334         }
01335
01336         // only check feature compatibility before DRIVER_OK is set
01337         if (status.features_ok() && !status.driver_ok()
01338             && !check_features_internal())
01339             status.features_ok() = 0;
01340
01341         // Note that if FEATURES_OK and DRIVER_OK are both updated to being set

```

```

01342 // at the same time the above check_features_internal() is skipped; this is
01343 // considered undefined behaviour but it is not prevented.
01344 if (status.running() && !check_queues())
01345 {
01346     current_status.device_needs_reset() = 1;
01347     _device_config->set_status(current_status);
01348     return;
01349 }
01350
01351 _device_config->set_status(status);
01352 }
01353
01354 int _set_status(unsigned new_status)
01355 {
01356     if (new_status == 0)
01357     {
01358         L4Re::Util::Dbg().printf("Resetting device\n");
01359         reset();
01360         _device_config->reset_hdr(true);
01361     }
01362
01363     Dev_config::Status status(new_status);
01364     check_and_update_status(status);
01365
01366     return 0;
01367 }
01368
01369 };
01370
01371 typedef Device_t<No_custom_data> Device;
01372
01373 } // namespace Svr
01374
01375 }

```

## 17.253 virtio

```

00001 // vi:ft=cpp
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2014-2020, 2023-2024 Kernkonzept GmbH.
00005  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00006  */
00007 */
00008 #pragma once
00009
00010 #include <l4/sys/types.h>
00011 #include <l4/cxx/bitfield>
00012 #include <l4/cxx/minmax>
00013 #include <l4/cxx/utils>
00014
00015 #include <limits.h>
00016 #include <string.h>
00017 #include <stdio.h>
00018
00019 #include "../virtqueue"
00020
00021 namespace L4virtio {
00022 namespace Svr {
00023
00024 struct Dev_status
00025 {
00026     unsigned char raw;
00027     Dev_status() = default;
00028
00029     explicit Dev_status(l4_uint32_t v) : raw(v) {}
00030
00031     CXX_BITFIELD_MEMBER(0, 0, acked, raw);
00032     CXX_BITFIELD_MEMBER(1, 1, driver, raw);
00033     CXX_BITFIELD_MEMBER(2, 2, driver_ok, raw);
00034     CXX_BITFIELD_MEMBER(3, 3, features_ok, raw);
00035     CXX_BITFIELD_MEMBER(6, 7, fail_state, raw);
00036     CXX_BITFIELD_MEMBER(6, 6, device_needs_reset, raw);
00037     CXX_BITFIELD_MEMBER(7, 7, failed, raw);
00038
00039     bool running() const
00040     {
00041         return (raw == 0xf);
00042     }
00043 };
00044
00045 struct Dev_features
00046 {

```

```

00068     l4_uint32_t raw;
00069     Dev_features() = default;
00070
00072     explicit Dev_features(l4_uint32_t v) : raw(v) {}
00073
00074     CXX_BITFIELD_MEMBER(28, 28, ring_indirect_desc, raw);
00075     CXX_BITFIELD_MEMBER(29, 29, ring_event_idx, raw);
00076 };
00077
00078
00087 class Virtqueue : public L4virtio::Virtqueue
00088 {
00089 public:
00093     class Head_desc
00094     {
00095     friend class Virtqueue;
00096     private:
00097         Virtqueue::Desc const *_d;
00098         Head_desc(Virtqueue *r, unsigned i) : _d(r->desc(i)) {}
00099
00100     public:
00102         Head_desc() : _d(0) {}
00103
00105         bool valid() const { return _d; }
00106
00108         explicit operator bool () const
00109         { return valid(); }
00110
00112         Desc const *desc() const
00113         { return _d; }
00114     };
00115
00116     struct Request : Head_desc
00117     {
00118         Virtqueue *ring = nullptr;
00119         Request() = default;
00120     private:
00121         friend class Virtqueue;
00122         Request(Virtqueue *r, unsigned i) : Head_desc(r, i), ring(r) {}
00123     };
00124
00125
00136     Request next_avail()
00137     {
00138         if (L4_LIKELY(_current_avail != _avail->idx))
00139         {
00140             rmb();
00141             unsigned head = _current_avail & _idx_mask;
00142             ++_current_avail;
00143             return Request(this, _avail->ring[head]);
00144         }
00145         return Request();
00146     }
00147
00160     void rewind_avail(Head_desc const &d)
00161     {
00162         unsigned head_idx = d._d - _desc;
00163         // Calculate the distance between _current_avail and head_idx, taking into
00164         // account that _current_avail might have wrapped around with respect to
00165         // _idx_mask in the meantime.
00166         _current_avail -= (_current_avail - head_idx) & _idx_mask;
00167     }
00168
00175     bool desc_avail() const
00176     {
00177         return _current_avail != _avail->idx;
00178     }
00179
00190     void consumed(Head_desc const &r, l4_uint32_t len = 0)
00191     {
00192         l4_uint16_t i = _used->idx & _idx_mask;
00193         _used->ring[i] = Used_elem(r._d - _desc, len);
00194         wmb();
00195         ++_used->idx;
00196     }
00197
00212     template<typename ITER>
00213     void consumed(ITER const &begin, ITER const &end)
00214     {
00215         l4_uint16_t added = 0;
00216         l4_uint16_t idx = _used->idx;
00217
00218         for (auto elem = begin ; elem != end; ++elem, ++added)
00219             _used->ring[(idx + added) & _idx_mask]
00220                 = Used_elem(elem->first._d - _desc, elem->second);
00221
00222         wmb();

```

```

00223     __used->idx += added;
00224 }
00225
00239 template<typename QUEUE_OBSERVER>
00240 void finish(Head_desc &d, QUEUE_OBSERVER *o, l4_uint32_t len = 0)
00241 {
00242     consumed(d, len);
00243     o->notify_queue(this);
00244     d._d = 0;
00245 }
00246
00261 template<typename ITER, typename QUEUE_OBSERVER>
00262 void finish(ITER const &begin, ITER const &end, QUEUE_OBSERVER *o)
00263 {
00264     consumed(begin, end);
00265     o->notify_queue(this);
00266 }
00267
00273 void disable_notify()
00274 {
00275     if (L4_LIKELY(ready()))
00276         __used->flags.no_notify() = 1;
00277 }
00278
00284 void enable_notify()
00285 {
00286     if (L4_LIKELY(ready()))
00287         __used->flags.no_notify() = 0;
00288 }
00289
00298 Desc const *desc(unsigned idx) const
00299 { return __desc + idx; }
00300
00301 };
00302
00306 struct Data_buffer
00307 {
00308     char *pos;
00309     l4_uint32_t left;
00310
00311     Data_buffer() = default;
00312
00322 template<typename T>
00323 explicit Data_buffer(T *p)
00324 : pos(reinterpret_cast<char *>(p)), left(sizeof(T))
00325 {}
00326
00336 template<typename T>
00337 void set(T *p)
00338 {
00339     pos = reinterpret_cast<char *>(p);
00340     left = sizeof(T);
00341 }
00342
00354 l4_uint32_t copy_to(Data_buffer *dst, l4_uint32_t max = UINT_MAX)
00355 {
00356     unsigned long bytes = cxx::min(cxx::min(left, dst->left), max);
00357     memcpy(dst->pos, pos, bytes);
00358     left -= bytes;
00359     pos += bytes;
00360     dst->left -= bytes;
00361     dst->pos += bytes;
00362     return bytes;
00363 }
00364
00375 l4_uint32_t skip(l4_uint32_t bytes)
00376 {
00377     unsigned long b = cxx::min(left, bytes);
00378     left -= b;
00379     pos += b;
00380     return b;
00381 }
00382
00388 bool done() const
00389 { return left == 0; }
00390 };
00391
00392 class Request_processor;
00393
00397 struct Bad_descriptor
00398 {
00400     enum Error
00401     {
00402         Bad_address,
00403         Bad_rights,
00404         Bad_flags,
00405         Bad_next,

```

```

00406     Bad_size
00407 };
00408
00410 Request_processor const *proc;
00411
00412 // The error code
00413 Error error;
00414
00421 Bad_descriptor(Request_processor const *proc, Error e)
00422 : proc(proc), error(e)
00423 {}
00424
00430 char const *message() const
00431 {
00432     static char const *const err[] =
00433     {
00434         [Bad_address] = "Descriptor address cannot be translated",
00435         [Bad_rights]   = "Insufficient memory access rights",
00436         [Bad_flags]    = "Invalid descriptor flags",
00437         [Bad_next]     = "The descriptor's `next` index is invalid",
00438         [Bad_size]     = "Invalid size of the memory block"
00439     };
00440
00441     if (error >= (sizeof(err) / sizeof(err[0])) || !err[error])
00442         return "Unknown error";
00443
00444     return err[error];
00445 }
00446 };
00447
00448
00472 class Request_processor
00473 {
00474 private:
00476     Virtqueue::Desc const *_table;
00477
00479     Virtqueue::Desc _current;
00480
00482     l4_uint16_t _num;
00483
00484 public:
00500     template<typename DESC_MAN, typename ...ARGS>
00501     void start(DESC_MAN *dm, Virtqueue *ring, Virtqueue::Head_desc const &request, ARGS... args)
00502     {
00503         _current = cxx::access_once(request.desc());
00504
00505         if (_current.flags.indirect())
00506         {
00507             dm->load_desc(_current, this, &_table);
00508             _num = _current.len / sizeof(Virtqueue::Desc);
00509             if (L4_UNLIKELY(!_num))
00510                 throw Bad_descriptor(this, Bad_descriptor::Bad_size);
00511
00512             _current = cxx::access_once(_table);
00513         }
00514         else
00515         {
00516             _table = ring->desc(0);
00517             _num = ring->num();
00518         }
00519
00520         dm->load_desc(_current, this, cxx::forward<ARGS>(args)...);
00521     }
00522
00533     template<typename DESC_MAN, typename ...ARGS>
00534     Virtqueue::Request const &start(DESC_MAN *dm, Virtqueue::Request const &request, ARGS... args)
00535     {
00536         start(dm, request.ring, request, cxx::forward<ARGS>(args)...);
00537         return request;
00538     }
00539
00545     Virtqueue::Desc::Flags current_flags() const
00546     { return _current.flags; }
00547
00553     bool has_more() const
00554     { return _current.flags.next(); }
00555
00569     template<typename DESC_MAN, typename ...ARGS>
00570     bool next(DESC_MAN *dm, ARGS... args)
00571     {
00572         if (!_current.flags.next())
00573             return false;
00574
00575         if (L4_UNLIKELY(_current.next >= _num))
00576             throw Bad_descriptor(this, Bad_descriptor::Bad_next);
00577
00578         _current = cxx::access_once(_table + _current.next);

```

```

00579
00580     if (0) // we ignore this for performance reasons
00581         if (L4_UNLIKELY(_current.flags.indirect()))
00582             throw Bad_descriptor(this, Bad_descriptor::Bad_flags);
00583
00584     // must throw an exception in case of a bad descriptor
00585     dm->load_desc(_current, this, cxx::forward<ARGS>(args)...);
00586     return true;
00587 }
00588 };
00589
00590 }
00591 }

```

## 17.254 virtio-block

```

00001 // vi:ft=cpp
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2015-2022, 2024 Kernkonzept GmbH.
00005  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00006  *           Manuel von Oltersdorff-Kalettka <manuel.kalettka@kernkonzept.com>
00007  *
00008  */
00009 #pragma once
00010
00011 #include <l4/sys/factory>
00012 #include <l4/sys/semaphore>
00013 #include <l4/re/dataspace>
00014 #include <l4/re/env>
00015 #include <l4/re/util/unique_cap>
00016 #include <l4/re/util/object_registry>
00017 #include <l4/re/error_helper>
00018
00019 #include <l4/util/atomic.h>
00020 #include <l4/util/bitops.h>
00021 #include <l4/l4virtio/client/l4virtio>
00022 #include <l4/l4virtio/l4virtio>
00023 #include <l4/l4virtio/virtqueue>
00024 #include <l4/l4virtio/virtio_block.h>
00025 #include <l4/sys/consts.h>
00026
00027 #include <cstring>
00028 #include <vector>
00029 #include <functional>
00030
00031 namespace L4virtio { namespace Driver {
00032
00036 class Block_device : public Device
00037 {
00038 public:
00039     typedef std::function<void(unsigned char)> Callback;
00040
00041 private:
00042     enum { Header_size = sizeof(l4virtio_block_header_t) };
00043
00044     struct Request
00045     {
00046         l4_uint16_t tail;
00047         Callback callback;
00048
00049         Request() : tail(Virtqueue::Eoq), callback(0) {}
00050     };
00051
00052 public:
00056     class Handle
00057     {
00058     {
00059         friend Block_device;
00060         l4_uint16_t head;
00061
00062         explicit Handle(l4_uint16_t descno) : head(descno) {}
00063
00064     public:
00065         Handle() : head(Virtqueue::Eoq) {}
00066         bool valid() const { return head != Virtqueue::Eoq; }
00067     };
00068
00092 void setup_device(L4::Cap<L4virtio::Device> srvcap, l4_size_t usermem,
00093                 void **userdata, Ptr<void> &user_devaddr,
00094                 L4::Cap<L4Re::Dataspace> qds = L4::Cap<L4Re::Dataspace>(),
00095                 l4_uint32_t fmask0 = -1U, l4_uint32_t fmask1 = -1U)
00096 {
00097     // Contact device.

```



```

00098     driver_connect(srvcap);
00099
00100     if (_config->device != L4VIRTIO_ID_BLOCK)
00101         L4Re::chksys(-L4_ENODEV, "Device is not a block device.");
00102
00103     if (_config->num_queues != 1)
00104         L4Re::chksys(-L4_EINVAL, "Invalid number of queues reported.");
00105
00106     // Memory is shared in one large dataspace which contains queues,
00107     // space for header/status and additional user-defined memory.
00108     unsigned queuesz = max_queue_size(0);
00109     l4_size_t totalsz = l4_round_page(usermem);
00110
00111     l4_uint64_t const header_offset =
00112         l4_round_size(_queue.total_size(queuesz),
00113             l4util_bsr(aligned(l4virtio_block_header_t)));
00114     l4_uint64_t const status_offset = header_offset + queuesz * Header_size;
00115     l4_uint64_t const usermem_offset = l4_round_page(status_offset + queuesz);
00116
00117     // reserve space for one header/status per descriptor
00118     // TODO Should be reduced to 1/3 but this way no freelist is needed.
00119     totalsz += usermem_offset;
00120
00121     auto *e = L4Re::Env::env();
00122     if (!qds.is_valid())
00123     {
00124         _ds = L4Re::chkcap(L4Re::Util::make_unique_cap<L4Re::Dataspace>(),
00125             "Allocate queue dataspace capability");
00126         L4Re::chksys(e->mem_alloc()->alloc(totalsz, _ds.get(),
00127             L4Re::Mem_alloc::Continuous
00128             | L4Re::Mem_alloc::Pinned),
00129             "Allocate memory for virtio structures");
00130         _queue_ds = _ds.get();
00131     }
00132     else
00133     {
00134         if (qds->size() < totalsz)
00135             L4Re::chksys(-L4_EINVAL, "External queue dataspace too small.");
00136         _queue_ds = qds;
00137     }
00138
00139     // Now sort out which region goes where in the dataspace.
00140     L4Re::chksys(e->rm()->attach(&_queue_region, totalsz,
00141         L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00142         L4::Ipc::make_cap_rw(_queue_ds), 0,
00143         L4_PAGESHIFT),
00144         "Attach dataspace for virtio structures");
00145
00146     l4_uint64_t devaddr;
00147     L4Re::chksys(register_ds(_queue_ds, 0, totalsz, &devaddr),
00148         "Register queue dataspace with device");
00149
00150     _queue.init_queue(queuesz, _queue_region.get());
00151
00152     config_queue(0, queuesz, devaddr, devaddr + _queue.avail_offset(),
00153         devaddr + _queue.used_offset());
00154
00155     _header_addr = devaddr + header_offset;
00156     _headers = reinterpret_cast<l4virtio_block_header_t *>(_queue_region.get()
00157         + header_offset);
00158
00159     _status_addr = devaddr + status_offset;
00160     _status = _queue_region.get() + status_offset;
00161
00162     user_devaddr = Ptr<void>(devaddr + usermem_offset);
00163     if (userdata)
00164         *userdata = _queue_region.get() + usermem_offset;
00165
00166     // setup the callback mechanism
00167     _pending.assign(queuesz, Request());
00168
00169     // Finish handshake with device.
00170     _config->driver_features_map[0] = fmask0;
00171     _config->driver_features_map[1] = fmask1;
00172     driver_acknowledge();
00173 }
00174
00178 l4virtio_block_config_t const &device_config() const
00179 {
00180     return *_config->device_config<l4virtio_block_config_t>();
00181 }
00182
00191 Handle start_request(l4_uint64_t sector, l4_uint32_t type,
00192     Callback callback)
00193 {
00194     l4_uint16_t descno = _queue.alloc_descriptor();
00195     if (descno == Virtqueue::Eoq)

```

```

00196         return Handle(Virtqueue::Eoq);
00197
00198     L4virtio::Virtqueue::Desc &desc = _queue.desc(descno);
00199     Request &req = _pending[descno];
00200
00201     // setup the header
00202     L4virtio_block_header_t &head = _headers[descno];
00203     head.type = type;
00204     head.ioprio = 0;
00205     head.sector = sector;
00206
00207     // and put it in the descriptor
00208     desc.addr = Ptr<void>(_header_addr + descno * Header_size);
00209     desc.len = Header_size;
00210     desc.flags.raw = 0; // no write, no indirect
00211
00212     req.tail = descno;
00213     req.callback = callback;
00214
00215     return Handle(descno);
00216 }
00217
00229 int add_block(Handle handle, Ptr<void> addr, l4_uint32_t size)
00230 {
00231     l4_uint16_t descno = _queue.alloc_descriptor();
00232     if (descno == Virtqueue::Eoq)
00233         return -L4_EAGAIN;
00234
00235     Request &req = _pending[handle.head];
00236     L4virtio::Virtqueue::Desc &desc = _queue.desc(descno);
00237     L4virtio::Virtqueue::Desc &prev = _queue.desc(req.tail);
00238
00239     prev.next = descno;
00240     prev.flags.next() = true;
00241
00242     desc.addr = addr;
00243     desc.len = size;
00244     desc.flags.raw = 0;
00245     if (_headers[handle.head].type > 0) // write or flush request
00246         desc.flags.write() = true;
00247
00248     req.tail = descno;
00249
00250     return L4_EOK;
00251 }
00252
00265 int send_request(Handle handle)
00266 {
00267     // add the status bit
00268     auto descno = _queue.alloc_descriptor();
00269     if (descno == Virtqueue::Eoq)
00270         return -L4_EAGAIN;
00271
00272     Request &req = _pending[handle.head];
00273     L4virtio::Virtqueue::Desc &desc = _queue.desc(descno);
00274     L4virtio::Virtqueue::Desc &prev = _queue.desc(req.tail);
00275
00276     prev.next = descno;
00277     prev.flags.next() = true;
00278
00279     desc.addr = Ptr<void>(_status_addr + descno);
00280     desc.len = 1;
00281     desc.flags.raw = 0;
00282     desc.flags.write() = true;
00283
00284     req.tail = descno;
00285
00286     send(_queue, handle.head);
00287
00288     return L4_EOK;
00289 }
00290
00306 int process_request(Handle handle)
00307 {
00308     // add the status bit
00309     auto descno = _queue.alloc_descriptor();
00310     if (descno == Virtqueue::Eoq)
00311         return -L4_EAGAIN;
00312
00313     L4virtio::Virtqueue::Desc &desc = _queue.desc(descno);
00314     L4virtio::Virtqueue::Desc &prev = _queue.desc(_pending[handle.head].tail);
00315
00316     prev.next = descno;
00317     prev.flags.next() = true;
00318
00319     desc.addr = Ptr<void>(_status_addr + descno);
00320     desc.len = 1;

```

```

00321     desc.flags.raw = 0;
00322     desc.flags.write() = true;
00323
00324     _pending[handle.head].tail = descno;
00325
00326     int ret = send_and_wait(_queue, handle.head);
00327     unsigned char status = _status[descno];
00328     free_request(handle);
00329
00330     if (ret < 0)
00331         return ret;
00332
00333     switch (status)
00334     {
00335     case L4VIRTIO_BLOCK_S_OK: return L4_EOK;
00336     case L4VIRTIO_BLOCK_S_IOERR: return -L4_EIO;
00337     case L4VIRTIO_BLOCK_S_UNSUPP: return -L4_ENOSYS;
00338     }
00339
00340     return -L4_EINVAL;
00341 }
00342
00343 void free_request(Handle handle)
00344 {
00345     if (handle.head != Virtqueue::Eoq
00346         && _pending[handle.head].tail != Virtqueue::Eoq)
00347         _queue.free_descriptor(handle.head, _pending[handle.head].tail);
00348     _pending[handle.head].tail = Virtqueue::Eoq;
00349 }
00350
00357 void process_used_queue()
00358 {
00359     for (l4_uint16_t descno = _queue.find_next_used();
00360          descno != Virtqueue::Eoq;
00361          descno = _queue.find_next_used()
00362          )
00363     {
00364         if (descno >= _queue.num() || _pending[descno].tail == Virtqueue::Eoq)
00365             L4Re::chksys(-L4_ENOSYS, "Bad descriptor number");
00366
00367         unsigned char status = _status[descno];
00368         free_request(Handle(descno));
00369
00370         if (_pending[descno].callback)
00371             _pending[descno].callback(status);
00372     }
00373 }
00374
00375 protected:
00376     L4Re::Util::Unique_cap<L4Re::Dataspace> _ds;
00377     L4::Cap<L4Re::Dataspace> _queue_ds;
00378
00379 private:
00380     L4Re::Rm::Unique_region<unsigned char *> _queue_region;
00381     l4virtio_block_header_t *_headers;
00382     unsigned char *_status;
00383     l4_uint64_t _header_addr;
00384     l4_uint64_t _status_addr;
00385     Virtqueue _queue;
00386     std::vector<Request> _pending;
00387 };
00388
00389 } }

```

## 17.255 virtio-block

```

00001 // vi:ft=cpp
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2017-2021, 2024 Kernkonzept GmbH.
00005  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00006  *
00007  */
00008 #pragma once
00009
00010 #include <l4/cxx/unique_ptr>
00011 #include <l4/re/util/unique_cap>
00012
00013 #include <climits>
00014
00015 #include <l4/l4virtio/virtio.h>
00016 #include <l4/l4virtio/virtio_block.h>
00017 #include <l4/l4virtio/server/l4virtio>

```

```

00018 #include <l4/sys/cxx/ipc_epiface>
00019
00020 namespace L4virtio { namespace Svr {
00021
00022 template <typename Ds_data> class Block_dev_base;
00023
00027 template<typename Ds_data>
00028 class Block_request
00029 {
00030     friend class Block_dev_base<Ds_data>;
00031     enum { Header_size = sizeof(l4virtio_block_header_t) };
00032
00033 public:
00034     struct Data_block
00035     {
00037         Driver_mem_region_t<Ds_data> *mem;
00039         void *addr;
00041         l4_uint32_t len;
00042
00043         Data_block() = default;
00044
00045         Data_block(Driver_mem_region_t<Ds_data> *m, Virtqueue::Desc const &desc,
00046                   Request_processor const *)
00047             : mem(m), addr(m->local(desc.addr)), len(desc.len)
00048         {}
00049     };
00050
00051
00052
00063     unsigned data_size() const
00064     {
00065         Request_processor rp;
00066         Data_block data;
00067
00068         rp.start(_mem_list, _request, &data);
00069
00070         unsigned total = data.len;
00071
00072         try
00073         {
00074             while (rp.has_more())
00075             {
00076                 rp.next(_mem_list, &data);
00077                 total += data.len;
00078             }
00079         }
00080         catch (Bad_descriptor const &e)
00081         {
00082             // need to convert the exception because e contains a raw pointer to rp
00083             throw L4::Runtime_error(-L4_EIO, "bad virtio descriptor");
00084         }
00085
00086         if (total < Header_size + 1)
00087             throw L4::Runtime_error(-L4_EIO, "virtio request too short");
00088
00089         return total - Header_size - 1;
00090     }
00091
00095     bool has_more()
00096     {
00097         // peek into the remaining data
00098         while (_data.len == 0 && _rp.has_more())
00099             _rp.next(_mem_list, &_data);
00100
00101         // there always must be one byte left for status
00102         return (_data.len > 1 || _rp.has_more());
00103     }
00104
00113     Data_block next_block()
00114     {
00115         Data_block out;
00116
00117         if (_data.len == 0)
00118         {
00119             if (!_rp.has_more())
00120                 throw L4::Runtime_error(-L4_EEXIST,
00121                                         "No more data blocks in virtio request");
00122
00123             if (_todo_blocks == 0)
00124                 throw Bad_descriptor(&_rp, Bad_descriptor::Bad_size);
00125             --_todo_blocks;
00126
00127             _rp.next(_mem_list, &_data);
00128         }
00129
00130         if (_data.len > _max_block_size)
00131             throw Bad_descriptor(&_rp, Bad_descriptor::Bad_size);

```

```

00132
00133     out = _data;
00134
00135     if (!_rp.has_more())
00136     {
00137         --(out.len);
00138         _data.len = 1;
00139         _data.addr = static_cast<char *>(_data.addr) + out.len;
00140     }
00141     else
00142         _data.len = 0; // is consumed
00143
00144     return out;
00145 }
00146
00148 l4virtio_block_header_t const &header() const
00149 { return _header; }
00150
00151 private:
00152 Block_request(Virtqueue::Request req, Driver_mem_list_t<Ds_data> *mem_list,
00153               unsigned max_blocks, l4_uint32_t max_block_size)
00154 : _mem_list(mem_list),
00155   _request(req),
00156   _todo_blocks(max_blocks),
00157   _max_block_size(max_block_size)
00158 {
00159     // read header which should be in the first block
00160     _rp.start(mem_list, _request, &_data);
00161     --_todo_blocks;
00162
00163     if (_data.len < Header_size)
00164         throw Bad_descriptor(&_rp, Bad_descriptor::Bad_size);
00165
00166     _header = *(static_cast<l4virtio_block_header_t *>(_data.addr));
00167
00168     _data.addr = static_cast<char *>(_data.addr) + Header_size;
00169     _data.len -= Header_size;
00170
00171     // if there is no space for status bit we cannot really recover
00172     if (!_rp.has_more() && _data.len == 0)
00173         throw Bad_descriptor(&_rp, Bad_descriptor::Bad_size);
00174 }
00175
00176 int release_request(Virtqueue *queue, l4_uint8_t status, unsigned sz)
00177 {
00178     // write back status
00179     // If there was an error on the way or the status byte is in its
00180     // own block, fast-forward to the last block.
00181     if (_rp.has_more())
00182     {
00183         while (_rp.next(_mem_list, &_data) && _todo_blocks > 0)
00184             --_todo_blocks;
00185
00186         if (_todo_blocks > 0 && _data.len > 0)
00187             *(static_cast<l4_uint8_t *>(_data.addr) + _data.len - 1) = status;
00188         else
00189             return -L4_EIO; // too many data blocks
00190     }
00191     else if (_data.len > 0)
00192         *(static_cast<l4_uint8_t *>(_data.addr)) = status;
00193     else
00194         return -L4_EIO; // no space for final status byte
00195
00196     // now release the head
00197     queue->consumed(_request, sz);
00198
00199     return L4_EOK;
00200 }
00201
00207 Driver_mem_list_t<Ds_data> *_mem_list;
00209 l4virtio_block_header_t _header;
00211 Request_processor _rp;
00213 Data_block _data;
00214
00216 Virtqueue::Request _request;
00218 unsigned _todo_blocks;
00220 l4_uint32_t _max_block_size;
00221 };
00222
00223 struct Block_features : public Dev_config::Features
00224 {
00225     Block_features() = default;
00226     Block_features(l4_uint32_t raw) : Dev_config::Features(raw) {}
00227
00229     CXX_BITFIELD_MEMBER( 1, 1, size_max, raw);
00231     CXX_BITFIELD_MEMBER( 2, 2, seg_max, raw);
00233     CXX_BITFIELD_MEMBER( 4, 4, geometry, raw);

```

```

00235 CXX_BITFIELD_MEMBER( 5, 5, ro, raw);
00237 CXX_BITFIELD_MEMBER( 6, 6, blk_size, raw);
00239 CXX_BITFIELD_MEMBER( 9, 9, flush, raw);
00241 CXX_BITFIELD_MEMBER(10, 10, topology, raw);
00243 CXX_BITFIELD_MEMBER(11, 11, config_wce, raw);
00245 CXX_BITFIELD_MEMBER(12, 12, mq, raw);
00247 CXX_BITFIELD_MEMBER(13, 13, discard, raw);
00249 CXX_BITFIELD_MEMBER(14, 14, write_zeroes, raw);
00250 };
00251
00252
00258 template <typename Ds_data>
00259 class Block_dev_base : public L4virtio::Svr::Device_t<Ds_data>
00260 {
00261 private:
00262     L4Re::Util::Unique_cap<L4::Irq> _kick_guest_irq;
00263     Virtqueue _queue;
00264     unsigned _vq_max;
00265     l4_uint32_t _max_block_size = UINT_MAX;
00266     Dev_config_t<l4virtio_block_config_t> _dev_config;
00267
00268 public:
00269     typedef Block_request<Ds_data> Request;
00270
00271 protected:
00272     Block_features negotiated_features() const
00273     { return _dev_config.negotiated_features(0); }
00274
00275     Block_features device_features() const
00276     { return _dev_config.host_features(0); }
00277
00278     void set_device_features(Block_features df)
00279     { _dev_config.host_features(0) = df.raw; }
00280
00290     void set_size_max(l4_uint32_t sz)
00291     {
00292         _dev_config.priv_config()->size_max = sz;
00293         Block_features df = device_features();
00294         df.size_max() = true;
00295         set_device_features(df);
00296
00297         _max_block_size = sz;
00298     }
00299
00304     void set_seg_max(l4_uint32_t sz)
00305     {
00306         _dev_config.priv_config()->seg_max = sz;
00307         Block_features df = device_features();
00308         df.seg_max() = true;
00309         set_device_features(df);
00310     }
00311
00315     void set_geometry(l4_uint16_t cylinders, l4_uint8_t heads, l4_uint8_t sectors)
00316     {
00317         l4virtio_block_config_t volatile *pc = _dev_config.priv_config();
00318         pc->geometry.cylinders = cylinders;
00319         pc->geometry.heads = heads;
00320         pc->geometry.sectors = sectors;
00321         Block_features df = device_features();
00322         df.geometry() = true;
00323         set_device_features(df);
00324     }
00325
00332     void set_blk_size(l4_uint32_t sz)
00333     {
00334         _dev_config.priv_config()->blk_size = sz;
00335         Block_features df = device_features();
00336         df.blk_size() = true;
00337         set_device_features(df);
00338     }
00339
00348     void set_topology(l4_uint8_t physical_block_exp,
00349                     l4_uint8_t alignment_offset,
00350                     l4_uint32_t min_io_size,
00351                     l4_uint32_t opt_io_size)
00352     {
00353         l4virtio_block_config_t volatile *pc = _dev_config.priv_config();
00354         pc->topology.physical_block_exp = physical_block_exp;
00355         pc->topology.alignment_offset = alignment_offset;
00356         pc->topology.min_io_size = min_io_size;
00357         pc->topology.opt_io_size = opt_io_size;
00358         Block_features df = device_features();
00359         df.topology() = true;
00360         set_device_features(df);
00361     }
00362
00364     void set_flush()

```

```

00365 {
00366     Block_features df = device_features();
00367     df.flush() = true;
00368     set_device_features(df);
00369 }
00370
00375 void set_config_wce(l4_uint8_t writeback)
00376 {
00377     l4virtio_block_config_t volatile *pc = _dev_config.priv_config();
00378     pc->writeback = writeback;
00379     Block_features df = device_features();
00380     df.config_wce() = true;
00381     set_device_features(df);
00382 }
00383
00388 l4_uint8_t get_writeback()
00389 {
00390     l4virtio_block_config_t volatile *pc = _dev_config.priv_config();
00391     return pc->writeback;
00392 }
00393
00402 void set_discard(l4_uint32_t max_discard_sectors, l4_uint32_t max_discard_seg,
00403                 l4_uint32_t discard_sector_alignment)
00404 {
00405     l4virtio_block_config_t volatile *pc = _dev_config.priv_config();
00406     pc->max_discard_sectors = max_discard_sectors;
00407     pc->max_discard_seg = max_discard_seg;
00408     pc->discard_sector_alignment = discard_sector_alignment;
00409     Block_features df = device_features();
00410     df.discard() = true;
00411     set_device_features(df);
00412 }
00413
00422 void set_write_zeroes(l4_uint32_t max_write_zeroes_sectors,
00423                      l4_uint32_t max_write_zeroes_seg,
00424                      l4_uint8_t write_zeroes_may_unmap)
00425 {
00426     l4virtio_block_config_t volatile *pc = _dev_config.priv_config();
00427     pc->max_write_zeroes_sectors = max_write_zeroes_sectors;
00428     pc->max_write_zeroes_seg = max_write_zeroes_seg;
00429     pc->write_zeroes_may_unmap = write_zeroes_may_unmap;
00430     Block_features df = device_features();
00431     df.write_zeroes() = true;
00432     set_device_features(df);
00433 }
00434
00435 public:
00444 Block_dev_base(l4_uint32_t vendor, unsigned queue_size, l4_uint64_t capacity,
00445               bool read_only)
00446 : L4virtio::Svr::Device_t<Ds_data>(&_dev_config),
00447   _vq_max(queue_size),
00448   _dev_config(vendor, L4VIRTIO_ID_BLOCK, 1)
00449 {
00450     this->reset_queue_config(0, queue_size);
00451
00452     Block_features df(0);
00453     df.ring_indirect_desc() = true;
00454     df.ro() = read_only;
00455     set_device_features(df);
00456
00457     _dev_config.set_host_feature(L4VIRTIO_FEATURE_VERSION_1);
00458
00459     _dev_config.priv_config()->capacity = capacity;
00460 }
00461
00465 virtual void reset_device() = 0;
00466
00470 virtual bool queue_stopped() = 0;
00471
00483 void finalize_request(cxx::unique_ptr<Request> req, unsigned sz,
00484                     l4_uint8_t status = L4VIRTIO_BLOCK_S_OK)
00485 {
00486     if (_dev_config.status().fail_state() || !_queue.ready())
00487         return;
00488
00489     if (req->release_request(&_queue, status, sz) < 0)
00490         this->device_error();
00491
00492     if (_queue.no_notify_guest())
00493         return;
00494
00495     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_VRING);
00496     _kick_guest_irq->trigger();
00497
00498     // Request can be dropped here.
00499 }
00500

```

```

00501 int reconfig_queue(unsigned idx) override
00502 {
00503     if (idx == 0 && this->setup_queue(&_queue, 0, _vq_max))
00504         return 0;
00505
00506     return -L4_EINVAL;
00507 }
00508
00509 void reset() override
00510 {
00511     _queue.disable();
00512     _dev_config.reset_queue(0, _vq_max);
00513     _dev_config.reset_hdr();
00514     reset_device();
00515 }
00516
00517 protected:
00518 bool check_for_new_requests()
00519 {
00520     if (!_queue.ready() || queue_stopped())
00521         return false;
00522
00523     if (_dev_config.status().fail_state())
00524         return false;
00525
00526     return _queue.desc_avail();
00527 }
00528
00530 cxx::unique_ptr<Request> get_request()
00531 {
00532     cxx::unique_ptr<Request> req;
00533
00534     if (!_queue.ready() || queue_stopped())
00535         return req;
00536
00537     if (_dev_config.status().fail_state())
00538         return req;
00539
00540     auto r = _queue.next_avail();
00541     if (!r)
00542         return req;
00543
00544     try
00545     {
00546         cxx::unique_ptr<Request> cur{
00547             new Request(r, &(this->_mem_info), _vq_max, _max_block_size)};
00548
00549         req = cxx::move(cur);
00550     }
00551     catch (Bad_descriptor const &e)
00552     {
00553         this->device_error();
00554         return req;
00555     }
00556
00557     return req;
00558 }
00559
00560 private:
00561 void register_single_driver_irq() override
00562 {
00563     _kick_guest_irq = L4Re::Util::Unique_cap<L4::Irq>(
00564         L4Re::chkcap(this->server_iface()->template rcv_cap<L4::Irq>(0)));
00565
00566     L4Re::chksys(this->server_iface()->realloc_rcv_cap(0));
00567 }
00568
00569 void trigger_driver_config_irq() override
00570 {
00571     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_CONFIG);
00572     _kick_guest_irq->trigger();
00573 }
00574
00575 bool check_queues() override
00576 {
00577     if (!_queue.ready())
00578     {
00579         reset();
00580         return false;
00581     }
00582
00583     return true;
00584 }
00585 };
00586
00587 template <typename Ds_data>
00588 struct Block_dev

```



```

00589 : Block_dev_base<Ds_data>,
00590 L4::Epiface_t<Block_dev<Ds_data>, L4virtio::Device>
00591 {
00592 private:
00593     class Irq_object : public L4::Irqep_t<Irq_object>
00594     {
00595     public:
00596         Irq_object(Block_dev<Ds_data> *parent) : _parent(parent) {}
00597
00598         void handle_irq()
00599         {
00600             _parent->kick();
00601         }
00602
00603     private:
00604         Block_dev<Ds_data> *_parent;
00605     };
00606     Irq_object _irq_handler;
00607
00608 protected:
00609     L4::Epiface *irq_iface()
00610     { return &_amp;_irq_handler; }
00611
00612 public:
00613     Block_dev(l4_uint32_t vendor, unsigned queue_size, l4_uint64_t capacity,
00614               bool read_only)
00615     : Block_dev_base<Ds_data>(vendor, queue_size, capacity, read_only),
00616       _irq_handler(this)
00617     {}
00618
00619     L4::Cap<void> register_obj(L4::Registry_iface *registry,
00620                               char const *service = 0)
00621     {
00622         L4Re::chkcap(registry->register_irq_obj(this->irq_iface()));
00623         L4::Cap<void> ret;
00624         if (service)
00625             ret = registry->register_obj(this, service);
00626         else
00627             ret = registry->register_obj(this);
00628         L4Re::chkcap(ret);
00629
00630         return ret;
00631     }
00632
00633     L4::Cap<void> register_obj(L4::Registry_iface *registry,
00634                               L4::Cap<L4::Rcv_endpoint> ep)
00635     {
00636         L4Re::chkcap(registry->register_irq_obj(this->irq_iface()));
00637
00638         return L4Re::chkcap(registry->register_obj(this, ep));
00639     }
00640
00641     typedef Block_request<Ds_data> Request;
00642     virtual bool process_request(cx::unique_ptr<Request> &&req) = 0;
00643
00644 protected:
00645     L4::Ipc_svr::Server_iface *server_iface() const override
00646     {
00647         return this->L4::Epiface::server_iface();
00648     }
00649
00650     void kick()
00651     {
00652         for (;;)
00653         {
00654             auto req = this->get_request();
00655             if (!req)
00656                 return;
00657             if (!this->process_request(cx::move(req)))
00658                 return;
00659         }
00660     }
00661
00662 private:
00663     L4::Cap<L4::Irq> device_notify_irq() const override
00664     {
00665         return L4::cap_cast<L4::Irq>(_irq_handler.obj_cap());
00666     }
00667 };
00668
00669 }
```

## 17.256 virtio-console

```

00001 // vi:ft=cpp
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2019-2024 Kernkonzept GmbH.
00005  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00006  *           Phillip Raffeck <phillip.raffeck@kernkonzept.com>
00007  *           Steffen Liebergeld <steffen.liebergeld@kernkonzept.com>
00008  *           Jan Klötzke <jan.kloetzke@kernkonzept.com>
00009  */
00010 #pragma once
00011
00012 #include <l4/l4virtio/server/l4virtio>
00013 #include <l4/re/error_helper>
00014
00015 namespace L4virtio { namespace Svr { namespace Console {
00016
00017 struct Features : Dev_config::Features
00018 {
00019     Features() = default;
00020     explicit Features(l4_uint32_t raw) : Dev_config::Features(raw) {}
00021     CXX_BITFIELD_MEMBER(0, 0, console_size, raw);
00022     CXX_BITFIELD_MEMBER(1, 1, console_multiport, raw);
00023     CXX_BITFIELD_MEMBER(2, 2, emerg_write, raw);
00024 };
00025
00026 struct Control_message
00027 {
00028     enum Events
00029     {
00030         Device_ready = 0,
00031         Device_add = 1,
00032         Device_remove = 2,
00033         Port_ready = 3,
00034         Console_port = 4,
00035         Resize = 5,
00036         Port_open = 6,
00037         Port_name = 7,
00038     };
00039     l4_uint32_t id;
00040     l4_uint16_t event;
00041     l4_uint16_t value;
00042     Control_message() {}
00043     Control_message(l4_uint32_t i, l4_uint16_t e, l4_uint16_t v)
00044         : id(i), event(e), value(v) {}
00045 };
00046
00047 struct Control_request
00048 {
00049     Control_message *msg;
00050     l4_uint32_t len;
00051     Driver_mem_region *mem;
00052 };
00053
00054 struct Port
00055 {
00056     enum Port_status
00057     {
00058         Port_disabled = 0,
00059         Port_added,
00060         Port_ready,
00061         Port_open,
00062         Port_failed,
00063         Port_num_states,
00064     };
00065     enum { Control_queue_size = 0x10 };
00066     Virtqueue tx;
00067     Virtqueue rx;
00068     Port_status status;
00069     Port_status reported_status;
00070     unsigned vq_max;
00071     Port() : status(Port_disabled), vq_max(Control_queue_size) {}
00072     Port(Port const &) = delete;
00073     Port &operator = (Port const &) = delete;
00074     virtual ~Port() = default;
00075     bool is_open() const
00076     { return status == Port_open; }

```

Generated for L4Re by Doxygen

```

00281     } __attribute__((packed));
00282
00283 public:
00293 explicit Virtio_con(unsigned max_ports, bool enable_multiport)
00294 : L4virtio::Svr::Device(&_dev_config),
00295   _num_ports(enable_multiport ? max_ports : 1),
00296   _dev_config(L4VIRTIO_VENDOR_KK, L4VIRTIO_ID_CONSOLE,
00297               enable_multiport ? max_ports * 2 + 2 : 2)
00298 {
00299     if (_num_ports < 1)
00300         L4Re::chksys(-L4_EINVAL, "At least one port is required.");
00301
00302     Features hf(0);
00303
00304     hf.console_multiport() = enable_multiport;
00305
00306     _dev_config.host_features(0) = hf.raw;
00307
00308     if (enable_multiport)
00309         _dev_config.priv_config()->max_nr_ports = _num_ports;
00310     _dev_config.reset_hdr();
00311 }
00312
00313 void reset_queue_configs()
00314 {
00315     for (unsigned q = 0; q < _dev_config.num_queues(); ++q)
00316         reset_queue_config(q, max_queue_size(q));
00317 }
00318
00319 int reconfig_queue(unsigned index) override
00320 {
00321     if (index >= _dev_config.num_queues())
00322         return -L4_ERANGE;
00323
00324     if (setup_queue(get_queue(index), index, max_queue_size(index)))
00325         return 0;
00326
00327     return -L4_EINVAL;
00328 }
00329
00334 bool multiport_enabled() const
00335 {
00336     return _negotiated_features.console_multiport()
00337         && _dev_config.num_queues() > Ctrl_rx;
00338 }
00339
00340 bool ctrl_queue_ready() const
00341 { return _ctrl_port.is_open(); }
00342
00343 bool check_features(void) override
00344 {
00345     _negotiated_features = Features(_dev_config.negotiated_features(0));
00346     return true;
00347 }
00348
00349 bool check_queues() override
00350 {
00351     // NOTE
00352     // The VIRTIO specification states:
00353     // "The port 0 receive and transmit queues always exist"
00354     // The linux driver however does not setup port 0 if the multiport feature
00355     // is negotiated.
00356     // We just go along with the linux driver and do not expect port 0 to be up,
00357     // if the multiport feature is negotiated.
00358
00359     if (multiport_enabled())
00360         // If MULTIPORT was negotiated, ctrl queues should be set up.
00361         return _ctrl_port.queues_ready();
00362
00363     // If MULTIPORT was not negotiated, port 0 should be set up.
00364     port(0)->status = Port::Port_open;
00365     return port(0)->queues_ready();
00366 }
00367
00379 int port_add(unsigned idx)
00380 {
00381     Port *p = port(idx);
00382
00383     if (p->status != Port::Port_disabled)
00384         return -L4_EPERM;
00385
00386     p->status = Port::Port_added;
00387     port_report_status(idx);
00388
00389     return L4_EOK;
00390 }
00391

```

```

00403 int port_remove(unsigned idx)
00404 {
00405     Port *p = port(idx);
00406
00407     if (p->status == Port::Port_disabled)
00408         return -L4_EPERM;
00409
00410     p->status = Port::Port_disabled;
00411     port_report_status(idx);
00412
00413     return L4_EOK;
00414 }
00415
00428 int port_open(unsigned idx, bool open)
00429 {
00430     Port *p = port(idx);
00431
00432     if ((open && p->status != Port::Port_ready)
00433         || (!open && p->status != Port::Port_open))
00434         return -L4_EPERM;
00435
00436     p->status = open ? Port::Port_open : Port::Port_ready;
00437     port_report_status(idx);
00438
00439     return L4_EOK;
00440 }
00441
00455 int port_name(unsigned idx, char const *name)
00456 {
00457     Port *p = port(idx);
00458
00459     if (p->status == Port::Port_disabled)
00460         return -L4_EPERM;
00461
00462     return send_control_message(idx, Control_message::Port_name, 0, name);
00463 }
00464
00487 int send_control_message(l4_uint32_t idx, l4_uint16_t event,
00488                         l4_uint16_t value = 0, const char *name = 0)
00489 {
00490     if (!ctrl_queue_ready())
00491         return -L4_ENODEV;
00492
00493     Virtqueue *q = &_ctrl_port.rx;
00494     if (!q->ready())
00495         return -L4_ENODEV;
00496
00497     Virtqueue::Request r = q->next_avail();
00498     if (!r)
00499         return -L4_EBUSY;
00500
00501     Request_processor rp;
00502     Control_request req;
00503     rp.start(this, r, &req);
00504
00505     if (req.len < sizeof(Control_message))
00506         return -L4_ENOMEM;
00507
00508     Control_message msg(idx, event, value);
00509
00510     memcpy(req.msg, &msg, sizeof(msg));
00511
00512     if (event == Control_message::Port_name && name)
00513     {
00514         size_t name_len = cxx::min(req.len - sizeof(msg), strlen(name));
00515         memcpy(reinterpret_cast<char*>(req.msg) + sizeof(msg), name, name_len);
00516         q->finish(r, this, sizeof(msg) + name_len);
00517     }
00518     else
00519         q->finish(r, this, sizeof(msg));
00520
00521     return L4_EOK;
00522 }
00523
00536 int handle_control_message()
00537 {
00538     // Report port state transitions if that failed in the past...
00539     if (_report_port_state)
00540     {
00541         _report_port_state = false;
00542
00543         for (unsigned i = 0; i < _num_ports; ++i)
00544             if (!port_report_status(i))
00545                 _report_port_state = true;
00546     }
00547
00548     Virtqueue *q = &_ctrl_port.tx;

```

```

00549     if (!q->ready())
00550         return -L4_ENODEV;
00551
00552     int ret = L4_EOK;
00553     Virtqueue::Request r;
00554     while ((r = q->next_avail()))
00555     {
00556         Request_processor rp;
00557         Control_request req;
00558
00559         rp.start(this, r, &req);
00560
00561         Control_message msg;
00562         if (req.len < sizeof(msg))
00563         {
00564             // Just ignore malformed input.
00565             q->finish(r, this);
00566             ret = -L4_EINVAL;
00567             continue;
00568         }
00569
00570         memcpy(&msg, req.msg, sizeof(msg));
00571         q->finish(r, this);
00572
00573         if (_ctrl_port.status == Port::Port_disabled)
00574         {
00575             // When the control queue is disabled, only device ready is accepted.
00576             if (msg.event == Control_message::Device_ready)
00577             {
00578                 if (msg.value)
00579                     _ctrl_port.status = Port::Port_open;
00580             }
00581
00582             process_device_ready(msg.value);
00583             continue;
00584         }
00585
00586         if (!ctrl_queue_ready())
00587             continue;
00588
00589         // Ignore invalid port ids
00590         if (msg.id >= max_ports())
00591             break;
00592
00593         switch (msg.event)
00594         {
00595             case Control_message::Port_ready:
00596                 process_port_ready(msg.id, msg.value);
00597                 break;
00598             case Control_message::Port_open:
00599                 process_port_open(msg.id, msg.value);
00600                 break;
00601             default:
00602                 ret = -L4_EINVAL;
00603                 break;
00604         }
00605     }
00606
00607     return ret;
00608 }
00609
00611 void load_desc(L4virtio::Virtqueue::Desc const &desc,
00612               Request_processor const *proc,
00613               L4virtio::Virtqueue::Desc const **table)
00614 {
00615     this->_mem_info.load_desc(desc, proc, table);
00616 }
00617
00619 void load_desc(L4virtio::Virtqueue::Desc const &desc,
00620               Request_processor const *proc,
00621               Control_request *data)
00622 {
00623     auto *region = this->_mem_info.find(desc.addr.get(), desc.len);
00624     if (L4_UNLIKELY(!region))
00625         throw Bad_descriptor(proc, Bad_descriptor::Bad_address);
00626
00627     data->msg = reinterpret_cast<Control_message *>(region->local(desc.addr));
00628     data->len = desc.len;
00629     data->mem = region;
00630 }
00631
00632 void reset() override
00633 {
00634     for (unsigned p = 0; p < _num_ports; ++p)
00635         port(p)->reset();
00636
00637     _ctrl_port.reset();

```

```

00638     reset_queue_configs();
00639     _dev_config.reset_hdr();
00640     _negotiated_features = Features(0);
00641     _report_port_state = false;
00642
00643     reset_device();
00644 }
00645
00652 virtual void reset_device() {}
00653
00663 virtual void notify_queue(Virtqueue *queue) = 0;
00664
00672 virtual Port *port(unsigned port) = 0;
00673 virtual Port const *port(unsigned port) const = 0;
00674
00685 virtual void process_device_ready(l4_uint16_t value) = 0;
00686
00698 virtual void process_port_ready(l4_uint32_t id, l4_uint16_t value)
00699 {
00700     Port *p = port(id);
00701
00702     switch (p->status)
00703     {
00704     case Port::Port_added:
00705     case Port::Port_ready:
00706         p->status = value ? Port::Port_ready : Port::Port_failed;
00707         break;
00708     case Port::Port_open:
00709         if (!value)
00710             p->status = Port::Port_failed;
00711         break;
00712     default:
00713         // invalid state for PORT_READY message
00714         break;
00715     }
00716 }
00717
00728 virtual void process_port_open(l4_uint32_t id, l4_uint16_t value) = 0;
00729
00730 unsigned max_ports() const
00731 { return _num_ports; }
00732
00733 private:
00734 bool is_control_queue(unsigned q) const
00735 { return q == Ctrl_rx || q == Ctrl_tx; }
00736
00737 unsigned queue_to_port(unsigned q) const
00738 { return (q == 0 || q == 1) ? 0 : (q / 2) - 1; }
00739
00748 unsigned max_queue_size(unsigned q) const
00749 {
00750     if (is_control_queue(q))
00751         return _ctrl_port.vq_max;
00752
00753     return port(queue_to_port(q))->vq_max;
00754 }
00755
00764 Virtqueue *get_queue(unsigned q)
00765 {
00766     Port *p;
00767     if (is_control_queue(q))
00768         p = &_ctrl_port;
00769     else
00770         p = port(queue_to_port(q));
00771
00772     if (q & 1)
00773         return &p->tx;
00774     else
00775         return &p->rx;
00776 }
00777
00788 bool port_report_status(unsigned idx)
00789 {
00790     Port *p = port(idx);
00791     while (p->status != p->reported_status)
00792     {
00793         auto const &trans
00794             = Port::state_transitions[p->reported_status][p->status];
00795
00796         if (trans.event >= 0
00797             && send_control_message(idx, trans.event, trans.value) < 0)
00798         {
00799             _report_port_state = true;
00800             return false;
00801         }
00802
00803         p->reported_status = trans.next;

```

```

00804     }
00805
00806     return true;
00807 }
00808
00809 unsigned _num_ports;
00810 bool _report_port_state = false;
00811
00812 protected:
00813     Dev_config_t<Serial_config_space> _dev_config;
00814     Port _ctrl_port;
00815     Features _negotiated_features{0};
00816 };
00817
00818 }}} // name space

```

## 17.257 virtio-console-device

```

00001 // vi:ft=cpp
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2019-2024 Kernkonzept GmbH.
00005  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00006  *            Phillip Raffeck <phillip.raffeck@kernkonzept.com>
00007  *            Steffen Liebergeld <steffen.liebergeld@kernkonzept.com>
00008  *            Jan Klötzke <jan.kloetzke@kernkonzept.com>
00009  */
00010 #pragma once
00011
00012 #include <l4/cxx/bitmap>
00013 #include <l4/cxx/static_vector>
00014 #include <l4/l4virtio/server/l4virtio>
00015 #include <l4/l4virtio/server/virtio-console>
00016 #include <l4/re/error_helper>
00017
00018 namespace L4virtio { namespace Svr { namespace Console {
00019
00020     struct Device_port : public Port
00021     {
00022     public:
00023         struct Buffer : Data_buffer
00024         {
00025         public:
00026             Buffer() = default;
00027             Buffer(Driver_mem_region const &r,
00028                  Virtqueue::Desc const &d,
00029                  Request_processor const *)
00030             {
00031                 pos = static_cast<char *>(r->local(d.addr));
00032                 left = d.len;
00033             }
00034         };
00035
00036         Request_processor rp;
00037         Virtqueue::Request request;
00038         Buffer src;
00039
00040         bool poll_in_req = true;
00041         bool poll_out_req = true;
00042
00043         void reset() override
00044         {
00045             Port::reset();
00046             request = Virtqueue::Request();
00047             poll_in_req = true;
00048             poll_out_req = true;
00049         }
00050     };
00051
00052     class Device
00053     : public Virtio_console
00054     {
00055     public:
00056         class Irq_object : public L4::Irqep_t<Irq_object>
00057         {
00058         public:
00059             Irq_object(Device *parent) : _parent(parent) {}
00060
00061             void handle_irq() { _parent->kick(); }
00062
00063         private:
00064             Device *_parent;
00065         };
00066
00067     protected:
00068         L4::Epiface *irq_iface()

```



```

00134     { return &_irq_handler; }
00135
00136 public:
00145 explicit Device(unsigned vq_max)
00146 : Virtio_con(1, false),
00147   _irq_handler(this),
00148   _ports(cxx::make_unique<Device_port[]>(1))
00149 {
00150     _ports[0].vq_max = vq_max;
00151     reset_queue_configs();
00152 }
00153
00163 explicit Device(unsigned vq_max, unsigned ports)
00164 : Virtio_con(ports, true),
00165   _irq_handler(this),
00166   _ports(cxx::make_unique<Device_port[]>(ports))
00167 {
00168     for (unsigned i = 0; i < ports; ++i)
00169         _ports[i].vq_max = vq_max;
00170     reset_queue_configs();
00171 }
00172
00182 explicit Device(cxx::static_vector<unsigned> const &vq_max_nums)
00183 : Virtio_con(vq_max_nums.size(), true),
00184   _irq_handler(this),
00185   _ports(cxx::make_unique<Device_port[]>(max_ports()))
00186 {
00187     for (unsigned i = 0; i < vq_max_nums.size(); ++i)
00188         _ports[i].vq_max = vq_max_nums[i];
00189     reset_queue_configs();
00190 }
00191
00192 void register_single_driver_irq() override
00193 {
00194     _kick_driver_irq = L4Re::Util::Unique_cap<L4::Irq>(
00195         L4Re::chkcap(server_iface()->rcv_cap<L4::Irq>(0)));
00196     L4Re::chksys(server_iface()->realloc_rcv_cap(0));
00197 }
00198
00199 L4::Cap<L4::Irq> device_notify_irq() const override
00200 { return _irq_handler.obj_cap(); }
00201
00202 void notify_queue(Virtqueue *queue) override
00203 {
00204     if (queue->no_notify_guest())
00205         return;
00206
00207     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_VRING);
00208     _kick_driver_irq->trigger();
00209 }
00210
00214 virtual void rx_data_available(unsigned port) = 0;
00215
00219 virtual void tx_space_available(unsigned port) = 0;
00220
00224 virtual bool queues_stopped()
00225 { return false; }
00226
00227 void trigger_driver_config_irq() override
00228 {
00229     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_CONFIG);
00230     _kick_driver_irq->trigger();
00231 }
00232
00233 void kick()
00234 {
00235     if (queues_stopped())
00236         return;
00237
00238     // We're not interested in logging any errors, just ignore return value.
00239     handle_control_message();
00240
00241     for (unsigned i = 0; i < max_ports(); ++i)
00242     {
00243         auto &p = _ports[i];
00244         if (p.poll_in_req && p.tx_ready() && p.tx.desc_avail())
00245         {
00246             p.poll_in_req = false;
00247             rx_data_available(i);
00248         }
00249
00250         if (p.poll_out_req && p.rx_ready() && p.rx.desc_avail())
00251         {
00252             p.poll_out_req = false;
00253             tx_space_available(i);
00254         }
00255     }

```

```

00256     }
00257
00272 unsigned port_read(char *buf, unsigned len, unsigned port = 0)
00273 {
00274     unsigned total = 0;
00275     Device_port &p = _ports[port];
00276     Virtqueue *q = &p.tx;
00277
00278     Data_buffer dst;
00279     dst.pos = buf;
00280     dst.left = len;
00281
00282     while (dst.left)
00283     {
00284         try
00285         {
00286             // Make sure we have a valid request where we can read data from
00287             if (!p.request.valid())
00288             {
00289                 p.request = p.tx_ready() ? q->next_avail()
00290                     : Virtqueue::Request();
00291                 if (!p.request.valid())
00292                     break;
00293
00294                 p.rp.start(mem_info(), p.request, &p.src);
00295             }
00296
00297             total += p.src.copy_to(&dst);
00298
00299             // We might have eaten up the current descriptor. Move to the next
00300             // if this is the case. At the end of the descriptor chain we have
00301             // to retire the current request altogether.
00302             if (!p.src.left)
00303             {
00304                 if (!p.rp.next(mem_info(), &p.src))
00305                 {
00306                     q->finish(p.request, this);
00307                     p.request = Virtqueue::Request();
00308                 }
00309             }
00310         }
00311         catch (Bad_descriptor const &)
00312         {
00313             q->finish(p.request, this);
00314             p.request = Virtqueue::Request();
00315             device_error();
00316             break;
00317         }
00318     }
00319
00320     if (total < len)
00321         p.poll_in_req = true;
00322
00323     return total;
00324 }
00325
00341 unsigned port_write(char const *buf, unsigned len, unsigned port = 0)
00342 {
00343     unsigned total = 0;
00344     Device_port &p = _ports[port];
00345     Virtqueue *q = &p.rx;
00346
00347     Data_buffer src;
00348     src.pos = const_cast<char*>(buf);
00349     src.left = len;
00350
00351     Request_processor rp;
00352     while (src.left)
00353     {
00354         auto r = p.rx_ready() ? q->next_avail() : Virtqueue::Request();
00355         if (!r.valid())
00356             break;
00357
00358         l4_uint32_t chunk = 0;
00359         try
00360         {
00361             Device_port::Buffer dst;
00362             rp.start(mem_info(), r, &dst);
00363
00364             for (;;)
00365             {
00366                 chunk += src.copy_to(&dst);
00367                 if (!src.left)
00368                     break;
00369                 if (!rp.next(mem_info(), &dst))
00370                     break;
00371             }

```

```

00372     }
00373     catch (Bad_descriptor const &)
00374     {
00375         device_error();
00376     }
00377
00378     q->finish(r, this, chunk);
00379     total += chunk;
00380 }
00381
00382 if (total < len)
00383     p.poll_out_req = true;
00384
00385 return total;
00386 }
00387
00399 void process_device_ready(l4_uint16_t value) override
00400 {
00401     if (!value)
00402         return;
00403
00404     for (unsigned i = 0; i < max_ports(); ++i)
00405         port_add(i);
00406 }
00407
00422 void process_port_ready(l4_uint32_t id, l4_uint16_t value) override
00423 {
00424     Virtio_con::process_port_ready(id, value);
00425
00426     Port *p = port(id);
00427     if (p->status == Port::Port_failed)
00428         port_remove(id);
00429     else if (p->status == Port::Port_ready)
00430         port_open(id, true);
00431 }
00432
00443 virtual void process_port_open(l4_uint32_t id, l4_uint16_t value)
00444 {
00445     static_cast<void>(id);
00446     static_cast<void>(value);
00447 }
00448
00449 protected:
00450 Port* port(unsigned idx) override
00451 {
00452     return &_amp;ports[idx];
00453 }
00454
00455 Port const *port(unsigned idx) const override
00456 {
00457     return &_amp;ports[idx];
00458 }
00459
00460 private:
00461 Irq_object _irq_handler;
00462 cxx::unique_ptr<Device_port[]> _ports;
00463 L4Re::Util::Unique_cap<L4::Irq> _kick_driver_irq;
00464 };
00465
00466 }}} // name space

```

## 17.258 virtio-gpio-device

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2025 Kernkonzept GmbH.
00004  * Author(s): Christian Pötzsch <christian.poetzsch@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 #include <l4/re/error_helper>
00012 #include <l4/sys/cxx/ipc_epiface>
00013
00014 #include <l4/l4virtio/server/virtio>
00015 #include <l4/l4virtio/server/l4virtio>
00016 #include <l4/l4virtio/l4virtio>
00017
00018 #include <l4/re/error_helper>
00019 #include <l4/re/util/object_registry>
00020 #include <l4/re/util/br_manager>

```

```

00021 #include <l4/sys/cxx/ipc_epiface>
00022 #include <l4/cxx/pair>
00023
00024 #include <vector>
00025 #include <memory>
00026
00027 namespace L4virtio {
00028 namespace Svr {
00029
00030 /* GPIO message status types */
00031 enum : l4_uint8_t
00032 {
00033     Gpio_status_ok = 0x0,
00034     Gpio_status_err = 0x1
00035 };
00036
00037 /* GPIO message types */
00038 enum : l4_uint8_t
00039 {
00040     Gpio_msg_get_line_names = 0x1,
00041     Gpio_msg_get_direction = 0x2,
00042     Gpio_msg_set_direction = 0x3,
00043     Gpio_msg_get_value = 0x4,
00044     Gpio_msg_set_value = 0x5,
00045     Gpio_msg_set_irq_type = 0x6
00046 };
00047
00048 /* GPIO value types */
00049 enum : l4_uint8_t
00050 {
00051     Gpio_low = 0x0,
00052     Gpio_high = 0x1
00053 };
00054
00055 /* GPIO direction types */
00056 enum : l4_uint8_t
00057 {
00058     Gpio_direction_none = 0x0,
00059     Gpio_direction_out = 0x1,
00060     Gpio_direction_in = 0x2
00061 };
00062
00063 /* GPIO interrupt types */
00064 enum : l4_uint8_t
00065 {
00066     Gpio_irq_type_none = 0x0,
00067     Gpio_irq_type_edge_rising = 0x1,
00068     Gpio_irq_type_edge_falling = 0x2,
00069     Gpio_irq_type_edge_both = 0x3,
00070     Gpio_irq_type_level_high = 0x4,
00071     Gpio_irq_type_level_low = 0x8
00072 };
00073
00074 /* GPIO interrupt status types */
00075 enum : l4_uint8_t
00076 {
00077     Gpio_irq_status_invalid = 0x0,
00078     Gpio_irq_status_valid = 0x1
00079 };
00080
00081 struct Gpio_request
00082 {
00083     l4_uint16_t type;
00084     l4_uint16_t gpio;
00085     l4_uint32_t value;
00086 };
00087 static_assert(sizeof(Gpio_request) == 8,
00088     "Gpio_request contains padding bytes.");
00089
00090 struct Gpio_response
00091 {
00092     l4_uint8_t status;
00093     l4_uint8_t value;
00094 };
00095 static_assert(sizeof(Gpio_response) == 2,
00096     "Gpio_response contains padding bytes.");
00097
00098 struct Gpio_irq_request
00099 {
00100     l4_uint16_t gpio;
00101 };
00102 static_assert(sizeof(Gpio_irq_request) == 2,
00103     "Gpio_irq_request contains padding bytes.");
00104
00105 struct Gpio_irq_response
00106 {
00107     l4_uint8_t status;

```

```

00108 };
00109 static_assert(sizeof(Gpio_irq_response) == 1,
00110               "Gpio_irq_response contains padding bytes.");
00111
00112 struct Gpio_request_msg
00113 {
00114     struct Gpio_request *in_hdr = nullptr;
00115     struct Gpio_response *out_hdr = nullptr;
00116 };
00117
00118 struct Gpio_irq_request_msg
00119 {
00120     struct Gpio_irq_request *in_hdr = nullptr;
00121     struct Gpio_irq_response *out_hdr = nullptr;
00122 };
00123
00140 template <typename Request_handler,
00141           typename Epiface = L4virtio::Device>
00142 class Virtio_gpio : public L4virtio::Svr::Device,
00143                    public L4::Epiface_t<Virtio_gpio<Request_handler,
00144                                         Epiface>,
00145                                         Epiface>
00146 {
00147 private:
00148     enum
00149     {
00150         queue_size = 128,
00151     };
00152
00153 public:
00154     using Gpio_request_handler = Request_handler;
00155
00166     struct Irq_handler
00167     {
00168         Irq_handler(Virtio_gpio *gpio, L4virtio::Svr::Virtqueue *q,
00169                    L4virtio::Svr::Virtqueue::Head_desc const &head,
00170                    l4_uint8_t *status)
00171             : _gpio(gpio), _q(q), _head(head), _status(status)
00172         {}
00173
00174         void handle_irq()
00175         {
00176             *_status = Gpio_irq_status_valid;
00177             _q->finish(_head, _gpio, sizeof(Gpio_irq_response));
00178         }
00179
00180         void cancel()
00181         {
00182             *_status = Gpio_irq_status_invalid;
00183             _q->finish(_head, _gpio, sizeof(Gpio_irq_response));
00184         }
00185
00186     private:
00187         Virtio_gpio *_gpio;
00188         L4virtio::Svr::Virtqueue *_q;
00189         L4virtio::Svr::Virtqueue::Head_desc _head;
00190         l4_uint8_t *_status;
00191     };
00192
00198     struct Host_irq : L4::Irqep_t<Host_irq>
00199     {
00200         explicit Host_irq(Virtio_gpio *gpio)
00201             : L4::Irqep_t<Host_irq>(), _gpio(gpio) {}
00202
00203         void handle_irq()
00204         { _gpio->handle_queue(); }
00205
00206     private:
00207         Virtio_gpio *_gpio;
00208     };
00209
00213     struct Request_processor : L4virtio::Svr::Request_processor
00214     {
00215         Request_processor(L4virtio::Svr::Virtqueue *q, Gpio_request_handler *hndlr,
00216                          Virtio_gpio *gpio)
00217             : _q(q), _req_handler(hndlr), _gpio(gpio), _head(), _req()
00218         {}
00219
00220     protected:
00221         bool init_queue()
00222         {
00223             auto r = _q->next_avail();
00224
00225             if (L4_UNLIKELY(!r))
00226                 return false;
00227
00228             _head = start(_gpio->mem_info(), r, &_req);

```

```

00229         return true;
00230     }
00231 }
00232
00240 template <typename T>
00241 T get_request()
00242 {
00243     T req;
00244     req.in_hdr = reinterpret_cast<decltype(T::in_hdr)>(_req.pos);
00245
00246     // Need the next output buffer.
00247     if (!next(_gpio->mem_info(), &_req) || !_current_flags().write())
00248         return req;
00249
00250     req.out_hdr = reinterpret_cast<decltype(T::out_hdr)>(_req.pos);
00251
00252     return req;
00253 }
00254
00255 struct Data_buffer : public L4virtio::Svr::Data_buffer
00256 {
00257     Data_buffer()
00258     {
00259         pos = nullptr;
00260         left = 0;
00261     }
00262     // This constructor is called from within start, so make it available.
00263     Data_buffer(L4virtio::Svr::Driver_mem_region const *r,
00264                L4virtio::Svr::Virtqueue::Desc const &d,
00265                L4virtio::Svr::Request_processor const *)
00266     {
00267         pos = static_cast<char *>(r->local(d.addr));
00268         left = d.len;
00269     }
00270 };
00271
00272 L4virtio::Svr::Virtqueue *_q;
00273 Gpio_request_handler *_req_handler;
00274 Virtio_gpio *_gpio;
00275 L4virtio::Svr::Virtqueue::Head_desc _head;
00276 Data_buffer _req;
00277 };
00278
00279 // Handler for the gpio request queue
00280 struct Req_processor : Request_processor
00281 {
00282     using Request_processor::Request_processor;
00283
00284     void handle_request()
00285     {
00286         if (!this->_head)
00287             if (!this->init_queue())
00288                 return;
00289
00290         using Consumed_entry =
00291             cxx::Pair<L4virtio::Svr::Virtqueue::Head_desc, l4_uint32_t>;
00292         std::vector<Consumed_entry> consumed;
00293
00294         for (;;)
00295         {
00296             Gpio_request_msg req = this->template get_request<Gpio_request_msg>();
00297             if (!req.in_hdr || !req.out_hdr)
00298             {
00299                 this->_gpio->device_error();
00300                 break;
00301             }
00302
00303             // default response is error
00304             req.out_hdr->status = Gpio_status_err;
00305             switch (req.in_hdr->type)
00306             {
00307                 case Gpio_msg_get_line_names:
00308                     // we don't support this
00309                     break;
00310                 case Gpio_msg_get_direction:
00311                 {
00312                     if (this->_req_handler->get_direction(req.in_hdr->gpio,
00313                                                         &req.out_hdr->value))
00314                         req.out_hdr->status = Gpio_status_ok;
00315                     break;
00316                 }
00317                 case Gpio_msg_set_direction:
00318                 {
00319                     if (req.in_hdr->value == Gpio_direction_none ||
00320                         req.in_hdr->value == Gpio_direction_out ||
00321                         req.in_hdr->value == Gpio_direction_in)

```

```

00323         {
00324             if (this->_req_handler->set_direction(req.in_hdr->gpio,
00325                                                 req.in_hdr->value))
00326                 req.out_hdr->status = Gpio_status_ok;
00327         }
00328         break;
00329     }
00330     case Gpio_msg_get_value:
00331     {
00332         if (this->_req_handler->get_value(req.in_hdr->gpio,
00333                                         &req.out_hdr->value))
00334             req.out_hdr->status = Gpio_status_ok;
00335         break;
00336     }
00337     case Gpio_msg_set_value:
00338     {
00339         if (req.in_hdr->value == Gpio_low ||
00340             req.in_hdr->value == Gpio_high)
00341         {
00342             if (this->_req_handler->set_value(req.in_hdr->gpio,
00343                                             req.in_hdr->value))
00344                 req.out_hdr->status = Gpio_status_ok;
00345             break;
00346         }
00347     }
00348     case Gpio_msg_set_irq_type:
00349     {
00350         if (req.in_hdr->value == Gpio_irq_type_none ||
00351             req.in_hdr->value == Gpio_irq_type_edge_rising ||
00352             req.in_hdr->value == Gpio_irq_type_edge_falling ||
00353             req.in_hdr->value == Gpio_irq_type_edge_both ||
00354             req.in_hdr->value == Gpio_irq_type_level_high ||
00355             req.in_hdr->value == Gpio_irq_type_level_low)
00356         {
00357             if (this->_req_handler->set_irq_type(req.in_hdr->gpio,
00358                                                 req.in_hdr->value))
00359                 req.out_hdr->status = Gpio_status_ok;
00360             break;
00361         }
00362     }
00363 }
00364
00365 // Save the descriptors which are done
00366 consumed.emplace_back(this->_head, sizeof(Gpio_response));
00367
00368 if (!this->init_queue())
00369     break;
00370 }
00371
00372 // Put all finished descriptors back into the used list and notify the
00373 // driver.
00374 this->_q->finish(consumed.begin(), consumed.end(), this->_gpio);
00375
00376 this->_head = Virtqueue::Head_desc();
00377 }
00378 };
00379
00380 // Handler for the gpio event queue
00381 struct Irq_req_processor : Request_processor
00382 {
00383     using Request_processor::Request_processor;
00384
00385     void handle_request()
00386     {
00387         if (!this->_head)
00388             if (!this->init_queue())
00389                 return;
00390
00391         for (;;)
00392         {
00393             // There is only one type of message in the event queue. This
00394             // basically arms (unmask) the irq.
00395             Gpio_irq_request_msg req = this->template get_request<Gpio_irq_request_msg>();
00396             if (!req.in_hdr || !req.out_hdr)
00397             {
00398                 this->_gpio->device_error();
00399                 break;
00400             }
00401
00402             // Save the virtio descriptor for this event in an extra Irq_handler
00403             // object. The descriptor will be returned to the client when the irq
00404             // is triggered or canceled.
00405             this->_req_handler->enable_irq(req.in_hdr->gpio,
00406                                         std::make_shared<Irq_handler>(this->_gpio,
00407                                                                     this->_q,
00408                                                                     this->_head,
00409                                                                     &req.out_hdr->status));

```

```

00410
00411         if (!this->init_queue())
00412             break;
00413     }
00414
00415     this->_head = Virtqueue::Head_desc();
00416 }
00417 };
00418
00419 struct Features : public L4virtio::Svr::Dev_config::Features
00420 {
00421     Features() = default;
00422     Features(l4_uint32_t raw) : L4virtio::Svr::Dev_config::Features(raw) {}
00423
00424     CXX_BITFIELD_MEMBER(0, 0, gpio_f_irq, raw);
00425 };
00426
00427 struct Gpio_config_space
00428 {
00429     l4_uint16_t ngpio;
00430     l4_uint8_t padding[2];
00431     l4_uint32_t gpio_names_size;
00432 };
00433
00434 Virtio_gpio(Gpio_request_handler *hndlr,
00435             L4Re::Util::Object_registry *registry,
00436             l4_uint16_t ngpio)
00437 : L4virtio::Svr::Device(&_dev_config),
00438   _registry(registry),
00439   _dev_config(L4VIRTIO_VENDOR_KK, L4VIRTIO_ID_GPIO, 2),
00440   _host_irq(this),
00441   _req_processor(&_q[0], hndlr, this),
00442   _irq_req_processor(&_q[1], hndlr, this)
00443 {
00444     init_mem_info(2);
00445
00446     for (size_t i = 0; i < 2; i++)
00447     {
00448         reset_queue_config(i, queue_size);
00449         setup_queue(&_q[i], i, queue_size);
00450     }
00451
00452     registry->register_irq_obj(&_host_irq);
00453
00454     Features hf(0);
00455     hf.ring_indirect_desc() = true;
00456     hf.gpio_f_irq() = true;
00457     _dev_config.host_features(0) = hf.raw;
00458     _dev_config.set_host_feature(L4VIRTIO_FEATURE_VERSION_1);
00459
00460     // fill gpio config space
00461     _dev_config.priv_config()->ngpio = ngpio;
00462     _dev_config.priv_config()->gpio_names_size = 0; // not supported
00463
00464     _dev_config.reset_hdr();
00465 }
00466
00467 ~Virtio_gpio()
00468 { _registry->unregister_obj(&_host_irq); }
00469
00470 void notify_queue(L4virtio::Svr::Virtqueue *queue)
00471 {
00472     if (queue->no_notify_guest())
00473         return;
00474
00475     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_VRING);
00476     L4Re::chkipc(_notify_guest_irq->trigger(), "trigger guest irq");
00477 }
00478
00479 void handle_queue()
00480 {
00481     _req_processor.handle_request();
00482     _irq_req_processor.handle_request();
00483 }
00484
00485 void reset() override
00486 {}
00487
00488 bool check_queues() override
00489 { return true; }
00490
00491 int reconfig_queue(unsigned idx) override
00492 {
00493     if (idx >= sizeof(_q) / sizeof(_q[0]))
00494         return -L4_ERANGE;
00495
00496     return setup_queue(_q + idx, idx, queue_size);

```



```

00497     }
00498
00499     void trigger_driver_config_irq() override
00500     {
00501         _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_CONFIG);
00502         _notify_guest_irq->trigger();
00503     }
00504
00505     L4::Ipc_svr::Server_iface *server_iface() const override
00506     { return L4::Epiface::server_iface(); }
00507
00508     long op_set_status(L4virtio::Device::Rights r, unsigned status)
00509     { return L4virtio::Svr::Device::op_set_status(r, status); }
00510
00511     long op_config_queue(L4virtio::Device::Rights r, unsigned queue)
00512     { return L4virtio::Svr::Device::op_config_queue(r, queue); }
00513
00514     long op_device_config(L4virtio::Device::Rights r,
00515                          L4::Ipc::Cap<L4Re::Dataspace> &config_ds,
00516                          l4_addr_t &ds_offset)
00517     { return L4virtio::Svr::Device::op_device_config(r, config_ds, ds_offset); }
00518
00519     L4::Cap<L4::Irq> device_notify_irq() const override
00520     { return L4::cap_cast<L4::Irq>(_host_irq.obj_cap()); }
00521
00522     void register_single_driver_irq() override
00523     {
00524         _notify_guest_irq = L4Re::chkcap
00525             (server_iface()->template rcv_cap<L4::Irq>(0));
00526
00527         L4Re::chksys(server_iface()->realloc_rcv_cap(0));
00528     }
00529
00530 private:
00531     L4Re::Util::Object_registry *_registry;
00532     L4virtio::Svr::Dev_config_t<Gpio_config_space> _dev_config;
00533     Host_irq _host_irq;
00534     L4::Cap<L4::Irq> _notify_guest_irq;
00535     L4virtio::Svr::Virtqueue _q[2];
00536     Req_processor _req_processor;
00537     Irq_req_processor _irq_req_processor;
00538 };
00539
00540 } // namespace Svr
00541 } // namespace L4virtio

```

## 17.259 virtio-i2c-device

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2024 Kernkonzept GmbH.
00004  * Author(s): Martin Kuetzler <martin.kuetzler@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 #include <l4/re/error_helper>
00012 #include <l4/sys/cxx/ipc_epiface>
00013
00014 #include <l4/l4virtio/server/virtio>
00015 #include <l4/l4virtio/server/l4virtio>
00016 #include <l4/l4virtio/l4virtio>
00017
00018 #include <l4/re/error_helper>
00019 #include <l4/re/util/object_registry>
00020 #include <l4/re/util/br_manager>
00021 #include <l4/sys/cxx/ipc_epiface>
00022
00023 #include <vector>
00024 #include <l4/cxx/pair>
00025
00026 namespace L4virtio {
00027 namespace Svr {
00028
00029     enum : l4_uint8_t
00030     {
00031         I2c_msg_ok = 0,
00032         I2c_msg_err = 1,
00033     };
00034
00035     struct I2c_request_flags

```



```

00141     }
00142
00143 };
00144
00145 Request_processor(L4virtio::Svr::Virtqueue *q, I2c_request_handler *hndlr,
00146                 Virtio_i2c *i2c)
00147 : _q(q), _req_handler(hndlr), _i2c(i2c), _head(), _req(),
00148   _fail_next(false)
00149 {}
00150
00151 bool init_queue()
00152 {
00153     auto r = _q->next_avail();
00154
00155     if (L4_UNLIKELY(!r))
00156         return false;
00157
00158     _head = start(_i2c->mem_info(), r, &_req);
00159
00160     return true;
00161 }
00162
00172 I2c_req get_request()
00173 {
00174     I2c_req request;
00175     memcpy(&request.out_hdr, _req.pos, sizeof(I2c_out_hdr));
00176
00177     // number of bytes to be written in the answer.
00178     request.write_size = sizeof(I2c_in_hdr);
00179     request.buf_len = 0;
00180
00181     Data_buffer req;
00182     // 2nd part: either the optional buffer or the in_hdr
00183     if (next(_i2c->mem_info(), &req))
00184     {
00185
00186         request.buf_len += req.left;
00187         request.buf = reinterpret_cast<l4_uint8_t *>(req.pos);
00188     }
00189
00190     // 3rd part: in_hdr
00191     if (next(_i2c->mem_info(), &req))
00192     {
00193         // 2nd part was indeed a buffer
00194         if (request.out_hdr.flags.m_rd())
00195             request.write_size += request.buf_len;
00196
00197         // actual 3rd part
00198         request.in_hdr = reinterpret_cast<I2c_in_hdr *>(req.pos);
00199     }
00200     else
00201     {
00202         // no 3rd part, 2nd part is in_hdr;
00203         request.in_hdr = reinterpret_cast<I2c_in_hdr *>(request.buf);
00204         request.buf = nullptr;
00205         request.buf_len = 0;
00206     }
00207
00208     return request;
00209 }
00210
00211 void handle_request()
00212 {
00213     if (!_head)
00214         if (!init_queue())
00215             return;
00216
00217     using Consumed_entry =
00218         cxx::Pair<L4virtio::Svr::Virtqueue::Head_desc, l4_uint32_t>;
00219     std::vector<Consumed_entry> consumed;
00220
00221     for (;;)
00222     {
00223         auto r = get_request();
00224         if (_fail_next)
00225         {
00226             r.set_status(I2c_msg_err);
00227             _fail_next = r.out_hdr.flags.fail_next();
00228         }
00229         else
00230         {
00231             bool ok;
00232             l4_uint16_t i2c_addr = r.out_hdr.addr >> 1;
00233             if (r.out_hdr.flags.m_rd())
00234                 ok = _req_handler->handle_read(i2c_addr, r.buf, r.buf_len);
00235             else
00236                 ok = _req_handler->handle_write(i2c_addr, r.buf, r.buf_len);

```

```

00237         if (ok)
00238         {
00239             r.set_status(I2c_msg_ok);
00240             _fail_next = false;
00241         }
00242         else
00243         {
00244             r.set_status(I2c_msg_err);
00245             _fail_next = r.out_hdr.flags.fail_next();
00246         }
00247     }
00248     consumed.emplace_back(_head, r.write_size);
00249     if (!init_queue())
00250         break;
00251 }
00252
00253 _q->finish(consumed.begin(), consumed.end(), _i2c);
00254
00255 _head = Virtqueue::Head_desc();
00256 }
00257
00258 private:
00259     L4virtio::Svr::Virtqueue *_q;
00260     I2c_request_handler *_req_handler;
00261     Virtio_i2c *_i2c;
00262     L4virtio::Svr::Virtqueue::Head_desc _head;
00263     Data_buffer _req;
00264     bool _fail_next;
00265 };
00266
00267 struct Features : public L4virtio::Svr::Dev_config::Features
00268 {
00269     Features() = default;
00270     Features(L4_uint32_t raw) : L4virtio::Svr::Dev_config::Features(raw) {}
00271
00272     // This feature is mandatory. The driver is requested to abort communication
00273     // if this is not offered.
00274     CXX_BITFIELD_MEMBER(0, 0, zero_length_request, raw);
00275 };
00276
00277 Virtio_i2c(I2c_request_handler *hndlr, L4Re::Util::Object_registry *registry)
00278 : L4virtio::Svr::Device(&_dev_config),
00279   _dev_config(L4VIRTIO_VENDOR_KK, L4VIRTIO_ID_I2C, Num_request_queues),
00280   _req_handler(hndlr),
00281   _host_irq(this),
00282   _request_processor(&_q, hndlr, this)
00283 {
00284     init_mem_info(2);
00285     reset_queue_config(0, queue_size);
00286     setup_queue(&_q, 0, queue_size);
00287     registry->register_irq_obj(&_host_irq);
00288
00289     Features hf(0);
00290     hf.ring_indirect_desc() = true;
00291     hf.zero_length_request() = true;
00292     _dev_config.host_features(0) = hf.raw;
00293     _dev_config.set_host_feature(L4VIRTIO_FEATURE_VERSION_1);
00294     _dev_config.reset_hdr();
00295 }
00296
00297 void notify_queue(L4virtio::Svr::Virtqueue *)
00298 {
00299     if (_q.no_notify_guest())
00300         return;
00301
00302     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_VRING);
00303     L4Re::chkipc(_notify_guest_irq->trigger(), "trigger guest irq");
00304 }
00305
00306 void handle_queue()
00307 {
00308     _request_processor.handle_request();
00309 }
00310
00311 void reset() override
00312 {
00313 }
00314
00315 bool check_queues() override
00316 {
00317     return true;
00318 }
00319
00320 int reconfig_queue(unsigned idx) override
00321 {
00322     if (idx != 0)
00323         return -L4_ERANGE;

```

```

00324
00325     setup_queue(&q, 0, queue_size);
00326
00327     return L4_EOK;
00328 }
00329
00330 void trigger_driver_config_irq() override
00331 {
00332     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_CONFIG);
00333     _notify_guest_irq->trigger();
00334 }
00335
00336 L4::Ipc_svr::Server_iface *server_iface() const override
00337 {
00338     return L4::Epiface::server_iface();
00339 }
00340
00341 long op_set_status(L4virtio::Device::Rights r, unsigned status)
00342 {
00343     return L4virtio::Svr::Device::op_set_status(r, status);
00344 }
00345
00346 long op_config_queue(L4virtio::Device::Rights r, unsigned queue)
00347 {
00348     return L4virtio::Svr::Device::op_config_queue(r, queue);
00349 }
00350
00351 long op_device_config(L4virtio::Device::Rights r,
00352                      L4::Ipc::Cap<L4Re::Dataspace> &config_ds,
00353                      l4_addr_t &ds_offset)
00354 {
00355     return L4virtio::Svr::Device::op_device_config(r, config_ds, ds_offset);
00356 }
00357
00358 L4::Cap<L4::Irq> device_notify_irq() const override
00359 {
00360     return L4::cap_cast<L4::Irq>(_host_irq.obj_cap());
00361 }
00362
00363 void register_single_driver_irq() override
00364 {
00365     _notify_guest_irq = L4Re::chkcapi
00366         (server_iface()->template rcv_cap<L4::Irq>(0));
00367     L4Re::chksys(server_iface()->realloc_rcv_cap(0));
00368 }
00369
00370
00371 private:
00372     L4virtio::Svr::Dev_config_t<L4virtio::Svr::No_custom_data> _dev_config;
00373     I2c_request_handler *_req_handler;
00374     L4virtio::Svr::Virtqueue _q;
00375     Host_irq _host_irq;
00376     L4::Cap<L4::Irq> _notify_guest_irq;
00377     Request_processor _request_processor;
00378 };
00379
00380
00381 } // namespace Svr
00382 } // namespace L4virtio

```

## 17.260 virtio-rng-device

```

00001 // vi:ft=c++: -- Mode: C++ --
00002 /*
00003  * Copyright (C) 2024 Kernkonzept GmbH.
00004  * Author(s): Martin Kuettler <martin.kuettler@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 #include <l4/re/error_helper>
00012 #include <l4/sys/cxx/ipc_epiface>
00013
00014 #include <l4/l4virtio/server/virtio>
00015 #include <l4/l4virtio/server/l4virtio>
00016 #include <l4/l4virtio/l4virtio>
00017
00018 namespace L4virtio {
00019 namespace Svr {
00020
00021     template <typename Rnd_state, typename Epiface = L4virtio::Device>

```

```

00033 class Virtio_rng : public L4virtio::Svr::Device,
00034                     public L4::Epiface_t<Virtio_rng<Rnd_state>, Epiface>
00035 {
00036 private:
00037     enum
00038     {
00039         Num_request_queues = 1,
00040         queue_size = 128,
00041     };
00042
00043 public:
00044     using Random_state = Rnd_state;
00045
00051     class Host_irq : public L4::Irqep_t<Host_irq>
00052     {
00053     public:
00054         explicit Host_irq(Virtio_rng *rng) : L4::Irqep_t<Host_irq>(), _rng(rng) {}
00055
00056         void handle_irq()
00057         {
00058             _rng->handle_queue();
00059         }
00060
00061     private:
00062         Virtio_rng *_rng;
00063     };
00064
00068     class Request_processor : public L4virtio::Svr::Request_processor
00069     {
00070     public:
00071         struct Data_buffer : public L4virtio::Svr::Data_buffer
00072         {
00073             Data_buffer() = default;
00074             // This constructor is called from within start, so make it available.
00075             Data_buffer(L4virtio::Svr::Driver_mem_region const *r,
00076                        L4virtio::Svr::Virtqueue::Desc const &d,
00077                        L4virtio::Svr::Request_processor const *)
00078             {
00079                 pos = static_cast<char *>(r->local(d.addr));
00080                 left = d.len;
00081             }
00082         };
00083
00084         Request_processor(L4virtio::Svr::Virtqueue *q, Random_state *rnd,
00085                          Virtio_rng *rng)
00086             : _q(q), _rnd(rnd), _rng(rng), _head() {}
00087
00088         bool init_queue()
00089         {
00090             auto r = _q->next_avail();
00091
00092             if (L4_UNLIKELY(!r))
00093                 return false;
00094
00095             _head = start(_rng->mem_info(), r, &_req);
00096
00097             return true;
00098         }
00099
00100         void handle_request()
00101         {
00102             if (!_head)
00103                 if (!init_queue())
00104                     return;
00105
00106             for (;;)
00107             {
00108                 auto const pos = reinterpret_cast<unsigned char *>(_req.pos);
00109                 _rnd->get_random(_req.left, pos);
00110                 _q->finish(_head, _rng, _req.left);
00111                 if (!init_queue())
00112                     break;
00113             }
00114             return;
00115         }
00116
00117     private:
00118         L4virtio::Svr::Virtqueue *_q;
00119         Random_state *_rnd;
00120         Virtio_rng *_rng;
00121         L4virtio::Svr::Virtqueue::Head_desc _head;
00122         Data_buffer _req;
00123     };
00124
00125     Virtio_rng(Random_state *rnd, L4::Registry_iface *registry)
00126         : L4virtio::Svr::Device(&_dev_config),
00127         _dev_config(L4VIRTIO_VENDOR_KK, L4VIRTIO_ID_RNG, Num_request_queues),

```

```

00128     _rnd(rnd),
00129     _host_irq(this),
00130     _request_processor(&_q, rnd, this)
00131 {
00132     init_mem_info(2);
00133     reset_queue_config(0, queue_size);
00134     setup_queue(&_q, 0, queue_size);
00135     registry->register_irq_obj(&_host_irq);
00136
00137     L4virtio::Svr::Dev_config::Features hf;
00138     hf.ring_indirect_desc() = true;
00139     _dev_config.host_features(0) = hf.raw;
00140     _dev_config.set_host_feature(L4VIRTIO_FEATURE_VERSION_1);
00141     _dev_config.reset_hdr();
00142 }
00143
00144 void notify_queue(L4virtio::Svr::Virtqueue *)
00145 {
00146     if (_q.no_notify_guest())
00147         return;
00148
00149     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_VRING);
00150     L4Re::chkipc(_notify_guest_irq->trigger(), "trigger guest irq");
00151 }
00152
00153 void handle_queue()
00154 {
00155     _request_processor.handle_request();
00156 }
00157
00158 void reset() override
00159 {
00160 }
00161
00162 bool check_queues() override
00163 {
00164     return true;
00165 }
00166
00167 int reconfig_queue(unsigned idx) override
00168 {
00169     if (idx != 0)
00170         return -L4_ERANGE;
00171
00172     setup_queue(&_q, 0, queue_size);
00173
00174     return L4_EOK;
00175 }
00176
00177 void trigger_driver_config_irq() override
00178 {
00179     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_CONFIG);
00180     _notify_guest_irq->trigger();
00181 }
00182
00183 L4::Ipc_svr::Server_iface *server_iface() const override
00184 {
00185     return L4::Epiface::server_iface();
00186 }
00187
00188 long op_set_status(L4virtio::Device::Rights r, unsigned status)
00189 {
00190     return L4virtio::Svr::Device::op_set_status(r, status);
00191 }
00192
00193 long op_config_queue(L4virtio::Device::Rights r, unsigned queue)
00194 {
00195     return L4virtio::Svr::Device::op_config_queue(r, queue);
00196 }
00197
00198 long op_device_config(L4virtio::Device::Rights r,
00199                      L4::Ipc::Cap<L4Re::Dataspace> &config_ds,
00200                      l4_addr_t &ds_offset)
00201 {
00202     return L4virtio::Svr::Device::op_device_config(r, config_ds, ds_offset);
00203 }
00204
00205 L4::Cap<L4::Irq> device_notify_irq() const override
00206 {
00207     return L4::cap_cast<L4::Irq>(_host_irq.obj_cap());
00208 }
00209
00210 void register_single_driver_irq() override
00211 {
00212     _notify_guest_irq = L4Re::chkcap
00213         (server_iface()->template rcv_cap<L4::Irq>(0));
00214 }

```

```

00215     L4Re::chksys(server_iface()->realloc_rcv_cap(0));
00216 }
00217
00218
00219 private:
00220     L4virtio::Svr::Dev_config_t<L4virtio::Svr::No_custom_data>_dev_config;
00221     Random_state *_rnd;
00222     L4virtio::Svr::Virtqueue _q;
00223     Host_irq _host_irq;
00224     L4::Cap<L4::Irq> _notify_guest_irq;
00225     Request_processor _request_processor;
00226 };
00227
00228 } // namespace Svr
00229 } // namespace L4virtio

```

## 17.261 virtio-scmi-device

```

00001 // vi:ft=c++
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2024 Kernkonzept GmbH.
00005  * Author(s): Christian Pötzsch <christian.poetzsch@kernkonzept.com>
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include <l4/cxx/bitmap>
00012 #include <l4/l4virtio/l4virtio>
00013 #include <l4/l4virtio/server/l4virtio>
00014 #include <l4/l4virtio/server/virtio>
00015 #include <l4/re/util/object_registry>
00016
00017 #include <map>
00018 #include <vector>
00019
00020 namespace L4virtio { namespace Svr { namespace Scmi {
00021
00022     enum
00023     {
00024         Version = 0x20000
00025     };
00026
00027     enum
00028     {
00029         Success = 0,
00030         Not_supported = -1,
00031         Invalid_parameters = -2,
00032         Denied = -3,
00033         Not_found = -4,
00034         Out_of_range = -5,
00035         Busy = -6,
00036         Comms_error = -7,
00037         Generic_error = -8,
00038         Hardware_error = -9,
00039         Protocol_error = -10
00040     };
00041
00042     struct Scmi_hdr_t
00043     {
00044         l4_uint32_t hdr_raw = 0;
00045         CXX_BITFIELD_MEMBER(18, 27, token, hdr_raw);
00046         CXX_BITFIELD_MEMBER(10, 17, protocol_id, hdr_raw);
00047         CXX_BITFIELD_MEMBER( 8,  9, message_type, hdr_raw);
00048         CXX_BITFIELD_MEMBER( 0,  7, message_id, hdr_raw);
00049     };
00050
00051     enum
00052     {
00053         Base_protocol = 0x10,
00054         Power_domain_management_protocol = 0x11,
00055         System_power_management_protocol = 0x12,
00056         Performance_domain_management_protocol = 0x13,
00057         Clock_management_protocol = 0x14,
00058         Sensor_management_protocol = 0x15,
00059         Reset_domain_management_protocol = 0x16,
00060         Voltage_domain_management_protocol = 0x17
00061     };
00062
00063     enum
00064     {
00065         Protocol_version = 0x0,

```



```

00071 Protocol_attributes = 0x1,
00072 Protocol_message_attributes = 0x2,
00073 };
00074
00076 enum
00077 {
00078     Base_discover_vendor = 0x3,
00079     Base_discover_sub_vendor = 0x4,
00080     Base_discover_implementation_version = 0x5,
00081     Base_discover_list_protocols = 0x6,
00082     Base_discover_agent = 0x7,
00083     Base_notify_errors = 0x8,
00084     Base_set_device_permissions = 0x9,
00085     Base_set_protocol_permissions = 0xa,
00086     Base_reset_agent_configuration = 0xb
00087 };
00088
00090 struct Base_attr_t
00091 {
00092     14_uint32_t attr_raw = 0;
00093     CXX_BITFIELD_MEMBER(8, 15, nagents, attr_raw);
00094     CXX_BITFIELD_MEMBER(0, 7, nprots, attr_raw);
00095 };
00096
00098 enum
00099 {
00100     Performance_domain_attributes = 0x3,
00101     Performance_describe_levels = 0x4,
00102     Performance_limits_set = 0x5,
00103     Performance_limits_get = 0x6,
00104     Performance_level_set = 0x7,
00105     Performance_level_get = 0x8,
00106     Performance_notify_limits = 0x9,
00107     Performance_notify_level = 0xa,
00108     Performance_describe_fastchannel = 0xb,
00109 };
00110
00112 struct Performance_attr_t
00113 {
00114     14_uint32_t attr_raw = 0;
00115     CXX_BITFIELD_MEMBER(16, 16, power, attr_raw);
00116     CXX_BITFIELD_MEMBER(0, 15, domains, attr_raw);
00117
00118     14_uint32_t stat_addr_low = 0;
00119     14_uint32_t stat_addr_high = 0;
00120     14_uint32_t stat_len = 0;
00121 };
00122
00124 struct Performance_domain_attr_t
00125 {
00126     14_uint32_t attr_raw = 0;
00127     CXX_BITFIELD_MEMBER(31, 31, set_limits, attr_raw);
00128     CXX_BITFIELD_MEMBER(30, 30, set_perf_level, attr_raw);
00129     CXX_BITFIELD_MEMBER(29, 29, perf_limits_change_notify, attr_raw);
00130     CXX_BITFIELD_MEMBER(28, 28, perf_level_change_notify, attr_raw);
00131     CXX_BITFIELD_MEMBER(27, 27, fast_channel, attr_raw);
00132
00133     14_uint32_t rate_limit_raw = 0;
00134     CXX_BITFIELD_MEMBER(0, 19, rate_limit, rate_limit_raw);
00135
00136     14_uint32_t sustained_freq = 0;
00137     14_uint32_t sustained_perf_level = 0;
00138     char name[16] = { 0 };
00139 };
00140
00142 struct Performance_describe_levels_n_t
00143 {
00144     14_uint32_t num_levels_raw = 0;
00145     CXX_BITFIELD_MEMBER(16, 31, nremain_perf_levels, num_levels_raw);
00146     CXX_BITFIELD_MEMBER(0, 11, nperf_levels, num_levels_raw);
00147 };
00148
00150 struct Performance_describe_level_t
00151 {
00152     14_uint32_t perf_level = 0;
00153     14_uint32_t power_cost = 0;
00154     14_uint16_t trans_latency = 0;
00155     14_uint16_t res0 = 0;
00156 };
00157
00158 template<typename OBSERV>
00159 struct Queue_worker : Request_processor
00160 {
00161     Queue_worker(OBSERV *o, Virtqueue *queue)
00162     : o(o), q(queue)
00163     {}
00164

```

```

00165 bool init_queue()
00166 {
00167     auto r = q->next_avail();
00168
00169     if (L4_UNLIKELY(!r))
00170         return false;
00171
00172     head = start(o->mem_info(), r, &req);
00173
00174     return true;
00175 }
00176
00177 bool next()
00178 { return Request_processor::next(o->mem_info(), &req); }
00179
00180 void finish(l4_uint32_t total)
00181 { q->finish(head, o, total); }
00182
00183 template<typename T>
00184 l4_ssize_t read(Data_buffer *buf, T *data, l4_size_t s = sizeof(T))
00185 {
00186     buf->pos = reinterpret_cast<char *>(data);
00187     buf->left = s;
00188     l4_size_t chunk = 0;
00189     for (;;)
00190     {
00191         chunk += req.copy_to(buf);
00192         if (req.done())
00193             next();
00194         if (!buf->left)
00195             break;
00196     }
00197     if (chunk != s)
00198         return -1;
00199     return chunk;
00200 }
00201
00202 template<typename T>
00203 l4_ssize_t write(Data_buffer *buf, T *data, l4_size_t s = sizeof(T))
00204 {
00205     buf->pos = reinterpret_cast<char *>(data);
00206     buf->left = s;
00207     l4_size_t chunk = 0;
00208     for (;;)
00209     {
00210         chunk += buf->copy_to(&req);
00211         if (req.done())
00212             next();
00213         if (!buf->left)
00214             break;
00215     }
00216     if (chunk != s)
00217         return -1;
00218     return chunk;
00219 }
00220
00221 l4_ssize_t handle_request()
00222 {
00223     try
00224     {
00225         if (!head && L4_UNLIKELY(!init_queue()))
00226             return 0;
00227
00228         for (;;)
00229         {
00230             l4_ssize_t total = 0;
00231             l4_ssize_t res = 0;
00232             Scmi_hdr_t hdr;
00233             Data_buffer buf = Data_buffer(&hdr);
00234             if ((res = read(&buf, &hdr)) < 0)
00235                 return res;
00236
00237             // Search/execute handler for given protocol
00238             auto proto = o->proto(hdr.protocol_id());
00239             if (proto)
00240             {
00241                 if ((res = proto->handle_request(hdr, buf, this)) < 0)
00242                     return res;
00243                 total += res;
00244             }
00245             else
00246             {
00247                 if ((res = write(&buf, &hdr)) < 0)
00248                     return res;
00249                 total += res;
00250
00251                 l4_int32_t status = Not_supported;

```

```

00252         if ((res = write(&buf, &status)) < 0)
00253             return res;
00254         total += res;
00255     }
00256
00257     finish(total);
00258
00259     head = Virtqueue::Head_desc();
00260     if (L4_UNLIKELY(!init_queue()))
00261         return 0;
00262     }
00263 }
00264 catch (L4virtio::Svr::Bad_descriptor const &e)
00265 {
00266     return e.error;
00267 }
00268 return 0;
00269 }
00270
00271 private:
00272 struct Buffer : Data_buffer
00273 {
00274     Buffer() = default;
00275     Buffer(L4virtio::Svr::Driver_mem_region const *r,
00276           Virtqueue::Desc const &d, Request_processor const *)
00277     {
00278         pos = static_cast<char *>(r->local(d.addr));
00279         left = d.len;
00280     }
00281 };
00282
00283 Virtqueue::Head_desc head;
00284 Buffer req;
00285
00286 OBSERV *o;
00287 Virtqueue *q;
00288 };
00289
00290 template<typename OBSERV>
00291 struct Proto
00292 {
00293     virtual l4_ssize_t handle_request(Scmi_hdr_t &hdr,
00294                                       Data_buffer &buf,
00295                                       Queue_worker<OBSERV> *qw) = 0;
00296 };
00297
00298 class Scmi_dev : public L4virtio::Svr::Device
00299 {
00300     struct Features : L4virtio::Svr::Dev_config::Features
00301     {
00302         Features() = default;
00303         Features(l4_uint32_t raw) : L4virtio::Svr::Dev_config::Features(raw) {}
00304     };
00305
00306     struct Host_irq : public L4::Irqep_t<Host_irq>
00307     {
00308         Scmi_dev *c;
00309         explicit Host_irq(Scmi_dev *c) : c(c) {}
00310         void handle_irq() { c->kick(); }
00311     };
00312
00313     enum
00314     {
00315         Queue_size = 0x10
00316     };
00317
00318 public:
00319     Scmi_dev(L4Re::Util::Object_registry *registry)
00320     : L4virtio::Svr::Device(&_dev_config),
00321       _dev_config(L4VIRTIO_VENDOR_KK, L4VIRTIO_ID_SCMI, 1),
00322       _host_irq(this),
00323       _request_worker(this, &q[0])
00324     {
00325         init_mem_info(2);
00326
00327         L4Re::chkcap(registry->register_irq_obj(&_host_irq),
00328                     "Register irq object");
00329
00330         Features hf(0);
00331         hf.ring_indirect_desc() = true;
00332
00333         _dev_config.host_features(0) = hf.raw;
00334
00335         _dev_config.set_host_feature(L4VIRTIO_FEATURE_VERSION_1);
00336         _dev_config.reset_hdr();
00337
00338         reset();
00339     }

```

```

00377     }
00378
00380 void add_proto(l4_uint32_t id, Proto<Scmi_dev> *proto)
00381 { _protos.insert({id, proto}); }
00382
00383 Proto<Scmi_dev> *proto(l4_uint32_t id) const
00384 {
00385     if (_protos.find(id) != _protos.end())
00386         return _protos.at(id);
00387     return nullptr;
00388 }
00389
00390 void notify_queue(L4virtio::Virtqueue *queue)
00391 {
00392     if (queue->no_notify_guest())
00393         return;
00394
00395     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_VRING);
00396     _kick_guest_irq->trigger();
00397 }
00398
00399 private:
00400 L4::Cap<L4::Irq> device_notify_irq() const override
00401 { return L4::cap_cast<L4::Irq>(_host_irq.obj_cap()); }
00402
00403 void register_single_driver_irq() override
00404 {
00405     _kick_guest_irq = L4Re::Util::Unique_cap<L4::Irq>(
00406         L4Re::chkcap(server_iface()->template rcv_cap<L4::Irq>(0)));
00407
00408     L4Re::chksys(server_iface()->realloc_rcv_cap(0));
00409 }
00410
00411 void kick()
00412 {
00413     if (_request_worker.handle_request() < 0)
00414         device_error();
00415 }
00416
00417 void reset() override
00418 {
00419     for (Virtqueue &q : _q)
00420         q.disable();
00421
00422     for (l4_uint32_t i = 0; i < _dev_config.num_queues(); i++)
00423         reset_queue_config(i, Queue_size);
00424 }
00425
00426 bool check_queues() override
00427 {
00428     return true;
00429 }
00430
00431 int reconfig_queue(unsigned idx) override
00432 {
00433     if (idx >= sizeof(_q) / sizeof(_q[0]))
00434         return -L4_ERANGE;
00435
00436     return setup_queue(_q + idx, idx, Queue_size);
00437 }
00438
00439 void trigger_driver_config_irq() override
00440 {
00441     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_CONFIG);
00442     _kick_guest_irq->trigger();
00443 }
00444
00445 L4virtio::Svr::Dev_config_t<L4virtio::Svr::No_custom_data> _dev_config;
00446 Host_irq _host_irq;
00447 L4Re::Util::Unique_cap<L4::Irq> _kick_guest_irq;
00448 Virtqueue _q[1];
00449 Queue_worker<Scmi_dev> _request_worker;
00450 std::map<l4_uint32_t, Proto<Scmi_dev> *> _protos;
00451 };
00452
00458 class Base_proto : public Proto<Scmi_dev>
00459 {
00460     virtual l4_int32_t fill_attr(Base_attr_t *attr) const = 0;
00461
00462     virtual std::vector<l4_uint32_t> prots() const = 0;
00463
00464     l4_ssize_t handle_request(Scmi_hdr_t &hdr, Data_buffer &buf,
00465                               Queue_worker<Scmi_dev> *qw) override
00466     {
00467         l4_ssize_t total = 0;
00468         l4_ssize_t res = 0;
00469         switch (hdr.message_id())

```

```
00473     {
00474     case Protocol_version:
00475     {
00476         if ((res = qw->write(&buf, &hdr)) < 0)
00477             return res;
00478         total += res;
00479
00480         struct
00481         {
00482             14_int32_t status = Success;
00483             14_uint32_t version = Version;
00484         } version;
00485         if ((res = qw->write(&buf, &version)) < 0)
00486             return res;
00487         total += res;
00488         break;
00489     }
00490     case Protocol_attributes:
00491     {
00492         if ((res = qw->write(&buf, &hdr)) < 0)
00493             return res;
00494         total += res;
00495
00496         Base_attr_t ba;
00497         14_int32_t status = fill_attr(&ba);
00498         if ((res = qw->write(&buf, &status)) < 0)
00499             return res;
00500         total += res;
00501         if (status == Success)
00502         {
00503             if ((res = qw->write(&buf, &ba)) < 0)
00504                 return res;
00505             total += res;
00506         }
00507         break;
00508     }
00509     case Protocol_message_attributes:
00510     {
00511         14_uint32_t msg_id = 0;
00512         if ((res = qw->read(&buf, &msg_id)) < 0)
00513             return res;
00514
00515         if ((res = qw->write(&buf, &hdr)) < 0)
00516             return res;
00517         total += res;
00518
00519         struct
00520         {
00521             14_int32_t status = Not_found;
00522             14_uint32_t attr = 0;
00523         } attr;
00524         if (msg_id >= Protocol_version &&
00525             msg_id <= Base_discover_list_protocols)
00526             attr.status = Success;
00527
00528         if ((res = qw->write(&buf, &attr)) < 0)
00529             return res;
00530         total += res;
00531         break;
00532     }
00533     case Base_discover_vendor:
00534     {
00535         if ((res = qw->write(&buf, &hdr)) < 0)
00536             return res;
00537         total += res;
00538
00539         struct
00540         {
00541             14_int32_t status = Success;
00542             14_uint8_t vendor_identifier[16] = { "L4Re" };
00543         } vendor;
00544         if ((res = qw->write(&buf, &vendor)) < 0)
00545             return res;
00546         total += res;
00547         break;
00548     }
00549     case Base_discover_sub_vendor:
00550     {
00551         if ((res = qw->write(&buf, &hdr)) < 0)
00552             return res;
00553         total += res;
00554
00555         struct
00556         {
00557             14_int32_t status = Success;
00558             14_uint8_t vendor_identifier[16] = { "Scmi" };
00559         } vendor;
```

```

00560         if ((res = qw->write(&buf, &vendor)) < 0)
00561             return res;
00562         total += res;
00563         break;
00564     }
00565     case Base_discover_implementation_version:
00566     {
00567         if ((res = qw->write(&buf, &hdr)) < 0)
00568             return res;
00569         total += res;
00570
00571         struct
00572         {
00573             l4_int32_t status = Success;
00574             l4_uint32_t version = 1;
00575         } version;
00576         if ((res = qw->write(&buf, &version)) < 0)
00577             return res;
00578         total += res;
00579         break;
00580     }
00581     case Base_discover_list_protocols:
00582     {
00583         l4_uint32_t skip = 0;
00584         if ((res = qw->read(&buf, &skip)) < 0)
00585             return res;
00586
00587         if ((res = qw->write(&buf, &hdr)) < 0)
00588             return res;
00589         total += res;
00590
00591         auto p = prots();
00592         struct
00593         {
00594             l4_int32_t status = Success;
00595             l4_uint32_t num;
00596         } proto;
00597         proto.num = p.size();
00598         if ((res = qw->write(&buf, &proto)) < 0)
00599             return res;
00600         total += res;
00601
00602         // Array of uint32 where 4 protos are packed into one uint32. So
00603         // round up to 4 bytes and fill the array byte by byte.
00604         l4_uint8_t parr[(p.size() + 3) / 4 * 4] = { 0 };
00605         for (l4_size_t i = 0; i < p.size(); i++)
00606             parr[i] = p.at(i);
00607
00608         if ((res = qw->write(&buf, parr, sizeof(parr))) < 0)
00609             return res;
00610         total += res;
00611         break;
00612     }
00613     default:
00614     {
00615         if ((res = qw->write(&buf, &hdr)) < 0)
00616             return res;
00617         total += res;
00618
00619         l4_int32_t status = Not_supported;
00620         if ((res = qw->write(&buf, &status)) < 0)
00621             return res;
00622         total += res;
00623         break;
00624     }
00625 }
00626
00627 return total;
00628 }
00629 };
00630
00662 class Perf_proto : public Proto<Scmi_dev>
00663 {
00664     virtual l4_int32_t fill_attr(Performance_attr_t *attr) const = 0;
00665
00666     virtual l4_int32_t fill_domain_attr(l4_uint32_t domain_id,
00667                                         Performance_domain_attr_t *attr) const = 0;
00668
00669     virtual l4_int32_t fill_describe_levels_n(l4_uint32_t domain_id,
00670                                                l4_uint32_t level_idx,
00671                                                Performance_describe_levels_n_t *attr) const = 0;
00672
00673     virtual l4_int32_t fill_describe_levels(l4_uint32_t domain_id,
00674                                              l4_uint32_t level_idx,
00675                                              l4_uint32_t num,
00676                                              Performance_describe_level_t *attr) const = 0;
00677
00678
00679
00680
00681
00682
00683
00684

```

```

00686 virtual l4_int32_t level_set(l4_uint32_t domain_id,
00687                             l4_uint32_t perf_level) = 0;
00688
00690 virtual l4_int32_t level_get(l4_uint32_t domain_id,
00691                             l4_uint32_t *perf_level) const = 0;
00692
00693 l4_ssize_t handle_request(Scmi_hdr_t &hdr, Data_buffer &buf,
00694                          Queue_worker<Scmi_dev> *qw) override
00695 {
00696     l4_ssize_t total = 0;
00697     l4_ssize_t res = 0;
00698     switch (hdr.message_id())
00699     {
00700     case Protocol_version:
00701     {
00702         if ((res = qw->write(&buf, &hdr)) < 0)
00703             return res;
00704         total += res;
00705
00706         struct
00707         {
00708             l4_int32_t status = Success;
00709             l4_uint32_t version = Version;
00710         } version;
00711         if ((res = qw->write(&buf, &version)) < 0)
00712             return res;
00713         total += res;
00714         break;
00715     }
00716     case Protocol_attributes:
00717     {
00718         if ((res = qw->write(&buf, &hdr)) < 0)
00719             return res;
00720         total += res;
00721
00722         Performance_attr_t pa;
00723         l4_int32_t status = fill_attr(&pa);
00724         if ((res = qw->write(&buf, &status)) < 0)
00725             return res;
00726         total += res;
00727         if (status == Success)
00728         {
00729             if ((res = qw->write(&buf, &pa)) < 0)
00730                 return res;
00731             total += res;
00732         }
00733         break;
00734     }
00735     case Protocol_message_attributes:
00736     {
00737         l4_uint32_t msg_id = 0;
00738         if ((res = qw->read(&buf, &msg_id)) < 0)
00739             return res;
00740
00741         if ((res = qw->write(&buf, &hdr)) < 0)
00742             return res;
00743         total += res;
00744
00745         struct
00746         {
00747             l4_int32_t status = Not_found;
00748
00749             l4_uint32_t attr_raw = 0;
00750             CXX_BITFIELD_MEMBER(0, 0, fast_channel, attr_raw); // ignored
00751         } attr;
00752         if ((msg_id >= Protocol_version &&
00753             msg_id <= Performance_describe_levels) ||
00754             (msg_id >= Performance_level_set &&
00755             msg_id <= Performance_level_get))
00756             attr.status = Success;
00757
00758         if ((res = qw->write(&buf, &attr)) < 0)
00759             return res;
00760         total += res;
00761         break;
00762     }
00763     case Performance_domain_attributes:
00764     {
00765         l4_uint32_t domain_id = 0;
00766         if ((res = qw->read(&buf, &domain_id)) < 0)
00767             return res;
00768
00769         if ((res = qw->write(&buf, &hdr)) < 0)
00770             return res;
00771         total += res;
00772
00773         Performance_domain_attr_t attr;

```

```

00774         l4_int32_t status = fill_domain_attr(domain_id, &attr);
00775         if ((res = qw->write(&buf, &status)) < 0)
00776             return res;
00777         total += res;
00778         if (status == Success)
00779         {
00780             if ((res = qw->write(&buf, &attr)) < 0)
00781                 return res;
00782             total += res;
00783         }
00784         break;
00785     }
00786 case Performance_describe_levels:
00787     {
00788         struct
00789         {
00790             l4_uint32_t domain_id = 0;
00791             l4_uint32_t level_idx = 0;
00792         } param;
00793         if ((res = qw->read(&buf, &param)) < 0)
00794             return res;
00795
00796         if ((res = qw->write(&buf, &hdr)) < 0)
00797             return res;
00798         total += res;
00799
00800         // First figure out how many levels we support
00801         Performance_describe_levels_n_t attr;
00802         l4_int32_t status = fill_describe_levels_n(param.domain_id, param.level_idx,
00803                                                    &attr);
00804         if (status != Success)
00805         {
00806             // On error bail out early
00807             if ((res = qw->write(&buf, &status)) < 0)
00808                 return res;
00809             total += res;
00810             break;
00811         }
00812
00813         // Now fetch the actual levels
00814         Performance_describe_level_t attr1[attr.nperf_levels().get()];
00815         status = fill_describe_levels(param.domain_id, param.level_idx,
00816                                       attr.nperf_levels(), attr1);
00817         if ((res = qw->write(&buf, &status)) < 0)
00818             return res;
00819         total += res;
00820         if (status == Success)
00821         {
00822             // Write both answers to the client
00823             if ((res = qw->write(&buf, &attr)) < 0)
00824                 return res;
00825             total += res;
00826             if ((res = qw->write(&buf, attr1, sizeof(attr1))) < 0)
00827                 return res;
00828             total += res;
00829         }
00830         break;
00831     }
00832 case Performance_level_set:
00833     {
00834         struct
00835         {
00836             l4_uint32_t domain_id;
00837             l4_uint32_t perf_level;
00838         } param;
00839         if ((res = qw->read(&buf, &param)) < 0)
00840             return res;
00841
00842         if ((res = qw->write(&buf, &hdr)) < 0)
00843             return res;
00844         total += res;
00845
00846         l4_int32_t status = level_set(param.domain_id, param.perf_level);
00847         if ((res = qw->write(&buf, &status)) < 0)
00848             return res;
00849         total += res;
00850         break;
00851     }
00852 case Performance_level_get:
00853     {
00854         l4_uint32_t domain_id = 0;
00855         if ((res = qw->read(&buf, &domain_id)) < 0)
00856             return res;
00857
00858         if ((res = qw->write(&buf, &hdr)) < 0)
00859             return res;
00860         total += res;

```



```

00861
00862     l4_uint32_t perf_level;
00863     l4_int32_t status = level_get(domain_id, &perf_level);
00864     if ((res = qw->write(&buf, &status)) < 0)
00865         return res;
00866     total += res;
00867     if (status == Success)
00868     {
00869         if ((res = qw->write(&buf, &perf_level)) < 0)
00870             return res;
00871         total += res;
00872     }
00873     break;
00874 }
00875 default:
00876 {
00877     if ((res = qw->write(&buf, &hdr)) < 0)
00878         return res;
00879     total += res;
00880
00881     l4_int32_t status = Not_supported;
00882     if ((res = qw->write(&buf, &status)) < 0)
00883         return res;
00884     total += res;
00885     break;
00886 }
00887 }
00888 return total;
00889 }
00890 };
00891
00892 } /* Scmi */ } /* Svr */ } /* L4virtio */

```

## 17.262 virtio.h

```

00001 /* SPDX-License-Identifier: MIT */
00002 /*
00003  * Copyright (C) 2013-2022, 2024 Kernkonzept GmbH.
00004  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00005  *             Matthias Lange <matthias.lange@kernkonzept.com>
00006  *
00007  */
00008 #pragma once
00009
00010
00013
00027
00028 #include <l4/sys/compiler.h>
00029 #include <l4/sys/utcb.h>
00030 #include <l4/sys/ipc.h>
00031 #include <l4/sys/types.h>
00032
00034 enum L4_virtio_protocol
00035 {
00036     L4VIRTIO_PROTOCOL = 0,
00037 };
00038
00039 enum L4virtio_magic
00040 {
00041     L4VIRTIO_MAGIC = 0x74726976
00042 };
00043
00044 enum L4virtio_vendor
00045 {
00046     L4VIRTIO_VENDOR_KK = 0x44
00047 };
00048
00052 enum L4_virtio_opcodes
00053 {
00054     L4VIRTIO_OP_SET_STATUS          = 0,
00055     L4VIRTIO_OP_CONFIG_QUEUE        = 1,
00056     L4VIRTIO_OP_REGISTER_DS         = 3,
00057     L4VIRTIO_OP_DEVICE_CONFIG       = 4,
00058     L4VIRTIO_OP_GET_DEVICE_IRQ      = 5,
00059 };
00060
00062 enum L4virtio_device_ids
00063 {
00064     L4VIRTIO_ID_NET                 = 1,
00065     L4VIRTIO_ID_BLOCK                = 2,
00066     L4VIRTIO_ID_CONSOLE              = 3,
00067     L4VIRTIO_ID_RNG                  = 4,
00068     L4VIRTIO_ID_BALLOON              = 5,
00069     L4VIRTIO_ID_RPMMSG               = 7,

```

```

00070 L4VIRTIO_ID_SCSI          = 8,
00071 L4VIRTIO_ID_9P           = 9,
00072 L4VIRTIO_ID_RPROC_SERIAL = 11,
00073 L4VIRTIO_ID_CAIF         = 12,
00074 L4VIRTIO_ID_GPU          = 16,
00075 L4VIRTIO_ID_INPUT       = 18,
00076 L4VIRTIO_ID_VSOCK        = 19,
00077 L4VIRTIO_ID_CRYPT        = 20,
00078 L4VIRTIO_ID_FS           = 26,
00079 L4VIRTIO_ID_SCSI         = 32,
00080 L4VIRTIO_ID_I2C          = 34,
00081 L4VIRTIO_ID_GPIO        = 41,
00082
00083 L4VIRTIO_ID_SOCK          = 0x9999,
00084 };
00085
00087 enum L4virtio_device_status
00088 {
00089     L4VIRTIO_STATUS_ACKNOWLEDGE = 1,
00090     L4VIRTIO_STATUS_DRIVER      = 2,
00091     L4VIRTIO_STATUS_DRIVER_OK   = 4,
00092     L4VIRTIO_STATUS_FEATURES_OK = 8,
00093     L4VIRTIO_STATUS_DEVICE_NEEDS_RESET = 0x40,
00094     L4VIRTIO_STATUS_FAILED      = 0x80
00095 };
00096
00098 enum L4virtio_feature_bits
00099 {
00101     L4VIRTIO_FEATURE_VERSION_1 = 32,
00103     L4VIRTIO_FEATURE_CMD_CONFIG = 160
00104 };
00105
00110 enum L4_virtio_irq_status
00111 {
00112     L4VIRTIO_IRQ_STATUS_VRING = 1,
00113     L4VIRTIO_IRQ_STATUS_CONFIG = 2,
00114 };
00115
00119 enum L4_virtio_cmd
00120 {
00121     L4VIRTIO_CMD_NONE          = 0x00000000,
00122     L4VIRTIO_CMD_SET_STATUS    = 0x01000000,
00123     L4VIRTIO_CMD_CFG_QUEUE     = 0x02000000,
00124     L4VIRTIO_CMD_CFG_CHANGED   = 0x04000000,
00125     L4VIRTIO_CMD_NOTIFY_QUEUE  = 0x08000000,
00126     L4VIRTIO_CMD_MASK          = 0xff000000,
00127 };
00128
00132 typedef struct l4virtio_config_hdr_t
00133 {
00134     /* Virtio(0x00): device config */
00135     l4_uint32_t magic;
00136     l4_uint32_t version;
00137     l4_uint32_t device;
00138     l4_uint32_t vendor;
00139
00140     /* Virtio(0x10): device features */
00141     l4_uint32_t dev_features;
00142     l4_uint32_t dev_features_sel;
00143     l4_uint32_t _res1[2];
00144
00145     /* Virtio(0x20): driver features */
00146     l4_uint32_t driver_features;
00147     l4_uint32_t driver_features_sel;
00148
00149     /* L4Virtio(0x28): L4 queue */
00150     l4_uint32_t num_queues;
00151     l4_uint32_t queues_offset;
00152
00153     /* Virtio(0x30): queue status */
00154     l4_uint32_t queue_sel;
00155     l4_uint32_t queue_num_max;
00156     l4_uint32_t queue_num;
00157     l4_uint32_t _res3[2];
00158     l4_uint32_t queue_ready;
00159     l4_uint32_t _res4[2];
00160
00161     /* Virtio(0x50): queue notify */
00162     l4_uint32_t queue_notify;
00163     l4_uint32_t _res5[3];
00164
00165     /* Virtio(0x60): interrupt handling */
00166     l4_uint32_t irq_status;
00167     l4_uint32_t irq_ack;
00168     l4_uint32_t _res6[2];
00169
00170     /* Virtio(0x70): Device status register (read-only). The register must be

```

```

00171     * written using l4virtio_set_status(). */
00172     l4_uint32_t status;
00173
00174     /* L4Virtio(0x74): W: Event index to be used for config notifications (device to driver) */
00175     l4_uint32_t cfg_driver_notify_index;
00176     /* L4Virtio(0x78): R: Event index to be used for config notifications (driver to device) */
00177     l4_uint32_t cfg_device_notify_index;
00178
00179     /* L4Virtio(0x7c) L4 specific command register polled by the driver iff supported */
00180     l4_uint32_t cmd;
00181
00182     /* Virtio(0x80): queue descriptors */
00183     l4_uint64_t queue_desc;
00184     l4_uint32_t _res8[2];
00185     l4_uint64_t queue_avail;
00186     l4_uint32_t _res9[2];
00187     l4_uint64_t queue_used;
00188
00189     l4_uint32_t _res10[1];
00190
00191     /* Virtio(0xac): shared memory region */
00192     l4_uint32_t shm_sel;
00193     l4_uint64_t shm_len;
00194     l4_uint64_t shm_base;
00195
00196     /* L4Virtio(0xc0): use the unused space here for device and driver feature bitmaps */
00197     l4_uint32_t dev_features_map[6];
00198     l4_uint32_t _res11[2];
00199     l4_uint32_t driver_features_map[6];
00200     l4_uint32_t _res12[1];
00201
00202     /* Virtio(0xfc): config generation */
00203     l4_uint32_t generation;
00204 } l4virtio_config_hdr_t;
00205
00223 typedef struct l4virtio_config_queue_t
00224 {
00226     l4_uint16_t num_max;
00228     l4_uint16_t num;
00229
00231     l4_uint16_t ready;
00232
00234     l4_uint16_t driver_notify_index;
00235
00236     l4_uint64_t desc_addr;
00237     l4_uint64_t avail_addr;
00238     l4_uint64_t used_addr;
00239
00241     l4_uint16_t device_notify_index;
00242 } l4virtio_config_queue_t;
00243
00244 L4_BEGIN_DECLS
00245
00251 L4_INLINE l4virtio_config_queue_t *
00252 l4virtio_config_queues(l4virtio_config_hdr_t const *cfg)
00253 {
00254     return (l4virtio_config_queue_t *)(((l4_addr_t)cfg) + cfg->queues_offset);
00255 }
00256
00262 L4_INLINE void *
00263 l4virtio_device_config(l4virtio_config_hdr_t const *cfg)
00264 {
00265     return (void *)(((l4_addr_t)cfg) + 0x100);
00266 }
00267
00271 L4_INLINE void
00272 l4virtio_set_feature(l4_uint32_t *feature_map, unsigned feat)
00273 {
00274     unsigned idx = feat / 32;
00275
00276     if (idx < 8)
00277         feature_map[idx] |= 1UL << (feat % 32);
00278 }
00279
00283 L4_INLINE void
00284 l4virtio_clear_feature(l4_uint32_t *feature_map, unsigned feat)
00285 {
00286     unsigned idx = feat / 32;
00287
00288     if (idx < 8)
00289         feature_map[idx] &= ~(1UL << (feat % 32));
00290 }
00291
00295 L4_INLINE unsigned
00296 l4virtio_get_feature(l4_uint32_t *feature_map, unsigned feat)
00297 {
00298     unsigned idx = feat / 32;

```

```

00299
00300     if (idx >= 8)
00301         return 0;
00302
00303     return feature_map[idx] & (1UL << (feat % 32));
00304 }
00305
00311 L4_CV int
00312 l4virtio_set_status(l4_cap_idx_t cap, unsigned status) L4_NOTHROW;
00313
00319 L4_CV int
00320 l4virtio_config_queue(l4_cap_idx_t cap, unsigned queue) L4_NOTHROW;
00321
00327 L4_CV int
00328 l4virtio_register_ds(l4_cap_idx_t cap, l4_cap_idx_t ds_cap,
00329                     l4_uint64_t base, l4_umword_t offset,
00330                     l4_umword_t size) L4_NOTHROW;
00331
00337 L4_CV int
00338 l4virtio_device_config_ds(l4_cap_idx_t cap, l4_cap_idx_t config_ds,
00339                          l4_addr_t *ds_offset) L4_NOTHROW;
00340
00346 L4_CV int
00347 l4virtio_device_notification_irq(l4_cap_idx_t cap, unsigned index,
00348                                  l4_cap_idx_t irq) L4_NOTHROW;
00349
00350 L4_END_DECLS
00351

```

## 17.263 virtio\_block.h

```

00001 /* SPDX-License-Identifier: MIT */
00002 /*
00003  * (c) 2014 Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  */
00005
00006 #pragma once
00007
00013
00014 #include <l4/sys/types.h>
00015
00019 enum L4virtio_block_operations
00020 {
00021     L4VIRTIO_BLOCK_T_IN      = 0,
00022     L4VIRTIO_BLOCK_T_OUT     = 1,
00023     L4VIRTIO_BLOCK_T_FLUSH   = 4,
00024     L4VIRTIO_BLOCK_T_GET_ID   = 8,
00025     L4VIRTIO_BLOCK_T_DISCARD = 11,
00026     L4VIRTIO_BLOCK_T_WRITE_ZEROES = 13,
00027 };
00028
00032 enum L4virtio_block_status
00033 {
00034     L4VIRTIO_BLOCK_S_OK      = 0,
00035     L4VIRTIO_BLOCK_S_IOERR   = 1,
00036     L4VIRTIO_BLOCK_S_UNSUPP   = 2,
00037 };
00038
00042 typedef struct l4virtio_block_header_t
00043 {
00044     l4_uint32_t type;
00045     l4_uint32_t ioprio;
00046     l4_uint64_t sector;
00047 } l4virtio_block_header_t;
00048
00049 enum L4virtio_block_discard_flags_t
00050 {
00051     L4VIRTIO_BLOCK_DISCARD_F_UNMAP   = 0x00000001UL,
00052     L4VIRTIO_BLOCK_DISCARD_F_RESERVED = 0xFFFFFFFFUL,
00053 };
00054
00058 typedef struct l4virtio_block_discard_t
00059 {
00060     l4_uint64_t sector;
00061     l4_uint32_t num_sectors;
00062     l4_uint32_t flags;
00063 } l4virtio_block_discard_t;
00064
00068 typedef struct l4virtio_block_config_t
00069 {
00070     l4_uint64_t capacity;
00071     l4_uint32_t size_max;
00072     l4_uint32_t seg_max;

```

```

00073 struct l4virtio_block_config_geometry_t
00074 {
00075     l4_uint16_t cylinders;
00076     l4_uint8_t heads;
00077     l4_uint8_t sectors;
00078 } geometry;
00079 l4_uint32_t blk_size;
00080 struct l4virtio_block_config_topology_t
00081 {
00083     l4_uint8_t physical_block_exp;
00085     l4_uint8_t alignment_offset;
00087     l4_uint16_t min_io_size;
00089     l4_uint32_t opt_io_size;
00090 } topology;
00091 l4_uint8_t writeback;
00092 l4_uint8_t unused0[1];
00093 l4_uint16_t num_queues;
00094 l4_uint32_t max_discard_sectors;
00095 l4_uint32_t max_discard_seg;
00096 l4_uint32_t discard_sector_alignment;
00097 l4_uint32_t max_write_zeroes_sectors;
00098 l4_uint32_t max_write_zeroes_seg;
00099 l4_uint8_t write_zeroes_may_unmap;
00100 l4_uint8_t unused1[3];
00101 } l4virtio_block_config_t;
00102

```

## 17.264 virtio\_input.h

```

00001 /* SPDX-License-Identifier: MIT */
00002 /*
00003  * Copyright (C) 2019, 2022, 2024 Kernkonzept GmbH.
00004  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00005  */
00006 #pragma once
00007
00013
00014 #include <l4/sys/types.h>
00015
00019 enum l4virtio_input_config_select
00020 {
00021     L4VIRTIO_INPUT_CFG_UNSET = 0,
00022     L4VIRTIO_INPUT_CFG_ID_NAME = 1,
00023     L4VIRTIO_INPUT_CFG_ID_SERIAL = 2,
00024     L4VIRTIO_INPUT_CFG_ID_DEVIDS = 3,
00025     L4VIRTIO_INPUT_CFG_PROP_BITS = 0x10,
00026     L4VIRTIO_INPUT_CFG_EV_BITS = 0x11,
00027     L4VIRTIO_INPUT_CFG_ABS_INFO = 0x12
00028 };
00029
00033 typedef struct l4virtio_input_absinfo_t
00034 {
00035     l4_uint32_t min;
00036     l4_uint32_t max;
00037     l4_uint32_t fuzz;
00038     l4_uint32_t flat;
00039     l4_uint32_t res;
00040 } l4virtio_absinfo_t;
00041
00045 typedef struct l4virtio_input_devids_t
00046 {
00047     l4_uint16_t bustype;
00048     l4_uint16_t vendor;
00049     l4_uint16_t product;
00050     l4_uint16_t version;
00051 } l4virtio_input_devids_t;
00052
00056 typedef struct l4virtio_input_config_t
00057 {
00058     l4_uint8_t select;
00059     l4_uint8_t subsel;
00060     l4_uint8_t size;
00061     l4_uint8_t reserved[5];
00062     union
00063     {
00064         char string[128];
00065         l4_uint8_t bitmap[128];
00066         struct l4virtio_input_absinfo_t abs;
00067         struct l4virtio_input_devids_t ids;
00068     } u;
00069 } l4virtio_input_config_t;
00070
00074 typedef struct l4virtio_input_event_t

```

```

00075 {
00076     l4_uint16_t type;
00077     l4_uint16_t code;
00078     l4_uint32_t value;
00079 } l4virtio_input_event_t;
00080

```

## 17.265 virtqueue

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * (c) 2014 Alexander Warg <warg@os.inf.tu-dresden.de>
00005  */
00006
00007 #include <l4/re/util/debug>
00008 #include <l4/sys/types.h>
00009 #include <l4/sys/err.h>
00010 #include <l4/cxx/bitfield>
00011 #include <l4/cxx/exceptions>
00012 #include <stdint>
00013
00014 #pragma once
00015
00016 namespace L4virtio {
00017
00018 #if defined(__ARM_ARCH) && __ARM_ARCH == 7
00019 static inline void wmb() { asm volatile ("dmb ishst" : : : "memory"); }
00020 static inline void rmb() { asm volatile ("dmb ish" : : : "memory"); }
00021 #elif defined(__ARM_ARCH) && __ARM_ARCH >= 8
00022 static inline void wmb() { asm volatile ("dmb ishst" : : : "memory"); }
00023 static inline void rmb() { asm volatile ("dmb ishld" : : : "memory"); }
00024 #elif defined(__mips__)
00025 static inline void wmb() { asm volatile ("sync" : : : "memory"); }
00026 static inline void rmb() { asm volatile ("sync" : : : "memory"); }
00027 #elif defined(__amd64__) || defined(__i386__) || defined(__i686__)
00028 static inline void wmb() { asm volatile ("sfence" : : : "memory"); }
00029 static inline void rmb() { asm volatile ("lfence" : : : "memory"); }
00030 #elif defined(__riscv)
00031 static inline void wmb() { asm volatile ("fence ow, ow" : : : "memory"); }
00032 static inline void rmb() { asm volatile ("fence ir, ir" : : : "memory"); }
00033 #else
00034 #warning Missing proper memory write barrier
00035 static inline void wmb() { asm volatile ("": : : : "memory"); }
00036 static inline void rmb() { asm volatile ("": : : : "memory"); }
00037 #endif
00038
00039
00046 template< typename T >
00047 class Ptr
00048 {
00049 public:
00051     enum Invalid_type { Invalid };
00052
00053     Ptr() = default;
00054
00056     Ptr(Invalid_type) : _p(~0ULL) {}
00057
00059     explicit Ptr(l4_uint64_t vm_addr) : _p(vm_addr) {}
00060
00062     l4_uint64_t get() const { return _p; }
00063
00065     bool is_valid() const { return _p != ~0ULL; }
00066
00067 private:
00068     l4_uint64_t _p;
00069 };
00070
00071
00080 class Virtqueue
00081 {
00082 public:
00086     class Desc
00087     {
00088     public:
00092         struct Flags
00093         {
00094             l4_uint16_t raw;
00095             Flags() = default;
00096
00098             explicit Flags(l4_uint16_t v) : raw(v) {}
00099
00101             CXX_BITFIELD_MEMBER( 0, 0, next, raw);

```

```

00103     CXX_BITFIELD_MEMBER( 1, 1, write, raw);
00105     CXX_BITFIELD_MEMBER( 2, 2, indirect, raw);
00106 };
00107
00108     Ptr<void> addr;
00109     l4_uint32_t len;
00110     Flags flags;
00111     l4_uint16_t next;
00112
00116     void dump(unsigned idx) const
00117     {
00118         L4Re::Util::Dbg().printf("D[%04x]: %08llx (%x) f=%04x n=%04x\n",
00119                                 idx, addr.get(),
00120                                 len, static_cast<unsigned>(flags.raw),
00121                                 static_cast<unsigned>(next));
00122     }
00123 };
00124
00128     class Avail
00129     {
00130     public:
00134         struct Flags
00135         {
00136             l4_uint16_t raw;
00137             Flags() = default;
00138
00140             explicit Flags(l4_uint16_t v) : raw(v) {}
00141
00143             CXX_BITFIELD_MEMBER( 0, 0, no_irq, raw);
00144         };
00145
00146         Flags flags;
00147         l4_uint16_t idx;
00148         l4_uint16_t ring[];
00149     };
00150
00154     struct Used_elem
00155     {
00156         Used_elem() = default;
00157
00165         Used_elem(l4_uint16_t id, l4_uint32_t len) : id(id), len(len) {}
00166         l4_uint32_t id;
00167         l4_uint32_t len;
00168     };
00169
00173     class Used
00174     {
00175     public:
00179         struct Flags
00180         {
00181             l4_uint16_t raw;
00182             Flags() = default;
00183
00185             explicit Flags(l4_uint16_t v) : raw(v) {}
00186
00188             CXX_BITFIELD_MEMBER( 0, 0, no_notify, raw);
00189         };
00190
00191         Flags flags;
00192         l4_uint16_t idx;
00193         Used_elem ring[];
00194     };
00195
00196     protected:
00197     Desc *_desc = nullptr;
00198     Avail *_avail = nullptr;
00199     Used *_used = nullptr;
00200
00202     l4_uint16_t _current_avail = 0;
00203
00208     l4_uint16_t _idx_mask = 0;
00209
00213     Virtqueue() = default;
00214
00215     Virtqueue(Virtqueue const &) = delete;
00216     ~Virtqueue() = default;
00217
00218     public:
00224     void disable()
00225     { _desc = 0; }
00226
00230     enum
00231     {
00232         Desc_align = 4, //< Alignment of the descriptor table.
00233         Avail_align = 1, //< Alignment of the available ring.
00234         Used_align = 2, //< Alignment of the used ring.
00235     };

```

```

00236
00245 static unsigned long total_size(unsigned num)
00246 {
00247     static_assert(Desc_align >= Avail_align,
00248         "virtqueue alignment assumptions broken");
00249     return l4_round_size(desc_size(num) + avail_size(num), Used_align)
00250         + used_size(num);
00251 }
00252
00261 static unsigned long desc_size(unsigned num)
00262 { return num * 16; }
00263
00269 static unsigned long desc_align()
00270 { return Desc_align; }
00271
00279 static unsigned long avail_size(unsigned num)
00280 { return 2 * num + 6; }
00281
00287 static unsigned long avail_align()
00288 { return Avail_align; }
00289
00298 static unsigned long used_size(unsigned num)
00299 { return 8 * num + 6; }
00300
00306 static unsigned long used_align()
00307 { return Used_align; }
00308
00314 unsigned long total_size() const
00315 {
00316     return (reinterpret_cast<char *>(_used) - reinterpret_cast<char *>(_desc))
00317         + used_size(num());
00318 }
00319
00323 unsigned long avail_offset() const
00324 { return reinterpret_cast<char *>(_avail) - reinterpret_cast<char *>(_desc); }
00325
00329 unsigned long used_offset() const
00330 { return reinterpret_cast<char *>(_used) - reinterpret_cast<char *>(_desc); }
00331
00349 void setup(unsigned num, void *desc, void *avail, void *used)
00350 {
00351     if (num > 0x10000)
00352         throw L4::Runtime_error(-L4_EINVAL, "Queue too large.");
00353
00354     _idx_mask = num - 1;
00355     _desc = static_cast<Desc*>(desc);
00356     _avail = static_cast<Avail*>(avail);
00357     _used = static_cast<Used*>(used);
00358
00359     _current_avail = 0;
00360
00361     L4Re::Util::Dbg().printf("VQ[%p]: num=%d d:%p a:%p u:%p\n",
00362         this, num, _desc, _avail, _used);
00363 }
00364
00378 void setup_simple(unsigned num, void *ring)
00379 {
00380     l4_addr_t desc = reinterpret_cast<l4_addr_t>(ring);
00381     l4_addr_t avail = l4_round_size(desc + desc_size(num), Avail_align);
00382     void *used = reinterpret_cast<void *>(
00383         l4_round_size(avail + avail_size(num), Used_align));
00384     setup(num, ring, reinterpret_cast<void *>(avail), used);
00385 }
00386
00392 void dump(Desc const *d) const
00393 { d->dump(d - _desc); }
00394
00400 bool ready() const
00401 { return L4_LIKELY(_desc != 0); }
00402
00404 unsigned num() const
00405 { return _idx_mask + 1; }
00406
00414 bool no_notify_guest() const
00415 {
00416     return _avail->flags.no_irq();
00417 }
00418
00426 bool no_notify_host() const
00427 {
00428     return _used->flags.no_notify();
00429 }
00430
00436 void no_notify_host(bool value)
00437 {
00438     _used->flags.no_notify() = value;
00439 }

```



```

00440
00449  l4_uint16_t get_avail_idx() const { return _avail->idx; }
00450
00456  l4_uint16_t get_tail_avail_idx() const { return _current_avail; }
00457
00458  };
00459
00460  namespace Driver {
00461
00470  class Virtqueue : public L4virtio::Virtqueue
00471  {
00472  private:
00474      l4_uint16_t _next_free;
00475
00476  public:
00477      enum End_of_queue
00478      {
00479          // Indicates the end of the queue.
00480          Eoq = 0xFFFF
00481      };
00482
00483      Virtqueue() : _next_free(Eoq) {}
00484
00494  void initialize_rings(unsigned num)
00495  {
00496      _used->idx = 0;
00497      _avail->idx = 0;
00498
00499      // setup the freelist
00500      for (l4_uint16_t d = 0; d < num - 1; ++d)
00501          _desc[d].next = d + 1;
00502      _desc[num - 1].next = Eoq;
00503      _next_free = 0;
00504  }
00505
00522  void init_queue(unsigned num, void *desc, void *avail, void *used)
00523  {
00524      setup(num, desc, avail, used);
00525      initialize_rings(num);
00526  }
00527
00537  void init_queue(unsigned num, void *base)
00538  {
00539      setup_simple(num, base);
00540      initialize_rings(num);
00541  }
00542
00543
00558  l4_uint16_t alloc_descriptor()
00559  {
00560      l4_uint16_t idx = _next_free;
00561      if (idx == Eoq)
00562          return Eoq;
00563
00564      _next_free = _desc[idx].next;
00565
00566      return idx;
00567  }
00568
00574  void enqueue_descriptor(l4_uint16_t descno)
00575  {
00576      if (descno > _idx_mask)
00577          throw L4::Bounds_error();
00578
00579      _avail->ring[_avail->idx & _idx_mask] = descno; // _avail->idx expected to wrap
00580      wmb();
00581      ++_avail->idx;
00582  }
00583
00590  Desc &desc(l4_uint16_t descno)
00591  {
00592      if (descno > _idx_mask)
00593          throw L4::Bounds_error();
00594
00595      return _desc[descno];
00596  }
00597
00609  l4_uint16_t find_next_used(l4_uint32_t *len = nullptr)
00610  {
00611      if (_current_avail == _used->idx)
00612          return Eoq;
00613
00614      auto elem = _used->ring[_current_avail++ & _idx_mask];
00615
00616      if (len)
00617          *len = elem.len;
00618

```

```

00619     return elem.id;
00620 }
00621
00631 void free_descriptor(l4_uint16_t head, l4_uint16_t tail)
00632 {
00633     if (head > _idx_mask || tail > _idx_mask)
00634         throw L4::Bounds_error();
00635
00636     _desc[tail].next = _next_free;
00637     _next_free = head;
00638 }
00639 };
00640
00641 }
00642 } // namespace L4virtio

```

## 17.266 block\_device\_mgr.h

```

00001 /*
00002  * Copyright (C) 2018-2020, 2022-2024 Kernkonzept GmbH.
00003  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  *             Manuel von Oltersdorff-Kalettko <manuel.kalettko@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <cassert>
00011 #include <cstring>
00012 #include <memory>
00013 #include <string>
00014 #include <vector>
00015
00016 #include <l4/cxx/ref_ptr>
00017 #include <l4/cxx/ref_ptr_list>
00018 #include <l4/cxx/unique_ptr>
00019 #include <l4/re/error_helper>
00020 #include <l4/sys/factory>
00021 #include <l4/sys/cxx/ipc_epiface>
00022
00023 #include <l4/libblock-device/debug.h>
00024 #include <l4/libblock-device/errand.h>
00025 #include <l4/libblock-device/partition.h>
00026 #include <l4/libblock-device/part_device.h>
00027 #include <l4/libblock-device/virtio_client.h>
00028 #include <l4/libblock-device/scheduler.h>
00029
00030 namespace Block_device {
00031
00032     template <typename DEV>
00033     struct Simple_factory
00034     {
00035         using Device_type = DEV;
00036         using Client_type = Virtio_client<Device_type>;
00037
00038         static cxx::unique_ptr<Client_type>
00039         create_client(cxx::Ref_ptr<Device_type> const &dev,
00040                     unsigned numds, bool readonly)
00041         { return cxx::make_unique<Client_type>(dev, numds, readonly); }
00042     };
00043
00044     template <typename BASE_DEV>
00045     struct Partitionable_factory
00046     {
00047         using Device_type = BASE_DEV;
00048         using Client_type = Virtio_client<Device_type>;
00049
00050         static cxx::unique_ptr<Client_type>
00051         create_client(cxx::Ref_ptr<Device_type> const &dev,
00052                     unsigned numds, bool readonly)
00053         {
00054             return cxx::make_unique<Client_type>(dev, numds, readonly);
00055         }
00056
00057         static cxx::Ref_ptr<Device_type>
00058         create_partition(cxx::Ref_ptr<Device_type> const &dev, unsigned partition_id,
00059                        Partition_info const &pi)
00060         {
00061             return cxx::Ref_ptr<Device_type>{
00062                 new Partitioned_device<Device_type>(dev, partition_id, pi);
00063             };
00064         };
00065     };

```

```

00066
00077 template <typename DEV, typename FACTORY = Simple_factory<DEV>,
00078           typename SCHEDULER = Rr_scheduler<typename FACTORY::Device_type>
00079 class Device_mgr
00080 {
00081     using Device_factory_type = FACTORY;
00082     using Client_type = typename Device_factory_type::Client_type;
00083     using Device_type = typename Device_factory_type::Device_type;
00084     using Scheduler_type = SCHEDULER;
00085
00086     using Ds_vector = std::vector<L4::Cap<L4Re::Dataspace>;
00087
00088     using Pairing_callback = std::function<void(Device_type *)>;
00089
00093     struct Pending_client
00094     {
00096         std::string device_id;
00098         L4::Cap<L4::Rcv_endpoint> gate;
00100         int num_ds;
00102         bool readonly;
00103
00104         bool enable_trusted_ds_validation;
00105
00106         std::shared_ptr<Ds_vector const> trusted_dataspaces;
00107
00109         Pairing_callback pairing_cb;
00110
00111         Pending_client() = default;
00112
00113         Pending_client(L4::Cap<L4::Rcv_endpoint> g, std::string const &dev, int ds,
00114                       bool ro, bool enable_trusted_ds_validation,
00115                       std::shared_ptr<Ds_vector const> trusted_dataspaces,
00116                       Pairing_callback cb)
00117             : device_id(dev), gate(g), num_ds(ds), readonly(ro),
00118               enable_trusted_ds_validation(enable_trusted_ds_validation),
00119               trusted_dataspaces(trusted_dataspaces), pairing_cb(cb)
00120         {}
00121     };
00122
00123     class Connection : public cxx::Ref_obj_list_item<Connection>
00124     {
00125     public:
00126         explicit Connection(Device_mgr *mgr, cxx::Ref_ptr<Device_type> &dev)
00127             : _shutdown_state(Shutdown_type::Running),
00128               _device(cxx::move(dev)),
00129               _mgr(mgr)
00130         {}
00131
00132         L4::Cap<void> cap() const
00133         { return _interface ? _interface->obj_cap() : L4::Cap<void>(); }
00134
00135         void start_disk_scan(Errand::Callback const &callback)
00136         {
00137             _device->start_device_scan(
00138                 [=]()
00139                 {
00140                     scan_disk_partitions(callback, 0);
00141                 });
00142         }
00143
00144         void unregister_interfaces(L4::Registry_iface *registry) const
00145         {
00146             if (_interface)
00147                 registry->unregister_obj(_interface.get());
00148
00149             for (auto *sub : _subs)
00150                 sub->unregister_interfaces(registry);
00151         }
00152
00153         int create_interface_for(Pending_client *c, L4::Registry_iface *registry)
00154         {
00155             if (_shutdown_state != Shutdown_type::Running)
00156                 return -L4_EIO;
00157
00158             if (_interface)
00159                 return contains_device(c->device_id) ? -L4_EBUSY : -L4_ENODEV;
00160
00161             // check for match in partitions
00162
00163             bool busy = false;
00164             for (auto *sub : _subs)
00165             {
00166                 if (sub->_interface)
00167                     busy = true;
00168
00169                 int ret = sub->create_interface_for(c, registry);
00170

```

```

00171         if (ret != -L4_ENODEV) // includes L4_EOK
00172             return ret;
00173     }
00174
00175     if (!match_hid(c->device_id))
00176         return -L4_ENODEV;
00177
00178     if (busy)
00179         return -L4_EBUSY;
00180
00181     auto clt = Device_factory_type::create_client(_device, c->num_ds,
00182                                                 c->readonly);
00183
00184     clt->add_trusted_dataspaces(c->trusted_dataspaces);
00185     if (c->enable_trusted_ds_validation)
00186         clt->enable_trusted_ds_validation();
00187
00188     if (c->gate.is_valid())
00189     {
00190         if (!clt->register_obj(registry, c->gate).is_valid())
00191             return -L4_ENOMEM;
00192     }
00193     else
00194     {
00195         c->gate = L4::cap_reinterpret_cast<L4::Rcv_endpoint>(
00196             clt->register_obj(registry));
00197         if (!c->gate.is_valid())
00198             return -L4_ENOMEM;
00199     }
00200
00201     _mgr->_scheduler->add_client(clt.get());
00202     _interface.reset(clt.release());
00203
00204     // Let it be known that the client and the device paired
00205     if (c->pairing_cb)
00206         c->pairing_cb(_device.get());
00207     return L4_EOK;
00208 }
00209
00210 void check_clients(L4::Registry_iface *registry)
00211 {
00212     if (_interface)
00213     {
00214         if (_interface->obj_cap() && !_interface->obj_cap().validate().label())
00215             remove_client(registry);
00216
00217         return;
00218     }
00219
00220     // Sub-devices only need to be checked when the parent device was free.
00221     for (auto *sub : _subs)
00222         sub->check_clients(registry);
00223 }
00224
00226 void shutdown_event(Shutdown_type type)
00227 {
00228     // Set new shutdown state
00229     _shutdown_state = type;
00230     for (auto const &sub: _subs)
00231         sub->shutdown_event(type);
00232     if (_interface)
00233         _interface->shutdown_event(type);
00234 }
00235
00236 private:
00248 template <typename T = Device_factory_type>
00249 auto scan_disk_partitions(Errand::Callback const &callback, int)
00250     -> decltype((T::create_partition)(cxx::Ref_ptr<Device_type>(), 0, Partition_info(), void()))
00251 {
00252     auto reader = cxx::make_ref_obj<Partition_reader<Device_type>>(_device.get());
00253     // The reference to reader will be captured in the lambda passed to
00254     // reader's own read() method. At the same time, reader will store
00255     // the reference to the lambda.
00256     reader->read(
00257         [=]()
00258         {
00259             l4_size_t sz = reader->table_size();
00260
00261             for (l4_size_t i = 1; i <= sz; ++i)
00262             {
00263                 Partition_info info;
00264                 if (reader->get_partition(i, &info) < 0)
00265                     continue;
00266
00267                 auto conn = cxx::make_ref_obj<Connection>(
00268                     _mgr,
00269                     Device_factory_type::create_partition(_device, i, info));

```

```

00270         _subs.push_front(std::move(conn));
00271     }
00272
00273     callback();
00274
00275     // Prolong the life-span of reader until we are sure the reader is
00276     // not currently invoked (i.e. capture the last reference to it in
00277     // an independent timeout callback).
00278     Errand::schedule([reader]() {}, 0);
00279 });
00280
00281
00282 template <typename T = Device_factory_type>
00283 void scan_disk_partitions(Errand::Callback const &callback, long)
00284 { callback(); }
00285
00286 void remove_client(L4::Registry_iface *registry)
00287 {
00288     assert(_interface);
00289
00290     // This operation is idempotent.
00291     _interface->shutdown_event(Shutdown_type::Client_gone);
00292
00293     if (_interface->busy())
00294     {
00295         Dbg::trace().printf("Deferring dead client removal.\n");
00296
00297         // Cannot remove the client while it still has active I/O requests.
00298         // This means that the device did not abort its inflight requests in
00299         // its reset() callback. It is still desirable though to wait for
00300         // those requests to finish and defer the dead client removal until
00301         // later.
00302         Errand::schedule([this, registry]() { remove_client(registry); },
00303             10000);
00304         return;
00305     }
00306
00307     _interface->unregister_obj(registry);
00308     _mgr->_scheduler->remove_client(_interface.get());
00309     _interface.reset();
00310 }
00311
00312 bool contains_device(std::string const &name) const
00313 {
00314     if (match_hid(name))
00315         return true;
00316
00317     for (auto *sub : _subs)
00318         if (sub->contains_device(name))
00319             return true;
00320
00321     return false;
00322 }
00323
00324 bool match_hid(std::string const &name) const
00325 { return _device->match_hid(cxx::String(name.c_str(), name.length())); }
00326
00327 Shutdown_type _shutdown_state;
00328 cxx::Ref_ptr<Device_type> _device;
00329 cxx::unique_ptr<Client_type> _interface;
00330 cxx::Ref_ptr_list<Connection> _subs;
00331
00332 Device_mgr *_mgr;
00333 };
00334
00335 public:
00336 Device_mgr(L4::Registry_iface *registry)
00337 : _registry(registry)
00338 {
00339     _scheduler = cxx::make_unique<Scheduler_type>(registry);
00340 }
00341
00342 virtual ~Device_mgr()
00343 {
00344     for (auto *c : _connpts)
00345         c->unregister_interfaces(_registry);
00346 }
00347
00348 static int parse_device_name(std::string const &param, std::string &device)
00349 {
00350     std::string const partlabel("partlabel:");
00351     std::string const partuuid("partuuid:");
00352
00353     if (param.size() > partlabel.size()
00354         && param.compare(0, partlabel.size(), partlabel) == 0)
00355     {

```

```

00391         device = param.substr(partlabel.size());
00392         return L4_EOK;
00393     }
00394     else if (param.size() > partuuid.size()
00395             && param.compare(0, partuuid.size(), partuuid) == 0)
00396     {
00397         auto device_partuuid = param.substr(partuuid.size());
00398         if (!is_uuid(device_partuuid.c_str()))
00399         {
00400             Dbg::trace().printf("The 'partuuid:' parameter expects a UUID.\n");
00401             return -L4_EINVAL;
00402         }
00403
00404         device = device_partuuid;
00405         std::transform(device.begin(), device.end(), device.begin(),
00406             [](unsigned char c){ return std::toupper(c); });
00407         return L4_EOK;
00408     }
00409     else
00410     {
00411         device = param;
00412         if (is_uuid(param.c_str()))
00413             std::transform(device.begin(), device.end(), device.begin(),
00414                 [](unsigned char c) { return std::toupper(c); });
00415         return L4_EOK;
00416     }
00417 }
00418
00419 int add_static_client(L4::Cap<L4::Rcv_endpoint> client, const char *device,
00420                     int partno, int num_ds, bool readonly = false,
00421                     Pairing_callback cb = nullptr,
00422                     bool enable_trusted_ds_validation = false,
00423                     std::shared_ptr<Ds_vector const> trusted_dataspaces
00424                     = nullptr)
00425 {
00426     char _buf[30];
00427     const char *buf;
00428
00429     if (partno == 0)
00430     {
00431         Err().printf("Invalid partition number 0.\n");
00432         return -L4_ENODEV;
00433     }
00434
00435     if (partno != -1)
00436     {
00437         /* Could we avoid to make a string here and parsing this again
00438          * deeper in the stack? */
00439         snprintf(_buf, sizeof(_buf), "%s:%d", device, partno);
00440         buf = _buf;
00441     }
00442     else
00443         buf = device;
00444
00445     _pending_clients.emplace_back(client, buf, num_ds, readonly,
00446                                   enable_trusted_ds_validation,
00447                                   trusted_dataspaces, cb);
00448
00449     return L4_EOK;
00450 }
00451
00452 int create_dynamic_client(std::string const &device, int partno, int num_ds,
00453                          L4::Cap<void> *cap, bool readonly = false,
00454                          Pairing_callback cb = nullptr,
00455                          bool enable_trusted_ds_validation = false,
00456                          std::shared_ptr<Ds_vector const> trusted_dataspaces
00457                          = nullptr)
00458 {
00459     Pending_client clt;
00460
00461     // Maximum number of dataspaces that can be registered.
00462     clt.num_ds = num_ds;
00463
00464     clt.readonly = readonly;
00465
00466     clt.device_id = device;
00467
00468     clt.pairing_cb = cb;
00469
00470     clt.trusted_dataspaces = trusted_dataspaces;
00471
00472     clt.enable_trusted_ds_validation = enable_trusted_ds_validation;
00473
00474     if (partno > 0)
00475     {
00476         clt.device_id += ':';
00477         clt.device_id += std::to_string(partno);

```

```

00478     }
00479
00480     for (auto *c : _connpts)
00481     {
00482         int ret = c->create_interface_for(&clt, _registry);
00483
00484         if (ret == -L4_ENODEV)
00485             continue;
00486
00487         if (ret < 0)
00488             return ret;
00489
00490         // found the requested device
00491         *cap = clt.gate;
00492         return L4_EOK;
00493     }
00494
00495     return -L4_ENODEV;
00496 }
00497
00501 void check_clients()
00502 {
00503     for (auto *c : _connpts)
00504         c->check_clients(_registry);
00505 }
00506
00507 void add_disk(cxx::Ref_ptr<Device_type> &&device, Errand::Callback const &callback)
00508 {
00509     auto conn = cxx::make_ref_obj<Connection>(this, std::move(device));
00510
00511     conn->start_disk_scan(
00512         [=] ()
00513         {
00514             _connpts.push_front(conn);
00515             connect_static_clients(conn.get());
00516             callback();
00517         });
00518 }
00519
00521 void shutdown_event(Shutdown_type type)
00522 {
00523     l4_assert(type != Client_gone);
00524     l4_assert(type != Client_shutdown);
00525
00526     for (auto const &con : _connpts)
00527         con->shutdown_event(type);
00528 }
00529
00530 private:
00531 void connect_static_clients(Connection *con)
00532 {
00533     for (auto &c : _pending_clients)
00534     {
00535         Dbg::trace().printf("Checking existing client %s\n", c.device_id.c_str());
00536         if (!c.gate.is_valid())
00537             continue;
00538
00539         int ret = con->create_interface_for(&c, _registry);
00540
00541         if (ret == L4_EOK)
00542         {
00543             c.gate = L4::Cap<L4::Rcv_endpoint>();
00544             // There might be other clients waiting for other partitions.
00545             // Continue search.
00546             continue;
00547         }
00548
00549         if (ret != -L4_ENODEV)
00550             break;
00551     }
00552 }
00553
00554 static constexpr bool is_uuid(char const *s)
00555 {
00556     for (unsigned i = 0; i < 36; ++i)
00557         if (i == 8 || i == 13 || i == 18 || i == 23)
00558         {
00559             if (s[i] != '-')
00560                 return false;
00561         }
00562     else
00563     {
00564         if (!isxdigit(s[i]))
00565             return false;
00566     }
00567     return s[36] == '\0';
00568 }

```

```

00569
00571 L4::Registry_iface *_registry;
00573 cxx::Ref_ptr_list<Connection> _connpts;
00575 std::vector<Pending_client> _pending_clients;
00577 cxx::unique_ptr<Scheduler_type> _scheduler;
00578 };
00579
00580 } // name space

```

## 17.267 device.h

```

00001 /*
00002  * Copyright (C) 2018-2020, 2024 Kernkonzept GmbH.
00003  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/cxx/ref_ptr>
00010 #include <l4/cxx/string>
00011 #include <l4/re/dataspace>
00012 #include <l4/re/dma_space>
00013
00014 #include <l4/libblock-device/errand.h>
00015 #include <l4/libblock-device/types.h>
00016
00017 namespace Block_device {
00018
00032 struct Notification_domain
00033 {
00034 };
00035
00036 struct Device : public cxx::Ref_obj
00037 {
00038     virtual ~Device() = 0;
00039
00041     virtual Notification_domain const *notification_domain() const = 0;
00042
00044     virtual bool is_read_only() const = 0;
00046     virtual bool match_hid(cxx::String const &hid) const = 0;
00048     virtual l4_uint64_t capacity() const = 0;
00050     virtual l4_size_t sector_size() const = 0;
00052     virtual l4_size_t max_size() const = 0;
00054     virtual unsigned max_segments() const = 0;
00055
00057     virtual void reset() = 0;
00058
00060     virtual int dma_map(Block_device::Mem_region *region, l4_addr_t offset,
00061                        l4_size_t num_sectors, L4Re::Dma_space::Direction dir,
00062                        L4Re::Dma_space::Dma_addr *phys) = 0;
00063
00065     virtual int dma_unmap(L4Re::Dma_space::Dma_addr phys, l4_size_t num_sectors,
00066                          L4Re::Dma_space::Direction dir) = 0;
00067
00082     virtual int inout_data(l4_uint64_t sector,
00083                            Block_device::Inout_block const &blocks,
00084                            Block_device::Inout_callback const &cb,
00085                            L4Re::Dma_space::Direction dir) = 0;
00086
00097     virtual int flush(Block_device::Inout_callback const &cb) = 0;
00098
00100     virtual void start_device_scan(Block_device::Errand::Callback const &callback) = 0;
00101 };
00102
00103 inline Device::~~Device() = default;
00104
00108 template <typename DEV>
00109 struct Device_with_notification_domain : DEV
00110 {
00111     Notification_domain dom;
00112     Notification_domain const *notification_domain() const override
00113     { return &dom; }
00114 };
00115
00119 struct Device_discard_feature
00120 {
00121     struct Discard_info
00122     {
00123         unsigned max_discard_sectors = 0;
00124         unsigned max_discard_seg = 0;
00125         unsigned discard_sector_alignment = 1;
00126         unsigned max_write_zeroes_sectors = 0;

```



```

00127     unsigned max_write_zeroes_seg = 0;
00128     bool write_zeroes_may_unmap = false;
00129 };
00130
00131     virtual Discard_info discard_info() const = 0;
00132
00133     virtual int discard(l4_uint64_t offset,
00134                        Block_device::Inout_block const &blocks,
00135                        Block_device::Inout_callback const &cb, bool discard) = 0;
00136
00137 protected:
00138     ~Device_discard_feature() = default;
00139 };
00140
00141
00142 } // name space

```

## 17.268 errand.h

```

00001 /*
00002  * Copyright (C) 2014, 2020, 2024 Kernkonzept GmbH.
00003  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/re/env.h>
00010 #include <l4/re/util/object_registry>
00011 #include <l4/re/util/br_manager>
00012 #include <l4/cxx/ipc_timeout_queue>
00013 #include <l4/cxx/ref_ptr>
00014 #include <l4/cxx/exceptions>
00015 #include <l4/libblock-device/debug.h>
00016
00017 #include <functional>
00018
00019 namespace Block_device { namespace Errand {
00020
00021     extern L4::Ipc_svr::Server_iface *_sif;
00022
00023     typedef std::function<void()> Callback;
00024
00025     class Poll_errand
00026     : public L4::Ipc_svr::Timeout_queue::Timeout,
00027       public cxx::Ref_obj
00028     {
00029     public:
00030         void expired() final
00031         {
00032             // Recapture the reference pointer from the timeout queue.
00033             cxx::Ref_ptr<Poll_errand> p(this, false);
00034
00035             try
00036             {
00037                 if (_poll())
00038                     _callback(true);
00039                 else
00040                     if (--_retries <= 0)
00041                         _callback(false);
00042                     else
00043                         reschedule();
00044             }
00045             catch (L4::Runtime_error const &e)
00046             {
00047                 Err().printf("Polling task failed: %s\n", e.str());
00048             }
00049         }
00050
00051         void reschedule()
00052         {
00053             // create a place holder reference pointer for the timeout queue
00054             cxx::Ref_ptr<Poll_errand> p(this);
00055
00056             _sif->add_timeout(p.release(), l4_kip_clock(l4re_kip()) + _interval);
00057         }
00058
00059         // Class can only be instantiated as a reference counting object.
00060         template< typename T, typename... Args >
00061         friend
00062         cxx::Ref_ptr<T> cxx::make_ref_obj(Args &&... args);
00063
00064     private:

```

```

00073 Poll_errand(int retries, int interval,
00074             std::function<bool()> const &poll_func,
00075             std::function<void(bool)> const &callback)
00076 : _retries(retries),
00077   _interval(interval),
00078   _poll(poll_func),
00079   _callback(callback)
00080 {}
00081
00082 int _retries;
00083 int _interval;
00084 std::function<bool()> _poll;
00085 std::function<void(bool)> _callback;
00086 };
00087
00098 class Errand
00099 : public L4::Ipc_svr::Timeout_queue::Timeout,
00100   public cxx::Ref_obj
00101 {
00102 public:
00103   void expired() final
00104   {
00105     // Recapture the reference pointer from the timeout queue.
00106     cxx::Ref_ptr<Errand> p(this, false);
00107
00108     if (_callback)
00109     {
00110       try
00111       {
00112         _callback();
00113       }
00114       catch (L4::Runtime_error const &e)
00115       {
00116         Err().printf("Asynchronous task failed: %s\n", e.str());
00117       }
00118     }
00119   }
00120
00121   void reschedule(unsigned interval = 0)
00122   {
00123     // create a placeholder reference pointer for the timeout queue
00124     cxx::Ref_ptr<Errand> p(this);
00125
00126     _sif->add_timeout(p.release(), l4_kip_clock(l4re_kip()) + interval);
00127   }
00128
00129   // Class can only be instantiated as a reference counting object.
00130   template< typename T, typename... Args >
00131   friend
00132   cxx::Ref_ptr<T> cxx::make_ref_obj(Args &&... args);
00133
00134 private:
00135   Errand(Callback const &callback) : _callback(callback) {}
00136
00137   Callback _callback;
00138 };
00139
00140 struct Loop_hooks
00141 : L4::Ipc_svr::Timeout_queue_hooks<Loop_hooks, L4Re::Util::Br_manager>,
00142   L4::Ipc_svr::Ignore_errors
00143 {
00144   l4_kernel_clock_t now() { return l4_kip_clock(l4re_kip()); }
00145 };
00146
00147 using Errand_server = L4Re::Util::Registry_server<Loop_hooks>;
00148
00154 inline void set_server_iface(L4::Ipc_svr::Server_iface *sif) { _sif = sif; }
00155
00166 inline void schedule(Callback const &callback, int interval)
00167 {
00168   cxx::make_ref_obj<Errand>(callback)->reschedule(interval);
00169 }
00170
00191 inline void poll(int retries, int interval,
00192                 std::function<bool()> const &poll_func,
00193                 std::function<void(bool)> const &callback)
00194 {
00195   if (poll_func())
00196     callback(true);
00197   else
00198     cxx::make_ref_obj<Poll_errand>(retries, interval, poll_func,
00199                                   callback)->reschedule();
00200 }
00201
00202
00203 } } // name space

```

## 17.269 gpt.h

```

00001 /*
00002  * Copyright (C) 2018, 2024 Kernkonzept GmbH.
00003  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/sys/types.h>
00010
00011 namespace Block_device {
00012 namespace Gpt {
00013
00014 struct Header
00015 {
00016     char        signature[8];
00017     l4_uint32_t  version;
00018     l4_uint32_t  header_size;
00019     l4_uint32_t  crc;
00020     l4_uint32_t  _reserved;
00021     l4_uint64_t  current_lba;
00022     l4_uint64_t  backup_lba;
00023     l4_uint64_t  first_lba;
00024     l4_uint64_t  last_lba;
00025     char        disk_guid[16];
00026     l4_uint64_t  partition_array_lba;
00027     l4_uint32_t  partition_array_size;
00028     l4_uint32_t  entry_size;
00029     l4_uint32_t  crc_array;
00030 } __attribute__((packed));
00031
00032 struct Entry
00033 {
00034     unsigned char type_guid[16];
00035     unsigned char partition_guid[16];
00036     l4_uint64_t   first;
00037     l4_uint64_t   last;
00038     l4_uint64_t   flags;
00039     l4_uint16_t   name[36];
00040 };
00041
00042 } // namespace
00043
00044 namespace Pc_partition_table {
00045
00046 struct Part_table {
00047     l4_uint8_t    bootable;
00048     l4_uint8_t    first_sector_chs[3];
00049     l4_uint8_t    type;
00050     l4_uint8_t    last_sector_chs[3];
00051     l4_uint32_t    start_sector_lba;
00052     l4_uint32_t    num_sector_lba;
00053 } __attribute__((packed));
00054
00055 } // namespace
00056 } // namespace

```

## 17.270 inout\_memory.h

```

00001 /*
00002  * Copyright (C) 2014, 2019-2020, 2024 Kernkonzept GmbH.
00003  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/re/error_helper>
00010 #include <l4/re/env>
00011 #include <l4/re/util/unique_cap>
00012 #include <l4/re/rm>
00013 #include <l4/re/dma_space>
00014 #include <l4/cxx/ref_ptr>
00015
00016 #include <l4/libblock-device/types.h>
00017
00018 namespace Block_device {
00019
00024 template <typename DEV>
00025 class Inout_memory : public cxx::Ref_obj

```

```

00026 {
00027 public:
00028     using Device_type = DEV;
00029
00030     Inout_memory() : _paddr(0) {}
00031     Inout_memory(l4_uint32_t num_sectors, Device_type *dev,
00032                 L4Re::Dma_space::Direction dir)
00033     : _device(dev), _paddr(0), _dir(dir), _num_sectors(num_sectors)
00034     {
00035         auto lcap = L4Re::chkcap(L4Re::Util::make_unique_cap<L4Re::Dataspace>()),
00036                 "Allocate dataspace capability for IO memory.");
00037
00038         auto *e = L4Re::Env::env();
00039         long sz = num_sectors * _device->sector_size();
00040         L4Re::chksys(e->mem_alloc()->alloc(sz, lcap.get(),
00041                                           L4Re::Mem_alloc::Continuous
00042                                           | L4Re::Mem_alloc::Pinned),
00043                     "Allocate pinned memory.");
00044
00045         L4Re::chksys(e->rm()->attach(&_region, sz,
00046                                     L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00047                                     L4::Ipc::make_cap_rw(lcap.get()), 0,
00048                                     L4_PAGESHIFT),
00049                     "Attach IO memory.");
00050
00051         _mem_region =
00052             cxx::make_unique<Block_device::Mem_region>(0, sz, 0, cxx::move(lcap));
00053         L4Re::chksys(_device->dma_map(_mem_region.get(), 0, _num_sectors, dir,
00054                                     &_paddr),
00055                     "Lock memory region for DMA.");
00056     }
00057
00058     Inout_memory(Inout_memory const &) = delete;
00059     Inout_memory(Inout_memory &&) = delete;
00060
00061     Inout_memory &operator=(Inout_memory &&rhs)
00062     {
00063         if (this != &rhs)
00064         {
00065             _device = rhs._device;
00066             _mem_region = cxx::move(rhs._mem_region);
00067             _region = cxx::move(rhs._region);
00068             _paddr = rhs._paddr;
00069             _dir = rhs._dir;
00070             _num_sectors = rhs._num_sectors;
00071             rhs._paddr = 0;
00072         }
00073
00074         return *this;
00075     }
00076
00077     ~Inout_memory()
00078     {
00079         if (_paddr)
00080             unmap();
00081     }
00082
00083     void unmap()
00084     {
00085         L4Re::chksys(_device->dma_unmap(_paddr, _num_sectors, _dir));
00086         _paddr = 0;
00087     }
00088
00089     Inout_block inout_block() const
00090     {
00091         Inout_block blk;
00092
00093         blk.dma_addr = _paddr;
00094         blk.virt_addr = _region.get();
00095         blk.num_sectors = _num_sectors;
00096         blk.next.reset();
00097
00098         return blk;
00099     }
00100
00101     template <class T>
00102     T *get(unsigned offset) const
00103     { return reinterpret_cast<T *>(_region.get() + offset); }
00104
00105 private:
00106     Device_type *_device;
00107     cxx::unique_ptr<Block_device::Mem_region> _mem_region;
00108     L4Re::Rm::Unique_region<char *> _region;
00109     L4Re::Dma_space::Dma_addr _paddr;
00110     L4Re::Dma_space::Direction _dir;
00111     l4_uint32_t _num_sectors;

```

```

00113 };
00114
00115 } // name space

```

## 17.271 part\_device.h

```

00001 /*
00002  * Copyright (C) 2018-2022, 2024 Kernkonzept GmbH.
00003  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/cxx/ref_ptr>
00010
00011 #include <l4/libblock-device/device.h>
00012 #include <l4/libblock-device/partition.h>
00013
00014 #include <string>
00015 #include <locale>
00016 #include <codecvt>
00017
00018 namespace Block_device {
00019
00020 namespace Impl {
00021
00022     template <typename PART_DEV, typename BASE_DEV,
00023               bool = std::is_base_of<Device_discard_feature, BASE_DEV>::value>
00028     class Partitioned_device_discard_mixin : public BASE_DEV {};
00029
00030     template <typename PART_DEV, typename BASE_DEV>
00037     class Partitioned_device_discard_mixin<PART_DEV, BASE_DEV, true>
00038     : public BASE_DEV
00039     {
00040     public:
00041         using Base = BASE_DEV;
00042         using Part_device = PART_DEV;
00043
00044         typename Base::Discard_info discard_info() const override
00045         {
00046             return dev()->parent()->discard_info();
00047         }
00048
00049         int discard(l4_uint64_t offset, Inout_block const &blocks,
00050                   Inout_callback const &cb, bool discard) override
00051         {
00052             auto sz = dev()->partition_size();
00053
00054             if (offset > sz)
00055                 return -L4_EINVAL;
00056
00057             Inout_block const *cur = &blocks;
00058             while (cur)
00059             {
00060                 if (cur->sector >= sz - offset)
00061                     return -L4_EINVAL;
00062                 if (cur->num_sectors > sz)
00063                     return -L4_EINVAL;
00064                 if (offset + cur->sector > sz - cur->num_sectors)
00065                     return -L4_EINVAL;
00066
00067                 cur = cur->next.get();
00068             }
00069
00070             auto start = offset + dev()->partition_start();
00071             Dbg::trace("partition")
00072             .printf("Starting sector on disk: 0x%llx\n", start);
00073             return dev()->parent()->discard(start, blocks, cb, discard);
00074         }
00075
00076     private:
00077         Part_device const *dev() const
00078         { return static_cast<Part_device const *>(this); }
00079     };
00080
00081 }
00082
00083 template <typename BASE_DEV = Device>
00091 class Partitioned_device
00092 : public Impl::Partitioned_device_discard_mixin<Partitioned_device<BASE_DEV>, BASE_DEV>
00093 {
00094 public:

```

```

00095     using Device_type = BASE_DEV;
00096
00097     Partitioned_device(cxx::Ref_ptr<Device_type> const &dev,
00098                       unsigned partition_id, Partition_info const &pi)
00099     : _name(pi.name),
00100       _parent(dev),
00101       _start(pi.first),
00102       _size(pi.last - pi.first + 1)
00103     {
00104         if (pi.last < pi.first)
00105             L4Re::chksys(-L4_EINVAL,
00106                         "Last sector of partition before first sector.");
00107
00108         if (partition_id > 999)
00109             L4Re::chksys(-L4_EINVAL,
00110                         "Partition ID must be smaller than 1000.");
00111
00112         snprintf(_partition_id, sizeof(_partition_id), "%d", partition_id);
00113
00114         static_assert(sizeof(_guid) == sizeof(pi.guid), "String size mismatch");
00115         memcpy(_guid, pi.guid, sizeof(_guid));
00116     }
00117
00118     Notification_domain const *notification_domain() const override
00119     { return _parent->notification_domain(); }
00120
00121     bool is_read_only() const override
00122     { return _parent->is_read_only(); }
00123
00124     bool match_hid(cxx::String const &hid) const override
00125     {
00126         if (hid == cxx::String(_guid, 36))
00127             return true;
00128
00129         _Pragma("GCC diagnostic push");
00130         _Pragma("GCC diagnostic ignored \\"-Wdeprecated-declarations\\"");
00131         std::u16string whid =
00132             std::wstring_convert<std::codecvt_utf8_utf16<char16_t>, char16_t>{}
00133                 .from_bytes(std::string(hid.start(), hid.len()));
00134         _Pragma("GCC diagnostic pop");
00135         if (whid == _name)
00136             return true;
00137
00138         // check for identifier of form: <device_name>:<partition id>
00139         char const *delim = ":";
00140         char const *pos = hid.rfind(delim);
00141
00142         if (pos == hid.end() || !_parent->match_hid(cxx::String(hid.start(), pos)))
00143             return false;
00144
00145         return cxx::String(pos + 1, hid.end()) == cxx::String(_partition_id);
00146     }
00147
00148     l4_uint64_t capacity() const override
00149     { return _size * _parent->sector_size(); }
00150
00151     l4_size_t sector_size() const override
00152     { return _parent->sector_size(); }
00153
00154     l4_size_t max_size() const override
00155     { return _parent->max_size(); }
00156
00157     unsigned max_segments() const override
00158     { return _parent->max_segments(); }
00159
00160     void reset() override
00161     {}
00162
00163     int dma_map(Block_device::Mem_region *region, l4_addr_t offset,
00164                l4_size_t num_sectors, L4Re::Dma_space::Direction dir,
00165                L4Re::Dma_space::Dma_addr *phys) override
00166     { return _parent->dma_map(region, offset, num_sectors, dir, phys); }
00167
00168     int dma_unmap(L4Re::Dma_space::Dma_addr phys, l4_size_t num_sectors,
00169                  L4Re::Dma_space::Direction dir) override
00170     { return _parent->dma_unmap(phys, num_sectors, dir); }
00171
00172     int inout_data(l4_uint64_t sector, Inout_block const &blocks,
00173                   Inout_callback const &cb,
00174                   L4Re::Dma_space::Direction dir) override
00175     {
00176         if (sector >= _size)
00177             return -L4_EINVAL;
00178
00179         l4_uint64_t total = 0;
00180         Inout_block const *cur = &blocks;
00181         while (cur)

```

```

00182     {
00183         total += cur->num_sectors;
00184         cur = cur->next.get();
00185     }
00186
00187     if (total > _size - sector)
00188         return -L4_EINVAL;
00189
00190     Dbg::trace("partition").printf("Sector on disk: 0x%llx\n", sector + _start);
00191     return _parent->inout_data(sector + _start, blocks, cb, dir);
00192 }
00193
00194 int flush(Inout_callback const &cb) override
00195 {
00196     return _parent->flush(cb);
00197 }
00198
00199 void start_device_scan(Block_device::Errand::Callback const &callback) override
00200 { callback(); }
00201
00202 l4_uint64_t partition_size() const
00203 { return _size; }
00204
00205 l4_uint64_t partition_start() const
00206 { return _start; }
00207
00208 Device_type *parent() const
00209 { return _parent.get(); }
00210
00211
00212 private:
00213     char _guid[37];
00214     std::ul6string _name;
00215     char _partition_id[4];
00216     cxx::Ref_ptr<Device_type> _parent;
00217     l4_uint64_t _start;
00218     l4_uint64_t _size;
00219 };
00220
00221 } // name space

```

## 17.272 partition.h

```

00001 /*
00002  * Copyright (C) 2018, 2020-2024 Kernkonzept GmbH.
00003  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <cstring>
00010 #include <string>
00011 #include <cassert>
00012
00013 #include <l4/cxx/ref_ptr>
00014
00015 #include <l4/l4virtio/virtio_block.h>
00016
00017 #include <l4/libblock-device/debug.h>
00018 #include <l4/libblock-device/errand.h>
00019 #include <l4/libblock-device/inout_memory.h>
00020 #include <l4/libblock-device/gpt.h>
00021
00022 #include <l4/sys/cache.h>
00023
00024 namespace Block_device {
00025
00026 struct Partition_info
00027 {
00028     char guid[37];
00029     std::ul6string name;
00030     l4_uint64_t first;
00031     l4_uint64_t last;
00032     l4_uint64_t flags;
00033 };
00034
00035 template <typename DEV>
00036 class Partition_reader : public cxx::Ref_obj
00037 {
00038     enum
00039     {

```

```

00047     Max_partitions = 1024
00048 };
00049
00050 public:
00051     using Device_type = DEV;
00052
00053     Partition_reader(Device_type *dev)
00054     : _num_partitions(0),
00055       _dev(dev),
00056       _header(2, dev, L4Re::Dma_space::Direction::From_device)
00057     {}
00058
00059     void read(Errand::Callback const &callback)
00060     {
00061         _num_partitions = 0;
00062         _callback = callback;
00063
00064         // preparation: read the first two sectors
00065         _db = _header.inout_block();
00066         read_sectors(0, &Partition_reader::get_gpt);
00067     }
00068
00069     l4_size_t table_size() const
00070     { return _num_partitions; }
00071
00072     int get_partition(l4_size_t idx, Partition_info *inf) const
00073     {
00074         if (idx == 0 || idx > _num_partitions)
00075             return -L4_ERANGE;
00076
00077         unsigned secsz = _dev->sector_size();
00078         auto *header = _header.template get<Gpt::Header const>(secsz);
00079
00080         Gpt::Entry *e = _parray.template get<Gpt::Entry>((idx - 1) * header->entry_size);
00081
00082         if ((*((l4_uint64_t *) &e->partition_guid) == 0ULL)
00083             return -L4_ENODEV;
00084
00085         render_guid(e->partition_guid, inf->guid);
00086
00087         auto name =
00088             std::u16string((char16_t *)e->name, sizeof(e->name) / sizeof(e->name[0]));
00089         inf->name = name.substr(0, name.find((char16_t) 0));
00090
00091         inf->first = e->first;
00092         inf->last = e->last;
00093         inf->flags = e->flags;
00094
00095         auto info = Dbg::info();
00096         if (info.is_active())
00097         {
00098             info.printf("%3zu: %10lld %10lld %5gMiB [%3.7s]\n",
00099                         idx, e->first, e->last,
00100                         (e->last - e->first + 1.0) * secsz / (1 « 20),
00101                         inf->guid);
00102
00103             char buf[37];
00104             info.printf("    : Type: %s\n", render_guid(e->type_guid, buf));
00105         }
00106
00107         auto warn = Dbg::warn();
00108         if (inf->last < inf->first)
00109         {
00110             warn.printf(
00111                 "Invalid settings of %3zu. Last lba before first lba. Will ignore.\n",
00112                 idx);
00113             // Errors in the GPT shall not crash any service -- just ignore the
00114             // corresponding partition.
00115             return -L4_ENODEV;
00116         }
00117
00118         return L4_EOK;
00119     }
00120
00121 private:
00122     void invoke_callback()
00123     {
00124         assert(_callback);
00125         _callback();
00126         // Reset the callback to drop potential transitive self-references
00127         _callback = nullptr;
00128     }
00129
00130     void get_gpt(int error, l4_size_t)
00131     {
00132         _header.unmap();
00133     }

```



```

00134     if (error < 0)
00135     {
00136         // can't read from device, we are done
00137         invoke_callback();
00138         return;
00139     }
00140
00141     // prepare reading of the table from disk
00142     unsigned secsz = _dev->sector_size();
00143     auto *header = _header.template get<Gpt::Header const>(secsz);
00144
00145     auto info = Dbg::info();
00146     auto trace = Dbg::trace();
00147
00148     if (strncmp(header->signature, "EFI PART", 8) != 0)
00149     {
00150         info.printf("No GUID partition header found.\n");
00151         invoke_callback();
00152         return;
00153     }
00154
00155     // XXX check CRC32 of header
00156
00157     info.printf("GUID partition header found with up to %d partitions.\n",
00158         header->partition_array_size);
00159     char buf[37];
00160     info.printf("GUID: %s\n", render_guid(header->disk_guid, buf));
00161     trace.printf("Header positions: %llx (Backup: %llx)\n",
00162         header->current_lba, header->backup_lba);
00163     trace.printf("First + last: %llx and %llx\n",
00164         header->first_lba, header->last_lba);
00165     trace.printf("Partition table at %llx\n",
00166         header->partition_array_lba);
00167     trace.printf("Size of a partition entry: %d\n",
00168         header->entry_size);
00169
00170     info.printf("GUID partition header found with %d partitions.\n",
00171         header->partition_array_size);
00172
00173     _num_partitions = cxx::min<l4_uint32_t>(header->partition_array_size,
00174         Max_partitions);
00175
00176     l4_size_t arraysz = _num_partitions * header->entry_size;
00177     l4_size_t numsec = (arraysz - 1 + secsz) / secsz;
00178
00179     _parray = Inout_memory<Device_type>(numsec, _dev, L4Re::Dma_space::Direction::From_device);
00180     trace.printf("Reading GPT table @ 0x%p\n", _parray.template get<void>(0));
00181
00182     _db = _parray.inout_block();
00183     read_sectors(header->partition_array_lba, &Partition_reader::done_gpt);
00184 }
00185
00186 void done_gpt(int error, l4_size_t)
00187 {
00188     _parray.unmap();
00189
00190     // XXX check CRC32 of table
00191
00192     if (error < 0)
00193         _num_partitions = 0;
00194
00195     invoke_callback();
00196 }
00197
00198 void read_sectors(l4_uint64_t sector,
00199     void (Partition_reader::*func)(int, l4_size_t))
00200 {
00201     using namespace std::placeholders;
00202     auto next = std::bind(func, this, _1, _2);
00203
00204     l4_addr_t vstart = (l4_addr_t)_db.virt_addr;
00205     l4_addr_t vend = vstart + _db.num_sectors * _dev->sector_size();
00206     l4_cache_inv_data(vstart, vend);
00207
00208     Errand::poll(10, 10000,
00209         [=]()
00210         {
00211             int ret = _dev->inout_data(
00212                 sector, _db,
00213                 [next, vstart, vend](int error, l4_size_t size)
00214                 {
00215                     l4_cache_inv_data(vstart, vend);
00216                     next(error, size);
00217                 },
00218                 L4Re::Dma_space::Direction::From_device);
00219             if (ret < 0 && ret != -L4_EBUSY)
00220

```

```

00221         invoke_callback();
00222         return ret != -L4_EBUSY;
00223     },
00224     [=](bool ret) { if (!ret) invoke_callback(); }
00225 );
00226 }
00227
00228 static char const *render_guid(void const *guid_p, char buf[])
00229 {
00230     auto *p = static_cast<unsigned char const *>(guid_p);
00231     snprintf(buf, 37,
00232              "%02X%02X%02X%02X-%02X%02X-%02X%02X-%02X%02X%02X%02X%02X",
00233              p[3], p[2], p[1], p[0], p[5], p[4], p[7], p[6],
00234              p[8], p[9], p[10], p[11], p[12], p[13], p[14], p[15]);
00235
00236     return buf;
00237 }
00238
00239 l4_size_t _num_partitions;
00240 Inout_block _db;
00241 Device_type *_dev;
00242 Inout_memory<Device_type> _header;
00243 Inout_memory<Device_type> _pararray;
00244 Errand::Callback _callback;
00245 };
00246
00247
00248
00249 }

```

## 17.273 scheduler.h

```

00001 /*
00002  * Copyright (C) 2024 Kernkonzept GmbH.
00003  * Author(s): Jakub Jermar <jakub.jermar@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <vector>
00011
00012 #include <l4/cxx/unique_ptr>
00013 #include <l4/re/error_helper>
00014 #include <l4/sys/cxx/ipc_epiface>
00015
00016 #include <l4/libblock-device/debug.h>
00017 #include <l4/libblock-device/virtio_client.h>
00018
00019 namespace Block_device {
00020
00034 template <typename DEV>
00035 class Scheduler_base
00036 {
00037 protected:
00038     using Device_type = DEV;
00039     using Client_type = Virtio_client<Device_type>;
00040
00041 private:
00042     class Irq_object : public L4::Irqep_t<Irq_object>
00043     {
00044     public:
00045         Irq_object(Scheduler_base *parent) : _parent(parent) {}
00046
00047         void handle_irq() { _parent->schedule(); }
00048
00049 private:
00050         Scheduler_base *_parent;
00051     };
00052     Irq_object _irq_handler;
00053
00054     struct Context
00055     {
00056         cxx::unique_ptr<Pending_request> pending;
00057         Client_type *client;
00058
00059         bool device_busy;
00060         l4_size_t cost;
00061
00062         Context(Client_type *client) : client(client), device_busy(false), cost(0)
00063         {}
00064

```

```

00065     bool same_notification_domain(Client_type const *c) const
00066     { return c->notification_domain() == client->notification_domain(); }
00067 };
00068
00069 using Queue_type = std::vector<cxx::unique_ptr<Context>>;
00070 using Iterator_type = typename Queue_type::const_iterator;
00071
00072 public:
00073     Scheduler_base(L4::Registry_iface *registry)
00074     : _irq_handler(this), _registry(registry), _next(_clients.cend())
00075     {
00076         L4Re::chkcap(registry->register_irq_obj(&_irq_handler),
00077                     "Registering device notify IRQ object.");
00078     }
00079
00080     virtual ~Scheduler_base()
00081     {
00082         // We need to explicitly delete the IRQ object created in register_irq_obj()
00083         // ourselves. Even though unregister_obj() will unmap the cap, it might stay
00084         // alive because it was given out to the client. Hence it might be
00085         // dispatched even after unregister_obj() returned!
00086         L4::Cap<L4::Task>(L4Re::This_task)
00087             ->unmap(_irq_handler.obj_cap().fpage(),
00088                   L4_FP_ALL_SPACES | L4_FP_DELETE_OBJ);
00089         _registry->unregister_obj(&_irq_handler);
00090     }
00091
00095     virtual l4_size_t get_weight(Client_type const *) = 0;
00096
00100     virtual l4_size_t get_cost(Pending_request const &) = 0;
00101
00102     void add_client(Client_type *client)
00103     {
00104         Dbg::trace().printf("Adding client %p to request scheduler.\n", client);
00105
00106         // make sure the client uses the request scheduler's device_notify_irq
00107         client->set_device_notify_irq(
00108             L4::cap_cast<L4::Irq>(_irq_handler.obj_cap()));
00109
00110         client->set_client_invalidate_cb([this, client](bool fail_pending) {
00111             client_invalidate(client, fail_pending);
00112         });
00113
00114         client->set_client_idle_cb([this, client]() { client_idle(client); });
00115
00116         _clients.push_back(cxx::make_unique<Context>(client));
00117         _next = _clients.cend();
00118     }
00119
00120     void remove_client(Client_type *client)
00121     {
00122         Dbg::trace().printf("Removing client %p from request scheduler.\n", client);
00123         _clients.erase(std::remove_if(_clients.begin(), _clients.end(),
00124             [client](cxx::unique_ptr<Context> &c) {
00125                 return c->client == client;
00126             }));
00127         _next = _clients.cend();
00128     }
00129
00130     Queue_type const &clients()
00131     { return _clients; }
00132
00133 private:
00134     void client_invalidate(Client_type *client, bool fail_pending)
00135     {
00136         for (auto &c : _clients)
00137             if (c->client == client)
00138             {
00139                 c->device_busy = false;
00140                 c->cost = 0;
00141                 if (c->pending)
00142                 {
00143                     if (fail_pending)
00144                         c->pending->fail_request();
00145                     c->pending.reset();
00146                 }
00147             }
00148     }
00149
00150     void client_idle(Client_type *client)
00151     {
00152         bool resched = false;
00153         for (auto &c : _clients)
00154             if (c->device_busy && c->same_notification_domain(client))
00155             {
00156                 c->device_busy = false;
00157                 resched = true;
00158             }
00159     }

```

```

00160     }
00161
00162     if (resched)
00163     {
00164         // By triggering the scheduler asynchronously we make synchronous
00165         // request processing in the device implementation possible. In
00166         // any case we need to be careful not to start scheduling the
00167         // pending request which is being currently handled.
00168         L4::cap_cast<L4::Irq>(this->_irq_handler.obj_cap())->trigger();
00169     }
00170 }
00171
00181 bool handle_pending(Context *c)
00182 {
00183     auto cost = get_cost(*(c->pending));
00184
00185     if (c->cost + cost > get_weight(c->client))
00186     {
00187         Dbg::trace().printf("Preempting client %p (cost=%zu+%zu, weight=%zu)\n",
00188                             c->client, c->cost, cost, get_weight(c->client));
00189
00190         // Charge client's entire weight to force schedule() to give another
00191         // client a chance.
00192         c->cost = get_weight(c->client);
00193         return true;
00194     }
00195
00196     // Keep the pending request in its place while handling the request.
00197     // This helps to make sure that the scheduler will not try to schedule
00198     // new requests while handling the pending one.
00199     int ret = c->pending->handle_request();
00200     if (ret == -L4_EBUSY)
00201     {
00202         c->device_busy = true;
00203         return false;
00204     }
00205
00206     c->cost += cost;
00207
00208     if (ret < 0)
00209         c->pending.reset();
00210     else
00211         c->pending.release();
00212     return true;
00213 }
00214
00231 bool schedule_client(Context *c)
00232 {
00233     if (c->pending)
00234     {
00235         if (c->device_busy)
00236         {
00237             Dbg::trace().printf(
00238                 "Skipping pending request of client %p (busy).\n", c->client);
00239             return false;
00240         }
00241
00242         Dbg::trace().printf("Handling pending request of client %p.\n",
00243                             c->client);
00244         // The client has a pending request, we need to handle it first
00245         // before new requests can be processed. If we manage to handle it,
00246         // we need to check again in the next round for new requests.
00247         return handle_pending(c);
00248     }
00249
00250     if (c->client->check_for_new_requests())
00251     {
00252         auto req = c->client->get_request();
00253         if (req)
00254         {
00255             Dbg::trace().printf("Scheduling request from client %p.\n",
00256                                 c->client);
00257             c->pending = c->client->start_request(cxx::move(req));
00258             if (c->pending)
00259             {
00260                 // We processed one new request by turning it into a pending
00261                 // one and possibly sending it to the device (or not). We
00262                 // need to recheck only if the request was successfully sent
00263                 // to the device.
00264                 return handle_pending(c);
00265             }
00266             // We processed one new request immediately (e.g. failed
00267             // sanity check, runtime error or client state).
00268             return true;
00269         }
00270     }
00271 }

```

```

00272     return false;
00273 }
00274
00287 void schedule()
00288 {
00289     if (_clients.empty())
00290         return;
00291
00292     if (_next == _clients.cend())
00293         _next = _clients.cbegin();
00294
00295     (*_next)->cost = 0;
00296
00297     Iterator_type start(_next);
00298     bool recheck = false;
00299     for (;;)
00300     {
00301         bool progress = schedule_client(_next->get());
00302         // Move onto the next client only after the current client has depleted
00303         // its chances to process its queue or if it didn't make any forward
00304         // progress
00305         if (!progress || ((*_next)->cost >= get_weight((*_next)->client)))
00306         {
00307             ++_next;
00308             if (_next == _clients.cend())
00309                 _next = _clients.cbegin();
00310             (*_next)->cost = 0;
00311         }
00312         recheck |= progress;
00313         if (_next == start)
00314         {
00315             if (!recheck)
00316             {
00317                 // already processed all clients and requests, start with
00318                 // the next client next time
00319                 ++_next;
00320                 break;
00321             }
00322             else
00323                 recheck = false;
00324         }
00325     }
00326 }
00327
00328 L4::Registry_iface *_registry;
00329 Queue_type _clients;
00330 Iterator_type _next;
00331 };
00332
00339 template <typename DEV>
00340 struct Rr_scheduler : Scheduler_base<DEV>
00341 {
00342     using Scheduler_base<DEV>::Scheduler_base;
00343
00344     l4_size_t
00345     get_weight(typename Scheduler_base<DEV>::Client_type const *) override
00346     { return 1; }
00347
00348     l4_size_t get_cost(Pending_request const &) override
00349     { return 1; }
00350 };
00351
00352
00353 } // name space

```

## 17.274 l4/sys/scheduler.h File Reference

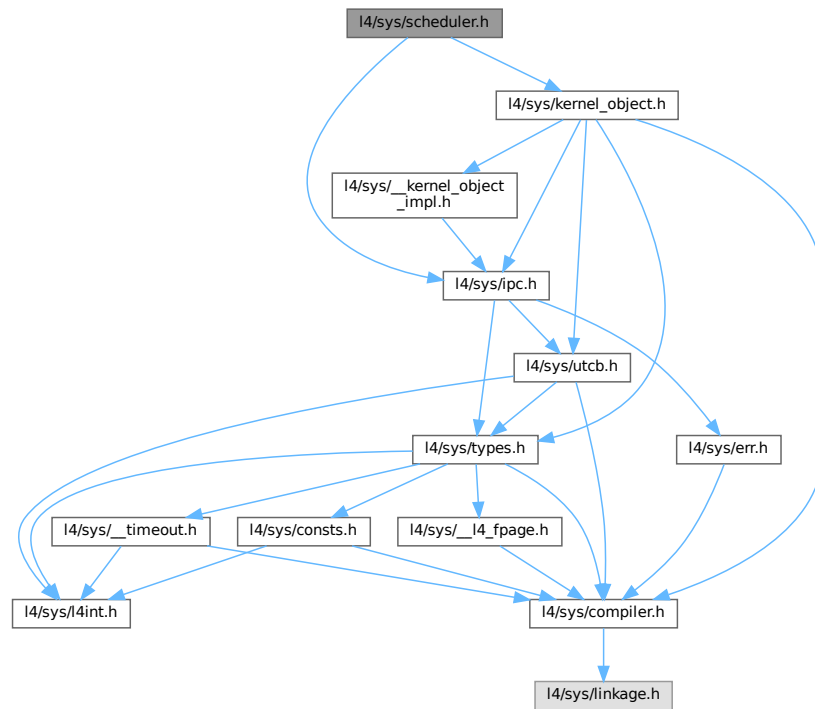
Scheduler object functions.

```

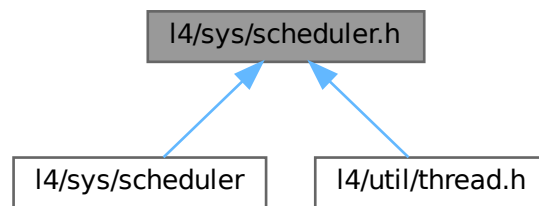
#include <l4/sys/kernel_object.h>
#include <l4/sys/ipc.h>

```

Include dependency graph for scheduler.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [l4\\_sched\\_cpu\\_set\\_t](#)  
*CPU sets.*
- struct [l4\\_sched\\_param\\_t](#)  
*Scheduler parameter set.*

## Typedefs

- typedef struct l4\_sched\_cpu\_set\_t **l4\_sched\_cpu\_set\_t**  
*CPU sets.*
- typedef struct l4\_sched\_param\_t **l4\_sched\_param\_t**  
*Scheduler parameter set.*

## Enumerations

- enum **L4\_scheduler\_classes** { **L4\_SCHEDULER\_CLASS\_FIXED\_PRIO** = 1UL << 1, **L4\_SCHEDULER\_CLASS\_WFQ** = 1UL << 2 }  
*Supported scheduler classes.*
- enum **L4\_scheduler\_ops** { **L4\_SCHEDULER\_INFO\_OP** = 0UL, **L4\_SCHEDULER\_RUN\_THREAD\_OP** = 1UL, **L4\_SCHEDULER\_IDLE\_TIME\_OP** = 2UL }  
*Operations on the Scheduler object.*

## Functions

- **l4\_sched\_cpu\_set\_t l4\_sched\_cpu\_set** (**l4\_umword\_t** offset, unsigned char granularity, **l4\_umword\_t** map=1) **L4\_NOTHROW**
- **l4\_msgtag\_t l4\_scheduler\_info** (**l4\_cap\_idx\_t** scheduler, **l4\_umword\_t** \*cpu\_max, **l4\_sched\_cpu\_set\_t** \*cpus) **L4\_NOTHROW**)  
*Get scheduler information.*
- **l4\_msgtag\_t l4\_scheduler\_info\_with\_classes** (**l4\_cap\_idx\_t** scheduler, **l4\_umword\_t** \*cpu\_max, **l4\_sched\_cpu\_set\_t** \*cpus, **l4\_umword\_t** \*sched\_classes) **L4\_NOTHROW**)  
*Get scheduler information.*
- **l4\_sched\_param\_t l4\_sched\_param** (unsigned prio, **l4\_umword\_t** quantum=0) **L4\_NOTHROW**  
*Construct scheduler parameter.*
- **l4\_msgtag\_t l4\_scheduler\_run\_thread** (**l4\_cap\_idx\_t** scheduler, **l4\_cap\_idx\_t** thread, **l4\_sched\_param\_t** const \*sp) **L4\_NOTHROW**)  
*Run a thread on a Scheduler.*
- **l4\_msgtag\_t l4\_scheduler\_idle\_time** (**l4\_cap\_idx\_t** scheduler, **l4\_sched\_cpu\_set\_t** const \*cpus, **l4\_kernel\_clock\_t** \*us) **L4\_NOTHROW**)  
*Query the idle time (in  $\mu$ s) of a CPU.*
- int **l4\_scheduler\_is\_online** (**l4\_cap\_idx\_t** scheduler, **l4\_umword\_t** cpu) **L4\_NOTHROW**  
*Query if a CPU is online.*

### 17.274.1 Detailed Description

Scheduler object functions.

Definition in file [scheduler.h](#).

## 17.275 scheduler.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 #include <l4/sys/kernel_object.h>
00015 #include <l4/sys/ipc.h>
00016
00040
00046 enum L4_scheduler_classes
00047 {
00049     L4_SCHEDULER_CLASS_FIXED_PRIO = 1UL < 1,
00051     L4_SCHEDULER_CLASS_WFQ        = 1UL < 2,
00052 };
00053
00058 typedef struct l4_sched_cpu_set_t
00059 {
00072     l4_umword_t gran_offset;
00073
00077     l4_umword_t map;
00078
00079 #ifdef __cplusplus
00081     unsigned char granularity() const { return gran_offset > 24; }
00083     unsigned offset() const { return gran_offset & 0x00ffffff; }
00090     void set(unsigned char granularity, unsigned offset)
00091     {
00092         gran_offset = (static_cast<l4_umword_t>(granularity) < 24)
00093             | (offset & 0x00ffffff);
00094     }
00095 #endif
00096 } l4_sched_cpu_set_t;
00097
00108 L4_INLINE l4_sched_cpu_set_t
00109 l4_sched_cpu_set(l4_umword_t offset, unsigned char granularity,
00110                 l4_umword_t map L4_DEFAULT_PARAM(1)) L4_NOTHROW;
00111
00128 L4_INLINE l4_msgtag_t
00129 l4_scheduler_info(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00130                  l4_sched_cpu_set_t *cpus)
00131     L4_NOTHROW __attribute__((nonnull (3)));
00132
00154 L4_INLINE l4_msgtag_t
00155 l4_scheduler_info_with_classes(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00156                               l4_sched_cpu_set_t *cpus,
00157                               l4_umword_t *sched_classes)
00158     L4_NOTHROW __attribute__((nonnull (3)));
00159
00163 L4_INLINE l4_msgtag_t
00164 l4_scheduler_info_u(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00165                    l4_sched_cpu_set_t *cpus, l4_umword_t *sched_classes,
00166                    l4_utcb_t *utcb) L4_NOTHROW __attribute__((nonnull (3, 5)));
00167
00168
00173 typedef struct l4_sched_param_t
00174 {
00176     l4_sched_cpu_set_t affinity;
00182     l4_umword_t prio;
00184     l4_umword_t quantum;
00185 } l4_sched_param_t;
00186
00195 L4_INLINE l4_sched_param_t
00196 l4_sched_param(unsigned prio,
00197                l4_umword_t quantum L4_DEFAULT_PARAM(0)) L4_NOTHROW;
00198
00206 L4_INLINE l4_msgtag_t
00207 l4_scheduler_run_thread(l4_cap_idx_t scheduler,
00208                        l4_cap_idx_t thread, l4_sched_param_t const *sp)
00209     L4_NOTHROW __attribute__((nonnull));
00210
00214 L4_INLINE l4_msgtag_t
00215 l4_scheduler_run_thread_u(l4_cap_idx_t scheduler, l4_cap_idx_t thread,
00216                           l4_sched_param_t const *sp, l4_utcb_t *utcb)
00217     L4_NOTHROW __attribute__((nonnull));
00218
00226 L4_INLINE l4_msgtag_t
00227 l4_scheduler_idle_time(l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *cpus,
00228                       l4_kernel_clock_t *us)

```



```

00229             L4_NOTHROW __attribute__((nonnull));
00230
00234 L4_INLINE l4_msgtag_t
00235 l4_scheduler_idle_time_u(l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *cpus,
00236                         l4_kernel_clock_t *us, l4_utcb_t *utcb)
00237             L4_NOTHROW __attribute__((nonnull));
00238
00239
00240
00251 L4_INLINE int
00252 l4_scheduler_is_online(l4_cap_idx_t scheduler, l4_umword_t cpu) L4_NOTHROW;
00253
00257 L4_INLINE int
00258 l4_scheduler_is_online_u(l4_cap_idx_t scheduler, l4_umword_t cpu,
00259                         l4_utcb_t *utcb) L4_NOTHROW __attribute__((nonnull));
00260
00261
00262
00269 enum l4_scheduler_ops
00270 {
00271     L4_SCHEDULER_INFO_OP      = 0UL,
00272     L4_SCHEDULER_RUN_THREAD_OP = 1UL,
00273     L4_SCHEDULER_IDLE_TIME_OP = 2UL,
00274 };
00275
00276 /***** Implementations *****/
00277
00278 L4_INLINE l4_sched_cpu_set_t
00279 l4_sched_cpu_set(l4_umword_t offset, unsigned char granularity,
00280                 l4_umword_t map) L4_NOTHROW
00281 {
00282     l4_sched_cpu_set_t cs;
00283     cs.gran_offset = ((l4_umword_t)granularity < 24) | (offset & 0x00ffffff);
00284     cs.map         = map;
00285     return cs;
00286 }
00287
00288 L4_INLINE l4_sched_param_t
00289 l4_sched_param(unsigned prio, l4_umword_t quantum) L4_NOTHROW
00290 {
00291     l4_sched_param_t sp;
00292     sp.prio          = prio;
00293     sp.quantum       = quantum;
00294     sp.affinity      = l4_sched_cpu_set(0, ~0, 1);
00295     return sp;
00296 }
00297
00298
00299 L4_INLINE l4_msgtag_t
00300 l4_scheduler_info_u(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00301                   l4_sched_cpu_set_t *cpus, l4_umword_t *sched_classes,
00302                   l4_utcb_t *utcb) L4_NOTHROW
00303 {
00304     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00305     l4_msgtag_t res;
00306
00307     m->mr[0] = L4_SCHEDULER_INFO_OP;
00308     m->mr[1] = cpus->gran_offset;
00309
00310     res = l4_ipc_call(scheduler, utcb, l4_msgtag(L4_PROTO_SCHEDULER, 2, 0, 0), L4_IPC_NEVER);
00311
00312     if (l4_msgtag_has_error(res))
00313         return res;
00314
00315     cpus->map = m->mr[0];
00316
00317     if (cpu_max)
00318         *cpu_max = m->mr[1];
00319
00320     if (sched_classes)
00321         *sched_classes = m->mr[2];
00322
00323     return res;
00324 }
00325
00326 L4_INLINE l4_msgtag_t
00327 l4_scheduler_run_thread_u(l4_cap_idx_t scheduler, l4_cap_idx_t thread,
00328                          l4_sched_param_t const *sp, l4_utcb_t *utcb) L4_NOTHROW
00329 {
00330     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00331     m->mr[0] = L4_SCHEDULER_RUN_THREAD_OP;
00332     m->mr[1] = sp->affinity.gran_offset;
00333     m->mr[2] = sp->affinity.map;
00334     m->mr[3] = sp->prio;
00335     m->mr[4] = sp->quantum;
00336     m->mr[5] = l4_map_obj_control(0, 0);
00337     m->mr[6] = l4_obj_fpage(thread, 0, L4_CAP_FPAGE_RWS).raw;

```

```

00338
00339     return l4_ipc_call(scheduler, utcb, l4_msgtag(L4_PROTO_SCHEDULER, 5, 1, 0), L4_IPC_NEVER);
00340 }
00341
00342 L4_INLINE l4_msgtag_t
00343 l4_scheduler_idle_time_u(l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *cpus,
00344                        l4_kernel_clock_t *us, l4_utcb_t *utcb) L4_NOTHROW
00345 {
00346     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00347     l4_msgtag_t res;
00348
00349     v->mr[0] = L4_SCHEDULER_IDLE_TIME_OP;
00350     v->mr[1] = cpus->gran_offset;
00351     v->mr[2] = cpus->map;
00352
00353     res = l4_ipc_call(scheduler, utcb,
00354                     l4_msgtag(L4_PROTO_SCHEDULER, 3, 0, 0), L4_IPC_NEVER);
00355
00356     if (l4_msgtag_has_error(res))
00357         return res;
00358
00359     *us = v->mr64[l4_utcb_mr64_idx(0)];
00360
00361     return res;
00362 }
00363
00364
00365 L4_INLINE int
00366 l4_scheduler_is_online_u(l4_cap_idx_t scheduler, l4_umword_t cpu,
00367                        l4_utcb_t *utcb) L4_NOTHROW
00368 {
00369     l4_sched_cpu_set_t s;
00370     l4_msgtag_t r;
00371     s.gran_offset = cpu;
00372     r = l4_scheduler_info_u(scheduler, NULL, &s, NULL, utcb);
00373     if (l4_msgtag_has_error(r) || l4_msgtag_label(r) < 0)
00374         return 0;
00375
00376     return s.map & 1;
00377 }
00378
00379
00380 L4_INLINE l4_msgtag_t
00381 l4_scheduler_info(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00382                  l4_sched_cpu_set_t *cpus) L4_NOTHROW
00383 {
00384     return l4_scheduler_info_u(scheduler, cpu_max, cpus, NULL, l4_utcb());
00385 }
00386
00387 L4_INLINE l4_msgtag_t
00388 l4_scheduler_info_with_classes(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00389                               l4_sched_cpu_set_t *cpus,
00390                               l4_umword_t *sched_classes) L4_NOTHROW
00391 {
00392     return l4_scheduler_info_u(scheduler, cpu_max, cpus, sched_classes, l4_utcb());
00393 }
00394
00395 L4_INLINE l4_msgtag_t
00396 l4_scheduler_run_thread(l4_cap_idx_t scheduler,
00397                        l4_cap_idx_t thread, l4_sched_param_t const *sp) L4_NOTHROW
00398 {
00399     return l4_scheduler_run_thread_u(scheduler, thread, sp, l4_utcb());
00400 }
00401
00402 L4_INLINE l4_msgtag_t
00403 l4_scheduler_idle_time(l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *cpus,
00404                        l4_kernel_clock_t *us) L4_NOTHROW
00405 {
00406     return l4_scheduler_idle_time_u(scheduler, cpus, us, l4_utcb());
00407 }
00408
00409 L4_INLINE int
00410 l4_scheduler_is_online(l4_cap_idx_t scheduler, l4_umword_t cpu) L4_NOTHROW
00411 {
00412     return l4_scheduler_is_online_u(scheduler, cpu, l4_utcb());
00413 }

```

## 17.276 types.h

```

00001 /*
00002  * (c) 2008-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  * License: see LICENSE.spdx (in this directory or the directories above)

```

```

00005  */
00006  #pragma once
00007
00008  #include <l4/vbus/vbus_types.h>
00009
00014  enum l4io_iomem_flags_t
00015  {
00016      L4IO_MEM_NONCACHED = 0,
00017      L4IO_MEM_CACHED   = 1,
00018      L4IO_MEM_USE_MTRR  = 2,
00019      L4IO_MEM_ATTR_MASK = 0xf,
00020
00021      // combinations
00022      L4IO_MEM_WRITE_COMBINED = L4IO_MEM_USE_MTRR | L4IO_MEM_CACHED,
00023
00024
00027      L4IO_MEM_USE_RESERVED_AREA = 0x40 « 8,
00029      L4IO_MEM_EAGER_MAP         = 0x80 « 8,
00030  };
00031
00036  enum l4io_device_types_t {
00037      L4IO_DEVICE_INVALID = 0,
00038      L4IO_DEVICE_PCI,
00039      L4IO_DEVICE_USB,
00040      L4IO_DEVICE_OTHER,
00041      L4IO_DEVICE_ANY = ~0
00042  };
00043
00048  enum l4io_resource_types_t {
00049      L4IO_RESOURCE_INVALID = L4VBUS_RESOURCE_INVALID,
00050      L4IO_RESOURCE_IRQ     = L4VBUS_RESOURCE_IRQ,
00051      L4IO_RESOURCE_MEM     = L4VBUS_RESOURCE_MEM,
00052      L4IO_RESOURCE_PORT    = L4VBUS_RESOURCE_PORT,
00053      L4IO_RESOURCE_ANY     = ~0
00054  };
00055
00056
00057  typedef l4vbus_device_handle_t l4io_device_handle_t;
00058  typedef unsigned l4io_resource_handle_t;
00059
00067  typedef l4vbus_resource_t l4io_resource_t;
00068
00072  typedef l4vbus_device_t l4io_device_t;

```

## 17.277 types.h

```

00001  /*
00002  * Copyright (C) 2018-2019, 2023-2024 Kernkonzept GmbH.
00003  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007  #pragma once
00008
00009  #include <functional>
00010
00011  #include <l4/cxx/unique_ptr>
00012  #include <l4/re/dma_space>
00013  #include <l4/l4virtio/server/l4virtio>
00014
00015  namespace Block_device {
00016
00018  enum Inout_flags
00019  {
00020      Inout_f_wb = 1,
00021      Inout_f_unmap = 2,
00022  };
00023
00024  enum Shutdown_type
00025  {
00027      Running = 0,
00029      // client had crashed.
00030      Client_gone,
00032      Client_shutdown,
00034      System_shutdown,
00036      System_suspend
00037  };
00038
00043  struct Dma_region_info
00044  {
00045      virtual ~Dma_region_info() = default;
00046  };
00047

```

```

00052 struct Mem_region_info
00053 {
00054     cxx::unique_ptr<Dma_region_info> dma_info;
00055 };
00056
00057 using Mem_region =
00058     L4virtio::Svr::Driver_mem_region_t<Mem_region_info>;
00059
00066 struct Inout_block
00067 {
00068     L4Re::Dma_space::Dma_addr dma_addr = 0;
00069     void *virt_addr = nullptr;
00071     l4_uint64_t sector = 0;
00072     l4_uint32_t num_sectors = 0;
00074     l4_uint32_t flags = 0;
00075     cxx::unique_ptr<Inout_block> next;
00076 };
00077
00078 typedef std::function<void(int, l4_size_t)> Inout_callback;
00079
00080 } // name space

```

## 17.278 l4/sys/types.h File Reference

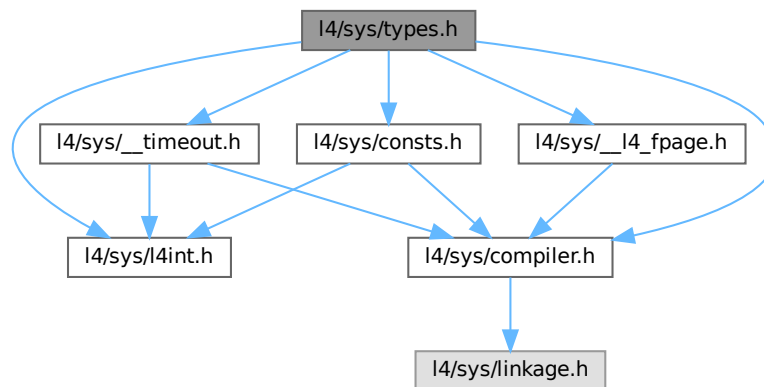
Common L4 ABI Data Types.

```

#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>
#include <l4/sys/consts.h>
#include <l4/sys/__l4_fpage.h>
#include <l4/sys/__timeout.h>

```

Include dependency graph for types.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [l4\\_msgtag\\_t](#)  
*Message tag data structure.*

## Typedefs

- typedef struct l4\_msgtag\_t [l4\\_msgtag\\_t](#)  
*Message tag data structure.*
- typedef unsigned long [l4\\_cap\\_idx\\_t](#)  
*Capability selector type.*

## Enumerations

- enum [L4\\_msgtag\\_protocol](#) {  
[L4\\_PROTO\\_NONE](#) = 0 , [L4\\_PROTO\\_ALLOW\\_SYSCALL](#) = 1 , [L4\\_PROTO\\_PF\\_EXCEPTION](#) = 1 ,  
[L4\\_PROTO\\_IRQ](#) = -1L ,  
[L4\\_PROTO\\_PAGE\\_FAULT](#) = -2L , [L4\\_PROTO\\_EXCEPTION](#) = -5L , [L4\\_PROTO\\_SIGMA0](#) = -6L ,  
[L4\\_PROTO\\_IO\\_PAGE\\_FAULT](#) = -8L ,  
[L4\\_PROTO\\_THREAD\\_GROUP](#) = -9L , [L4\\_PROTO\\_KOBJECT](#) = -10L , [L4\\_PROTO\\_TASK](#) = -11L ,  
[L4\\_PROTO\\_THREAD](#) = -12L ,  
[L4\\_PROTO\\_LOG](#) = -13L , [L4\\_PROTO\\_SCHEDULER](#) = -14L , [L4\\_PROTO\\_FACTORY](#) = -15L ,  
[L4\\_PROTO\\_VM](#) = -16L ,  
[L4\\_PROTO\\_DMA\\_SPACE](#) = -17L , [L4\\_PROTO\\_IRQ\\_SENDER](#) = -18L , [L4\\_PROTO\\_SEMAPHORE](#) = -20L ,  
[L4\\_PROTO\\_META](#) = -21L ,  
[L4\\_PROTO\\_IOMMU](#) = -22L , [L4\\_PROTO\\_DEBUGGER](#) = -23L , [L4\\_PROTO\\_SMCCC](#) = -24L ,  
[L4\\_PROTO\\_VCPU\\_CONTEXT](#) = -25L }  
*Message tag for IPC operations.*
- enum [L4\\_msgtag\\_flags](#) { [L4\\_MSGTAG\\_ERROR](#) , [L4\\_MSGTAG\\_TRANSFER\\_FPU](#) , [L4\\_MSGTAG\\_SCHEDULE](#)  
, [L4\\_MSGTAG\\_PROPAGATE](#) = 0x4000 , [L4\\_MSGTAG\\_FLAGS](#) }  
*Flags for message tags.*

## Functions

- [l4\\_msgtag\\_t l4\\_msgtag](#) (long label, unsigned words, unsigned items, unsigned flags) [L4\\_NOTHROW](#)  
*Create a message tag from the specified values.*
- long [l4\\_msgtag\\_label](#) ([l4\\_msgtag\\_t](#)) [L4\\_NOTHROW](#)  
*Get the protocol of tag.*
- unsigned [l4\\_msgtag\\_words](#) ([l4\\_msgtag\\_t](#)) [L4\\_NOTHROW](#)  
*Get the number of untyped words.*
- unsigned [l4\\_msgtag\\_items](#) ([l4\\_msgtag\\_t](#)) [L4\\_NOTHROW](#)  
*Get the number of typed items.*
- unsigned [l4\\_msgtag\\_flags](#) ([l4\\_msgtag\\_t](#)) [L4\\_NOTHROW](#)  
*Get the flags.*
- unsigned [l4\\_msgtag\\_has\\_error](#) ([l4\\_msgtag\\_t](#)) [L4\\_NOTHROW](#)  
*Test for error indicator flag.*
- unsigned [l4\\_msgtag\\_is\\_page\\_fault](#) ([l4\\_msgtag\\_t](#)) [L4\\_NOTHROW](#)  
*Test for page-fault protocol.*
- unsigned [l4\\_msgtag\\_is\\_exception](#) ([l4\\_msgtag\\_t](#)) [L4\\_NOTHROW](#)  
*Test for exception protocol.*
- unsigned [l4\\_msgtag\\_is\\_sigma0](#) ([l4\\_msgtag\\_t](#)) [L4\\_NOTHROW](#)

*Test for sigma0 protocol.*

- unsigned [l4\\_msgtag\\_is\\_io\\_page\\_fault](#) ([l4\\_msgtag\\_t](#) t) [L4\\_NOTHROW](#)

*Test for IO-page-fault protocol.*

- unsigned [l4\\_is\\_invalid\\_cap](#) ([l4\\_cap\\_idx\\_t](#) c) [L4\\_NOTHROW](#)

*Test if a capability selector is the invalid capability.*

- unsigned [l4\\_is\\_valid\\_cap](#) ([l4\\_cap\\_idx\\_t](#) c) [L4\\_NOTHROW](#)

*Test if a capability selector is a valid selector.*

- unsigned [l4\\_capability\\_equal](#) ([l4\\_cap\\_idx\\_t](#) c1, [l4\\_cap\\_idx\\_t](#) c2) [L4\\_NOTHROW](#)

*Test if the capability indices of two capability selectors are equal.*

- [l4\\_cap\\_idx\\_t](#) [l4\\_capability\\_next](#) ([l4\\_cap\\_idx\\_t](#) c) [L4\\_NOTHROW](#)

*Get the next capability selector after c.*

## 17.278.1 Detailed Description

Common [L4](#) ABI Data Types.

Definition in file [types.h](#).

## 17.278.2 Function Documentation

### 17.278.2.1 [l4\\_capability\\_next\(\)](#)

```
l4\_cap\_idx\_t l4\_capability\_next (  
    l4\_cap\_idx\_t c) \[inline\]
```

Get the next capability selector after c.

#### Parameters

<a href="#">c</a>	The capability selector for which the next selector shall be computed.
-------------------	------------------------------------------------------------------------

#### Returns

The next capability selector after c.

Definition at line [457](#) of file [types.h](#).

References [L4\\_CAP\\_OFFSET](#), [L4\\_INLINE](#), and [L4\\_NOTHROW](#).

## 17.279 types.h

[Go to the documentation of this file.](#)

```

00001 /*****
00007 */
00008 * (c) 2008-2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00010 *      Björn Döbel <doebel@os.inf.tu-dresden.de>,
00011 *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00012 *      economic rights: Technische Universität Dresden (Germany)
00013 *
00014 * License: see LICENSE.spdx (in this directory or the directories above)
00015 */
00016 /*****
00017 #pragma once
00018
00019 #include <l4/sys/l4int.h>
00020 #include <l4/sys/compiler.h>
00021 #include <l4/sys/consts.h>
00022
00023
00031
00038 enum L4_msgtag_protocol
00039 {
00040     L4_PROTO_NONE           = 0,
00041     L4_PROTO_ALLOW_SYSCALL = 1,
00042     L4_PROTO_PF_EXCEPTION  = 1,
00043
00044     L4_PROTO_IRQ           = -1L,
00045     L4_PROTO_PAGE_FAULT   = -2L,
00046     // -3L unused
00047     // -4L unused
00048     L4_PROTO_EXCEPTION    = -5L,
00049     L4_PROTO_SIGMA0       = -6L,
00050     L4_PROTO_IO_PAGE_FAULT = -8L,
00051     L4_PROTO_THREAD_GROUP = -9L,
00052     L4_PROTO_KOBJECT      = -10L,
00053     L4_PROTO_TASK         = -11L,
00054     L4_PROTO_THREAD       = -12L,
00055     L4_PROTO_LOG          = -13L,
00056     L4_PROTO_SCHEDULER    = -14L,
00057     L4_PROTO_FACTORY      = -15L,
00058     L4_PROTO_VM           = -16L,
00059     L4_PROTO_DMA_SPACE    = -17L,
00060     L4_PROTO_IRQ_SENDER   = -18L,
00061     // -19L unused
00062     L4_PROTO_SEMAPHORE    = -20L,
00063     L4_PROTO_META         = -21L,
00064     L4_PROTO_IOMMU        = -22L,
00065     L4_PROTO_DEBUGGER     = -23L,
00066     L4_PROTO_SMCCC        = -24L,
00067     L4_PROTO_VCPU_CONTEXT = -25L,
00068 };
00069
00070 enum L4_varg_type
00071 {
00072     L4_VARG_TYPE_NIL      = 0x00,
00073     L4_VARG_TYPE_UMWORD   = 0x01,
00074     L4_VARG_TYPE_MWORD    = 0x81,
00075     L4_VARG_TYPE_STRING   = 0x02,
00076     L4_VARG_TYPE_FPAGE    = 0x03,
00077
00078     L4_VARG_TYPE_SIGN     = 0x80,
00079 };
00080
00081
00086 enum L4_msgtag_flags
00087 {
00088     // flags for received IPC
00093     L4_MSGTAG_ERROR      = 0x8000,
00094
00095     // flags for sending IPC
00105     L4_MSGTAG_TRANSFER_FPU = 0x1000,
00114     L4_MSGTAG_SCHEDULE    = 0x2000,
00127     L4_MSGTAG_PROPAGATE   = 0x4000,
00128
00133     L4_MSGTAG_FLAGS       = 0xf000,
00134 };
00135
00136
00153 typedef struct l4_msgtag_t
00154 {
00155     l4_mword_t raw;
00156 #ifdef __cplusplus
00158     long label() const L4_NOTHROW

```

```

00159 {
00160 #if defined(__cplusplus) && (__cplusplus >= 202002L)
00161     return raw >> 16;
00162 #else
00163     return raw < 0 ? ~(~raw >> 16) : raw >> 16;
00164 #endif
00165 }
00167 void label(long v) L4_NOTHROW { raw = (raw & 0xffff) | ((l4_umword_t)v << 16); }
00169 unsigned words() const L4_NOTHROW { return raw & 0x3f; }
00171 unsigned items() const L4_NOTHROW { return (raw >> 6) & 0x3f; }
00178 unsigned flags() const L4_NOTHROW { return raw & 0xf000; }
00180 bool is_page_fault() const L4_NOTHROW { return label() == L4_PROTO_PAGE_FAULT; }
00182 bool is_exception() const L4_NOTHROW { return label() == L4_PROTO_EXCEPTION; }
00184 bool is_sigma0() const L4_NOTHROW { return label() == L4_PROTO_SIGMA0; }
00186 bool is_io_page_fault() const L4_NOTHROW { return label() == L4_PROTO_IO_PAGE_FAULT; }
00191 bool has_error() const L4_NOTHROW { return raw & L4_MSGTAG_ERROR; }
00192 #endif
00193 } l4_msgtag_t;
00194
00195
00196
00208 L4_INLINE l4_msgtag_t l4_msgtag(long label, unsigned words, unsigned items,
00209                                unsigned flags) L4_NOTHROW;
00210
00219 L4_INLINE long l4_msgtag_label(l4_msgtag_t t) L4_NOTHROW;
00220
00229 L4_INLINE unsigned l4_msgtag_words(l4_msgtag_t t) L4_NOTHROW;
00230
00239 L4_INLINE unsigned l4_msgtag_items(l4_msgtag_t t) L4_NOTHROW;
00240
00251 L4_INLINE unsigned l4_msgtag_flags(l4_msgtag_t t) L4_NOTHROW;
00252
00265 L4_INLINE unsigned l4_msgtag_has_error(l4_msgtag_t t) L4_NOTHROW;
00266
00275 L4_INLINE unsigned l4_msgtag_is_page_fault(l4_msgtag_t t) L4_NOTHROW;
00276
00285 L4_INLINE unsigned l4_msgtag_is_exception(l4_msgtag_t t) L4_NOTHROW;
00286
00295 L4_INLINE unsigned l4_msgtag_is_sigma0(l4_msgtag_t t) L4_NOTHROW;
00296
00305 L4_INLINE unsigned l4_msgtag_is_io_page_fault(l4_msgtag_t t) L4_NOTHROW;
00306
00319
00336 typedef unsigned long l4_cap_idx_t;
00337
00347 L4_INLINE unsigned l4_is_invalid_cap(l4_cap_idx_t c) L4_NOTHROW;
00348
00358 L4_INLINE unsigned l4_is_valid_cap(l4_cap_idx_t c) L4_NOTHROW;
00359
00373 L4_INLINE unsigned l4_capability_equal(l4_cap_idx_t c1, l4_cap_idx_t c2) L4_NOTHROW;
00374
00383 L4_INLINE l4_cap_idx_t l4_capability_next(l4_cap_idx_t c) L4_NOTHROW;
00384
00385 /* ***** */
00386 /* Implementation */
00387
00388 L4_INLINE unsigned
00389 l4_is_invalid_cap(l4_cap_idx_t c) L4_NOTHROW
00390 { return c & L4_INVALID_CAP_BIT; }
00391
00392 L4_INLINE unsigned
00393 l4_is_valid_cap(l4_cap_idx_t c) L4_NOTHROW
00394 { return !(c & L4_INVALID_CAP_BIT); }
00395
00396 L4_INLINE unsigned
00397 l4_capability_equal(l4_cap_idx_t c1, l4_cap_idx_t c2) L4_NOTHROW
00398 { return (c1 >> L4_CAP_SHIFT) == (c2 >> L4_CAP_SHIFT); }
00399
00400
00404 L4_INLINE
00405 l4_msgtag_t l4_msgtag(long label, unsigned words, unsigned items,
00406                      unsigned flags) L4_NOTHROW
00407 {
00408     return (l4_msgtag_t){ (l4_mword_t)((l4_umword_t)label << 16)
00409                          | (l4_mword_t)(words & 0x3f)
00410                          | (l4_mword_t)((items & 0x3f) << 6)
00411                          | (l4_mword_t)(flags & 0xf000)};
00412 }
00413
00414
00415
00416 L4_INLINE
00417 long l4_msgtag_label(l4_msgtag_t t) L4_NOTHROW
00418 {
00419     #if defined(__cplusplus) && (__cplusplus >= 202002L)
00420         return t.raw >> 16;
00421     #else

```



```

00422     return t.raw < 0 ? ~(~t.raw » 16) : t.raw » 16;
00423 #endif
00424 }
00425
00426 L4_INLINE
00427 unsigned l4_msgtag_words(l4_msgtag_t t) L4_NOTHROW
00428 { return t.raw & 0x3f; }
00429
00430 L4_INLINE
00431 unsigned l4_msgtag_items(l4_msgtag_t t) L4_NOTHROW
00432 { return (t.raw » 6) & 0x3f; }
00433
00434 L4_INLINE
00435 unsigned l4_msgtag_flags(l4_msgtag_t t) L4_NOTHROW
00436 { return t.raw & 0xf000; }
00437
00438
00439 L4_INLINE
00440 unsigned l4_msgtag_has_error(l4_msgtag_t t) L4_NOTHROW
00441 { return t.raw & L4_MSGTAG_ERROR; }
00442
00443
00444
00445 L4_INLINE unsigned l4_msgtag_is_page_fault(l4_msgtag_t t) L4_NOTHROW
00446 { return l4_msgtag_label(t) == L4_PROTO_PAGE_FAULT; }
00447
00448 L4_INLINE unsigned l4_msgtag_is_exception(l4_msgtag_t t) L4_NOTHROW
00449 { return l4_msgtag_label(t) == L4_PROTO_EXCEPTION; }
00450
00451 L4_INLINE unsigned l4_msgtag_is_sigma0(l4_msgtag_t t) L4_NOTHROW
00452 { return l4_msgtag_label(t) == L4_PROTO_SIGMA0; }
00453
00454 L4_INLINE unsigned l4_msgtag_is_io_page_fault(l4_msgtag_t t) L4_NOTHROW
00455 { return l4_msgtag_label(t) == L4_PROTO_IO_PAGE_FAULT; }
00456
00457 L4_INLINE l4_cap_idx_t l4_capability_next(l4_cap_idx_t c) L4_NOTHROW
00458 { return c + L4_CAP_OFFSET; }
00459
00460 #include <l4/sys/__l4_fpage.h>
00461 #include <l4/sys/__timeout.h>

```

## 17.280 virtio\_client.h

```

00001 /*
00002  * Copyright (C) 2018-2024 Kernkonzept GmbH.
00003  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/cxx/ref_ptr>
00010 #include <l4/cxx/unique_ptr_list>
00011 #include <l4/cxx/utis>
00012 #include <l4/sys/cache.h>
00013
00014 #include <l4/sys/task>
00015
00016 #include <l4/l4virtio/server/virtio-block>
00017
00018 #include <l4/libblock-device/debug.h>
00019 #include <l4/libblock-device/device.h>
00020 #include <l4/libblock-device/types.h>
00021 #include <l4/libblock-device/request.h>
00022
00023 namespace Block_device {
00024
00025 template <typename DEV>
00026 class Virtio_client
00027 : public L4virtio::Svr::Block_dev_base<Mem_region_info>,
00028     public L4::Epiface_t<Virtio_client<DEV>, L4virtio::Device>
00029 {
00030 protected:
00031     class Generic_pending_request : public Pending_request
00032     {
00033     protected:
00034         int check_error(int result)
00035         {
00036             if (result < 0 && result != -L4_EBUSY)
00037                 client->handle_request_error(result, this);
00038
00039             return result;
00040         }
00041     };

```

```

00041
00042 public:
00043     explicit Generic_pending_request(Virtio_client *c, cxx::unique_ptr<Request> &&req)
00044         : request(cxx::move(req)), client(c)
00045     {}
00046
00047     void fail_request() override
00048     {
00049         client->finalize_request(cxx::move(request), 0, L4VIRTIO_BLOCK_S_IOERR);
00050     }
00051
00052     cxx::unique_ptr<Request> request;
00053     Virtio_client *client;
00054 };
00055
00056 struct Pending_inout_request : public Generic_pending_request
00057 {
00058     Inout_block blocks;
00059     L4Re::Dma_space::Direction dir;
00060
00061     explicit Pending_inout_request(Virtio_client *c,
00062                                     cxx::unique_ptr<Request> &&req)
00063         : Generic_pending_request(c, cxx::move(req))
00064     {
00065         dir = this->request->header().type == L4VIRTIO_BLOCK_T_OUT
00066             ? L4Re::Dma_space::Direction::To_device
00067             : L4Re::Dma_space::Direction::From_device;
00068     }
00069
00070     ~Pending_inout_request() override
00071     {
00072         this->client->release_dma(this);
00073     }
00074
00075     int handle_request() override
00076     { return this->check_error(this->client->inout_request(this)); }
00077 };
00078
00079 struct Pending_flush_request : public Generic_pending_request
00080 {
00081     using Generic_pending_request::Generic_pending_request;
00082
00083     int handle_request() override
00084     { return this->check_error(this->client->flush_request(this)); }
00085 };
00086
00087 struct Pending_cmd_request : public Generic_pending_request
00088 {
00089     Inout_block blocks;
00090
00091     using Generic_pending_request::Generic_pending_request;
00092
00093     int handle_request() override
00094     {
00095         return this->check_error(this->client->discard_cmd_request(this, 0));
00096     }
00097 };
00098
00099 public:
00100     using Device_type = DEV;
00101
00102     Virtio_client(cxx::Ref_ptr<Device_type> const &dev, unsigned numds, bool readonly)
00103         : L4virtio::Svr::Block_dev_base<Mem_region_info>(L4VIRTIO_VENDOR_KK, 0x100,
00104                                                         dev->capacity() > 9,
00105                                                         dev->is_read_only()
00106                                                         || readonly),
00107             _client_invalidate_cb(nullptr),
00108             _client_idle_cb(nullptr),
00109             _numds(numds),
00110             _device(dev),
00111             _in_flight(0)
00112     {
00113         reset_client();
00114         init_discard_info(0);
00115     }
00116
00117     void reset_device() override
00118     {
00119         if (_client_invalidate_cb)
00120             _client_invalidate_cb(false);
00121         _device->reset();
00122         _negotiated_features.raw = 0;
00123     }
00124
00125     void reset_client()
00126     {
00127         init_mem_info(_numds);

```

```

00142     set_seg_max(_device->max_segments());
00143     set_size_max(_device->max_size());
00144     set_flush();
00145     set_config_wce(0); // starting in write-through mode
00146     _shutdown_state = Shutdown_type::Running;
00147     _negotiated_features.raw = 0;
00148 }
00149
00150 bool queue_stopped() override
00151 { return _shutdown_state == Shutdown_type::Client_gone; }
00152
00153 // make these interfaces public so that a request scheduler can invoke them
00154 using L4virtio::Svr::Block_dev_base<Mem_region_info>::check_for_new_requests;
00155 using L4virtio::Svr::Block_dev_base<Mem_region_info>::get_request;
00156
00157 // make it possible for the request scheduler to register a direct callback
00158 void set_client_invalidate_cb(std::function<void(bool)> &&cb)
00159 {
00160     _client_invalidate_cb = cb;
00161 }
00162
00163 void set_client_idle_cb(std::function<void()> &&cb)
00164 {
00165     _client_idle_cb = cb;
00166 }
00167
00168 // make it possible for the request scheduler to register a device notify IRQ
00169 void set_device_notify_irq(L4::Cap<L4::Irq> irq)
00170 {
00171     _device_notify_irq = irq;
00172 }
00173
00174 L4::Cap<L4::Irq> device_notify_irq() const override
00175 {
00176     return _device_notify_irq;
00177 }
00178
00184 cxx::unique_ptr<Pending_request> start_request(cxx::unique_ptr<Request> &&req)
00185 {
00186     auto trace = Dbg::trace("virtio");
00187
00188     cxx::unique_ptr<Pending_request> pending;
00189
00190     if (_shutdown_state != Shutdown_type::Running)
00191     {
00192         trace.printf("Failing requests as the client is shutting down\n");
00193         this->finalize_request(cxx::move(req), 0, L4VIRTIO_BLOCK_S_IOERR);
00194         return pending;
00195     }
00196
00197     trace.printf("request received: type 0x%x, sector 0x%llx\n",
00198                 req->header().type, req->header().sector);
00199     switch (req->header().type)
00200     {
00201     case L4VIRTIO_BLOCK_T_OUT:
00202     case L4VIRTIO_BLOCK_T_IN:
00203     {
00204         auto p = cxx::make_unique<Pending_inout_request>(this, cxx::move(req));
00205         int ret = build_inout_blocks(p.get());
00206         if (ret == L4_EOK)
00207             pending.reset(p.release());
00208         else
00209             handle_request_error(ret, p.get());
00210         break;
00211     }
00212     case L4VIRTIO_BLOCK_T_FLUSH:
00213     {
00214         auto p = cxx::make_unique<Pending_flush_request>(this, cxx::move(req));
00215         int ret = check_flush_request(p.get());
00216         if (ret == L4_EOK)
00217             pending.reset(p.release());
00218         else
00219             handle_request_error(ret, p.get());
00220         break;
00221     }
00222     case L4VIRTIO_BLOCK_T_WRITE_ZEROES:
00223     case L4VIRTIO_BLOCK_T_DISCARD:
00224     {
00225         auto p = cxx::make_unique<Pending_cmd_request>(this, cxx::move(req));
00226         int ret = build_discard_cmd_blocks(p.get());
00227         if (ret == L4_EOK)
00228             pending.reset(p.release());
00229         else
00230             handle_request_error(ret, p.get());
00231         break;
00232     }
00233     default:

```

```

00234         finalize_request(cx::move(req), 0, L4VIRTIO_BLOCK_S_UNSUPP);
00235         break;
00236     }
00237
00238     return pending;
00239 }
00240
00241 void task_finished(Generic_pending_request *preq, int error, l4_size_t sz)
00242 {
00243     _in_flight--;
00244
00245     // move on to the next request
00246
00247     // Only finalize if the client is still alive
00248     if (_shutdown_state != Client_gone)
00249         finalize_request(cx::move(preq->request), sz, error);
00250
00251     // New requests might be schedulable
00252     if (_client_idle_cb)
00253         _client_idle_cb();
00254
00255     // pending request can be dropped
00256     cx::unique_ptr<Pending_request> ureq(preq);
00257 }
00258
00259 void shutdown_event(Shutdown_type type)
00260 {
00261     // If the client is already in the Client_gone state, it means that it was
00262     // already shutdown and this is another go at its removal. This situation
00263     // can occur because at the time of its previous removal attempt there were
00264     // still I/O requests in progress.
00265     if (_shutdown_state == Client_gone)
00266         return;
00267
00268     // Transitions from System_shutdown are also not allowed, the initiator
00269     // should take care of graceful handling of this.
00270     l4_assert(_shutdown_state != System_shutdown);
00271     // If we are transitioning from System_suspend, it must be only to Running,
00272     // the initiator should handle this gracefully.
00273     l4_assert(_shutdown_state != System_suspend
00274             || type == Shutdown_type::Running);
00275
00276     // Update shutdown state of the client
00277     _shutdown_state = type;
00278
00279     if (type == Shutdown_type::Client_shutdown)
00280     {
00281         reset();
00282         reset_client();
00283         // Client_shutdown must transit to the Running state
00284         l4_assert(_shutdown_state == Shutdown_type::Running);
00285     }
00286
00287     if (type != Shutdown_type::Running)
00288     {
00289         if (_client_invalidate_cb)
00290             _client_invalidate_cb(type != Shutdown_type::Client_gone);
00291         _device->reset();
00292     }
00293 }
00294
00295 L4::Cap<void> register_obj(L4::Registry_iface *registry,
00296                           char const *service = 0)
00297 {
00298     L4::Cap<void> ret;
00299     if (service)
00300         ret = registry->register_obj(this, service);
00301     else
00302         ret = registry->register_obj(this);
00303     L4Re::chkcap(ret);
00304
00305     return ret;
00306 }
00307
00308 L4::Cap<void> register_obj(L4::Registry_iface *registry,
00309                           L4::Cap<L4::Rcv_endpoint> ep)
00310 {
00311     return L4Re::chkcap(registry->register_obj(this, ep));
00312 }
00313
00314 void unregister_obj(L4::Registry_iface *registry)
00315 {
00316     registry->unregister_obj(this);
00317 }
00318
00319 bool busy() const
00320 {

```

```

00341     return _in_flight != 0;
00342 }
00343
00344 Notification_domain const *notification_domain() const
00345 { return _device->notification_domain(); }
00346
00347 protected:
00348     L4::Ipc_svr::Server_iface *server_iface() const override
00349     {
00350         return this->L4::Epiface::server_iface();
00351     }
00352
00353 private:
00354     void release_dma(Pending_inout_request *req)
00355     {
00356         // unmap DMA regions
00357         Inout_block *cur = &req->blocks;
00358         while (cur)
00359         {
00360             if (cur->num_sectors)
00361                 _device->dma_unmap(cur->dma_addr, cur->num_sectors, req->dir);
00362             cur = cur->next.get();
00363         }
00364     }
00365
00366     int build_inout_blocks(Pending_inout_request *preq)
00367     {
00368         auto *req = preq->request.get();
00369         l4_size_t sps = _device->sector_size() >> 9;
00370         l4_uint64_t current_sector = req->header().sector / sps;
00371         l4_uint64_t sectors = _device->capacity() / _device->sector_size();
00372         auto dir = preq->dir;
00373
00374         l4_uint32_t flags = 0;
00375         if (req->header().type == L4VIRTIO_BLOCK_T_OUT)
00376         {
00377             // If RO was offered, every write must fail
00378             if (device_features().ro())
00379                 return -L4_EIO;
00380
00381             // Figure out whether the write has a write-through or write-back semantics
00382             if (_negotiated_features.config_wce())
00383             {
00384                 if (get_writeback() == 1)
00385                     flags = Block_device::Inout_f_wb;
00386             }
00387             else if (_negotiated_features.flush())
00388                 flags = Block_device::Inout_f_wb;
00389         }
00390
00391         // Check alignment of the first sector
00392         if (current_sector * sps != req->header().sector)
00393             return -L4_EIO;
00394
00395         Inout_block *last_blk = nullptr;
00396
00397         size_t seg = 0;
00398
00399         while (req->has_more())
00400         {
00401             Request::Data_block b;
00402
00403             if (++seg > _device->max_segments())
00404                 return -L4_EIO;
00405
00406             try
00407             {
00408                 b = req->next_block();
00409             }
00410             catch (L4virtio::Svr::Bad_descriptor const &e)
00411             {
00412                 Dbg::warn().printf("Descriptor error: %s\n", e.message());
00413                 return -L4_EIO;
00414             }
00415
00416             l4_size_t off = b.mem->ds_offset() + (l4_addr_t) b.addr
00417                 - (l4_addr_t) b.mem->local_base();
00418
00419             l4_size_t sz = b.len / _device->sector_size();
00420
00421             if (sz * _device->sector_size() != b.len)
00422             {
00423                 Dbg::warn().printf("Bad block size 0x%x\n", b.len);
00424                 return -L4_EIO;
00425             }
00426
00427             // Check bounds

```

```

00428         if (sz > sectors)
00429             return -L4_EIO;
00430         if (current_sector > sectors - sz)
00431             return -L4_EIO;
00432
00433         Inout_block *blk;
00434         if (last_blk)
00435             {
00436                 last_blk->next = cxx::make_unique<Inout_block>();
00437                 blk = last_blk->next.get();
00438             }
00439         else
00440             blk = &preq->blocks;
00441
00442         L4Re::Dma_space::Dma_addr phys;
00443         long ret = _device->dma_map(b.mem, off, sz, dir, &phys);
00444         if (ret < 0)
00445             return ret;
00446
00447         blk->dma_addr = phys;
00448         blk->virt_addr = b.addr;
00449         blk->num_sectors = sz;
00450         current_sector += sz;
00451         blk->flags = flags;
00452
00453         last_blk = blk;
00454     }
00455
00456     return L4_EOK;
00457 }
00458
00459 void maintain_cache_before_req(Pending_inout_request const *preq)
00460 {
00461     if (preq->dir == L4Re::Dma_space::None)
00462         return;
00463     for (Inout_block const *cur = &preq->blocks; cur; cur = cur->next.get())
00464     {
00465         l4_addr_t vstart = (l4_addr_t)cur->virt_addr;
00466         if (vstart)
00467             {
00468                 l4_size_t vsize = cur->num_sectors * _device->sector_size();
00469                 if (preq->dir == L4Re::Dma_space::From_device)
00470                     l4_cache_inv_data(vstart, vstart + vsize);
00471                 else if (preq->dir == L4Re::Dma_space::To_device)
00472                     l4_cache_clean_data(vstart, vstart + vsize);
00473                 else // L4Re::Dma_space::Bidirectional
00474                     l4_cache_flush_data(vstart, vstart + vsize);
00475             }
00476     }
00477 }
00478
00479 void maintain_cache_after_req(Pending_inout_request const *preq)
00480 {
00481     if (preq->dir == L4Re::Dma_space::None)
00482         return;
00483     for (Inout_block const *cur = &preq->blocks; cur; cur = cur->next.get())
00484     {
00485         l4_addr_t vstart = (l4_addr_t)cur->virt_addr;
00486         if (vstart)
00487             {
00488                 l4_size_t vsize = cur->num_sectors * _device->sector_size();
00489                 if (preq->dir != L4Re::Dma_space::To_device)
00490                     l4_cache_inv_data(vstart, vstart + vsize);
00491             }
00492     }
00493 }
00494
00495 int inout_request(Pending_inout_request *preq)
00496 {
00497     auto *req = preq->request.get();
00498     l4_uint64_t sector = req->header().sector / (_device->sector_size() >> 9);
00499
00500     maintain_cache_before_req(preq);
00501     int res = _device->inout_data(
00502         sector, preq->blocks,
00503         [this, preq](int error, l4_size_t sz) {
00504             maintain_cache_after_req(preq);
00505             task_finished(preq, error, sz);
00506         },
00507         preq->dir);
00508
00509     // request successfully submitted to device
00510     if (res >= 0)
00511         _in_flight++;
00512
00513     return res;
00514 }

```

```

00515
00516 int check_flush_request(Pending_flush_request *preq)
00517 {
00518     if (!_negotiated_features.flush())
00519         return -L4_ENOSYS;
00520
00521     auto *req = preq->request.get();
00522
00523     // sector must be zero for FLUSH
00524     if (req->header().sector)
00525         return -L4_ENOSYS;
00526
00527     return L4_EOK;
00528 }
00529
00530 int flush_request(Pending_flush_request *preq)
00531 {
00532     int res = _device->flush([this, preq](int error, l4_size_t sz) {
00533         task_finished(preq, error, sz);
00534     });
00535
00536     // request successfully submitted to device
00537     if (res >= 0)
00538         _in_flight++;
00539
00540     return res;
00541 }
00542
00543 bool check_features(void) override
00544 {
00545     _negotiated_features = negotiated_features();
00546     return true;
00547 }
00548
00549 template <typename T = Device_type>
00550 void init_discard_info(long) {}
00551
00552 template <typename T = Device_type>
00553 auto init_discard_info(int)
00554     -> decltype(((T*)0)->discard_info(), void())
00555 {
00556     _di = _device->discard_info();
00557
00558     // Convert sector sizes to virtio 512-byte sectors.
00559     size_t sps = _device->sector_size() >> 9;
00560     if (_di.max_discard_sectors)
00561         set_discard(_di.max_discard_sectors * sps, _di.max_discard_seg,
00562                     _di.discard_sector_alignment * sps);
00563     if (_di.max_write_zeroes_sectors)
00564         set_write_zeroes(_di.max_write_zeroes_sectors * sps,
00565                           _di.max_write_zeroes_seg, _di.write_zeroes_may_unmap);
00566 }
00567
00568 int build_discard_cmd_blocks(Pending_cmd_request *preq)
00569 {
00570     auto *req = preq->request.get();
00571     bool discard = (req->header().type == L4VIRTIO_BLOCK_T_DISCARD);
00572
00573     if (this->device_features().ro())
00574         return -L4_EIO;
00575
00576     // sector is used only for inout requests, it must be zero for WzD
00577     if (req->header().sector)
00578         return -L4_ENOSYS;
00579
00580     if (discard)
00581     {
00582         if (!_negotiated_features.discard())
00583             return -L4_ENOSYS;
00584     }
00585     else
00586     {
00587         if (!_negotiated_features.write_zeroes())
00588             return -L4_ENOSYS;
00589     }
00590
00591     auto *d = _device.get();
00592
00593     size_t seg = 0;
00594     size_t max_seg = discard ? _di.max_discard_seg : _di.max_write_zeroes_seg;
00595
00596     l4_size_t sps = d->sector_size() >> 9;
00597     l4_uint64_t sectors = d->capacity() / d->sector_size();
00598
00599     Inout_block *last_blk = nullptr;
00600
00601     while (req->has_more())

```

```

00602     {
00603         Request::Data_block b;
00604
00605         try
00606         {
00607             b = req->next_block();
00608         }
00609         catch (L4virtio::Svr::Bad_descriptor const &e)
00610         {
00611             Dbg::warn().printf("Descriptor error: %s\n", e.message());
00612             return -L4_EIO;
00613         }
00614
00615         auto *payload = reinterpret_cast<l4virtio_block_discard_t *>(b.addr);
00616
00617         size_t items = b.len / sizeof(payload[0]);
00618         if (items * sizeof(payload[0]) != b.len)
00619             return -L4_EIO;
00620
00621         if (seg + items > max_seg)
00622             return -L4_EIO;
00623         seg += items;
00624
00625         for (auto i = 0u; i < items; i++)
00626         {
00627             auto p = cxx::access_once<l4virtio_block_discard_t>(&payload[i]);
00628
00629             // Check sector size alignment. Discard sector alignment is not
00630             // strictly enforced as it is merely a hint to the driver.
00631             if (p.sector % sps != 0)
00632                 return -L4_EIO;
00633             if (p.num_sectors % sps != 0)
00634                 return -L4_EIO;
00635
00636             // Convert to the device sector size
00637             p.sector /= sps;
00638             p.num_sectors /= sps;
00639
00640             // Check bounds
00641             if (p.num_sectors > sectors)
00642                 return -L4_EIO;
00643             if (p.sector > sectors - p.num_sectors)
00644                 return -L4_EIO;
00645
00646             if (p.flags & L4VIRTIO_BLOCK_DISCARD_F_RESERVED)
00647                 return -L4_ENOSYS;
00648
00649             Inout_block *blk;
00650             if (last_blk)
00651             {
00652                 last_blk->next = cxx::make_unique<Inout_block>();
00653                 blk = last_blk->next.get();
00654             }
00655             else
00656                 blk = &preq->blocks;
00657
00658             blk->sector = p.sector;
00659             blk->num_sectors = p.num_sectors;
00660
00661             if (discard)
00662             {
00663                 if (p.flags & L4VIRTIO_BLOCK_DISCARD_F_UNMAP)
00664                     return -L4_ENOSYS;
00665                 if (p.num_sectors > _di.max_discard_sectors)
00666                     return -L4_EIO;
00667             }
00668             else
00669             {
00670                 if (p.flags & L4VIRTIO_BLOCK_DISCARD_F_UNMAP
00671                     && _di.write_zeroes_may_unmap)
00672                     blk->flags = Inout_f_unmap;
00673                 if (p.num_sectors > _di.max_write_zeroes_sectors)
00674                     return -L4_EIO;
00675             }
00676
00677             last_blk = blk;
00678         }
00679     }
00680
00681     return L4_EOK;
00682 }
00683
00684 template <typename T = Device_type>
00685 int discard_cmd_request(Pending_cmd_request *, long)
00686 { return -L4_EIO; }
00687
00688 template <typename T = Device_type>

```



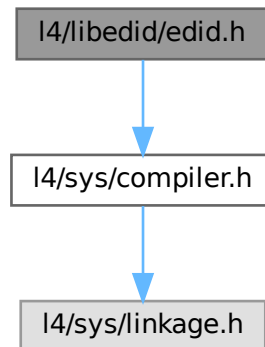
```

00689 auto discard_cmd_request(Pending_cmd_request *preq, int)
00690     -> decltype(((T*)0)->discard_info(), int())
00691 {
00692     auto *req = preq->request.get();
00693     bool discard = (req->header().type == L4VIRTIO_BLOCK_T_DISCARD);
00694
00695     int res = _device->discard(
00696         0, preq->blocks,
00697         [this, preq](int error, l4_size_t sz) { task_finished(preq, error, sz); },
00698         discard);
00699
00700     // request successfully submitted to device
00701     if (res >= 0)
00702         _in_flight++;
00703
00704     return res;
00705 }
00706
00707 // only use on errors that are not busy
00708 void handle_request_error(int error, Generic_pending_request *pending)
00709 {
00710     auto trace = Dbg::trace("virtio");
00711
00712     if (error == -L4_ENOSYS)
00713     {
00714         trace.printf("Unsupported operation.\n");
00715         finalize_request(cxx::move(pending->request), 0,
00716             L4VIRTIO_BLOCK_S_UNSUPP);
00717     }
00718     else
00719     {
00720         trace.printf("Got IO error: %d\n", error);
00721         finalize_request(cxx::move(pending->request), 0, L4VIRTIO_BLOCK_S_IOERR);
00722     }
00723 }
00724
00725 protected:
00726     L4::Cap<L4::Irq> _device_notify_irq;
00727     std::function<void(bool)> _client_invalidate_cb;
00728     std::function<void()> _client_idle_cb;
00729     unsigned _numds;
00730     Shutdown_type _shutdown_state;
00731     cxx::Ref_ptr<Device_type> _device;
00732     Device_discard_feature::Discard_info _di;
00733
00734     L4virtio::Svr::Block_features _negotiated_features;
00735
00736     unsigned _in_flight;
00737 };
00738
00739 } //name space

```

## 17.281 I4/libedid/edid.h File Reference

```
#include <l4/sys/compiler.h>
Include dependency graph for edid.h:
```



### Enumerations

- enum [Libedid\\_consts](#) { [Libedid\\_block\\_size](#) = 128 }  
*EDID constants.*

### Functions

- int [libedid\\_check\\_header](#) (const unsigned char \*edid)  
*Check for valid EDID header.*
- int [libedid\\_checksum](#) (const unsigned char \*edid)  
*Calculates the EDID checksum.*
- unsigned [libedid\\_version](#) (const unsigned char \*edid)  
*Returns the EDID version number.*
- unsigned [libedid\\_revision](#) (const unsigned char \*edid)  
*Returns the EDID revision number.*
- void [libedid\\_pnp\\_id](#) (const unsigned char \*edid, unsigned char \*id)  
*Extracts the display's PnP ID.*
- void [libedid\\_prefered\\_resolution](#) (const unsigned char \*edid, unsigned \*w, unsigned \*h)  
*Extract the display's preferred mode.*
- unsigned [libedid\\_num\\_ext\\_blocks](#) (const unsigned char \*edid)  
*Get the number of EDID extension blocks.*
- unsigned [libedid\\_dump\\_standard\\_timings](#) (const unsigned char \*edid)  
*Dump the standard timings to stdout.*
- void [libedid\\_dump](#) (const unsigned char \*edid)  
*Dump raw EDID data to stdout.*

## 17.282 edid.h

[Go to the documentation of this file.](#)

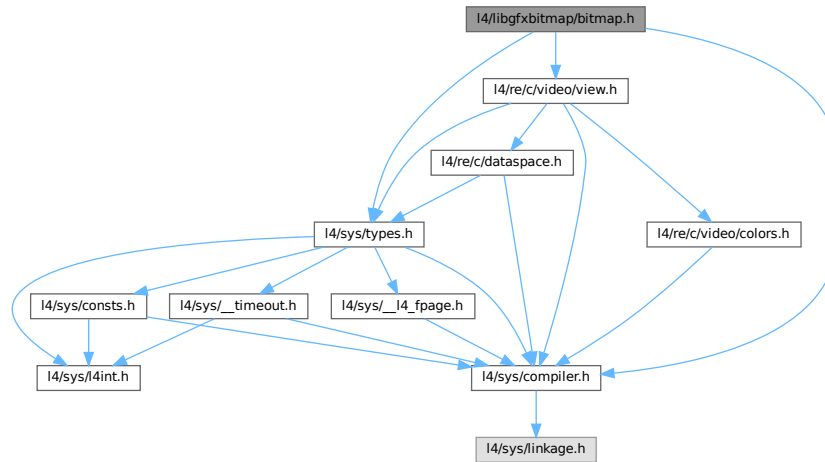
```
00001
00004 /*
00005  * (c) 2014 Matthias Lange <matthias.lange@kernkonzept.com>
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU Lesser General Public License 2.1.
00009  * Please see the COPYING-LGPL-2.1 file for details.
00010  */
00011 #pragma once
00012
00013 #include <l4/sys/compiler.h>
00014
00019
00023 enum Libedid_consts
00024 {
00025     Libedid_block_size = 128,
00026 };
00027
00028 __BEGIN_DECLS
00029
00037 int libedid_check_header(const unsigned char *edid);
00038
00046 int libedid_checksum(const unsigned char *edid);
00047
00055 unsigned libedid_version(const unsigned char *edid);
00056
00064 unsigned libedid_revision(const unsigned char *edid);
00065
00072 void libedid_pnp_id(const unsigned char *edid, unsigned char *id);
00073
00081 void libedid_prefered_resolution(const unsigned char *edid,
00082                                 unsigned *w, unsigned *h);
00083
00091 unsigned libedid_num_ext_blocks(const unsigned char *edid);
00092
00100 unsigned libedid_dump_standard_timings(const unsigned char *edid);
00101
00107 void libedid_dump(const unsigned char *edid);
00108
00110
00111 __END_DECLS
```

## 17.283 l4/libgfxbitmap/bitmap.h File Reference

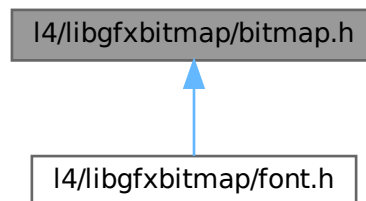
Bitmap renderer header file.

```
#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <l4/re/c/video/view.h>
```

Include dependency graph for `bitmap.h`:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `gfxbitmap_offset`  
offsets in `pmap[]` and `bmap[]`

## Param macros for `bmap_*`

Bitmap type - start least or start most significant bit

- `#define pSLIM_BMAP_START_MSB 0x02`  
*'pbm'-style: "The bits are stored eight per byte, high bit first low bit last." \*/ #define pSLIM\_BMAP\_START\_LSB /\* !< the other way round*
- typedef unsigned int `gfxbitmap_color_t`  
Standard color type.
- typedef unsigned int `gfxbitmap_color_pix_t`

*Specific color type.*

- [gfxbitmap\\_color\\_pix\\_t](#) [gfxbitmap\\_convert\\_color](#) ([l4re\\_video\\_view\\_info\\_t](#) \*vi, [gfxbitmap\\_color\\_t](#) rgb)

*Convert a color.*

- void [gfxbitmap\\_fill](#) ([l4\\_uint8\\_t](#) \*vfb, [l4re\\_video\\_view\\_info\\_t](#) \*vi, int x, int y, int w, int h, [gfxbitmap\\_color\\_pix\\_t](#) color)

*Fill a rectangular area with a color.*

- void [gfxbitmap\\_bmap](#) ([l4\\_uint8\\_t](#) \*vfb, [l4re\\_video\\_view\\_info\\_t](#) \*vi, [l4\\_int16\\_t](#) x, [l4\\_int16\\_t](#) y, [l4\\_uint32\\_t](#) w, [l4\\_uint32\\_t](#) h, [l4\\_uint8\\_t](#) \*bmap, [gfxbitmap\\_color\\_pix\\_t](#) fgc, [gfxbitmap\\_color\\_pix\\_t](#) bgc, struct [gfxbitmap\\_offset](#) \*offset, [l4\\_uint8\\_t](#) mode)

*Fill a rectangular area with a bicolor bitmap pattern.*

- void [gfxbitmap\\_set](#) ([l4\\_uint8\\_t](#) \*vfb, [l4re\\_video\\_view\\_info\\_t](#) \*vi, [l4\\_int16\\_t](#) x, [l4\\_int16\\_t](#) y, [l4\\_uint32\\_t](#) w, [l4\\_uint32\\_t](#) h, [l4\\_uint32\\_t](#) xoffs, [l4\\_uint32\\_t](#) yoffs, [l4\\_uint8\\_t](#) \*pmap, struct [gfxbitmap\\_offset](#) \*offset, [l4\\_uint32\\_t](#) pwidth)

*Set area from source area.*

- void [gfxbitmap\\_copy](#) ([l4\\_uint8\\_t](#) \*dest, [l4\\_uint8\\_t](#) \*src, [l4re\\_video\\_view\\_info\\_t](#) \*vi, int x, int y, int w, int h, int dx, int dy)

*Copy a rectangular area.*

## 17.283.1 Detailed Description

Bitmap renderer header file.

Definition in file [bitmap.h](#).

## 17.283.2 Typedef Documentation

### 17.283.2.1 [gfxbitmap\\_color\\_pix\\_t](#)

```
typedef unsigned int gfxbitmap\_color\_pix\_t
```

Specific color type.

This color type is specific for a particular framebuffer, it can be use to write pixel on a framebuffer. Use [gfxbitmap\\_convert\\_color](#) to convert from [gfxbitmap\\_color\\_t](#) to [gfxbitmap\\_color\\_pix\\_t](#).

Definition at line 65 of file [bitmap.h](#).

### 17.283.2.2 [gfxbitmap\\_color\\_t](#)

```
typedef unsigned int gfxbitmap\_color\_t
```

Standard color type.

It's a RGB type with 8bits for each channel, regardless of the framebuffer used.

Definition at line 56 of file [bitmap.h](#).

## 17.283.3 Function Documentation

### 17.283.3.1 gfxbitmap\_bmap()

```
void gfxbitmap_bmap (
    l4_uint8_t * vfb,
    l4re_video_view_info_t * vi,
    l4_int16_t x,
    l4_int16_t y,
    l4_uint32_t w,
    l4_uint32_t h,
    l4_uint8_t * bmap,
    gfxbitmap_color_pix_t fg,
    gfxbitmap_color_pix_t bg,
    struct gfxbitmap_offset * offset,
    l4_uint8_t mode)
```

Fill a rectangular area with a bicolor bitmap pattern.

#### Parameters

<i>vfb</i>	Frame buffer.
<i>vi</i>	Frame buffer information structure.
<i>x</i>	X position of area.
<i>y</i>	Y position of area.
<i>w</i>	Width of area.
<i>h</i>	Height of area.
<i>bmap</i>	Bitmap pattern.
<i>fg</i>	Foreground color.
<i>bg</i>	Background color.
<i>offset</i>	Offsets.
<i>mode</i>	Mode

See also

[pSLIM\\_BMAP\\_START\\_MSB](#) and [#pSLIM\\_BMAP\\_START\\_LSB](#).

### 17.283.3.2 gfxbitmap\_convert\_color()

```
gfxbitmap_color_pix_t gfxbitmap_convert_color (
    l4re_video_view_info_t * vi,
    gfxbitmap_color_t rgb)
```

Convert a color.

Converts a given color in standard format to the format used in the framebuffer.

### 17.283.3.3 gfxbitmap\_copy()

```
void gfxbitmap_copy (
    l4_uint8_t * dest,
    l4_uint8_t * src,
    l4re_video_view_info_t * vi,
    int x,
    int y,
    int w,
    int h,
    int dx,
    int dy)
```

Copy a rectangular area.

#### Parameters

<i>dest</i>	Destination frame buffer.
<i>src</i>	Source frame buffer.
<i>vi</i>	Frame buffer information structure.
<i>x</i>	Source X position of area.
<i>y</i>	Source Y position of area.
<i>w</i>	Width of area.
<i>h</i>	Height of area.
<i>dx</i>	Source X position of area.
<i>dy</i>	Source Y position of area.

References [L4\\_END\\_DECLS](#).

### 17.283.3.4 gfxbitmap\_fill()

```
void gfxbitmap_fill (
    l4_uint8_t * vfb,
    l4re_video_view_info_t * vi,
    int x,
    int y,
    int w,
    int h,
    gfxbitmap_color_pix_t color)
```

Fill a rectangular area with a color.

#### Parameters

<i>vfb</i>	Frame buffer.
<i>vi</i>	Frame buffer information structure.
<i>x</i>	X position of area.
<i>y</i>	Y position of area.

<i>w</i>	Width of area.
<i>h</i>	Height of area.
<i>color</i>	Color of area.

### 17.283.3.5 gfxbitmap\_set()

```
void gfxbitmap_set (
    14_uint8_t * vfb,
    14re_video_view_info_t * vi,
    14_int16_t x,
    14_int16_t y,
    14_uint32_t w,
    14_uint32_t h,
    14_uint32_t xoffs,
    14_uint32_t yoffs,
    14_uint8_t * pmap,
    struct gfxbitmap_offset * offset,
    14_uint32_t pwidth)
```

Set area from source area.

#### Parameters

<i>vfb</i>	Frame buffer.
<i>vi</i>	Frame buffer information structure.
<i>x</i>	X position of area.
<i>y</i>	Y position of area.
<i>w</i>	Width of area.
<i>h</i>	Height of area.
<i>pmap</i>	Source.
<i>xoffs</i>	X offset.
<i>yoffs</i>	Y offset.
<i>offset</i>	Offsets.
<i>pwidth</i>	Width of source in bytes.

## 17.284 bitmap.h

[Go to the documentation of this file.](#)

```
00001
00005 /*
00006  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU Lesser General Public License 2.1.
00010  * Please see the COPYING-LGPL-2.1 file for details.
00011  */
00012 #pragma once
00013
00014 #include <14/sys/compiler.h>
00015 #include <14/sys/types.h>
00016 #include <14/re/c/video/view.h>
00017
```



```

00027
00032
00033 L4_BEGIN_DECLS
00039 #define pSLIM_BMAP_START_MSB    0x02
00042 #define pSLIM_BMAP_START_LSB    0x01
00044
00049
00056 typedef unsigned int  gfxbitmap_color_t;
00057
00065 typedef unsigned int  gfxbitmap_color_pix_t;
00066
00068 struct gfxbitmap_offset
00069 {
00070     l4_uint32_t preskip_x;
00071     l4_uint32_t preskip_y;
00072     l4_uint32_t endskip_x;
00073 };
00074
00081 gfxbitmap_color_pix_t
00082 gfxbitmap_convert_color(l4re_video_view_info_t *vi, gfxbitmap_color_t rgb);
00083
00095 void
00096 gfxbitmap_fill(l4_uint8_t *vfb, l4re_video_view_info_t *vi,
00097               int x, int y, int w, int h, gfxbitmap_color_pix_t color);
00098
00116 void
00117 gfxbitmap_bmap(l4_uint8_t *vfb, l4re_video_view_info_t *vi,
00118               l4_int16_t x, l4_int16_t y, l4_uint32_t w,
00119               l4_uint32_t h, l4_uint8_t *bmap,
00120               gfxbitmap_color_pix_t fg, gfxbitmap_color_pix_t bg,
00121               struct gfxbitmap_offset *offset, l4_uint8_t mode);
00122
00138 void
00139 gfxbitmap_set(l4_uint8_t *vfb, l4re_video_view_info_t *vi,
00140               l4_int16_t x, l4_int16_t y, l4_uint32_t w,
00141               l4_uint32_t h, l4_uint32_t xoffs, l4_uint32_t yoffs,
00142               l4_uint8_t *pmap, struct gfxbitmap_offset *offset,
00143               l4_uint32_t pwidth);
00144
00158 void
00159 gfxbitmap_copy(l4_uint8_t *dest, l4_uint8_t *src, l4re_video_view_info_t *vi,
00160               int x, int y, int w, int h, int dx, int dy);
00162
00163 L4_END_DECLS

```

## 17.285 l4/libgfxbitmap/font.h File Reference

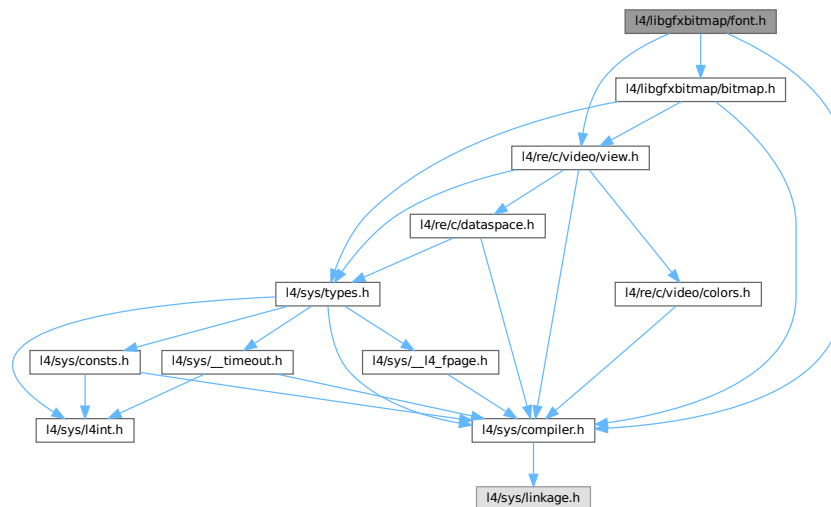
Bitmap font renderer header file.

```

#include <l4/sys/compiler.h>
#include <l4/re/c/video/view.h>
#include <l4/libgfxbitmap/bitmap.h>

```

Include dependency graph for font.h:



## Macros

- `#define GFXBITMAP_DEFAULT_FONT (void *)0`  
Constant to use for the default font.

## Enumerations

- `enum`  
Constant for length field.

## Functions

- `int gfxbitmap_font_init (void)`  
Initialize the library.
- `gfxbitmap_font_t gfxbitmap_font_get (const char *name)`  
Get a font descriptor.
- `unsigned gfxbitmap_font_width (gfxbitmap_font_t font)`  
Get the font width.
- `unsigned gfxbitmap_font_height (gfxbitmap_font_t font)`  
Get the font height.
- `void * gfxbitmap_font_data (gfxbitmap_font_t font, unsigned c)`  
Get bitmap font data for a specific character.
- `void gfxbitmap_font_text (void *fb, l4re_video_view_info_t *vi, gfxbitmap_font_t font, const char *text, unsigned len, unsigned x, unsigned y, gfxbitmap_color_pix_t fg, gfxbitmap_color_pix_t bg)`  
Render a string to a framebuffer.
- `void gfxbitmap_font_text_scale (void *fb, l4re_video_view_info_t *vi, gfxbitmap_font_t font, const char *text, unsigned len, unsigned x, unsigned y, gfxbitmap_color_pix_t fg, gfxbitmap_color_pix_t bg, int scale_x, int scale_y)`  
Render a string to a framebuffer, including scaling.

## Variables

- [L4\\_BEGIN\\_DECLS](#) typedef void \* **gfxbitmap\_font\_t**  
*Font.*

## 17.285.1 Detailed Description

Bitmap font renderer header file.

Definition in file [font.h](#).

## 17.285.2 Enumeration Type Documentation

### 17.285.2.1 anonymous enum

`anonymous enum`

Constant for length field.

Use this if the function should call strlen on the text argument itself.

Definition at line 38 of file [font.h](#).

## 17.285.3 Function Documentation

### 17.285.3.1 gfxbitmap\_font\_data()

```
void * gfxbitmap_font_data (  
    gfxbitmap\_font\_t font,  
    unsigned c)
```

Get bitmap font data for a specific character.

#### Parameters

<i>font</i>	Font.
<i>c</i>	Character.

#### Returns

Pointer to bmap data, NULL on error.

References [gfxbitmap\\_font\\_t](#), and [L4\\_CV](#).

### 17.285.3.2 gfxbitmap\_font\_get()

```
gfxbitmap\_font\_t gfxbitmap_font_get (  
    const char * name)
```

Get a font descriptor.

#### Parameters

---

<i>name</i>	Name of the font.
-------------	-------------------

**Returns**

A (opaque) font descriptor, or NULL if font could not be found.

References [gfxbitmap\\_font\\_t](#), and [L4\\_CV](#).

**17.285.3.3 gfxbitmap\_font\_height()**

```
unsigned gfxbitmap_font_height (
    gfxbitmap_font_t font)
```

Get the font height.

**Parameters**

<i>font</i>	Font.
-------------	-------

**Returns**

Font height, 0 if font height could not be retrieved.

References [gfxbitmap\\_font\\_t](#), and [L4\\_CV](#).

**17.285.3.4 gfxbitmap\_font\_init()**

```
int gfxbitmap_font_init (
    void )
```

Initialize the library.

This function must be called before any other font function of this library.

**Returns**

0 on success, other on error

References [L4\\_CV](#).

**17.285.3.5 gfxbitmap\_font\_text()**

```
void gfxbitmap_font_text (
    void * fb,
    l4re_video_view_info_t * vi,
    gfxbitmap_font_t font,
    const char * text,
    unsigned len,
    unsigned x,
    unsigned y,
    gfxbitmap_color_pix_t fg,
    gfxbitmap_color_pix_t bg)
```

Render a string to a framebuffer.

**Parameters**

<i>fb</i>	Pointer to frame buffer.
<i>vi</i>	Frame buffer info structure.
<i>font</i>	Font.
<i>text</i>	Text string.
<i>len</i>	Length of the text string.
<i>x</i>	Horizontal position in the frame buffer.
<i>y</i>	Vertical position in the frame buffer.
<i>fg</i>	Foreground color.
<i>bg</i>	Background color.

References [gfxbitmap\\_font\\_t](#), and [L4\\_CV](#).

### 17.285.3.6 gfxbitmap\_font\_text\_scale()

```
void gfxbitmap_font_text_scale (
    void * fb,
    l4re_video_view_info_t * vi,
    gfxbitmap_font_t font,
    const char * text,
    unsigned len,
    unsigned x,
    unsigned y,
    gfxbitmap_color_pix_t fg,
    gfxbitmap_color_pix_t bg,
    int scale_x,
    int scale_y)
```

Render a string to a framebuffer, including scaling.

#### Parameters

<i>fb</i>	Pointer to frame buffer.
<i>vi</i>	Frame buffer info structure.
<i>font</i>	Font.
<i>text</i>	Text string.
<i>len</i>	Length of the text string.
<i>x</i>	Horizontal position in the frame buffer.
<i>y</i>	Vertical position in the frame buffer.
<i>fg</i>	Foreground color.
<i>bg</i>	Background color.
<i>scale_x</i>	Horizonal scale factor.
<i>scale_y</i>	Vertical scale factor.

References [gfxbitmap\\_font\\_t](#), and [L4\\_END\\_DECLS](#).

### 17.285.3.7 gfxbitmap\_font\_width()

```
unsigned gfxbitmap_font_width (
    gfxbitmap_font_t font)
```

Get the font width.

#### Parameters

<i>font</i>	Font.
-------------	-------

#### Returns

Font width, 0 if font width could not be retrieved.

References [gfxbitmap\\_font\\_t](#), and [L4\\_CV](#).

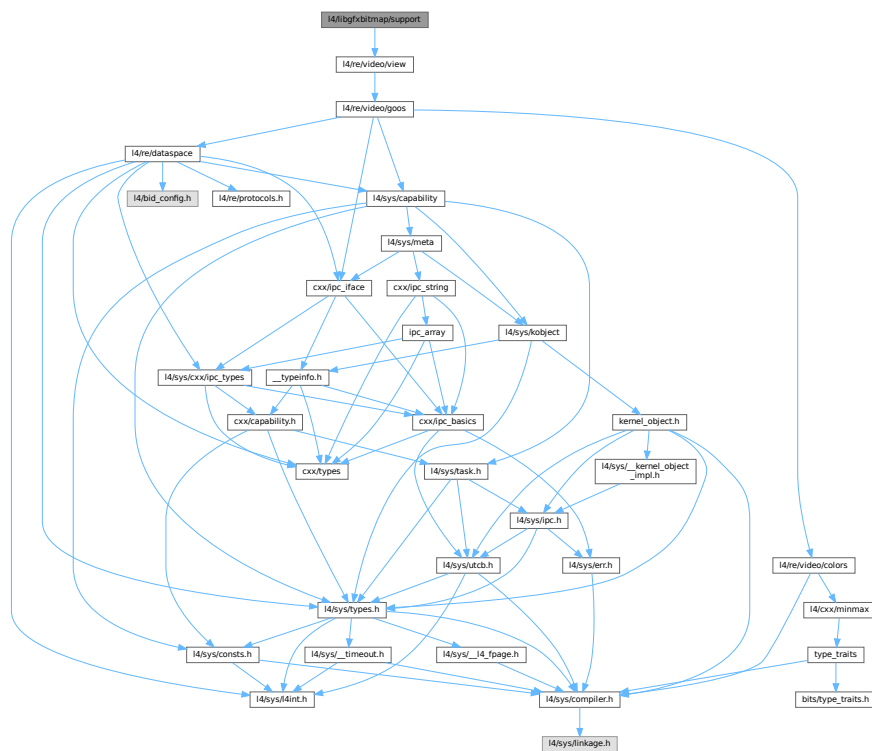
## 17.286 font.h

[Go to the documentation of this file.](#)

```
00001
00005 /*
00006  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * This file is part of TUD:OS and distributed under the terms of the
00009  * GNU Lesser General Public License 2.1.
00010  * Please see the COPYING-LGPL-2.1 file for details.
00011  */
00012 #pragma once
00013
00014 #include <l4/sys/compiler.h>
00015 #include <l4/re/c/video/view.h>
00016 #include <l4/libgfxbitmap/bitmap.h>
00017
00022
00027
00031 #define GFXBITMAP_DEFAULT_FONT (void *)0
00032
00038 enum { GFXBITMAP_USE_STRLEN = ~0U };
00039
00040 L4_BEGIN_DECLS
00041
00043 typedef void *gfxbitmap_font_t;
00044
00053 L4_CV int gfxbitmap_font_init(void);
00054
00062 L4_CV gfxbitmap_font_t gfxbitmap_font_get(const char *name);
00063
00070 L4_CV unsigned
00071 gfxbitmap_font_width(gfxbitmap_font_t font);
00072
00079 L4_CV unsigned
00080 gfxbitmap_font_height(gfxbitmap_font_t font);
00081
00089 L4_CV void *
00090 gfxbitmap_font_data(gfxbitmap_font_t font, unsigned c);
00091
00105 L4_CV void
00106 gfxbitmap_font_text(void *fb, l4re_video_view_info_t *vi,
00107                    gfxbitmap_font_t font, const char *text, unsigned len,
00108                    unsigned x, unsigned y,
00109                    gfxbitmap_color_pix_t fg, gfxbitmap_color_pix_t bg);
00110
00126 L4_CV void
00127 gfxbitmap_font_text_scale(void *fb, l4re_video_view_info_t *vi,
00128                          gfxbitmap_font_t font, const char *text, unsigned len,
00129                          unsigned x, unsigned y,
00130                          gfxbitmap_color_pix_t fg, gfxbitmap_color_pix_t bg,
00131                          int scale_x, int scale_y);
00132 L4_END_DECLS
```

Terminal support functionality.

Include dependency graph for support:



Terminal support functionality.

2009

Adam Lackorzynski [adam@os.inf.tu-dresden.de](mailto:adam@os.inf.tu-dresden.de)

Definition in file [support](#).

## 17.288 support

[Go to the documentation of this file.](#)

```

00001 /* vim:set ft=cpp: */
00009 /*
00010  * (c) 2009 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  * This file is part of TUD:OS and distributed under the terms of the
00013  * GNU Lesser General Public License 2.1.
00014  * Please see the COPYING-LGPL-2.1 file for details.
00015  */
00016 #ifndef __LIBTERM_SUPPORT_H__
00017 #define __LIBTERM_SUPPORT_H__
00018
00019 #include <l4/re/video/view>
00020
00021 void
00022 libterm_init_colors(L4Re::Video::View::Info *fbi);
00023
00024 int
00025 libterm_get_color(int mode, int color);
00026
00027 #endif

```

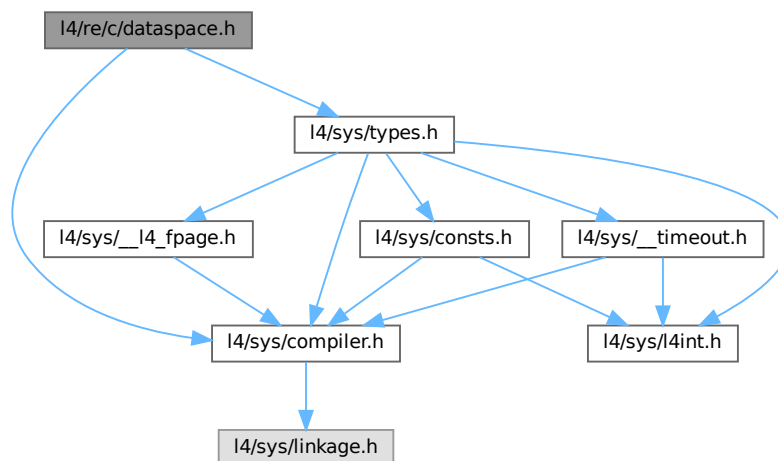
## 17.289 l4/re/c/dataspace.h File Reference

Data space C interface.

```
#include <l4/sys/compiler.h>
```

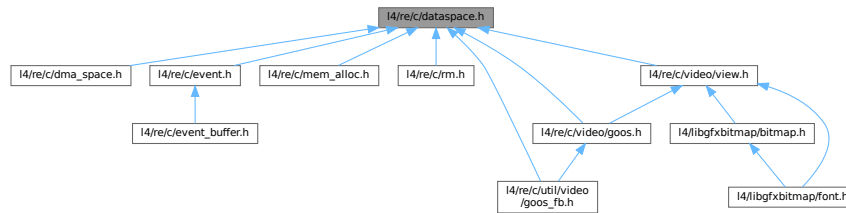
```
#include <l4/sys/types.h>
```

Include dependency graph for dataspace.h:





This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [l4re\\_ds\\_stats\\_t](#)  
*Information about the data space.*

## Enumerations

- enum [l4re\\_ds\\_map\\_flags](#) { }  
*Flags to specify the memory mapping type of a request.*

## Functions

- long [l4re\\_ds\\_clear](#) ([l4re\\_ds\\_t](#) ds, [l4re\\_ds\\_offset\\_t](#) offset, [l4re\\_ds\\_size\\_t](#) size) [L4\\_NOTHROW](#)  
*Clear parts of a dataspace.*
- long [l4re\\_ds\\_allocate](#) ([l4re\\_ds\\_t](#) ds, [l4re\\_ds\\_offset\\_t](#) offset, [l4re\\_ds\\_size\\_t](#) size) [L4\\_NOTHROW](#)  
*Allocate a range in the dataspace.*
- int [l4re\\_ds\\_copy\\_in](#) ([l4re\\_ds\\_t](#) ds, [l4re\\_ds\\_offset\\_t](#) dst\_offs, [l4re\\_ds\\_t](#) src, [l4re\\_ds\\_offset\\_t](#) src\_offs, [l4re\\_ds\\_size\\_t](#) size) [L4\\_NOTHROW](#)  
*Copy contents from another dataspace.*
- [l4re\\_ds\\_size\\_t](#) [l4re\\_ds\\_size](#) ([l4re\\_ds\\_t](#) ds) [L4\\_NOTHROW](#)  
*Get size of a dataspace.*
- [l4re\\_ds\\_flags\\_t](#) [l4re\\_ds\\_flags](#) ([l4re\\_ds\\_t](#) ds) [L4\\_NOTHROW](#)  
*Get flags of the dataspace.*
- int [l4re\\_ds\\_info](#) ([l4re\\_ds\\_t](#) ds, [l4re\\_ds\\_stats\\_t](#) \*stats) [L4\\_NOTHROW](#)  
*Get information on the dataspace.*
- int [l4re\\_ds\\_map\\_info](#) ([l4re\\_ds\\_t](#) ds, [l4\\_addr\\_t](#) \*start\_addr, [l4\\_addr\\_t](#) \*end\_addr) [L4\\_NOTHROW](#)  
*Get mapping range of dataspace.*

## Variables

- [L4\\_BEGIN\\_DECLS](#) typedef [l4\\_cap\\_idx\\_t](#) [l4re\\_ds\\_t](#)  
*Dataspace type.*

## 17.289.1 Detailed Description

Data space C interface.

Definition in file [dataspace.h](#).

## 17.290 dataspace.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00019
00020 #include <l4/sys/compiler.h>
00021 #include <l4/sys/types.h>
00022
00023 L4_BEGIN_DECLS
00024
00029 typedef l4_cap_idx_t l4re_ds_t;
00030 typedef l4_uint64_t l4re_ds_size_t;
00031 typedef l4_uint64_t l4re_ds_offset_t;
00032 typedef l4_uint64_t l4re_ds_map_addr_t;
00033 typedef unsigned long l4re_ds_flags_t;
00034
00039 typedef struct {
00040     l4re_ds_size_t size;
00041     l4re_ds_flags_t flags;
00042 } l4re_ds_stats_t;
00043
00048 enum l4re_ds_map_flags {
00049     L4RE_DS_F_R    = L4_FPAGE_RO,
00050     L4RE_DS_F_W    = L4_FPAGE_W,
00051     L4RE_DS_F_X    = L4_FPAGE_X,
00052     L4RE_DS_F_RW   = L4_FPAGE_RW,
00053     L4RE_DS_F_RX   = L4_FPAGE_RX,
00054     L4RE_DS_F_RWX  = L4_FPAGE_RWX,
00055
00056     L4RE_DS_F_RIGHTS_MASK = 0x0f,
00057
00058     L4RE_DS_F_NORMAL      = 0x00,
00059     L4RE_DS_F_CACHEABLE  = L4RE_DS_F_NORMAL,
00060     L4RE_DS_F_BUFFERABLE = 0x10,
00061     L4RE_DS_F_UNCACHEABLE = 0x20,
00062     L4RE_DS_F_CACHING_MASK = 0x30,
00063     L4RE_DS_F_CACHING_SHIFT = 4,
00064 };
00065
00071 L4_CV int
00072 l4re_ds_map(l4re_ds_t ds,
00073             l4re_ds_offset_t offset,
00074             l4re_ds_flags_t flags,
00075             l4re_ds_map_addr_t local_addr,
00076             l4re_ds_map_addr_t min_addr,
00077             l4re_ds_map_addr_t max_addr) L4_NOTHROW;
00078
00084 L4_CV int
00085 l4re_ds_map_region(l4re_ds_t ds,
00086                   l4re_ds_offset_t offset,
00087                   l4re_ds_flags_t flags,
00088                   l4re_ds_map_addr_t min_addr,
00089                   l4re_ds_map_addr_t max_addr) L4_NOTHROW;
00090
00097 L4_CV long
00098 l4re_ds_clear(l4re_ds_t ds, l4re_ds_offset_t offset,
00099              l4re_ds_size_t size) L4_NOTHROW;
00100
00107 L4_CV long
00108 l4re_ds_allocate(l4re_ds_t ds,
00109                 l4re_ds_offset_t offset,
00110                 l4re_ds_size_t size) L4_NOTHROW;
00111
00118 L4_CV int
00119 l4re_ds_copy_in(l4re_ds_t ds, l4re_ds_offset_t dst_offs,
00120                l4re_ds_t src, l4re_ds_offset_t src_offs,
00121                l4re_ds_size_t size) L4_NOTHROW;
00122
00129 L4_CV l4re_ds_size_t
00130 l4re_ds_size(l4re_ds_t ds) L4_NOTHROW;
00131
00138 L4_CV l4re_ds_flags_t
00139 l4re_ds_flags(l4re_ds_t ds) L4_NOTHROW;
00140
00147 L4_CV int
00148 l4re_ds_info(l4re_ds_t ds, l4re_ds_stats_t *stats) L4_NOTHROW;

```

```

00149
00156 L4_CV int
00157 l4re_ds_map_info(l4re_ds_t ds,
00158                  l4_addr_t *start_addr, l4_addr_t *end_addr) L4_NOTHROW;
00159
00160 L4_END_DECLS

```

## 17.291 l4/re/c/dma\_space.h File Reference

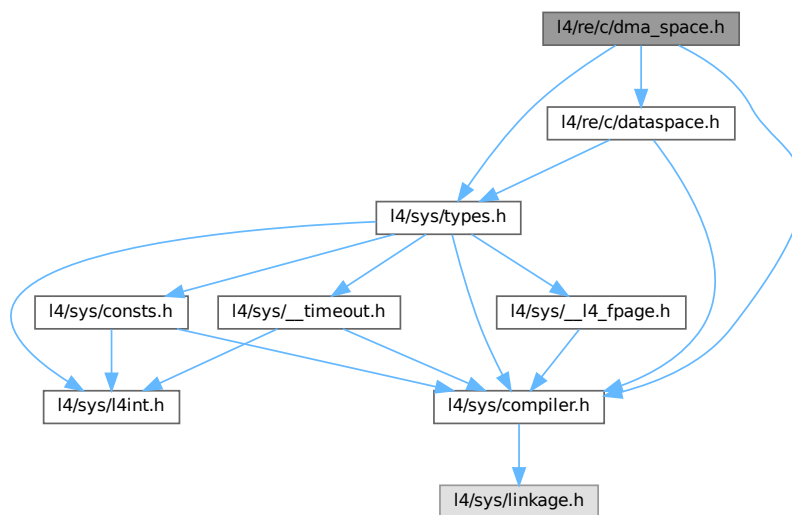
DMA space C interface.

```

#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <l4/re/c/dataspace.h>

```

Include dependency graph for dma\_space.h:



### Typedefs

- typedef `l4_cap_idx_t l4re_dma_space_t`  
DMA space capability type.
- typedef `l4_uint64_t l4re_dma_space_dma_addr_t`  
Data type for DMA addresses.

### Enumerations

- enum `l4re_dma_space_direction` { `L4RE_DMA_SPACE_BIDIRECTIONAL`, `L4RE_DMA_SPACE_TO_DEVICE`, `L4RE_DMA_SPACE_FROM_DEVICE`, `L4RE_DMA_SPACE_NONE` }  
Direction of the DMA transfers.
- enum `l4re_dma_space_space_attris` { `L4RE_DMA_SPACE_COHERENT` = 1 << 0, `L4RE_DMA_SPACE_PHYS_SPACE` = 1 << 1 }  
Attributes assigned to the DMA space when associated with a specific device.

## Functions

- long [l4re\\_dma\\_space\\_map](#) ([l4re\\_dma\\_space\\_t](#) dma, [l4re\\_ds\\_t](#) src, [l4re\\_ds\\_offset\\_t](#) offset, [l4\\_size\\_t](#) \*size, unsigned long attrs, enum [l4re\\_dma\\_space\\_direction](#) dir, [l4re\\_dma\\_space\\_dma\\_addr\\_t](#) \*dma\_addr) [L4\\_NOTHROW](#)  
*Map the given part of this data space into the DMA address space.*
- long [l4re\\_dma\\_space\\_unmap](#) ([l4re\\_dma\\_space\\_t](#) dma, [l4re\\_dma\\_space\\_dma\\_addr\\_t](#) dma\_addr, [l4\\_size\\_t](#) size, unsigned long attrs, enum [l4re\\_dma\\_space\\_direction](#) dir) [L4\\_NOTHROW](#)  
*Unmap the given part of this data space from the DMA address space.*
- long [l4re\\_dma\\_space\\_associate](#) ([l4re\\_dma\\_space\\_t](#) dma, [l4\\_cap\\_idx\\_t](#) dma\_task, unsigned long attr) [L4\\_NOTHROW](#)  
*Associate a (kernel) [DMA space](#) for a device to this [Dma\\_space](#).*
- long [l4re\\_dma\\_space\\_disassociate](#) ([l4re\\_dma\\_space\\_t](#) dma)  
*Disassociate the (kernel) [DMA space](#) from this [Dma\\_space](#).*

### 17.291.1 Detailed Description

DMA space C interface.

Definition in file [dma\\_space.h](#).

### 17.291.2 Enumeration Type Documentation

#### 17.291.2.1 [l4re\\_dma\\_space\\_direction](#)

enum [l4re\\_dma\\_space\\_direction](#)

Direction of the DMA transfers.

#### Enumerator

<a href="#">L4RE_DMA_SPACE_BIDIRECTIONAL</a>	device reads and writes to the memory
<a href="#">L4RE_DMA_SPACE_TO_DEVICE</a>	device reads the memory
<a href="#">L4RE_DMA_SPACE_FROM_DEVICE</a>	device writes to the memory
<a href="#">L4RE_DMA_SPACE_NONE</a>	device is coherently connected

Definition at line 27 of file [dma\\_space.h](#).

#### 17.291.2.2 [l4re\\_dma\\_space\\_space\\_attrbs](#)

enum [l4re\\_dma\\_space\\_space\\_attrbs](#)

Attributes assigned to the DMA space when associated with a specific device.

See also

[Space\\_attrbs](#)

#### Enumerator

L4RE_DMA_SPACE_COHERENT	The device is connected coherently with the cache. This means that the map() and unmap() do not need to sync CPU caches before and after DMA.
L4RE_DMA_SPACE_PHYS_SPACE	The DMA space has no DMA task assigned and uses the CPUs physical memory.

Definition at line 38 of file [dma\\_space.h](#).

## 17.292 dma\_space.h

[Go to the documentation of this file.](#)

```

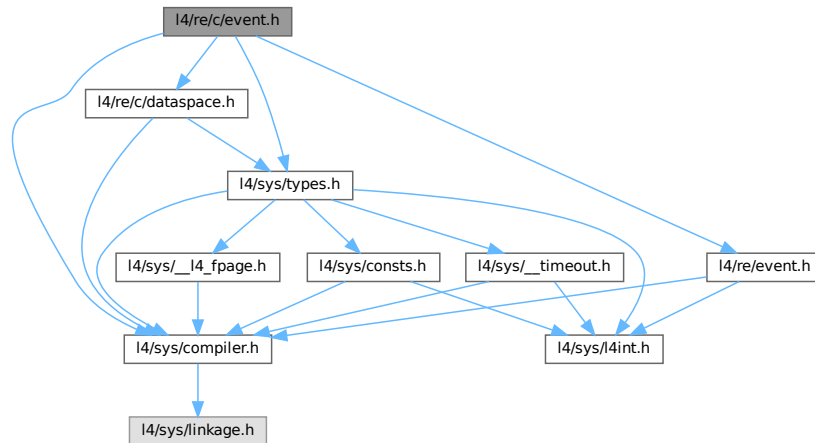
00001
00005 /*
00006  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00007  *
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010 #pragma once
00011
00017
00018 #include <l4/sys/compiler.h>
00019 #include <l4/sys/types.h>
00020 #include <l4/re/c/dataspace.h>
00021
00022 L4_BEGIN_DECLS
00023
00027 enum l4re_dma_space_direction
00028 {
00029     L4RE_DMA_SPACE_BIDIRECTIONAL,
00030     L4RE_DMA_SPACE_TO_DEVICE,
00031     L4RE_DMA_SPACE_FROM_DEVICE,
00032     L4RE_DMA_SPACE_NONE
00033 };
00034
00038 enum l4re_dma_space_space_attrbs
00039 {
00040     L4RE_DMA_SPACE_COHERENT = 1 << 0,
00041     L4RE_DMA_SPACE_PHYS_SPACE = 1 << 1,
00042 };
00043
00049 typedef l4_cap_idx_t l4re_dma_space_t;
00050
00052 typedef l4_uint64_t l4re_dma_space_dma_addr_t;
00053
00060 L4_CV long
00061 l4re_dma_space_map(l4re_dma_space_t dma, l4re_ds_t src,
00062                   l4re_ds_offset_t offset,
00063                   l4_size_t * size, unsigned long attrs,
00064                   enum l4re_dma_space_direction dir,
00065                   l4re_dma_space_dma_addr_t *dma_addr) L4_NOTHROW;
00066
00067
00074 L4_CV long
00075 l4re_dma_space_unmap(l4re_dma_space_t dma, l4re_dma_space_dma_addr_t dma_addr,
00076                     l4_size_t size, unsigned long attrs,
00077                     enum l4re_dma_space_direction dir) L4_NOTHROW;
00078
00085 L4_CV long
00086 l4re_dma_space_associate(l4re_dma_space_t dma, l4_cap_idx_t dma_task,
00087                          unsigned long attr) L4_NOTHROW;
00088
00095 L4_CV long
00096 l4re_dma_space_disassociate(l4re_dma_space_t dma);
00097
00098
00099 L4_END_DECLS

```

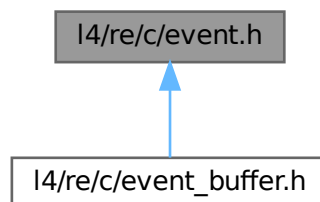
## 17.293 l4/re/c/event.h File Reference

Event C interface.

```
#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <l4/re/c/dataspace.h>
#include <l4/re/event.h>
Include dependency graph for event.h:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [l4re\\_event\\_t](#)  
*Event structure used in buffer.*

## Functions

- long [l4re\\_event\\_get\\_buffer](#) (const [l4\\_cap\\_idx\\_t](#) server, const [l4re\\_ds\\_t](#) ds) [L4\\_NOTHROW](#)  
*Get an event signal buffer.*
- long [l4re\\_event\\_get\\_num\\_streams](#) (const [l4\\_cap\\_idx\\_t](#) server) [L4\\_NOTHROW](#)  
*Get number of streams.*

- `long l4re_event_get_stream_info` (const `l4_cap_idx_t` server, int idx, `l4re_event_stream_info_t` \*info) `L4_NOTHROW`  
*Get information on a stream.*
- `long l4re_event_get_stream_info_for_id` (const `l4_cap_idx_t` server, `l4_umword_t` stream\_id, `l4re_event_stream_info_t` \*info) `L4_NOTHROW`  
*Get info for a stream given a stream id.*
- `long l4re_event_get_axis_info` (const `l4_cap_idx_t` server, `l4_umword_t` id, unsigned naxes, unsigned const \*axis, `l4re_event_absinfo_t` \*info) `L4_NOTHROW`  
*Get Axis information for a stream.*

## 17.293.1 Detailed Description

Event C interface.

Definition in file [event.h](#).

## 17.294 event.h

[Go to the documentation of this file.](#)

```

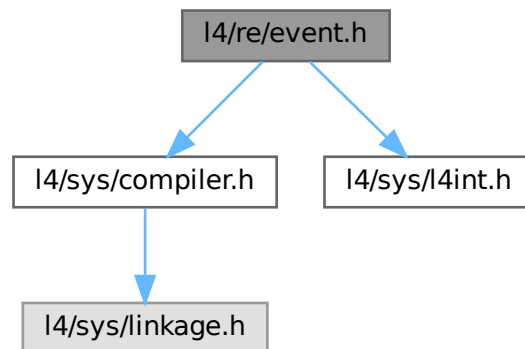
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00019
00020 #include <l4/sys/compiler.h>
00021 #include <l4/sys/types.h>
00022 #include <l4/re/c/dataspace.h>
00023 #include <l4/re/event.h>
00024
00025 L4_BEGIN_DECLS
00026
00030 typedef struct
00031 {
00032     long long time;
00033     unsigned short type;
00034     unsigned short code;
00035     int value;
00036     l4_umword_t stream_id;
00037 } l4re_event_t;
00038
00050 L4_CV long
00051 l4re_event_get_buffer(const l4_cap_idx_t server,
00052                      const l4re_ds_t ds) L4_NOTHROW;
00053
00064 L4_CV long
00065 l4re_event_get_num_streams(const l4_cap_idx_t server) L4_NOTHROW;
00066
00079 L4_CV long
00080 l4re_event_get_stream_info(const l4_cap_idx_t server,
00081                            int idx, l4re_event_stream_info_t *info) L4_NOTHROW;
00082
00095 L4_CV long
00096 l4re_event_get_stream_info_for_id(const l4_cap_idx_t server,
00097                                   l4_umword_t stream_id,
00098                                   l4re_event_stream_info_t *info) L4_NOTHROW;
00099
00115 L4_CV long
00116 l4re_event_get_axis_info(const l4_cap_idx_t server, l4_umword_t id,
00117                           unsigned naxes, unsigned const *axis,
00118                           l4re_event_absinfo_t *info) L4_NOTHROW;
00119
00120 L4_END_DECLS

```

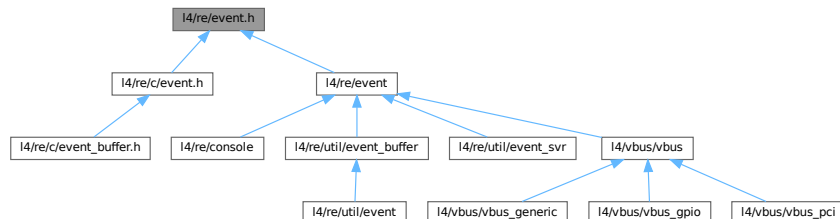
## 17.295 I4/re/event.h File Reference

Events.

```
#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>
Include dependency graph for event.h:
```



This graph shows which files directly or indirectly include this file:



### 17.295.1 Detailed Description

Events.

Definition in file [event.h](#).

## 17.296 event.h

[Go to the documentation of this file.](#)

```
00001
00005 /*
```



```

00006  * (c) 2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007  *      economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00013 #include <l4/sys/compiler.h>
00014 #include <l4/sys/l4int.h>
00015
00016 typedef struct L4_EXPORT_TYPE l4re_event_stream_id_t
00017 {
00018     l4_uint16_t bustype;
00019     l4_uint16_t vendor;
00020     l4_uint16_t product;
00021     l4_uint16_t version;
00022 } l4re_event_stream_id_t;
00023
00024 typedef struct L4_EXPORT_TYPE l4re_event_absinfo_t
00025 {
00026     l4_int32_t value;
00027     l4_int32_t min;
00028     l4_int32_t max;
00029     l4_int32_t fuzz;
00030     l4_int32_t flat;
00031     l4_int32_t resolution;
00032 } l4re_event_absinfo_t;
00033
00034 enum l4re_event_stream_max_values_t
00035 {
00036     L4RE_EVENT_EV_MAX   = 0x1f,
00037     L4RE_EVENT_KEY_MAX  = 0x1ff,
00038     L4RE_EVENT_REL_MAX  = 0xf,
00039     L4RE_EVENT_ABS_MAX  = 0x3f,
00040     L4RE_EVENT_PROP_MAX = 0x1f,
00041     L4RE_EVENT_SW_MAX   = 0xf, // should be >= L4RE_SW_MAX
00042 };
00043
00044 enum l4re_event_stream_props_t
00045 {
00046     L4RE_EVENT_STREAM_CALIBRATE = 0x001,
00047 };
00048
00049
00050 #define __UNUM_B(x) ((x+1) + sizeof(unsigned long)*8 - 1) / (sizeof(unsigned long)*8)
00051
00052 typedef struct L4_EXPORT_TYPE l4re_event_stream_info_t
00053 {
00054     l4_umword_t stream_id;
00055     char name[32];
00056     char phys[32];
00057     l4re_event_stream_id_t id;
00058
00059     unsigned long propbits[__UNUM_B(L4RE_EVENT_PROP_MAX)];
00060
00061     unsigned long evbits[__UNUM_B(L4RE_EVENT_EV_MAX)];
00062     unsigned long keybits[__UNUM_B(L4RE_EVENT_KEY_MAX)];
00063     unsigned long relbits[__UNUM_B(L4RE_EVENT_REL_MAX)];
00064     unsigned long absbits[__UNUM_B(L4RE_EVENT_ABS_MAX)];
00065     unsigned long swbits[__UNUM_B(L4RE_EVENT_SW_MAX)];
00066 } l4re_event_stream_info_t;
00067
00068
00069 typedef struct L4_EXPORT_TYPE l4re_event_stream_state_t
00070 {
00071     unsigned long keybits[__UNUM_B(L4RE_EVENT_KEY_MAX)];
00072     unsigned long swbits[__UNUM_B(L4RE_EVENT_SW_MAX)];
00073 } l4re_event_stream_state_t;
00074
00075 #undef __UNUM_B
00076

```

## 17.297 event\_buffer.h

```

00001 #pragma once
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #include <l4/sys/compiler.h>

```

```

00010 #include <l4/sys/linkage.h>
00011 #include <l4/re/c/event.h>
00012
00013 L4_BEGIN_DECLS
00014
00015 typedef struct l4re_event_buffer_consumer_t
00016 {
00017     unsigned long _obj_buf[8];
00018 } l4re_event_buffer_consumer_t;
00019
00020 L4_CV void
00021 l4re_event_free(l4re_event_t *e) L4_NOTHROW;
00022
00023 L4_CV long
00024 l4re_event_buffer_attach(l4re_event_buffer_consumer_t *evbuf,
00025                          l4re_ds_t ds, l4_cap_idx_t rm) L4_NOTHROW;
00026
00027 L4_CV long
00028 l4re_event_buffer_detach(l4re_event_buffer_consumer_t *evbuf,
00029                          l4_cap_idx_t rm) L4_NOTHROW;
00030
00031 L4_CV l4re_event_t *
00032 l4re_event_buffer_next(l4re_event_buffer_consumer_t *evbuf) L4_NOTHROW;
00033
00034 typedef L4_CV void l4re_event_buffer_cb_t(l4re_event_t *ev, void *data);
00035
00036 L4_CV void
00037 l4re_event_buffer_consumer_foreach_available_event(l4re_event_buffer_consumer_t *evbuf,
00038                                                    void *data, l4re_event_buffer_cb_t *cb);
00039
00040
00041 L4_CV void
00042 l4re_event_buffer_consumer_process(l4re_event_buffer_consumer_t *evbuf,
00043                                   l4_cap_idx_t irq, l4_cap_idx_t thread, void *data,
00044                                   l4re_event_buffer_cb_t *cb);
00045
00046 L4_END_DECLS

```

## 17.298 l4/re/c/inhibitor.h File Reference

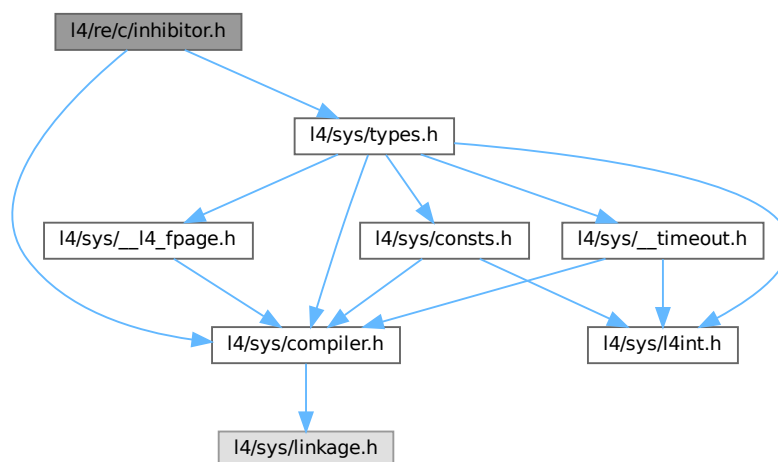
Inhibitor C interface.

```

#include <l4/sys/compiler.h>
#include <l4/sys/types.h>

```

Include dependency graph for inhibitor.h:



## Functions

- [L4\\_BEGIN\\_DECLS](#) long [l4re\\_inhibitor\\_acquire](#) ([l4\\_cap\\_idx\\_t](#) cap, [l4\\_umword\\_t](#) id, char const \*reason)  
*Acquire an inhibitor lock.*
- long [l4re\\_inhibitor\\_release](#) ([l4\\_cap\\_idx\\_t](#) cap, [l4\\_umword\\_t](#) id)  
*Release an inhibitor lock.*
- long [l4re\\_inhibitor\\_next\\_lock\\_info](#) ([l4\\_cap\\_idx\\_t](#) cap, char \*name, unsigned len, [l4\\_mword\\_t](#) current\_id)  
*Get information for the next available inhibitor lock.*

## 17.298.1 Detailed Description

Inhibitor C interface.

Definition in file [inhibitor.h](#).

## 17.298.2 Function Documentation

### 17.298.2.1 l4re\_inhibitor\_acquire()

```
L4_BEGIN_DECLS long l4re_inhibitor_acquire (
    l4_cap_idx_t cap,
    l4_umword_t id,
    char const * reason)
```

Acquire an inhibitor lock.

#### Parameters

<i>cap</i>	Capability for the Inhibitor object (see <a href="#">L4Re::Inhibitor</a> )
<i>id</i>	ID of the inhibitor lock that shall be acquired.
<i>reason</i>	Reason why the inhibitor lock is acquired. (Used for informing the user or debugging.)

#### Returns

0 for success, <0 on error.

#### See also

[L4Re::Inhibitor::acquire\(\)](#).

References [L4\\_CV](#), and [L4\\_EXPORT](#).

### 17.298.2.2 l4re\_inhibitor\_next\_lock\_info()

```
long l4re_inhibitor_next_lock_info (
    l4_cap_idx_t cap,
    char * name,
    unsigned len,
    l4_mword_t current_id)
```

Get information for the next available inhibitor lock.

#### Parameters

<i>cap</i>	Capability to the Inhibitor object (see <a href="#">L4Re::Inhibitor</a> )
<i>name</i>	A pointer to a buffer for the name of the lock.
<i>len</i>	The length of the available buffer (usually <a href="#">L4Re::Inhibitor::Name_max</a> is used).
<i>current↔ _id</i>	The ID of the last available lock, use -1 to get the first lock.

### Return values

<i>&gt;0</i>	The ID of the next available lock if there is one (in this case name shall contain the name of the inhibitor lock).
<i>-L4_ENODEV</i>	if there are no more locks.
<i>&lt;0</i>	Any other negative failure value.

### See also

[L4Re::Inhibitor::next\\_lock\\_info\(\)](#).

References [L4\\_END\\_DECLS](#).

### 17.298.2.3 l4re\_inhibitor\_release()

```
long l4re_inhibitor_release (
    l4_cap_idx_t cap,
    l4_umword_t id)
```

Release an inhibitor lock.

### Parameters

<i>cap</i>	Capability for the Inhibitor object (see <a href="#">L4Re::Inhibitor</a> ).
<i>id</i>	ID of inhibitor that shall be released.

### Returns

0 for success, <0 on error.

### See also

[L4Re::Inhibitor::release\(\)](#).

References [L4\\_CV](#), and [L4\\_EXPORT](#).

## 17.299 inhibitor.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * (c) 2014 Steffen Liebergeld <steffen.liebergeld@kernkonzept.com>
00003  *
00004  * License: see LICENSE.spdx (in this directory or the directories above)
00005  */
00006 #pragma once
00007
00013 #include <l4/sys/compiler.h>
00014 #include <l4/sys/types.h>
00015
00016 L4_BEGIN_DECLS
00017
00028 L4_CV long L4_EXPORT
00029 l4re_inhibitor_acquire(l4_cap_idx_t cap, l4_umword_t id,
00030                      char const *reason);
00031
00039 L4_CV long L4_EXPORT
00040 l4re_inhibitor_release(l4_cap_idx_t cap, l4_umword_t id);
00041
00057 L4_CV long L4_EXPORT
00058 l4re_inhibitor_next_lock_info(l4_cap_idx_t cap, char *name,
00059                              unsigned len, l4_mword_t current_id);
00060
00061 L4_END_DECLS
00062

```

## 17.300 l4/re/c/log.h File Reference

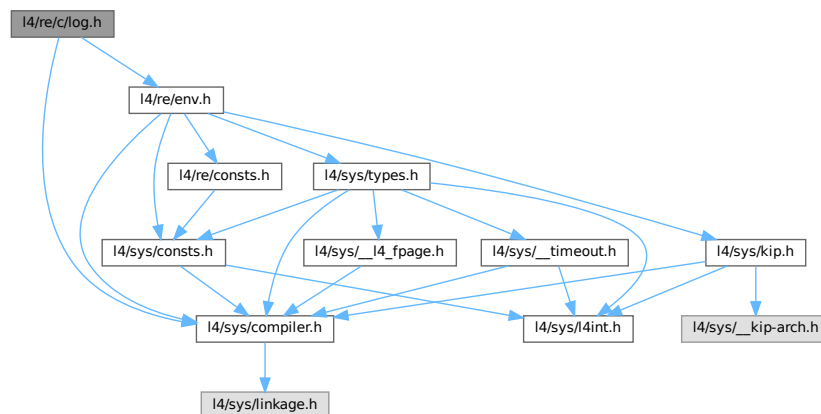
Log C interface.

```

#include <l4/re/env.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for log.h:



### Functions

- **L4\_BEGIN\_DECLS** void **l4re\_log\_print** (char const \*string) **L4\_NOTHROW**  
Write a null terminated string to the default log.
- void **l4re\_log\_printn** (char const \*string, int len) **L4\_NOTHROW**  
Write a string of a given length to the default log.

- void `l4re_log_print_srv` (const `l4_cap_idx_t` logcap, char const \*string) `L4_NOTHROW`  
Write a null terminated string to a log.
- void `l4re_log_printn_srv` (const `l4_cap_idx_t` logcap, char const \*string, int len) `L4_NOTHROW`  
Write a string of a given length to a log.

## 17.300.1 Detailed Description

Log C interface.

Definition in file [log.h](#).

## 17.301 log.h

[Go to the documentation of this file.](#)

```

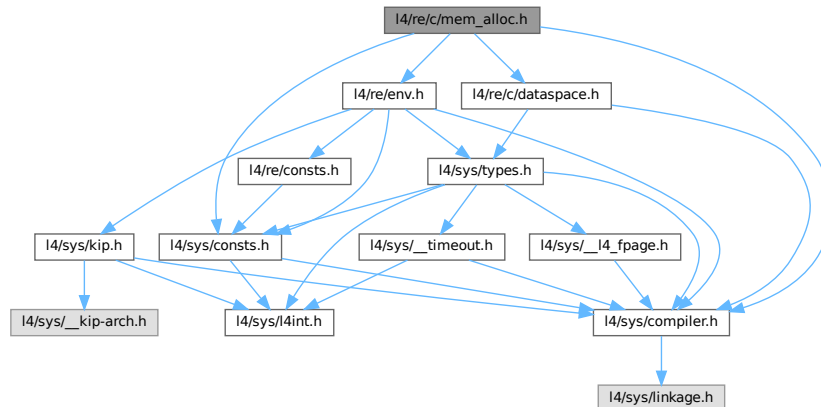
00001
00005 /*
00006  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *      economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00018
00019 #include <l4/re/env.h>
00020 #include <l4/sys/compiler.h>
00021
00022 L4_BEGIN_DECLS
00023
00032 L4_CV L4_INLINE void
00033 l4re_log_print(char const *string) L4_NOTHROW;
00034
00044 L4_CV L4_INLINE void
00045 l4re_log_printn(char const *string, int len) L4_NOTHROW;
00046
00047
00048
00049
00059 L4_CV void
00060 l4re_log_print_srv(const l4_cap_idx_t logcap,
00061                   char const *string) L4_NOTHROW;
00062
00073 L4_CV void
00074 l4re_log_printn_srv(const l4_cap_idx_t logcap,
00075                    char const *string, int len) L4_NOTHROW;
00076
00077
00078 /***** Implementations *****/
00079
00080 L4_CV L4_INLINE void
00081 l4re_log_print(char const *string) L4_NOTHROW
00082 {
00083     l4re_log_print_srv(l4re_global_env->log, string);
00084 }
00085
00086 L4_CV L4_INLINE void
00087 l4re_log_printn(char const *string, int len) L4_NOTHROW
00088 {
00089     l4re_log_printn_srv(l4re_global_env->log, string, len);
00090 }
00091
00092 L4_END_DECLS

```

## 17.302 I4/re/c/mem\_alloc.h File Reference

Memory allocator C interface.

```
#include <l4/re/env.h>
#include <l4/sys/compiler.h>
#include <l4/sys/consts.h>
#include <l4/re/c/dataspace.h>
Include dependency graph for mem_alloc.h:
```



### Enumerations

- enum [l4re\\_ma\\_flags](#)  
*Flags for requesting memory at the memory allocator.*

### Functions

- long [l4re\\_ma\\_alloc](#) (long size, [l4re\\_ds\\_t](#) const mem, unsigned long flags) [L4\\_NOTHROW](#)  
*Allocate memory.*
- long [l4re\\_ma\\_alloc\\_align](#) (long size, [l4re\\_ds\\_t](#) const mem, unsigned long flags, unsigned long align) [L4\\_NOTHROW](#)  
*Allocate memory.*
- long [l4re\\_ma\\_alloc\\_align\\_srv](#) ([l4\\_cap\\_idx\\_t](#) srv, long size, [l4re\\_ds\\_t](#) const mem, unsigned long flags, unsigned long align) [L4\\_NOTHROW](#)  
*Allocate memory.*

### 17.302.1 Detailed Description

Memory allocator C interface.

Definition in file [mem\\_alloc.h](#).

## 17.303 mem\_alloc.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *      economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00013 #include <l4/re/env.h>
00014 #include <l4/sys/compiler.h>
00015 #include <l4/sys/consts.h>
00016
00017 #include <l4/re/c/dataspace.h>
00018
00024
00025 L4_BEGIN_DECLS
00026
00032 enum l4re_ma_flags {
00033     L4RE_MA_CONTINUOUS = 0x01,
00034     L4RE_MA_PINNED    = 0x02,
00035     L4RE_MA_SUPER_PAGES = 0x04,
00036 };
00037
00038
00069 L4_CV L4_INLINE long
00070 l4re_ma_alloc(long size, l4re_ds_t const mem,
00071              unsigned long flags) L4_NOTHROW;
00072
00106 L4_CV L4_INLINE long
00107 l4re_ma_alloc_align(long size, l4re_ds_t const mem,
00108                    unsigned long flags, unsigned long align) L4_NOTHROW;
00109
00128 L4_CV long
00129 l4re_ma_alloc_align_srv(l4_cap_idx_t srv, long size,
00130                        l4re_ds_t const mem, unsigned long flags,
00131                        unsigned long align) L4_NOTHROW;
00132
00133 /***** Implementation *****/
00134
00135 L4_CV L4_INLINE long
00136 l4re_ma_alloc(long size, l4re_ds_t const mem,
00137              unsigned long flags) L4_NOTHROW
00138 {
00139     return l4re_ma_alloc_align_srv(l4re_global_env->mem_alloc, size, mem,
00140                                   flags, 0);
00141 }
00142
00143 L4_CV L4_INLINE long
00144 l4re_ma_alloc_align(long size, l4re_ds_t const mem,
00145                    unsigned long flags, unsigned long align) L4_NOTHROW
00146 {
00147     return l4re_ma_alloc_align_srv(l4re_global_env->mem_alloc, size, mem,
00148                                   flags, align);
00149 }
00150
00151 L4_END_DECLS

```

## 17.304 l4/re/c/namespace.h File Reference

Namespace functions, C interface.

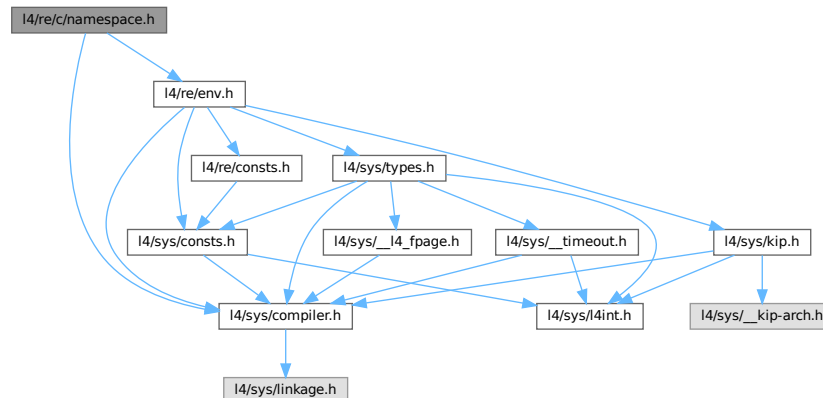
```

#include <l4/re/env.h>
#include <l4/sys/compiler.h>

```



Include dependency graph for namespace.h:



## Enumerations

- enum [l4re\\_ns\\_register\\_flags](#)  
Namespace register flags.

## Functions

- long [l4re\\_ns\\_query\\_to\\_srv](#) ([l4re\\_namespace\\_t](#) srv, char const \*name, [l4\\_cap\\_idx\\_t](#) const cap, int timeout) [L4\\_NOTHROW](#)  
Query the name space for the object named by *name*.
- long [l4re\\_ns\\_query\\_srv](#) ([l4re\\_namespace\\_t](#) srv, char const \*name, [l4\\_cap\\_idx\\_t](#) const cap) [L4\\_NOTHROW](#)  
Query the name space for the object named by *name*.
- long [l4re\\_ns\\_register\\_obj\\_srv](#) ([l4re\\_namespace\\_t](#) srv, char const \*name, [l4\\_cap\\_idx\\_t](#) const obj, unsigned flags) [L4\\_NOTHROW](#)  
Register an object with a name.

## Variables

- [L4\\_BEGIN\\_DECLS](#) typedef [l4\\_cap\\_idx\\_t](#) [l4re\\_namespace\\_t](#)  
Namespace type.

### 17.304.1 Detailed Description

Namespace functions, C interface.

Definition in file [namespace.h](#).

## 17.305 namespace.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00019
00020 #include <l4/re/env.h>
00021 #include <l4/sys/compiler.h>
00022
00029 enum l4re_ns_register_flags {
00030     L4RE_NS_REGISTER_RO = L4_FPAGE_RO,
00031     L4RE_NS_REGISTER_DIR = 0x10,
00032     L4RE_NS_REGISTER_RW = L4_FPAGE_RX,
00033     L4RE_NS_REGISTER_RWS = L4_FPAGE_RWX,
00034     L4RE_NS_REGISTER_S = L4_FPAGE_W,
00035 };
00036
00037 L4_BEGIN_DECLS
00038
00043 typedef l4_cap_idx_t l4re_namespace_t;
00044
00045
00046
00055 L4_CV long
00056 l4re_ns_query_to_srv(l4re_namespace_t srv, char const *name,
00057                     l4_cap_idx_t const cap, int timeout) L4_NOTHROW;
00058
00075 L4_CV L4_INLINE long
00076 l4re_ns_query_srv(l4re_namespace_t srv, char const *name,
00077                  l4_cap_idx_t const cap) L4_NOTHROW;
00078
00086 L4_CV long
00087 l4re_ns_register_obj_srv(l4re_namespace_t srv, char const *name,
00088                          l4_cap_idx_t const obj, unsigned flags) L4_NOTHROW;
00089
00090
00091
00092 /***** Implementation *****/
00093
00094 L4_CV L4_INLINE long
00095 l4re_ns_query_srv(l4re_namespace_t srv, char const *name,
00096                  l4_cap_idx_t const cap) L4_NOTHROW
00097 {
00098     return l4re_ns_query_to_srv(srv, name, cap, 40000);
00099 }
00100
00101
00102 L4_END_DECLS

```

## 17.306 l4/re/c/parent.h File Reference

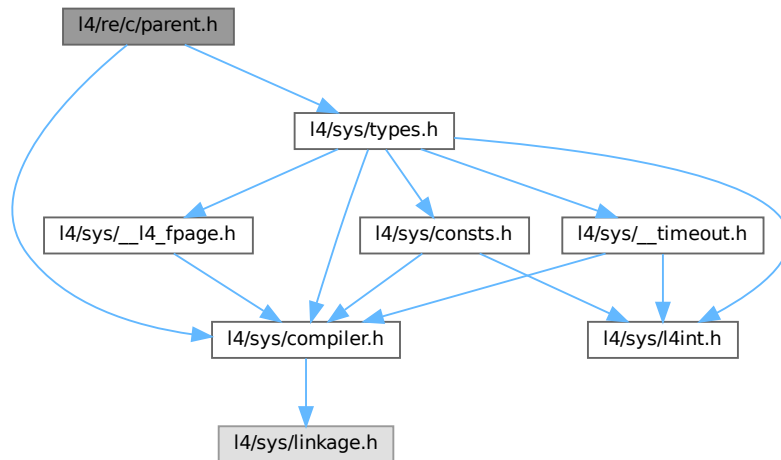
Parent C interface.

```

#include <l4/sys/compiler.h>
#include <l4/sys/types.h>

```

Include dependency graph for parent.h:



## Functions

- [L4\\_BEGIN\\_DECLS](#) long [l4re\\_parent\\_signal](#) ([l4\\_cap\\_idx\\_t](#) parent, unsigned long sig, unsigned long val)  
Send a signal using the parent protocol.

## 17.306.1 Detailed Description

Parent C interface.

Definition in file [parent.h](#).

## 17.306.2 Function Documentation

### 17.306.2.1 l4re\_parent\_signal()

```

L4_BEGIN_DECLS long l4re_parent_signal (
    l4_cap_idx_t parent,
    unsigned long sig,
    unsigned long val)

```

Send a signal using the parent protocol.

## Parameters

<i>parent</i>	Capability index of parent to send signal to.
<i>sig</i>	Signal to send
<i>val</i>	Value of the signal

## Return values

0	Success
<0	IPC error

See also

[L4Re::Parent::signal](#)

References [L4\\_END\\_DECLS](#).

## 17.307 parent.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (C) 2024 Kernkonzept GmbH.
00003  * Author(s): Marcus Haehnel <marcus.haehnel@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00012
00013 #pragma once
00014
00019
00020 #include <l4/sys/compiler.h>
00021 #include <l4/sys/types.h>
00022
00023 L4_BEGIN_DECLS
00024
00037 L4_CV long L4_EXPORT
00038 l4re_parent_signal(l4_cap_idx_t parent, unsigned long sig, unsigned long val);
00039
00040 L4_END_DECLS

```

## 17.308 l4/re/c/rm.h File Reference

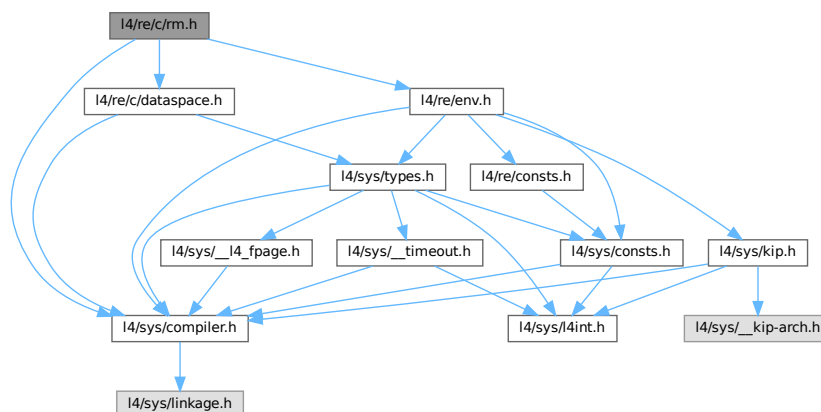
Region map interface, C interface.

```

#include <l4/re/env.h>
#include <l4/re/c/dataspace.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for rm.h:



## Enumerations

- enum `l4re_rm_flags_values` {  
`L4RE_RM_F_R` = `L4RE_DS_F_R` , `L4RE_RM_F_W` = `L4RE_DS_F_W` , `L4RE_RM_F_X` = `L4RE_DS_F_X`  
, `L4RE_RM_F_RX` = `L4RE_DS_F_RX` ,  
`L4RE_RM_F_RW` = `L4RE_DS_F_RW` , `L4RE_RM_F_RWX` = `L4RE_DS_F_RWX` , `L4RE_RM_F_KERNEL`  
= `0x100` , `L4RE_RM_F_DETACH_FREE` = `0x200` ,  
`L4RE_RM_F_PAGER` = `0x400` , `L4RE_RM_F_RESERVED` = `0x800` , `L4RE_RM_CACHING_SHIFT` = `4` ,  
`L4RE_RM_F_CACHING` = `L4RE_DS_F_CACHING_MASK` ,  
`L4RE_RM_REGION_FLAGS` = `0xffff` , `L4RE_RM_F_CACHE_NORMAL` = `L4RE_DS_F_NORMAL` ,  
`L4RE_RM_F_CACHE_BUFFERED` = `L4RE_DS_F_BUFFERABLE` , `L4RE_RM_F_CACHE_UNCACHED`  
= `L4RE_DS_F_UNCACHABLE` ,  
`L4RE_RM_F_SEARCH_ADDR` = `0x020000` , `L4RE_RM_F_IN_AREA` = `0x040000` , `L4RE_RM_F_EAGER_MAP`  
= `0x080000` , `L4RE_RM_F_NO_EAGER_MAP` = `0x100000` ,  
`L4RE_RM_F_ATTACH_FLAGS` = `0x1f0000` }

*Flags for region operations.*

## Functions

- int `l4re_rm_reserve_area` (`l4_addr_t` \*start, unsigned long size, `l4re_rm_flags_t` flags, unsigned char align) `L4_NOTHROW`  
*Reserve the given area in the region map.*
- int `l4re_rm_free_area` (`l4_addr_t` addr) `L4_NOTHROW`  
*Free an area from the region map.*
- int `l4re_rm_attach` (void \*\*start, unsigned long size, `l4re_rm_flags_t` flags, `l4re_ds_t` mem, `l4re_rm_offset_t` offs, unsigned char align) `L4_NOTHROW`  
*Attach a data space to a region.*
- int `l4re_rm_detach` (void \*addr) `L4_NOTHROW`  
*Detach and unmap a region from the address space in the current task.*
- int `l4re_rm_detach_ds` (void \*addr, `l4re_ds_t` \*ds) `L4_NOTHROW`  
*Detach and unmap a region and return affected dataspace in the current task.*
- int `l4re_rm_detach_unmap` (`l4_addr_t` addr, `l4_cap_idx_t` task) `L4_NOTHROW`  
*Detach and unmap in specified task.*
- int `l4re_rm_detach_ds_unmap` (void \*addr, `l4re_ds_t` \*ds, `l4_cap_idx_t` task) `L4_NOTHROW`  
*Detach and unmap in specified task.*
- int `l4re_rm_find` (`l4_addr_t` \*addr, unsigned long \*size, `l4re_rm_offset_t` \*offset, `l4re_rm_flags_t` \*flags, `l4re_ds_t` \*m) `L4_NOTHROW`  
*Find a region given an address and size.*
- int `l4re_rm_get_info` (`l4_addr_t` addr, char \*name, unsigned int len, `l4re_rm_offset_t` \*backing\_offset) `L4_NOTHROW`  
*Return auxiliary information of a region.*
- void `l4re_rm_show_lists` (void) `L4_NOTHROW`  
*Dump region map internal data structures.*
- int `l4re_rm_reserve_area_srv` (`l4_cap_idx_t` rm, `l4_addr_t` \*start, unsigned long size, `l4re_rm_flags_t` flags, unsigned char align) `L4_NOTHROW`
- int `l4re_rm_free_area_srv` (`l4_cap_idx_t` rm, `l4_addr_t` addr) `L4_NOTHROW`
- int `l4re_rm_attach_srv` (`l4_cap_idx_t` rm, void \*\*start, unsigned long size, `l4re_rm_flags_t` flags, `l4re_ds_t` mem, `l4re_rm_offset_t` offs, unsigned char align) `L4_NOTHROW`
- int `l4re_rm_detach_srv` (`l4_cap_idx_t` rm, `l4_addr_t` addr, `l4re_ds_t` \*ds, `l4_cap_idx_t` task) `L4_NOTHROW`
- int `l4re_rm_find_srv` (`l4_cap_idx_t` rm, `l4_addr_t` \*addr, unsigned long \*size, `l4re_rm_offset_t` \*offset, `l4re_rm_flags_t` \*flags, `l4re_ds_t` \*m) `L4_NOTHROW`
- int `l4re_rm_get_info_srv` (`l4_cap_idx_t` rm, `l4_addr_t` addr, char \*name, unsigned int len, `l4re_rm_offset_t` \*backing\_offset) `L4_NOTHROW`
- void `l4re_rm_show_lists_srv` (`l4_cap_idx_t` rm) `L4_NOTHROW`  
*Dump region map internal data structures.*

## 17.308.1 Detailed Description

Region map interface, C interface.

Definition in file [rm.h](#).

## 17.309 rm.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00019
00020 #include <l4/re/env.h>
00021 #include <l4/re/c/dataspace.h>
00022 #include <l4/sys/compiler.h>
00023
00024 L4_BEGIN_DECLS
00025
00030 enum l4re_rm_flags_values {
00031     L4RE_RM_F_R      = L4RE_DS_F_R,
00032     L4RE_RM_F_W      = L4RE_DS_F_W,
00033     L4RE_RM_F_X      = L4RE_DS_F_X,
00034     L4RE_RM_F_RX     = L4RE_DS_F_RX,
00035     L4RE_RM_F_RW     = L4RE_DS_F_RW,
00036     L4RE_RM_F_RWX    = L4RE_DS_F_RWX,
00037
00038     L4RE_RM_F_KERNEL      = 0x100,
00039     L4RE_RM_F_DETACH_FREE = 0x200,
00040     L4RE_RM_F_PAGER      = 0x400,
00041     L4RE_RM_F_RESERVED   = 0x800,
00042
00043     L4RE_RM_CACHING_SHIFT = 4,
00044
00046     L4RE_RM_F_CACHING      = L4RE_DS_F_CACHING_MASK,
00047
00048     L4RE_RM_REGION_FLAGS = 0xffff,
00049
00051     L4RE_RM_F_CACHE_NORMAL      = L4RE_DS_F_NORMAL,
00052
00054     L4RE_RM_F_CACHE_BUFFERED = L4RE_DS_F_BUFFERABLE,
00055
00057     L4RE_RM_F_CACHE_UNCACHED = L4RE_DS_F_UNCACHEABLE,
00058
00059     L4RE_RM_F_SEARCH_ADDR      = 0x020000,
00060     L4RE_RM_F_IN_AREA          = 0x040000,
00061     L4RE_RM_F_EAGER_MAP        = 0x080000,
00062     L4RE_RM_F_NO_EAGER_MAP     = 0x100000,
00063     L4RE_RM_F_ATTACH_FLAGS     = 0x1f0000,
00064 };
00065
00066 typedef l4_uint32_t l4re_rm_flags_t;
00067 typedef l4_uint64_t l4re_rm_offset_t;
00068
00077 L4_CV L4_INLINE int
00078 l4re_rm_reserve_area(l4_addr_t *start, unsigned long size,
00079                     l4re_rm_flags_t flags, unsigned char align) L4_NOTHROW;
00080
00089 L4_CV L4_INLINE int
00090 l4re_rm_free_area(l4_addr_t addr) L4_NOTHROW;
00091
00142 L4_CV L4_INLINE int
00143 l4re_rm_attach(void **start, unsigned long size, l4re_rm_flags_t flags,
00144               l4re_ds_t mem, l4re_rm_offset_t offs,
00145               unsigned char align) L4_NOTHROW;
00146
00147
00165 L4_CV L4_INLINE int
00166 l4re_rm_detach(void *addr) L4_NOTHROW;
00167
00187 L4_CV L4_INLINE int

```

```

00188 l4re_rm_detach_ds(void *addr, l4re_ds_t *ds) L4_NOTHROW;
00189
00201 L4_CV L4_INLINE int
00202 l4re_rm_detach_unmap(l4_addr_t addr, l4_cap_idx_t task) L4_NOTHROW;
00203
00218 L4_CV L4_INLINE int
00219 l4re_rm_detach_ds_unmap(void *addr, l4re_ds_t *ds,
00220                          l4_cap_idx_t task) L4_NOTHROW;
00221
00222
00229 L4_CV L4_INLINE int
00230 l4re_rm_find(l4_addr_t *addr, unsigned long *size,
00231             l4re_rm_offset_t *offset,
00232             l4re_rm_flags_t *flags, l4re_ds_t *m) L4_NOTHROW;
00233
00241 L4_CV L4_INLINE int
00242 l4re_rm_get_info(l4_addr_t addr,
00243                 char *name, unsigned int len,
00244                 l4re_rm_offset_t *backing_offset) L4_NOTHROW;
00245
00246
00253 L4_CV L4_INLINE void
00254 l4re_rm_show_lists(void) L4_NOTHROW;
00255
00256
00257 /*
00258  * Variants of functions that also take a capability of the region map
00259  * service.
00260  */
00261
00262
00267 L4_CV int
00268 l4re_rm_reserve_area_srv(l4_cap_idx_t rm, l4_addr_t *start, unsigned long size,
00269                          l4re_rm_flags_t flags, unsigned char align) L4_NOTHROW;
00270
00275 L4_CV int
00276 l4re_rm_free_area_srv(l4_cap_idx_t rm, l4_addr_t addr) L4_NOTHROW;
00277
00282 L4_CV int
00283 l4re_rm_attach_srv(l4_cap_idx_t rm, void **start, unsigned long size,
00284                   l4re_rm_flags_t flags, l4re_ds_t mem,
00285                   l4re_rm_offset_t offs,
00286                   unsigned char align) L4_NOTHROW;
00287
00288
00293 L4_CV int
00294 l4re_rm_detach_srv(l4_cap_idx_t rm, l4_addr_t addr,
00295                   l4re_ds_t *ds, l4_cap_idx_t task) L4_NOTHROW;
00296
00297
00302 L4_CV int
00303 l4re_rm_find_srv(l4_cap_idx_t rm, l4_addr_t *addr,
00304                 unsigned long *size, l4re_rm_offset_t *offset,
00305                 l4re_rm_flags_t *flags, l4re_ds_t *m) L4_NOTHROW;
00306
00307
00312 L4_CV int
00313 l4re_rm_get_info_srv(l4_cap_idx_t rm, l4_addr_t addr,
00314                     char *name, unsigned int len,
00315                     l4re_rm_offset_t *backing_offset) L4_NOTHROW;
00316
00321 L4_CV void
00322 l4re_rm_show_lists_srv(l4_cap_idx_t rm) L4_NOTHROW;
00323
00324
00325 /***** Implementations *****/
00326
00327 L4_CV L4_INLINE int
00328 l4re_rm_reserve_area(l4_addr_t *start, unsigned long size,
00329                     l4re_rm_flags_t flags, unsigned char align) L4_NOTHROW
00330 {
00331     return l4re_rm_reserve_area_srv(l4re_global_env->rm, start, size,
00332                                     flags, align);
00333 }
00334
00335 L4_CV L4_INLINE int
00336 l4re_rm_free_area(l4_addr_t addr) L4_NOTHROW
00337 {
00338     return l4re_rm_free_area_srv(l4re_global_env->rm, addr);
00339 }
00340
00341 L4_CV L4_INLINE int
00342 l4re_rm_attach(void **start, unsigned long size, l4re_rm_flags_t flags,
00343               l4re_ds_t mem, l4re_rm_offset_t offs,
00344               unsigned char align) L4_NOTHROW
00345 {
00346     return l4re_rm_attach_srv(l4re_global_env->rm, start, size,

```

```

00347             flags, mem, offs, align);
00348 }
00349
00350
00351 L4_CV L4_INLINE int
00352 l4re_rm_detach(void *addr) L4_NOTHROW
00353 {
00354     return l4re_rm_detach_srv(l4re_global_env->rm,
00355                               (l4_addr_t)addr, 0, L4_BASE_TASK_CAP);
00356 }
00357
00358 L4_CV L4_INLINE int
00359 l4re_rm_detach_unmap(l4_addr_t addr, l4_cap_idx_t task) L4_NOTHROW
00360 {
00361     return l4re_rm_detach_srv(l4re_global_env->rm, addr, 0, task);
00362 }
00363
00364 L4_CV L4_INLINE int
00365 l4re_rm_detach_ds(void *addr, l4re_ds_t *ds) L4_NOTHROW
00366 {
00367     return l4re_rm_detach_srv(l4re_global_env->rm, (l4_addr_t)addr,
00368                               ds, L4_BASE_TASK_CAP);
00369 }
00370
00371 L4_CV L4_INLINE int
00372 l4re_rm_detach_ds_unmap(void *addr, l4re_ds_t *ds, l4_cap_idx_t task) L4_NOTHROW
00373 {
00374     return l4re_rm_detach_srv(l4re_global_env->rm, (l4_addr_t)addr,
00375                               ds, task);
00376 }
00377
00378 L4_CV L4_INLINE int
00379 l4re_rm_find(l4_addr_t *addr, unsigned long *size,
00380             l4re_rm_offset_t *offset,
00381             l4re_rm_flags_t *flags, l4re_ds_t *m) L4_NOTHROW
00382 {
00383     return l4re_rm_find_srv(l4re_global_env->rm, addr, size, offset, flags, m);
00384 }
00385
00386 L4_CV L4_INLINE void
00387 l4re_rm_show_lists(void) L4_NOTHROW
00388 {
00389     l4re_rm_show_lists_srv(l4re_global_env->rm);
00390 }
00391
00392
00393
00394 L4_CV L4_INLINE int
00395 l4re_rm_get_info(l4_addr_t addr, char *name, unsigned int len,
00396                 l4re_rm_offset_t *backing_offset) L4_NOTHROW
00397 {
00398     return l4re_rm_get_info_srv(l4re_global_env->rm, addr, name, len,
00399                                 backing_offset);
00400 }
00401
00402 L4_END_DECLS

```

## 17.310 l4re/c/util/cap\_alloc.h File Reference

Capability allocator C interface.

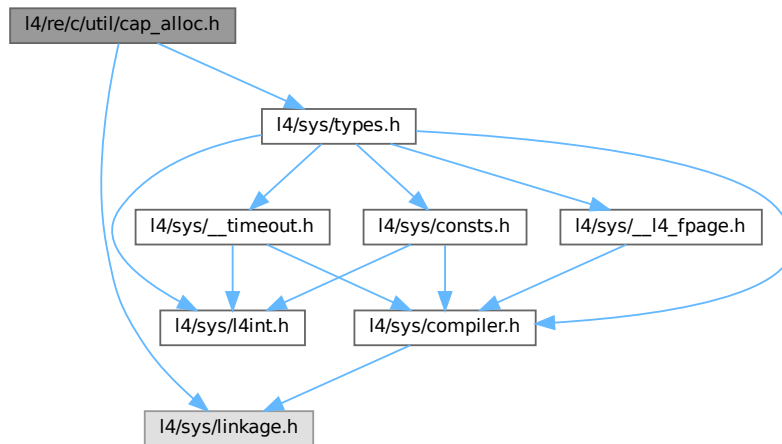
```

#include <l4/sys/types.h>
#include <l4/sys/linkage.h>

```



Include dependency graph for cap\_alloc.h:



## Functions

- `L4_BEGIN_DECLS l4_cap_idx_t l4re_util_cap_alloc (void) L4_NOTHROW`  
Get free capability index at capability allocator.
- `void l4re_util_cap_free (l4_cap_idx_t cap) L4_NOTHROW`  
Return capability index to capability allocator.
- `void l4re_util_cap_free_um (l4_cap_idx_t cap) L4_NOTHROW`  
Return capability index to capability allocator, and unmaps the object.
- `long l4re_util_cap_last (void) L4_NOTHROW`  
Return last capability index the allocator can return.

### 17.310.1 Detailed Description

Capability allocator C interface.

Definition in file [cap\\_alloc.h](#).

## 17.311 cap\_alloc.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00019
00020 #include <l4/sys/types.h>
00021 #include <l4/sys/linkage.h>

```

```

00022
00023 L4_BEGIN_DECLS
00024
00029 L4_CV l4_cap_idx_t
00030 l4re_util_cap_alloc(void) L4_NOTHROW;
00031
00036 L4_CV void
00037 l4re_util_cap_free(l4_cap_idx_t cap) L4_NOTHROW;
00038
00044 L4_CV void
00045 l4re_util_cap_free_um(l4_cap_idx_t cap) L4_NOTHROW;
00046
00052 L4_CV long
00053 l4re_util_cap_last(void) L4_NOTHROW;
00054
00055 L4_END_DECLS

```

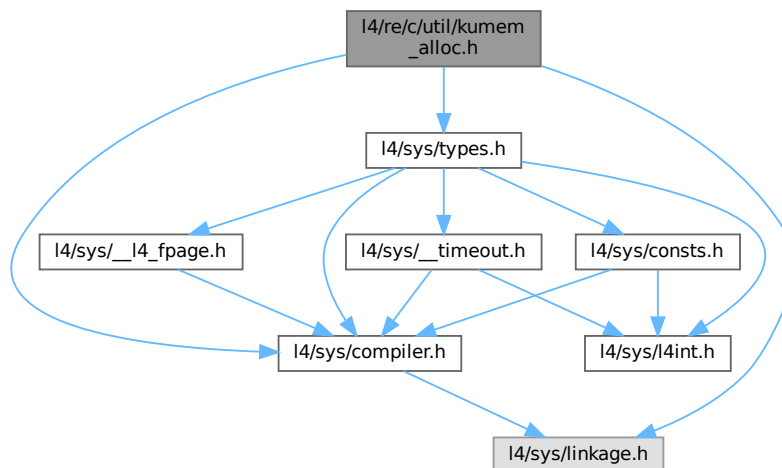
## 17.312 l4/re/c/util/kumem\_alloc.h File Reference

Kumem allocator utility C interface.

```

#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <l4/sys/linkage.h>
Include dependency graph for kumem_alloc.h:

```



### Functions

- `L4_BEGIN_DECLS` `int l4re_util_kumem_alloc (l4_addr_t *mem, unsigned pages_order, l4_cap_idx_t task, l4_cap_idx_t rm) L4_NOTHROW`  
Allocate state area.

### 17.312.1 Detailed Description

Kumem allocator utility C interface.

Definition in file `kumem_alloc.h`.

## 17.312.2 Function Documentation

### 17.312.2.1 l4re\_util\_kumem\_alloc()

```
L4_BEGIN_DECLS int l4re_util_kumem_alloc (
    l4_addr_t * mem,
    unsigned pages_order,
    l4_cap_idx_t task,
    l4_cap_idx_t rm)
```

Allocate state area.

#### Parameters

out	<i>mem</i>	Pointer to memory that has been allocated.
	<i>pages_order</i>	Size to allocate, in log2 pages.
	<i>task</i>	Task to use for allocation.
	<i>rm</i>	Region manager to use for allocation.

#### Return values

0	for success
<0	error code on failure

#### Note

The amount of kernel-user memory that can be allocated at once is limited by the used kernel implementation. The minimum allocatable amount is one page. A portable implementation should not depend on allocations greater than 16KiB to succeed.

#### Examples

[examples/sys/aliens/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

References [L4\\_END\\_DECLS](#), and [L4\\_NOTHROW](#).

## 17.313 kumem\_alloc.h

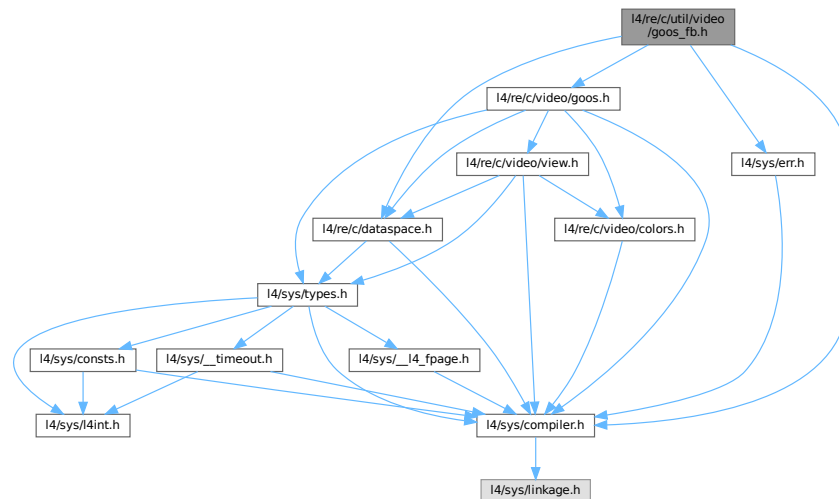
[Go to the documentation of this file.](#)

```
00001
00005 /*
00006  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00019
00020 #include <l4/sys/compiler.h>
00021 #include <l4/sys/types.h>
00022 #include <l4/sys/linkage.h>
00023
00024 L4_BEGIN_DECLS
00025
00029 L4_CV int
00030 l4re_util_kumem_alloc(l4_addr_t *mem, unsigned pages_order,
00031                      l4_cap_idx_t task, l4_cap_idx_t rm) L4_NOTHROW;
00032
00033 L4_END_DECLS
```

## 17.314 l4/re/c/util/video/goos\_fb.h File Reference

Framebuffer utility functionality.

```
#include <l4/sys/compiler.h>
#include <l4/re/c/video/goos.h>
#include <l4/sys/err.h>
#include <l4/re/c/dataspace.h>
Include dependency graph for goos_fb.h:
```



### 17.314.1 Detailed Description

Framebuffer utility functionality.

Definition in file [goos\\_fb.h](#).

## 17.315 goos\_fb.h

[Go to the documentation of this file.](#)

```
00001
00002 /*
00003  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <l4/sys/compiler.h>
00011 #include <l4/re/c/video/goos.h>
00012 #include <l4/sys/err.h>
00013 #include <l4/re/c/dataspace.h>
00014
00015 L4_BEGIN_DECLS
00016
00017 typedef struct
00018 {
00019     unsigned long _obj_buf[6];
00020 } l4re_util_video_goos_fb_t;
00021
```

```

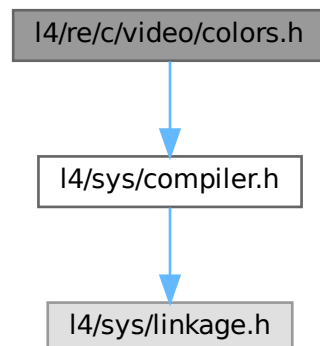
00024
00025 L4_CV int
00026 l4re_util_video_goos_fb_setup_name(l4re_util_video_goos_fb_t *goosfb,
00027                                     char const *name) L4_NOTHROW;
00028
00029 L4_CV void
00030 l4re_util_video_goos_fb_destroy(l4re_util_video_goos_fb_t *goosfb) L4_NOTHROW;
00031
00032 L4_CV int
00033 l4re_util_video_goos_fb_view_info(l4re_util_video_goos_fb_t *goosfb,
00034                                   l4re_video_view_info_t *info) L4_NOTHROW;
00035
00036 L4_CV void *
00037 l4re_util_video_goos_fb_attach_buffer(l4re_util_video_goos_fb_t *goosfb) L4_NOTHROW;
00038
00039 L4_CV int
00040 l4re_util_video_goos_fb_refresh(l4re_util_video_goos_fb_t *goosfb,
00041                                int x, int y, int w, int h) L4_NOTHROW;
00042
00043 L4_CV l4re_ds_t
00044 l4re_util_video_goos_fb_buffer(l4re_util_video_goos_fb_t *goosfb) L4_NOTHROW;
00045
00046 L4_CV l4_cap_idx_t
00047 l4re_util_video_goos_fb_goos(l4re_util_video_goos_fb_t *goosfb) L4_NOTHROW;
00048
00049 L4_END_DECLS

```

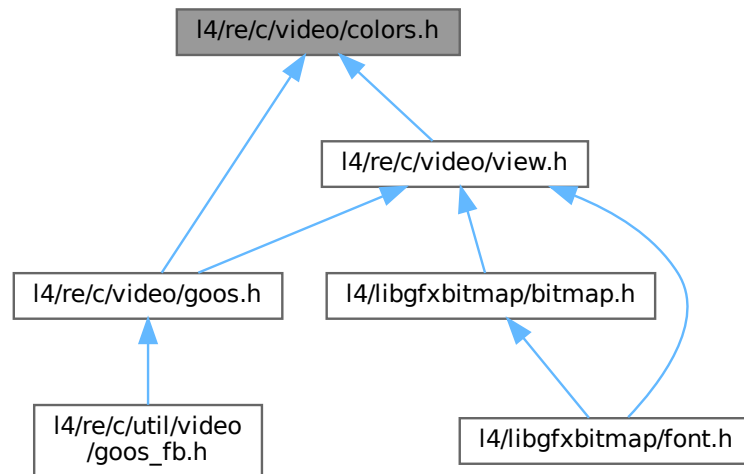
## 17.316 l4/re/c/video/colors.h File Reference

#include <l4/sys/compiler.h>

Include dependency graph for colors.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [l4re\\_video\\_color\\_component\\_t](#)  
*Color component structure.*
- struct [l4re\\_video\\_pixel\\_info\\_t](#)  
*Pixel\_info structure.*

## Typedefs

- typedef struct l4re\_video\_color\_component\_t **l4re\_video\_color\_component\_t**  
*Color component structure.*
- typedef struct l4re\_video\_pixel\_info\_t **l4re\_video\_pixel\_info\_t**  
*Pixel\_info structure.*

## 17.316.1 Detailed Description

### Note

The C interface of L4Re::Video does *NOT* reflect the full C++ interface on purpose. Use the C++ interface where possible.

Definition in file [colors.h](#).

## 17.317 colors.h

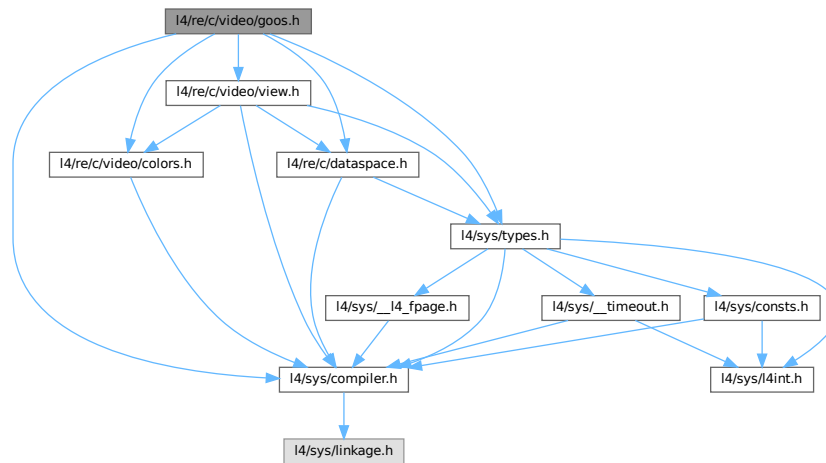
[Go to the documentation of this file.](#)

```
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 #include <l4/sys/compiler.h>
00015
00020 typedef struct l4re_video_color_component_t
00021 {
00022     unsigned char size;
00023     unsigned char shift;
00024 } __attribute__((packed)) l4re_video_color_component_t;
00025
00030 typedef struct l4re_video_pixel_info_t
00031 {
00032     l4re_video_color_component_t r, g, b, a;
00033     unsigned char bytes_per_pixel;
00034 } l4re_video_pixel_info_t;
00035
00036 L4_BEGIN_DECLS
00037
00038 L4_INLINE L4_CV int
00039 l4re_video_bits_per_pixel(l4re_video_pixel_info_t *p) L4_NOTHROW;
00040
00041 /* *****
00042  * Implementations */
00043
00044 L4_INLINE L4_CV int
00045 l4re_video_bits_per_pixel(l4re_video_pixel_info_t *p) L4_NOTHROW
00046 {
00047     return p->r.size + p->b.size + p->g.size + p->a.size;
00048 }
00049
00050 L4_END_DECLS
```

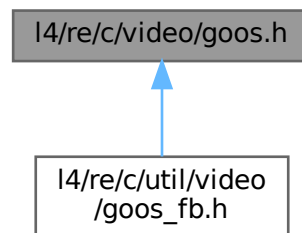
## 17.318 l4re/c/video/goos.h File Reference

```
#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <l4/re/c/dataspace.h>
#include <l4/re/c/video/colors.h>
#include <l4/re/c/video/view.h>
```

Include dependency graph for goos.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [l4re\\_video\\_goos\\_info\\_t](#)  
*Goos information structure.*

## Typedefs

- typedef [l4\\_cap\\_idx\\_t](#) [l4re\\_video\\_goos\\_t](#)  
*Goos object type.*

## Enumerations

- enum [l4re\\_video\\_goos\\_info\\_flags\\_t](#) { [F\\_l4re\\_video\\_goos\\_auto\\_refresh](#) = 0x01 , [F\\_l4re\\_video\\_goos\\_pointer](#) = 0x02 , [F\\_l4re\\_video\\_goos\\_dynamic\\_views](#) = 0x04 , [F\\_l4re\\_video\\_goos\\_dynamic\\_buffers](#) = 0x08 }  
*Flags of information on the goos.*



## Functions

- `L4_BEGIN_DECLS` `int` `l4re_video_goos_info` (`l4re_video_goos_t` goos, `l4re_video_goos_info_t` \*ginfo) `L4_NOTHROW`  
*Get information on a goos.*
- `int` `l4re_video_goos_refresh` (`l4re_video_goos_t` goos, `int` x, `int` y, `int` w, `int` h) `L4_NOTHROW`  
*Flush a rectangle of pixels of the goos screen.*
- `int` `l4re_video_goos_create_buffer` (`l4re_video_goos_t` goos, `unsigned long` size, `l4_cap_idx_t` buffer) `L4_NOTHROW`  
*Create a new buffer (memory buffer) for pixel data.*
- `int` `l4re_video_goos_delete_buffer` (`l4re_video_goos_t` goos, `unsigned` idx) `L4_NOTHROW`  
*Delete a pixel buffer.*
- `int` `l4re_video_goos_get_static_buffer` (`l4re_video_goos_t` goos, `unsigned` idx, `l4_cap_idx_t` buffer) `L4_NOTHROW`  
*Get the data-space capability of the static pixel buffer.*
- `int` `l4re_video_goos_create_view` (`l4re_video_goos_t` goos, `l4re_video_view_t` \*view) `L4_NOTHROW`  
*Create a new view (.*
- `int` `l4re_video_goos_delete_view` (`l4re_video_goos_t` goos, `l4re_video_view_t` \*view) `L4_NOTHROW`  
*Delete a view.*
- `int` `l4re_video_goos_get_view` (`l4re_video_goos_t` goos, `unsigned` idx, `l4re_video_view_t` \*view) `L4_NOTHROW`  
*Get a view for the given index.*

### 17.318.1 Detailed Description

#### Note

The C interface of L4Re::Video does *NOT* reflect the full C++ interface on purpose. Use the C++ where possible.

Definition in file [goos.h](#).

## 17.319 goos.h

[Go to the documentation of this file.](#)

```

00001
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <l4/sys/compiler.h>
00011 #include <l4/sys/types.h>
00012 #include <l4/re/c/dataspace.h>
00013 #include <l4/re/c/video/colors.h>
00014 #include <l4/re/c/video/view.h>
00015
00016
00017
00018
00019
00020
00021 enum l4re_video_goos_info_flags_t
00022 {
00023     F_l4re_video_goos_auto_refresh = 0x01,
00024     F_l4re_video_goos_pointer      = 0x02,
00025     F_l4re_video_goos_dynamic_views = 0x04,
00026     F_l4re_video_goos_dynamic_buffers = 0x08,
00027 };
00028
00029

```

```

00041 typedef struct
00042 {
00043     unsigned long width;
00044     unsigned long height;
00045     unsigned flags;
00046     unsigned num_static_views;
00047     unsigned num_static_buffers;
00048     l4re_video_pixel_info_t pixel_info;
00049 } l4re_video_goos_info_t;
00050
00055 typedef l4_cap_idx_t l4re_video_goos_t;
00056
00057 L4_BEGIN_DECLS
00058
00070 L4_CV int
00071 l4re_video_goos_info(l4re_video_goos_t goos,
00072                     l4re_video_goos_info_t *ginfo) L4_NOTHROW;
00073
00083 L4_CV int
00084 l4re_video_goos_refresh(l4re_video_goos_t goos, int x, int y, int w,
00085                        int h) L4_NOTHROW;
00086
00098 L4_CV int
00099 l4re_video_goos_create_buffer(l4re_video_goos_t goos, unsigned long size,
00100                             l4_cap_idx_t buffer) L4_NOTHROW;
00101
00109 L4_CV int
00110 l4re_video_goos_delete_buffer(l4re_video_goos_t goos, unsigned idx) L4_NOTHROW;
00111
00122 L4_CV int
00123 l4re_video_goos_get_static_buffer(l4re_video_goos_t goos, unsigned idx,
00124                                  l4_cap_idx_t buffer) L4_NOTHROW;
00125
00132 L4_CV int
00133 l4re_video_goos_create_view(l4re_video_goos_t goos,
00134                             l4re_video_view_t *view) L4_NOTHROW;
00135
00143 L4_CV int
00144 l4re_video_goos_delete_view(l4re_video_goos_t goos,
00145                             l4re_video_view_t *view) L4_NOTHROW;
00146
00147
00159 L4_CV int
00160 l4re_video_goos_get_view(l4re_video_goos_t goos, unsigned idx,
00161                          l4re_video_view_t *view) L4_NOTHROW;
00162
00163 L4_END_DECLS

```

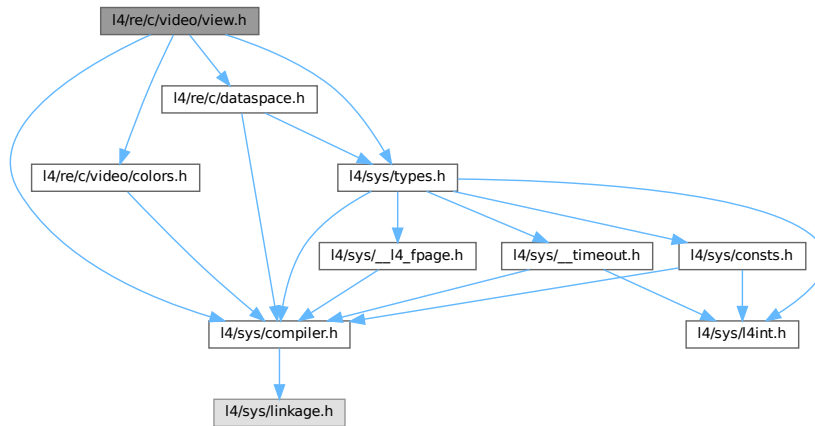
## 17.320 l4re/c/video/view.h File Reference

```

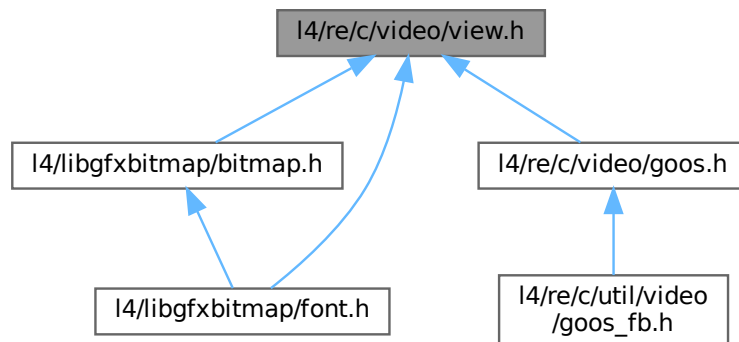
#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <l4/re/c/dataspace.h>
#include <l4/re/c/video/colors.h>

```

Include dependency graph for view.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [l4re\\_video\\_view\\_info\\_t](#)  
View information structure.
- struct [l4re\\_video\\_view\\_t](#)  
C representation of a goos view.

## Typedefs

- typedef struct l4re\_video\_view\_info\_t [l4re\\_video\\_view\\_info\\_t](#)  
View information structure.
- typedef struct l4re\_video\_view\_t [l4re\\_video\\_view\\_t](#)  
C representation of a goos view.

## Enumerations

- enum `l4re_video_view_info_flags_t` {  
`F_l4re_video_view_none` = 0x00 , `F_l4re_video_view_set_buffer` = 0x01 , `F_l4re_video_view_set_buffer_offset` = 0x02 , `F_l4re_video_view_set_bytes_per_line` = 0x04 ,  
`F_l4re_video_view_set_pixel` = 0x08 , `F_l4re_video_view_set_position` = 0x10 , `F_l4re_video_view_dyn_allocated` = 0x20 , `F_l4re_video_view_set_background` = 0x40 ,  
`F_l4re_video_view_set_flags` = 0x80 , **`F_l4re_video_view_fully_dynamic`** , `F_l4re_video_view_above` = 0x01000 , `F_l4re_video_view_flags_mask` = 0xff000 }

*Flags of information on a view.*

## Functions

- `L4_BEGIN_DECLS` int `l4re_video_view_refresh` (`l4re_video_view_t` \*view, int x, int y, int w, int h) `L4_NOTHROW`  
*Flush the given rectangle of pixels of the given view.*
- int `l4re_video_view_get_info` (`l4re_video_view_t` \*view, `l4re_video_view_info_t` \*info) `L4_NOTHROW`  
*Retrieve information about the given view.*
- int `l4re_video_view_set_info` (`l4re_video_view_t` \*view, `l4re_video_view_info_t` \*info) `L4_NOTHROW`  
*Set properties of the view.*
- int `l4re_video_view_set_viewport` (`l4re_video_view_t` \*view, int x, int y, int w, int h, unsigned long bofs) `L4_NOTHROW`  
*Set the viewport parameters of a view.*
- int `l4re_video_view_stack` (`l4re_video_view_t` \*view, `l4re_video_view_t` \*pivot, int behind) `L4_NOTHROW`  
*Change the stacking order in the stack of visible views.*

## 17.320.1 Detailed Description

### Note

The C interface of `L4Re::Video` does *NOT* reflect the full C++ interface on purpose. Use the C++ where possible.

Definition in file [view.h](#).

## 17.321 view.h

[Go to the documentation of this file.](#)

```
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 #include <l4/sys/compiler.h>
00015 #include <l4/sys/types.h>
00016 #include <l4/re/c/dataspace.h>
00017 #include <l4/re/c/video/colors.h>
00018
00023 enum l4re_video_view_info_flags_t
00024 {
00025     F_l4re_video_view_none           = 0x00,
00026     F_l4re_video_view_set_buffer     = 0x01,
00027     F_l4re_video_view_set_buffer_offset = 0x02,
```

```

00028 F_l4re_video_view_set_bytes_per_line = 0x04,
00029 F_l4re_video_view_set_pixel          = 0x08,
00030 F_l4re_video_view_set_position        = 0x10,
00031 F_l4re_video_view_dyn_allocated       = 0x20,
00032 F_l4re_video_view_set_background     = 0x40,
00033 F_l4re_video_view_set_flags          = 0x80,
00034 F_l4re_video_view_fully_dynamic      = F_l4re_video_view_set_buffer
00035                                     | F_l4re_video_view_set_buffer_offset
00036                                     | F_l4re_video_view_set_bytes_per_line
00037                                     | F_l4re_video_view_set_pixel
00038                                     | F_l4re_video_view_set_position
00039                                     | F_l4re_video_view_dyn_allocated,
00040
00041 F_l4re_video_view_above                = 0x01000,
00042 F_l4re_video_view_flags_mask          = 0xff000,
00043 };
00044
00049 typedef struct l4re_video_view_info_t
00050 {
00051     unsigned          flags;
00052     unsigned          view_index;
00053     unsigned long     xpos, ypos, width, height;
00054     unsigned long     buffer_offset;
00055     unsigned long     bytes_per_line;
00056     l4re_video_pixel_info_t pixel_info;
00057     unsigned          buffer_index;
00058 } l4re_video_view_info_t;
00059
00060
00068 typedef struct l4re_video_view_t
00069 {
00070     l4_cap_idx_t goos;
00071     unsigned idx;
00072 } l4re_video_view_t;
00073
00074
00075 L4_BEGIN_DECLS
00076
00087 L4_CV int
00088 l4re_video_view_refresh(l4re_video_view_t *view, int x, int y, int w,
00089                        int h) L4_NOTHROW;
00090
00098 L4_CV int
00099 l4re_video_view_get_info(l4re_video_view_t *view,
00100                          l4re_video_view_info_t *info) L4_NOTHROW;
00101
00113 L4_CV int
00114 l4re_video_view_set_info(l4re_video_view_t *view,
00115                          l4re_video_view_info_t *info) L4_NOTHROW;
00116
00133 L4_CV int
00134 l4re_video_view_set_viewport(l4re_video_view_t *view, int x, int y, int w,
00135                              int h, unsigned long bofs) L4_NOTHROW;
00136
00147 L4_CV int
00148 l4re_video_view_stack(l4re_video_view_t *view, l4re_video_view_t *pivot,
00149                      int behind) L4_NOTHROW;
00150
00151 L4_END_DECLS
00152

```

## 17.322 console

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 #include <l4/re/video/goos>
00013 #include <l4/re/event>
00014
00015 namespace L4Re {
00016
00022
00023
00028 class L4_EXPORT Console :
00029     public L4::Kobject_2t<Console, Video::Goos, Event, L4::PROTO_EMPTY>

```



## Data Structures

- class [L4Re::Dataspace](#)  
*Interface for memory-like objects.*
- struct [L4Re::Dataspace::F](#)  
*Dataspace flags definitions.*
- struct [L4Re::Dataspace::Stats](#)  
*Information about the dataspace.*

## Namespaces

- namespace [L4Re](#)  
*L4Re C++ Interfaces.*

### 17.323.1 Detailed Description

Dataspace interface.

Definition in file [dataspace](#).

## 17.324 dataspace

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00010  *               Björn Döbel <doebel@os.inf.tu-dresden.de>,
00011  *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00012  *               economic rights: Technische Universität Dresden (Germany)
00013  *
00014  * License: see LICENSE.spdx (in this directory or the directories above)
00015  */
00016
00017 #pragma once
00018
00019 #include <l4/bid_config.h>
00020 #include <l4/sys/types.h>
00021 #include <l4/sys/l4int.h>
00022 #include <l4/sys/capability>
00023 #include <l4/re/protocols.h>
00024 #include <l4/sys/cxx/ipc_types>
00025 #include <l4/sys/cxx/ipc_iface>
00026 #include <l4/sys/cxx/types>
00027
00028 namespace L4Re
00029 {
00030
00031     // MISSING:
00032     // * size support in map, mapped size in reply
00033
00050     class L4_EXPORT Dataspace :
00051     public L4::Kobject_t<Dataspace, L4::Kobject, L4RE_PROTO_DATASPACE,
00052                             L4::Type_info::Demand_t<1> >
00053     {
00054     public:
00055
00057         struct F
00058         {
00059             enum
00060             {
00061                 Caching_shift = 4,
00062             };
00063

```

```

00070     enum Flags
00071     {
00072         R    = L4_FPAGE_RO,
00073         Ro   = L4_FPAGE_RO,
00074         RW   = L4_FPAGE_RW,
00075         W    = L4_FPAGE_W,
00076         X    = L4_FPAGE_X,
00077         RX   = L4_FPAGE_RX,
00078         RWX  = L4_FPAGE_RWX,
00079         Rights_mask = 0x0f,
00080
00081         Normal      = 0x00,
00082         Cacheable   = Normal,
00083         Bufferable   = 0x10,
00084         Uncacheable = 0x20,
00085         Caching_mask = 0x30,
00086     };
00087
00088     L4_TYPES_FLAGS_OPS_DEF(Flags);
00089 };
00090
00091 struct Flags : L4::Types::Flags_ops_t<Flags>
00092 {
00093     unsigned long raw;
00094     Flags() = default;
00095     explicit constexpr Flags(unsigned long f) : raw(f) {}
00096     constexpr Flags(F::Flags f) : raw(f) {}
00097     constexpr bool r() const { return raw & L4_FPAGE_RO; }
00098     constexpr bool w() const { return raw & L4_FPAGE_W; }
00099     constexpr bool x() const { return raw & L4_FPAGE_X; }
00100
00101     constexpr unsigned long fpage_rights() const
00102     { return raw & 0xf; }
00103 };
00104
00105 typedef l4_uint64_t Size;
00106 typedef l4_uint64_t Offset;
00107 typedef l4_uint64_t Map_addr;
00108
00109 struct Stats
00110 {
00111     Size size;
00112     Flags flags;
00113 };
00114
00115 long map(Offset offset, Flags flags, Map_addr local_addr,
00116         Map_addr min_addr, Map_addr max_addr,
00117         L4::Cap<L4::Task> dst = L4::Cap<L4::Task>::Invalid) const noexcept;
00118
00119 long map_region(Offset offset, Flags flags,
00120                Map_addr min_addr, Map_addr max_addr,
00121                L4::Cap<L4::Task> dst = L4::Cap<L4::Task>::Invalid) const noexcept;
00122
00123 L4_RPC(long, clear, (Offset offset, Size size));
00124
00125 L4_RPC(long, allocate, (Offset offset, Size size));
00126
00127 L4_RPC(long, copy_in, (Offset dst_offs, L4::Ipc::Cap<Dataspace> src,
00128                      Offset src_offs, Size size));
00129
00130 Size size() const noexcept;
00131
00132 Flags flags() const noexcept;
00133
00134 L4_RPC(long, info, (Stats *stats));
00135
00136 L4_RPC_NF(long, map, (Offset offset, Map_addr spot,
00137                     Flags flags, L4::Ipc::Rcv_fpage r,
00138                     L4::Ipc::Snd_fpage &fp));
00139
00140 #ifdef CONFIG_MMU
00141     L4_RPC_NF(long, map_info, (l4_addr_t *start_addr, l4_addr_t *end_addr));
00142     inline long map_info([[maybe_unused]] l4_addr_t *start_addr,
00143                        [[maybe_unused]] l4_addr_t *end_addr)
00144     { return 0; }
00145 #else
00146     L4_RPC(long, map_info, (l4_addr_t *start_addr, l4_addr_t *end_addr));
00147 #endif
00148 private:
00149     long __map(Offset offset, unsigned char *order, Flags flags,
00150               Map_addr local_addr, L4::Cap<L4::Task> dst) const noexcept;
00151 public:
00152     typedef L4::Typeid::Rpcs<map_t, clear_t, info_t, copy_in_t,

```

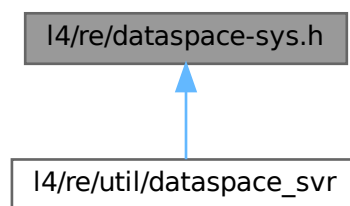


```
00319             allocate_t, map_info_t> RpcS;  
00320  
00321 };  
00322  
00323 }  
00324
```

## 17.325 l4/re/dataspace-sys.h File Reference

Dataspace protocol defintion.

This graph shows which files directly or indirectly include this file:



### Namespaces

- namespace [L4Re](#)  
*L4Re C++ Interfaces.*

### Enumerations

- enum [L4Re::Dataspace\\_::Opcodes](#)  
*Data-space communication-protocol opcodes.*

### 17.325.1 Detailed Description

Dataspace protocol defintion.

Definition in file [dataspace-sys.h](#).

## 17.326 dataspace-sys.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 namespace L4Re
00015 {
00016     namespace Dataspace_
00017     {
00023         enum Opcodes { Map, Clear, Stats, Copy, Take, Release, Allocate };
00024     };
00025 };
00026

```

## 17.327 dbg\_events

```

00001 // vim:ft=cpp
00002
00003 #pragma once
00004
00005 #include <l4/sys/cxx/ipc_epiface>
00006 #include <l4/sys/utcb.h>
00007 #include <l4/re/remote_access>
00008 #include <l4/re/rm>
00009
00010 namespace L4Re {
00011     struct Dbg_events : L4::Kobject_t<Dbg_events, L4::Kobject, 0,
00012                                     L4::Type_info::Demand_t<2> >
00013     {
00014         L4_INLINE_RPC(int, request_backtrace, (l4_exc_regs_t regs,
00015                                             L4::Ipc::Cap<L4Re::Remote_access> raif,
00016                                             L4::Ipc::Cap<L4Re::Rm> rmif));
00017
00018         typedef L4::Typeid::Rpc<request_backtrace_t> Rpc;
00019     };
00020 } // L4Re

```

## 17.328 l4/re/dma\_space File Reference

```

#include <l4/sys/types.h>
#include <l4/sys/l4int.h>
#include <l4/sys/capability>
#include <l4/re/dataspace>
#include <l4/re/protocols.h>
#include <l4/sys/cxx/types>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_iface>

```

```

graph TD
    I4_re_dma_space["I4/re/dma_space"]
    I4_libblock_device_types_h["I4/libblock-device/types.h"]
    I4_vbus_vbus["I4/vbus/vbus"]
    I4_libblock_device_inout_memory_h["I4/libblock-device/inout_memory.h"]
    I4_vbus_vbus_generic["I4/vbus/vbus_generic"]
    I4_vbus_vbus_gpio["I4/vbus/vbus_gpio"]
    I4_vbus_vbus_pci["I4/vbus/vbus_pci"]
    I4_libblock_device_partition_h["I4/libblock-device/partition.h"]
    I4_libblock_device_device_h["I4/libblock-device/device.h"]
    I4_libblock_device_virtio_client_h["I4/libblock-device/virtio_client.h"]
    I4_libblock_device_part_device_h["I4/libblock-device/part_device.h"]
    I4_libblock_device_scheduler_h["I4/libblock-device/scheduler.h"]
    I4_libblock_device_block_device_mgr_h["I4/libblock-device/block_device_mgr.h"]

    I4_re_dma_space --> I4_libblock_device_types_h
    I4_re_dma_space --> I4_vbus_vbus
    I4_re_dma_space --> I4_libblock_device_inout_memory_h
    I4_libblock_device_types_h --> I4_libblock_device_device_h
    I4_vbus_vbus --> I4_vbus_vbus_generic
    I4_vbus_vbus --> I4_vbus_vbus_gpio
    I4_vbus_vbus --> I4_vbus_vbus_pci
    I4_libblock_device_inout_memory_h --> I4_libblock_device_partition_h
    I4_libblock_device_device_h --> I4_libblock_device_part_device_h
    I4_libblock_device_device_h --> I4_libblock_device_virtio_client_h
    I4_libblock_device_partition_h --> I4_libblock_device_part_device_h
    I4_libblock_device_virtio_client_h --> I4_libblock_device_scheduler_h
    I4_libblock_device_part_device_h --> I4_libblock_device_block_device_mgr_h
    I4_libblock_device_scheduler_h --> I4_libblock_device_block_device_mgr_h
    I4_libblock_device_block_device_mgr_h --> I4_libblock_device_block_device_mgr_h
  
```

- class `L4Re::Dma_space`  
*Managed DMA Address Space.*

- namespace **L4Re**  
*L4Re C++ Interfaces.*

## 17.329 dma\_space

[Go to the documentation of this file.](#)

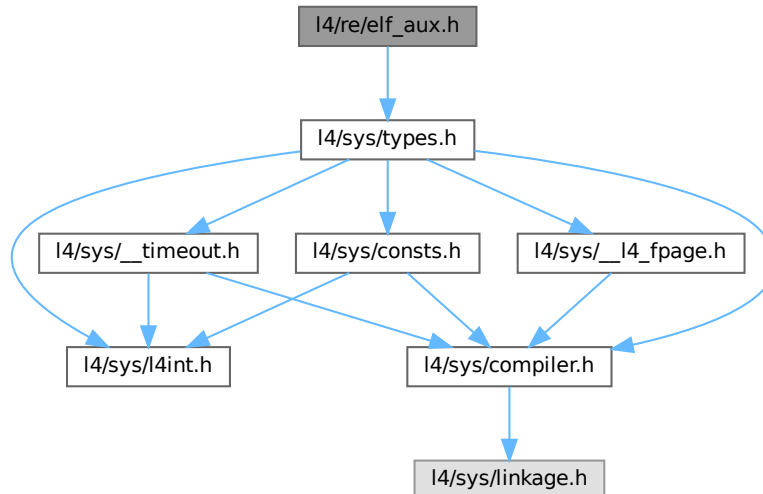
```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00006 /*
00007  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011
00012 #pragma once
00013
00014 #include <l4/sys/types.h>
00015 #include <l4/sys/l4int.h>
00016 #include <l4/sys/capability>
00017 #include <l4/re/dataspace>
00018 #include <l4/re/protocols.h>
00019 #include <l4/sys/cxx/types>
00020 #include <l4/sys/cxx/ipc_types>
00021 #include <l4/sys/cxx/ipc_iface>
00022
00023 namespace L4Re
00024 {
00025
00052 class Dma_space :
00053     public L4::Kobject_0t< Dma_space,
00054                          L4RE_PROTO_DMA_SPACE,
00055                          L4::Type_info::Demand_t<1> >
00056 {
00057 public:
00059     typedef l4_uint64_t Dma_addr;
00060
00064     enum Direction
00065     {
00066         Bidirectional,
00067         To_device,
00068         From_device,
00069         None
00070     };
00071
00076     enum Attribute
00077     {
00089         No_sync
00090     };
00091
00097     typedef L4::Types::Flags<Attribute> Attributes;
00098
00104     enum Space_attrib
00105     {
00112         Coherent,
00113
00118         Phys_space
00119     };
00120
00122     typedef L4::Types::Flags<Space_attrib> Space_attribs;
00123
00159     L4_INLINE_RPC(
00160         long, map, (L4::Ipc::Cap<L4Re::Dataspace> src,
00161                   L4Re::Dataspace::Offset offset,
00162                   L4::Ipc::In_out<l4_size_t *> size,
00163                   Attributes attrs, Direction dir,
00164                   Dma_addr *dma_addr));
00165
00176     L4_INLINE_RPC(
00177         long, unmap, (Dma_addr dma_addr,
00178                      l4_size_t size, Attributes attrs, Direction dir));
00179
00202     L4_INLINE_RPC(
00203         long, associate, (L4::Ipc::Opt<L4::Ipc::Cap<L4::Task> > dma_task,
00204                          Space_attribs attr),
00205         L4::Ipc::Call_t<L4_CAP_FPAGE_RW>);
00206
00218     L4_INLINE_RPC(
00219         long, disassociate, (),
00220         L4::Ipc::Call_t<L4_CAP_FPAGE_RW>);
00221
00222     typedef L4::Typeid::Rpcs<map_t, unmap_t, associate_t, disassociate_t> Rpcs;
00223 };
00224
00225 }
```

## 17.330 l4/re/elf\_aux.h File Reference

Auxiliary information for binaries.

```
#include <l4/sys/types.h>
Include dependency graph for elf_aux.h:
```



### Data Structures

- struct `l4re_elf_aux_t`  
*Generic header for each auxiliary vector element.*
- struct `l4re_elf_aux_vma_t`  
*Auxiliary vector element for a reserved virtual memory area.*
- struct `l4re_elf_aux_mword_t`  
*Auxiliary vector element for a single unsigned data word.*

### Macros

- `#define L4RE_ELF_AUX_ELEM` `const __attribute__((used, section(".ro.l4re_elf_aux"), aligned(sizeof(l4_umword_t))))`  
*Define an auxiliary vector element.*
- `#define L4RE_ELF_AUX_ELEM_T` `(type, id, tag, val...)`  
*Define an auxiliary vector element.*

### Typedefs

- `typedef struct l4re_elf_aux_t l4re_elf_aux_t`  
*Generic header for each auxiliary vector element.*
- `typedef struct l4re_elf_aux_vma_t l4re_elf_aux_vma_t`  
*Auxiliary vector element for a reserved virtual memory area.*
- `typedef struct l4re_elf_aux_mword_t l4re_elf_aux_mword_t`  
*Auxiliary vector element for a single unsigned data word.*

## Enumerations

- enum {  
[L4RE\\_ELF\\_AUX\\_T\\_NONE](#) = 0 , [L4RE\\_ELF\\_AUX\\_T\\_VMA](#) , [L4RE\\_ELF\\_AUX\\_T\\_STACK\\_SIZE](#) ,  
[L4RE\\_ELF\\_AUX\\_T\\_STACK\\_ADDR](#) ,  
[L4RE\\_ELF\\_AUX\\_T\\_KIP\\_ADDR](#) , [L4RE\\_ELF\\_AUX\\_T\\_EX\\_REGS\\_FLAGS](#) }

### 17.330.1 Detailed Description

Auxiliary information for binaries.

Definition in file [elf\\_aux.h](#).

## 17.331 [elf\\_aux.h](#)

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007  *      economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00013 #include <l4/sys/types.h>
00014
00015
00028
00041 #define L4RE_ELF_AUX_ELEM const __attribute__((used, section(".rol4re_elf_aux"),
    aligned(sizeof(l4_umword_t))))
00042
00056 #define L4RE_ELF_AUX_ELEM_T(type, id, tag, val...) \
00057     static L4RE_ELF_AUX_ELEM type id = {tag, sizeof(type), val}
00058
00059 enum
00060 {
00064     L4RE\_ELF\_AUX\_T\_NONE = 0,
00065
00069     L4RE\_ELF\_AUX\_T\_VMA,
00070
00075     L4RE\_ELF\_AUX\_T\_STACK\_SIZE,
00076
00081     L4RE\_ELF\_AUX\_T\_STACK\_ADDR,
00082
00087     L4RE\_ELF\_AUX\_T\_KIP\_ADDR,
00088
00092     L4RE\_ELF\_AUX\_T\_EX\_REGS\_FLAGS,
00093 };
00094
00098 typedef struct l4re_elf_aux_t
00099 {
00100     l4_umword_t type;
00101     l4_umword_t length;
00102 } l4re_elf_aux_t;
00103
00107 typedef struct l4re_elf_aux_vma_t
00108 {
00109     l4_umword_t type;
00110     l4_umword_t length;
00111     l4_umword_t start;
00112     l4_umword_t end;
00113 } l4re_elf_aux_vma_t;
00114
00118 typedef struct l4re_elf_aux_mword_t
00119 {
00120     l4_umword_t type;
00121     l4_umword_t length;
00122     l4_umword_t value;
00123 } l4re_elf_aux_mword_t;
00124

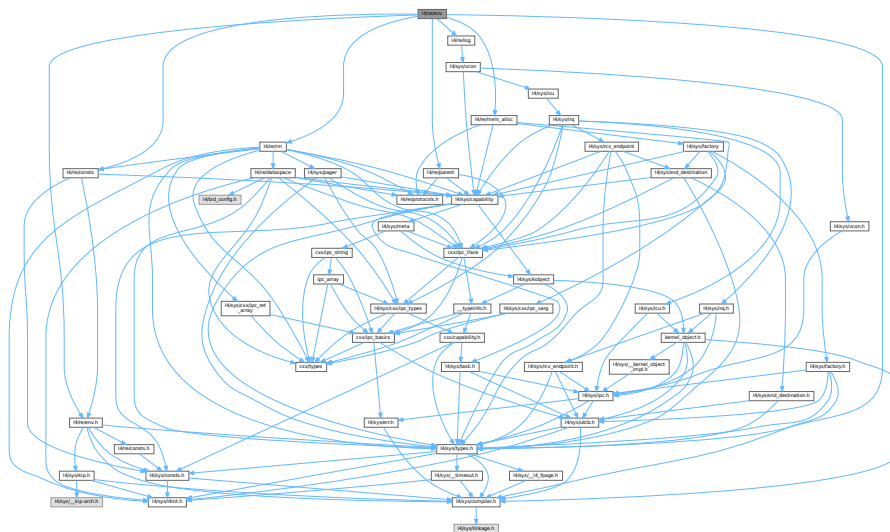
```

## 17.332 I4/re/env File Reference

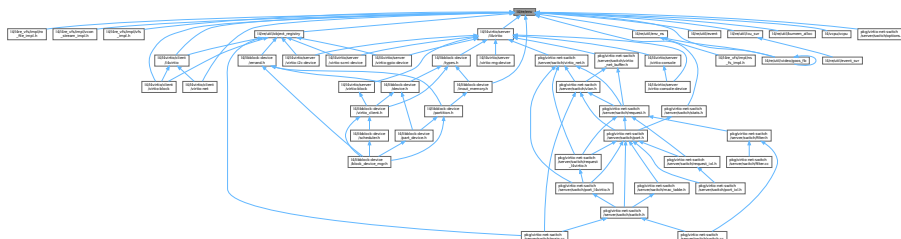
Environment interface.

```
#include <l4/sys/types.h>
#include <l4/re/rm>
#include <l4/re/parent>
#include <l4/re/mem_alloc>
#include <l4/re/log>
#include <l4/re/consts>
#include <l4/re/env.h>
```

Include dependency graph for env:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `L4Re::Env`  
*C++ interface of the initial environment that is provided to an `L4` task.*

## Namespaces

- namespace **L4**  
*L4 low-level kernel interface.*
- namespace **L4Re**  
*L4Re C++ Interfaces.*

## 17.332.1 Detailed Description

Environment interface.

Definition in file [env](#).

## 17.333 env

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00010 *      Björn Döbel <doebel@os.inf.tu-dresden.de>
00011 *      economic rights: Technische Universität Dresden (Germany)
00012 *
00013 * License: see LICENSE.spdx (in this directory or the directories above)
00014 */
00015 #pragma once
00016
00017 #include <l4/sys/types.h>
00018
00019 #include <l4/re/rm>
00020 #include <l4/re/parent>
00021 #include <l4/re/mem_alloc>
00022 #include <l4/re/log>
00023 #include <l4/re/consts>
00024
00025 #include <l4/re/env.h>
00026
00027 namespace L4 {
00028 class Scheduler;
00029 }
00030
00034 namespace L4Re
00035 {
00036     class Itas;
00037     struct Dbg_events;
00038
00078     class L4_EXPORT Env
00079     {
00080     private:
00081         l4re_env_t _env;
00082     public:
00083
00087         typedef l4re_env_cap_entry_t Cap_entry;
00088
00096         static Env const *env() noexcept
00097         { return reinterpret_cast<Env*>(l4re_global_env); }
00098
00103         L4::Cap<Parent> parent() const noexcept
00104         { return L4::Cap<Parent>(_env.parent); }
00109         L4::Cap<Mem_alloc> mem_alloc() const noexcept
00110         { return L4::Cap<Mem_alloc>(_env.mem_alloc); }
00114         L4::Cap<L4::Factory> user_factory() const noexcept
00115         { return L4::Cap<L4::Factory>(_env.mem_alloc); }
00120         L4::Cap<Rm> rm() const noexcept
00121         { return L4::Cap<Rm>(_env.rm); }
00126         L4::Cap<Log> log() const noexcept
00127         { return L4::Cap<Log>(_env.log); }
00132         L4::Cap<L4::Thread> main_thread() const noexcept
00133         { return L4::Cap<L4::Thread>(_env.main_thread); }
00138         L4::Cap<L4::Task> task() const noexcept
00139         { return L4::Cap<L4::Task>(L4RE_THIS_TASK_CAP); }
00144         L4::Cap<L4::Factory> factory() const noexcept
00145         { return L4::Cap<L4::Factory>(_env.factory); }
00152         l4_cap_idx_t first_free_cap() const noexcept
00153         { return _env.first_free_cap; }
00158         l4_fpage_t utcb_area() const noexcept
00159         { return _env.utcb_area; }
00167         l4_addr_t first_free_utcb() const noexcept
00168         { return _env.first_free_utcb; }
00169
00174         Cap_entry const *initial_caps() const noexcept
00175         { return _env.caps; }
00176
00185         Cap_entry const *get(char const *name, unsigned l) const noexcept

```



```

00186     { return l4re_env_get_cap_l(name, l, &_env); }
00187
00196     template< typename T >
00197     L4::Cap<T> get_cap(char const *name, unsigned l) const noexcept
00198     {
00199         if (Cap_entry const *e = get(name, l))
00200             return L4::Cap<T>(e->cap);
00201
00202         return L4::Cap<T>(-L4_ENOENT);
00203     }
00204
00211     template< typename T >
00212     L4::Cap<T> get_cap(char const *name) const noexcept
00213     { return get_cap<T>(name, __builtin_strlen(name)); }
00214
00219     void parent(L4::Cap<Parent> const &c) noexcept
00220     { _env.parent = c.cap(); }
00225     void mem_alloc(L4::Cap<Mem_alloc> const &c) noexcept
00226     { _env.mem_alloc = c.cap(); }
00231     void rm(L4::Cap<Rm> const &c) noexcept
00232     { _env.rm = c.cap(); }
00237     void log(L4::Cap<Log> const &c) noexcept
00238     { _env.log = c.cap(); }
00243     void main_thread(L4::Cap<L4::Thread> const &c) noexcept
00244     { _env.main_thread = c.cap(); }
00249     void factory(L4::Cap<L4::Factory> const &c) noexcept
00250     { _env.factory = c.cap(); }
00255     void first_free_cap(l4_cap_idx_t c) noexcept
00256     { _env.first_free_cap = c; }
00261     void utcb_area(l4_fpage_t utcbs) noexcept
00262     { _env.utcb_area = utcbs; }
00267     void first_free_utcb(l4_addr_t u) noexcept
00268     { _env.first_free_utcb = u; }
00269
00275     L4::Cap<L4::Scheduler> scheduler() const noexcept
00276     { return L4::Cap<L4::Scheduler>(_env.scheduler); }
00277
00282     void scheduler(L4::Cap<L4::Scheduler> const &c) noexcept
00283     { _env.scheduler = c.cap(); }
00284
00294     L4::Cap<Itas> itas() const noexcept
00295     { return L4::Cap<Itas>(_env.itas); }
00296
00301     void itas(L4::Cap<Itas> const &c) noexcept
00302     { _env.itas = c.cap(); }
00303
00310     L4::Cap<Dbg_events> dbg_events() const noexcept
00311     { return L4::Cap<Dbg_events>(_env.dbg_events); }
00312
00320     void dbg_events(L4::Cap<Dbg_events> const &dbg_events) noexcept
00321     { _env.dbg_events = dbg_events.cap(); }
00322
00327     void initial_caps(Cap_entry *first) noexcept
00328     { _env.caps = first; }
00329 };
00330 };

```

## 17.334 l4/re/env.h File Reference

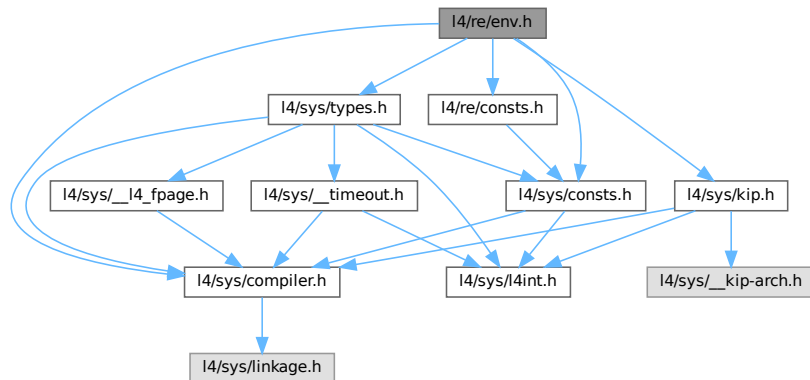
Environment interface.

```

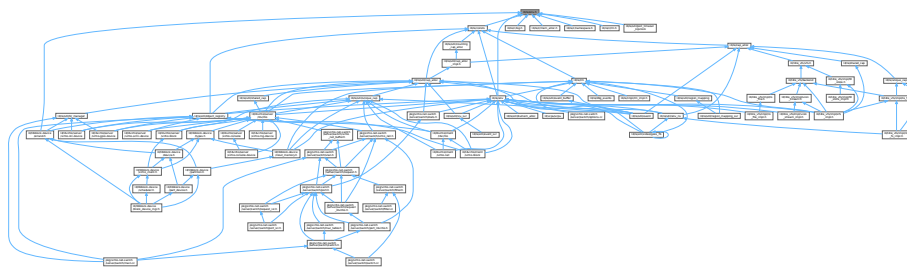
#include <l4/sys/consts.h>
#include <l4/sys/types.h>
#include <l4/sys/kip.h>
#include <l4/sys/compiler.h>
#include <l4/re/consts.h>

```

Include dependency graph for env.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [l4re\\_env\\_cap\\_entry\\_t](#)  
*Entry in the [L4Re](#) environment array for the named initial objects.*
- struct [l4re\\_env\\_t](#)  
*Initial environment data structure.*

## Typedefs

- typedef struct [l4re\\_env\\_cap\\_entry\\_t](#) [l4re\\_env\\_cap\\_entry\\_t](#)  
*Entry in the [L4Re](#) environment array for the named initial objects.*
- typedef struct [l4re\\_env\\_t](#) [l4re\\_env\\_t](#)  
*Initial environment data structure.*

## Functions

- [l4re\\_env\\_t](#) \* [l4re\\_env](#) (void) [L4\\_NOTHROW](#)  
*Get [L4Re](#) initial environment.*
- [l4\\_kernel\\_info\\_t](#) const \* [l4re\\_kip](#) (void) [L4\\_NOTHROW](#)  
*Get Kernel Info Page.*

- [l4\\_cap\\_idx\\_t l4re\\_env\\_get\\_cap](#) (char const \*name) [L4\\_NOTHROW](#)  
*Get the capability selector for the object named name.*
- [l4\\_cap\\_idx\\_t l4re\\_env\\_get\\_cap\\_e](#) (char const \*name, [l4re\\_env\\_t](#) const \*) [L4\\_NOTHROW](#)  
*Get the capability selector for the object named name.*
- [l4re\\_env\\_cap\\_entry\\_t](#) const \* [l4re\\_env\\_get\\_cap\\_l](#) (char const \*name, unsigned l, [l4re\\_env\\_t](#) const \*) [L4\\_NOTHROW](#)  
*Get the full [l4re\\_env\\_cap\\_entry\\_t](#) for the object named name.*

### 17.334.1 Detailed Description

Environment interface.

Definition in file [env.h](#).

### 17.334.2 Typedef Documentation

#### 17.334.2.1 l4re\_env\_t

```
typedef struct l4re_env_t l4re_env_t
```

Initial environment data structure.

See also

[Initial environment](#)

## 17.335 env.h

[Go to the documentation of this file.](#)

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 #include <l4/sys/consts.h>
00015 #include <l4/sys/types.h>
00016 #include <l4/sys/kip.h>
00017 #include <l4/sys/compiler.h>
00018
00019 #include <l4/re/consts.h>
00020
00034
00039 typedef struct l4re_env_cap_entry_t
00040 {
00044     l4_cap_idx_t cap;
00045
00050     l4_umword_t flags;
00051
00055     char name[16];
00056 #ifdef __cplusplus
00057
00061     l4re_env_cap_entry_t() L4\_NOTHROW : cap(L4_INVALID_CAP), flags(~0UL) {}
00062
00070     l4re_env_cap_entry_t(char const *n, l4_cap_idx_t c, l4_umword_t f = 0) L4\_NOTHROW
00071     : cap(c), flags(f)
00072     {
```

```

00073     for (unsigned i = 0; n && i < sizeof(name); ++i, ++n)
00074     {
00075         name[i] = *n;
00076     if (!*n)
00077         break;
00078     }
00079 }
00080
00081 static bool is_valid_name(char const *n) L4_NOTHROW
00082 {
00083     for (unsigned i = 0; *n; ++i, ++n)
00084         if (i > sizeof(name))
00085             return false;
00086
00087     return true;
00088 }
00089 #endif
00090 } l4re_env_cap_entry_t;
00091
00092
00093 typedef struct l4re_env_t
00094 {
00095     l4_cap_idx_t parent;
00096     l4_cap_idx_t rm;
00097     l4_cap_idx_t mem_alloc;
00098     l4_cap_idx_t log;
00099     l4_cap_idx_t main_thread;
00100     l4_cap_idx_t factory;
00101     l4_cap_idx_t scheduler;
00102     l4_cap_idx_t itas;
00103     l4_cap_idx_t dbg_events;
00104     l4_cap_idx_t first_free_cap;
00105     l4_fpage_t utcb_area;
00106     l4_addr_t first_free_utcb;
00107     l4re_env_cap_entry_t *caps;
00108 } l4re_env_t;
00109
00110 extern l4re_env_t *l4re_global_env;
00111
00112
00113 L4_INLINE l4re_env_t *l4re_env(void) L4_NOTHROW;
00114
00115 /*
00116  * FIXME: this seems to be at the wrong place here
00117  */
00118 L4_INLINE l4_kernel_info_t const *l4re_kip(void) L4_NOTHROW;
00119
00120
00121 L4_INLINE l4_cap_idx_t
00122 l4re_env_get_cap(char const *name) L4_NOTHROW;
00123
00124 L4_INLINE l4_cap_idx_t
00125 l4re_env_get_cap_e(char const *name, l4re_env_t const *e) L4_NOTHROW;
00126
00127 L4_INLINE l4re_env_cap_entry_t const *
00128 l4re_env_get_cap_l(char const *name, unsigned l, l4re_env_t const *e) L4_NOTHROW;
00129
00130 L4_INLINE
00131 l4re_env_t *l4re_env(void) L4_NOTHROW
00132 { return l4re_global_env; }
00133
00134 L4_INLINE
00135 l4_kernel_info_t const *l4re_kip(void) L4_NOTHROW
00136 { return l4_kip(); }
00137
00138 L4_INLINE l4re_env_cap_entry_t const *
00139 l4re_env_get_cap_l(char const *name, unsigned l, l4re_env_t const *e) L4_NOTHROW
00140 {
00141     l4re_env_cap_entry_t const *c = e->caps;
00142     for (; c && c->flags != ~0UL; ++c)
00143     {
00144         unsigned i;
00145         for (i = 0;
00146              i < sizeof(c->name) && i < l && c->name[i] && name[i] && name[i] == c->name[i];
00147              ++i)
00148             ;
00149
00150         if (i == l && (i == sizeof(c->name) || !c->name[i]))
00151             return c;
00152     }
00153     return NULL;
00154 }
00155
00156 L4_INLINE l4_cap_idx_t
00157 l4re_env_get_cap_e(char const *name, l4re_env_t const *e) L4_NOTHROW
00158 {
00159     unsigned l;

```

```

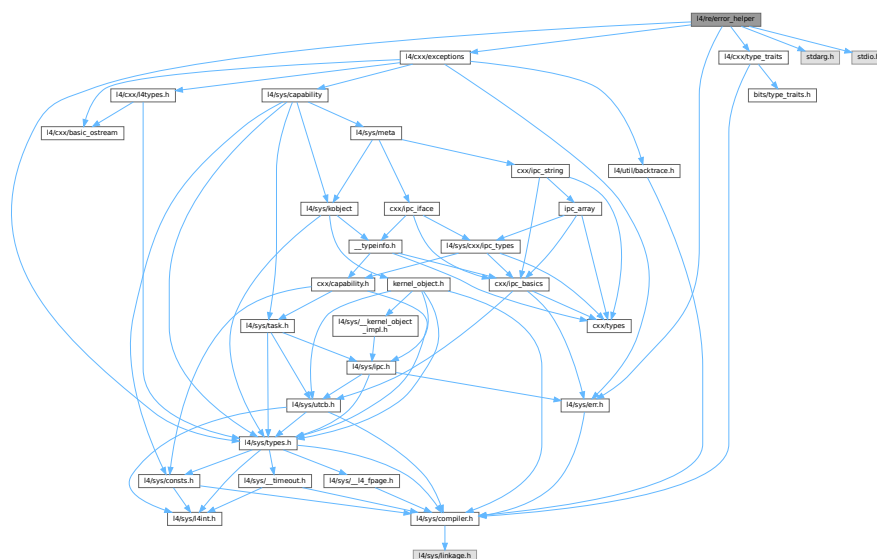
00210     14re_env_cap_entry_t const *r;
00211     for (l = 0; name[l]; ++l) ;
00212     r = 14re_env_get_cap_l(name, l, e);
00213     if (r)
00214         return r->cap;
00215
00216     return L4_INVALID_CAP;
00217 }
00218
00219 L4_INLINE l4_cap_idx_t
00220 14re_env_get_cap(char const *name) L4_NOTHROW
00221 { return 14re_env_get_cap_e(name, 14re_env()); }
00222

```

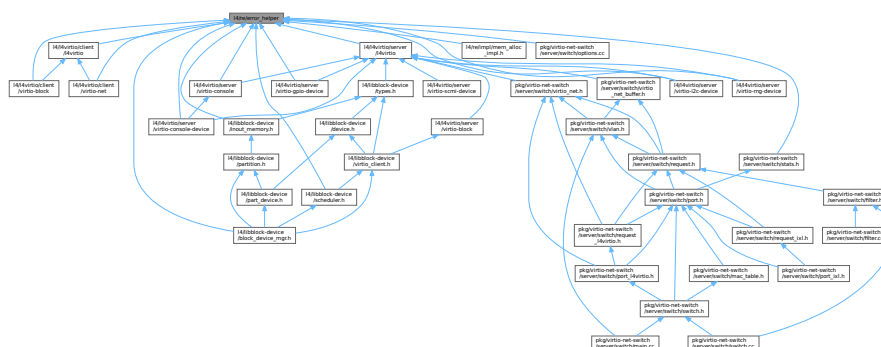
Error helper.

```
#include <l4/sys/types.h>
#include <l4/cxx/exceptions>
#include <l4/cxx/type_traits>
#include <l4/sys/err.h>
#include <stdarg.h>
#include <stdio.h>
```

Include dependency graph for error helper:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [L4Re](#)  
*L4Re C++ Interfaces.*

## Functions

- void [L4Re::throw\\_error](#) (long err, char const \*extra="")  
*Generate C++ exception.*
- long [L4Re::chksys](#) (long err, char const \*extra="", long ret=0)  
*Generate C++ exception on error.*
- long [L4Re::chksys](#) ([l4\\_msgtag\\_t](#) const &t, char const \*extra="", [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)(), long ret=0)  
*Generate C++ exception on error.*
- long [L4Re::chksys](#) ([l4\\_msgtag\\_t](#) const &t, [l4\\_utcb\\_t](#) \*utcb, char const \*extra="")  
*Generate C++ exception on error.*
- template<typename T>  
T [L4Re::chkcap](#) (T &&cap, char const \*extra="", long err=[L4\\_ENOMEM](#))  
*Check for valid capability or raise C++ exception.*
- [l4\\_msgtag\\_t](#) [L4Re::chkipc](#) ([l4\\_msgtag\\_t](#) tag, char const \*extra="", [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)())  
*Test a message tag for IPC errors.*

### 17.336.1 Detailed Description

Error helper.

Definition in file [error\\_helper](#).

## 17.337 error\_helper

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -- Mode: C++ --
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00005  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00006  *      economic rights: Technische Universität Dresden (Germany)
00007  *
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010 #pragma once
00011
00012 #include <l4/sys/types.h>
00013 #include <l4/cxx/exceptions>
00014 #include <l4/cxx/type_traits>
00015 #include <l4/sys/err.h>
00016
00017 #include <stdarg.h>
00018 #include <stdio.h>
00019
00020 namespace L4Re {
00021 #ifdef __EXCEPTIONS
00022 [[noreturn]] inline void throw_error(long err, char const *extra = "")
00023 {
00024     switch (err)
00025     {
00026     case -L4_ENOENT: throw (L4::Element_not_found(extra));
00027     case -L4_ENOMEM: throw (L4::Out_of_memory(extra));
00028     case -L4_EEXIST: throw (L4::Element_already_exists(extra));
00029     case -L4_ERANGE: throw (L4::Bounds_error(extra));
00030     }
00031 }

```

```

00045     default: throw (L4::Runtime_error(err, extra));
00046     }
00047 }
00048
00049 [[noreturn]] inline void throw_error_fmt(long err, char const *const fmt, ...)
00050     __attribute__((format(printf, 2, 3)));
00051 [[noreturn]] inline void throw_error_fmt(long err, char const *const fmt, ...)
00052 {
00053     char extra[80];
00054     va_list argp;
00055     va_start(argp, fmt);
00056     vsnprintf(extra, sizeof(extra), fmt, argp);
00057     va_end(argp);
00058     throw_error(err, extra);
00059 }
00060
00071 inline
00072 long chksys(long err, char const *extra = "", long ret = 0)
00073 {
00074     if (L4_UNLIKELY(err < 0))
00075         throw_error(ret ? ret : err, extra);
00076
00077     return err;
00078 }
00079
00092 inline
00093 long chksys(l4_msgtag_t const &t, char const *extra = "",
00094             l4_utcb_t *utcb = l4_utcb(), long ret = 0)
00095 {
00096     if (L4_UNLIKELY(t.has_error()))
00097         throw_error(ret ? ret : l4_error_u(t, utcb), extra);
00098     else if (L4_UNLIKELY(t.label() < 0))
00099         throw_error(ret ? ret: t.label(), extra);
00100
00101     return t.label();
00102 }
00103
00115 inline
00116 long chksys(l4_msgtag_t const &t, l4_utcb_t *utcb, char const *extra = "")
00117 { return chksys(t, extra, utcb); }
00118
00119 #if 0
00120 inline
00121 long chksys(long ret, long err, char const *extra = "")
00122 {
00123     if (L4_UNLIKELY(ret < 0))
00124         throw_error(err, extra);
00125
00126     return ret;
00127 }
00128 #endif
00129
00146 template<typename T>
00147 inline
00148 #if __cplusplus >= 201103L
00149 T chkcap(T &&cap, char const *extra = "", long err = -L4_ENOMEM)
00150 #else
00151 T chkcap(T cap, char const *extra = "", long err = -L4_ENOMEM)
00152 #endif
00153 {
00154     if (L4_UNLIKELY(!cap.is_valid()))
00155         throw_error(err ? err : cap.invalid_cap_error(), extra);
00156
00157 #if __cplusplus >= 201103L
00158     return cxx::forward<T>(cap);
00159 #else
00160     return cap;
00161 #endif
00162 }
00163
00178 inline
00179 l4_msgtag_t
00180 chkipc(l4_msgtag_t tag, char const *extra = "",
00181        l4_utcb_t *utcb = l4_utcb())
00182 {
00183     if (L4_UNLIKELY(tag.has_error()))
00184         chksys(l4_error_u(tag, utcb), extra);
00185
00186     return tag;
00187 }
00188 #endif
00189
00190 }

```

## 17.338 event-sys.h

```

00001 /*
00002  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009 namespace L4Re
00010 {
00011     namespace Event_
00012     {
00013         enum Opcodes
00014         {
00015             Get, Get_num_streams, Get_stream_info, Get_stream_info_for_id,
00016             Get_axis_info, Get_stream_state_for_id
00017         };
00018     };
00019 };
00020 
```

## 17.339 event\_enums.h

```

00001 #pragma once
00002
00003 /*
00004  *
00005  *
00006  * Constants for L4Re events ...
00007  */
00008
00009
00010 enum L4Re_events_key
00011 {
00012     L4RE_KEY_RESERVED           = 0,
00013     L4RE_KEY_ESC                = 1,
00014     L4RE_KEY_1                  = 2,
00015     L4RE_KEY_2                  = 3,
00016     L4RE_KEY_3                  = 4,
00017     L4RE_KEY_4                  = 5,
00018     L4RE_KEY_5                  = 6,
00019     L4RE_KEY_6                  = 7,
00020     L4RE_KEY_7                  = 8,
00021     L4RE_KEY_8                  = 9,
00022     L4RE_KEY_9                  = 10,
00023     L4RE_KEY_0                  = 11,
00024     L4RE_KEY_MINUS              = 12,
00025     L4RE_KEY_EQUAL              = 13,
00026     L4RE_KEY_BACKSPACE         = 14,
00027     L4RE_KEY_TAB                = 15,
00028     L4RE_KEY_Q                  = 16,
00029     L4RE_KEY_W                  = 17,
00030     L4RE_KEY_E                  = 18,
00031     L4RE_KEY_R                  = 19,
00032     L4RE_KEY_T                  = 20,
00033     L4RE_KEY_Y                  = 21,
00034     L4RE_KEY_U                  = 22,
00035     L4RE_KEY_I                  = 23,
00036     L4RE_KEY_O                  = 24,
00037     L4RE_KEY_P                  = 25,
00038     L4RE_KEY_LEFTBRACE         = 26,
00039     L4RE_KEY_RIGHTBRACE        = 27,
00040     L4RE_KEY_ENTER              = 28,
00041     L4RE_KEY_LEFTCTRL          = 29,
00042     L4RE_KEY_A                  = 30,
00043     L4RE_KEY_S                  = 31,
00044     L4RE_KEY_D                  = 32,
00045     L4RE_KEY_F                  = 33,
00046     L4RE_KEY_G                  = 34,
00047     L4RE_KEY_H                  = 35,
00048     L4RE_KEY_J                  = 36,
00049     L4RE_KEY_K                  = 37,
00050     L4RE_KEY_L                  = 38,
00051     L4RE_KEY_SEMICOLON         = 39,
00052     L4RE_KEY_APOSTROPHE        = 40,
00053     L4RE_KEY_GRAVE             = 41,
00054     L4RE_KEY_LEFTSHIFT         = 42,
00055     L4RE_KEY_BACKSLASH         = 43,
00056     L4RE_KEY_Z                  = 44,
00057     L4RE_KEY_X                  = 45,
00058     L4RE_KEY_C                  = 46,

```



```
00059 L4RE_KEY_V = 47,
00060 L4RE_KEY_B = 48,
00061 L4RE_KEY_N = 49,
00062 L4RE_KEY_M = 50,
00063 L4RE_KEY_COMMA = 51,
00064 L4RE_KEY_DOT = 52,
00065 L4RE_KEY_SLASH = 53,
00066 L4RE_KEY_RIGHTSHIFT = 54,
00067 L4RE_KEY_KPASTERISK = 55,
00068 L4RE_KEY_LEFTALT = 56,
00069 L4RE_KEY_SPACE = 57,
00070 L4RE_KEY_CAPSLOCK = 58,
00071 L4RE_KEY_F1 = 59,
00072 L4RE_KEY_F2 = 60,
00073 L4RE_KEY_F3 = 61,
00074 L4RE_KEY_F4 = 62,
00075 L4RE_KEY_F5 = 63,
00076 L4RE_KEY_F6 = 64,
00077 L4RE_KEY_F7 = 65,
00078 L4RE_KEY_F8 = 66,
00079 L4RE_KEY_F9 = 67,
00080 L4RE_KEY_F10 = 68,
00081 L4RE_KEY_NUMLOCK = 69,
00082 L4RE_KEY_SCROLLLOCK = 70,
00083 L4RE_KEY_KP7 = 71,
00084 L4RE_KEY_KP8 = 72,
00085 L4RE_KEY_KP9 = 73,
00086 L4RE_KEY_KPMINUS = 74,
00087 L4RE_KEY_KP4 = 75,
00088 L4RE_KEY_KP5 = 76,
00089 L4RE_KEY_KP6 = 77,
00090 L4RE_KEY_KPPLUS = 78,
00091 L4RE_KEY_KP1 = 79,
00092 L4RE_KEY_KP2 = 80,
00093 L4RE_KEY_KP3 = 81,
00094 L4RE_KEY_KP0 = 82,
00095 L4RE_KEY_KPDOT = 83,
00096 L4RE_KEY_ZENKAKUHANKAKU = 85,
00097 L4RE_KEY_102ND = 86,
00098 L4RE_KEY_F11 = 87,
00099 L4RE_KEY_F12 = 88,
00100 L4RE_KEY_RO = 89,
00101 L4RE_KEY_KATAKANA = 90,
00102 L4RE_KEY_HIRAGANA = 91,
00103 L4RE_KEY_HENKAN = 92,
00104 L4RE_KEY_KATAKANAHIRAGANA = 93,
00105 L4RE_KEY_MUHENKAN = 94,
00106 L4RE_KEY_KPJPCOMMA = 95,
00107 L4RE_KEY_KPENTER = 96,
00108 L4RE_KEY_RIGHTCTRL = 97,
00109 L4RE_KEY_KPSLASH = 98,
00110 L4RE_KEY_SYSRQ = 99,
00111 L4RE_KEY_RIGHTALT = 100,
00112 L4RE_KEY_LINEFEED = 101,
00113 L4RE_KEY_HOME = 102,
00114 L4RE_KEY_UP = 103,
00115 L4RE_KEY_PAGEUP = 104,
00116 L4RE_KEY_LEFT = 105,
00117 L4RE_KEY_RIGHT = 106,
00118 L4RE_KEY_END = 107,
00119 L4RE_KEY_DOWN = 108,
00120 L4RE_KEY_PAGEDOWN = 109,
00121 L4RE_KEY_INSERT = 110,
00122 L4RE_KEY_DELETE = 111,
00123 L4RE_KEY_MACRO = 112,
00124 L4RE_KEY_MUTE = 113,
00125 L4RE_KEY_VOLUMEDOWN = 114,
00126 L4RE_KEY_VOLUMEUP = 115,
00127 L4RE_KEY_POWER = 116,
00128 L4RE_KEY_KPEQUAL = 117,
00129 L4RE_KEY_KPPLUSMINUS = 118,
00130 L4RE_KEY_PAUSE = 119,
00131 L4RE_KEY_KPCOMMA = 121,
00132 L4RE_KEY_HANGEUL = 122,
00133 L4RE_KEY_HANGUEL = L4RE_KEY_HANGEUL,
00134 L4RE_KEY_HANJA = 123,
00135 L4RE_KEY_YEN = 124,
00136 L4RE_KEY_LEFTMETA = 125,
00137 L4RE_KEY_RIGHTMETA = 126,
00138 L4RE_KEY_COMPOSE = 127,
00139 L4RE_KEY_STOP = 128,
00140 L4RE_KEY_AGAIN = 129,
00141 L4RE_KEY_PROPS = 130,
00142 L4RE_KEY_UNDO = 131,
00143 L4RE_KEY_FRONT = 132,
00144 L4RE_KEY_COPY = 133,
00145 L4RE_KEY_OPEN = 134,
```

00146	L4RE_KEY_PASTE	= 135,
00147	L4RE_KEY_FIND	= 136,
00148	L4RE_KEY_CUT	= 137,
00149	L4RE_KEY_HELP	= 138,
00150	L4RE_KEY_MENU	= 139,
00151	L4RE_KEY_CALC	= 140,
00152	L4RE_KEY_SETUP	= 141,
00153	L4RE_KEY_SLEEP	= 142,
00154	L4RE_KEY_WAKEUP	= 143,
00155	L4RE_KEY_FILE	= 144,
00156	L4RE_KEY_SENDFILE	= 145,
00157	L4RE_KEY_DELETEFILE	= 146,
00158	L4RE_KEY_XFER	= 147,
00159	L4RE_KEY_PROG1	= 148,
00160	L4RE_KEY_PROG2	= 149,
00161	L4RE_KEY_WWW	= 150,
00162	L4RE_KEY_MSDOS	= 151,
00163	L4RE_KEY_COFFEE	= 152,
00164	L4RE_KEY_DIRECTION	= 153,
00165	L4RE_KEY_CYCLEWINDOWS	= 154,
00166	L4RE_KEY_MAIL	= 155,
00167	L4RE_KEY_BOOKMARKS	= 156,
00168	L4RE_KEY_COMPUTER	= 157,
00169	L4RE_KEY_BACK	= 158,
00170	L4RE_KEY_FORWARD	= 159,
00171	L4RE_KEY_CLOSECD	= 160,
00172	L4RE_KEY_EJECTCD	= 161,
00173	L4RE_KEY_EJECTCLOSECD	= 162,
00174	L4RE_KEY_NEXTSONG	= 163,
00175	L4RE_KEY_PLAYPAUSE	= 164,
00176	L4RE_KEY_PREVIOUSSONG	= 165,
00177	L4RE_KEY_STOPCD	= 166,
00178	L4RE_KEY_RECORD	= 167,
00179	L4RE_KEY_REWIND	= 168,
00180	L4RE_KEY_PHONE	= 169,
00181	L4RE_KEY_ISO	= 170,
00182	L4RE_KEY_CONFIG	= 171,
00183	L4RE_KEY_HOMEPAGE	= 172,
00184	L4RE_KEY_REFRESH	= 173,
00185	L4RE_KEY_EXIT	= 174,
00186	L4RE_KEY_MOVE	= 175,
00187	L4RE_KEY_EDIT	= 176,
00188	L4RE_KEY_SCROLLUP	= 177,
00189	L4RE_KEY_SCROLLDOWN	= 178,
00190	L4RE_KEY_KPLEFTPAREN	= 179,
00191	L4RE_KEY_KPRIGHTPAREN	= 180,
00192	L4RE_KEY_NEW	= 181,
00193	L4RE_KEY_REDO	= 182,
00194	L4RE_KEY_F13	= 183,
00195	L4RE_KEY_F14	= 184,
00196	L4RE_KEY_F15	= 185,
00197	L4RE_KEY_F16	= 186,
00198	L4RE_KEY_F17	= 187,
00199	L4RE_KEY_F18	= 188,
00200	L4RE_KEY_F19	= 189,
00201	L4RE_KEY_F20	= 190,
00202	L4RE_KEY_F21	= 191,
00203	L4RE_KEY_F22	= 192,
00204	L4RE_KEY_F23	= 193,
00205	L4RE_KEY_F24	= 194,
00206	L4RE_KEY_PLAYCD	= 200,
00207	L4RE_KEY_PAUSECD	= 201,
00208	L4RE_KEY_PROG3	= 202,
00209	L4RE_KEY_PROG4	= 203,
00210	L4RE_KEY_SUSPEND	= 205,
00211	L4RE_KEY_CLOSE	= 206,
00212	L4RE_KEY_PLAY	= 207,
00213	L4RE_KEY_FASTFORWARD	= 208,
00214	L4RE_KEY_BASSBOOST	= 209,
00215	L4RE_KEY_PRINT	= 210,
00216	L4RE_KEY_HP	= 211,
00217	L4RE_KEY_CAMERA	= 212,
00218	L4RE_KEY_SOUND	= 213,
00219	L4RE_KEY_QUESTION	= 214,
00220	L4RE_KEY_EMAIL	= 215,
00221	L4RE_KEY_CHAT	= 216,
00222	L4RE_KEY_SEARCH	= 217,
00223	L4RE_KEY_CONNECT	= 218,
00224	L4RE_KEY_FINANCE	= 219,
00225	L4RE_KEY_SPORT	= 220,
00226	L4RE_KEY_SHOP	= 221,
00227	L4RE_KEY_ALTERASE	= 222,
00228	L4RE_KEY_CANCEL	= 223,
00229	L4RE_KEY_BRIGHTNESSDOWN	= 224,
00230	L4RE_KEY_BRIGHTNESSUP	= 225,
00231	L4RE_KEY_MEDIA	= 226,
00232	L4RE_KEY_SWITCHVIDEOMODE	= 227,

```
00233 L4RE_KEY_KBDILLUMTOGGLE = 228,
00234 L4RE_KEY_KBDILLUMDOWN = 229,
00235 L4RE_KEY_KBDILLUMUP = 230,
00236 L4RE_KEY_SEND = 231,
00237 L4RE_KEY_REPLY = 232,
00238 L4RE_KEY_FORWARDMAIL = 233,
00239 L4RE_KEY_SAVE = 234,
00240 L4RE_KEY_DOCUMENTS = 235,
00241 L4RE_KEY_UNKNOWN = 240,
00242 L4RE_KEY_OK = 0x160,
00243 L4RE_KEY_SELECT = 0x161,
00244 L4RE_KEY_GOTO = 0x162,
00245 L4RE_KEY_CLEAR = 0x163,
00246 L4RE_KEY_POWER2 = 0x164,
00247 L4RE_KEY_OPTION = 0x165,
00248 L4RE_KEY_INFO = 0x166,
00249 L4RE_KEY_TIME = 0x167,
00250 L4RE_KEY_VENDOR = 0x168,
00251 L4RE_KEY_ARCHIVE = 0x169,
00252 L4RE_KEY_PROGRAM = 0x16a,
00253 L4RE_KEY_CHANNEL = 0x16b,
00254 L4RE_KEY_FAVORITES = 0x16c,
00255 L4RE_KEY_EPG = 0x16d,
00256 L4RE_KEY_PVR = 0x16e,
00257 L4RE_KEY_MHP = 0x16f,
00258 L4RE_KEY_LANGUAGE = 0x170,
00259 L4RE_KEY_TITLE = 0x171,
00260 L4RE_KEY_SUBTITLE = 0x172,
00261 L4RE_KEY_ANGLE = 0x173,
00262 L4RE_KEY_ZOOM = 0x174,
00263 L4RE_KEY_MODE = 0x175,
00264 L4RE_KEY_KEYBOARD = 0x176,
00265 L4RE_KEY_SCREEN = 0x177,
00266 L4RE_KEY_PC = 0x178,
00267 L4RE_KEY_TV = 0x179,
00268 L4RE_KEY_TV2 = 0x17a,
00269 L4RE_KEY_VCR = 0x17b,
00270 L4RE_KEY_VCR2 = 0x17c,
00271 L4RE_KEY_SAT = 0x17d,
00272 L4RE_KEY_SAT2 = 0x17e,
00273 L4RE_KEY_CD = 0x17f,
00274 L4RE_KEY_TAPE = 0x180,
00275 L4RE_KEY_RADIO = 0x181,
00276 L4RE_KEY_TUNER = 0x182,
00277 L4RE_KEY_PLAYER = 0x183,
00278 L4RE_KEY_TEXT = 0x184,
00279 L4RE_KEY_DVD = 0x185,
00280 L4RE_KEY_AUX = 0x186,
00281 L4RE_KEY_MP3 = 0x187,
00282 L4RE_KEY_AUDIO = 0x188,
00283 L4RE_KEY_VIDEO = 0x189,
00284 L4RE_KEY_DIRECTORY = 0x18a,
00285 L4RE_KEY_LIST = 0x18b,
00286 L4RE_KEY_MEMO = 0x18c,
00287 L4RE_KEY_CALENDAR = 0x18d,
00288 L4RE_KEY_RED = 0x18e,
00289 L4RE_KEY_GREEN = 0x18f,
00290 L4RE_KEY_YELLOW = 0x190,
00291 L4RE_KEY_BLUE = 0x191,
00292 L4RE_KEY_CHANNELUP = 0x192,
00293 L4RE_KEY_CHANNELDOWN = 0x193,
00294 L4RE_KEY_FIRST = 0x194,
00295 L4RE_KEY_LAST = 0x195,
00296 L4RE_KEY_AB = 0x196,
00297 L4RE_KEY_NEXT = 0x197,
00298 L4RE_KEY_RESTART = 0x198,
00299 L4RE_KEY_SLOW = 0x199,
00300 L4RE_KEY_SHUFFLE = 0x19a,
00301 L4RE_KEY_BREAK = 0x19b,
00302 L4RE_KEY_PREVIOUS = 0x19c,
00303 L4RE_KEY_DIGITS = 0x19d,
00304 L4RE_KEY_TEEN = 0x19e,
00305 L4RE_KEY_TWEN = 0x19f,
00306 L4RE_KEY_DEL_EOL = 0x1c0,
00307 L4RE_KEY_DEL_EOS = 0x1c1,
00308 L4RE_KEY_INS_LINE = 0x1c2,
00309 L4RE_KEY_DEL_LINE = 0x1c3,
00310 L4RE_KEY_FN = 0x1d0,
00311 L4RE_KEY_FN_ESC = 0x1d1,
00312 L4RE_KEY_FN_F1 = 0x1d2,
00313 L4RE_KEY_FN_F2 = 0x1d3,
00314 L4RE_KEY_FN_F3 = 0x1d4,
00315 L4RE_KEY_FN_F4 = 0x1d5,
00316 L4RE_KEY_FN_F5 = 0x1d6,
00317 L4RE_KEY_FN_F6 = 0x1d7,
00318 L4RE_KEY_FN_F7 = 0x1d8,
00319 L4RE_KEY_FN_F8 = 0x1d9,
```

```
00320     L4RE_KEY_FN_F9           = 0x1da,
00321     L4RE_KEY_FN_F10          = 0x1db,
00322     L4RE_KEY_FN_F11          = 0x1dc,
00323     L4RE_KEY_FN_F12          = 0x1dd,
00324     L4RE_KEY_FN_1            = 0x1de,
00325     L4RE_KEY_FN_2            = 0x1df,
00326     L4RE_KEY_FN_D            = 0x1e0,
00327     L4RE_KEY_FN_E            = 0x1e1,
00328     L4RE_KEY_FN_F            = 0x1e2,
00329     L4RE_KEY_FN_S            = 0x1e3,
00330     L4RE_KEY_FN_B            = 0x1e4,
00331     L4RE_KEY_MAX              = 0x1ff,
00332 };
00333
00334 enum L4Re_events_rel
00335 {
00336     L4RE_REL_X                = 0x00,
00337     L4RE_REL_Y                = 0x01,
00338     L4RE_REL_Z                = 0x02,
00339     L4RE_REL_RX               = 0x03,
00340     L4RE_REL_RY               = 0x04,
00341     L4RE_REL_RZ               = 0x05,
00342     L4RE_REL_HWHEEL           = 0x06,
00343     L4RE_REL_DIAL             = 0x07,
00344     L4RE_REL_WHEEL            = 0x08,
00345     L4RE_REL_MISC             = 0x09,
00346     L4RE_REL_MAX              = 0x0f,
00347 };
00348
00349 enum L4Re_events_snd
00350 {
00351     L4RE_SND_CLICK            = 0x00,
00352     L4RE_SND_BELL             = 0x01,
00353     L4RE_SND_TONE             = 0x02,
00354     L4RE_SND_MAX              = 0x07,
00355 };
00356
00357 enum L4Re_events_rep
00358 {
00359     L4RE_REP_DELAY            = 0x00,
00360     L4RE_REP_PERIOD           = 0x01,
00361     L4RE_REP_MAX              = 0x01,
00362 };
00363
00364 enum L4Re_events_led
00365 {
00366     L4RE_LED_NUML             = 0x00,
00367     L4RE_LED_CAPSL            = 0x01,
00368     L4RE_LED_SCROLLL          = 0x02,
00369     L4RE_LED_COMPOSE          = 0x03,
00370     L4RE_LED_KANA             = 0x04,
00371     L4RE_LED_SLEEP            = 0x05,
00372     L4RE_LED_SUSPEND          = 0x06,
00373     L4RE_LED_MUTE             = 0x07,
00374     L4RE_LED_MISC             = 0x08,
00375     L4RE_LED_MAIL             = 0x09,
00376     L4RE_LED_CHARGING          = 0x0a,
00377     L4RE_LED_MAX              = 0x0f,
00378 };
00379
00380 enum L4Re_events_btn
00381 {
00382     L4RE_BTN_MISC              = 0x100,
00383     L4RE_BTN_0                 = 0x100,
00384     L4RE_BTN_1                 = 0x101,
00385     L4RE_BTN_2                 = 0x102,
00386     L4RE_BTN_3                 = 0x103,
00387     L4RE_BTN_4                 = 0x104,
00388     L4RE_BTN_5                 = 0x105,
00389     L4RE_BTN_6                 = 0x106,
00390     L4RE_BTN_7                 = 0x107,
00391     L4RE_BTN_8                 = 0x108,
00392     L4RE_BTN_9                 = 0x109,
00393     L4RE_BTN_MOUSE             = 0x110,
00394     L4RE_BTN_LEFT              = 0x110,
00395     L4RE_BTN_RIGHT             = 0x111,
00396     L4RE_BTN_MIDDLE            = 0x112,
00397     L4RE_BTN_SIDE              = 0x113,
00398     L4RE_BTN_EXTRA             = 0x114,
00399     L4RE_BTN_FORWARD           = 0x115,
00400     L4RE_BTN_BACK              = 0x116,
00401     L4RE_BTN_TASK              = 0x117,
00402     L4RE_BTN_JOYSTICK          = 0x120,
00403     L4RE_BTN_TRIGGER           = 0x120,
00404     L4RE_BTN_THUMB             = 0x121,
00405     L4RE_BTN_THUMB2            = 0x122,
00406     L4RE_BTN_TOP               = 0x123,
```

```

00407     L4RE_BTN_TOP2           = 0x124,
00408     L4RE_BTN_PINKIE         = 0x125,
00409     L4RE_BTN_BASE          = 0x126,
00410     L4RE_BTN_BASE2         = 0x127,
00411     L4RE_BTN_BASE3         = 0x128,
00412     L4RE_BTN_BASE4         = 0x129,
00413     L4RE_BTN_BASE5         = 0x12a,
00414     L4RE_BTN_BASE6         = 0x12b,
00415     L4RE_BTN_DEAD          = 0x12f,
00416     L4RE_BTN_GAMEPAD       = 0x130,
00417     L4RE_BTN_A              = 0x130,
00418     L4RE_BTN_B              = 0x131,
00419     L4RE_BTN_C              = 0x132,
00420     L4RE_BTN_X              = 0x133,
00421     L4RE_BTN_Y              = 0x134,
00422     L4RE_BTN_Z              = 0x135,
00423     L4RE_BTN_TL             = 0x136,
00424     L4RE_BTN_TR             = 0x137,
00425     L4RE_BTN_TL2           = 0x138,
00426     L4RE_BTN_TR2           = 0x139,
00427     L4RE_BTN_SELECT        = 0x13a,
00428     L4RE_BTN_START          = 0x13b,
00429     L4RE_BTN_MODE           = 0x13c,
00430     L4RE_BTN_THUMBL         = 0x13d,
00431     L4RE_BTN_THUMBR        = 0x13e,
00432     L4RE_BTN_DIGI           = 0x140,
00433     L4RE_BTN_TOOL_PEN       = 0x140,
00434     L4RE_BTN_TOOL_RUBBER    = 0x141,
00435     L4RE_BTN_TOOL_BRUSH     = 0x142,
00436     L4RE_BTN_TOOL_PENCIL   = 0x143,
00437     L4RE_BTN_TOOL_AIRBRUSH  = 0x144,
00438     L4RE_BTN_TOOL_FINGER    = 0x145,
00439     L4RE_BTN_TOOL_MOUSE     = 0x146,
00440     L4RE_BTN_TOOL_LENS      = 0x147,
00441     L4RE_BTN_TOUCH          = 0x14a,
00442     L4RE_BTN_STYLUS         = 0x14b,
00443     L4RE_BTN_STYLUS2       = 0x14c,
00444     L4RE_BTN_TOOL_DOUBLETAP = 0x14d,
00445     L4RE_BTN_TOOL_TRIPLETAP = 0x14e,
00446     L4RE_BTN_WHEEL          = 0x150,
00447     L4RE_BTN_GEAR_DOWN      = 0x150,
00448     L4RE_BTN_GEAR_UP        = 0x151,
00449 };
00450
00451 enum L4Re_events_sw
00452 {
00453     L4RE_SW_0   = 0x00,
00454     L4RE_SW_1   = 0x01,
00455     L4RE_SW_2   = 0x02,
00456     L4RE_SW_3   = 0x03,
00457     L4RE_SW_4   = 0x04,
00458     L4RE_SW_5   = 0x05,
00459     L4RE_SW_6   = 0x06,
00460     L4RE_SW_7   = 0x07,
00461     L4RE_SW_MAX = 0x0f,
00462 };
00463
00464 enum L4Re_events_ev
00465 {
00466     L4RE_EV_SYN      = 0x00,
00467     L4RE_EV_KEY      = 0x01,
00468     L4RE_EV_REL      = 0x02,
00469     L4RE_EV_ABS      = 0x03,
00470     L4RE_EV_MSC      = 0x04,
00471     L4RE_EV_SW       = 0x05,
00472     L4RE_EV_LED      = 0x11,
00473     L4RE_EV_SND      = 0x12,
00474     L4RE_EV_REP      = 0x14,
00475     L4RE_EV_FF       = 0x15,
00476     L4RE_EV_PWR      = 0x16,
00477     L4RE_EV_FF_STATUS = 0x17,
00478     L4RE_EV_WINDOW    = 0x18,
00479     L4RE_EV_PM        = 0x1e, // power management signals
00480     L4RE_EV_MAX       = 0x1f,
00481 };
00482
00483 enum L4Re_events_syn
00484 {
00485     L4RE_SYN_REPORT      = 0,
00486     L4RE_SYN_CONFIG      = 1,
00487     L4RE_SYN_MT_REPORT   = 2,
00488
00489     L4RE_SYN_STREAM_CFG   = 0x80,
00490 };
00491
00492 enum L4Re_stream_cfg
00493 {

```

```

00494     L4RE_SYN_STREAM_NEW    = 0,
00495     L4RE_SYN_STREAM_CLOSE = 1,
00496 };
00497
00498 enum L4Re_events_abs
00499 {
00500     L4RE_ABS_X                = 0x00,
00501     L4RE_ABS_Y                = 0x01,
00502     L4RE_ABS_Z                = 0x02,
00503     L4RE_ABS_RX               = 0x03,
00504     L4RE_ABS_RY               = 0x04,
00505     L4RE_ABS_RZ               = 0x05,
00506     L4RE_ABS_THROTTLE        = 0x06,
00507     L4RE_ABS_RUDDER           = 0x07,
00508     L4RE_ABS_WHEEL            = 0x08,
00509     L4RE_ABS_GAS              = 0x09,
00510     L4RE_ABS_BRAKE            = 0x0a,
00511     L4RE_ABS_HAT0X            = 0x10,
00512     L4RE_ABS_HAT0Y            = 0x11,
00513     L4RE_ABS_HAT1X            = 0x12,
00514     L4RE_ABS_HAT1Y            = 0x13,
00515     L4RE_ABS_HAT2X            = 0x14,
00516     L4RE_ABS_HAT2Y            = 0x15,
00517     L4RE_ABS_HAT3X            = 0x16,
00518     L4RE_ABS_HAT3Y            = 0x17,
00519     L4RE_ABS_PRESSURE          = 0x18,
00520     L4RE_ABS_DISTANCE         = 0x19,
00521     L4RE_ABS_TILT_X           = 0x1a,
00522     L4RE_ABS_TILT_Y           = 0x1b,
00523     L4RE_ABS_TOOL_WIDTH       = 0x1c,
00524     L4RE_ABS_VOLUME           = 0x20,
00525     L4RE_ABS_MISC              = 0x28,
00526     L4RE_ABS_MT_TOUCH_MAJOR   = 0x30,
00527     L4RE_ABS_MT_TOUCH_MINOR   = 0x31,
00528     L4RE_ABS_MT_WIDTH_MAJOR   = 0x32,
00529     L4RE_ABS_MT_WIDTH_MINOR   = 0x33,
00530     L4RE_ABS_MT_ORIENTATION   = 0x34,
00531     L4RE_ABS_MT_POSITION_X    = 0x35,
00532     L4RE_ABS_MT_POSITION_Y    = 0x36,
00533     L4RE_ABS_MT_TOOL_TYPE     = 0x37,
00534     L4RE_ABS_MT_BLOB_ID       = 0x38,
00535     L4RE_ABS_MT_TRACKING_ID    = 0x39,
00536     L4RE_ABS_MT_PRESSURE      = 0x3a,
00537     L4RE_ABS_MT_DISTANCE      = 0x3b,
00538
00539     L4RE_ABS_MAX               = 0x3f,
00540 };
00541
00542 enum L4Re_events_msc
00543 {
00544     L4RE_MSC_SERIAL           = 0x00,
00545     L4RE_MSC_PULSELED         = 0x01,
00546     L4RE_MSC_GESTURE          = 0x02,
00547     L4RE_MSC_RAW               = 0x03,
00548     L4RE_MSC_SCAN              = 0x04,
00549     L4RE_MSC_MAX               = 0x07,
00550 };
00551
00552 enum L4Re_events_properties
00553 {
00554     L4RE_EVENT_PROP_POINTER    = 0x00,
00555     L4RE_EVENT_PROP_DIRECT     = 0x01,
00556     L4RE_EVENT_PROP_BUTTONPAD  = 0x02,
00557     L4RE_EVENT_PROP_SEMI_MT    = 0x03,
00558     //L4RE_EVENT_PROP_MAX      = 0x1f
00559 };

```

## 17.340 l4/re/impl/dataspace\_impl.h File Reference

Dataspace client stub implementation.

```

#include <l4/re/dataspace>
#include <l4/sys/cxx/ipc_client>
#include <l4/sys/cxx/consts>

```



```

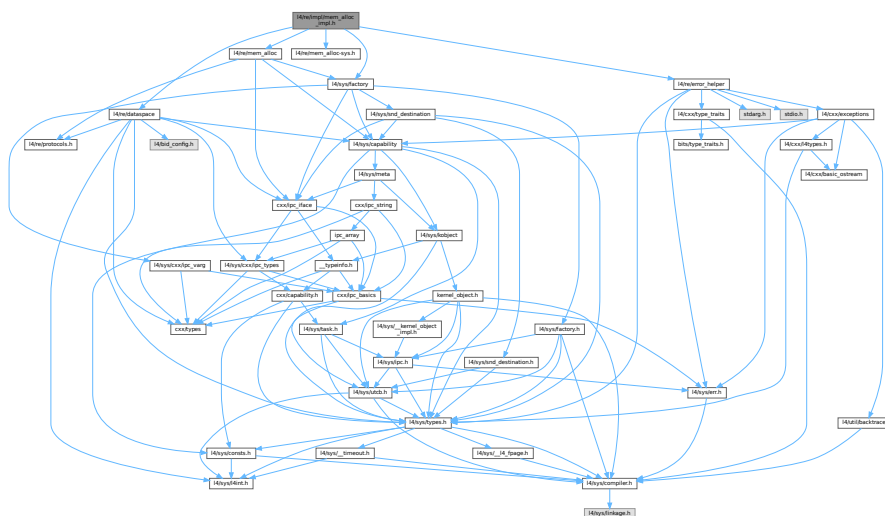
00016 L4_RPC_DEF(L4Re::Dataspace::clear);
00017 L4_RPC_DEF(L4Re::Dataspace::allocate);
00018 L4_RPC_DEF(L4Re::Dataspace::copy_in);
00019 L4_RPC_DEF(L4Re::Dataspace::info);
00020 L4_RPC_DEF(L4Re::Dataspace::map_info);
00021
00022 namespace L4Re {
00023
00024
00025 long
00026 Dataspace::__map(Dataspace::Offset offset, unsigned char *size,
00027                 Dataspace::Flags flags,
00028                 Dataspace::Map_addr local_addr,
00029                 L4::Cap<L4::Task> dst) const noexcept
00030 {
00031     Map_addr spot = local_addr & ~(~0ULL « L4_umword_t(*size));
00032     Map_addr base = local_addr & (~0ULL « L4_umword_t(*size));
00033     L4::Ipc::Rcv_fpage r = L4::Ipc::Rcv_fpage::mem(base, *size, 0, dst);
00034
00035     L4::Ipc::Snd_fpage fp;
00036     long err = map_t::call(c(), offset, spot, flags, r, fp, L4_utcb());
00037     if (L4_UNLIKELY(err < 0))
00038         return err;
00039
00040     *size = fp.rcv_order();
00041     return err;
00042 }
00043
00044 long
00045 Dataspace::map_region(Dataspace::Offset offset, Dataspace::Flags flags,
00046                      Dataspace::Map_addr min_addr,
00047                      Dataspace::Map_addr max_addr,
00048                      L4::Cap<L4::Task> dst) const noexcept
00049 {
00050     min_addr = L4::trunc_page(min_addr);
00051     max_addr = L4::round_page(max_addr);
00052     unsigned char order = L4_LOG2_PAGESIZE;
00053
00054     long err = 0;
00055
00056     while (min_addr < max_addr)
00057     {
00058         unsigned char order_mapped;
00059         order_mapped = order
00060             = L4::max_order(order, min_addr, min_addr, max_addr, min_addr);
00061
00062         err = __map(offset, &order_mapped, flags, min_addr, dst);
00063         if (L4_UNLIKELY(err < 0))
00064             return err;
00065
00066         if (order > order_mapped)
00067             order = order_mapped;
00068
00069         min_addr += Map_addr(1) « order;
00070         offset += Map_addr(1) « order;
00071
00072         if (min_addr >= max_addr)
00073             return 0;
00074
00075         while (min_addr != L4::trunc_order(min_addr, order)
00076              || max_addr < L4::round_order(min_addr + 1, order))
00077             --order;
00078     }
00079
00080     return 0;
00081 }
00082
00083
00084 long
00085 Dataspace::map(Dataspace::Offset offset, Dataspace::Flags flags,
00086               Dataspace::Map_addr local_addr,
00087               Dataspace::Map_addr min_addr,
00088               Dataspace::Map_addr max_addr,
00089               L4::Cap<L4::Task> dst) const noexcept
00090 {
00091     min_addr = L4::trunc_page(min_addr);
00092     max_addr = L4::round_page(max_addr);
00093     local_addr = L4::trunc_page(local_addr);
00094     unsigned char order
00095         = L4::max_order(L4_LOG2_PAGESIZE, local_addr, min_addr, max_addr, local_addr);
00096
00097     return __map(offset, &order, flags, local_addr, dst);
00098 }
00099
00100 Dataspace::Size
00101 Dataspace::size() const noexcept
00102 {

```



## 17.342 l4/re/impl/mem\_alloc impl.h File Reference

```
#include <l4/re/mem_alloc>
#include <l4/re/mem_alloc-sys.h>
#include <l4/re/dataspace>
#include <l4/re/error_helper>
#include <l4/sys/factory>
Include dependency graph for mem_alloc_impl.h:
```



- namespace **L4Re**  
*L4Re C++ Interfaces.*

Definition in file `mem_alloc_impl.h`.

## 17.343 mem\_alloc\_impl.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #include <l4/re/mem_alloc>
00013 #include <l4/re/mem_alloc-sys.h>
00014 #include <l4/re/dataspace>
00015 #include <l4/re/error_helper>
00016
00017 #include <l4/sys/factory>
00018
00019
00020 namespace L4Re
00021 {
00022
00023     long
00024     Mem_alloc::alloc(long size,
00025                     L4::Cap<Dataspace> mem, unsigned long flags,
00026                     unsigned long align, l4_addr_t paddr) const noexcept
00027     {
00028         L4::Cap<L4::Factory> f(cap());
00029         auto call = f->create(mem, L4Re::Dataspace::Protocol);
00030         call « l4_mword_t(size)
00031             « l4_umword_t(flags)
00032             « l4_umword_t(align);
00033         if (flags & Fixed_paddr)
00034             call « l4_umword_t(paddr);
00035         return l4_error(call);
00036     }
00037
00038 };

```

## 17.344 l4/re/impl/namespace\_impl.h File Reference

Namespace client stub implementation.

```

#include <l4/re/namespace>
#include <l4/util/util.h>
#include <l4/sys/cxx/ipc_client>
#include <l4/sys/assert.h>
#include <string.h>

```



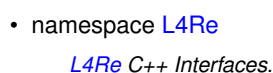
```

00021 L4_RPC_DEF(L4Re::Namespace::register_obj);
00022 L4_RPC_DEF(L4Re::Namespace::unlink);
00023
00024 namespace L4Re {
00025
00026 long
00027 Namespace::_query(char const *name, unsigned len,
00028                  L4::Cap<void> const &target,
00029                  l4_umword_t *local_id, bool iterate) const noexcept
00030 {
00031     l4_assert(target.is_valid());
00032
00033     L4::Cap<Namespace> ns = c();
00034     L4::Ipc::Array<char const, unsigned long> _name(len, name);
00035
00036     while (_name.length > 0)
00037     {
00038         L4::Ipc::Snd_fpage cap;
00039         L4::Opcode dummy;
00040         int err = query_t::call(ns, _name,
00041                                L4::Ipc::Small_buf(target.cap(),
00042                                                    local_id
00043                                                    ? L4_RCV_ITEM_LOCAL_ID
00044                                                    : 0),
00045                                cap, dummy, _name);
00046         if (err < 0)
00047             return err;
00048
00049         bool const partly = err & Partly_resolved;
00050         if (cap.id_received())
00051         {
00052             *local_id = cap.data();
00053             return _name.length;
00054         }
00055
00056         if (partly && iterate)
00057             ns = L4::cap_cast<Namespace>(target);
00058         else
00059             return err;
00060     }
00061
00062     return _name.length;
00063 }
00064
00065 long
00066 Namespace::query(char const *name, unsigned len, L4::Cap<void> const &target,
00067                 int timeout, l4_umword_t *local_id, bool iterate) const noexcept
00068 {
00069     if (L4_UNLIKELY(len == 0))
00070         return -L4_EINVAL;
00071
00072     if (L4_UNLIKELY(timeout < 0))
00073         return -L4_EINVAL;
00074
00075     long ret;
00076     long rem = timeout;
00077     long to = 0;
00078
00079     if (rem)
00080         to = 10;
00081
00082     do
00083     {
00084         ret = _query(name, len, target, local_id, iterate);
00085
00086         if (ret >= 0)
00087             return ret;
00088
00089         if (L4_UNLIKELY(ret != -L4_EAGAIN))
00090             return ret;
00091
00092         if (rem == to)
00093             return ret;
00094
00095         l4_sleep(to);
00096
00097         if (rem > 0)
00098         {
00099             rem -= to;
00100             if (rem < 0)
00101             {
00102                 to = rem = 0;
00103                 continue; // one final try
00104             }
00105         }
00106
00107         if (to < 100)

```

### 17.346 I4/re/impl/rm impl.h File Reference

```
#include <l4/bid_config.h>
#include <l4/re/rm>
#include <l4/re/dataspace>
#include <l4/sys/cxx/ipc_client>
#include <l4/sys/task>
#include <l4/sys/err.h>
Include dependency graph for rm_impl.h:
```



Generated for L4Re by Doxygen

## 17.347 rm\_impl.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #include <l4/bid_config.h>
00013 #include <l4/re/rm>
00014 #include <l4/re/dataspace>
00015
00016 #include <l4/sys/cxx/ipc_client>
00017
00018 #include <l4/sys/task>
00019 #include <l4/sys/err.h>
00020
00021 L4_RPC_DEF(L4Re::Rm::reserve_area);
00022 L4_RPC_DEF(L4Re::Rm::free_area);
00023 L4_RPC_DEF(L4Re::Rm::attach);
00024 L4_RPC_DEF(L4Re::Rm::detach);
00025 L4_RPC_DEF(L4Re::Rm::get_regions);
00026 L4_RPC_DEF(L4Re::Rm::get_areas);
00027 L4_RPC_DEF(L4Re::Rm::find);
00028 L4_RPC_DEF(L4Re::Rm::get_info);
00029
00030 namespace L4Re
00031 {
00032
00033     long
00034     Rm::attach(l4_addr_t *start, unsigned long size, Rm::Flags flags,
00035               L4::Ipc::Cap<Dataspace> mem, Rm::Offset offs,
00036               unsigned char align, L4::Cap<L4::Task> const task,
00037               char const *name, Rm::Offset backing_offset) const noexcept
00038     {
00039         if ((flags & F::Rights_mask) == Flags(0)
00040             || (flags & (F::Reserved | F::Kernel)))
00041             mem = L4::Ipc::Cap<L4Re::Dataspace>();
00042
00043         char const n = '\0';
00044         long e = attach_t::call(c(), start, size, flags, mem, offs, align,
00045                                mem.cap().cap(), name ? name : &n, backing_offset);
00046         if (e < 0)
00047             return e;
00048
00049 #ifndef CONFIG_MMU
00050         if ((flags & (F::Eager_map | F::No_eager_map)) == F::Eager_map)
00051 #else
00052         if (!(flags & F::No_eager_map) && mem.is_valid())
00053 #endif
00054             e = mem.cap()->map_region(offs, map_flags(flags), *start, *start + size,
00055                                       task);
00056
00057         return e;
00058     }
00059
00060     int
00061     Rm::detach(l4_addr_t start, unsigned long size, L4::Cap<Dataspace> *mem,
00062               L4::Cap<L4::Task> task, unsigned flags) const noexcept
00063     {
00064         l4_addr_t rstart = 0, rsize = 0;
00065         l4_cap_idx_t mem_cap = L4_INVALID_CAP;
00066         long e = detach_t::call(c(), start, size, flags, rstart, rsize, mem_cap);
00067         if (L4_UNLIKELY(e < 0))
00068             return e;
00069
00070         if (mem)
00071             *mem = L4::Cap<L4Re::Dataspace>(mem_cap);
00072
00073         if (!task.is_valid())
00074             return e;
00075
00076         rsize = l4_round_page(rsize);
00077         unsigned order = L4_LOG2_PAGESIZE;
00078         unsigned long sz = (1UL << order);
00079         for (unsigned long p = rstart; rsize; p += sz, rsize -= sz)
00080         {
00081             while (sz > rsize)
00082             {
00083                 --order;
00084                 sz >>= 1;
00085             }

```

```

00086
00087     for (;;)
00088     {
00089         unsigned long m = sz << 1;
00090         if (m > rsize)
00091             break;
00092
00093         if (p & (m - 1))
00094             break;
00095
00096         ++order;
00097         sz <= 1;
00098     }
00099
00100     task->unmap(l4_fpage(p, order, L4_FPAGE_RWX),
00101                L4_FP_ALL_SPACES);
00102 }
00103
00104 return e;
00105 }
00106 }

```

## 17.348 inhibitor

```

00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Steffen Liebergeld <steffen.liebergeld@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/sys/capability>
00010 #include <l4/sys/cxx/ipc_iface>
00011 #include <l4/sys/cxx/ipc_string>
00012 #include <l4/re/protocols.h>
00013
00014 namespace L4Re {
00015
00038 class Inhibitor :
00039     public L4::Kobject_t<Inhibitor, L4::Kobject, L4RE_PROTO_INHIBITOR>
00040 {
00041 public:
00042     enum
00043     {
00044         Name_max = 20
00045     };
00046
00057     L4_INLINE_RPC(long, acquire, (l4_umword_t id, L4::Ipc::String<> reason));
00058
00067     L4_INLINE_RPC(long, release, (l4_umword_t id));
00068
00084     long next_lock_info(char *name, unsigned len, l4_mword_t current_id = -1,
00085                        l4_utcb_t *utcb = l4_utcb())
00086     {
00087         L4::Ipc::String<char> name_buf(len, name);
00088         long r = next_lock_info_t::call(c(), &current_id, name_buf, utcb);
00089         if (r < 0)
00090             return r;
00091
00092         return current_id;
00093     }
00094
00095     L4_INLINE_RPC_NF(long, next_lock_info, (L4::Ipc::In_out<l4_mword_t *> current_id,
00096                                           L4::Ipc::String<char> &name));
00097
00098     typedef L4::Typeid::Rpcs<acquire_t, release_t, next_lock_info_t> Rpcs;
00099 };
00100
00101 }

```

## 17.349 inhibitor-sys.h

```

00001 /*
00002  * (c) 2014 Steffen Liebergeld <steffen.liebergeld@kernkonzept.com>
00003  *
00004  * License: see LICENSE.spdx (in this directory or the directories above)
00005  */

```

```

00006 #pragma once
00007
00008 namespace L4Re {
00009     namespace Inhibitor_ {
00015         enum Opcodes { Acquire, Release, Next_lock_info };
00016     }
00017 }

```

## 17.350 itas

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2025 Kernkonzept GmbH.
00004  * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 #include <l4/sys/cxx/ipc_iface>
00012 #include <l4/sys/cxx/ipc_types>
00013 #include <l4/sys/cxx/types>
00014 #include <l4/sys/l4int.h>
00015 #include <l4/sys/thread>
00016 #include <l4/sys/types.h>
00017
00018 #include <signal.h>
00019 #include <sys/time.h>
00020
00021 namespace L4Re
00022 {
00023
00030 class L4_EXPORT Itas :
00031     public L4::Kobject_t<Itas, L4::Kobject,
00032                         L4RE_PROTO_ITAS,
00033                         L4::Type_info::Demand_t<2> >
00034 {
00035 public:
00048     L4_INLINE_RPC(int, register_thread, (L4::Ipc::Cap<L4::Thread> parent,
00049                                         L4::Ipc::Cap<L4::Thread> thread_cap,
00050                                         l4_addr_t thread_utcb));
00060     L4_INLINE_RPC(int, unregister_thread, (L4::Ipc::Cap<L4::Thread> thread));
00061
00062     // sigaction.sa_flags is usually `unsigned long`, except for MIPS...
00063     enum : unsigned
00064     {
00065         Ignore_sigaction = ~0U
00066     };
00067
00079     L4_INLINE_RPC(int, sigaction, (int signum,
00080                                   const struct sigaction *act,
00081                                   struct sigaction *oldact));
00082
00095     L4_INLINE_RPC(int, sigaltstack, (L4::Ipc::Cap<L4::Thread> thread,
00096                                     const struct sigaltstack *ss,
00097                                     struct sigaltstack *oss));
00098
00113     L4_INLINE_RPC(int, sigprocmask, (L4::Ipc::Cap<L4::Thread> thread,
00114                                     int how, sigset_t const *set,
00115                                     sigset_t *oldset));
00116
00126     L4_INLINE_RPC(int, sigpending, (L4::Ipc::Cap<L4::Thread> thread,
00127                                    sigset_t *set));
00128
00138     L4_INLINE_RPC(int, setitimer, (int which,
00139                                   const struct itimerval *new_value,
00140                                   struct itimerval *old_value));
00141
00150     L4_INLINE_RPC(int, getitimer, (int which, struct itimerval *curr_value));
00151
00158     L4_INLINE_RPC(int, raise, (L4::Ipc::Cap<L4::Thread> thread, int sig));
00159
00160     typedef L4::Typeid::Rpc<
00161         register_thread_t, unregister_thread_t, sigaction_t, sigaltstack_t,
00162         sigprocmask_t, sigpending_t, setitimer_t, getitimer_t, raise_t
00163     > Rpc<
00164     >;
00165
00166 }

```

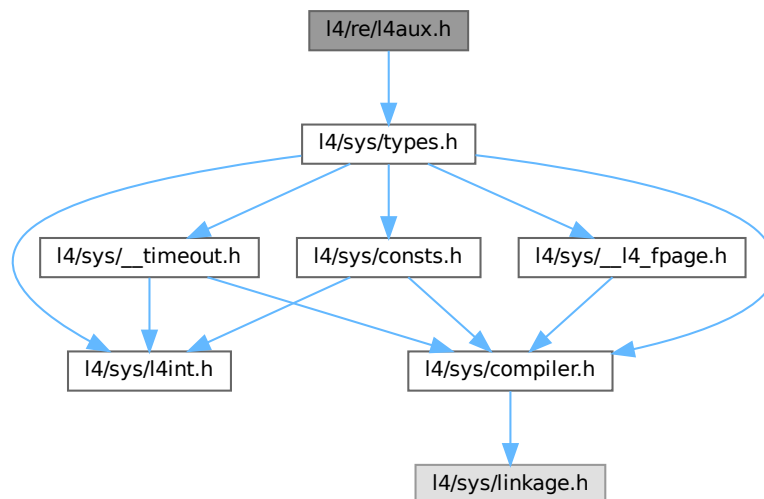


## 17.351 l4/re/l4aux.h File Reference

Auxiliary definitions.

```
#include <l4/sys/types.h>
```

Include dependency graph for l4aux.h:



### Data Structures

- struct [l4re\\_aux\\_t](#)  
*Auxiliary descriptor.*

### Typedefs

- typedef struct l4re\_aux\_t [l4re\\_aux\\_t](#)  
*Auxiliary descriptor.*

### Enumerations

- enum [l4re\\_aux\\_ldr\\_flags\\_t](#)  
*Flags for program loading.*

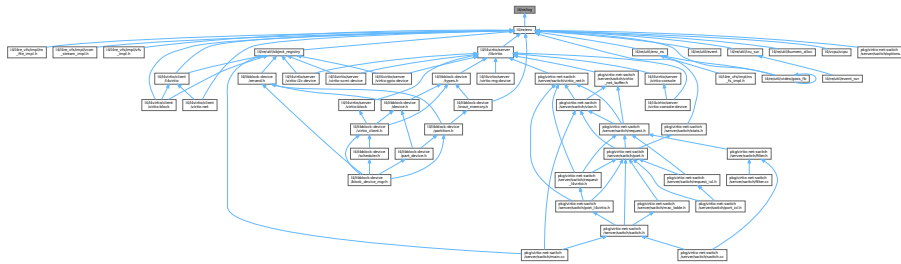
## 17.351.1 Detailed Description

Auxiliary definitions.

Definition in file [l4aux.h](#).



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4Re::Log](#)  
*Log interface class.*

## Namespaces

- namespace [L4Re](#)  
*L4Re C++ Interfaces.*

### 17.353.1 Detailed Description

Log interface.

Definition in file [log](#).

## 17.354 log

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #pragma once
00015
00016 #include <l4/sys/vcon>
00017
00018 namespace L4Re {
00019
00033 class L4_EXPORT Log : public L4::Kobject_t<Log, L4::Vcon, L4::PROTO_EMPTY>
00034 {
00035 public:
00036
00043     void printn(char const *string, int len) const noexcept;
00044
00050     void print(char const *string) const noexcept;
00051 };
00052 }

```

## 17.355 l4/re/log-sys.h File Reference

Log protocol definition.

### Namespaces

- namespace [L4Re](#)  
*[L4Re](#) C++ Interfaces.*

### Enumerations

- enum [L4Re::Log\\_::Opcodes](#)  
*Logging-service communication-protocol opcodes.*

### 17.355.1 Detailed Description

Log protocol definition.

Definition in file [log-sys.h](#).

## 17.356 log-sys.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 namespace L4Re
00015 {
00016     namespace Log_
00017     {
00023         enum Opcodes { Print };
00024     };
00025 };

```

## 17.357 l4/re/mem\_alloc File Reference

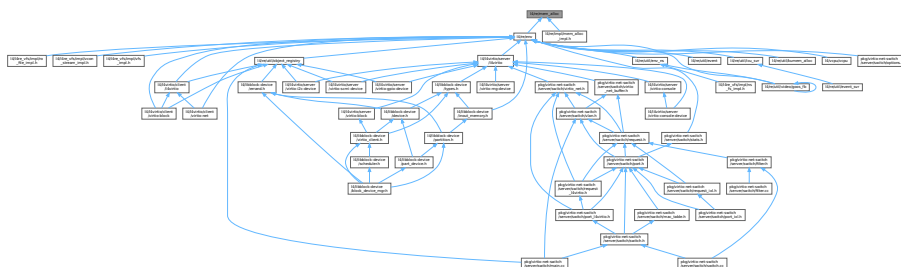
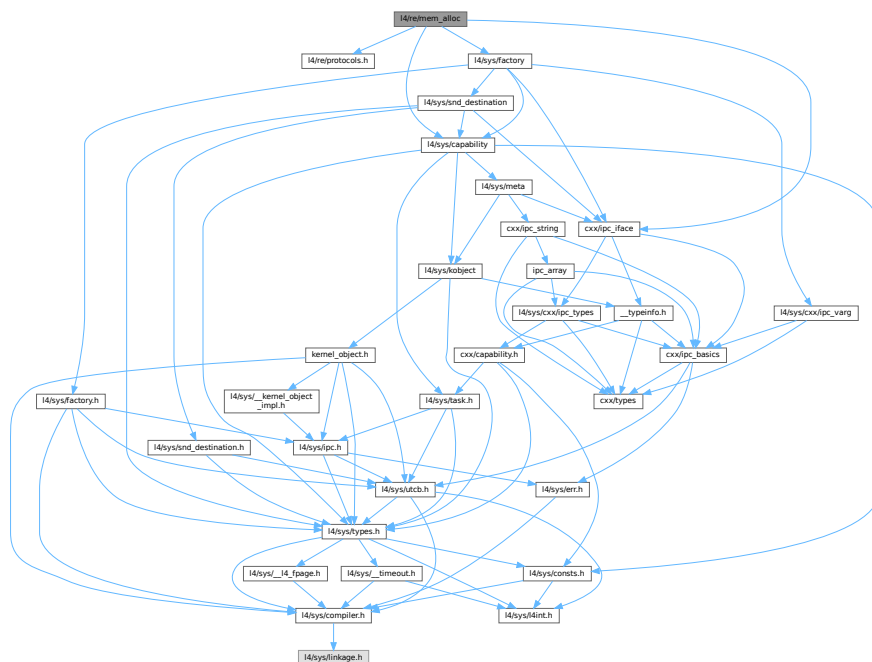
Memory allocator interface.

```

#include <l4/re/protocols.h>
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_iface>

```

Include dependency graph for mem\_alloc:



- [Link to the Internet](#)

## 17.357.1 Detailed Description

Memory allocator interface.

Definition in file [mem\\_alloc](#).

## 17.358 mem\_alloc

[Go to the documentation of this file.](#)

```

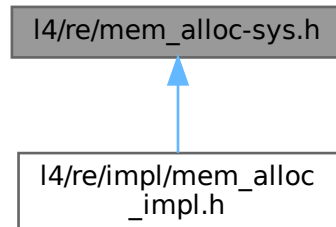
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * Copyright (C) 2014-2016, 2019, 2021, 2024 Kernkonzept GmbH.
00009  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00010  */
00011 /*
00012  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00013  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00014  *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00015  *               economic rights: Technische Universität Dresden (Germany)
00016  *
00017  * License: see LICENSE.spdx (in this directory or the directories above)
00018  */
00019 #pragma once
00020
00021 #include <l4/re/protocols.h>
00022 #include <l4/sys/capability>
00023 #include <l4/sys/cxx/ipc_iface>
00024 #include <l4/sys/factory>
00025
00026 namespace L4Re {
00027 class Dataspace;
00028
00029 // MISSING:
00030 // * alignment constraints
00031 // * shall we support superpages in noncont memory?
00032
00052 class L4_EXPORT Mem_alloc :
00053     public L4::Kobject_t<Mem_alloc, L4::Factory, L4RE_PROTO_MEM_ALLOC>
00054 {
00055 public:
00062     enum Mem_alloc_flags
00063     {
00064         Continuous    = 0x01,
00065         Pinned        = 0x02,
00066         Super_pages   = 0x04,
00067         Fixed_paddr   = 0x08,
00068     };
00069 };
00070
00074 struct Stats
00075 {
00083     l4_size_t quota;
00084
00094     l4_size_t quota_used;
00095
00102     l4_size_t mem_limit;
00103
00116     l4_size_t mem_used;
00117
00126     l4_size_t mem_free;
00127 };
00128
00157 long alloc(long size, L4::Cap<Dataspace> mem,
00158             unsigned long flags = 0, unsigned long align = 0,
00159             l4_addr_t paddr = 0) const noexcept;
00160
00169 L4_INLINE_RPC(long, info, (Stats &stats));
00170
00171 typedef L4::Typeid::Rpc<info_t> Rpc;
00172 };
00173
00174 };

```

## 17.359 l4/re/mem\_alloc-sys.h File Reference

Memory allocator protocol definitions.

This graph shows which files directly or indirectly include this file:



### Namespaces

- namespace [L4Re](#)  
*L4Re C++ Interfaces.*

### Enumerations

- enum [L4Re::Mem\\_alloc\\_::Opcodes](#)  
*Memory-allocator communication-protocol opcodes.*

### 17.359.1 Detailed Description

Memory allocator protocol definitions.

Definition in file [mem\\_alloc-sys.h](#).

## 17.360 mem\_alloc-sys.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 namespace L4Re
00015 {
00016     namespace Mem_alloc_
00017     {
00023         enum Opcodes { Alloc, Free };
00024     };
00025 };
  
```





## 17.362 mmio\_space

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00004  * Copyright (C) 2017-2018, 2022, 2024 Kernkonzept GmbH.
00005  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00013 #pragma once
00014
00015 #include <l4/re/protocols.h>
00016 #include <l4/sys/capability>
00017 #include <l4/sys/cxx/ipc_types>
00018 #include <l4/sys/cxx/ipc_iface>
00019
00020 namespace L4Re
00021 {
00022
00045 struct L4_EXPORT Mmio_space
00046 : public L4::Kobject_t<Mmio_space, L4::Kobject, L4RE_PROTO_MMIO_SPACE>
00047 {
00049     enum Access_width
00050     {
00051         Wd_8bit = 0,
00052         Wd_16bit = 1,
00053         Wd_32bit = 2,
00054         Wd_64bit = 3
00055     };
00056
00058     typedef l4_uint64_t Addr;
00059
00074     L4_INLINE_RPC(long, mmio_read, (Addr addr, char width, l4_uint64_t *value));
00075
00090     L4_INLINE_RPC(long, mmio_write, (Addr addr, char width, l4_uint64_t value));
00091
00092     typedef L4::Typeid::Rpc<mmio_read_t, mmio_write_t> Rpc;
00093 };
00094
00095 }
```

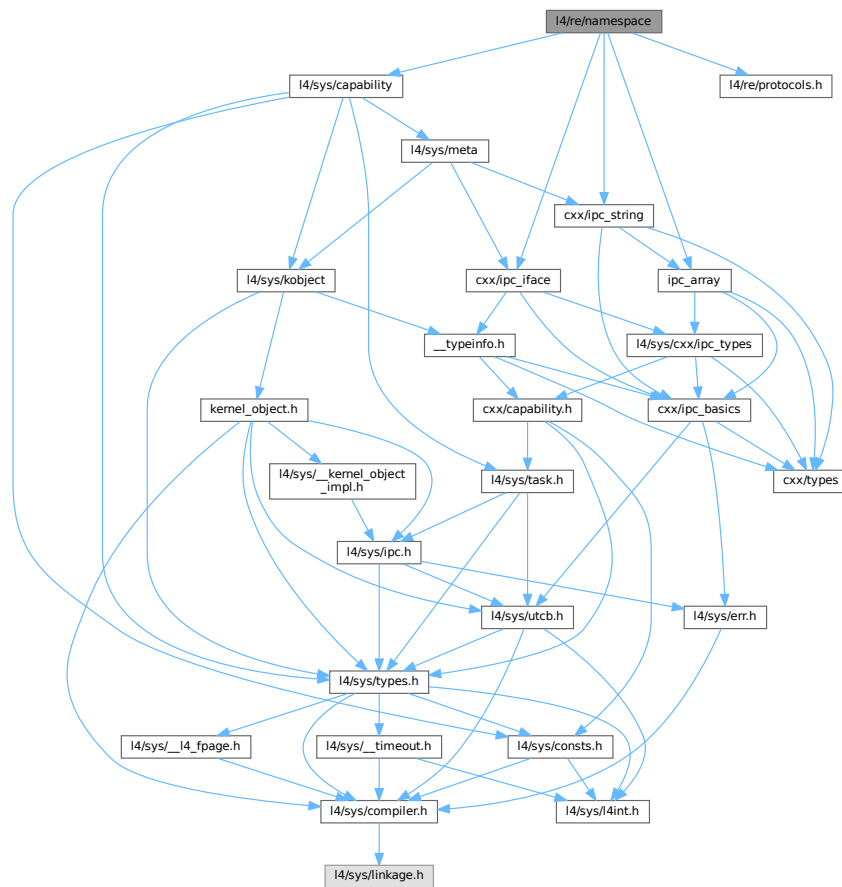
## 17.363 l4/re/namespace File Reference

Namespace interface.

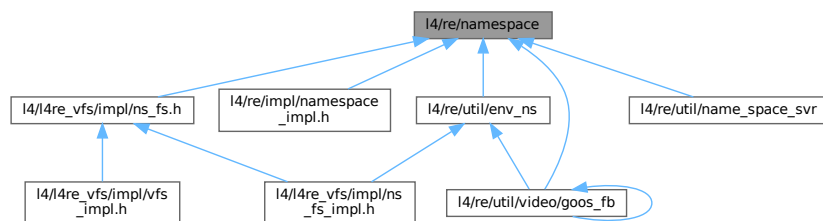
```

#include <l4/sys/capability>
#include <l4/re/protocols.h>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/cxx/ipc_array>
#include <l4/sys/cxx/ipc_string>
```

Include dependency graph for namespace:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4Re::Namespace](#)  
*Name-space interface.*

## Namespaces

- namespace [L4Re](#)  
*L4Re C++ Interfaces.*

### 17.363.1 Detailed Description

Namespace interface.

Definition in file [namespace](#).

## 17.364 namespace

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00004  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00005  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00006  *      Björn Döbel <doebel@os.inf.tu-dresden.de>
00007  *      economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00013 #include <l4/sys/capability>
00014 #include <l4/re/protocols.h>
00015 #include <l4/sys/cxx/ipc_iface>
00016 #include <l4/sys/cxx/ipc_array>
00017 #include <l4/sys/cxx/ipc_string>
00018
00019 namespace L4Re {
00020
00021 class L4_EXPORT Namespace :
00022 public L4::Kobject<Namespace, L4::Kobject, L4RE_PROTO_NAMESPACE,
00023     L4::Type_info::Demand_t<1> >
00024 {
00025 public:
00026     enum Register_flags
00027     {
00028         Ro      = L4_CAP_FPAGE_RO,
00029         Rw      = L4_CAP_FPAGE_RW,
00030         Rs      = L4_CAP_FPAGE_RS,
00031         Rws     = L4_CAP_FPAGE_RWS,
00032         Strong  = L4_CAP_FPAGE_S,
00033         Trusted = 0x008,
00034
00035         Cap_flags = Ro | Rw | Strong | Trusted,
00036
00037         Link      = 0x100,
00038         Overwrite = 0x200,
00039     };
00040
00041     enum Query_result_flags
00042     {
00043         Partly_resolved = 0x020,
00044     };
00045
00046     enum Query_timeout
00047     {
00048         To_default      = 3600000,
00049         To_non_blocking = 0,
00050     };
00051
00052     L4_RPC_NF(
00053         long, query, (L4::Ipc::Array_ref<char const, unsigned long> name,
00054             L4::Ipc::Small_buf cap,
00055             L4::Ipc::Snd_fpage &snd_cap, L4::Ipc::Opt<L4::Opcode &> dummy,
00056             L4::Ipc::Opt<L4::Ipc::Array_ref<char const, unsigned long> &> out_name));
00057
00058     long query(char const *name, L4::Cap<void> const &cap,
00059         int timeout = To_default,
00060         l4_umword_t *local_id = 0, bool iterate = true) const noexcept;
00061
00062     long query(char const *name, unsigned len, L4::Cap<void> const &cap,
00063         int timeout = To_default,
00064         l4_umword_t *local_id = 0, bool iterate = true) const noexcept;
00065
00066     L4_RPC_NF(long, register_obj, (unsigned flags,
00067         L4::Ipc::Array<char const, unsigned long> name,
00068         L4::Ipc::Opt< L4::Ipc::Cap<void> > obj),
00069         L4::Ipc::Call_t<L4_CAP_FPAGE_W>);

```

```

00141
00165 long register_obj(char const *name, L4::Ipc::Cap<void> obj,
00166                  unsigned flags = Rw) const noexcept
00167 {
00168     return register_obj_t::call(c(), flags,
00169                                L4::Ipc::Array<char const, unsigned long>(
00170                                    __builtin_strlen(name), name),
00171                                obj);
00172 }
00173
00174 L4_RPC_NF_OP(3, // backward compatibility opcode
00175             long, unlink, (L4::Ipc::Array<char const, unsigned long> name),
00176                         L4::Ipc::Call_t<L4_CAP_FPAGE_W>);
00177
00192 long unlink(char const* name)
00193 {
00194     return unlink_t::call(c(), L4::Ipc::Array<char const, unsigned long>(
00195                                     __builtin_strlen(name), name));
00196 }
00197
00198 typedef L4::Typeid::Rpc<query_t, register_obj_t, unlink_t> Rpcs;
00199
00200 private:
00201     long _query(char const *name, unsigned len,
00202                L4::Cap<void> const &target, l4_umword_t *local_id,
00203                bool iterate) const noexcept;
00204
00205 };
00206
00207 };

```

## 17.365 l4/re/namespace-sys.h File Reference

Namespace protocol definitions.

### Namespaces

- namespace [L4Re](#)  
[L4Re C++ Interfaces](#).

### Enumerations

- enum [L4Re::Namespace\\_::Opcodes](#)  
*Name-space communication-protocol opcodes.*

### 17.365.1 Detailed Description

Namespace protocol definitions.

Definition in file [namespace-sys.h](#).

## 17.366 namespace-sys.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 namespace L4Re {
00015     namespace Namespace_
00016     {
00022         enum Opcodes { Query, Register, Link, Unlink };
00023     };
00024 };

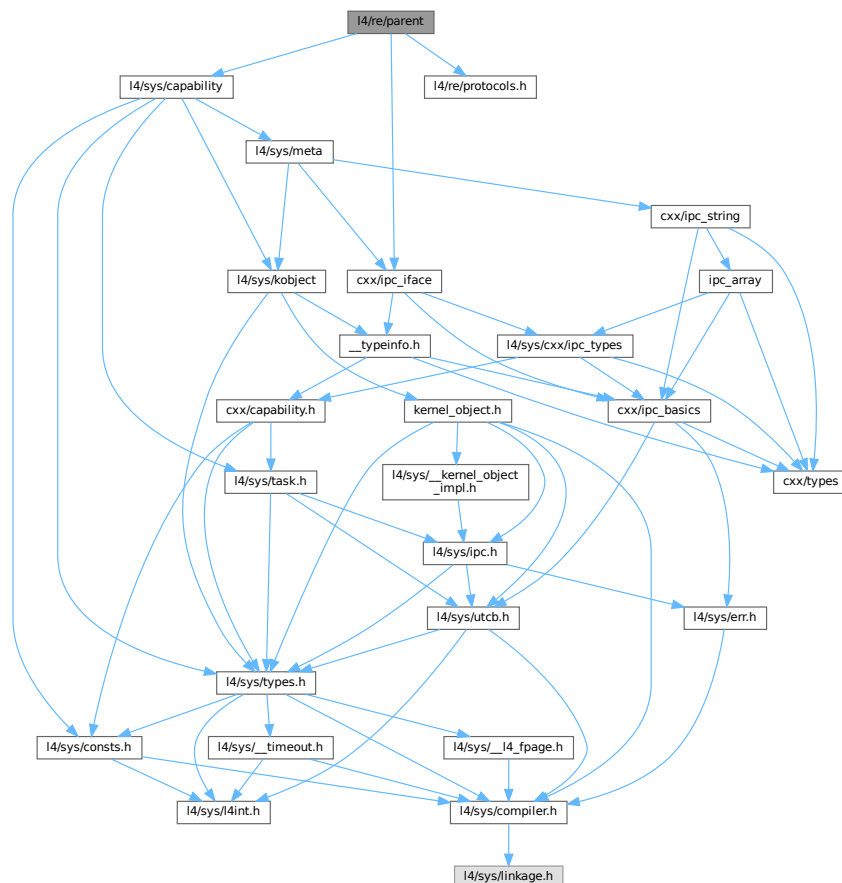
```

## 17.367 I4/re/parent File Reference

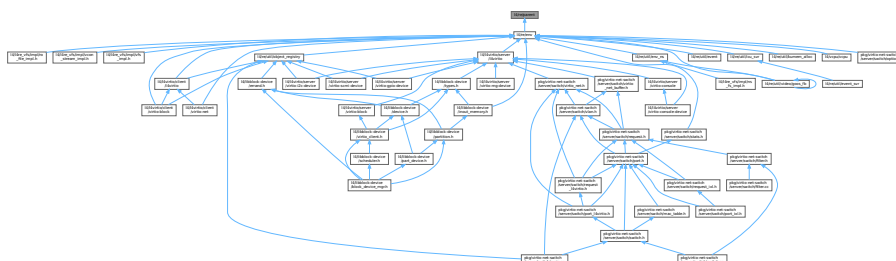
Parent interface.

```
#include <l4/sys/capability>
#include <l4/re/protocols.h>
#include <l4/sys/cxx/ipc_iface>
```

Include dependency graph for parent:



This graph shows which files directly or indirectly include this file:



### Data Structures

- class [L4Re::Parent](#)  
*Parent interface.*

## Namespaces

- namespace [L4Re](#)  
*L4Re C++ Interfaces.*

### 17.367.1 Detailed Description

Parent interface.

Definition in file [parent](#).

## 17.368 parent

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #pragma once
00015
00016 #include <l4/sys/capability>
00017 #include <l4/re/protocols.h>
00018 #include <l4/sys/cxx/ipc_iface>
00019
00020 namespace L4Re {
00021
00034
00042 class L4_EXPORT Parent :
00043     public L4::Kobject_t<Parent, L4::Kobject, L4RE_PROTO_PARENT>
00044 {
00045     public:
00061     L4_INLINE_RPC(long, signal, (unsigned long sig, unsigned long val));
00062     typedef L4::Typeid::Rpc<signal_t> Rpc;
00063 };
00064 };
00065

```

## 17.369 l4/re/parent-sys.h File Reference

Parent protocol definition.

## Namespaces

- namespace [L4Re](#)  
*L4Re C++ Interfaces.*

## Enumerations

- enum [L4Re::Parent\\_::Opcodes](#)  
*Parent communication-protocol opcodes.*

## 17.369.1 Detailed Description

Parent protocol definition.

Definition in file [parent-sys.h](#).

## 17.370 parent-sys.h

[Go to the documentation of this file.](#)

```

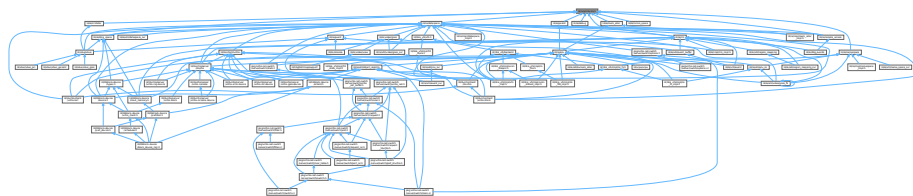
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 namespace L4Re
00015 {
00016     namespace Parent_
00017     {
00023         enum Opcodes { Signal };
00024     };
00025 };

```

## 17.371 I4/re/protocols.h File Reference

[L4Re](#) Protocol Constants (C version).

This graph shows which files directly or indirectly include this file:



### Enumerations

- enum [L4re\\_protocols](#) {  
[L4RE\\_PROTO\\_DATASPACE](#) = 0x4000 , [L4RE\\_PROTO\\_NAMESPACE](#) , [L4RE\\_PROTO\\_PARENT](#) ,  
[L4RE\\_PROTO\\_GOOS](#) ,  
[L4RE\\_PROTO\\_RSVD\\_1](#) , [L4RE\\_PROTO\\_RM](#) , [L4RE\\_PROTO\\_EVENT](#) , [L4RE\\_PROTO\\_INHIBITOR](#) ,  
[L4RE\\_PROTO\\_DMA\\_SPACE](#) , [L4RE\\_PROTO\\_MMIO\\_SPACE](#) , [L4RE\\_PROTO\\_ITAS](#) , [L4RE\\_PROTO\\_MEM\\_ALLOC](#)  
, [L4RE\\_PROTO\\_REMOTE\\_ACCESS](#) , [L4RE\\_PROTO\\_DEBUG](#) = ~0x7fffL }

Common [L4Re](#) Protocol Constants.

## 17.371.1 Detailed Description

[L4Re](#) Protocol Constants (C version).

Definition in file [protocols.h](#).

## 17.372 protocols.h

[Go to the documentation of this file.](#)

```
00001
00005
00006 /*
00007  * (c) 2015 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011
00012 #pragma once
00013
00019
00024 enum L4re_protocols
00025 {
00026     L4RE_PROTO_DATASPACE = 0x4000,
00027     L4RE_PROTO_NAMESPACE,
00028     L4RE_PROTO_PARENT,
00029     L4RE_PROTO_GOOS,
00030     L4RE_PROTO_RSVD_1,
00031     L4RE_PROTO_RM,
00032     L4RE_PROTO_EVENT,
00033     L4RE_PROTO_INHIBITOR,
00034     L4RE_PROTO_DMA_SPACE,
00035     L4RE_PROTO_MMIO_SPACE,
00036     L4RE_PROTO_ITAS,
00037     L4RE_PROTO_MEM_ALLOC,
00038     L4RE_PROTO_REMOTE_ACCESS,
00039
00040     L4RE_PROTO_DEBUG = ~0x7fffL
00041 };
00042
```

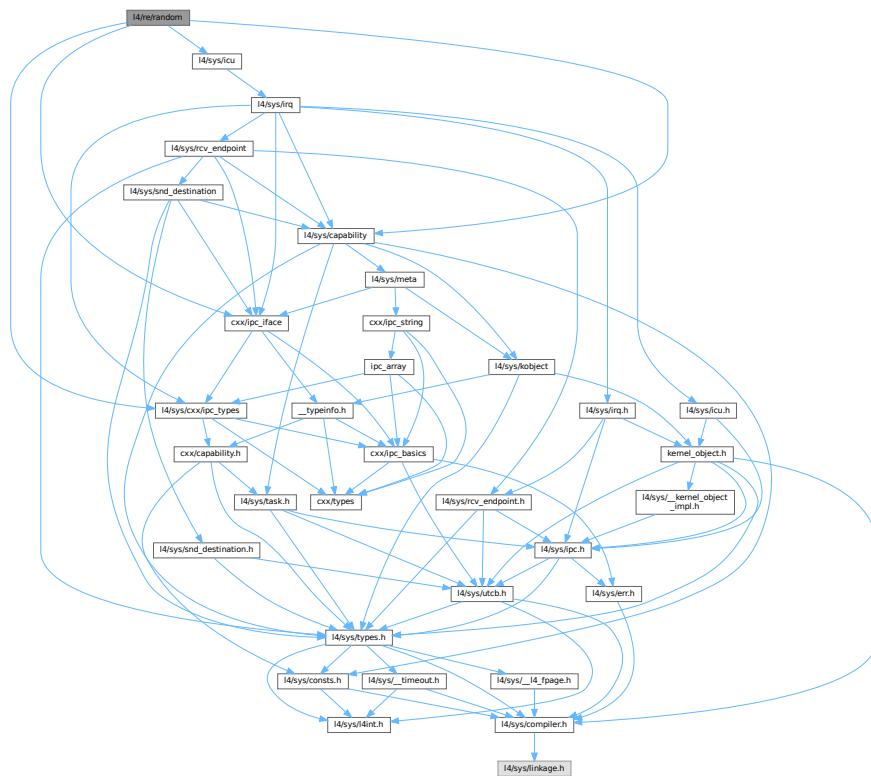
## 17.373 l4/re/random File Reference

Random number generator interface definition.

```
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/icu>
```



Include dependency graph for random:



## Data Structures

- struct [L4Re::Random](#)  
*Low-bandwidth interface for random number generators.*

## Namespaces

- namespace [L4Re](#)  
*L4Re C++ Interfaces.*

### 17.373.1 Detailed Description

Random number generator interface definition.

Definition in file [random](#).

## 17.374 random

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2019-2020, 2022, 2024 Kernkonzept GmbH.
00004  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00012 #pragma once
00013
00014 #include <l4/sys/capability>
00015 #include <l4/sys/cxx/ipc_types>
00016 #include <l4/sys/cxx/ipc_iface>
00017 #include <l4/sys/icu>
00018
00019 namespace L4Re
00020 {
00021
00033 struct L4_EXPORT Random
00034 : public L4::Kobject_t<Random, L4::Icu>
00035 {
00060     L4_INLINE_RPC(long, get_random, (l4_size_t size,
00061                                     L4::Ipc::Array<char, unsigned long> *buffer));
00062
00063     typedef L4::Typeid::Rpcs<get_random_t> Rpcs;
00064 };
00065
00066 } // namespace

```

## 17.375 remote\_access

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2025 Adam Lackorzynski <adam@l4re.org>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <l4/re/protocols.h>
00011 #include <l4/re/dataspace>
00012 #include <l4/sys/cxx/ipc_iface>
00013 #include <l4/sys/cxx/ipc_types>
00014 #include <l4/sys/cxx/types>
00015 #include <l4/sys/l4int.h>
00016
00017 namespace L4Re
00018 {
00019
00020 struct L4_EXPORT Remote_access
00021 : public L4::Kobject_t<Remote_access, Dataspace, L4RE_PROTO_REMOTE_ACCESS>
00022 {
00023     enum Access_width
00024     {
00025         Wd_8bit = 0,
00026         Wd_16bit = 1,
00027         Wd_32bit = 2,
00028         Wd_64bit = 3
00029     };
00030
00031     L4_INLINE_RPC(long, read_mem, (l4_addr_t addr, char width, l4_uint64_t *value));
00032     L4_INLINE_RPC(long, write_mem, (l4_addr_t addr, char width, l4_uint64_t value));
00033
00034     //L4_INLINE_RPC(long, get_regs, (unsigned TBD_THREAD_ID, l4_exc_regs_t *regs));
00035     //L4_INLINE_RPC(long, set_regs, (unsigned TBD_THREAD_ID, l4_exc_regs_t regs));
00036
00037     // !!! sizeof(l4_fpu_regs_t) > sizeof(utcb)
00038     //L4_INLINE_RPC(long, get_fpu_regs, (l4_fpu_regs_t *regs));
00039     //L4_INLINE_RPC(long, set_fpu_regs, (l4_fpu_regs_t regs));
00040
00041     L4_INLINE_RPC(long, terminate, (int exit_code), L4::Ipc::Send_only);
00042
00043     typedef L4::Typeid::Rpcs<read_mem_t, write_mem_t, terminate_t> Rpcs;
00044 };
00045
00046 }

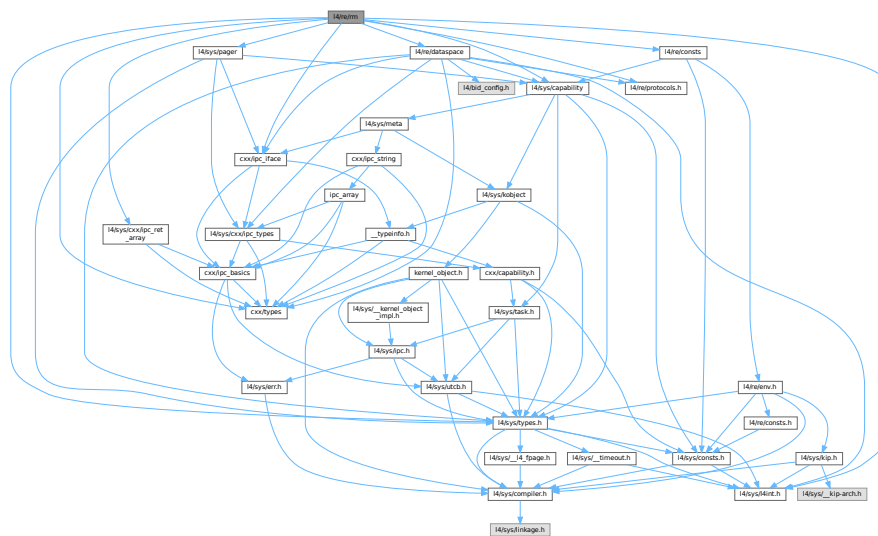
```

## 17.376 I4/re/rm File Reference

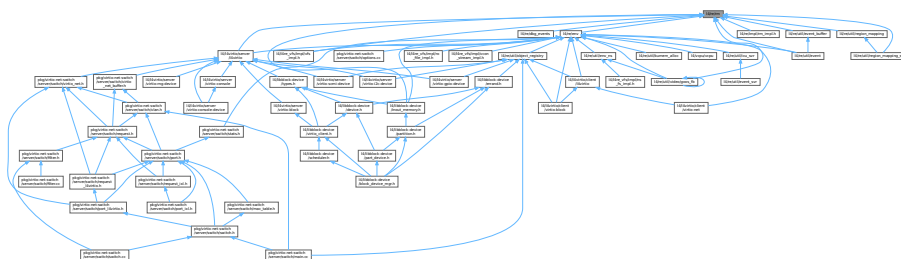
### Region mapper interface.

```
#include <l4/sys/types.h>
#include <l4/sys/l4int.h>
#include <l4/sys/capability>
#include <l4/re/protocols.h>
#include <l4/sys/pager>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/cxx/ipc_ret_array>
#include <l4/sys/cxx/types>
#include <l4/re/consts>
#include <l4/re/dataspace>
```

Include dependency graph for rm:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `L4Re::Rm`  
*Region* map.
- struct `L4Re::Rm::F`  
*Rm* flags definitions.

- class [L4Re::Rm::Unique\\_region< T >](#)  
*Unique region.*
- struct [L4Re::Rm::Region](#)  
*A region is a range of virtual addresses which is backed by content.*
- struct [L4Re::Rm::Area](#)  
*An area is a range of virtual addresses which is reserved, see [L4Re::Rm::reserve\\_area\(\)](#).*

## Namespaces

- namespace [L4Re](#)  
*[L4Re](#) C++ Interfaces.*

## 17.376.1 Detailed Description

Region mapper interface.

Definition in file [rm](#).

## 17.377 rm

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *                Alexander Warg <warg@os.inf.tu-dresden.de>,
00010  *                Björn Döbel <doebel@os.inf.tu-dresden.de>,
00011  *                Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00012  *                economic rights: Technische Universität Dresden (Germany)
00013  *
00014  * License: see LICENSE.spdx (in this directory or the directories above)
00015  */
00016 #pragma once
00017
00018 #include <l4/sys/types.h>
00019 #include <l4/sys/l4int.h>
00020 #include <l4/sys/capability>
00021 #include <l4/re/protocols.h>
00022 #include <l4/sys/pager>
00023 #include <l4/sys/cxx/ipc_iface>
00024 #include <l4/sys/cxx/ipc_ret_array>
00025 #include <l4/sys/cxx/types>
00026 #include <l4/re/consts>
00027 #include <l4/re/dataspace>
00028
00029 namespace L4Re {
00030
00073
00081 class L4_EXPORT Rm :
00082     public L4::Kobject_t<Rm, L4::Pager, L4RE_PROTO_RM,
00083         L4::Type_info::Demand_t<1> >
00084 {
00085 public:
00086     typedef L4Re::Dataspace::Offset Offset;
00087
00089     enum Detach_result
00090     {
00091         Detached_ds    = 0,
00092         Kept_ds        = 1,
00093         Split_ds       = 2,
00094         Detach_result_mask = 3,
00095
00096         Detach_again = 4,
00097     };
00098
00099

```

```

00101 enum Region_flag_shifts
00102 {
00104     Caching_shift    = Dataspace::F::Caching_shift,
00105 };
00106
00108 struct F
00109 {
00111     enum Attach_flags : l4_uint32_t
00112     {
00114         Search_addr    = 0x20000,
00116         In_area        = 0x40000,
00118         Eager_map      = 0x80000,
00120         No_eager_map   = 0x100000,
00122         Attach_mask     = 0x1f0000,
00123     };
00124
00125     L4_TYPES_FLAGS_OPS_DEF(Attach_flags);
00126
00128     enum Region_flags : l4_uint16_t
00129     {
00131         Rights_mask    = 0x0f,
00133         R              = Dataspace::F::R,
00135         W              = Dataspace::F::W,
00137         X              = Dataspace::F::X,
00139         RW             = Dataspace::F::RW,
00141         RX             = Dataspace::F::RX,
00143         RWX            = Dataspace::F::RWX,
00144
00146         Kernel         = 0x100,
00148         Detach_free     = 0x200,
00150         Pager          = 0x400,
00152         Reserved       = 0x800,
00153
00156         Caching_mask   = Dataspace::F::Caching_mask,
00159         Cache_normal    = Dataspace::F::Normal,
00161         Cache_buffered  = Dataspace::F::Bufferable,
00163         Cache_uncached  = Dataspace::F::Uncacheable,
00164
00166         Ds_map_mask     = 0xff,
00167
00169         Region_flags_mask = 0xffff,
00170     };
00171
00172     L4_TYPES_FLAGS_OPS_DEF(Region_flags);
00173
00174     friend constexpr Dataspace::Flags map_flags(Region_flags rf)
00175     {
00176         return Dataspace::Flags(static_cast<l4_uint16_t>(rf) & Ds_map_mask);
00177     }
00178
00179     struct Flags : L4::Types::Flags_ops_t<Flags>
00180     {
00181         l4_uint32_t raw;
00182         Flags() = default;
00183         explicit constexpr Flags(l4_uint32_t f) : raw(f) {}
00184         constexpr Flags(Attach_flags rf) : raw(static_cast<l4_uint32_t>(rf)) {}
00185         constexpr Flags(Region_flags rf) : raw(static_cast<l4_uint32_t>(rf)) {}
00186
00187         friend constexpr Dataspace::Flags map_flags(Flags f)
00188         {
00189             return Dataspace::Flags(f.raw & Ds_map_mask);
00190         }
00191
00192         constexpr Region_flags region_flags() const
00193         {
00194             return Region_flags(raw & Region_flags_mask);
00195         }
00196
00197         constexpr Attach_flags attach_flags() const
00198         {
00199             return Attach_flags(raw & Attach_mask);
00200         }
00201
00202         constexpr bool r() const { return raw & L4_FPAGE_RO; }
00203         constexpr bool w() const { return raw & L4_FPAGE_W; }
00204         constexpr bool x() const { return raw & L4_FPAGE_X; }
00205         constexpr unsigned cap_rights() const
00206         { return w() ? L4_CAP_FPAGE_RW : L4_CAP_FPAGE_RO; }
00207     };
00208
00209     friend constexpr Flags operator | (Region_flags l, Attach_flags r)
00210     { return Flags(l) | Flags(r); }
00211
00212     friend constexpr Flags operator | (Attach_flags l, Region_flags r)
00213     { return Flags(l) | Flags(r); }
00214 };

```

```

00215
00216 using Attach_flags = F::Attach_flags;
00217 using Region_flags = F::Region_flags;
00218 using Flags = F::Flags;
00219
00221 enum Detach_flags
00222 {
00232     Detach_exact    = 1,
00242     Detach_overlap = 2,
00243
00251     Detach_keep = 4,
00252 };
00253
00280 long reserve_area(l4_addr_t *start, unsigned long size,
00281                  Flags flags = Flags(0),
00282                  unsigned char align = L4_PAGESHIFT) const noexcept
00283 { return reserve_area_t::call(c(), start, size, flags, align); }
00284
00285 L4_RPC_NF(long, reserve_area, (L4::Ipc::In_out<l4_addr_t *> start,
00286                               unsigned long size,
00287                               Flags flags,
00288                               unsigned char align));
00289
00305 template< typename T >
00306 long reserve_area(T **start, unsigned long size,
00307                  Flags flags = Flags(0),
00308                  unsigned char align = L4_PAGESHIFT) const noexcept
00309 {
00310     return reserve_area_t::call(c(), reinterpret_cast<l4_addr_t*>(start), size,
00311                                flags, align);
00312 }
00313
00326 L4_RPC(long, free_area, (l4_addr_t addr));
00327
00328 L4_RPC_NF(long, attach, (L4::Ipc::In_out<l4_addr_t *> start,
00329                          unsigned long size, Flags flags,
00330                          L4::Ipc::Opt<L4::Ipc::Cap<Dataspace> > mem,
00331                          Offset offs, unsigned char align,
00332                          L4::Ipc::Opt<l4_cap_idx_t> client_cap,
00333                          L4::Ipc::String<> name, Offset backing_offset));
00334
00335 L4_RPC_NF(long, detach, (l4_addr_t addr, unsigned long size, unsigned flags,
00336                          l4_addr_t &start, l4_addr_t &rsz,
00337                          l4_cap_idx_t &mem_cap));
00338
00397 long attach(l4_addr_t *start, unsigned long size, Flags flags,
00398             L4::Ipc::Cap<Dataspace> mem, Offset offs = 0,
00399             unsigned char align = L4_PAGESHIFT,
00400             L4::Cap<L4::Task> const task
00401             = L4::Cap<L4::Task>::Invalid,
00402             char const *name = nullptr,
00403             Offset backing_offset = 0) const noexcept;
00404
00408 template< typename T >
00409 long attach(T **start, unsigned long size, Flags flags,
00410            L4::Ipc::Cap<Dataspace> mem, Offset offs = 0,
00411            unsigned char align = L4_PAGESHIFT,
00412            L4::Cap<L4::Task> const task
00413            = L4::Cap<L4::Task>::Invalid,
00414            char const *name = nullptr,
00415            Offset backing_offset = 0) const noexcept
00416 {
00417     union X { l4_addr_t a; T* t; };
00418     X *x = reinterpret_cast<X*>(start);
00419     return attach(&x->a, size, flags, mem, offs, align, task,
00420                  name, backing_offset);
00421 }
00422
00423 #if __cplusplus >= 201103L
00434 template< typename T >
00435 class Unique_region
00436 {
00437 private:
00438     T _addr;
00439     L4::Cap<Rm> _rm;
00440
00441 public:
00442     Unique_region(Unique_region const &) = delete;
00443     Unique_region &operator = (Unique_region const &) = delete;
00444
00448     Unique_region() noexcept
00449     : _addr(0), _rm(L4::Cap<Rm>::Invalid) {}
00450
00456     explicit Unique_region(T addr) noexcept
00457     : _addr(addr), _rm(L4::Cap<Rm>::Invalid) {}
00458
00465     Unique_region(T addr, L4::Cap<Rm> const &rm) noexcept

```

```

00466 : _addr(addr), _rm(rm) {}
00467
00473 Unique_region(Unique_region &&o) noexcept : _addr(o.get()), _rm(o._rm)
00474 { o.release(); }
00475
00481 Unique_region &operator = (Unique_region &&o) noexcept
00482 {
00483     if (&o != this)
00484     {
00485         if (_rm.is_valid())
00486             _rm->detach(reinterpret_cast<l4_addr_t>(_addr), 0);
00487         _rm = o._rm;
00488         _addr = o.release();
00489     }
00490     return *this;
00491 }
00492
00498 ~Unique_region() noexcept
00499 {
00500     if (_rm.is_valid())
00501         _rm->detach(reinterpret_cast<l4_addr_t>(_addr), 0);
00502 }
00503
00509 T get() const noexcept
00510 { return _addr; }
00511
00517 T release() noexcept
00518 {
00519     _rm = L4::Cap<Rm>::Invalid;
00520     return _addr;
00521 }
00522
00529 void reset(T addr, L4::Cap<Rm> const &rm) noexcept
00530 {
00531     if (_rm.is_valid())
00532         _rm->detach(l4_addr_t(_addr), 0);
00533
00534     _rm = rm;
00535     _addr = addr;
00536 }
00537
00541 void reset() noexcept
00542 { reset(0, L4::Cap<Rm>::Invalid); }
00543
00549 bool is_valid() const noexcept
00550 { return _rm.is_valid(); }
00551
00553 T operator * () const noexcept { return _addr; }
00554
00556 T operator -> () const noexcept { return _addr; }
00557 };
00558
00559 template< typename T >
00560 long attach(Unique_region<T> *start, unsigned long size, Flags flags,
00561             L4::Ipc::Cap<Dataspace> mem, Offset offs = 0,
00562             unsigned char align = L4_PAGESHIFT,
00563             L4::Cap<L4::Task> const task
00564             = L4::Cap<L4::Task>::Invalid,
00565             char const *name = nullptr,
00566             Offset backing_offset = 0) const noexcept
00567 {
00568     l4_addr_t addr = reinterpret_cast<l4_addr_t>(start->get());
00569
00570     long res = attach(&addr, size, flags, mem, offs, align, task,
00571                     name, backing_offset);
00572     if (res < 0)
00573         return res;
00574
00575     start->reset(reinterpret_cast<T>(addr), L4::Cap<Rm>(cap()));
00576     return res;
00577 }
00578 #endif
00579
00597 int detach(l4_addr_t addr, L4::Cap<Dataspace> *mem,
00598           L4::Cap<L4::Task> const &task = This_task) const noexcept;
00599
00603 int detach(void *addr, L4::Cap<Dataspace> *mem,
00604           L4::Cap<L4::Task> const &task = This_task) const noexcept;
00605
00625 int detach(l4_addr_t start, unsigned long size, L4::Cap<Dataspace> *mem,
00626           L4::Cap<L4::Task> const &task) const noexcept;
00627
00672 long find(l4_addr_t *addr, unsigned long *size, Offset *offset,
00673           L4Re::Rm::Flags *flags, L4::Cap<Dataspace> *m) noexcept
00674 { return find_t::call(c(), addr, size, flags, offset, m); }
00675
00676 L4_RPC_NF(long, find, (L4::Ipc::In_out<l4_addr_t *> addr,

```

```

00677         L4::Ipc::In_out<unsigned long *> size,
00678         L4Re::Rm::Flags *flags, Offset *offset,
00679         L4::Ipc::As_value<L4::Cap<Dataspace> > *m));
00680
00686     struct Region
00687     {
00688         l4_addr_t start;
00689         l4_addr_t end;
00690         F::Region_flags flags;
00691     };
00692
00699     struct Area
00700     {
00701         l4_addr_t start;
00702         l4_addr_t end;
00703     };
00704
00719     L4_RPC(long, get_regions, (l4_addr_t start, L4::Ipc::Ret_array<Region> regions));
00720
00735     L4_RPC(long, get_areas, (l4_addr_t start, L4::Ipc::Ret_array<Area> areas));
00736
00737     L4_RPC(long, get_info, (l4_addr_t addr, L4::Ipc::String<char> &name,
00752         Offset &backing_offset));
00753
00754     int detach(l4_addr_t start, unsigned long size, L4::Cap<Dataspace> *mem,
00755         L4::Cap<L4::Task> task, unsigned flags) const noexcept;
00756
00757     typedef L4::Typeid::Rpcs<attach_t, detach_t, find_t,
00758         reserve_area_t, free_area_t,
00759         get_regions_t, get_areas_t,
00760         get_info_t> Rpcs;
00761 };
00762
00763
00764 inline int
00765 Rm::detach(l4_addr_t addr, L4::Cap<Dataspace> *mem,
00766     L4::Cap<L4::Task> const &task) const noexcept
00767 { return detach(addr, 1, mem, task, Detach_overlap); }
00768
00769 inline int
00770 Rm::detach(void *addr, L4::Cap<Dataspace> *mem,
00771     L4::Cap<L4::Task> const &task) const noexcept
00772 {
00773     return detach(reinterpret_cast<l4_addr_t>(addr), 1, mem, task,
00774         Detach_overlap);
00775 }
00776
00777 inline int
00778 Rm::detach(l4_addr_t addr, unsigned long size, L4::Cap<Dataspace> *mem,
00779     L4::Cap<L4::Task> const &task) const noexcept
00780 { return detach(addr, size, mem, task, Detach_exact); }
00781
00782 };

```

## 17.378 I4/re/rm-sys.h File Reference

Region mapper protocol definitions.

### Namespaces

- namespace [L4Re](#)  
*L4Re C++ Interfaces.*

### Enumerations

- enum [L4Re::Rm::Opcodes](#)  
*Region-map communication-protocol opcodes.*





## Data Structures

- class [L4Re::Util::Cap\\_alloc\\_base](#)  
*Capability allocator.*

## Namespaces

- namespace [L4Re](#)  
*L4Re C++ Interfaces.*
- namespace [L4Re::Util](#)  
*Documentation of the L4 Runtime Environment utility functionality in C++.*

### 17.380.1 Detailed Description

Bitmap capability allocator.

Definition in file [bitmap\\_cap\\_alloc](#).

## 17.381 bitmap\_cap\_alloc

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014
00015 #pragma once
00016
00017 #include <l4/re/util/item_alloc>
00018 #include <l4/sys/assert.h>
00019 #include <l4/sys/capability>
00020 #include <l4/sys/task.h>
00021
00022 namespace L4Re { namespace Util {
00023
00028 class Cap_alloc_base
00029 {
00030 private:
00031     long _bias;
00032     Item_alloc_base _items;
00033
00034 public:
00035     template <unsigned COUNT>
00036     struct Storage
00037     {
00038         typename Bitmap_base::Word<COUNT>::Type _bits[Bitmap_base::Word<COUNT>::Size];
00039     };
00040
00041     enum State { Free = 0, Allocated, Unknown };
00042     Cap_alloc_base(long max, void *mem, long bias = 0, void * = 0)
00043         noexcept : _bias(bias), _items(max, mem) {}
00044
00045     L4::Cap<void> alloc() noexcept
00046     {
00047         long cap = _items.alloc();
00048         if (cap < 0)
00049             return L4::Cap<void>::Invalid;
00050
00051         return L4::Cap<void>((cap + _bias) « L4_CAP_SHIFT);
00052     }
00053
00054     long hint() const { return _items.hint(); }

```

```

00055
00059     template< typename T >
00060     L4::Cap<T> alloc() noexcept
00061     { return L4::Cap<T>(alloc().cap()); }
00062
00063     State is_allocated(L4::Cap<void> c) const noexcept
00064     {
00065         long idx = (c.cap() » L4_CAP_SHIFT);
00066
00067         if (idx < _bias)
00068             return Unknown;
00069
00070         idx -= _bias;
00071         if (idx >= _items.size())
00072             return Unknown;
00073
00074         return _items.is_allocated(idx) ? Allocated : Free;
00075     }
00076
00080     template< typename T>
00081     void free(L4::Cap<T> const &cap, l4_cap_idx_t task = L4_INVALID_CAP,
00082              l4_umword_t unmap_flags = L4_FP_ALL_SPACES) noexcept
00083     {
00084         long idx = (cap.cap() » L4_CAP_SHIFT);
00085         if (idx < _bias)
00086             return;
00087
00088         idx -= _bias;
00089         if (idx >= _items.size())
00090             return;
00091
00092         l4_assert(_items.is_allocated(idx));
00093
00094         if (l4_is_valid_cap(task))
00095             l4_task_unmap(task, cap.fpage(), unmap_flags | 2);
00096
00097         _items.free(idx);
00098     }
00099
00100     // since we have no counters assume counter always > 0
00101     void take(L4::Cap<void>) noexcept {}
00102     bool release(L4::Cap<void>, l4_cap_idx_t /*task*/ = L4_INVALID_CAP,
00103                 unsigned /*unmap_flags*/ = L4_FP_ALL_SPACES) noexcept
00104     { return false; }
00105
00106     long last() noexcept
00107     {
00108         return _items.size() + _bias - 1;
00109     }
00110 };
00111
00112 template< long Size >
00113 class Cap_alloc : public Cap_alloc_base
00114 {
00115 private:
00116     typename Bitmap_base::Word<Size>::Type _bits[Bitmap_base::Word<Size>::Size];
00117
00118 public:
00119     explicit Cap_alloc(long bias = 0) noexcept
00120         : Cap_alloc_base(Size, _bits, bias) {}
00121
00122 };
00123
00124 }
00125 }

```

## 17.382 br\_manager

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <l4/re/util/cap_alloc>
00011 #include <l4/sys/cxx/ipc_server_loop>
00012 #include <l4/cxx/ipc_timeout_queue>
00013 #include <l4/sys/assert.h>
00014
00015 namespace L4Re { namespace Util {

```

```

00016
00025 class Br_manager : public L4::Ipc_svr::Server_iface
00026 {
00027 private:
00028     enum { _mem = 0, _ports = 0 };
00029     enum { Brs_per_timeout = sizeof(l4_kernel_clock_t) / sizeof(l4_umword_t) };
00030
00031 public:
00033     Br_manager() : _caps(0), _cap_flags(L4_RCV_ITEM_LOCAL_ID) {}
00034
00035     Br_manager(Br_manager const &) = delete;
00036     Br_manager &operator = (Br_manager const &) = delete;
00037
00038     Br_manager(Br_manager &&) = delete;
00039     Br_manager &operator = (Br_manager &&) = delete;
00040
00041     ~Br_manager()
00042     {
00043         // Slots for received capabilities are placed at the beginning of the
00044         // (shadowed) buffer registers. Free those.
00045         for (unsigned i = 0; i < _caps; ++i)
00046             cap_alloc.free(L4::Cap<void>(_brs[i] & L4_CAP_MASK));
00047     }
00048
00049     /*
00050     * This implementation dynamically manages assignment of buffer registers for
00051     * the necessary amount of receive buffers allocated by all calls to this
00052     * function.
00053     */
00054     int alloc_buffer_demand(Demand const &d) override
00055     {
00056         using L4::Ipc::Small_buf;
00057
00058         // memory and IO port receive windows currently not supported
00059         if (d.mem || d.ports)
00060             return -L4_EINVAL;
00061
00062         // take extra buffers for a possible timeout and for a zero terminator
00063         if (d.caps + d.mem * 2 + d.ports * 2 + Brs_per_timeout + 1
00064             > L4_UTCB_GENERIC_BUFFERS_SIZE)
00065             return -L4_ERANGE;
00066
00067         if (d.caps > _caps)
00068         {
00069             while (_caps < d.caps)
00070             {
00071                 L4::Cap<void> cap = cap_alloc.alloc();
00072                 if (!cap)
00073                     return -L4_ENOMEM;
00074
00075                 reinterpret_cast<Small_buf*>(_brs[_caps])
00076                     = Small_buf(cap.cap(), _cap_flags);
00077                 ++_caps;
00078             }
00079             _brs[_caps] = 0;
00080         }
00081
00082         return L4_EOK;
00083     }
00084
00085
00086     L4::Cap<void> get_rcv_cap(int i) const override
00087     {
00088         if (i < 0 || i >= _caps)
00089             return L4::Cap<void>::Invalid;
00090
00091         return L4::Cap<void>(_brs[i] & L4_CAP_MASK);
00092     }
00093
00094     int realloc_rcv_cap(int i) override
00095     {
00096         using L4::Ipc::Small_buf;
00097
00098         if (i < 0 || i >= _caps)
00099             return -L4_EINVAL;
00100
00101         L4::Cap<void> cap = cap_alloc.alloc();
00102         if (!cap)
00103             return -L4_ENOMEM;
00104
00105         reinterpret_cast<Small_buf*>(_brs[i])
00106             = Small_buf(cap.cap(), _cap_flags);
00107
00108         return L4_EOK;
00109     }
00110
00118     void set_rcv_cap_flags(unsigned long flags)

```

```

00119 {
00120     l4_assert(_caps == 0);
00121
00122     _cap_flags = flags;
00123 }
00124
00126 int add_timeout(L4::Ipc_svr::Timeout *, l4_kernel_clock_t) override
00127 { return -L4_ENOSYS; }
00128
00130 int remove_timeout(L4::Ipc_svr::Timeout *) override
00131 { return -L4_ENOSYS; }
00132
00134 void setup_wait(l4_utcb_t *utcb, L4::Ipc_svr::Reply_mode)
00135 {
00136     l4_buf_regs_t *br = l4_utcb_br_u(utcb);
00137     br->bdr = 0;
00138     for (unsigned i = 0; i <= _caps; ++i)
00139         br->br[i] = _brs[i];
00140 }
00141
00142 protected:
00144 unsigned first_free_br() const
00145 {
00146     // The last BR (64-bit) or the last two BRs (32-bit); this is constant.
00147     return L4_UTCB_GENERIC_BUFFERS_SIZE - Brs_per_timeout;
00148     // We could also do the following dynamic approach:
00149     // return _caps + _mem + _ports + 1
00150 }
00151
00152 private:
00153     unsigned short _caps;
00154     unsigned long _cap_flags;
00155
00156     l4_umword_t _brs[L4_UTCB_GENERIC_BUFFERS_SIZE];
00157 };
00158
00165 struct Br_manager_hooks
00166 : L4::Ipc_svr::Ignore_errors,
00167   L4::Ipc_svr::Default_timeout,
00168   L4::Ipc_svr::Compound_reply,
00169   Br_manager
00170 {};
00171
00179 struct Br_manager_timeout_hooks :
00180     public L4::Ipc_svr::Timeout_queue_hooks<Br_manager_timeout_hooks, Br_manager>,
00181     public L4::Ipc_svr::Ignore_errors
00182 {
00183     public:
00184         static l4_kernel_clock_t now()
00185         { return l4_kip_clock(l4re_kip()); }
00186 };
00187
00188 {}
00189

```

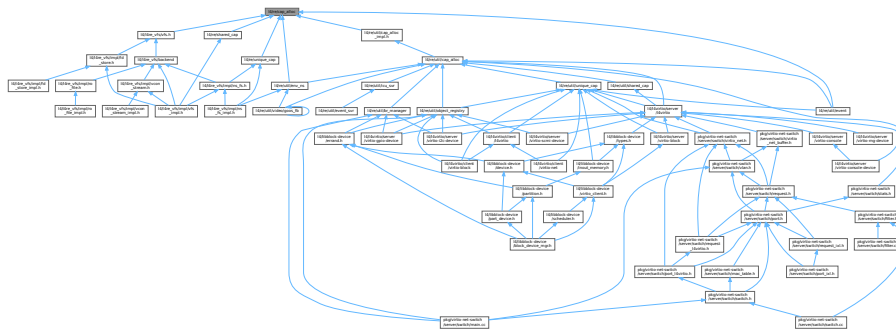
## 17.383 l4/re/util/cap File Reference

Capability utility functions.





This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4Re::Cap\\_alloc](#)  
*Capability allocator interface.*
- class [L4Re::Smart\\_cap\\_auto< Unmap\\_flags >](#)  
*Helper for [Unique\\_cap](#) and [Unique\\_del\\_cap](#).*
- class [L4Re::Smart\\_count\\_cap< Unmap\\_flags >](#)  
*Helper for [Ref\\_cap](#) and [Ref\\_del\\_cap](#).*

## Namespaces

- namespace [L4Re](#)  
*[L4Re](#) C++ Interfaces.*

## 17.385.1 Detailed Description

Abstract capability-allocator interface.

Definition in file [cap\\_alloc](#).

## 17.386 cap\_alloc

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010 #include <l4/sys/task>
00011 #include <l4/sys/smart_capability>
00012 #include <l4/re/consts>
00013 #include <l4/cxx/type_traits>
00014 namespace L4Re {
00015
```



```

00030 class Cap_alloc
00031 {
00032 private:
00033     void operator = (Cap_alloc const &);
00034
00035 protected:
00036     Cap_alloc(Cap_alloc const &) {}
00037     Cap_alloc() {}
00038
00039 public:
00040
00041     virtual L4::Cap<void> alloc() noexcept = 0;
00042     virtual void take(L4::Cap<void> cap) noexcept = 0;
00043
00044     template< typename T >
00045     L4::Cap<T> alloc() noexcept
00046     { return L4::cap_cast<T>(alloc()); }
00047
00048     virtual void free(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
00049                      unsigned unmap_flags = L4_FP_ALL_SPACES) noexcept = 0;
00050     virtual bool release(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
00051                        unsigned unmap_flags = L4_FP_ALL_SPACES) noexcept = 0;
00052
00053     virtual ~Cap_alloc() = 0;
00054 };
00055
00056 template<typename ALLOC>
00057 struct Cap_alloc_t : ALLOC, L4Re::Cap_alloc
00058 {
00059     template<typename ...ARGS>
00060     Cap_alloc_t(ARGS &&...args) : ALLOC(cxx::forward<ARGS>(args)...) {}
00061
00062     L4::Cap<void> alloc() noexcept override { return ALLOC::alloc(); }
00063     void take(L4::Cap<void> cap) noexcept override { ALLOC::take(cap); }
00064
00065     template <typename T>
00066     L4::Cap<T> alloc() noexcept
00067     {
00068         return L4::cap_cast<T>(alloc());
00069     }
00070
00071     void free(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
00072              unsigned unmap_flags = L4_FP_ALL_SPACES) noexcept override
00073     { ALLOC::free(cap, task, unmap_flags); }
00074
00075     bool release(L4::Cap<void> cap, l4_cap_idx_t task,
00076                 unsigned unmap_flags) noexcept override
00077     { return ALLOC::release(cap, task, unmap_flags); }
00078
00079     void operator delete(void *) {}
00080 };
00081
00082 inline
00083 Cap_alloc::~~Cap_alloc()
00084 {}
00085
00086 extern Cap_alloc *virt_cap_alloc;
00087
00088 template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00089 class Smart_cap_auto
00090 {
00091 private:
00092     Cap_alloc *_ca;
00093
00094 public:
00095     Smart_cap_auto() : _ca(0) {}
00096     Smart_cap_auto(Cap_alloc *ca) : _ca(ca) {}
00097
00098     void free(L4::Cap_base &c)
00099     {
00100         if (c.is_valid() && _ca)
00101             _ca->free(L4::Cap<void>(c.cap()), This_task, Unmap_flags);
00102         invalidate(c);
00103     }
00104
00105     static void invalidate(L4::Cap_base &c)
00106     {
00107         if (c.is_valid())
00108             c.invalidate();
00109     }
00110 };
00111
00112 template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00113 class Smart_count_cap
00114 {

```



[illegible]

- class `L4Re::Util::Smart_cap_auto< Unmap_flags >`  
*Helper for `Unique_cap` and `Unique_del_cap`.*
- class `L4Re::Util::Smart_count_cap< Unmap_flags >`  
*Helper for `Ref_cap` and `Ref_del_cap`.*
- struct `L4Re::Util::Ref_cap< T >`  
*Automatic capability that implements automatic free and unmap of the capability selector.*
- struct `L4Re::Util::Ref_del_cap< T >`  
*Automatic capability that implements automatic free and unmap+delete of the capability selector.*

- namespace [L4Re](#)  
[L4Re C++ Interfaces](#).
- namespace [L4Re::Util](#)  
*Documentation of the [L4 Runtime Environment](#) utility functionality in C++.*

- `template<typename T>`  
`Ref_cap< T >::Cap L4Re::Util::make_ref_cap ()`  
*Allocate a capability slot and wrap it in a `Ref_cap`.*
- `template<typename T>`  
`Ref_del_cap< T >::Cap L4Re::Util::make_ref_del_cap ()`  
*Allocate a capability slot and wrap it in a `Ref_del_cap`.*

- `_Cap_alloc` & `L4Re::Util::cap_alloc`  
*Capability allocator.*

## 17.387.1 Detailed Description

Capability allocator.

Definition in file [cap\\_alloc](#).

## 17.388 cap\_alloc

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00004  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00005  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00006  *      economic rights: Technische Universität Dresden (Germany)
00007  *
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010
00011 #pragma once
00012
00013 #include <l4/re/util/cap_alloc_impl.h>
00014 #include <l4/sys/smart_capability>
00015 #include <l4/sys/task>
00016 #include <l4/re/consts>
00017
00018 namespace L4Re { namespace Util {
00019
00020     extern _Cap_alloc &cap_alloc;
00021
00022     template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00023     class Smart_cap_auto
00024     {
00025     public:
00026         static void free(L4::Cap_base &c)
00027         {
00028             if (c.is_valid())
00029             {
00030                 cap_alloc.free(L4::Cap<void>(c.cap()), This_task, Unmap_flags);
00031                 c.invalidate();
00032             }
00033         }
00034
00035         static void invalidate(L4::Cap_base &c)
00036         {
00037             if (c.is_valid())
00038                 c.invalidate();
00039         }
00040     };
00041
00042     template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00043     class Smart_count_cap
00044     {
00045     public:
00046         static void free(L4::Cap_base &c) noexcept
00047         {
00048             if (c.is_valid())
00049             {
00050                 if (cap_alloc.release(L4::Cap<void>(c.cap()), This_task, Unmap_flags))
00051                     c.invalidate();
00052             }
00053         }
00054
00055         static void invalidate(L4::Cap_base &c) noexcept
00056         {
00057             if (c.is_valid())
00058                 c.invalidate();
00059         }
00060
00061         static L4::Cap_base copy(L4::Cap_base const &src)
00062         {
00063             cap_alloc.take(L4::Cap<void>(src.cap()));
00064             return src;
00065         }
00066     };
00067
00068 };
00069
00070
00071
00072
00073
00074
00075
00076
00077
00078
00079
00080
00081
00082
00083
00084
00085
00086
00087
00088
00089
00090
00091
00092
00093
00094
00095
00096
00097
00098
00099
00100
00101
00102
00103
00104
00105
00106
00107
00108
00109
00110

```

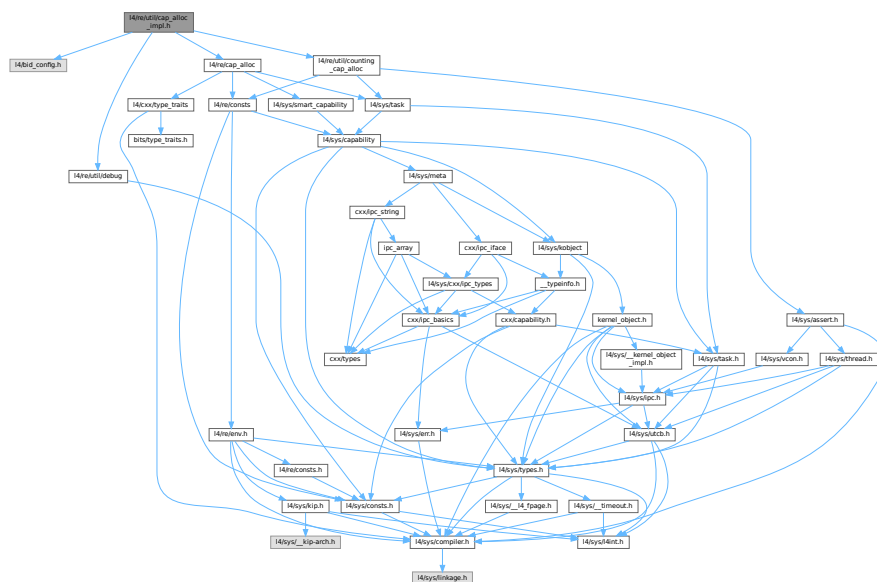
```
00111
00141 template< typename T >
00142 struct Ref_cap
00143 {
00144     typedef L4::Smart_cap<T, Smart_count_cap<L4_FP_ALL_SPACES> > Cap;
00145 };
00146
00182 template< typename T >
00183 struct Ref_del_cap
00184 {
00185     typedef L4::Smart_cap<T, Smart_count_cap<L4_FP_DELETE_OBJ> > Cap;
00186 };
00187
00193 template< typename T >
00194 typename Ref_cap<T>::Cap
00195 make_ref_cap() { return typename Ref_cap<T>::Cap(cap_alloc.alloc<T>()); }
00196
00202 template< typename T >
00203 typename Ref_del_cap<T>::Cap
00204 make_ref_del_cap()
00205 { return typename Ref_del_cap<T>::Cap(cap_alloc.alloc<T>()); }
00206
00208
00209 }}
00210
```

## 17.389 l4/re/util/cap\_alloc\_impl.h File Reference

### Capability allocator implementation.

```
#include <l4/bid_config.h>
#include <l4/re/cap_alloc>
#include <l4/re/util/counting_cap_alloc>
#include <l4/re/util/debug>
```

Include dependency graph for cap\_alloc\_impl.h:





```

00023 namespace L4Re { namespace Util {
00024
00025 using _Cap_alloc_impl = Cap_alloc_base;
00026
00027 }}
00028
00029 #elif defined(CONFIG_L4RE_COUNTING_CAP_ALLOC)
00030
00031 #include <l4/re/util/counting_cap_alloc>
00032 #include <l4/re/util/debug>
00033
00034 namespace L4Re { namespace Util {
00035
00036 // RISC-V does not natively support subword atomics, such as __atomic_load_1.
00037 // The RISC-V gcc developers have decided to emulate these via libatomic, which
00038 // is automatically linked against.
00039 #if defined(__GCC_HAVE_SYNC_COMPARE_AND_SWAP_1) || defined(ARCH_arm) || defined(ARCH_riscv)
00040 using _Cap_alloc_impl
00041     = Counting_cap_alloc<L4Re::Util::Counter_atomic<unsigned char>,
00042                         L4Re::Util::Dbg>;
00043 #elif defined(ARCH_sparc)
00044 using _Cap_alloc_impl
00045     = Counting_cap_alloc<L4Re::Util::Counter<unsigned char>,
00046                         L4Re::Util::Dbg>;
00047 #warning "Thread-safe capability allocator not available!"
00048 #else
00049 #error "Unsupported platform"
00050 #endif
00051
00052 }}
00053
00054 #else
00055 #error "No supported capability allocator selected"
00056 #endif
00057
00058 namespace L4Re { namespace Util {
00059
00060 class _Cap_alloc final : public L4Re::Cap_alloc, private _Cap_alloc_impl
00061 {
00062 public:
00063     template <unsigned COUNT>
00064     using Storage = _Cap_alloc_impl::Storage<COUNT>;
00065
00066     using _Cap_alloc_impl::_Cap_alloc_impl; // Expose underlying constructor
00067     void operator delete(void *) {} // Prevent global operator delete reference
00068
00069     L4::Cap<void> alloc() noexcept override
00070     { return _Cap_alloc_impl::alloc(); }
00071
00072     template< typename T >
00073     L4::Cap<T> alloc() noexcept
00074     { return L4::cap_cast<T>(alloc()); }
00075
00076     void take(L4::Cap<void> cap) noexcept override
00077     { _Cap_alloc_impl::take(cap); }
00078
00079     void free(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
00080              unsigned unmap_flags = L4_FP_ALL_SPACES) noexcept override
00081     { _Cap_alloc_impl::free(cap, task, unmap_flags); }
00082
00083     bool release(L4::Cap<void> cap, l4_cap_idx_t task,
00084                 unsigned unmap_flags) noexcept override
00085     { return _Cap_alloc_impl::release(cap, task, unmap_flags); }
00086
00087     using _Cap_alloc_impl::last;
00088 };
00089
00090
00091
00092
00093
00094
00095
00096

```

## 17.391 l4/re/util/counting\_cap\_alloc File Reference

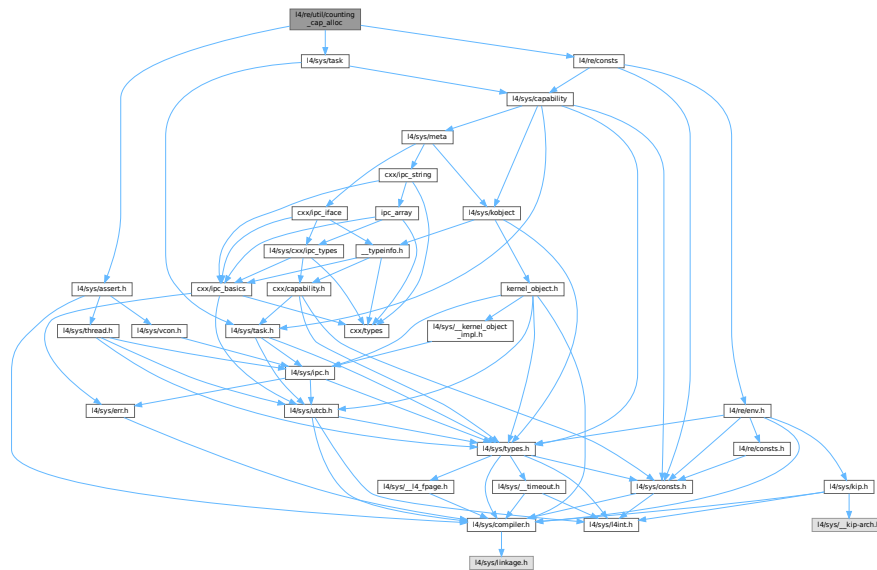
Reference-counting capability allocator.

```

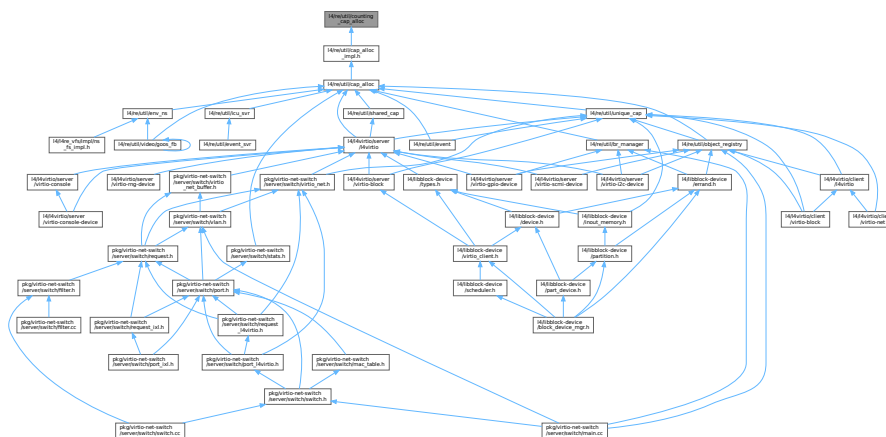
#include <l4/sys/task>
#include <l4/sys/assert.h>

```

```
#include <linux/re/consts>
Include dependency graph for counting_cap_alloc:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `L4Re::Util::Counter< COUNTER >`  
*Counter for `Counting_cap_alloc` with variable data width.*
- struct `L4Re::Util::Counter_atomic< COUNTER >`  
*Thread safe version of counter for `Counting_cap_alloc`.*
- class `L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >`  
*Internal reference-counting cap allocator.*



## Namespaces

- namespace [L4Re](#)  
*[L4Re](#) C++ Interfaces.*
- namespace [L4Re::Util](#)  
*Documentation of the [L4](#) Runtime Environment utility functionality in C++.*

### 17.391.1 Detailed Description

Reference-counting capability allocator.

Definition in file [counting\\_cap\\_alloc](#).

## 17.392 counting\_cap\_alloc

[Go to the documentation of this file.](#)

```

00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012
00013 #pragma once
00014
00015 #include <l4/sys/task>
00016 #include <l4/sys/assert.h>
00017 #include <l4/re/consts>
00018
00019 namespace L4Re { namespace Util {
00020
00026 template< typename COUNTER = unsigned char >
00027 struct Counter
00028 {
00029     typedef COUNTER Type;
00030     Type _cnt;
00031
00032     static Type nil() { return 0; }
00033     static Type unused() { return 0; }
00034
00035     void free() { _cnt = 0; }
00036     bool is_free() const { return _cnt == 0; }
00037     bool is_saturated() const { return static_cast<Type>(_cnt + 1) == 0; }
00038
00050     bool inc()
00051     {
00052         if (is_saturated())
00053             return true; // no change and no warning
00054         ++_cnt;
00055         if (is_saturated())
00056             return false; // warn caller that counter is now saturated
00057         else
00058             return true; // success
00059     }
00060
00067     Type dec()
00068     {
00069         if (is_saturated())
00070             return _cnt; // no change
00071         else
00072             return --_cnt; // success
00073     }
00074
00075     bool try_alloc()
00076     {
00077         if (_cnt == 0)
00078         {
00079             _cnt = 1;
00080             return true;
00081         }
00082         return false;

```

```

00083     }
00084 };
00085
00097 template< typename COUNTER = unsigned char >
00098 struct Counter_atomic
00099 {
00100     typedef COUNTER Type;
00101     Type _cnt;
00102
00103     static Type nil() { return 0; }
00104     static Type unused() { return 1; }
00105
00106     bool is_free() const { return __atomic_load_n(&_cnt, __ATOMIC_RELAXED) == 0; }
00107     static bool is_saturated(Type cnt) { return static_cast<Type>(cnt + 1) == 0; }
00108
00109     bool try_alloc()
00110     {
00111         Type expected = nil();
00112         // Use "acquire" memory ordering. Any operations tied to the capability slot
00113         // must only be observable after the slot has been occupied.
00114         return __atomic_compare_exchange_n(&_cnt, &expected, 2, false,
00115                                           __ATOMIC_ACQUIRE, __ATOMIC_RELAXED);
00116     }
00117
00121     bool inc()
00122     {
00123         Type old_cnt = __atomic_load_n(&_cnt, __ATOMIC_RELAXED);
00124         Type new_cnt;
00125         do
00126         {
00127             if (is_saturated(old_cnt))
00128                 return true; // no change and no warning
00129             new_cnt = old_cnt + 1;
00130         }
00131         while (!__atomic_compare_exchange_n(&_cnt, &old_cnt, new_cnt, false,
00132                                           __ATOMIC_RELAXED, __ATOMIC_RELAXED));
00133         if (is_saturated(new_cnt))
00134             return false; // warn caller that counter is now saturated
00135         else
00136             return true; // success
00137     }
00138
00142     Type dec()
00143     {
00144         Type old_cnt = __atomic_load_n(&_cnt, __ATOMIC_RELAXED);
00145         Type new_cnt;
00146         do
00147         {
00148             if (is_saturated(old_cnt))
00149                 return old_cnt; // no change
00150             new_cnt = old_cnt - 1;
00151         }
00152         while (!__atomic_compare_exchange_n(&_cnt, &old_cnt, new_cnt, false,
00153                                           __ATOMIC_RELAXED, __ATOMIC_RELAXED));
00154         return new_cnt; // success
00155     }
00156
00157     void free()
00158     {
00159         // Use "release" memory ordering to make sure that any operations tied to
00160         // the capability slot are observable by other threads before the slot can
00161         // be reused.
00162         __atomic_store_n(&_cnt, 0, __ATOMIC_RELEASE);
00163     }
00164 };
00165
00190 template <typename COUNTERTYPE, typename Dbg>
00191 class Counting_cap_alloc
00192 {
00193 private:
00194     void operator = (Counting_cap_alloc const &) { }
00195     typedef COUNTERTYPE Counter;
00196
00197     COUNTERTYPE *_items;
00198     long _free_hint;
00199     long _bias;
00200     long _capacity;
00201     Dbg *_dbg;
00202
00203 public:
00204
00205     template <unsigned COUNT>
00206     struct Storage
00207     {
00208         COUNTERTYPE _buf[COUNT];
00209         typedef COUNTERTYPE Buf_type[COUNT];
00210         enum { Size = COUNT };

```

```

00211     };
00212
00213     Counting_cap_alloc(long capacity, void *m, long bias, Dbg *dbg) noexcept
00214     : _items((Counter*)m), _free_hint(0), _bias(bias), _capacity(capacity),
00215       _dbg(dbg)
00216     {}
00217
00218 protected:
00224     Counting_cap_alloc() noexcept
00225     : _items(0), _free_hint(0), _bias(0), _capacity(0)
00226     {}
00227
00242     void setup(void *m, long capacity, long bias, Dbg *dbg) noexcept
00243     {
00244         _items = static_cast<Counter*>(m);
00245         _capacity = capacity;
00246         _bias = bias;
00247         _dbg = dbg;
00248     }
00249
00250 public:
00257     L4::Cap<void> alloc() noexcept
00258     {
00259         long free_hint = __atomic_load_n(&_free_hint, __ATOMIC_RELAXED);
00260
00261         for (long i = free_hint; i < _capacity; ++i)
00262             if (_items[i].try_alloc())
00263             {
00264                 _free_hint = i + 1;
00265                 return L4::Cap<void>((i + _bias) « L4_CAP_SHIFT);
00266             }
00267
00268         // _free_hint is not necessarily correct in case of multi-threading! Make
00269         // sure we don't miss any potentially free slots.
00270         for (long i = 0; i < free_hint && i < _capacity; ++i)
00271             if (_items[i].try_alloc())
00272             {
00273                 _free_hint = i + 1;
00274                 return L4::Cap<void>((i + _bias) « L4_CAP_SHIFT);
00275             }
00276
00277         return L4::Cap<void>::Invalid;
00278     }
00279
00281     template <typename T>
00282     L4::Cap<T> alloc() noexcept
00283     {
00284         return L4::cap_cast<T>(alloc());
00285     }
00286
00287
00296     void take(L4::Cap<void> cap) noexcept
00297     {
00298         long c;
00299         if (!range_check_and_get_idx(cap, &c))
00300             return;
00301
00302         if (!L4_UNLIKELY(_items[c].inc()))
00303             _dbg->printf("Warning: Reference counter of cap 0x%lx now saturated!\n",
00304                         cap.cap() » L4_CAP_SHIFT);
00305     }
00306
00307
00321     bool free(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
00322              unsigned unmap_flags = L4_FP_ALL_SPACES) noexcept
00323     {
00324         long c;
00325         if (!range_check_and_get_idx(cap, &c))
00326             return false;
00327
00328         l4_assert(!_items[c].is_free());
00329
00330         if (l4_is_valid_cap(task))
00331             l4_task_unmap(task, cap.fpage(), unmap_flags);
00332
00333         if (c < _free_hint)
00334             _free_hint = c;
00335
00336         _items[c].free();
00337
00338         return true;
00339     }
00340
00359     bool release(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
00360                 unsigned unmap_flags = L4_FP_ALL_SPACES) noexcept
00361     {
00362         long c;

```

```

00363     if (!range_check_and_get_idx(cap, &c))
00364         return false;
00365
00366     l4_assert(!_items[c].is_free());
00367
00368     if (_items[c].dec() == Counter::unused())
00369     {
00370         if (task != L4_INVALID_CAP)
00371             l4_task_unmap(task, cap.fpage(), unmap_flags);
00372
00373         if (c < _free_hint)
00374             _free_hint = c;
00375
00376         // Let others allocate this slot only after the l4_task_unmap() has
00377         // finished.
00378         _items[c].free();
00379
00380         return true;
00381     }
00382     return false;
00383 }
00384
00388 long last() noexcept
00389 {
00390     return _capacity + _bias - 1;
00391 }
00392
00393 private:
00394     bool range_check_and_get_idx(L4::Cap<void> cap, long *c)
00395     {
00396         *c = cap.cap() » L4_CAP_SHIFT;
00397         if (*c < _bias)
00398             return false;
00399
00400         *c -= _bias;
00401
00402         return *c < _capacity;
00403     }
00404 };
00405
00406 }}

```

## 17.393 dataspace\_svr

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00005  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00006  *      economic rights: Technische Universität Dresden (Germany)
00007  *
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010 #pragma once
00011
00012 #include <string.h>
00013 #include <stddef.h>
00014 #include <l4/bid_config.h>
00015 #include <l4/sys/types.h>
00016 #include <l4/cxx/minmax>
00017 #include <l4/re/dataspace>
00018 #include <l4/re/dataspace-sys.h>
00019 #include <l4/sys/cxx/ipc_legacy>
00020
00021 namespace L4Re { namespace Util {
00022
00029 class Dataspace_svr
00030 {
00031 public:
00032     L4_RPC_LEGACY_DISPATCH(L4Re::Dataspace);
00033
00034     typedef L4::Ipc::Snd_fpage::Map_type Map_type;
00035     typedef L4::Ipc::Snd_fpage::Cacheopt Cache_type;
00036
00037     Dataspace_svr() noexcept
00038     : _ds_start(0), _ds_size(0), _map_flags(L4::Ipc::Snd_fpage::Map),
00039       _cache_flags(L4::Ipc::Snd_fpage::Cached)
00040     {}
00041
00042     virtual ~Dataspace_svr() noexcept {}
00043
00057     int map(Dataspace::Offset offset,
00058            Dataspace::Map_addr local_addr,

```

```

00059         Dataspace::Flags flags,
00060         Dataspace::Map_addr min_addr,
00061         Dataspace::Map_addr max_addr,
00062         L4::Ipc::Snd_fpage &memory)
00063     {
00064         memory = L4::Ipc::Snd_fpage();
00065
00066         offset = l4_trunc_page(offset);
00067         local_addr = l4_trunc_page(local_addr);
00068
00069         if (!check_limit(offset))
00070         {
00071             #if 0
00072                 printf("limit failed: off=%lx sz=%lx\n", offset, size());
00073             #endif
00074             return -L4_ERANGE;
00075         }
00076
00077         min_addr = l4_trunc_page(min_addr);
00078         max_addr = l4_round_page(max_addr);
00079
00080         l4_addr_t addr = _ds_start + offset;
00081         unsigned char order = L4_PAGESHIFT;
00082
00083         while (order < 30 /* limit to 1GB flexpage */)
00084         {
00085             l4_addr_t map_base = l4_trunc_size(addr, order + 1);
00086             if (map_base < _ds_start)
00087                 break;
00088
00089             if (map_base + (1UL << (order + 1)) - 1 > (_ds_start + round_size() - 1))
00090                 break;
00091
00092             map_base = l4_trunc_size(local_addr, order + 1);
00093             if (map_base < min_addr)
00094                 break;
00095
00096             if (map_base + (1UL << (order + 1)) - 1 > max_addr - 1)
00097                 break;
00098
00099             l4_addr_t mask = ~(~0UL << (order + 1));
00100             if (local_addr == ~0UL || ((addr ^ local_addr) & mask))
00101                 break;
00102
00103             ++order;
00104         }
00105
00106         l4_addr_t map_base = l4_trunc_size(addr, order);
00107
00108         Dataspace::Map_addr b = map_base;
00109         unsigned send_order = order;
00110         int err = map_hook(offset /*map_base - _ds_start*/, order, flags,
00111                             &b, &send_order);
00112         if (err < 0)
00113             return err;
00114
00115         l4_fpage_t fpage = l4_fpage(b, send_order, flags.fpage_rights());
00116
00117         memory = L4::Ipc::Snd_fpage(fpage, local_addr, _map_flags, _cache_flags);
00118
00119         return L4_EOK;
00120     }
00121
00136     virtual int map_hook([[maybe_unused]] Dataspace::Offset offs,
00137                          [[maybe_unused]] unsigned order,
00138                          [[maybe_unused]] Dataspace::Flags flags,
00139                          [[maybe_unused]] Dataspace::Map_addr *base,
00140                          [[maybe_unused]] unsigned *send_order)
00141     {
00142         return 0;
00143     }
00144
00150     virtual void take() noexcept
00151     {}
00152
00160     virtual unsigned long release() noexcept
00161     { return 0; }
00162
00175     virtual long copy([[maybe_unused]] l4_addr_t dst_offs,
00176                      [[maybe_unused]] l4_umword_t src_id,
00177                      [[maybe_unused]] l4_addr_t src_offs,
00178                      [[maybe_unused]] unsigned long size) noexcept
00179     {
00180         return -L4_ENODEV;
00181     }
00182
00192     virtual long clear(unsigned long offs, unsigned long size) const noexcept

```

```

00193 {
00194     if (!check_limit(offsets))
00195         return -L4_ERANGE;
00196
00197     unsigned long sz = size = cxx::min(size, round_size() - offsets);
00198
00199     while (sz)
00200     {
00201         unsigned long b_addr = _ds_start + offsets;
00202         unsigned long b_sz = cxx::min(size - offsets, sz);
00203
00204         memset(reinterpret_cast<void *>(b_addr), 0, b_sz);
00205
00206         offsets += b_sz;
00207         sz -= b_sz;
00208     }
00209
00210     return 0;
00211 }
00212
00224 virtual long allocate([[maybe_unused]] l4_addr_t offset,
00225                      [[maybe_unused]] l4_size_t size,
00226                      [[maybe_unused]] unsigned access) noexcept
00227 {
00228     return -L4_ENODEV;
00229 }
00230
00236 virtual unsigned long page_shift() const noexcept
00237 { return L4_LOG2_PAGESIZE; }
00238
00244 virtual bool is_static() const noexcept
00245 { return true; }
00246
00259 virtual long map_info([[maybe_unused]] l4_addr_t &start_addr,
00260                      [[maybe_unused]] l4_addr_t &end_addr) noexcept
00261 { return -L4_EPERM; }
00262
00263
00264 long op_map(L4Re::Dataspace::Rights rights,
00265            L4Re::Dataspace::Offset offset,
00266            L4Re::Dataspace::Map_addr spot,
00267            L4Re::Dataspace::Flags flags,
00268            L4::Ipc::Snd_fpage &fp)
00269 {
00270     auto rf = map_flags(rights);
00271
00272     if (!rf.w() && flags.w())
00273         return -L4_EPERM;
00274
00275     return map(offset, spot, flags & rf, 0, ~0, fp);
00276 }
00277
00278 long op_allocate(L4Re::Dataspace::Rights rights,
00279                 L4Re::Dataspace::Offset offset,
00280                 L4Re::Dataspace::Size size)
00281 { return allocate(offset, size, rights & 3); }
00282
00283 long op_copy_in(L4Re::Dataspace::Rights rights,
00284                L4Re::Dataspace::Offset dst_offs,
00285                L4::Ipc::Snd_fpage const &src_cap,
00286                L4Re::Dataspace::Offset src_offs,
00287                L4Re::Dataspace::Size sz)
00288 {
00289     if (!src_cap.id_received())
00290         return -L4_EINVAL;
00291
00292     if (!(rights & L4_CAP_FPAGE_W))
00293         return -L4_EACCESS;
00294
00295     if (sz == 0)
00296         return L4_EOK;
00297
00298     return copy(dst_offs, src_cap.data(), src_offs, sz);
00299 }
00300
00301 long op_info(L4Re::Dataspace::Rights rights, L4Re::Dataspace::Stats &s)
00302 {
00303     s.size = size();
00304     // only return writable if really writable
00305     s.flags = Dataspace::Flags(0);
00306     if (map_flags(rights).w())
00307         s.flags |= Dataspace::F::W;
00308     return L4_EOK;
00309 }
00310
00311 long op_clear(L4Re::Dataspace::Rights rights,
00312              L4Re::Dataspace::Offset offset,

```

```

00313         L4Re::Dataspace::Size size)
00314     {
00315         if (!map_flags(rights).w())
00316             return -L4_EACCESS;
00317
00318         return clear(offset, size);
00319     }
00320
00321     long op_map_info(L4Re::Dataspace::Rights,
00322                     [[maybe_unused]] l4_addr_t &start_addr,
00323                     [[maybe_unused]] l4_addr_t &end_addr)
00324     {
00325 #ifdef CONFIG_MMU
00326         return 0;
00327 #else
00328         return map_info(start_addr, end_addr);
00329 #endif
00330     }
00331
00332 protected:
00333     unsigned long size() const noexcept
00334     { return _ds_size; }
00335     unsigned long map_flags() const noexcept
00336     { return _map_flags; }
00337     unsigned long page_size() const noexcept
00338     { return 1UL < page_shift(); }
00339     unsigned long round_size() const noexcept
00340     { return l4_round_size(size(), page_shift()); }
00341     bool check_limit(l4_addr_t offset) const noexcept
00342     { return offset < round_size(); }
00343
00344     L4Re::Dataspace::Flags
00345     map_flags(L4Re::Dataspace::Rights rights = L4_CAP_FPAGE_W) const noexcept
00346     {
00347         auto f = (_rw_flags & L4Re::Dataspace::Flags(0x0f)) | L4Re::Dataspace::F::Caching_mask;
00348         if (!(rights & L4_CAP_FPAGE_W))
00349             f &= ~L4Re::Dataspace::F::W;
00350
00351         return f;
00352     }
00353
00354 protected:
00355     void size(unsigned long size) noexcept { _ds_size = size; }
00356
00357     l4_addr_t _ds_start;
00358     l4_size_t _ds_size;
00359     Map_type _map_flags;
00360     Cache_type _cache_flags;
00361     L4Re::Dataspace::Flags _rw_flags;
00362 };
00363
00364 }}

```

## 17.394 l4/re/debug File Reference

Debug interface.

```

#include <l4/sys/capability>
#include <l4/re/protocols.h>
#include <l4/sys/cxx/ipc_iface>

```





## 17.395 debug

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00015 #include <l4/sys/capability>
00016 #include <l4/re/protocols.h>
00017 #include <l4/sys/cxx/ipc_iface>
00018
00019 namespace L4Re {
00033
00040 class L4_EXPORT Debug_obj :
00041     public L4::Kobject_t<Debug_obj, L4::Kobject, L4RE_PROTO_DEBUG>
00042 {
00043 public:
00044
00056     L4_INLINE_RPC(long, debug, (unsigned long function));
00057     typedef L4::Typeid::Rpc_nocode<debug_t> Rpcs;
00058 };
00059
00060 template<typename BASE>
00061 class Debug_obj_t :
00062     public L4::Kobject_2t<Debug_obj_t<BASE>, BASE, Debug_obj, L4::PROTO_EMPTY>
00063 {
00064     typedef L4::Typeid::Rpcs<> Rpcs;
00065 };
00066 }
```

## 17.396 debug

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #pragma once
00015
00016 #include <l4/sys/types.h>
00017
00018 namespace L4Re { namespace Util {
00019 class Err
00020 {
00021 public:
00022     enum Level
00023     {
00024         Normal = 0,
00025         Fatal,
00026     };
00027
00028     static char const *const levels[];
00029
00030     void tag() const
00031     { printf("%s: %s", _component, levels[_l]); }
00032
00033     int printf(char const *fmt, ...) const
00034         __attribute__((format(printf,2,3)));
00035
00036     int cprintf(char const *fmt, ...) const
00037         __attribute__((format(printf,2,3)));
00038
00039     constexpr Err(Level l, char const *component) : _l(l), _component(component)
00040     {}
00041
00042 private:
00043     Level _l;
00044     char const *_component;
00045 };
00046
00047 }
```

```

00048 class Dbg
00049 {
00050 private:
00051     void tag() const;
00052
00053 #ifndef NDEBUG
00054
00055     unsigned long _m;
00056     char const *const _component;
00057     char const *const _subsys;
00058
00059 # ifnndef __clang__
00060
00061     int printf_impl(char const *fmt, ...) const
00062         __attribute__((format(printf, 2, 3)));
00063
00064     int cprintf_impl(char const *fmt, ...) const
00065         __attribute__((format(printf, 2, 3)));
00066
00067 # endif
00068
00069 public:
00070     static unsigned long level;
00071
00072     static void set_level(unsigned long l) { level = l; }
00073
00074     bool is_active() const { return _m & level; }
00075
00076 # ifdef __clang__
00077
00078     int printf(char const *fmt, ...) const
00079         __attribute__((format(printf, 2, 3)));
00080
00081     int cprintf(char const *fmt, ...) const
00082         __attribute__((format(printf, 2, 3)));
00083
00084 # else
00085
00086     int __attribute__((always_inline, format(printf, 2, 3)))
00087     printf(char const *fmt, ...) const
00088     {
00089         if (!(level & _m))
00090             return 0;
00091
00092         return printf_impl(fmt, __builtin_va_arg_pack());
00093     }
00094
00095     int __attribute__((always_inline, format(printf, 2, 3)))
00096     cprintf(char const *fmt, ...) const
00097     {
00098         if (!(level & _m))
00099             return 0;
00100
00101         return cprintf_impl(fmt, __builtin_va_arg_pack());
00102     }
00103
00104 # endif
00105
00106     explicit constexpr
00107     Dbg() : _m(1), _component(0), _subsys(0) {}
00108
00109     explicit constexpr
00110     Dbg(unsigned long mask, char const *comp, char const *subs)
00111     : _m(mask), _component(comp), _subsys(subs)
00112     {}
00113
00114 #else
00115
00116 public:
00117     static void set_level(unsigned long) {}
00118     bool is_active() const { return false; }
00119
00120     int printf(char const * /*fmt*/, ...) const
00121         __attribute__((format(printf, 2, 3)))
00122     { return 0; }
00123
00124     int cprintf(char const * /*fmt*/, ...) const
00125         __attribute__((format(printf, 2, 3)))
00126     { return 0; }
00127
00128     explicit constexpr
00129     Dbg() {}
00130
00131     explicit constexpr
00132     Dbg(unsigned long, char const *, char const *) {}
00133
00134 #endif

```

```

00135
00136 };
00137
00138 }}
00139

```

## 17.397 env\_ns

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002
00003 #pragma once
00004
00005 #include <l4/re/cap_alloc>
00006 #include <l4/re/util/cap_alloc>
00007 #include <l4/re/namespace>
00008 #include <l4/re/env>
00009 #include <string.h>
00010
00011 namespace L4Re { namespace Util {
00012
00013 class Env_ns
00014 {
00015 private:
00016     L4Re::Cap_alloc *_ca;
00017     Env const *_env;
00018
00019 public:
00020     explicit Env_ns(Env const *env = Env::env(),
00021                    L4Re::Cap_alloc *ca = &L4Re::Util::cap_alloc)
00022         : _ca(ca), _env(env) {}
00023
00024     L4::Cap<void>
00025     query(char const *name, unsigned len, int timeout = Namespace::To_default,
00026           l4_umword_t *local_id = 0, bool iterate = true) const noexcept
00027     {
00028         typedef Env::Cap_entry Cap_entry;
00029
00030         // Skip possible first slash
00031         if (len && name[0] == '/')
00032         {
00033             ++name;
00034             --len;
00035         }
00036
00037         char const *n = name;
00038         for (; len && *n != '/'; ++n, --len) // Count first path element
00039             ;
00040
00041         Cap_entry const *e = _env->get(name, n - name);
00042         if (!e)
00043             return L4::Cap<void>(-L4_ENOENT);
00044
00045         if (len > 0 && *n == '/')
00046         {
00047             L4::Cap<L4Re::Namespace> ns(e->cap);
00048             L4::Cap<void> cap = _ca->alloc<void>();
00049
00050             if (!cap.is_valid())
00051                 return L4::Cap<void>(-L4_ENOMEM);
00052
00053             long r = ns->query(n + 1, len - 1, cap, timeout, local_id, iterate);
00054             if (r >= 0)
00055                 return cap;
00056
00057             _ca->free(cap);
00058
00059             return L4::Cap<void>(r);
00060         }
00061
00062         return L4::Cap<void>(e->cap);
00063     }
00064
00065     L4::Cap<void>
00066     query(char const *name, int timeout = Namespace::To_default,
00067           l4_umword_t *local_id = 0, bool iterate = true) const noexcept
00068     { return query(name, __builtin_strlen(name), timeout, local_id, iterate); }
00069
00070     template<typename T >
00071     L4::Cap<T>
00072     query(char const *name, int timeout = Namespace::To_default,
00073           l4_umword_t *local_id = 0, bool iterate = true) const noexcept
00074     {
00075         return L4::cap_cast<T>(query(name, __builtin_strlen(name),

```

```

00076                                     timeout, local_id, iterate));
00077     }
00078 };
00079
00080 }}

```

## 17.398 event

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 #include <l4/sys/capability>
00013 #include <l4/sys/irq>
00014 #include <l4/sys/cxx/ipc_iface>
00015 #include <l4/sys/cxx/ipc_array>
00016 #include <l4/re/dataspace>
00017 #include <l4/re/event.h>
00018
00019 namespace L4Re {
00020
00021
00022
00023
00024 typedef l4re_event_stream_id_t Event_stream_id;
00025 typedef l4re_event_absinfo_t Event_absinfo;
00026
00027 class L4_EXPORT Event_stream_bitmap_h
00028 {
00029 protected:
00030     static unsigned __get_idx(unsigned idx)
00031     { return idx / (sizeof(unsigned long)*8); }
00032
00033     static unsigned long __get_mask(unsigned idx)
00034     { return 1ul << (idx % (sizeof(unsigned long)*8)); }
00035
00036     static bool __get_bit(unsigned long const *bm, unsigned max, unsigned idx)
00037     {
00038         if (idx <= max)
00039             return bm[__get_idx(idx)] & __get_mask(idx);
00040         return false;
00041     }
00042
00043     static void __set_bit(unsigned long *bm, unsigned max, unsigned idx, bool v)
00044     {
00045         if (idx > max)
00046             return;
00047
00048         if (v)
00049             bm[__get_idx(idx)] |= __get_mask(idx);
00050         else
00051             bm[__get_idx(idx)] &= ~__get_mask(idx);
00052     }
00053 };
00054
00055 class L4_EXPORT Event_stream_info
00056 : public l4re_event_stream_info_t,
00057   private Event_stream_bitmap_h
00058 {
00059 public:
00060     bool get_propbit(unsigned idx) const
00061     { return __get_bit(propbits, L4RE_EVENT_PROP_MAX, idx); }
00062
00063     void set_propbit(unsigned idx, bool v)
00064     { __set_bit(propbits, L4RE_EVENT_PROP_MAX, idx, v); }
00065
00066     bool get_evbit(unsigned idx) const
00067     { return __get_bit(evbits, L4RE_EVENT_EV_MAX, idx); }
00068
00069     void set_evbit(unsigned idx, bool v)
00070     { __set_bit(evbits, L4RE_EVENT_EV_MAX, idx, v); }
00071
00072     bool get_keybit(unsigned idx) const
00073     { return __get_bit(keybits, L4RE_EVENT_KEY_MAX, idx); }
00074
00075     void set_keybit(unsigned idx, bool v)
00076     { __set_bit(keybits, L4RE_EVENT_KEY_MAX, idx, v); }
00077
00078
00079
00080
00081
00082
00083
00084
00085
00086
00087
00088
00089
00090
00091
00092
00093

```

```

00094 bool get_relbit(unsigned idx) const
00095 { return __get_bit(relbits, L4RE_EVENT_REL_MAX, idx); }
00096
00097 void set_relbit(unsigned idx, bool v)
00098 { __set_bit(relbits, L4RE_EVENT_REL_MAX, idx, v); }
00099
00100 bool get_absbit(unsigned idx) const
00101 { return __get_bit(absbits, L4RE_EVENT_ABS_MAX, idx); }
00102
00103 void set_absbit(unsigned idx, bool v)
00104 { __set_bit(absbits, L4RE_EVENT_ABS_MAX, idx, v); }
00105
00106 bool get_swbit(unsigned idx) const
00107 { return __get_bit(swbits, L4RE_EVENT_SW_MAX, idx); }
00108
00109 void set_swbit(unsigned idx, bool v)
00110 { __set_bit(swbits, L4RE_EVENT_SW_MAX, idx, v); }
00111 };
00112
00113 class L4_EXPORT Event_stream_state
00114 : public l4re_event_stream_state_t,
00115 private Event_stream_bitmap_h
00116 {
00117 public:
00118 bool get_keybit(unsigned idx) const
00119 { return __get_bit(keybits, L4RE_EVENT_KEY_MAX, idx); }
00120
00121 void set_keybit(unsigned idx, bool v)
00122 { __set_bit(keybits, L4RE_EVENT_KEY_MAX, idx, v); }
00123
00124 bool get_swbit(unsigned idx) const
00125 { return __get_bit(swbits, L4RE_EVENT_SW_MAX, idx); }
00126
00127 void set_swbit(unsigned idx, bool v)
00128 { __set_bit(swbits, L4RE_EVENT_SW_MAX, idx, v); }
00129 };
00130
00131 class L4_EXPORT Event :
00132 public L4::Kobject_t<Event, L4::Icu, L4RE_PROTO_EVENT>
00133 {
00134 public:
00135 L4_RPC(long, get_buffer, (L4::Ipc::Out<L4::Cap<Dataspace> > ds));
00136
00137 L4_RPC(long, get_num_streams, ());
00138
00139 L4_RPC(long, get_stream_info, (int idx, Event_stream_info *info));
00140
00141 L4_RPC(long, get_stream_info_for_id, (l4_umword_t stream_id, Event_stream_info *info));
00142
00143 L4_RPC_NF(long, get_axis_info, (l4_umword_t stream_id,
00144                                L4::Ipc::Array<unsigned const, unsigned long> axes,
00145                                L4::Ipc::Array<Event_absinfo, unsigned long> &info));
00146
00147 long get_axis_info(l4_umword_t stream_id, unsigned naxes,
00148                   unsigned const *axis, Event_absinfo *info) const noexcept
00149 {
00150     L4::Ipc::Array<Event_absinfo, unsigned long> i(naxes, info);
00151     return get_axis_info_t::call(c(), stream_id,
00152                                  L4::Ipc::Array<unsigned const, unsigned long>(naxes, axis), i);
00153 }
00154
00155 L4_RPC(long, get_stream_state_for_id, (l4_umword_t stream_id,
00156                                       Event_stream_state *state));
00157
00158 typedef L4::TypeId::Rpc<
00159     get_buffer_t,
00160     get_num_streams_t,
00161     get_stream_info_t,
00162     get_stream_info_for_id_t,
00163     get_axis_info_t,
00164     get_stream_state_for_id_t
00165 > Rpcs;
00166 };
00167
00168 struct L4_EXPORT Default_event_payload
00169 {
00170     unsigned short type;
00171     unsigned short code;
00172     int value;
00173     l4_umword_t stream_id;
00174 };
00175
00176 template< typename PAYLOAD = Default_event_payload >
00177 class L4_EXPORT Event_buffer_t
00178 {
00179 public:

```

```

00249
00253 struct Event
00254 {
00255     long long time;
00256     PAYLOAD payload;
00257
00261     void free() noexcept { l4_mb(); time = 0; }
00262 };
00263
00264 private:
00265     Event *_current;
00266     Event *_begin;
00267     Event const *_end;
00268
00269     void inc() noexcept
00270     {
00271         ++_current;
00272         if (_current == _end)
00273             _current = _begin;
00274     }
00275
00276 public:
00277
00278     Event_buffer_t() : _current(0), _begin(0), _end(0) {}
00279
00280     void reset()
00281     {
00282         for (Event *i = _begin; i != _end; ++i)
00283             i->time = 0;
00284         _current = _begin;
00285     }
00286
00293     Event_buffer_t(void *buffer, l4_addr_t size)
00294     : _current(static_cast<Event*>(buffer)), _begin(_current),
00295       _end(_begin + size / sizeof(Event))
00296     { reset(); }
00297
00303     Event *next() noexcept
00304     {
00305         Event *c = _current;
00306         if (c->time)
00307         {
00308             inc();
00309             return c;
00310         }
00311         return 0;
00312     }
00313
00320     bool put(Event const &ev) noexcept
00321     {
00322         Event *c = _current;
00323         if (c->time)
00324             return false;
00325
00326         inc();
00327         c->payload = ev.payload;
00328         l4_wmb();
00329         c->time = ev.time;
00330         return true;
00331     }
00332 };
00333
00334 typedef Event_buffer_t<Default_event_payload> Event_buffer;
00335
00336 }

```

## 17.399 l4/re/util/event File Reference

```

#include <l4/re/cap_alloc>
#include <l4/re/util/cap_alloc>
#include <l4/re/util/unique_cap>
#include <l4/re/env>
#include <l4/re/rm>
#include <l4/re/util/event_buffer>
#include <l4/sys/factory>

```



```

00039 {
00040     Mode_irq,
00041     Mode_polling,
00042 };
00043
00058 template<typename IRQ_TYPE>
00059 int init(L4::Cap<L4Re::Event> event,
00060         L4Re::Env const *env = L4Re::Env::env(),
00061         L4Re::Cap_alloc *ca = &L4Re::Util::cap_alloc)
00062 {
00063     Unique_cap<L4Re::Dataspace> ev_ds(ca->alloc<L4Re::Dataspace>());
00064     if (!ev_ds.is_valid())
00065         return -L4_ENOMEM;
00066
00067     int r;
00068
00069     Unique_del_cap<IRQ_TYPE> ev_irq(ca->alloc<IRQ_TYPE>());
00070     if (!ev_irq.is_valid())
00071         return -L4_ENOMEM;
00072
00073     if ((r = l4_error(env->factory()->create(ev_irq.get()))))
00074         return r;
00075
00076     if ((r = l4_error(event->bind(0, ev_irq.get()))))
00077         return r;
00078
00079     if ((r = event->get_buffer(ev_ds.get())))
00080         return r;
00081
00082     long sz = ev_ds->size();
00083     if (sz < 0)
00084         return sz;
00085
00086     Rm::Unique_region<void*> buf;
00087
00088     if ((r = env->rm()->attach(&buf, sz,
00089                             L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00090                             L4::Ipc::make_cap_rw(ev_ds.get()))))
00091         return r;
00092
00093     _ev_buffer = L4Re::Event_buffer_t<PAYLOAD>(buf.get(), sz);
00094     _ev_ds     = cxx::move(ev_ds);
00095     _ev_irq    = cxx::move(ev_irq);
00096     _buf       = cxx::move(buf);
00097
00098     return 0;
00099 }
00100
00112 int init_poll(L4::Cap<L4Re::Event> event,
00113              L4Re::Env const *env = L4Re::Env::env(),
00114              L4Re::Cap_alloc *ca = &L4Re::Util::cap_alloc)
00115 {
00116     Unique_cap<L4Re::Dataspace> ev_ds(ca->alloc<L4Re::Dataspace>());
00117     if (!ev_ds.is_valid())
00118         return -L4_ENOMEM;
00119
00120     int r;
00121
00122     if ((r = event->get_buffer(ev_ds.get())))
00123         return r;
00124
00125     long sz = ev_ds->size();
00126     if (sz < 0)
00127         return sz;
00128
00129     Rm::Unique_region<void*> buf;
00130
00131     if ((r = env->rm()->attach(&buf, sz,
00132                             L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00133                             L4::Ipc::make_cap_rw(ev_ds.get()))))
00134         return r;
00135
00136     _ev_buffer = L4Re::Event_buffer_t<PAYLOAD>(buf.get(), sz);
00137     _ev_ds     = cxx::move(ev_ds);
00138     _buf       = cxx::move(buf);
00139
00140     return 0;
00141 }
00142
00148 L4Re::Event_buffer_t<PAYLOAD> &buffer() { return _ev_buffer; }
00149
00155 L4::Cap<L4::Triggerable> irq() const { return _ev_irq.get(); }
00156
00157 private:
00158     Unique_cap<L4Re::Dataspace> _ev_ds;
00159     Unique_del_cap<L4::Triggerable> _ev_irq;
00160     L4Re::Event_buffer_t<PAYLOAD> _ev_buffer;

```



```

00161   Rm::Unique_region<void*> _buf;
00162 };
00163
00164 typedef Event_t<Default_event_payload> Event;
00165
00166 }}

```

## 17.401 event\_buffer

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 #include <l4/re/event>
00013 #include <l4/re/event-sys.h>
00014 #include <l4/re/rm>
00015
00016 #include <string.h>
00017
00018 namespace L4Re { namespace Util {
00019
00024 template< typename PAYLOAD >
00025 class Event_buffer_t : public L4Re::Event_buffer_t<PAYLOAD>
00026 {
00027 private:
00028     void *_buf;
00029 public:
00035     void *_buf() const noexcept { return _buf; }
00036
00045     long attach(L4::Cap<L4Re::Dataspace> ds, L4::Cap<L4Re::Rm> rm) noexcept
00046     {
00047         l4_addr_t sz = ds->size();
00048         _buf = 0;
00049
00050         long r = rm->attach(&_buf, sz,
00051                             L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00052                             L4::Ipc::make_cap_rw(ds));
00053         if (r < 0)
00054             return r;
00055
00056         *static_cast<L4Re::Event_buffer_t<PAYLOAD>*>(this)
00057           = L4Re::Event_buffer_t<PAYLOAD>(_buf, sz);
00058         return 0;
00059     }
00060
00068     long detach(L4::Cap<L4Re::Rm> rm) noexcept
00069     {
00070         L4::Cap<L4Re::Dataspace> ds;
00071         if (_buf)
00072             return rm->detach(_buf, &ds);
00073         return 0;
00074     }
00075 };
00076
00077 template< typename PAYLOAD >
00083 class Event_buffer_consumer_t : public Event_buffer_t<PAYLOAD>
00084 {
00085 public:
00086
00093     template< typename CB, typename D >
00094     void foreach_available_event(CB const &cb, D data = D())
00095     {
00096         typename Event_buffer_t<PAYLOAD>::Event *e;
00097         while ((e = Event_buffer_t<PAYLOAD>::next()))
00098         {
00099             cb(e, data);
00100             e->free();
00101         }
00102     }
00103
00114     template< typename CB, typename D >
00115     void process(L4::Cap<L4::Irq> irq,
00116                 L4::Cap<L4::Thread> thread,
00117                 CB const &cb, D data = D())
00118     {

```

```

00119
00120     if (l4_error(irq->bind_thread(thread, 0)))
00121         return;
00122
00123     while (1)
00124     {
00125         long r;
00126         r = l4_ipc_error(l4_irq_receive(irq.cap(), L4_IPC_NEVER),
00127                         l4_utcb());
00128         if (r)
00129             continue;
00130
00131         foreach_available_event(cb, data);
00132     }
00133 }
00134 };
00135
00136 typedef Event_buffer_t<Default_event_payload> Event_buffer;
00137 typedef Event_buffer_consumer_t<Default_event_payload> Event_buffer_consumer;
00138
00139 }}

```

## 17.402 event\_svr

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 #include <l4/re/event_enums.h>
00013 #include <l4/re/event>
00014 #include <l4/re/event-sys.h>
00015 #include <l4/re/util/icu_svr>
00016 #include <l4/cxx/minmax>
00017
00018 #include <l4/sys/cxx/ipc_legacy>
00019
00020 namespace L4Re { namespace Util {
00021
00022     template< typename SVR >
00023     class Event_svr : public Icu_cap_array_svr<SVR>
00024     {
00025     private:
00026         typedef Icu_cap_array_svr<SVR> Icu_svr;
00027
00028     protected:
00029         L4::Cap<L4Re::Dataspace> _ds;
00030         typename Icu_svr::Irq _irq;
00031
00032     public:
00033         Event_svr() : Icu_svr(1, &_irq) {}
00034
00035         L4_RPC_LEGACY_DISPATCH(L4Re::Event);
00036         L4_RPC_LEGACY_USING(Icu_svr);
00037
00038         long op_get_buffer(L4Re::Event::Rights, L4::Ipc::Cap<L4Re::Dataspace> &ds)
00039         {
00040             static_cast<SVR*>(this)->reset_event_buffer();
00041             ds = L4::Ipc::Cap<L4Re::Dataspace>(_ds, L4_CAP_FPAGE_RW);
00042             return 0;
00043         }
00044
00045         long op_get_num_streams(L4Re::Event::Rights)
00046         { return static_cast<SVR*>(this)->get_num_streams(); }
00047
00048         long op_get_stream_info(L4Re::Event::Rights, int idx, Event_stream_info &info)
00049         { return static_cast<SVR*>(this)->get_stream_info(idx, &info); }
00050
00051         long op_get_stream_info_for_id(L4Re::Event::Rights, l4_umword_t id,
00052                                         Event_stream_info &info)
00053         { return static_cast<SVR*>(this)->get_stream_info_for_id(id, &info); }
00054
00055         long op_get_axis_info(L4Re::Event::Rights, l4_umword_t id,
00056                               L4::Ipc::Array_in_buf<unsigned, unsigned long> const &axes,
00057                               L4::Ipc::Array_ref<Event_absinfo, unsigned long> &info)
00058         {
00059             unsigned naxes = cxx::min<unsigned>(L4RE_ABS_MAX, axes.length);

```

```

00066
00067     info.length = 0;
00068
00069     Event_absinfo _info[L4RE_ABS_MAX];
00070     int r = static_cast<SVR*>(this)->get_axis_info(id, naxes, axes.data, _info);
00071     if (r < 0)
00072         return r;
00073
00074     for (unsigned i = 0; i < naxes; ++i)
00075         info.data[i] = _info[i];
00076
00077     info.length = naxes;
00078     return r;
00079 }
00080
00081 long op_get_stream_state_for_id(L4Re::Event::Rights, l4_umword_t stream_id,
00082                               Event_stream_state &state)
00083 { return static_cast<SVR*>(this)->get_stream_state_for_id(stream_id, &state); }
00084
00085 int get_num_streams() const { return 0; }
00086 int get_stream_info(int, L4Re::Event_stream_info *)
00087 { return -L4_EINVAL; }
00088 int get_stream_info_for_id(l4_umword_t, L4Re::Event_stream_info *)
00089 { return -L4_EINVAL; }
00090 int get_axis_info(l4_umword_t, unsigned /*naxes*/, unsigned const * /*axes*/,
00091                  L4Re::Event_absinfo *)
00092 { return -L4_EINVAL; }
00093 int get_stream_state_for_id(l4_umword_t, L4Re::Event_stream_state *)
00094 { return -L4_EINVAL; }
00095 };
00096
00097 }

```

## 17.403 icu\_svr

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2009-2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010
00011 #include <l4/sys/types.h>
00012
00013 #include <l4/sys/icu>
00014 #include <l4/sys/task>
00015 #include <l4/re/env>
00016 #include <l4/re/util/cap_alloc>
00017 #include <l4/sys/cxx/ipc_legacy>
00018
00019 namespace L4Re { namespace Util {
00020
00021 template<typename ICU >
00022 class Icu_svr
00023 {
00024 private:
00025     ICU const *this_icu() const { return static_cast<ICU const *>(this); }
00026     ICU *this_icu() { return static_cast<ICU*>(this); }
00027
00028 public:
00029     L4_RPC_LEGACY_DISPATCH(L4::Icu);
00030
00031     int op_bind(L4::Icu::Rights, l4_umword_t irqnum,
00032                L4::Ipc::Snd_fpage irq_fp);
00033     int op_unbind(L4::Icu::Rights, l4_umword_t irqnum,
00034                  L4::Ipc::Snd_fpage irq_fp);
00035     int op_info(L4::Icu::Rights, L4::Icu::_Info &info);
00036     int op_msi_info(L4::Icu::Rights, l4_umword_t irqnum,
00037                    l4_uint64_t source, l4_icu_msi_info_t &info);
00038     int op_mask(L4::Icu::Rights, l4_umword_t irqnum);
00039     int op_unmask(L4::Icu::Rights, l4_umword_t irqnum);
00040     int op_set_mode(L4::Icu::Rights, l4_umword_t, l4_umword_t)
00041     { return 0; }
00042 };
00043
00044 template<typename ICU> inline
00045 int
00046 Icu_svr<ICU>::op_bind(L4::Icu::Rights, l4_umword_t irqnum,
00047                       L4::Ipc::Snd_fpage irq_fp)
00048 {

```

```

00049     typename ICU::Irq *irq = this_icu()->icu_get_irq(irqnum);
00050     if (!irq)
00051         return -L4_EINVAL;
00052
00053     return irq->bind(this_icu(), irq_fp);
00054 }
00055
00056 template<typename ICU> inline
00057 int
00058 Icu_svr<ICU>::op_unbind(L4::Icu::Rights, l4_umword_t irqnum,
00059                       L4::Ipc::Snd_fpage irq_fp)
00060 {
00061     typename ICU::Irq *irq = this_icu()->icu_get_irq(irqnum);
00062     if (!irq)
00063         return -L4_EINVAL;
00064
00065     return irq->unbind(this_icu(), irq_fp);
00066 }
00067
00068 template<typename ICU> inline
00069 int
00070 Icu_svr<ICU>::op_info(L4::Icu::Rights, L4::Icu::_Info &info)
00071 {
00072     l4_icu_info_t i;
00073     this_icu()->icu_get_info(&i);
00074     info.features = i.features;
00075     info.nr_irqs = i.nr_irqs;
00076     info.nr_msis = i.nr_msis;
00077     return 0;
00078 }
00079
00080 template<typename ICU> inline
00081 int
00082 Icu_svr<ICU>::op_msi_info(L4::Icu::Rights, l4_umword_t irqnum,
00083                          l4_uint64_t source, l4_icu_msi_info_t &info)
00084 {
00085     typename ICU::Irq *irq = this_icu()->icu_get_irq(irqnum);
00086     if (!irq)
00087         return -L4_EINVAL;
00088     return irq->msi_info(source, &info);
00089 }
00090
00091 template<typename ICU> inline
00092 int
00093 Icu_svr<ICU>::op_mask(L4::Icu::Rights, l4_umword_t irqnum)
00094 {
00095     typename ICU::Irq *irq = this_icu()->icu_get_irq(irqnum);
00096     if (irq)
00097         irq->mask(true);
00098     return -L4_ENOREPLY;
00099 }
00100
00101 template<typename ICU> inline
00102 int
00103 Icu_svr<ICU>::op_unmask(L4::Icu::Rights, l4_umword_t irqnum)
00104 {
00105     typename ICU::Irq *irq = this_icu()->icu_get_irq(irqnum);
00106     if (irq)
00107         irq->mask(false);
00108     return -L4_ENOREPLY;
00109 }
00110
00111
00112 template< typename ICU >
00113 class Icu_cap_array_svr : public Icu_svr<ICU>
00114 {
00115 protected:
00116     static void free_irq_cap(L4::Cap<L4::Irq> &cap)
00117     {
00118         if (cap)
00119         {
00120             L4Re::Util::cap_alloc.free(cap);
00121             cap.invalidate();
00122         }
00123     }
00124
00125 public:
00126     class Irq
00127     {
00128     public:
00129         Irq() {}
00130         ~Irq() { ICU::free_irq_cap(_cap); }
00131
00132         void trigger() const
00133         {
00134             if (_cap)
00135                 _cap->trigger();

```

```

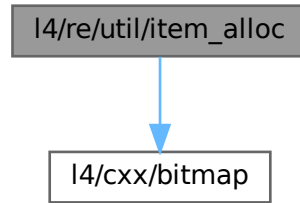
00136     }
00137
00138     int bind(ICU *, L4::Ipc::Snd_fpage const &irq_fp);
00139     int unbind(ICU *, L4::Ipc::Snd_fpage const &irq_fp);
00140     void mask(bool /*mask*/) const
00141     { }
00142
00143     int msi_info(l4_uint64_t, l4_icu_msi_info_t *) const
00144     { return -L4_EINVAL; }
00145
00146     L4::Cap<L4::Irq> cap() const { return _cap; }
00147
00148 private:
00149     L4::Cap<L4::Irq> _cap;
00150 };
00151
00152 private:
00153     Irq *_irqs;
00154     unsigned _nr_irqs;
00155
00156 public:
00157
00158     Icu_cap_array_svr(unsigned nr_irqs, Irq *irqs)
00159     : _irqs(irqs), _nr_irqs(nr_irqs)
00160     {}
00161
00162     Irq *icu_get_irq(l4_umword_t irqnum)
00163     {
00164         if (irqnum >= _nr_irqs)
00165             return 0;
00166
00167         return _irqs + irqnum;
00168     }
00169
00170     void icu_get_info(l4_icu_info_t *inf)
00171     {
00172         inf->features = 0;
00173         inf->nr_irqs = _nr_irqs;
00174         inf->nr_msis = 0;
00175     }
00176 };
00177
00178 template< typename ICU >
00179 int
00180 Icu_cap_array_svr<ICU>::Irq::bind(ICU *cfb, L4::Ipc::Snd_fpage const &irq_fp)
00181 {
00182     if (!irq_fp.cap_received())
00183         return -L4_EINVAL;
00184
00185     L4::Cap<L4::Irq> irq = cfb->server_iface()->template_rcv_cap<L4::Irq>(0);
00186     if (!irq)
00187         return -L4_EINVAL;
00188
00189     int r = cfb->server_iface()->realloc_rcv_cap(0);
00190     if (r < 0)
00191         return r;
00192
00193     ICU::free_irq_cap(_cap);
00194     _cap = irq;
00195     return 0;
00196 }
00197
00198 template< typename ICU >
00199 int
00200 Icu_cap_array_svr<ICU>::Irq::unbind(ICU *, L4::Ipc::Snd_fpage const &/*irq_fp*/)
00201 {
00202     ICU::free_irq_cap(_cap);
00203     _cap = L4::Cap<L4::Irq>::Invalid;
00204     return 0;
00205 }
00206
00207
00208 }

```

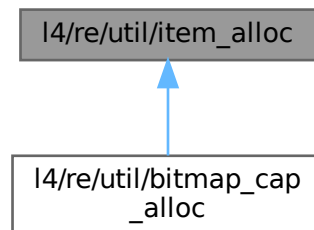
## 17.404 I4/re/util/item\_alloc File Reference

Item allocator.

```
#include <l4/cxx/bitmap>
Include dependency graph for item_alloc:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4Re::Util::Item\\_alloc\\_base](#)  
*Item allocator.*
- class [L4Re::Util::Bitmap\\_base](#)  
*Basic bitmap abstraction.*
- class [L4Re::Util::Bitmap< BITS >](#)  
*A static bitmap.*

## Namespaces

- namespace [L4Re](#)  
*L4Re C++ Interfaces.*
- namespace [L4Re::Util](#)  
*Documentation of the L4 Runtime Environment utility functionality in C++.*

## 17.404.1 Detailed Description

Item allocator.

Definition in file [item\\_alloc](#).

## 17.405 item\_alloc

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00004  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00005  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00006  *      economic rights: Technische Universität Dresden (Germany)
00007  *
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010
00011 #pragma once
00012
00013 #include <l4/cxx/bitmap>
00014
00015 namespace L4Re { namespace Util {
00016
00017     using cxx::Bitmap_base;
00018     using cxx::Bitmap;
00019
00020     class Item_alloc_base
00021     {
00022     private:
00023         long _capacity;
00024         long _free_hint;
00025         Bitmap_base _bits;
00026
00027         void hint(long hint)
00028         { __atomic_store_n(&_free_hint, hint, __ATOMIC_RELAXED); }
00029
00030     public:
00031         bool is_allocated(long item) const noexcept
00032         { return _bits[item]; }
00033
00034         long hint() const { return __atomic_load_n(&_free_hint, __ATOMIC_RELAXED); }
00035
00036         bool alloc(long item) noexcept
00037         {
00038             return !_bits.atomic_get_and_set(item);
00039         }
00040
00041         void free(long item) noexcept
00042         {
00043             if (item < hint())
00044                 hint(item);
00045
00046             _bits.atomic_clear_bit(item);
00047         }
00048
00049         Item_alloc_base(long size, void *mem) noexcept
00050             : _capacity(size), _free_hint(0), _bits(mem)
00051         {}
00052
00053         long alloc() noexcept
00054         {
00055             long free_hint = hint();
00056
00057             for (long i = free_hint; i < _capacity; ++i)
00058                 if (alloc(i))
00059                     hint(i + 1);
00060             return i;
00061         }
00062
00063         // _free_hint is not necessarily correct in case of multi-threading! Make
00064         // sure we don't miss any potentially free slots.
00065         for (long i = 0; i < free_hint && i < _capacity; ++i)
00066             if (alloc(i))
00067                 hint(i + 1);
00067     }
00068 }

```





## Functions

- `int L4Re::Util::kumem_alloc(l4_addr_t *mem, unsigned pages_order, L4::Cap< L4::Task > task=L4Re::Env::env() ->task(), L4::Cap< L4Re::Rm > rm=L4Re::Env::env() ->rm()) noexcept`

*Allocate state area.*

### 17.406.1 Detailed Description

Kumem allocator helper.

Definition in file [kumem\\_alloc](#).

## 17.407 kumem\_alloc

[Go to the documentation of this file.](#)

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00004  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00005  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00006  *      economic rights: Technische Universität Dresden (Germany)
00007  *
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010 #pragma once
00011 #include <l4/re/env>
00012 namespace L4Re { namespace Util {
00013     int
00014     kumem_alloc(l4_addr_t *mem, unsigned pages_order,
00015                 L4::Cap<L4::Task> task = L4Re::Env::env()->task(),
00016                 L4::Cap<L4Re::Rm> rm = L4Re::Env::env()->rm()) noexcept;
00017 }}
00018 }
```

## 17.408 name\_space\_svr

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010 #include <l4/cxx/avl_tree>
00011 #include <l4/cxx/std_ops>
00012 #include <l4/sys/cxx/ipc_epiface>
00013 #include <l4/cxx/string>
00014 #include <l4/re/util/debug>
00015 #include <l4/sys/capability>
00016 #include <l4/re/namespace>
00017 #include <stddef.h>
00018 #include <string.h>
00019 namespace L4Re { namespace Util { namespace Names {
00020     class Name : public cxx::String
00021     {
00022     public:
```

```

00031
00032 Name(const char *name = "") : String(name, __builtin_strlen(name)) {}
00033 Name(const char *name, unsigned long len) : String(name, len) {}
00034 Name(cxx::String const &n) : String(n) {}
00035 char const *name() const { return start(); }
00036 bool operator < (Name const &r) const
00037 {
00038     unsigned long l = cxx::min(len(), r.len());
00039     int v = memcmp(start(), r.start(), l);
00040     return v < 0 || (v == 0 && len() < r.len());
00041 }
00042 };
00043
00044
00045 class Obj
00046 {
00047 protected:
00048     unsigned _f;
00049     union
00050     {
00051         l4_cap_idx_t _cap;
00052         L4::Epiface *_obj;
00053     };
00054
00055 public:
00056     enum Flags
00057     {
00058         F_rw          = L4Re::Namespace::Rw,
00059         F_strong       = L4Re::Namespace::Strong,
00060
00061         F_trusted      = L4Re::Namespace::Trusted,
00062
00063         F_rights_mask = F_rw | F_strong | F_trusted,
00064
00065         F_cap          = 0x100,
00066         F_local        = 0x200,
00067         F_replacable   = 0x400,
00068         F_base_mask    = 0xf00,
00069     };
00070
00071     unsigned flags() const { return _f; }
00072     void restrict_flags(unsigned max_rights)
00073     { _f &= (~F_rights_mask | (max_rights & F_rights_mask)); }
00074
00075     bool is_rw() const { return (_f & F_rw) == F_rw; }
00076     bool is_strong() const { return _f & F_strong; }
00077
00078     bool is_valid() const { return _f & F_cap; }
00079     bool is_complete() const { return is_valid(); }
00080     bool is_local() const { return _f & F_local; }
00081     bool is_replacable() const { return _f & F_replacable; }
00082     bool is_trusted() const { return _f & F_trusted; }
00083
00084     L4::Epiface *obj() const { if (is_local()) return _obj; return 0; }
00085     L4::Cap<void> cap() const
00086     {
00087         if (!is_local())
00088             return L4::Cap<void>(_cap);
00089         if (!_obj)
00090             return L4::Cap<void>::Invalid;
00091         return _obj->obj_cap();
00092     }
00093
00094     void set(Obj const &o, unsigned flags)
00095     {
00096         *this = o;
00097         restrict_flags(flags);
00098     }
00099
00100     explicit Obj(unsigned flags = 0)
00101     : _f(flags), _cap(L4_INVALID_CAP)
00102     {}
00103
00104     Obj(unsigned f, L4::Cap<void> const &cap)
00105     : _f((f & ~F_base_mask) | F_cap), _cap(cap.cap())
00106     {}
00107
00108     Obj(unsigned f, L4::Epiface *o)
00109     : _f((f & ~F_base_mask) | F_cap | F_local), _obj(o)
00110     {}
00111
00112     void reset(unsigned flags)
00113     {
00114         _f = (_f & F_replacable) | (flags & ~(F_cap | F_local));
00115     }

```

```

00121     _cap = L4_INVALID_CAP;
00122 }
00123
00124
00125 };
00126
00127
00131 class Entry : public cxx::Avl_tree_node
00132 {
00133 private:
00134     friend class Name_space;
00135     Name _n;
00136     Obj _o;
00137
00138     bool _dynamic;
00139
00140 public:
00141     Entry(Name const &n, Obj const &o, bool dynamic = false)
00142     : _n(n), _o(o), _dynamic(dynamic) {}
00143
00144     Name const &name() const { return _n; }
00145     Obj const *obj() const { return &_amp;o; }
00146     Obj *obj() { return &_amp;o; }
00147     void obj(Obj const &o) { _o = o; }
00148
00149     bool is_placeholder() const
00150     { return !obj()->is_complete(); }
00151
00152     bool is_dynamic() const { return _dynamic; }
00153
00154     void set(Obj const &o)
00155     {
00156         obj()->set(o, obj()->flags());
00157     }
00158
00159 private:
00160     void * operator new (size_t s);
00161     void operator delete(void *b);
00162
00163 };
00164
00165 struct Names_get_key
00166 {
00167     typedef Name Key_type;
00168     static Key_type const &key_of(Entry const *e)
00169     { return e->name(); }
00170 };
00171
00172
00180 class Name_space
00181 {
00182     friend class Entry;
00183
00184 private:
00185     typedef cxx::Avl_tree<Entry, Names_get_key> Tree;
00186     Tree _tree;
00187
00188 protected:
00189     L4Re::Util::Dbg const &dbg;
00190     L4Re::Util::Err const &err;
00191
00192 public:
00193
00194     typedef Tree::Const_iterator Const_iterator;
00195
00196     Const_iterator begin() const { return _tree.begin(); }
00197     Const_iterator end() const { return _tree.end(); }
00198
00199     Name_space(L4Re::Util::Dbg const &dbg, L4Re::Util::Err const &err)
00200     : _dbg(dbg), _err(err)
00201     {}
00202
00206     virtual ~Name_space() {}
00207
00208     Entry *find(Name const &name) const { return _tree.find_node(name); }
00209     Entry *remove(Name const &name) { return _tree.remove(name); }
00210     Entry *find_iter(Name const &pname) const
00211     {
00212         Name name = pname;
00213         _dbg.printf("resolve '%.*s': ", name.len(), name.start());
00214         Name_space const *ns = this;
00215         while (ns)
00216         {
00217             cxx::String::Index sep = name.find("/");
00218             cxx::String part;
00219             if (!name.eof(sep))
00220                 part = name.head(sep);

```

```

00221         else
00222             part = name;
00223
00224         _dbg.cprintf(" '%.*s'", part.len(), part.start());
00225         Entry *o = ns->find(Name(part.start(), part.len()));
00226
00227         if (!o)
00228         {
00229             _dbg.cprintf(": resolution failed: '%.*s' remaining\n",
00230                 name.len(), name.start());
00231             return 0;
00232         }
00233
00234         auto const *obj = o->obj()->obj();
00235         ns = dynamic_cast<Name_space const *>(obj);
00236         if (ns)
00237         {
00238             if (!name.eof(sep))
00239             {
00240                 name = name.substr(sep + 1);
00241                 continue;
00242             }
00243         }
00244
00245         _dbg.cprintf(": found object: %p (%s)\n",
00246             obj, obj ? typeid(*obj).name() : "");
00247
00248         return o;
00249     }
00250
00251     return 0;
00252 }
00253
00254 bool insert(Entry *e) { return _tree.insert(e).second; }
00255
00256 void dump(bool rec = false, int indent = 0) const;
00257
00258 protected:
00259 // server support -----
00272 virtual Entry *alloc_dynamic_entry(Name const &n, unsigned flags) = 0;
00273
00279 virtual void free_dynamic_entry(Entry *e) = 0;
00280
00303 virtual int get_epiface(l4_umword_t data, bool is_local, L4::Epiface **lo) = 0;
00304
00317 virtual int copy_receive_cap(L4::Cap<void> *cap) = 0;
00318
00327 virtual void free_capability(L4::Cap<void> cap) = 0;
00328
00337 virtual void free_epiface(L4::Epiface *epiface) = 0;
00338
00339 int insert_entry(Name const &name, unsigned flags, Entry **e)
00340 {
00341     Entry *n = find(name);
00342     if (n && n->obj()->is_valid())
00343     {
00344         if (!(flags & L4Re::Namespace::Overwrite)
00345             && n->obj()->cap().validate(L4_BASE_TASK_CAP).label() > 0)
00346             return -L4_EEXIST;
00347
00348         if (n->obj()->is_local())
00349             free_epiface(n->obj()->obj());
00350         else
00351             free_capability(n->obj()->cap());
00352
00353         if (n->is_dynamic())
00354         {
00355             remove(n->name());
00356             free_dynamic_entry(n);
00357             n = 0;
00358         }
00359         else
00360         {
00361             if (!n->obj()->is_replacable())
00362                 return -L4_EEXIST;
00363             n->obj()->reset(Obj::F_rw);
00364         }
00365     }
00366
00367     flags &= L4Re::Namespace::Cap_flags;
00368     if (!n)
00369     {
00370         if (!(n = alloc_dynamic_entry(name, flags)))
00371             return -L4_ENOMEM;
00372         else
00373         {
00374             if (!insert(n))

```

```

00375         {
00376             free_dynamic_entry(n);
00377             return -L4_ENOENT;
00378         }
00379     }
00380 }
00381
00382 *e = n;
00383 return 0;
00384 }
00385
00386 public:
00387 // server interface -----
00388 int op_query(L4Re::Namespace::Rights,
00389             L4::Ipc::Array_in_buf<char, unsigned long> const &name,
00390             L4::Ipc::Snd_fpage &snd_cap, L4::Ipc::Opt<L4::Opcode> &dummy,
00391             L4::Ipc::Opt<L4::Ipc::Array_ref<char, unsigned long> > &out_name)
00392 {
00393     #if 1
00394         _dbg.printf("query: [%ld] '%.*s'\n", name.length,
00395                     static_cast<int>(name.length), name.data);
00396     #endif
00397
00398     char const *sep
00399     = static_cast<char const*>(memchr(name.data, '/', name.length));
00400     unsigned long part;
00401     if (sep)
00402         part = sep - name.data;
00403     else
00404         part = name.length;
00405
00406     Entry *n = find(Name(name.data, part));
00407     if (!n)
00408         return -L4_ENOENT;
00409     else if (!n->obj()->is_valid())
00410         return -L4_EAGAIN;
00411     else
00412     {
00413         if (n->obj()->cap().validate(L4_BASE_TASK_CAP).label() <= 0)
00414         {
00415             if (n->obj()->is_local())
00416                 free_epiface(n->obj()->obj());
00417             else
00418                 free_capability(n->obj()->cap());
00419
00420             if (n->is_dynamic())
00421             {
00422                 remove(n->name());
00423                 free_dynamic_entry(n);
00424             }
00425             return -L4_ENOENT;
00426         }
00427
00428         // make picky clients happy
00429         dummy.set_valid();
00430         // prevent warning about writing uninitialized data in IPC framework
00431         dummy = 0;
00432
00433         l4_umword_t result = 0;
00434
00435         out_name.set_valid();
00436         if (part < name.length)
00437         {
00438             result |= L4Re::Namespace::Partly_resolved;
00439             memcpy(out_name->data, name.data + part + 1, name.length - part - 1);
00440             out_name->length = name.length - part - 1;
00441         }
00442         else
00443             out_name->length = 0;
00444
00445         unsigned flags = L4_CAP_FPAGE_R;
00446         if (n->obj()->is_rw()) flags |= L4_CAP_FPAGE_W;
00447         if (n->obj()->is_strong()) flags |= L4_CAP_FPAGE_S;
00448
00449         snd_cap = L4::Ipc::Snd_fpage(n->obj()->cap(), flags);
00450         _dbg.printf(" result = %lx flgs=%x strg=%d\n",
00451                     result, flags, static_cast<int>(n->obj()->is_strong()));
00452         return result;
00453     }
00454 }
00455
00456 int op_register_obj(L4Re::Namespace::Rights, unsigned flags,
00457                   L4::Ipc::Array_in_buf<char, unsigned long> const &name,
00458                   L4::Ipc::Snd_fpage &cap)
00459 {
00460     if (name.length == 0 || memchr(name.data, '/', name.length))
00461         return -L4_EINVAL;

```

```

00462
00463     L4::Cap<void> reg_cap(L4_INVALID_CAP);
00464     L4::Epiface *src_o = 0;
00465
00466     // Did we receive something we have handed out ourselves? If yes,
00467     // register the object under the given name but do not allocate
00468     // anything more.
00469     if (cap.id_received() || cap.local_id_received())
00470     {
00471         if (int ret = get_epiface(cap.data(), cap.local_id_received(), &src_o))
00472             return ret;
00473
00474         // Make sure rights are restricted to the mapped rights.
00475         flags &= (cap.data() & 0x3UL) | ~0x3UL;
00476     }
00477     else if (cap.cap_received())
00478     {
00479         if (int ret = copy_receive_cap(&reg_cap))
00480             return ret;
00481     }
00482     else if (!cap.is_valid())
00483     {
00484         reg_cap = L4::Cap<void>::Invalid;
00485     }
00486     else
00487         return -L4_EINVAL;
00488
00489     // got a valid entry to register
00490     _dbg.printf("register: '%.*s' flags=%x\n", static_cast<int>(name.length),
00491               name.data, flags);
00492
00493     Name _name(name.data, name.length);
00494
00495     Entry *n;
00496     if (int r = insert_entry(_name, flags, &n))
00497     {
00498         if (cap.cap_received())
00499             free_capability(reg_cap);
00500         if (src_o)
00501             free_epiface(src_o);
00502
00503         return r;
00504     }
00505
00506     if (src_o)
00507         n->set(Names::Obj(flags & L4Re::Namespace::Cap_flags, src_o));
00508     else if (reg_cap.is_valid())
00509         n->set(Names::Obj(flags & L4Re::Namespace::Cap_flags, reg_cap));
00510
00511     return 0;
00512 }
00513
00514 int op_unlink(L4Re::Namespace::Rights,
00515              L4::Ipc::Array_in_buf<char, unsigned long> const &name)
00516 {
00517     #if 1
00518     _dbg.printf("unlink: [%ld] '%.*s'\n", name.length,
00519               static_cast<int>(name.length), name.data);
00520     #endif
00521
00522     char const *sep
00523         = static_cast<char const*>(memchr(name.data, '/', name.length));
00524     unsigned long part;
00525     if (sep)
00526         part = sep - name.data;
00527     else
00528         part = name.length;
00529
00530     Entry *n = find(Name(name.data, part));
00531     if (!n || !n->obj()->is_valid())
00532         return -L4_ENOENT;
00533
00534     if (n->obj()->is_local())
00535         free_epiface(n->obj()->obj());
00536     else
00537         free_capability(n->obj()->cap());
00538
00539     if (n->is_dynamic())
00540     {
00541         remove(n->name());
00542         free_dynamic_entry(n);
00543     }
00544     else
00545         return -L4_EACCESS;
00546
00547     return 0;

```

```

00549     }
00550 };
00551
00552 }}}
00553

```

## 17.409 object\_registry

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 #include <l4/re/util/cap_alloc>
00013 #include <l4/re/util/unique_cap>
00014 #include <l4/re/consts>
00015 #include <l4/re/env>
00016
00017 #include <l4/sys/cxx/ipc_server_loop>
00018 #include <l4/sys/factory>
00019 #include <l4/sys/task>
00020 #include <l4/sys/thread>
00021 #include <l4/sys/ipc_gate>
00022
00023 #include <l4/cxx/exceptions>
00024
00025 namespace L4Re { namespace Util {
00026
00041 class Object_registry :
00042     public L4::Basic_registry,
00043     public L4::Registry_iface
00044 {
00049     struct Null_handler : L4::Epiface_t<Null_handler, L4::Kobject>
00050     {};
00051
00052 protected:
00053     L4::Cap<L4::Thread> _server;
00054     L4::Cap<L4::Factory> _factory;
00055     L4::Ipc_svr::Server_iface *_sif;
00056
00057 private:
00058     Null_handler _null_handler;
00059
00060 public:
00066     explicit
00067     Object_registry(L4::Ipc_svr::Server_iface *sif)
00068     : _server(L4Re::Env::env()->main_thread()),
00069       _factory(L4Re::Env::env()->factory()),
00070       _sif(sif)
00071     {}
00072
00081     Object_registry(L4::Ipc_svr::Server_iface *sif,
00082                     L4::Cap<L4::Thread> server,
00083                     L4::Cap<L4::Factory> factory)
00084     : _server(server), _factory(factory), _sif(sif)
00085     {}
00086
00087 private:
00088     typedef L4::Ipc_svr::Server_iface Server_iface;
00089     typedef Server_iface::Demand Demand;
00090
00091     L4::Cap<L4::Rcv_endpoint>
00092     _register_ep(L4::Epiface *o, L4::Cap<L4::Rcv_endpoint> ep,
00093                 Demand const &demand)
00094     {
00095         int err = _sif->alloc_buffer_demand(demand);
00096         if (err < 0)
00097             return L4::Cap<L4::Rcv_endpoint>(err | L4_INVALID_CAP_BIT);
00098
00099         err = o->set_server(_sif, ep);
00100         if (err < 0)
00101             return L4::Cap<L4::Rcv_endpoint>(err | L4_INVALID_CAP_BIT);
00102
00103         l4_umword_t id = l4_umword_t(o);
00104         err = l4_error(ep->bind_thread(_server, id));
00105         if (err < 0)
00106             return L4::Cap<L4::Rcv_endpoint>(err | L4_INVALID_CAP_BIT);

```

```

00107
00108     return ep;
00109 }
00110
00111 L4::Cap<void> _register_ep(L4::Epiface *o, char const *service,
00112                          Demand const &demand)
00113 {
00114     L4::Cap<L4::Rcv_endpoint> cap = L4Re::Env::env()->get_cap<L4::Rcv_endpoint>(service);
00115     if (!cap.is_valid())
00116         return cap;
00117
00118     return _register_ep(o, cap, demand);
00119 }
00120
00121 L4::Cap<void> _register_gate(L4::Epiface *o, Demand const &demand)
00122 {
00123     int err = _sif->alloc_buffer_demand(demand);
00124     if (err < 0)
00125         return L4::Cap<void>(err | L4_INVALID_CAP_BIT);
00126
00127     auto cap = L4Re::Util::make_unique_cap<L4::Kobject>();
00128
00129     if (!cap.is_valid())
00130         return cap.get();
00131
00132     l4_umword_t id = l4_umword_t(o);
00133     err = l4_error(_factory->create_gate(cap.get(), _server, id));
00134     if (err < 0)
00135         return L4::Cap<void>(err | L4_INVALID_CAP_BIT);
00136
00137     err = o->set_server(_sif, cap.get(), true);
00138     if (err < 0)
00139         return L4::Cap<void>(err | L4_INVALID_CAP_BIT);
00140
00141     return cap.release();
00142 }
00143
00144 L4::Cap<L4::Irq> _register_irq(L4::Epiface *o,
00145                              Demand const &demand)
00146 {
00147     int err = _sif->alloc_buffer_demand(demand);
00148     if (err < 0)
00149         return L4::Cap<L4::Irq>(err | L4_INVALID_CAP_BIT);
00150
00151     auto cap = L4Re::Util::make_unique_cap<L4::Irq>();
00152
00153     if (!cap.is_valid())
00154         return cap.get();
00155
00156     l4_umword_t id = l4_umword_t(o);
00157     err = l4_error(_factory->create(cap.get()));
00158     if (err < 0)
00159         return L4::Cap<L4::Irq>(err | L4_INVALID_CAP_BIT);
00160
00161     err = o->set_server(_sif, cap.get(), true);
00162     if (err < 0)
00163         return L4::Cap<L4::Irq>(err | L4_INVALID_CAP_BIT);
00164
00165     err = l4_error(cap->bind_thread(_server, id));
00166     if (err < 0)
00167         return L4::Cap<L4::Irq>(err | L4_INVALID_CAP_BIT);
00168
00169     return cap.release();
00170 }
00171
00172 static Demand _get_buffer_demand(L4::Epiface *o)
00173 { return o->get_buffer_demand(); }
00174
00175 template<typename T>
00176 static Demand _get_buffer_demand(T *,
00177                                 typename L4::Kobject_typeid<typename T::Interface>::Demand
00178                                 d = typename L4::Kobject_typeid<typename T::Interface>::Demand())
00179 { return d; }
00180
00181 public:
00182 L4::Cap<void> register_obj(L4::Epiface *o, char const *service) override
00183 {
00184     return _register_ep(o, service, _get_buffer_demand(o));
00185 }
00186
00187 L4::Cap<void> register_obj(L4::Epiface *o) override
00188 {
00189     return _register_gate(o, _get_buffer_demand(o));
00190 }
00191
00192 L4::Cap<L4::Irq> register_irq_obj(L4::Epiface *o) override
00193 {

```



```

00229     return _register_irq(o, _get_buffer_demand(o));
00230 }
00231
00245 L4::Cap<L4::Rcv_endpoint>
00246 register_obj(L4::Epiface *o, L4::Cap<L4::Rcv_endpoint> ep) override
00247 {
00248     return _register_ep(o, ep, _get_buffer_demand(o));
00249 }
00250
00251
00262 void unregister_obj(L4::Epiface *o, bool unmap = true) override
00263 {
00264     L4::Epiface::Stored_cap c;
00265
00266     if (!o || !o->obj_cap().is_valid())
00267         return;
00268
00269     c = o->obj_cap();
00270
00271     if (unmap)
00272         L4::Cap<L4::Task> (L4Re::This_task)->unmap(c.fpage(), L4_FP_ALL_SPACES);
00273
00274     // make sure unhandled ipc ends up with the null handler
00275     L4::Thread::Modify_senders todo;
00276     todo.add(~3UL, reinterpret_cast<l4_umword_t>(o),
00277             ~0UL, reinterpret_cast<l4_umword_t>
00278                 (static_cast<L4::Epiface *>(&_null_handler)));
00279     _server->modify_senders(todo);
00280
00281     // we use bit 4 to indicated an internally allocated cap
00282     if (c.managed())
00283         cap_alloc.free(c);
00284
00285     o->set_server(0, L4::Cap<void>::Invalid);
00286 }
00287 };
00288
00292 template< typename LOOP_HOOKS = L4::Ipc_svr::Default_loop_hooks >
00293 class Registry_server : public L4::Server<LOOP_HOOKS>
00294 {
00295 private:
00296     typedef L4::Server<LOOP_HOOKS> Base;
00297     Object_registry _registry;
00298
00299 public:
00305     Registry_server() : _registry(this)
00306     {}
00307
00317     Registry_server(l4_utcb_t *, L4::Cap<L4::Thread> server,
00318                   L4::Cap<L4::Factory> factory) L4_DEPRECATED("Omit UTCB pointer argument")
00319     : _registry(this, server, factory)
00320     {}
00321
00328     Registry_server(L4::Cap<L4::Thread> server,
00329                   L4::Cap<L4::Factory> factory)
00330     : _registry(this, server, factory)
00331     {}
00332
00334     Object_registry const *registry() const { return &_registry; }
00336     Object_registry *registry() { return &_registry; }
00337
00344     void L4_NORETURN loop(l4_utcb_t *utcb = l4_utcb())
00345     { Base::template loop<L4::Runtime_error, Object_registry &>(_registry, utcb); }
00346
00355     template <typename Printer>
00356     void L4_NORETURN loop_dbg(Printer printer, l4_utcb_t *utcb = l4_utcb())
00357     {
00358         Base::template loop_dbg<L4::Runtime_error, Object_registry &, Printer>
00359             (_registry, printer, utcb);
00360     }
00361 };
00362 }

```

## 17.410 poll\_timeout\_kipclock

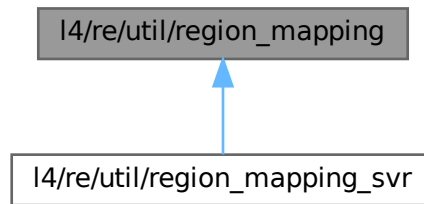
```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2012 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008

```



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [L4Re](#)  
*L4Re C++ Interfaces.*
- namespace [L4Re::Util](#)  
*Documentation of the [L4 Runtime Environment](#) utility functionality in C++.*

## 17.411.1 Detailed Description

Region handling.

Definition in file [region\\_mapping](#).

## 17.412 region\_mapping

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00010  *               Björn Döbel <doebel@os.inf.tu-dresden.de>
00011  *               economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * License: see LICENSE.spdx (in this directory or the directories above)
00014  */
00015
00016 #pragma once
00017
00018 #include <l4/cxx/avl_map>
00019 #include <l4/sys/types.h>
00020 #include <l4/re/rm>
00021
00022
00023 namespace L4Re { namespace Util {
00024   class Region
00025   {
00026   private:
00027     l4_addr_t _start, _end;
00028     #ifdef CONFIG_L4RE_REGION_INFO
00029     char _dbg_name[40]; // Not a 0-terminating string
00030     unsigned char _dbg_name_len = 0;
00031     static_assert(sizeof(_dbg_name) < 256);
00032     Rm::Offset _dbg_backing_offset = 0;

```

```

00033 #endif
00034
00035 public:
00036     Region() noexcept : _start(~0UL), _end(~0UL) {}
00037     Region(l4_addr_t addr) noexcept : _start(addr), _end(addr) {}
00038     Region(l4_addr_t start, l4_addr_t end) noexcept
00039         : _start(start), _end(end) {}
00040     Region(l4_addr_t start, l4_addr_t end,
00041           char const *name, unsigned name_len,
00042           Rm::Offset backing_offset) noexcept
00043         : _start(start), _end(end)
00044     {
00045 #ifdef CONFIG_L4RE_REGION_INFO
00046         _dbg_name_len = name_len > sizeof(_dbg_name)
00047             ? sizeof(_dbg_name) : name_len;
00048         for (unsigned i = 0; i < _dbg_name_len; ++i)
00049             _dbg_name[i] = name[i];
00050
00051         _dbg_backing_offset = backing_offset;
00052 #else
00053         (void)name;
00054         (void)name_len;
00055         (void)backing_offset;
00056 #endif
00057     }
00058     l4_addr_t start() const noexcept { return _start; }
00059     l4_addr_t end() const noexcept { return _end; }
00060     unsigned long size() const noexcept { return end() - start() + 1; }
00061     bool invalid() const noexcept { return _start == ~0UL && _end == ~0UL; }
00062     bool operator < (Region const &o) const noexcept
00063     { return end() < o.start(); }
00064     bool contains(Region const &o) const noexcept
00065     { return o.start() >= start() && o.end() <= end(); }
00066     bool operator == (Region const &o) const noexcept
00067     { return o.start() == start() && o.end() == end(); }
00068     ~Region() noexcept {}
00069
00070 #ifdef CONFIG_L4RE_REGION_INFO
00071     char const *name() const { return _dbg_name; }
00072     unsigned char name_len() const { return _dbg_name_len; }
00073     Rm::Offset backing_offset() const { return _dbg_backing_offset; }
00074 #else
00075     char const *name() const { return "N/A"; }
00076     unsigned char name_len() const { return 3; }
00077     Rm::Offset backing_offset() const { return 0; }
00078 #endif
00079 };
00080
00081 template< typename DS, typename OPS >
00082 class Region_handler
00083 {
00084 private:
00085     L4Re::Rm::Offset _offs;
00086     DS _mem;
00087     l4_cap_idx_t _client_cap = L4_INVALID_CAP;
00088     L4Re::Rm::Region_flags _flags;
00089
00090 public:
00091     typedef DS Dataspace;
00092     typedef OPS Ops;
00093     typedef typename OPS::Map_result Map_result;
00094
00095     Region_handler() noexcept : _offs(0), _mem(), _flags() {}
00096     Region_handler(Dataspace const &mem, l4_cap_idx_t client_cap,
00097                   L4Re::Rm::Offset offset = 0,
00098                   L4Re::Rm::Region_flags flags = L4Re::Rm::Region_flags(0)) noexcept
00099         : _offs(offset), _mem(mem), _client_cap(client_cap), _flags(flags)
00100     {}
00101
00102     Dataspace const &memory() const noexcept
00103     {
00104         return _mem;
00105     }
00106
00107     l4_cap_idx_t client_cap_idx() const noexcept
00108     {
00109         return _client_cap;
00110     }
00111
00112     L4Re::Rm::Offset offset() const noexcept
00113     {
00114         return _offs;
00115     }
00116
00117     constexpr bool is_ro() const noexcept
00118     {
00119         return !(_flags & L4Re::Rm::F::W);

```

```

00120     }
00121
00122     L4Re::Rm::Region_flags caching() const noexcept
00123     {
00124         return _flags & L4Re::Rm::F::Caching_mask;
00125     }
00126
00127     L4Re::Rm::Region_flags flags() const noexcept
00128     {
00129         return _flags;
00130     }
00131
00132     Region_handler operator + (l4_int64_t offset) const noexcept
00133     {
00134         Region_handler n = *this; n._offs += offset; return n;
00135     }
00136
00137     void free(l4_addr_t start, unsigned long size) const noexcept
00138     {
00139         Ops::free(this, start, size);
00140     }
00141
00142     int map(l4_addr_t addr, Region const &r, bool writable,
00143            Map_result *result) const
00144     {
00145         return Ops::map(this, addr, r, writable, result);
00146     }
00147
00148     int map_info(l4_addr_t *start_addr, l4_addr_t *end_addr) const
00149     {
00150         return Ops::map_info(this, start_addr, end_addr);
00151     }
00152
00153 };
00154
00155
00156 template< typename Hdlr, template<typename T> class Alloc >
00157 class Region_map
00158 {
00159 protected:
00160     typedef cxx::Avl_map< Region, Hdlr, cxx::Lt_functor, Alloc > Tree;
00161     Tree _rm;
00162     Tree _am;
00163
00164 private:
00165     l4_addr_t _start;
00166     l4_addr_t _end;
00167
00168 protected:
00169     void set_limits(l4_addr_t start, l4_addr_t end) noexcept
00170     {
00171         _start = start;
00172         _end = end;
00173     }
00174
00175 public:
00176     typedef typename Tree::Item_type Item;
00177     typedef typename Tree::Node Node;
00178     typedef typename Tree::Key_type Key_type;
00179     typedef Hdlr Region_handler;
00180
00181     typedef typename Tree::Iterator Iterator;
00182     typedef typename Tree::Const_iterator Const_iterator;
00183     typedef typename Tree::Rev_iterator Rev_iterator;
00184     typedef typename Tree::Const_rev_iterator Const_rev_iterator;
00185
00186     Iterator begin() noexcept { return _rm.begin(); }
00187     Const_iterator begin() const noexcept { return _rm.begin(); }
00188     Iterator end() noexcept { return _rm.end(); }
00189     Const_iterator end() const noexcept { return _rm.end(); }
00190
00191     Iterator area_begin() noexcept { return _am.begin(); }
00192     Const_iterator area_begin() const noexcept { return _am.begin(); }
00193     Iterator area_end() noexcept { return _am.end(); }
00194     Const_iterator area_end() const noexcept { return _am.end(); }
00195     Node area_find(Key_type const &c) const noexcept { return _am.find_node(c); }
00196
00197     l4_addr_t min_addr() const noexcept { return _start; }
00198     l4_addr_t max_addr() const noexcept { return _end; }
00199
00200
00201     Region_map(l4_addr_t start, l4_addr_t end) noexcept : _start(start), _end(end) {}
00202
00203     Node find(Key_type const &key) const noexcept
00204     {
00205         Node n = _rm.find_node(key);
00206         if (!n)

```

```

00207         return Node();
00208
00209     // 'find' should find any region overlapping with the searched one, the
00210     // caller should check for further requirements
00211     if (0)
00212         if (!n->first.contains(key))
00213             return Node();
00214
00215     return n;
00216 }
00217
00218 Node lower_bound(Key_type const &key) const noexcept
00219 {
00220     Node n = _rm.lower_bound_node(key);
00221     return n;
00222 }
00223
00224 Node lower_bound_area(Key_type const &key) const noexcept
00225 {
00226     Node n = _am.lower_bound_node(key);
00227     return n;
00228 }
00229
00230 l4_addr_t attach_area(l4_addr_t addr, unsigned long size,
00231                      L4Re::Rm::Flags flags = L4Re::Rm::Flags(0),
00232                      unsigned char align = L4_PAGESHIFT) noexcept
00233 {
00234     if (size < 2)
00235         return L4_INVALID_ADDR;
00236
00237     Region c;
00238
00239     if (!(flags & L4Re::Rm::F::Search_addr))
00240     {
00241         c = Region(addr, addr + size - 1);
00242         Node r = _am.find_node(c);
00243         if (r)
00244             return L4_INVALID_ADDR;
00245     }
00246
00247     while (flags & L4Re::Rm::F::Search_addr)
00248     {
00249         if (addr < min_addr() || (addr + size - 1) > max_addr())
00250             addr = min_addr();
00251         addr = find_free(addr, max_addr(), size, align, flags);
00252         if (addr == L4_INVALID_ADDR)
00253             return L4_INVALID_ADDR;
00254     }
00255
00256     c = Region(addr, addr + size - 1);
00257     Node r = _am.find_node(c);
00258     if (!r)
00259         break;
00260
00261     if (r->first.end() >= max_addr())
00262         return L4_INVALID_ADDR;
00263
00264     addr = r->first.end() + 1;
00265 }
00266
00267 if (_am.insert(c, Hdlr(typename Hdlr::Dataspace(), 0, 0, flags.region_flags())).second == 0)
00268     return addr;
00269
00270 return L4_INVALID_ADDR;
00271 }
00272
00273 bool detach_area(l4_addr_t addr) noexcept
00274 {
00275     if (_am.remove(addr))
00276         return false;
00277
00278     return true;
00279 }
00280
00281 void *attach(void *addr, unsigned long size, Hdlr const &hdlr,
00282             L4Re::Rm::Flags attach_flags = L4Re::Rm::Flags(0),
00283             unsigned char align = L4_PAGESHIFT,
00284             char const *name = nullptr, unsigned name_len = 0,
00285             L4Re::Rm::Offset backing_offset = 0) noexcept
00286 {
00287     if (size < 2)
00288         return L4_INVALID_PTR;
00289
00290     l4_addr_t beg, end;
00291     int err = hdlr.map_info(&beg, &end);
00292     if (err > 0)
00293     {

```

```

00294         // Mapping address determined by underlying dataspace. Make sure we
00295         // prevent any additional alignment. We already know the place!
00296         beg += hdlr.offset();
00297         end = beg + size - 1U;
00298         align = L4_PAGESHIFT;
00299
00300         // In case of exact mappings, the supplied address must match because
00301         // we cannot remap.
00302         if (!(attach_flags & L4Re::Rm::F::Search_addr)
00303             && reinterpret_cast<l4_addr_t>(addr) != beg)
00304             return L4_INVALID_PTR;
00305
00306         // When searching for a suitable address, the start must cover the
00307         // dataspace beginning to "find" the right spot.
00308         if ((attach_flags & L4Re::Rm::F::Search_addr)
00309             && reinterpret_cast<l4_addr_t>(addr) > beg)
00310             return L4_INVALID_PTR;
00311     }
00312     else if (err == 0)
00313     {
00314         beg = reinterpret_cast<l4_addr_t>(addr);
00315         end = max_addr();
00316     }
00317     else if (err < 0)
00318         return L4_INVALID_PTR;
00319
00320     if (attach_flags & L4Re::Rm::F::In_area)
00321     {
00322         Node r = _am.find_node(Region(beg, beg + size - 1));
00323         if (!r || (r->second.flags() & L4Re::Rm::F::Reserved))
00324             return L4_INVALID_PTR;
00325
00326         end = r->first.end();
00327     }
00328
00329     if (attach_flags & L4Re::Rm::F::Search_addr)
00330     {
00331         beg = find_free(beg, end, size, align, attach_flags);
00332         if (beg == L4_INVALID_ADDR)
00333             return L4_INVALID_PTR;
00334     }
00335
00336     if (!(attach_flags & (L4Re::Rm::F::Search_addr | L4Re::Rm::F::In_area))
00337         && _am.find_node(Region(beg, beg + size - 1)))
00338         return L4_INVALID_PTR;
00339
00340     if (beg < min_addr() || beg + size - 1 > end)
00341         return L4_INVALID_PTR;
00342
00343     if (_rm.insert(Region(beg, beg + size - 1,
00344                          name, name_len, backing_offset), hdlr).second
00345         == 0)
00346         return reinterpret_cast<void*>(beg);
00347
00348     return L4_INVALID_PTR;
00349 }
00350
00351 int detach(void *addr, unsigned long sz, unsigned flags,
00352           Region *reg, Hdlr *hdlr) noexcept
00353 {
00354     l4_addr_t a = reinterpret_cast<l4_addr_t>(addr);
00355     Region dr(a, a + sz - 1);
00356     Region res(~0UL, 0);
00357
00358     Node r = find(dr);
00359     if (!r)
00360         return -L4_ENOENT;
00361
00362     Region g = r->first;
00363     Hdlr const &h = r->second;
00364
00365     if (flags & L4Re::Rm::Detach_overlap || dr.contains(g))
00366     {
00367         // successful removal of the AVL tree item also frees the node
00368         Hdlr h_copy = h;
00369
00370         if (_rm.remove(g))
00371             return -L4_ENOENT;
00372
00373         if (!(flags & L4Re::Rm::Detach_keep) && (h_copy.flags() & L4Re::Rm::F::Detach_free))
00374             h_copy.free(0, g.size());
00375
00376         if (hdlr)
00377             *hdlr = h_copy;
00378         if (reg)
00379             *reg = g;
00380     }

```

```

00381         if (find(dr))
00382             return Rm::Detached_ds | Rm::Detach_again;
00383         else
00384             return Rm::Detached_ds;
00385     }
00386     else if (dr.start() <= g.start())
00387     {
00388         // move the start of a region
00389
00390         if (!(flags & L4Re::Rm::Detach_keep) && (h.flags() & L4Re::Rm::F::Detach_free))
00391             h.free(0, dr.end() + 1 - g.start());
00392
00393         unsigned long sz = dr.end() + 1 - g.start();
00394         Item &cn = const_cast<Item &>(*r);
00395         cn.first = Region(dr.end() + 1, g.end());
00396         cn.second = cn.second + sz;
00397         if (hdlr)
00398             *hdlr = Hdlr();
00399         if (reg)
00400             *reg = Region(g.start(), dr.end());
00401         if (find(dr))
00402             return Rm::Kept_ds | Rm::Detach_again;
00403         else
00404             return Rm::Kept_ds;
00405     }
00406     else if (dr.end() >= g.end())
00407     {
00408         // move the end of a region
00409
00410         if (!(flags & L4Re::Rm::Detach_keep)
00411             && (h.flags() & L4Re::Rm::F::Detach_free))
00412             h.free(dr.start() - g.start(), g.end() + 1 - dr.start());
00413
00414         Item &cn = const_cast<Item &>(*r);
00415         cn.first = Region(g.start(), dr.start() - 1);
00416         if (hdlr)
00417             *hdlr = Hdlr();
00418         if (reg)
00419             *reg = Region(dr.start(), g.end());
00420
00421         if (find(dr))
00422             return Rm::Kept_ds | Rm::Detach_again;
00423         else
00424             return Rm::Kept_ds;
00425     }
00426     else if (g.contains(dr))
00427     {
00428         // split a single region that contains the new region
00429
00430         if (!(flags & L4Re::Rm::Detach_keep) && (h.flags() & L4Re::Rm::F::Detach_free))
00431             h.free(dr.start() - g.start(), dr.size());
00432
00433         // first move the end off the existing region before the new one
00434         Item &cn = const_cast<Item &>(*r);
00435         cn.first = Region(g.start(), dr.start()-1);
00436
00437         int err;
00438
00439         // insert a second region for the remaining tail of
00440         // the old existing region
00441         err = _rm.insert(Region(dr.end() + 1, g.end()),
00442             h + (dr.end() + 1 - g.start())).second;
00443
00444         if (err)
00445             return err;
00446
00447         if (hdlr)
00448             *hdlr = h;
00449         if (reg)
00450             *reg = dr;
00451         return Rm::Split_ds;
00452     }
00453     return -L4_ENOENT;
00454 }
00455
00456 l4_addr_t find_free(l4_addr_t start, l4_addr_t end, l4_addr_t size,
00457     unsigned char align, L4Re::Rm::Flags attach_flags) const noexcept;
00458 };
00459
00460
00461 template< typename Hdlr, template<typename T> class Alloc >
00462 l4_addr_t
00463 Region_map<Hdlr, Alloc>::find_free(l4_addr_t start, l4_addr_t end,
00464     unsigned long size, unsigned char align, L4Re::Rm::Flags attach_flags) const noexcept
00465 {
00466     l4_addr_t addr = start;

```



```

00468
00469     if (addr == ~0UL || addr < min_addr() || addr >= end)
00470         addr = min_addr();
00471
00472     addr = l4_round_size(addr, align);
00473     Node r;
00474
00475     for(;;)
00476     {
00477         if (addr > 0 && addr - 1 > end - size)
00478             return L4_INVALID_ADDR;
00479
00480         Region c(addr, addr + size - 1);
00481         r = _rm.find_node(c);
00482
00483         if (!r)
00484         {
00485             if (!(attach_flags & L4Re::Rm::F::In_area) && (r = _am.find_node(c)))
00486             {
00487                 if (r->first.end() > end - size)
00488                     return L4_INVALID_ADDR;
00489
00490                 addr = l4_round_size(r->first.end() + 1, align);
00491                 continue;
00492             }
00493             break;
00494         }
00495         else if (r->first.end() > end - size)
00496             return L4_INVALID_ADDR;
00497
00498         addr = l4_round_size(r->first.end() + 1, align);
00499     }
00500
00501     if (!r)
00502         return addr;
00503
00504     return L4_INVALID_ADDR;
00505 }
00506
00507 }}

```

## 17.413 region\_mapping\_svr

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/sys/types.h>
00010 #include <l4/re/rm>
00011 #include <l4/re/util/region_mapping>
00012
00013 namespace L4Re { namespace Util {
00014
00015     template<typename DERIVED, typename Dbg>
00016     struct Rm_server
00017     {
00018     private:
00019         DERIVED *rm() { return static_cast<DERIVED*>(this); }
00020         DERIVED const *rm() const { return static_cast<DERIVED const*>(this); }
00021     public:
00022
00023         long op_attach(L4Re::Rm::Rights, l4_addr_t &_start,
00024             unsigned long size, Rm::Flags flags,
00025             L4::Ipc::Snd_fpage ds_cap, L4Re::Rm::Offset offs,
00026             unsigned char align, l4_cap_idx_t client_cap_idx,
00027             L4::Ipc::String<> name, L4Re::Rm::Offset backing_offset)
00028         {
00029             typename DERIVED::Dataspace ds;
00030
00031             if (!(flags & (Rm::F::Reserved | Rm::F::Kernel)))
00032             {
00033                 if (long r = rm()->validate_ds
00034                     (static_cast<DERIVED*>(this)->server_iface(), ds_cap,
00035                     flags.region_flags(), &ds))
00036                     return r;
00037             }
00038
00039             size = l4_round_page(size);
00040         }
00041     };
00042 } }

```

```

00049     l4_addr_t start = l4_trunc_page(_start);
00050
00051     if (size < L4_PAGESIZE)
00052         return -L4_EINVAL;
00053
00054     Rm::Region_flags r_flags = flags.region_flags();
00055     Rm::Attach_flags a_flags = flags.attach_flags();
00056
00057     typename DERIVED::Region_handler handler(ds, client_cap_idx, offs, r_flags);
00058     start = l4_addr_t(rm()->attach(reinterpret_cast<void*>(start), size,
00059                                     handler, a_flags, align,
00060                                     name.data,
00061                                     // L4::Ipc::String includes terminating '\0'
00062                                     name.length ? name.length - 1 : 0,
00063                                     backing_offset));
00064
00065     if (start == L4_INVALID_ADDR)
00066         return -L4_EADDRNOTAVAIL;
00067
00068     _start = start;
00069     return L4_EOK;
00070 }
00071
00072 long op_free_area(L4Re::Rm::Rights, l4_addr_t start)
00073 {
00074     if (!rm()->detach_area(start))
00075         return -L4_ENOENT;
00076
00077     return L4_EOK;
00078 }
00079
00080 long op_find(L4Re::Rm::Rights, l4_addr_t &addr, unsigned long &size,
00081              L4Re::Rm::Flags &flags, L4Re::Rm::Offset &offset,
00082              L4::Cap<L4Re::Dataspace> &m)
00083 {
00084     if (!DERIVED::Have_find)
00085         return -L4_EPERM;
00086
00087     Rm::Flags flag_area { 0 };
00088
00089     typename DERIVED::Node r = rm()->find(Region(addr, addr + size - 1));
00090     if (!r)
00091     {
00092         r = rm()->area_find(Region(addr, addr + size - 1));
00093         if (!r)
00094             return -L4_ENOENT;
00095         flag_area = Rm::F::In_area;
00096     }
00097
00098     addr = r->first.start();
00099     size = r->first.end() + 1 - addr;
00100
00101     flags = r->second.flags() | flag_area;
00102     offset = r->second.offset();
00103     m = L4::Cap<L4Re::Dataspace>(DERIVED::find_res(r->second.memory()));
00104     return L4_EOK;
00105 }
00106
00107 long op_detach(L4Re::Rm::Rights, l4_addr_t addr,
00108               unsigned long size, unsigned flags,
00109               l4_addr_t &start, l4_addr_t &rsz,
00110               l4_cap_idx_t &mem_cap)
00111 {
00112     Region r;
00113     typename DERIVED::Region_handler h;
00114     int err = rm()->detach(reinterpret_cast<void*>(addr), size, flags, &r, &h);
00115     if (err < 0)
00116     {
00117         start = rsz = 0;
00118         mem_cap = L4_INVALID_CAP;
00119         return err;
00120     }
00121
00122     if (r.invalid())
00123     {
00124         start = rsz = 0;
00125         mem_cap = L4_INVALID_CAP;
00126         return -L4_ENOENT;
00127     }
00128
00129     start = r.start();
00130     rsz = r.size();
00131     mem_cap = h.client_cap_idx();
00132     return err;
00133 }
00134
00135 long op_reserve_area(L4Re::Rm::Rights, l4_addr_t &start, unsigned long size,

```

```

00148             L4Re::Rm::Flags flags, unsigned char align)
00149 {
00150     start = rm()->attach_area(start, size, flags, align);
00151     if (start == L4_INVALID_ADDR)
00152         return -L4_EADDRNOTAVAIL;
00153     return L4_EOK;
00154 }
00155
00156 long op_get_regions(L4Re::Rm::Rights, l4_addr_t addr,
00160                     L4::Ipc::Ret_array<L4Re::Rm::Region> regions)
00161 {
00162     typename DERIVED::Node r;
00163     unsigned num = 0;
00164     while ((r = rm()->lower_bound(Region(addr))))
00165     {
00166         Rm::Region &x = regions.value[num];
00167         x.start = r->first.start();
00168         x.end = r->first.end();
00169         x.flags = r->second.flags();
00170
00171         if (++num >= regions.max)
00172             break;
00173
00174         if (x.end >= rm()->max_addr())
00175             break;
00176         addr = x.end + 1;
00177     }
00178     return num;
00179 }
00180
00181 long op_get_areas(L4Re::Rm::Rights, l4_addr_t addr,
00185                     L4::Ipc::Ret_array<L4Re::Rm::Area> areas)
00186 {
00187     typename DERIVED::Node r;
00188     unsigned num = 0;
00189     while ((r = rm()->lower_bound_area(Region(addr))))
00190     {
00191         Rm::Area &x = areas.value[num];
00192         x.start = r->first.start();
00193         x.end = r->first.end();
00194
00195         if (++num >= areas.max)
00196             break;
00197
00198         if (x.end >= rm()->max_addr())
00199             break;
00200
00201         addr = x.end + 1;
00202     }
00203     return num;
00204 }
00205
00206 private:
00207     static void pager_set_result(L4::Ipc::Opt<L4::Ipc::Snd_fpage> *fp,
00208                                 L4::Ipc::Snd_fpage const &f)
00209     { *fp = f; }
00210
00211     static void pager_set_result(L4::Ipc::Opt<L4::Ipc::Snd_fpage> *, ...)
00212     {}
00213 public:
00214
00215 long op_io_page_fault(L4::Io_pager::Rights, l4_fpage_t, l4_umword_t,
00219                     L4::Ipc::Opt<L4::Ipc::Snd_fpage> &)
00220 {
00221     // generate exception
00222     return -L4_ENOMEM;
00223 }
00224
00225 long op_page_fault(L4::Pager::Rights, l4_umword_t addr, l4_umword_t pc,
00226                    L4::Ipc::Opt<L4::Ipc::Snd_fpage> &fp)
00227 {
00228     Dbg(Dbg::Server).printf("page fault: %lx pc=%lx\n", addr, pc);
00229
00230     bool need_w = addr & 2;
00231     bool need_x = addr & 4;
00232
00233     typename DERIVED::Node n = rm()->find(addr);
00234
00235     if (!n || !n->second.memory())
00236     {
00237         Dbg(Dbg::Warn, "rm").printf("unhandled %s page fault at 0x%lx pc=0x%lx\n",
00238                                     need_w ? "write" :
00239                                     need_x ? "instruction" : "read", addr, pc);
00240         // generate exception
00241         return -L4_ENOMEM;
00242     }
00243 }

```

```

00244     if (!(n->second.flags() & L4Re::Rm::F::W) && need_w)
00245     {
00246         Dbg(Dbg::Warn, "rm").printf("write page fault in readonly region at 0x%lx pc=0x%lx\n",
00247                                     addr, pc);
00248         // generate exception
00249         return -L4_EACCESS;
00250     }
00251
00252     if (!(n->second.flags() & L4Re::Rm::F::X) && need_x)
00253     {
00254         Dbg(Dbg::Warn, "rm").printf("instruction page fault in non-exec region at 0x%lx pc=0x%lx\n",
00255                                     addr, pc);
00256         // generate exception
00257         return -L4_EACCESS;
00258     }
00259
00260     // This check is optional but avoids doing a map operation that will not
00261     // work. We shall never get here from a page-fault but only from
00262     // artificial page handling.
00263     if (n->second.flags() & (L4Re::Rm::F::Kernel | L4Re::Rm::F::Reserved))
00264     {
00265         Dbg(Dbg::Warn, "rm").printf("page fault handling in kernel-memory provided region or reserved
00266 region at 0x%lx pc=0x%lx\n",
00267                                     addr, pc);
00268         // generate exception
00269         return -L4_ENODEV;
00270     }
00271
00272     typename DERIVED::Region_handler::Ops::Map_result map_res;
00273     if (int err = n->second.map(addr, n->first, need_w, &map_res))
00274     {
00275         Dbg(Dbg::Warn, "rm").printf("mapping for page fault failed with error %d at 0x%lx pc=0x%lx\n",
00276                                     err, addr, pc);
00277         // generate exception
00278         return -L4_ENOMEM;
00279     }
00280     pager_set_result(&fp, map_res);
00281     return L4_EOK;
00282 }
00283
00284 long op_get_info(L4Re::Rm::Rights, l4_addr_t addr,
00285                 L4::Ip::String<char> &name, L4Re::Rm::Offset &backing_offset)
00286 {
00287     #ifdef CONFIG_L4RE_REGION_INFO
00288     typename DERIVED::Node r = rm()->find(Region(addr));
00289     if (!r)
00290         return -L4_ENOENT;
00291     backing_offset = r->first.backing_offset();
00292     unsigned long i;
00293     char const *src = r->first.name();
00294     unsigned src_len = r->first.name_len();
00295     for (i = 0; i < src_len && i < name.length - 1; ++i)
00296         name.data[i] = src[i];
00297     name.length = i + 1;
00298     name.data[i] = '\0';
00299     return L4_EOK;
00300     #else
00301     (void)addr;
00302     (void)name;
00303     (void)backing_offset;
00304     return -L4_ENOSYS;
00305     #endif
00306 }
00307 };
00308
00309 }}

```

## 17.414 l4/re/shared\_cap File Reference

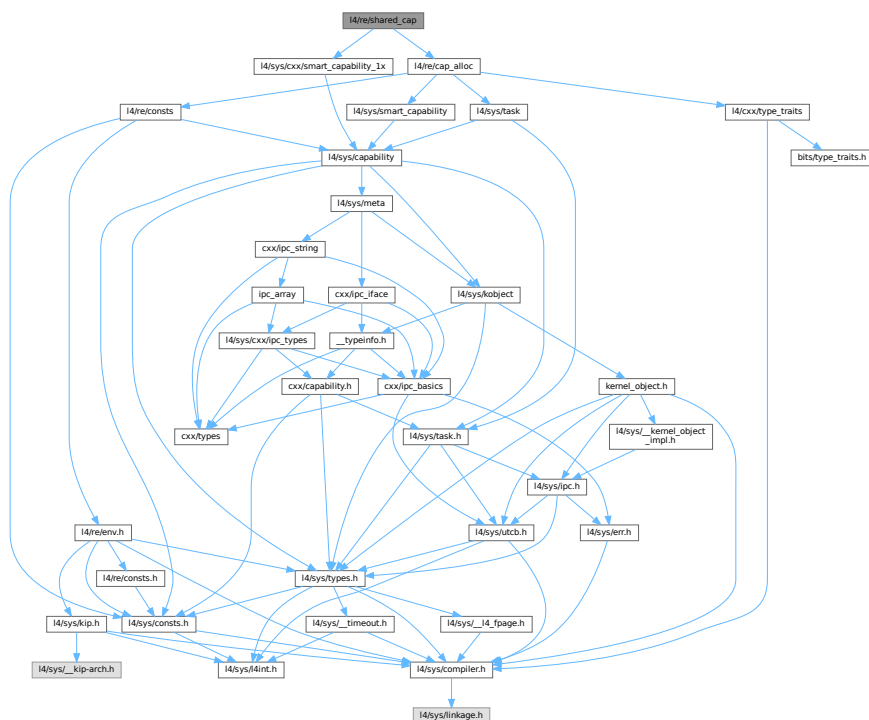
Shared\_cap / Shared\_del\_cap.

```

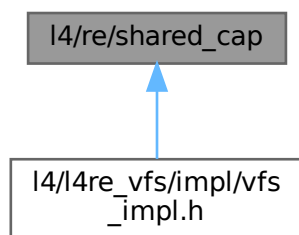
#include <l4/re/cap_alloc>
#include <l4/sys/cxx/smart_capability_1x>

```

Include dependency graph for shared\_cap:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace **L4Re**  
*L4Re C++ Interfaces.*

## Typedefs

- `template<typename T>`  
`using L4Re::Shared_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES>>`

*Shared capability that implements automatic free and unmap of the capability selector.*

- `template<typename T>`  
`using L4Re::shared_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES>>`  
*Shared capability that implements automatic free and unmap of the capability selector.*
- `template<typename T>`  
`using L4Re::Shared_del_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>>`  
*Shared capability that implements automatic free and unmap+delete of the capability selector.*
- `template<typename T>`  
`using L4Re::shared_del_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>>`  
*Shared capability that implements automatic free and unmap+delete of the capability selector.*

## Functions

- `template<typename T>`  
`Shared_cap< T > L4Re::make_shared_cap (L4Re::Cap_alloc *ca)`  
*Allocate a capability slot and wrap it in a [Shared\\_cap](#).*
- `template<typename T>`  
`Shared_del_cap< T > L4Re::make_shared_del_cap (L4Re::Cap_alloc *ca)`  
*Allocate a capability slot and wrap it in a [Shared\\_del\\_cap](#).*

## 17.414.1 Detailed Description

`Shared_cap` / `Shared_del_cap`.

Definition in file [shared\\_cap](#).

## 17.415 shared\_cap

[Go to the documentation of this file.](#)

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2018 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011
00012 #pragma once
00013
00014 #include <l4/re/cap_alloc>
00015 #include <l4/sys/cxx/smart_capability_1x>
00016
00017 namespace L4Re {
00018
00032 template< typename T >
00033 using Shared_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES>;
00035 template< typename T >
00036 using shared_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES>;
00037
00047 template< typename T >
00048 Shared_cap<T>
00049 make_shared_cap(L4Re::Cap_alloc *ca)
00050 { return Shared_cap<T>(ca->alloc<T>(), ca); }
00051
00068 template< typename T >
00069 using Shared_del_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>;
00071 template<typename T>
00072 using shared_del_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>;
00073
00083 template< typename T >
00084 Shared_del_cap<T>
00085 make_shared_del_cap(L4Re::Cap_alloc *ca)
00086 { return Shared_del_cap<T>(ca->alloc<T>(), ca); }
00087
00088 } // namespace L4Re
```



## Typedefs

- `template<typename T>`  
`using L4Re::Util::Shared_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES>>`  
*Shared capability that implements automatic free and unmap of the capability selector.*
- `template<typename T>`  
`using L4Re::Util::shared_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES>>`  
*Shared capability that implements automatic free and unmap of the capability selector.*
- `template<typename T>`  
`using L4Re::Util::Shared_del_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>>`  
*Shared capability that implements automatic free and unmap+delete of the capability selector.*
- `template<typename T>`  
`using L4Re::Util::shared_del_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>>`  
*Shared capability that implements automatic free and unmap+delete of the capability selector.*

## Functions

- `template<typename T>`  
`Shared_cap< T > L4Re::Util::make_shared_cap ()`  
*Allocate a capability slot and wrap it in a [Shared\\_cap](#).*
- `template<typename T>`  
`Shared_del_cap< T > L4Re::Util::make_shared_del_cap ()`  
*Allocate a capability slot and wrap it in a [Shared\\_del\\_cap](#).*

### 17.416.1 Detailed Description

`Shared_cap` / `Shared_del_cap`.

Definition in file [shared\\_cap](#).

## 17.417 shared\_cap

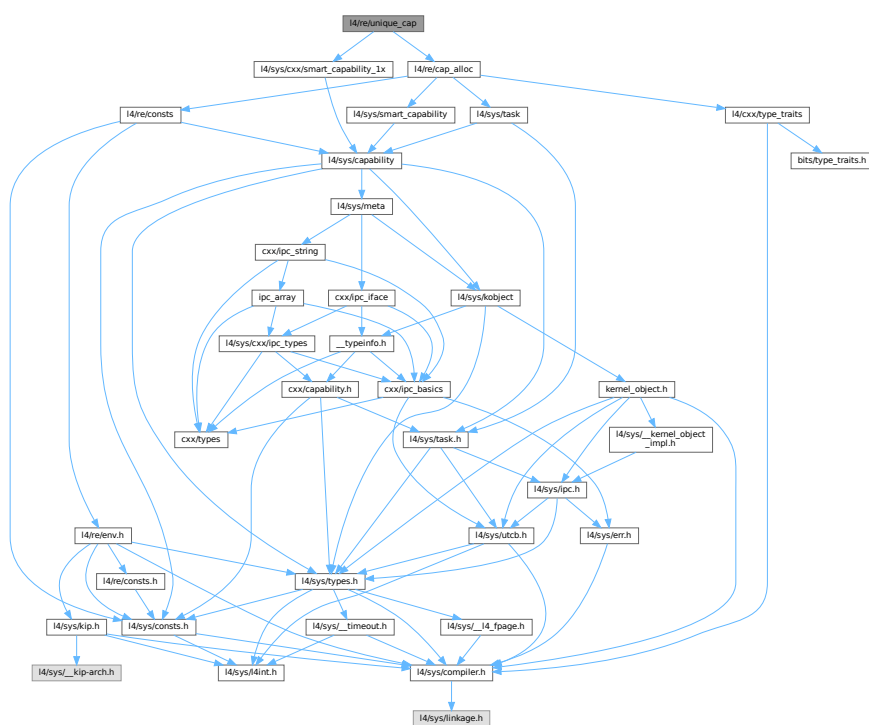
[Go to the documentation of this file.](#)

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011
00012 #pragma once
00013
00014 #include <l4/re/util/cap_alloc>
00015 #include <l4/sys/cxx/smart_capability_lx>
00016
00017 namespace L4Re { namespace Util {
00018
00047 template< typename T >
00048 using Shared_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES>;
00050 template< typename T >
00051 using shared_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_ALL_SPACES>;
00052
00058 template< typename T >
00059 Shared_cap<T>
00060 make_shared_cap()
00061 { return Shared_cap<T>(cap_alloc.alloc<T>()); }
00062
00097 template< typename T >
00098 using Shared_del_cap = L4::Detail::Shared_cap_impl<T, Smart_count_cap<L4_FP_DELETE_OBJ>;
00100 template< typename T >
```

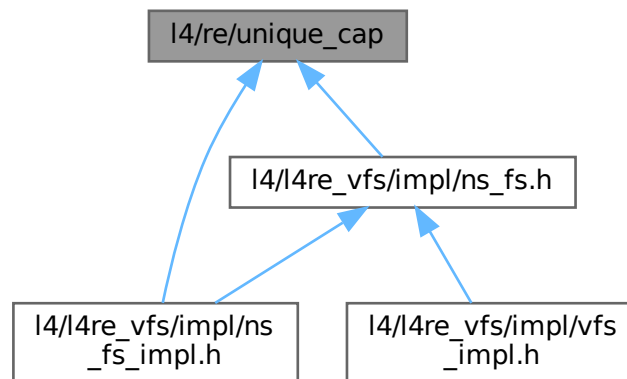


### 17.418 I4/re/unique\_cap File Reference

```
#include <l4/re/cap_alloc>
#include <l4/sys/cxx/smart_capability_1x>
Include dependency graph for unique_cap:
```



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [L4Re](#)  
[L4Re](#) C++ Interfaces.

## Typedefs

- `template<typename T>`  
`using L4Re::Unique\_cap = L4::Detail::Unique_cap_impl<T, Smart\_cap\_auto<L4\_FP\_ALL\_SPACES>>`  
*Unique capability that implements automatic free and unmap of the capability selector.*
- `template<typename T>`  
`using L4Re::unique\_cap = L4::Detail::Unique_cap_impl<T, Smart\_cap\_auto<L4\_FP\_ALL\_SPACES>>`  
*Unique capability that implements automatic free and unmap of the capability selector.*
- `template<typename T>`  
`using L4Re::Unique\_del\_cap = L4::Detail::Unique_cap_impl<T, Smart\_cap\_auto<L4\_FP\_DELETE\_OBJ>>`  
*Unique capability that implements automatic free and unmap+delete of the capability selector.*
- `template<typename T>`  
`using L4Re::unique\_del\_cap = L4::Detail::Unique_cap_impl<T, Smart\_cap\_auto<L4\_FP\_DELETE\_OBJ>>`  
*Unique capability that implements automatic free and unmap+delete of the capability selector.*

## Functions

- `template<typename T>`  
`Unique\_cap< T > L4Re::make\_unique\_cap (L4Re::Cap\_alloc *ca)`  
*Allocate a capability slot and wrap it in an [Unique\\_cap](#).*
- `template<typename T>`  
`Unique\_del\_cap< T > L4Re::make\_unique\_del\_cap (L4Re::Cap\_alloc *ca)`  
*Allocate a capability slot and wrap it in an [Unique\\_del\\_cap](#).*

## 17.418.1 Detailed Description

Unique\_cap / Unique\_del\_cap.

Definition in file [unique\\_cap](#).

## 17.419 unique\_cap

[Go to the documentation of this file.](#)

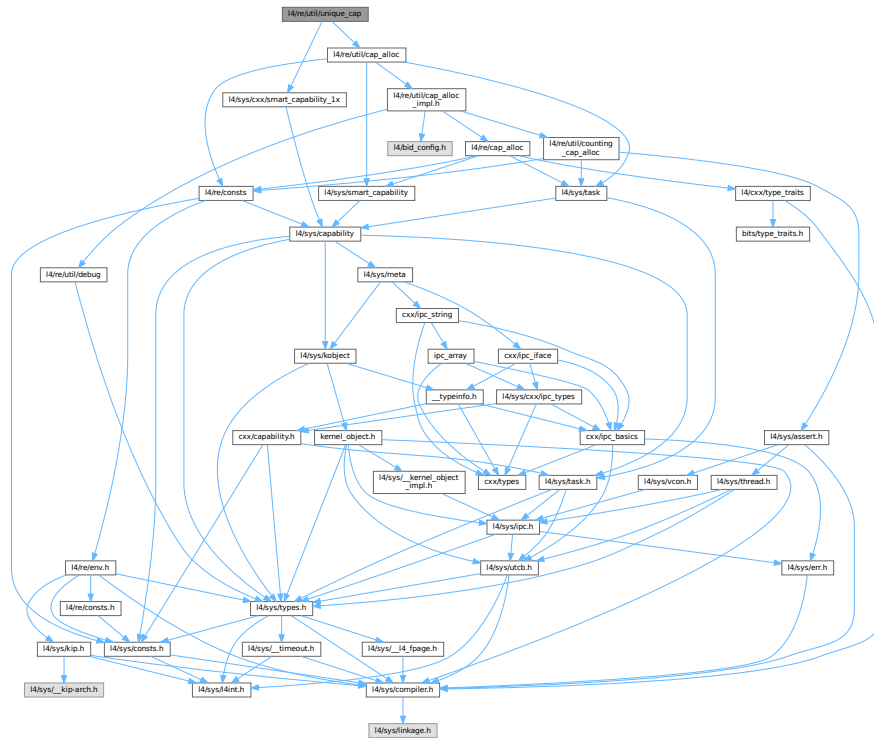
```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011
00012 #pragma once
00013
00014 #include <l4/re/cap_alloc>
00015 #include <l4/sys/cxx/smart_capability_lx>
00016
00017 namespace L4Re {
00018
00030 template< typename T >
00031 using Unique_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>;
00033 template< typename T >
00034 using unique_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>;
00035
00045 template< typename T >
00046 Unique_cap<T>
00047 make_unique_cap(L4Re::Cap_alloc *ca)
00048 { return Unique_cap<T>(ca->alloc<T>(), ca); }
00049
00063 template< typename T >
00064 using Unique_del_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>;
00066 template<typename T>
00067 using unique_del_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>;
00068
00078 template< typename T >
00079 Unique_del_cap<T>
00080 make_unique_del_cap(L4Re::Cap_alloc *ca)
00081 { return Unique_del_cap<T>(ca->alloc<T>(), ca); }
00082
00083 }
```

## 17.420 l4/re/util/unique\_cap File Reference

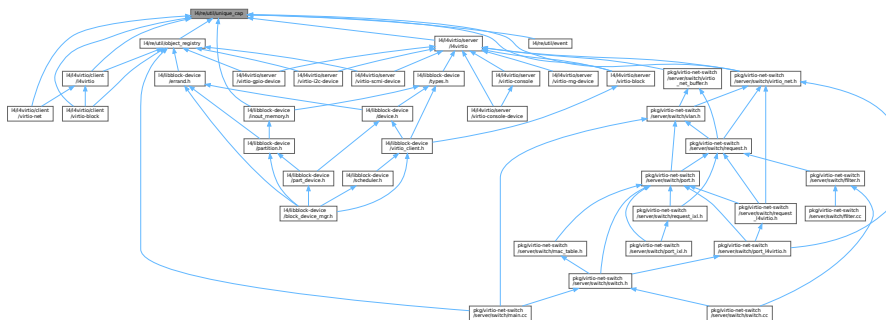
Unique\_cap / Unique\_del\_cap.

```
#include <l4/re/util/cap_alloc>
#include <l4/sys/cxx/smart_capability_lx>
```

Include dependency graph for unique\_cap:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [L4Re](#)  
*[L4Re C++ Interfaces](#).*
- namespace [L4Re::Util](#)  
*Documentation of the [L4 Runtime Environment](#) utility functionality in C++.*

## Typedefs

- `template<typename T>`  
`using L4Re::Util::Unique_cap = L4::Detail::Unique_cap impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>>`

*Unique capability that implements automatic free and unmap of the capability selector.*

- `template<typename T>`  
`using L4Re::Util::unique_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>>`

*Unique capability that implements automatic free and unmap of the capability selector.*

- `template<typename T>`  
`using L4Re::Util::Unique_del_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>>`

*Unique capability that implements automatic free and unmap+delete of the capability selector.*

- `template<typename T>`  
`using L4Re::Util::unique_del_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>>`

*Unique capability that implements automatic free and unmap+delete of the capability selector.*

## Functions

- `template<typename T>`  
`Unique_cap< T > L4Re::Util::make_unique_cap ()`  
*Allocate a capability slot and wrap it in an `Unique_cap`.*
- `template<typename T>`  
`Unique_del_cap< T > L4Re::Util::make_unique_del_cap ()`  
*Allocate a capability slot and wrap it in an `Unique_del_cap`.*

## 17.420.1 Detailed Description

Unique\_cap / Unique\_del\_cap.

Definition in file [unique\\_cap](#).

## 17.421 unique\_cap

[Go to the documentation of this file.](#)

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011
00012 #pragma once
00013
00014 #include <l4/re/util/cap_alloc>
00015 #include <l4/sys/cxx/smart_capability_1x>
00016
00017 namespace L4Re { namespace Util {
00018
00042 template< typename T >
00043 using Unique_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>;
00044 template< typename T >
00045 using unique_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_ALL_SPACES>;
00046
00053 template< typename T >
00054 Unique_cap<T>
00055 make_unique_cap()
00056 { return Unique_cap<T>(cap_alloc.alloc<T>()); }
00057
00085 template< typename T >
00086 using Unique_del_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>;
00087 template< typename T >
00088 using unique_del_cap = L4::Detail::Unique_cap_impl<T, Smart_cap_auto<L4_FP_DELETE_OBJ>;
00089
00096 template< typename T >
00097 Unique_del_cap<T>
00098 make_unique_del_cap()
00099 { return Unique_del_cap<T>(cap_alloc.alloc<T>()); }
00100
00101 }}
00102
```

## 17.422 vcon\_svr

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2011 Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *                      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00005  *                      Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00006  *                      economic rights: Technische Universität Dresden (Germany)
00007  *
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010 #pragma once
00011
00012 #include <l4/sys/types.h>
00013 #include <l4/sys/vcon>
00014 #include <l4/sys/cxx/ipc_legacy>
00015 #include <l4/cxx/minmax>
00016
00017 namespace L4Re { namespace Util {
00018
00035 template< typename SVR >
00036 class Vcon_svr
00037 {
00038 public:
00039     L4_RPC_LEGACY_DISPATCH(L4::Vcon);
00040
00041     l4_msgtag_t op_dispatch(l4_utcb_t *utcb, l4_msgtag_t tag, L4::Vcon::Rights)
00042     {
00043         l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00044         L4::Opcode op = m->mr[0];
00045
00046         switch (op)
00047         {
00048             case L4_VCON_WRITE_OP:
00049                 if (tag.words() < 3)
00050                     return l4_msgtag(-L4_ENOREPLY, 0, 0, 0);
00051
00052                 this_vcon()->vcon_write(reinterpret_cast<char const *>(&m->mr[2]),
00053                                         m->mr[1]);
00054                 return l4_msgtag(-L4_ENOREPLY, 0, 0, 0);
00055             case L4_VCON_SET_ATTR_OP:
00056             {
00057                 if (tag.words() < 4)
00058                     return l4_msgtag(-L4_EINVAL, 0, 0, 0);
00059
00060                 auto attr = reinterpret_cast<l4_vcon_attr_t const *>(&m->mr[1]);
00061                 return l4_msgtag(this_vcon()->vcon_set_attr(attr), 0, 0, 0);
00062             }
00063             case L4_VCON_GET_ATTR_OP:
00064             {
00065                 auto attr = reinterpret_cast<l4_vcon_attr_t *>(&m->mr[1]);
00066                 return l4_msgtag(this_vcon()->vcon_get_attr(attr), 4, 0, 0);
00067             }
00068             default:
00069                 break;
00070         }
00071
00072         unsigned const max_size = sizeof(l4_utcb_mr()->mr) - sizeof(l4_utcb_mr()->mr[0]);
00073         char buf[max_size];
00074
00075         unsigned size = cxx::min<unsigned>(op >> 16, max_size);
00076
00077         // Hmm, could we avoid the double copy here?
00078         l4_umword_t v = this_vcon()->vcon_read(buf, size);
00079         unsigned bytes = v & L4_VCON_READ_SIZE_MASK;
00080
00081         if (bytes <= size)
00082             v |= L4_VCON_READ_STAT_DONE;
00083
00084         m->mr[0] = v;
00085         __builtin_memcpy(&m->mr[1], buf, bytes);
00086
00087         return l4_msgtag(0, l4_bytes_to_mwords(bytes) + 1, 0, 0);
00088     }
00089
00090     unsigned vcon_read(char *buf, unsigned size) noexcept;
00091     void vcon_write(const char *buf, unsigned size) noexcept;
00092     int vcon_set_attr(l4_vcon_attr_t const *) noexcept
00093     { return -L4_EOK; }
00094     int vcon_get_attr(l4_vcon_attr_t *attr) noexcept
00095     {
00096         attr->l_flags = attr->o_flags = attr->i_flags = 0;
00097         return -L4_EOK;
00098     }
00099 }
00100

```

```

00101
00102 private:
00103     SVR const *this_vcon() const { return static_cast<SVR const *>(this); }
00104     SVR *this_vcon() { return static_cast<SVR *>(this); }
00105 };
00106
00107 }}

```

## 17.423 goos\_fb

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include <l4/re/env>
00012 #include <l4/re/namespace>
00013 #include <l4/re/rm>
00014 #include <l4/re/util/cap_alloc>
00015 #include <l4/re/util/env_ns>
00016 #include <l4/re/util/video/goos_fb>
00017 #include <l4/re/video/goos>
00018
00019 namespace L4Re { namespace Util { namespace Video {
00020
00021     class Goos_fb
00022     {
00023     private:
00024         L4::Cap<L4Re::Video::Goos> _goos;
00025         L4Re::Video::View _view;
00026         L4::Cap<L4Re::Dataspace> _buffer;
00027
00028         enum Flags
00029         {
00030             F_dyn_buffer = 0x01,
00031             F_dyn_view   = 0x02,
00032             F_dyn_goos    = 0x04,
00033         };
00034         unsigned _flags;
00035
00036         unsigned _buffer_index;
00037
00038     private:
00039         long init()
00040         {
00041             using namespace L4Re::Video;
00042
00043             Goos::Info gi;
00044             long ret = _goos->info(&gi);
00045             if (ret < 0)
00046                 return ret;
00047
00048             if (gi.has_dynamic_views())
00049             {
00050                 ret = _goos->create_view(&_view);
00051                 if (ret < 0)
00052                     return ret;
00053                 _flags |= F_dyn_view;
00054             }
00055             else // we just assume view 0 to be our's and ignore other possible views
00056                 _view = _goos->view(0);
00057
00058             View::Info vi;
00059             ret = _view.info(&vi);
00060             if (ret < 0)
00061                 return ret;
00062             _buffer = cap_alloc.alloc<L4Re::Dataspace>();
00063             if (!_buffer)
00064                 return -L4_ENOMEM;
00065
00066             if (vi.has_static_buffer())
00067             {
00068                 ret = _goos->get_static_buffer(vi.buffer_index, _buffer);
00069                 if (ret < 0)
00070                     return ret;
00071             }
00072
00073         }
00074     }
00075 }
00076 }

```

```

00074     else
00075     {
00076         unsigned long buffer_sz = gi.pixel_info.bytes_per_pixel() * gi.width
00077                                     * gi.height;
00078         ret = _goos->create_buffer(buffer_sz, _buffer);
00079         if (ret < 0)
00080             return ret;
00081
00082         _buffer_index = static_cast<unsigned>(ret);
00083         _flags |= F_dyn_buffer;
00084
00085         // use the allocated buffer, at offset 0
00086         vi.buffer_index = _buffer_index;
00087         vi.buffer_offset = 0;
00088         vi.pixel_info = gi.pixel_info;
00089         vi.bytes_per_line = gi.width * gi.pixel_info.bytes_per_pixel();
00090
00091         // we want a fullscreen view
00092         vi.xpos = 0;
00093         vi.ypos = 0;
00094         vi.width = gi.width;
00095         vi.height = gi.height;
00096
00097         ret = _view.set_info(vi);
00098         if (ret < 0)
00099             return ret;
00100
00101         ret = _view.push_top();
00102         if (ret < 0)
00103             return ret;
00104     }
00105
00106     return 0;
00107 }
00108
00109 Goos_fb(Goos_fb const &);
00110 void operator = (Goos_fb const &);
00111
00112 public:
00113 Goos_fb()
00114 : _goos(L4_INVALID_CAP), _buffer(L4_INVALID_CAP), _flags(0), _buffer_index(0)
00115 {}
00116
00117 long init(L4Re::Cap<L4Re::Video::Goos> goos)
00118 {
00119     _goos = goos;
00120     return init();
00121 }
00122
00123 long init(char const *name)
00124 {
00125     Env_ns ns;
00126     _goos = ns.query<L4Re::Video::Goos>(name);
00127     if (!_goos)
00128         return _goos.cap();
00129
00130     _flags |= F_dyn_goos;
00131
00132     return init();
00133 }
00134
00135 ~Goos_fb()
00136 {
00137     if (!_goos.is_valid())
00138         return;
00139
00140     if (_flags & F_dyn_view)
00141         _goos->delete_view(_view);
00142
00143     if (_flags & F_dyn_buffer)
00144         _goos->delete_buffer(_buffer_index);
00145
00146     if (_buffer.is_valid())
00147         cap_alloc.free(_buffer);
00148
00149     if (_flags & F_dyn_goos)
00150         cap_alloc.free(_goos);
00151 }
00152
00153 int view_info(L4Re::Video::View::Info *info)
00154 { return _view.info(info); }
00155
00156 L4Re::Video::View const *view() const { return &_view; }
00157 L4Re::Video::View *view() { return &_view; }
00158
00159 L4Re::Cap<L4Re::Dataspace> buffer() const { return _buffer; }
00160 void *attach_buffer()

```



```

00161 {
00162     void *fb_addr = 0;
00163     if (!_goos)
00164         return nullptr;
00165
00166     long ret = L4Re::Env::env()->rm()
00167         ->attach(&fb_addr, _buffer->size(),
00168             L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW, _buffer,
00169             0, L4_SUPERPAGESHIFT);
00170     if (ret < 0)
00171         return nullptr;
00172
00173     return fb_addr;
00174 }
00175
00176 int refresh(int x, int y, int w, int h)
00177 { return _view.refresh(x, y, w, h); }
00178
00179 L4::Cap<L4Re::Video::Goos> goos() const { return _goos; }
00180 };
00181 }}}

```

## 17.424 goos\_svr

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *     Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *     economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 #include <l4/re/dataspace>
00013 #include <l4/re/video/goos>
00014 #include <l4/re/video/goos-sys.h>
00015
00016 #include <l4/sys/capability>
00017 #include <l4/sys/cxx/ipc_legacy>
00018
00019 namespace L4Re { namespace Util { namespace Video {
00020
00021     class Goos_svr
00022     {
00023     public:
00024         typedef L4Re::Video::Goos::Rights Rights;
00025     protected:
00026         L4::Cap<L4Re::Dataspace> _fb_ds;
00027         L4Re::Video::Goos::Info _screen_info;
00028         L4Re::Video::View::Info _view_info;
00029     public:
00030         L4_RPC_LEGACY_DISPATCH(L4Re::Video::Goos);
00031         L4::Cap<L4Re::Dataspace> get_fb() const { return _fb_ds; }
00032
00033         L4Re::Video::Goos::Info const *screen_info() const { return &_screen_info; }
00034
00035         L4Re::Video::View::Info const *view_info() const { return &_view_info; }
00036
00037         virtual int refresh(int x, int y, int w, int h)
00038         { (void)x; (void)y; (void)w; (void)h; return -L4_ENOSYS; }
00039
00040         void init_infos()
00041         {
00042             using L4Re::Video::View;
00043
00044             _view_info.flags = View::F_none;
00045
00046             _view_info.view_index = 0;
00047             _view_info.xpos = 0;
00048             _view_info.ypos = 0;
00049             _view_info.width = _screen_info.width;
00050             _view_info.height = _screen_info.height;
00051             _view_info.pixel_info = _screen_info.pixel_info;
00052             _view_info.buffer_index = 0;
00053         }
00054
00055         virtual ~Goos_svr() {}
00056
00057         long op_view_info(Rights, unsigned idx, L4Re::Video::View::Info &info)
00058         {

```

```

00100     if (idx != 0)
00101         return -L4_ERANGE;
00102
00103     info = _view_info;
00104     return L4_EOK;
00105 }
00106
00107 long op_info(Rights, L4Re::Video::Goos::Info &info)
00108 {
00109     info = _screen_info;
00110     return L4_EOK;
00111 }
00112
00113 long op_get_static_buffer(Rights, unsigned idx,
00114                           L4::Ipc::Cap<L4Re::Dataspace> &ds)
00115 {
00116     if (idx != 0)
00117         return -L4_ERANGE;
00118
00119     ds = L4::Ipc::Cap<L4Re::Dataspace>(_fb_ds, L4_CAP_FPAGE_RW);
00120     return L4_EOK;
00121 }
00122
00123 long op_refresh(Rights, int x, int y, int w, int h)
00124 { return refresh(x, y, w, h); }
00125
00126 long op_view_refresh(Rights, unsigned idx, int x, int y, int w, int h)
00127 {
00128     if (idx != 0)
00129         return -L4_ERANGE;
00130
00131     return refresh(x, y, w, h);
00132 }
00133
00134 long op_set_view_info(Rights, unsigned, L4Re::Video::View::Info)
00135 { return -L4_ENOSYS; }
00136
00137 long op_view_stack(Rights, unsigned, unsigned, bool)
00138 { return -L4_ENOSYS; }
00139
00140 long op_delete_view(Rights, unsigned)
00141 { return -L4_ENOSYS; }
00142
00143 long op_create_view(Rights)
00144 { return -L4_ENOSYS; }
00145
00146 long op_create_buffer(Rights, unsigned long,
00147                       L4::Ipc::Cap<L4Re::Dataspace> &)
00148 { return -L4_ENOSYS; }
00149
00150 long op_delete_buffer(Rights, unsigned)
00151 { return -L4_ENOSYS; }
00152 };
00153
00154
00155 }}}

```

## 17.425 colors

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 #include <l4/sys/compiler.h>
00013 #include <l4/cxx/minmax>
00014
00015 namespace L4Re { namespace Video {
00016
00021 class L4_EXPORT Color_component
00022 {
00023 private:
00024     unsigned char _bits;
00025     unsigned char _shift;
00026
00027 public:
00029     Color_component() : _bits(0), _shift(0) {}

```

```

00030
00036 Color_component(unsigned char bits, unsigned char shift)
00037 : _bits(bits), _shift(shift) {}
00038
00043 unsigned char size() const { return _bits; }
00044
00049 unsigned char shift() const { return _shift; }
00050
00055 bool operator == (Color_component const &o) const
00056 { return _shift == o._shift && _bits == o._bits; }
00057
00063 int get(unsigned long v) const
00064 {
00065     return ((v > _shift) & ~(~0UL << _bits)) << (16UL - _bits);
00066 }
00067
00073 long unsigned set(int v) const
00074 { return (static_cast<unsigned long>(v) > (16UL - _bits)) << _shift; }
00075
00080 template< typename OUT >
00081 void dump(OUT &s) const
00082 {
00083     s.printf("%d(%d)", static_cast<int>(size()), static_cast<int>(shift()));
00084 }
00085 } __attribute__((packed));
00086
00094 class L4_EXPORT Pixel_info
00095 {
00096 private:
00097     Color_component _r, _g, _b, _a;
00098     unsigned char _bpp;
00099 public:
00105 Color_component const &r() const { return _r; }
00106
00111 Color_component const &g() const { return _g; }
00112
00117 Color_component const &b() const { return _b; }
00118
00123 Color_component const &a() const { return _a; }
00124
00131 Color_component const padding() const
00132 {
00133     unsigned char top_bit = cxx::max<unsigned char>(_r.size() + _r.shift(),
00134                                                     _g.size() + _g.shift());
00135     top_bit = cxx::max<unsigned char>(top_bit, _b.size() + _b.shift());
00136     top_bit = cxx::max<unsigned char>(top_bit, _a.size() + _a.shift());
00137
00138     unsigned char bits = _bpp * 8;
00139
00140     if (top_bit < bits)
00141         return Color_component(bits - top_bit, top_bit);
00142
00143     return Color_component(0, 0);
00144 }
00145
00150 unsigned char bytes_per_pixel() const { return _bpp; }
00151
00156 unsigned char bits_per_pixel() const
00157 { return _r.size() + _g.size() + _b.size() + _a.size(); }
00158
00163 bool has_alpha() const { return _a.size() > 0; }
00164
00169 void r(Color_component const &c) { _r = c; }
00170
00175 void g(Color_component const &c) { _g = c; }
00176
00181 void b(Color_component const &c) { _b = c; }
00182
00187 void a(Color_component const &c) { _a = c; }
00188
00193 void bytes_per_pixel(unsigned char bpp) { _bpp = bpp; }
00194
00198 Pixel_info() : _bpp(0) {};
00199
00212 Pixel_info(unsigned char bpp, char r, char rs, char g, char gs,
00213             char b, char bs, char a = 0, char as = 0)
00214 : _r(r, rs), _g(g, gs), _b(b, bs), _a(a, as), _bpp(bpp)
00215 {}
00216
00223 template<typename VBI>
00224 explicit Pixel_info(VBI const *vbi)
00225 : _r(vbi->red_mask_size, vbi->red_field_position),
00226   _g(vbi->green_mask_size, vbi->green_field_position),
00227   _b(vbi->blue_mask_size, vbi->blue_field_position),
00228   _bpp((vbi->bits_per_pixel + 7) / 8)
00229 {}

```

```

00230
00236 bool operator == (Pixel_info const &o) const
00237 {
00238     return _r == o._r && _g == o._g && _b == o._b && _a == o._a && _bpp == o._bpp;
00239 }
00240
00245 template< typename OUT >
00246 void dump(OUT &s) const
00247 {
00248     s.printf("RGBA(%d):%d(%d):%d(%d):%d(%d)",
00249             static_cast<int>(bytes_per_pixel()),
00250             static_cast<int>(r().size()), static_cast<int>(r().shift()),
00251             static_cast<int>(g().size()), static_cast<int>(g().shift()),
00252             static_cast<int>(b().size()), static_cast<int>(b().shift()),
00253             static_cast<int>(a().size()), static_cast<int>(a().shift()));
00254 }
00255 };
00256
00257
00258 }}
00259
00260

```

## 17.426 goos

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include <l4/sys/capability>
00012 #include <l4/re/dataspace>
00013 #include <l4/re/video/colors>
00014 #include <l4/sys/cxx/ipc_iface>
00015
00016 namespace L4Re { namespace Video {
00017
00026 class L4_EXPORT Goos;
00027
00039 class L4_EXPORT View
00040 {
00041 private:
00042     friend class Goos;
00043
00044     L4::Cap<Goos> _goos;
00045     unsigned _view_idx;
00046
00047     View(l4_cap_idx_t goos, unsigned idx)
00048     : _goos(goos), _view_idx(_goos.is_valid() ? idx : ~0U) {}
00049
00050     unsigned view_index() const noexcept
00051     { return _goos.is_valid() ? _view_idx : ~0U; }
00052
00053 public:
00054     View() : _goos(L4::Cap<Goos>::Invalid), _view_idx(~0U) {}
00055
00059     enum Flags
00060     {
00061         F_none                = 0x00,
00062         F_set_buffer           = 0x01,
00063         F_set_buffer_offset    = 0x02,
00064         F_set_bytes_per_line   = 0x04,
00065         F_set_pixel            = 0x08,
00066         F_set_position         = 0x10,
00067         F_dyn_allocated        = 0x20,
00068         F_set_background       = 0x40,
00069         F_set_flags            = 0x80,
00070
00072         F_fully_dynamic        = F_set_buffer | F_set_buffer_offset | F_set_bytes_per_line
00073                               | F_set_pixel | F_set_position | F_dyn_allocated,
00074     };
00075
00082     enum V_flags
00083     {
00084         F_above                = 0x1000,
00085         F_flags_mask           = 0xff000,
00086     };
00087

```

```

00091 struct Info
00092 {
00093     unsigned flags           = 0;
00094     unsigned view_index     = 0;
00095
00096     unsigned long xpos      = 0;
00097     unsigned long ypos      = 0;
00098     unsigned long width     = 0;
00099     unsigned long height    = 0;
00100     unsigned long buffer_offset = 0;
00101     unsigned long bytes_per_line = 0;
00102     Pixel_info pixel_info;
00103     unsigned buffer_index    = 0;
00104
00106     bool has_static_buffer() const { return !(flags & F_set_buffer); }
00108     bool has_static_buffer_offset() const { return !(flags & F_set_buffer_offset); }
00109
00111     bool has_set_buffer() const { return flags & F_set_buffer; }
00113     bool has_set_buffer_offset() const { return flags & F_set_buffer_offset; }
00115     bool has_set_bytes_per_line() const { return flags & F_set_bytes_per_line; }
00117     bool has_set_pixel() const { return flags & F_set_pixel; }
00119     bool has_set_position() const { return flags & F_set_position; }
00120
00122     template< typename OUT >
00123     void dump(OUT &s) const
00124     {
00125         s.printf("View::Info:\n"
00126                 "  flags: %x\n"
00127                 "  size: %ldx%ld\n"
00128                 "  pos: %ldx%ld\n"
00129                 "  bytes_per_line: %ld\n"
00130                 "  buffer_offset: %lx\n"
00131                 "  ",
00132                 flags, width, height, xpos, ypos,
00133                 bytes_per_line, buffer_offset);
00134         pixel_info.dump(s);
00135         s.printf("\n");
00136     }
00137 };
00138
00146     int info(Info *info) const noexcept;
00147
00158     int set_info(Info const &info) const noexcept;
00159
00171     int set_viewport(int scr_x, int scr_y, int w, int h, unsigned long buf_offset) const noexcept;
00172
00182     int stack(View const &pivot, bool behind = true) const noexcept;
00183
00185     int push_top() const noexcept
00186     { return stack(View(), true); }
00187
00189     int push_bottom() const noexcept
00190     { return stack(View(), false); }
00191
00202     int refresh(int x, int y, int w, int h) const noexcept;
00203
00205     bool valid() const { return _goos.is_valid(); }
00206 };
00207
00208
00223 class L4_EXPORT Goos :
00224     public L4::Kobject_t<Goos, L4::Kobject, L4RE_PROTO_GOOS>
00225 {
00226 public:
00228     enum Flags
00229     {
00230         F_auto_refresh      = 0x01,
00231         F_pointer           = 0x02,
00232         F_dynamic_views     = 0x04,
00233         F_dynamic_buffers   = 0x08,
00234     };
00235
00237     struct Info
00238     {
00239         unsigned long width;
00240         unsigned long height;
00241         unsigned flags;
00242         unsigned num_static_views;
00243         unsigned num_static_buffers;
00244         Pixel_info pixel_info;
00245
00248         bool auto_refresh() const { return flags & F_auto_refresh; }
00250         bool has_pointer() const { return flags & F_pointer; }
00252         bool has_dynamic_views() const { return flags & F_dynamic_views; }
00254         bool has_dynamic_buffers() const { return flags & F_dynamic_buffers; }
00255
00256         Info()

```

```

00257     : width(0), height(0), flags(0), num_static_views(0),
00258       num_static_buffers(0) {}
00259 };
00260
00268 L4_INLINE_RPC(long, info, (Info *info));
00269
00278 L4_RPC(long, get_static_buffer, (unsigned idx,
00279                                L4::Ipc::Out<L4::Cap<L4Re::Dataspace> > rbuf));
00280
00289 L4_RPC(long, create_buffer, (unsigned long size,
00290                             L4::Ipc::Out<L4::Cap<L4Re::Dataspace> > rbuf));
00291
00299 L4_INLINE_RPC(long, delete_buffer, (unsigned idx));
00300
00301 // Use a wrapper for this RPC as we encapsulate the View
00302 L4_INLINE_RPC_NF(long, create_view, ());
00303
00312 int create_view(View *view, l4_utcb_t *utcb = l4_utcb()) const noexcept
00313 {
00314     long r = create_view_t::call(c(), utcb);
00315     if (r < 0)
00316         return r;
00317     *view = View(cap(), r);
00318     return r;
00319 }
00320
00321 // Use a wrapper as Views are encapsulated
00322 L4_INLINE_RPC_NF(long, delete_view, (unsigned index));
00323
00332 int delete_view(View const &v, l4_utcb_t *utcb = l4_utcb()) const noexcept
00333 {
00334     return delete_view_t::call(c(), v._view_idx, utcb);
00335 }
00336
00342 View view(unsigned index) const noexcept;
00343
00347 L4_INLINE_RPC(long, refresh, (int x, int y, int w, int h));
00348
00349 // those are used by the View
00350 L4_INLINE_RPC(long, view_info, (unsigned index, View::Info *info));
00351 L4_INLINE_RPC(long, set_view_info, (unsigned index, View::Info const &info));
00352 L4_INLINE_RPC(long, view_stack, (unsigned index, unsigned pivot, bool behind));
00353 L4_INLINE_RPC(long, view_refresh, (unsigned index, int x, int y, int w, int h));
00354
00355 typedef L4::TypeId::Rpc<
00356     info_t, get_static_buffer_t, create_buffer_t, create_view_t, delete_buffer_t,
00357     delete_view_t, view_info_t, set_view_info_t, view_stack_t, view_refresh_t,
00358     refresh_t
00359 > Rpc<
00360 >;
00361
00362 inline View
00363 Goos::view(unsigned index) const noexcept
00364 { return View(cap(), index); }
00365
00366 inline int
00367 View::info(Info *info) const noexcept
00368 { return _goos->view_info(_view_idx, info); }
00369
00370 inline int
00371 View::set_info(Info const &info) const noexcept
00372 { return _goos->set_view_info(_view_idx, info); }
00373
00374 inline int
00375 View::stack(View const &pivot, bool behind) const noexcept
00376 { return _goos->view_stack(_view_idx, pivot._view_idx, behind); }
00377
00378 inline int
00379 View::refresh(int x, int y, int w, int h) const noexcept
00380 { return _goos->view_refresh(_view_idx, x, y, w, h); }
00381
00382 inline int
00383 View::set_viewport(int scr_x, int scr_y, int w, int h,
00384                   unsigned long buf_offset) const noexcept
00385 {
00386     Info i;
00387     i.flags = F_set_buffer_offset | F_set_position;
00388     i.buffer_offset = buf_offset;
00389     i.buffer_index = 0;
00390     i.view_index = 0;
00391     i.bytes_per_line = 0;
00392     i.pixel_info = Pixel_info();
00393     i.xpos = scr_x;
00394     i.ypos = scr_y;
00395     i.width = w;
00396     i.height = h;
00397     return set_info(i);

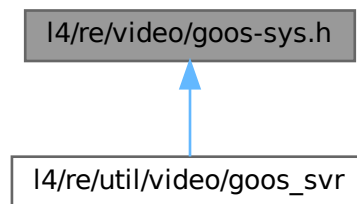
```

```
00398 }  
00399  
00400 }}
```

## 17.427 l4/re/video/goos-sys.h File Reference

Goos protocol definition.

This graph shows which files directly or indirectly include this file:



### Namespaces

- namespace [L4Re](#)  
*L4Re C++ Interfaces.*

### Enumerations

- enum [L4Re::Video::Goos\\_::Opcodes](#)  
*Frame buffer communication-protocol opcodes.*

### 17.427.1 Detailed Description

Goos protocol definition.

Definition in file [goos-sys.h](#).

## 17.428 goos-sys.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Björn Döbel <doebel@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014
00015 namespace L4Re { namespace Video {
00016     namespace Goos_
00017     {
00023         enum Opcodes
00024         {
00025             Info, Get_buffer, Create_buffer, Create_view,
00026             Delete_buffer, Delete_view,
00027             View_info, View_set_info, View_stack, View_refresh,
00028             Screen_refresh
00029         };
00030     };
00031 }
```

## 17.429 view

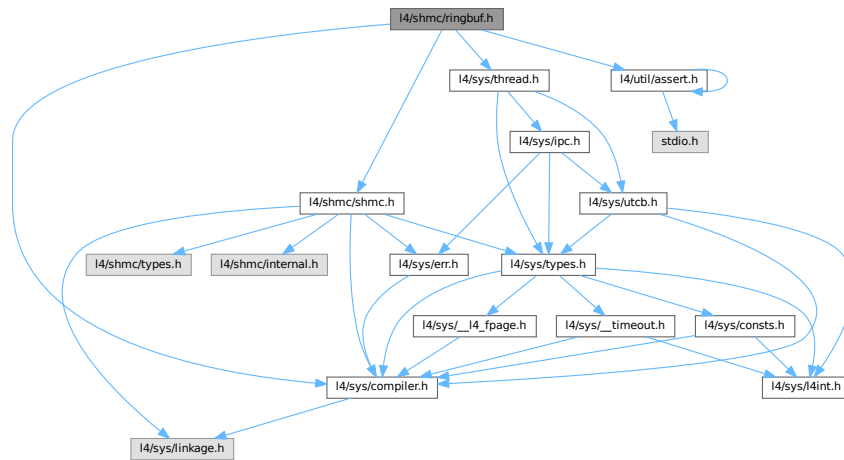
```
00001 // vi:set ft=c++: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include <l4/re/video/goos>
00012
```

## 17.430 l4/shmc/ringbuf.h File Reference

```
#include <l4/shmc/shmc.h>
#include <l4/util/assert.h>
#include <l4/sys/compiler.h>
#include <l4/sys/thread.h>
```



Include dependency graph for ringbuf.h:



## Data Structures

- struct [l4shmc\\_ringbuf\\_head\\_t](#)  
*Head field of a ring buffer.*
- struct [l4shmc\\_ringbuf\\_t](#)  
*Ring buffer.*

## Macros

- #define [L4SHMC\\_RINGBUF\\_HEAD](#)(ringbuf)  
*Get ring buffer head pointer.*
- #define [L4SHMC\\_RINGBUF\\_DATA](#)(ringbuf)  
*Get ring buffer data pointer.*
- #define [L4SHMC\\_RINGBUF\\_DATA\\_SIZE](#)(ringbuf)  
*Get size of data area.*

## Functions

- int [l4shmc\\_rb\\_init\\_buffer](#) ([l4shmc\\_ringbuf\\_t](#) \*buf, [l4shmc\\_area\\_t](#) \*area, char const \*chunk\_name, char const \*signal\_name, unsigned size)  
*Initialize a ring buffer by creating an SHMC chunk and the corresponding signals.*
- void [l4shmc\\_rb\\_deinit\\_buffer](#) ([l4shmc\\_ringbuf\\_t](#) \*buf)  
*De-init a ring buffer.*
- int [l4shmc\\_rb\\_attach\\_sender](#) ([l4shmc\\_ringbuf\\_t](#) \*buf, char const \*signal\_name, [l4\\_cap\\_idx\\_t](#) owner)  
*Attach to sender signal of a ring buffer.*
- char \* [l4shmc\\_rb\\_sender\\_alloc\\_packet](#) ([l4shmc\\_ringbuf\\_head\\_t](#) \*head, unsigned psize)  
*Allocate a packet of a given size within the ring buffer.*
- void [l4shmc\\_rb\\_sender\\_put\\_data](#) ([l4shmc\\_ringbuf\\_t](#) \*buf, char \*addr, char \*data, unsigned dsize)  
*Copy data into a previously allocated packet.*
- int [l4shmc\\_rb\\_sender\\_next\\_copy\\_in](#) ([l4shmc\\_ringbuf\\_t](#) \*buf, char \*data, unsigned size, int block\_if\_↵ necessary)

- Copy in packet from an external data source.*
- void [l4shmc\\_rb\\_sender\\_commit\\_packet](#) ([l4shmc\\_ringbuf\\_t](#) \*buf)
  - Tell the consumer that new data is available.*
- int [l4shmc\\_rb\\_init\\_receiver](#) ([l4shmc\\_ringbuf\\_t](#) \*buf, [l4shmc\\_area\\_t](#) \*area, char const \*chunk\_name, char const \*signal\_name)
  - Initialize receive buffer.*
- void [l4shmc\\_rb\\_attach\\_receiver](#) ([l4shmc\\_ringbuf\\_t](#) \*buf, [l4\\_cap\\_idx\\_t](#) owner)
  - Attach to receiver signal of a ring buffer.*
- int [l4shmc\\_rb\\_receiver\\_wait\\_for\\_data](#) ([l4shmc\\_ringbuf\\_t](#) \*buf, int blocking)
  - Check if (and optionally block until) new data is ready.*
- int [l4shmc\\_rb\\_receiver\\_copy\\_out](#) ([l4shmc\\_ringbuf\\_head\\_t](#) \*head, char \*target, unsigned \*tsize)
  - Copy data out of the buffer.*
- void [l4shmc\\_rb\\_receiver\\_notify\\_done](#) ([l4shmc\\_ringbuf\\_t](#) \*buf)
  - Notify producer that space is available.*
- int [l4shmc\\_rb\\_receiver\\_read\\_next\\_size](#) ([l4shmc\\_ringbuf\\_head\\_t](#) \*head)
  - Have a look at the ring buffer and see which size the next packet to be read has.*

## 17.430.1 Function Documentation

### 17.430.1.1 [l4shmc\\_rb\\_attach\\_receiver\(\)](#)

```
void l4shmc_rb_attach_receiver (
    l4shmc\_ringbuf\_t * buf,
    l4\_cap\_idx\_t owner)
```

Attach to receiver signal of a ring buffer.

Attach owner to the receiver-side signal of a ring buffer, which is triggered whenever new data has been produced.

This is split from initialization, because you may not know the owner cap when initializing the buffer.

#### Parameters

<i>buf</i>	pointer to ring buffer struct
<i>owner</i>	owner thread

References [L4\\_CV](#).

### 17.430.1.2 [l4shmc\\_rb\\_attach\\_sender\(\)](#)

```
int l4shmc_rb_attach_sender (
    l4shmc\_ringbuf\_t * buf,
    char const * signal_name,
    l4\_cap\_idx\_t owner)
```

Attach to sender signal of a ring buffer.

Attach owner to the sender-side signal of a ring buffer, which is triggered whenever new space has been freed in the buffer for the sender to write to.

This is split from initialization, because you may not know the owner cap when initializing the buffer.

#### Parameters

<i>buf</i>	pointer to ring buffer struct
<i>signal_name</i>	signal base name
<i>owner</i>	owner thread

#### Returns

0 on success, error otherwise

References [L4\\_CV](#).

#### 17.430.1.3 l4shmc\_rb\_deinit\_buffer()

```
void l4shmc_rb_deinit_buffer (  
    l4shmc_ringbuf_t * buf)
```

De-init a ring buffer.

#### Parameters

<i>buf</i>	pointer to ring buffer struct
------------	-------------------------------

References [L4\\_CV](#).

#### 17.430.1.4 l4shmc\_rb\_init\_buffer()

```
int l4shmc_rb_init_buffer (  
    l4shmc_ringbuf_t * buf,  
    l4shmc_area_t * area,  
    char const * chunk_name,  
    char const * signal_name,  
    unsigned size)
```

Initialize a ring buffer by creating an SHMC chunk and the corresponding signals.

This needs to be done by one of the participating parties when setting up communication channel.

#### Precondition

area has been attached using [l4shmc\\_attach\(\)](#).

#### Parameters

<i>buf</i>	pointer to ring buffer struct
<i>area</i>	pointer to SHMC area
<i>chunk_name</i>	name of SHMC chunk to create in area

<i>signal_name</i>	base name for SHMC signals to create
<i>size</i>	chunk size

**Returns**

0 on success, error otherwise

References [L4\\_CV](#).

**17.430.1.5 l4shmc\_rb\_init\_receiver()**

```
int l4shmc_rb_init_receiver (  
    l4shmc_ringbuf_t * buf,  
    l4shmc_area_t * area,  
    char const * chunk_name,  
    char const * signal_name)
```

Initialize receive buffer.

Initialize the receiver-side of a ring buffer. This requires the underlying SHMC chunk and the corresponding signals to be valid already (read: to be initialized by the sender).

**Precondition**

chunk & signals have been created and initialized by the sender side

**Parameters**

<i>buf</i>	pointer to ring buffer struct
<i>area</i>	pointer to SHMC area
<i>chunk_name</i>	name of SHMC chunk to create in area
<i>signal_name</i>	base name for SHMC signals to create

**Returns**

0 on success, error otherwise

References [L4\\_CV](#).

**17.430.1.6 l4shmc\_rb\_receiver\_copy\_out()**

```
int l4shmc_rb_receiver_copy_out (  
    l4shmc_ringbuf_head_t * head,  
    char * target,  
    unsigned * tsize)
```

Copy data out of the buffer.

**Parameters**

	<i>head</i>	ring buffer head pointer
	<i>target</i>	valid target buffer
<i>in, out</i>	<i>size</i>	size of target buffer (must be $\geq$ packet size!); contains the real data size

**Returns**

0 on success, negative error otherwise

References [L4\\_CV](#).

**17.430.1.7 l4shmc\_rb\_receiver\_notify\_done()**

```
void l4shmc_rb_receiver_notify_done (
    l4shmc_ringbuf_t * buf)
```

Notify producer that space is available.

**Parameters**

<i>buf</i>	pointer to ring buffer struct
------------	-------------------------------

References [L4\\_CV](#).

**17.430.1.8 l4shmc\_rb\_receiver\_read\_next\_size()**

```
int l4shmc_rb_receiver_read_next_size (
    l4shmc_ringbuf_head_t * head)
```

Have a look at the ring buffer and see which size the next packet to be read has.

Does not modify anything.

**Returns**

size of next buffer or -1 if no data available

References [L4\\_CV](#), and [L4\\_END\\_DECLS](#).

**17.430.1.9 l4shmc\_rb\_receiver\_wait\_for\_data()**

```
int l4shmc_rb_receiver_wait_for_data (
    l4shmc_ringbuf_t * buf,
    int blocking)
```

Check if (and optionally block until) new data is ready.

**Parameters**

<i>buf</i>	pointer to ring buffer struct
<i>blocking</i>	block if data is not available immediately

Returns immediately, if data is available.

#### Returns

0 success, data available, != 0 otherwise

References [L4\\_CV](#).

#### 17.430.1.10 l4shmc\_rb\_sender\_alloc\_packet()

```
char * l4shmc_rb_sender_alloc_packet (
    l4shmc_ringbuf_head_t * head,
    unsigned psize)
```

Allocate a packet of a given size within the ring buffer.

This packet may wrap around at the end of the buffer. Users need to be aware of that.

#### Parameters

<i>head</i>	ring buffer head pointer
<i>psize</i>	packet size

#### Returns

valid address on success

#### Return values

<i>NULL</i>	if not enough space available
-------------	-------------------------------

References [L4\\_CV](#).

#### 17.430.1.11 l4shmc\_rb\_sender\_commit\_packet()

```
void l4shmc_rb_sender_commit_packet (
    l4shmc_ringbuf_t * buf)
```

Tell the consumer that new data is available.

#### Parameters

---

<i>buf</i>	pointer to ring buffer struct
------------	-------------------------------

References [L4\\_CV](#).

#### 17.430.1.12 l4shmc\_rb\_sender\_next\_copy\_in()

```
int l4shmc_rb_sender_next_copy_in (  
    l4shmc_ringbuf_t * buf,  
    char * data,  
    unsigned size,  
    int block_if_necessary)
```

Copy in packet from an external data source.

This is the function you'll want to use. Just pass it a buffer pointer and let the lib do the work.

##### Parameters

<i>buf</i>	pointer to ring buffer struct
<i>data</i>	valid buffer
<i>size</i>	data size
<i>block_if_necessary</i>	bool: block if buffer currently full

##### Return values

<i>0</i>	on success
<i>-L4_ENOMEM</i>	if block == false and no space available

References [L4\\_CV](#).

#### 17.430.1.13 l4shmc\_rb\_sender\_put\_data()

```
void l4shmc_rb_sender_put_data (  
    l4shmc_ringbuf_t * buf,  
    char * addr,  
    char * data,  
    unsigned dsize)
```

Copy data into a previously allocated packet.

This function is wrap-around aware.

##### Parameters

<i>buf</i>	pointer to ring buffer struct
------------	-------------------------------

<i>addr</i>	valid destination (allocate with <code>alloc_packet()</code> )
<i>data</i>	data source
<i>dsize</i>	data size

References [L4\\_CV](#).

## 17.431 ringbuf.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * (c) 2010 Björn Döbel <doebel@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  * This file is part of TUD:OS and distributed under the terms of the
00005  * GNU Lesser General Public License 2.1.
00006  * Please see the COPYING-LGPL-2.1 file for details.
00007  */
00008
00012 #pragma once
00013
00014 #include <l4/shmc/shmc.h>
00015 #include <l4/util/assert.h>
00016 #include <l4/sys/compiler.h>
00017 #include <l4/sys/thread.h>
00018
00019 L4_BEGIN_DECLS
00020
00035
00046
00047 /*
00048  * Turn on ringbuf poisoning. This will add magic values to the ringbuf
00049  * header as well as each packet header and check that these values are
00050  * valid all the time.
00051  */
00052 #define L4SHMC_RINGBUF_POISONING 1
00053
00059 typedef struct
00060 {
00061     volatile l4_uint32_t lock;
00062     unsigned data_size;
00063     #if L4SHMC_RINGBUF_POISONING
00064         char magic1;
00065     #endif
00066     unsigned next_read;
00067     unsigned next_write;
00068     #if L4SHMC_RINGBUF_POISONING
00069         char magic2;
00070     #endif
00071     unsigned bytes_filled;
00072     unsigned sender_waits;
00073     #if L4SHMC_RINGBUF_POISONING
00074         char magic3;
00075     #endif
00076     char data[];
00077 } l4shmc_ringbuf_head_t;
00078
00079
00085 typedef struct
00086 {
00087     l4shmc_area_t *_area;
00088     l4_cap_idx_t _owner;
00089     l4shmc_chunk_t _chunk;
00090     unsigned _size;
00091     char *_chunkname;
00092     char *_signame;
00093     l4shmc_ringbuf_head_t *_addr;
00094     l4shmc_signal_t _signal_full;
00095     l4shmc_signal_t _signal_empty;
00096 } l4shmc_ringbuf_t;
00097
00098
00105 #define L4SHMC_RINGBUF_HEAD(ringbuf) ((l4shmc_ringbuf_head_t*) ((ringbuf)->_addr))
00106
00107
00114 #define L4SHMC_RINGBUF_DATA(ringbuf) (L4SHMC_RINGBUF_HEAD(ringbuf)->data)
00115

```



```

00116
00123 #define L4SHMC_RINGBUF_DATA_SIZE(ringbuf) ((ringbuf)->_size - sizeof(l4shmc_ringbuf_head_t))
00124
00125 enum lock_content
00126 {
00127     lock_cont_min = 4,
00128     locked        = 5,
00129     unlocked      = 6,
00130     lock_cont_max = 7,
00131 };
00132
00133 static L4_CV inline void l4shmc_rb_lock(l4shmc_ringbuf_head_t *head)
00134 {
00135     ASSERT_NOT_NULL(head);
00136     ASSERT_ASSERT(head->lock > lock_cont_min);
00137     ASSERT_ASSERT(head->lock < lock_cont_max);
00138
00139     while (!l4util_cmpxchg32(&head->lock, unlocked, locked))
00140         l4_thread_yield();
00141 }
00142
00143
00144 static L4_CV inline void l4shmc_rb_unlock(l4shmc_ringbuf_head_t *head)
00145 {
00146     ASSERT_NOT_NULL(head);
00147     ASSERT_ASSERT(head->lock > lock_cont_min);
00148     ASSERT_ASSERT(head->lock < lock_cont_max);
00149
00150     head->lock = unlocked;
00151 }
00152
00153 /*****
00154  * Initialization *
00155  *****/
00156
00173 L4_CV int l4shmc_rb_init_buffer(l4shmc_ringbuf_t *buf, l4shmc_area_t *area,
00174                                char const *chunk_name,
00175                                char const *signal_name, unsigned size);
00176
00182 L4_CV void l4shmc_rb_deinit_buffer(l4shmc_ringbuf_t *buf);
00183
00184
00185
00186 /*****
00187  * RINGBUF SENDER *
00188  *****/
00189
00206 L4_CV int l4shmc_rb_attach_sender(l4shmc_ringbuf_t *buf, char const *signal_name,
00207                                    l4_cap_idx_t owner);
00208
00209
00222 L4_CV char *l4shmc_rb_sender_alloc_packet(l4shmc_ringbuf_head_t *head,
00223                                             unsigned psize);
00224
00225
00236 L4_CV void l4shmc_rb_sender_put_data(l4shmc_ringbuf_t *buf, char *addr,
00237                                       char *data, unsigned dsize);
00238
00239
00254 L4_CV int l4shmc_rb_sender_next_copy_in(l4shmc_ringbuf_t *buf, char *data,
00255                                           unsigned size, int block_if_necessary);
00256
00257
00263 L4_CV void l4shmc_rb_sender_commit_packet(l4shmc_ringbuf_t *buf);
00264
00265
00266 /*****
00267  * RINGBUF RECEIVER *
00268  *****/
00269
00286 L4_CV int l4shmc_rb_init_receiver(l4shmc_ringbuf_t *buf, l4shmc_area_t *area,
00287                                    char const *chunk_name,
00288                                    char const *signal_name);
00289
00290
00303 L4_CV void l4shmc_rb_attach_receiver(l4shmc_ringbuf_t *buf, l4_cap_idx_t owner);
00304
00305
00316 L4_CV int l4shmc_rb_receiver_wait_for_data(l4shmc_ringbuf_t *buf, int blocking);
00317
00318
00328 L4_CV int l4shmc_rb_receiver_copy_out(l4shmc_ringbuf_head_t *head, char *target,
00329                                         unsigned *tsize);
00330
00331
00337 L4_CV void l4shmc_rb_receiver_notify_done(l4shmc_ringbuf_t *buf);
00338

```

```

00339
00346 L4_CV int l4shmc_rb_receiver_read_next_size(l4shmc_ringbuf_head_t *head);
00347
00348 L4_END_DECLS

```

## 17.432 l4/shmc/shmc.h File Reference

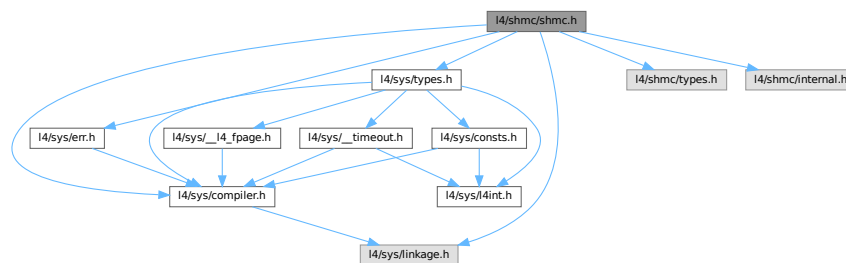
Shared memory library header file.

```

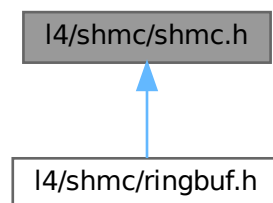
#include <l4/sys/compiler.h>
#include <l4/sys/linkage.h>
#include <l4/sys/types.h>
#include <l4/sys/err.h>
#include <l4/shmc/types.h>
#include <l4/shmc/internal.h>

```

Include dependency graph for shmc.h:



This graph shows which files directly or indirectly include this file:



### Functions

- [L4\\_BEGIN\\_DECLS](#) long [l4shmc\\_create](#) (char const \*shmc\_name)  
*Create a shared memory area.*
- long [l4shmc\\_attach](#) (char const \*shmc\_name, l4shmc\_area\_t \*shmarea)  
*Attach to a shared memory area.*
- long [l4shmc\\_get\\_client\\_nr](#) (l4shmc\_area\_t const \*shmarea)

- Determine the client number of the shared memory region.*

  - long [l4shmc\\_mark\\_client\\_initialized](#) (l4shmc\_area\_t \*shmarea)

*Mark this shared memory client as 'initialized'.*
- long [l4shmc\\_get\\_initialized\\_clients](#) (l4shmc\_area\_t \*shmarea, l4\_umword\_t \*bitmask)

*Fetch the `_clients_init_done` bitmask of the shared memory area.*
- long [l4shmc\\_add\\_chunk](#) (l4shmc\_area\_t \*shmarea, char const \*chunk\_name, l4\_umword\_t chunk\_capacity, l4shmc\_chunk\_t \*chunk)

*Add a chunk in the shared memory area.*
- long [l4shmc\\_add\\_signal](#) (l4shmc\_area\_t \*shmarea, char const \*signal\_name, l4shmc\_signal\_t \*signal)

*Add a signal for the shared memory area.*
- long [l4shmc\\_trigger](#) (l4shmc\_signal\_t \*signal)

*Trigger a signal.*
- long [l4shmc\\_chunk\\_try\\_to\\_take](#) (l4shmc\_chunk\_t \*chunk)

*Try to mark chunk busy.*
- long [l4shmc\\_chunk\\_try\\_to\\_take\\_for\\_writing](#) (l4shmc\_chunk\_t \*chunk)

*Try to mark chunk busy writing.*
- long [l4shmc\\_chunk\\_try\\_to\\_take\\_for\\_overwriting](#) (l4shmc\_chunk\_t \*chunk)

*Try to mark the chunk busy writing after it was ready for reading.*
- long [l4shmc\\_chunk\\_try\\_to\\_take\\_for\\_reading](#) (l4shmc\_chunk\_t \*chunk)

*Try to mark chunk busy reading.*
- long [l4shmc\\_chunk\\_ready](#) (l4shmc\_chunk\_t \*chunk, l4\_umword\_t size)

*Mark chunk as filled (ready).*
- long [l4shmc\\_chunk\\_ready\\_sig](#) (l4shmc\_chunk\_t \*chunk, l4\_umword\_t size)

*Mark chunk as filled (ready) and signal consumer.*
- long [l4shmc\\_get\\_chunk](#) (l4shmc\_area\_t \*shmarea, char const \*chunk\_name, l4shmc\_chunk\_t \*chunk)

*Get chunk out of shared memory area.*
- long [l4shmc\\_get\\_chunk\\_to](#) (l4shmc\_area\_t \*shmarea, char const \*chunk\_name, l4\_umword\_t timeout\_ms, l4shmc\_chunk\_t \*chunk)

*Get chunk out of shared memory area, with timeout.*
- long [l4shmc\\_iterate\\_chunk](#) (l4shmc\_area\_t const \*shmarea, char const \*\*chunk\_name, long offs)

*Iterate over names of all existing chunks.*
- long [l4shmc\\_attach\\_signal](#) (l4shmc\_area\_t \*shmarea, char const \*signal\_name, l4\_cap\_idx\_t thread, l4shmc\_signal\_t \*signal)

*Attach to signal.*
- long [l4shmc\\_get\\_signal](#) (l4shmc\_area\_t \*shmarea, char const \*signal\_name, l4shmc\_signal\_t \*signal)

*Get signal object from the shared memory area.*
- long [l4shmc\\_enable\\_signal](#) (l4shmc\_signal\_t \*signal)

*Enable a signal.*
- long [l4shmc\\_enable\\_chunk](#) (l4shmc\_chunk\_t \*chunk)

*Enable a signal connected with a chunk.*
- long [l4shmc\\_wait\\_any](#) (l4shmc\_signal\_t \*\*retsignal)

*Wait on any signal.*
- long [l4shmc\\_wait\\_any\\_try](#) (l4shmc\_signal\_t \*\*retsignal)

*Check whether any waited signal has an event pending.*
- long [l4shmc\\_wait\\_any\\_to](#) (l4\_timeout\_t timeout, l4shmc\_signal\_t \*\*retsignal)

*Wait for any signal with timeout.*
- long [l4shmc\\_wait\\_signal](#) (l4shmc\_signal\_t \*signal)

*Wait on a specific signal.*
- long [l4shmc\\_wait\\_signal\\_to](#) (l4shmc\_signal\_t \*signal, l4\_timeout\_t timeout)

*Wait on a specific signal, with timeout.*
- long [l4shmc\\_wait\\_signal\\_try](#) (l4shmc\_signal\_t \*signal)

- Check whether a specific signal has an event pending.*

  - long [l4shmc\\_wait\\_chunk](#) (l4shmc\_chunk\_t \*chunk)

*Wait on a specific chunk.*
- long [l4shmc\\_wait\\_chunk\\_to](#) (l4shmc\_chunk\_t \*chunk, [l4\\_timeout\\_t](#) timeout)

*Check whether a specific chunk has an event pending, with timeout.*
- long [l4shmc\\_wait\\_chunk\\_try](#) (l4shmc\_chunk\_t \*chunk)

*Check whether a specific chunk has an event pending.*
- long [l4shmc\\_chunk\\_consumed](#) (l4shmc\_chunk\_t \*chunk)

*Mark a chunk as free.*
- long [l4shmc\\_connect\\_chunk\\_signal](#) (l4shmc\_chunk\_t \*chunk, l4shmc\_signal\_t \*signal)

*Connect a signal with a chunk.*
- long [l4shmc\\_is\\_chunk\\_ready](#) (l4shmc\_chunk\_t const \*chunk)

*Check whether data is available.*
- long [l4shmc\\_is\\_chunk\\_clear](#) (l4shmc\_chunk\_t const \*chunk)

*Check whether chunk is free.*
- void \* [l4shmc\\_chunk\\_ptr](#) (l4shmc\_chunk\_t const \*chunk)

*Get data pointer to chunk.*
- long [l4shmc\\_chunk\\_size](#) (l4shmc\_chunk\_t const \*chunk)

*Get current size of a chunk.*
- long [l4shmc\\_chunk\\_capacity](#) (l4shmc\_chunk\_t const \*chunk)

*Get capacity of a chunk.*
- l4shmc\_signal\_t \* [l4shmc\\_chunk\\_signal](#) (l4shmc\_chunk\_t const \*chunk)

*Get the registered signal of a chunk.*
- [l4\\_cap\\_idx\\_t](#) [l4shmc\\_signal\\_cap](#) (l4shmc\_signal\_t const \*signal)

*Get the signal capability of a signal.*
- long [l4shmc\\_check\\_magic](#) (l4shmc\_chunk\_t const \*chunk)

*Check magic value of a chunk.*
- long [l4shmc\\_area\\_size](#) (l4shmc\_area\_t const \*shmarea)

*Get size of shared memory area.*
- long [l4shmc\\_area\\_size\\_free](#) (l4shmc\_area\_t const \*shmarea)

*Get free size of shared memory area.*
- long [l4shmc\\_area\\_overhead](#) (void)

*Get memory overhead per area that is not available for chunks.*
- long [l4shmc\\_chunk\\_overhead](#) (void)

*Get memory overhead required in addition to the chunk capacity for adding one chunk.*

### 17.432.1 Detailed Description

Shared memory library header file.

Definition in file [shmc.h](#).

## 17.433 shmc.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Björn Döbel <doebel@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU Lesser General Public License 2.1.
00011  * Please see the COPYING-LGPL-2.1 file for details.
00012  */
00013 #pragma once
00014
00015 #include <l4/sys/compiler.h>
00016 #include <l4/sys/linkage.h>
00017 #include <l4/sys/types.h>
00018 #include <l4/sys/err.h>
00019
00069
00070 #define __INCLUDED_FROM_L4SHMC_H__
00071 #include <l4/shmc/types.h>
00072
00073 L4_BEGIN_DECLS
00074
00087 L4_CV long
00088 l4shmc_create(char const *shmc_name);
00089
00111 L4_CV long
00112 l4shmc_attach(char const *shmc_name, l4shmc_area_t *shmarea);
00113
00122 L4_CV long
00123 l4shmc_get_client_nr(l4shmc_area_t const *shmarea);
00124
00137 L4_CV long
00138 l4shmc_mark_client_initialized(l4shmc_area_t *shmarea);
00139
00152 L4_CV long
00153 l4shmc_get_initialized_clients(l4shmc_area_t *shmarea, l4_umword_t *bitmask);
00154
00167 L4_CV long
00168 l4shmc_add_chunk(l4shmc_area_t *shmarea, char const *chunk_name,
00169                 l4_umword_t chunk_capacity, l4shmc_chunk_t *chunk);
00170
00182 L4_CV long
00183 l4shmc_add_signal(l4shmc_area_t *shmarea, char const *signal_name,
00184                  l4shmc_signal_t *signal);
00185
00195 L4_CV L4_INLINE long
00196 l4shmc_trigger(l4shmc_signal_t *signal);
00197
00207 L4_CV L4_INLINE long
00208 l4shmc_chunk_try_to_take(l4shmc_chunk_t *chunk);
00209
00221 L4_CV L4_INLINE long
00222 l4shmc_chunk_try_to_take_for_writing(l4shmc_chunk_t *chunk);
00223
00238 L4_CV L4_INLINE long
00239 l4shmc_chunk_try_to_take_for_overwriting(l4shmc_chunk_t *chunk);
00240
00250 L4_CV L4_INLINE long
00251 l4shmc_chunk_try_to_take_for_reading(l4shmc_chunk_t *chunk);
00252
00263 L4_CV L4_INLINE long
00264 l4shmc_chunk_ready(l4shmc_chunk_t *chunk, l4_umword_t size);
00265
00276 L4_CV L4_INLINE long
00277 l4shmc_chunk_ready_sig(l4shmc_chunk_t *chunk, l4_umword_t size);
00278
00290 L4_CV L4_INLINE long
00291 l4shmc_get_chunk(l4shmc_area_t *shmarea, char const *chunk_name,
00292                  l4shmc_chunk_t *chunk);
00293
00307 L4_CV long
00308 l4shmc_get_chunk_to(l4shmc_area_t *shmarea, char const *chunk_name,
00309                     l4_umword_t timeout_ms, l4shmc_chunk_t *chunk);
00310
00324 L4_CV long
00325 l4shmc_iterate_chunk(l4shmc_area_t const *shmarea, char const **chunk_name,
00326                      long offs);
00327
00340 L4_CV long
00341 l4shmc_attach_signal(l4shmc_area_t *shmarea, char const *signal_name,
00342                      l4_cap_idx_t thread, l4shmc_signal_t *signal);
00343

```

```

00344
00356 L4_CV long
00357 l4shmc_get_signal(l4shmc_area_t *shmarea, char const *signal_name,
00358                   l4shmc_signal_t *signal);
00359
00373 L4_CV long
00374 l4shmc_enable_signal(l4shmc_signal_t *signal);
00375
00389 L4_CV long
00390 l4shmc_enable_chunk(l4shmc_chunk_t *chunk);
00391
00401 L4_CV L4_INLINE long
00402 l4shmc_wait_any(l4shmc_signal_t **retsignal);
00403
00417 L4_CV L4_INLINE long
00418 l4shmc_wait_any_try(l4shmc_signal_t **retsignal);
00419
00434 L4_CV long
00435 l4shmc_wait_any_to(l4_timeout_t timeout, l4shmc_signal_t **retsignal);
00436
00446 L4_CV L4_INLINE long
00447 l4shmc_wait_signal(l4shmc_signal_t *signal);
00448
00459 L4_CV long
00460 l4shmc_wait_signal_to(l4shmc_signal_t *signal, l4_timeout_t timeout);
00461
00475 L4_CV L4_INLINE long
00476 l4shmc_wait_signal_try(l4shmc_signal_t *signal);
00477
00487 L4_CV L4_INLINE long
00488 l4shmc_wait_chunk(l4shmc_chunk_t *chunk);
00489
00504 L4_CV long
00505 l4shmc_wait_chunk_to(l4shmc_chunk_t *chunk, l4_timeout_t timeout);
00506
00520 L4_CV L4_INLINE long
00521 l4shmc_wait_chunk_try(l4shmc_chunk_t *chunk);
00522
00532 L4_CV L4_INLINE long
00533 l4shmc_chunk_consumed(l4shmc_chunk_t *chunk);
00534
00545 L4_CV long
00546 l4shmc_connect_chunk_signal(l4shmc_chunk_t *chunk, l4shmc_signal_t *signal);
00547
00557 L4_CV L4_INLINE long
00558 l4shmc_is_chunk_ready(l4shmc_chunk_t const *chunk);
00559
00569 L4_CV L4_INLINE long
00570 l4shmc_is_chunk_clear(l4shmc_chunk_t const *chunk);
00571
00580 L4_CV L4_INLINE void *
00581 l4shmc_chunk_ptr(l4shmc_chunk_t const *chunk);
00582
00591 L4_CV L4_INLINE long
00592 l4shmc_chunk_size(l4shmc_chunk_t const *chunk);
00593
00602 L4_CV L4_INLINE long
00603 l4shmc_chunk_capacity(l4shmc_chunk_t const *chunk);
00604
00614 L4_CV L4_INLINE l4shmc_signal_t *
00615 l4shmc_chunk_signal(l4shmc_chunk_t const *chunk);
00616
00625 L4_CV L4_INLINE l4_cap_idx_t
00626 l4shmc_signal_cap(l4shmc_signal_t const *signal);
00627
00637 L4_CV L4_INLINE long
00638 l4shmc_check_magic(l4shmc_chunk_t const *chunk);
00639
00649 L4_CV long
00650 l4shmc_area_size(l4shmc_area_t const *shmarea);
00651
00661 L4_CV long
00662 l4shmc_area_size_free(l4shmc_area_t const *shmarea);
00663
00670 L4_CV long
00671 l4shmc_area_overhead(void);
00672
00680 L4_CV long
00681 l4shmc_chunk_overhead(void);
00682
00683 #include <l4/shmc/internal.h>
00684
00685 L4_END_DECLS

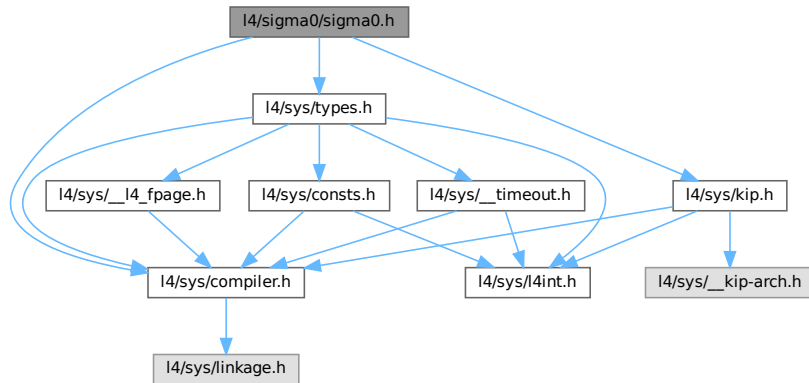
```

## 17.434 l4/sigma0/sigma0.h File Reference

Sigma0 interface.

```
#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <l4/sys/kip.h>
```

Include dependency graph for sigma0.h:



### Macros

- **#define SIGMA0\_REQ\_MAGIC** ~0xFFUL  
*Request magic.*
- **#define SIGMA0\_REQ\_MASK** ~0xFFUL  
*Request mask.*
- **#define SIGMA0\_REQ\_ID\_MASK** 0xF0  
*ID mask.*
- **#define SIGMA0\_REQ\_ID\_FPAGE\_RAM** 0x60  
*RAM.*
- **#define SIGMA0\_REQ\_ID\_FPAGE\_IOMEM** 0x70  
*I/O memory.*
- **#define SIGMA0\_REQ\_ID\_FPAGE\_IOMEM\_CACHED** 0x80  
*Cached I/O memory.*
- **#define SIGMA0\_REQ\_ID\_FPAGE\_ANY** 0x90  
*Any.*
- **#define SIGMA0\_REQ\_ID\_KIP** 0xA0  
*KIP.*
- **#define SIGMA0\_REQ\_ID\_DEBUG\_DUMP** 0xC0  
*Debug dump.*
- **#define SIGMA0\_IS\_MAGIC\_REQ**(d1)  
*Check if magic.*
- **#define SIGMA0\_REQ**(x)  
*Construct.*
- **#define SIGMA0\_REQ\_FPAGE\_RAM** (SIGMA0\_REQ(FPAGE\_RAM))  
*RAM.*

- #define **SIGMA0\_REQ\_FPAGE\_IOMEM** ([SIGMA0\\_REQ](#)(FPAGE\_IOMEM))  
*I/O memory.*
- #define **SIGMA0\_REQ\_FPAGE\_IOMEM\_CACHED** ([SIGMA0\\_REQ](#)(FPAGE\_IOMEM\_CACHED))  
*Cache I/O memory.*
- #define **SIGMA0\_REQ\_FPAGE\_ANY** ([SIGMA0\\_REQ](#)(FPAGE\_ANY))  
*Any.*
- #define **SIGMA0\_REQ\_KIP** ([SIGMA0\\_REQ](#)(KIP))  
*KIP.*
- #define **SIGMA0\_REQ\_DEBUG\_DUMP** ([SIGMA0\\_REQ](#)(DEBUG\_DUMP))  
*Debug dump.*

## Enumerations

- enum [l4sigma0\\_return\\_flags\\_t](#) {  
    [L4SIGMA0\\_OK](#) , [L4SIGMA0\\_NOTALIGNED](#) , [L4SIGMA0\\_IPCERROR](#) , [L4SIGMA0\\_NOFPAGE](#) ,  
    [L4SIGMA0\\_4](#) , [L4SIGMA0\\_5](#) , [L4SIGMA0\\_SMALLERFPAGE](#) }  
*Return flags of libsigma0 functions.*

## Functions

- [L4\\_BEGIN\\_DECLS](#) [l4\\_kernel\\_info\\_t](#) \* [l4sigma0\\_map\\_kip](#) ([l4\\_cap\\_idx\\_t](#) sigma0, void \*addr, unsigned log2↵\_size)  
*Map the kernel info page from sigma0 to addr.*
- int [l4sigma0\\_map\\_mem](#) ([l4\\_cap\\_idx\\_t](#) sigma0, [l4\\_addr\\_t](#) phys, [l4\\_addr\\_t](#) virt, [l4\\_addr\\_t](#) size)  
*Request a memory mapping from sigma0.*
- int [l4sigma0\\_map\\_iomem](#) ([l4\\_cap\\_idx\\_t](#) sigma0, [l4\\_addr\\_t](#) phys, [l4\\_addr\\_t](#) virt, [l4\\_addr\\_t](#) size, int cached)  
*Request IO memory from sigma0.*
- int [l4sigma0\\_map\\_anypage](#) ([l4\\_cap\\_idx\\_t](#) sigma0, [l4\\_addr\\_t](#) map\_area, unsigned log2\_map\_size, [l4\\_addr\\_t](#) \*base, unsigned sz)  
*Request an arbitrary free page of RAM.*
- void [l4sigma0\\_debug\\_dump](#) ([l4\\_cap\\_idx\\_t](#) sigma0)  
*Request sigma0 to dump internal debug information.*
- char const \* [l4sigma0\\_map\\_errstr](#) (int err)  
*Get user readable error messages for the return codes.*

### 17.434.1 Detailed Description

Sigma0 interface.

Definition in file [sigma0.h](#).



## 17.435 sigma0.h

[Go to the documentation of this file.](#)

```

00001
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00010  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00011  *      economic rights: Technische Universität Dresden (Germany)
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #ifndef __L4_SIGMA0_SIGMA0_H
00015 #define __L4_SIGMA0_SIGMA0_H
00016
00024
00025 #include <l4/sys/compiler.h>
00026 #include <l4/sys/types.h>
00027 #include <l4/sys/kip.h>
00028
00035 #undef SIGMA0_REQ_MAGIC
00036 #undef SIGMA0_REQ_MASK
00037
00038 # define SIGMA0_REQ_MAGIC    ~0xFFUL
00039 # define SIGMA0_REQ_MASK    ~0xFFUL
00040
00041 /* Starting with 0x60 allows to detect components which still use the old
00042  * constants (0x00 ... 0x50) */
00043 #define SIGMA0_REQ_ID_MASK    0xF0
00044 #define SIGMA0_REQ_ID_FPAGE_RAM    0x60
00045 #define SIGMA0_REQ_ID_FPAGE_IOMEM    0x70
00046 #define SIGMA0_REQ_ID_FPAGE_IOMEM_CACHED    0x80
00047 #define SIGMA0_REQ_ID_FPAGE_ANY    0x90
00048 #define SIGMA0_REQ_ID_KIP    0xA0
00049 #define SIGMA0_REQ_ID_DEBUG_DUMP    0xC0
00050
00051 #define SIGMA0_IS_MAGIC_REQ(d1) \
00052     ((d1 & SIGMA0_REQ_MASK) == SIGMA0_REQ_MAGIC)
00053
00054 #define SIGMA0_REQ(x) \
00055     (SIGMA0_REQ_MAGIC + SIGMA0_REQ_ID_ ## x)
00056
00057 /* Use these constants in your code! */
00058 #define SIGMA0_REQ_FPAGE_RAM    (SIGMA0_REQ(FPAGE_RAM))
00059 #define SIGMA0_REQ_FPAGE_IOMEM    (SIGMA0_REQ(FPAGE_IOMEM))
00060 #define SIGMA0_REQ_FPAGE_IOMEM_CACHED    (SIGMA0_REQ(FPAGE_IOMEM_CACHED))
00061 #define SIGMA0_REQ_FPAGE_ANY    (SIGMA0_REQ(FPAGE_ANY))
00062 #define SIGMA0_REQ_KIP    (SIGMA0_REQ(KIP))
00063 #define SIGMA0_REQ_DEBUG_DUMP    (SIGMA0_REQ(DEBUG_DUMP))
00064
00065
00070
00074 enum l4sigma0_return_flags_t {
00075     L4SIGMA0_OK,
00076     L4SIGMA0_NOTALIGNED,
00077     L4SIGMA0_IPCERROR,
00078     L4SIGMA0_NOFPAGE,
00079     L4SIGMA0_4,
00080     L4SIGMA0_5,
00081     L4SIGMA0_SMALLERFPAGE,
00082 };
00083
00084 L4_BEGIN_DECLS
00085
00095 L4_CV l4_kernel_info_t *
00096 l4sigma0_map_kip(l4_cap_idx_t sigma0, void *addr, unsigned log2_size);
00097
00127 L4_CV int l4sigma0_map_mem(l4_cap_idx_t sigma0,
00128                             l4_addr_t phys, l4_addr_t virt, l4_addr_t size);
00129
00152 L4_CV int l4sigma0_map_iomem(l4_cap_idx_t sigma0, l4_addr_t phys,
00153                               l4_addr_t virt, l4_addr_t size, int cached);
00178 L4_CV int l4sigma0_map_anypage(l4_cap_idx_t sigma0, l4_addr_t map_area,
00179                                 unsigned log2_map_size, l4_addr_t *base,
00180                                 unsigned sz);
00181
00190 L4_CV void l4sigma0_debug_dump(l4_cap_idx_t sigma0);
00191
00199 L4_INLINE char const *l4sigma0_map_errstr(int err);
00200
00202
00203
00204 /* Implementations */
00205
00206 L4_INLINE char const *l4sigma0_map_errstr(int err)
00207 {
00208     switch (err)

```

```

00209     {
00210         case 0: return "No error";
00211         case -1: return "Phys, virt or size not aligned";
00212         case -2: return "IPC error";
00213         case -3: return "No fpage received";
00214 #ifndef SIGMA0_REQ_MAGIC
00215         case -4: return "Bad physical address (old protocol only)";
00216 #endif
00217         case -6: return "Superpage requested but smaller flexpage received";
00218         case -7: return "Cannot map I/O memory cacheable (old protocol only)";
00219         default: return "Unknown error";
00220     }
00221 }
00222
00223
00224 L4_END_DECLS
00225
00226 #endif /* ! __L4_SIGMA0_SIGMA0_H */

```

## 17.436 \_\_kernel\_object\_impl.h

```

00001
00006 #pragma once
00007
00008 #include <l4/sys/ipc.h>
00009
00010 L4_INLINE l4_msgtag_t
00011 l4_invoke_debugger(l4_cap_idx_t obj, l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW
00012 {
00013     l4_msgtag_t t2;
00014     unsigned const words = l4_msgtag_words(tag);
00015     l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);
00016
00017     if (l4_is_invalid_cap(obj))
00018         return l4_msgtag(-L4_EINVAL, 0, 0, 0);
00019
00020     if (words + 2 > L4_UTCB_GENERIC_DATA_SIZE)
00021         return l4_msgtag(-L4_EMSTOOLONG, 0, 0, 0);
00022
00023     mr->mr[0] += 0x100;
00024     mr->mr[words] = L4_ITEM_MAP;
00025     mr->mr[words + 1] = l4_obj_fpage(obj, 0, L4_CAP_FPAGE_RWS).raw;
00026     t2 = l4_msgtag(L4_PROTO_DEBUGGER, words, 1, l4_msgtag_flags(tag));
00027
00028     return l4_ipc_call(L4_BASE_DEBUGGER_CAP, utcb, t2, L4_IPC_NEVER);
00029 }
00030

```

## 17.437 l4/sys/\_\_ktrace-impl.h File Reference

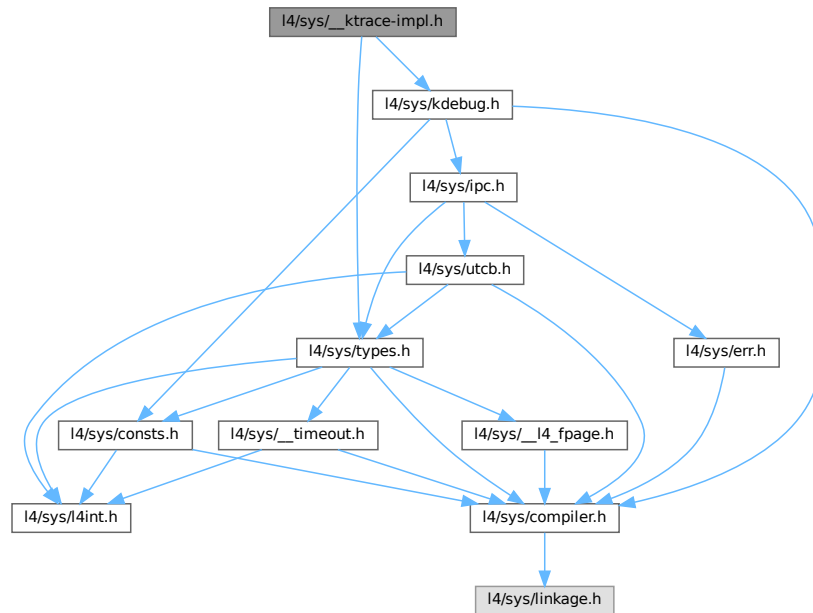
[L4](#) kernel event tracing.

```

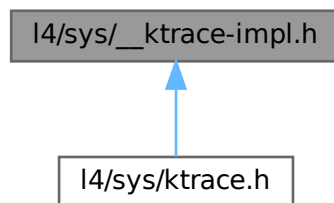
#include <l4/sys/types.h>
#include <l4/sys/kdebug.h>

```

Include dependency graph for \_\_\_ktrace-impl.h:



This graph shows which files directly or indirectly include this file:



## Functions

- [l4\\_umword\\_t fiasco\\_tbuf\\_log](#) (const char \*text)  
Create new trace-buffer entry with describing <text>.
- [l4\\_umword\\_t fiasco\\_tbuf\\_log\\_3val](#) (const char \*text, [l4\\_umword\\_t](#) v1, [l4\\_umword\\_t](#) v2, [l4\\_umword\\_t](#) v3)  
Create new trace-buffer entry with describing <text> and three additional values.
- void **fiasco\_tbuf\_clear** (void)  
Clear trace-buffer.
- void **fiasco\_tbuf\_dump** (void)  
Dump trace-buffer to kernel console.
- [l4\\_umword\\_t fiasco\\_tbuf\\_log\\_binary](#) (const unsigned char \*data)  
Create new trace-buffer entry with binary data.

## 17.437.1 Detailed Description

[L4](#) kernel event tracing.

Definition in file [\\_\\_ktrace-impl.h](#).

## 17.438 \_\_ktrace-impl.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *          Björn Döbel <doebel@os.inf.tu-dresden.de>,
00009  *          Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *          economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #pragma once
00015
00016 #include <l4/sys/types.h>
00017 #include <l4/sys/kdebug.h>
00018
00019 /*****
00020  *** Implementation
00021  *****/
00022
00023 L4_INLINE l4_umword_t
00024 fiasco_tbuf_log(const char *text)
00025 {
00026     enum { TBUF_LOG = L4_KDEBUG_GROUP_TRACE + 0x01 };
00027     return l4_error(__kdebug_text(TBUF_LOG, text, __builtin_strlen(text)));
00028 }
00029
00030 L4_INLINE l4_umword_t
00031 fiasco_tbuf_log_3val(const char *text, l4_umword_t v1, l4_umword_t v2,
00032                     l4_umword_t v3)
00033 {
00034     enum { TBUF_LOG_3VAL = L4_KDEBUG_GROUP_TRACE + 0x04 };
00035     return l4_error(__kdebug_3_text(TBUF_LOG_3VAL, text,
00036                                     __builtin_strlen(text), v1, v2, v3));
00037 }
00038
00039 L4_INLINE void
00040 fiasco_tbuf_clear(void)
00041 {
00042     enum { TBUF_CLEAR = L4_KDEBUG_GROUP_TRACE + 0x02 };
00043     __kdebug_op(TBUF_CLEAR);
00044 }
00045
00046 L4_INLINE void
00047 fiasco_tbuf_dump(void)
00048 {
00049     enum { TBUF_DUMP = L4_KDEBUG_GROUP_TRACE + 0x03 };
00050     __kdebug_op(TBUF_DUMP);
00051 }
00052
00053 L4_INLINE l4_umword_t
00054 fiasco_tbuf_log_binary(const unsigned char *data)
00055 {
00056     enum { TBUF_LOG_BIN = L4_KDEBUG_GROUP_TRACE + 0x08 };
00057     return l4_error(__kdebug_text(TBUF_LOG_BIN, (const char *)data, 24));
00058 }
00059

```

## 17.439 \_\_l4\_fpage.h

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *          Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *          Björn Döbel <doebel@os.inf.tu-dresden.de>,
00010  *          Torsten Frenzel <frenzel@os.inf.tu-dresden.de>

```

```

00011  *      economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * License: see LICENSE.spdx (in this directory or the directories above)
00014  */
00015 #pragma once
00016
00017 #include <l4/sys/compiler.h>
00018
00043
00048 enum L4_fpage_consts
00049 {
00050     L4_FPAGE_RIGHTS_SHIFT = 0,
00051     L4_FPAGE_TYPE_SHIFT   = 4,
00052     L4_FPAGE_SIZE_SHIFT   = 6,
00053     L4_FPAGE_ADDR_SHIFT   = 12,
00054
00055     L4_FPAGE_RIGHTS_BITS  = 4,
00056     L4_FPAGE_TYPE_BITS    = 2,
00057     L4_FPAGE_SIZE_BITS    = 6,
00058     L4_FPAGE_ADDR_BITS    = L4_MWORD_BITS - L4_FPAGE_ADDR_SHIFT,
00059
00061     L4_FPAGE_RIGHTS_MASK  = ((1UL < L4_FPAGE_RIGHTS_BITS) - 1)
00062                             < L4_FPAGE_RIGHTS_SHIFT,
00063     L4_FPAGE_TYPE_MASK    = ((1UL < L4_FPAGE_TYPE_BITS) - 1)
00064                             < L4_FPAGE_TYPE_SHIFT,
00065     L4_FPAGE_SIZE_MASK    = ((1UL < L4_FPAGE_SIZE_BITS) - 1)
00066                             < L4_FPAGE_SIZE_SHIFT,
00067     L4_FPAGE_ADDR_MASK    = ~0UL < L4_FPAGE_ADDR_SHIFT,
00069     L4_FPAGE_RIGHTS_ALL   = L4_FPAGE_RIGHTS_MASK,
00070 };
00071
00076 typedef union {
00077     l4_umword_t fpage;
00078     l4_umword_t raw;
00079 } l4_fpage_t;
00080
00084 enum
00085 {
00092     L4_WHOLE_ADDRESS_SPACE = 63
00093 };
00094
00099 typedef struct {
00100     l4_umword_t snd_base;
00101     l4_fpage_t fpage;
00102 } l4_snd_fpage_t;
00103
00104
00118 enum L4_fpage_rights
00119 {
00120     L4_FPAGE_X      = 1,
00121     L4_FPAGE_W      = 2,
00122     L4_FPAGE_RO     = 4,
00123     L4_FPAGE_RW     = L4_FPAGE_RO | L4_FPAGE_W,
00124     L4_FPAGE_RX     = L4_FPAGE_RO | L4_FPAGE_X,
00125     L4_FPAGE_RWX    = L4_FPAGE_RW | L4_FPAGE_X,
00126 };
00127
00148 enum L4_cap_fpage_rights
00149 {
00157     L4_CAP_FPAGE_W      = 0x1,
00169     L4_CAP_FPAGE_S      = 0x2,
00175     L4_CAP_FPAGE_R      = 0x4,
00176     L4_CAP_FPAGE_RO     = 0x4,
00185     L4_CAP_FPAGE_D      = 0x8,
00192     L4_CAP_FPAGE_RW     = L4_CAP_FPAGE_R | L4_CAP_FPAGE_W,
00199     L4_CAP_FPAGE_RS     = L4_CAP_FPAGE_R | L4_CAP_FPAGE_S,
00206     L4_CAP_FPAGE_RWS    = L4_CAP_FPAGE_RW | L4_CAP_FPAGE_S,
00212     L4_CAP_FPAGE_RWSD   = L4_CAP_FPAGE_RWS | L4_CAP_FPAGE_D,
00218     L4_CAP_FPAGE_RWD    = L4_CAP_FPAGE_RW | L4_CAP_FPAGE_D,
00224     L4_CAP_FPAGE_RSD    = L4_CAP_FPAGE_RS | L4_CAP_FPAGE_D,
00225 };
00226
00230 enum L4_fpage_type
00231 {
00232     L4_FPAGE_SPECIAL = 0,
00235     L4_FPAGE_MEMORY  = 1,
00236     L4_FPAGE_IO      = 2,
00237     L4_FPAGE_OBJ     = 3,
00238 };
00239
00243 enum L4_fpage_control
00244 {
00247     L4_FPAGE_CONTROL_OFFSET_SHIFT = 12,
00250     L4_FPAGE_CONTROL_MASK = ~0UL < L4_FPAGE_CONTROL_OFFSET_SHIFT,
00251 };
00252
00262 enum L4_obj_fpage_ctl

```

```

00263 {
00264     L4_FPAGE_C_REF_CNT      = 0x00,
00265     L4_FPAGE_C_NO_REF_CNT  = 0x10,
00266
00267     L4_FPAGE_C_OBJ_RIGHT1  = 0x20,
00268     L4_FPAGE_C_OBJ_RIGHT2  = 0x40,
00269     L4_FPAGE_C_OBJ_RIGHT3  = 0x80,
00270     L4_FPAGE_C_OBJ_RIGHTS  = 0xe0,
00271
00277     L4_FPAGE_C_IPCGATE_SVR = L4_FPAGE_C_OBJ_RIGHT1
00278 };
00279
00280
00291 enum l4_fpage_cacheability_opt_t
00292 {
00295     L4_FPAGE_CACHE_OPT      = 0x1,
00296
00299     L4_FPAGE_CACHEABLE      = 0x3,
00300
00303     L4_FPAGE_BUFFERABLE     = 0x5,
00304
00307     L4_FPAGE_UNCACHEABLE    = 0x1
00308 };
00309
00310
00314 enum
00315 {
00319     L4_WHOLE_IOADDRESS_SPACE = 16,
00320
00322     L4_IOPORT_MAX             = (1L << L4_WHOLE_IOADDRESS_SPACE)
00323 };
00324
00325
00326
00341 L4_INLINE l4_fpage_t
00342 l4_fpage(l4_addr_t address, unsigned int order, unsigned char rights) L4_NOTHROW;
00343
00355 L4_INLINE l4_fpage_t
00356 l4_fpage_all(void) L4_NOTHROW;
00357
00364 L4_INLINE l4_fpage_t
00365 l4_fpage_invalid(void) L4_NOTHROW;
00366
00367
00378 L4_INLINE l4_fpage_t
00379 l4_iofpage(unsigned long port, unsigned int order) L4_NOTHROW;
00380
00381
00394 L4_INLINE l4_fpage_t
00395 l4_obj_fpage(l4_cap_idx_t obj, unsigned int order, unsigned char rights) L4_NOTHROW;
00396
00406 L4_INLINE int
00407 l4_is_fpage_writable(l4_fpage_t fp) L4_NOTHROW;
00408
00409
00440
00456 L4_INLINE l4_umword_t
00457 l4_map_control(l4_umword_t spot, unsigned char cache, unsigned grant) L4_NOTHROW;
00458
00472 L4_INLINE l4_umword_t
00473 l4_map_obj_control(l4_umword_t spot, unsigned grant) L4_NOTHROW;
00474
00483 L4_INLINE unsigned
00484 l4_fpage_rights(l4_fpage_t f) L4_NOTHROW;
00485
00494 L4_INLINE unsigned
00495 l4_fpage_type(l4_fpage_t f) L4_NOTHROW;
00496
00507 L4_INLINE unsigned
00508 l4_fpage_size(l4_fpage_t f) L4_NOTHROW;
00509
00520 L4_INLINE unsigned long
00521 l4_fpage_page(l4_fpage_t f) L4_NOTHROW;
00522
00536 L4_INLINE l4_addr_t
00537 l4_fpage_memaddr(l4_fpage_t f) L4_NOTHROW;
00538
00552 L4_INLINE l4_cap_idx_t
00553 l4_fpage_obj(l4_fpage_t f) L4_NOTHROW;
00554
00568 L4_INLINE unsigned long
00569 l4_fpage_ioport(l4_fpage_t f) L4_NOTHROW;
00570
00580 L4_INLINE l4_fpage_t
00581 l4_fpage_set_rights(l4_fpage_t src, unsigned char new_rights) L4_NOTHROW;
00582
00594 L4_INLINE int

```

```

00595 l4_fpage_contains(l4_fpage_t fpage, l4_addr_t addr, unsigned order) L4_NOTHROW;
00596
00613 L4_INLINE unsigned char
00614 l4_fpage_max_order(unsigned char order, l4_addr_t addr,
00615                    l4_addr_t min_addr, l4_addr_t max_addr,
00616                    l4_addr_t hotspot L4_DEFAULT_PARAM(0));
00617
00627 L4_INLINE int
00628 l4_is_fpage_valid(l4_fpage_t fp) L4_NOTHROW;
00629
00630 /*****
00631  * Implementations
00632  *****/
00633
00634 L4_INLINE unsigned
00635 l4_fpage_rights(l4_fpage_t f) L4_NOTHROW
00636 {
00637     return (f.raw & L4_FPAGE_RIGHTS_MASK) » L4_FPAGE_RIGHTS_SHIFT;
00638 }
00639
00640 L4_INLINE unsigned
00641 l4_fpage_type(l4_fpage_t f) L4_NOTHROW
00642 {
00643     return (f.raw & L4_FPAGE_TYPE_MASK) » L4_FPAGE_TYPE_SHIFT;
00644 }
00645
00646 L4_INLINE unsigned
00647 l4_fpage_size(l4_fpage_t f) L4_NOTHROW
00648 {
00649     return (f.raw & L4_FPAGE_SIZE_MASK) » L4_FPAGE_SIZE_SHIFT;
00650 }
00651
00652 L4_INLINE unsigned long
00653 l4_fpage_page(l4_fpage_t f) L4_NOTHROW
00654 {
00655     return (f.raw & L4_FPAGE_ADDR_MASK) » L4_FPAGE_ADDR_SHIFT;
00656 }
00657
00658 L4_INLINE unsigned long
00659 l4_fpage_ioport(l4_fpage_t f) L4_NOTHROW
00660 {
00661     return (f.raw & L4_FPAGE_ADDR_MASK) » L4_FPAGE_ADDR_SHIFT;
00662 }
00663
00664 L4_INLINE l4_addr_t
00665 l4_fpage_memaddr(l4_fpage_t f) L4_NOTHROW
00666 {
00667     return f.raw & L4_FPAGE_ADDR_MASK;
00668 }
00669
00670 L4_INLINE l4_cap_idx_t
00671 l4_fpage_obj(l4_fpage_t f) L4_NOTHROW
00672 {
00673     return f.raw & L4_FPAGE_ADDR_MASK;
00674 }
00675
00677 L4_INLINE l4_fpage_t
00678 __l4_fpage_generic(unsigned long address, unsigned int type,
00679                   unsigned int order, unsigned char rights) L4_NOTHROW;
00680
00681 L4_INLINE l4_fpage_t
00682 __l4_fpage_generic(unsigned long address, unsigned int type,
00683                   unsigned int order, unsigned char rights) L4_NOTHROW
00684 {
00685     l4_fpage_t t;
00686     t.raw = ((rights « L4_FPAGE_RIGHTS_SHIFT) & L4_FPAGE_RIGHTS_MASK)
00687           | ((type « L4_FPAGE_TYPE_SHIFT) & L4_FPAGE_TYPE_MASK)
00688           | ((order « L4_FPAGE_SIZE_SHIFT) & L4_FPAGE_SIZE_MASK)
00689           | ((address & L4_FPAGE_ADDR_MASK) & L4_FPAGE_ADDR_MASK);
00690     return t;
00691 }
00692
00693 L4_INLINE l4_fpage_t
00694 l4_fpage_set_rights(l4_fpage_t src, unsigned char new_rights) L4_NOTHROW
00695 {
00696     l4_fpage_t f;
00697     f.raw = ((L4_FPAGE_TYPE_MASK | L4_FPAGE_SIZE_MASK | L4_FPAGE_ADDR_MASK) & src.raw)
00698           | ((new_rights « L4_FPAGE_RIGHTS_SHIFT) & L4_FPAGE_RIGHTS_MASK);
00699     return f;
00700 }
00701
00702 L4_INLINE l4_fpage_t
00703 l4_fpage(l4_addr_t address, unsigned int order, unsigned char rights) L4_NOTHROW
00704 {
00705     return __l4_fpage_generic(address, L4_FPAGE_MEMORY, order, rights);
00706 }
00707

```

```

00708 L4_INLINE l4_fpage_t
00709 l4_iofpage(unsigned long port, unsigned int order) L4_NOTHROW
00710 {
00711     return __l4_fpage_generic(port << L4_FPAGE_ADDR_SHIFT, L4_FPAGE_IO, order, L4_FPAGE_RW);
00712 }
00713
00714 L4_INLINE l4_fpage_t
00715 l4_obj_fpage(l4_cap_idx_t obj, unsigned int order, unsigned char rights) L4_NOTHROW
00716 {
00717     static_assert((unsigned long)L4_CAP_SHIFT >= L4_FPAGE_ADDR_SHIFT,
00718         "Capability index does not fit into fpage.");
00719     return __l4_fpage_generic(obj, L4_FPAGE_OBJ, order, rights);
00720 }
00721
00722 L4_INLINE l4_fpage_t
00723 l4_fpage_all(void) L4_NOTHROW
00724 {
00725     return __l4_fpage_generic(0, L4_FPAGE_SPECIAL, L4_WHOLE_ADDRESS_SPACE, 0);
00726 }
00727
00728 L4_INLINE l4_fpage_t
00729 l4_fpage_invalid(void) L4_NOTHROW
00730 {
00731     return __l4_fpage_generic(0, L4_FPAGE_SPECIAL, 0, 0);
00732 }
00733
00734
00735 L4_INLINE int
00736 l4_is_fpage_writable(l4_fpage_t fp) L4_NOTHROW
00737 {
00738     return l4_fpage_rights(fp) & L4_FPAGE_W;
00739 }
00740
00741 L4_INLINE l4_umword_t
00742 l4_map_control(l4_umword_t snd_base, unsigned char cache, unsigned grant) L4_NOTHROW
00743 {
00744     return (snd_base & L4_FPAGE_CONTROL_MASK)
00745         | ((l4_umword_t)cache << 4) | L4_ITEM_MAP | grant;
00746 }
00747
00748 L4_INLINE l4_umword_t
00749 l4_map_obj_control(l4_umword_t snd_base, unsigned grant) L4_NOTHROW
00750 {
00751     return l4_map_control(snd_base, 0, grant);
00752 }
00753
00754 L4_INLINE int
00755 l4_fpage_contains(l4_fpage_t fpage, l4_addr_t addr, unsigned log2size) L4_NOTHROW
00756 {
00757     l4_addr_t fa = l4_fpage_memaddr(fpage);
00758     return (fa <= addr)
00759         && (fa + (1UL << l4_fpage_size(fpage)) >= addr + (1UL << log2size));
00760 }
00761
00762 L4_INLINE unsigned char
00763 l4_fpage_max_order(unsigned char order, l4_addr_t addr,
00764     l4_addr_t min_addr, l4_addr_t max_addr,
00765     l4_addr_t hotspot)
00766 {
00767     while (order < 30 /* limit to 1GB flexpages */)
00768     {
00769         l4_addr_t mask;
00770         l4_addr_t base = l4_trunc_size(addr, order + 1);
00771         if (base < min_addr)
00772             return order;
00773
00774         if (base + (1UL << (order + 1)) - 1 > max_addr - 1)
00775             return order;
00776
00777         mask = ~(~0UL << (order + 1));
00778         if (hotspot == ~0UL || ((addr ^ hotspot) & mask))
00779             break;
00780
00781         ++order;
00782     }
00783
00784     return order;
00785 }
00786
00787 L4_INLINE int
00788 l4_is_fpage_valid(l4_fpage_t fp) L4_NOTHROW
00789 {
00790     return l4_fpage_type(fp) != L4_FPAGE_SPECIAL || l4_fpage_size(fp) != 0;
00791 }

```



## 17.440 \_\_platform\_control-arm.h

```

00001 /*
00002  * Copyright (C) 2024 Kernkonzept GmbH.
00003  * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/sys/types.h>
00010
00011
00029 L4_INLINE l4_msgtag_t
00030 l4_platform_ctl_set_task_asid(l4_cap_idx_t pfc,
00031                               l4_cap_idx_t task,
00032                               l4_umword_t asid) L4_NOTHROW;
00033
00034
00038 L4_INLINE l4_msgtag_t
00039 l4_platform_ctl_set_task_asid_u(l4_cap_idx_t pfc,
00040                                 l4_cap_idx_t task,
00041                                 l4_umword_t asid,
00042                                 l4_utcb_t *utcb) L4_NOTHROW;
00043
00044 /* IMPLEMENTATION -----*/
00045
00046 L4_INLINE l4_msgtag_t
00047 l4_platform_ctl_set_task_asid_u(l4_cap_idx_t pfc,
00048                                 l4_cap_idx_t task,
00049                                 l4_umword_t asid,
00050                                 l4_utcb_t *utcb) L4_NOTHROW
00051 {
00052     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00053     v->mr[0] = L4_PLATFORM_CTL_SET_TASK_ASID_OP;
00054     v->mr[1] = asid;
00055     v->mr[2] = l4_map_obj_control(0, 0);
00056     v->mr[3] = l4_obj_fpage(task, 0, L4_CAP_FPAGE_RWS).raw;
00057     return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 2, 1, 0),
00058                       L4_IPC_NEVER);
00059 }
00060
00061 L4_INLINE l4_msgtag_t
00062 l4_platform_ctl_set_task_asid(l4_cap_idx_t pfc,
00063                               l4_cap_idx_t task,
00064                               l4_umword_t asid) L4_NOTHROW
00065 {
00066     return l4_platform_ctl_set_task_asid_u(pfc, task, asid, l4_utcb());
00067 }

```

## 17.441 \_\_task-arm.h

```

00001 /*
00002  * (c) 2018 Adam Lackorzynski <adam@l4re.org>
00003  *
00004  * License: see LICENSE.spdx (in this directory or the directories above)
00005  */
00006 #pragma once
00007
00008 #include <l4/sys/types.h>
00009
00013 L4_INLINE l4_msgtag_t
00014 l4_task_vgicc_map_u(l4_cap_idx_t task, l4_fpage_t vgicc_fpage,
00015                    l4_utcb_t *u) L4_NOTHROW;
00016
00028 L4_INLINE l4_msgtag_t
00029 l4_task_vgicc_map(l4_cap_idx_t task, l4_fpage_t vgicc_fpage) L4_NOTHROW;
00030
00031 /* IMPLEMENTATION -----*/
00032
00033 #include <l4/sys/ipc.h>
00034
00035 L4_INLINE l4_msgtag_t
00036 l4_task_vgicc_map_u(l4_cap_idx_t task, l4_fpage_t vgicc_fpage,
00037                    l4_utcb_t *u) L4_NOTHROW
00038 {
00039     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00040     v->mr[0] = L4_TASK_MAP_VGICC_ARM_OP;
00041     v->mr[1] = vgicc_fpage.raw;
00042     return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2, 0, 0), L4_IPC_NEVER);
00043 }
00044

```

```

00045 L4_INLINE l4_msgtag_t
00046 l4_task_vgicc_map(l4_cap_idx_t task, l4_fpage_t vgicc_fpage) L4_NOTHROW
00047 {
00048     return l4_task_vgicc_map_u(task, vgicc_fpage, l4_utcb());
00049 }

```

## 17.442 \_\_timeout.h

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *                Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *                Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *                economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #ifndef L4_SYS_TIMEOUT_H__
00015 #define L4_SYS_TIMEOUT_H__
00016
00017 #include <l4/sys/l4int.h>
00018 #include <l4/sys/compiler.h>
00019
00025
00040 typedef struct l4_timeout_s {
00041     l4_uint16_t t;
00042 } __attribute__((packed)) l4_timeout_s;
00043
00044
00052 typedef union l4_timeout_t
00053 {
00054     l4_uint32_t raw;
00055     struct
00056     {
00057         #ifdef __BIG_ENDIAN__
00058             l4_timeout_s snd;
00059             l4_timeout_s rcv;
00060         #else
00061             l4_timeout_s rcv;
00062             l4_timeout_s snd;
00063         #endif
00064     } p;
00065 } l4_timeout_t;
00066
00067
00073 #define L4_IPC_TIMEOUT_0 ((l4_timeout_s){0x0400})
00074 #define L4_IPC_TIMEOUT_NEVER ((l4_timeout_s){0})
00075 #define L4_IPC_NEVER_INITIALIZER {0}
00076 #define L4_IPC_NEVER ((l4_timeout_t){0})
00077 #define L4_IPC_RECV_TIMEOUT_0 ((l4_timeout_t){0x00000400})
00078 #define L4_IPC_SEND_TIMEOUT_0 ((l4_timeout_t){0x04000000})
00079 #define L4_IPC_BOTH_TIMEOUT_0 ((l4_timeout_t){0x04000400})
00080
00085 #define L4_TIMEOUT_US_NEVER (~0ULL)
00086
00091 #define L4_TIMEOUT_US_MAX ((1ULL << 41) - 1)
00092
00094
00104 L4_CONSTEXPR L4_INLINE
00105 l4_timeout_s l4_timeout_rel(unsigned man, unsigned exp) L4_NOTHROW;
00106
00107
00117 L4_CONSTEXPR L4_INLINE
00118 l4_timeout_t l4_ipc_timeout(unsigned snd_man, unsigned snd_exp,
00119                             unsigned rcv_man, unsigned rcv_exp) L4_NOTHROW;
00120
00130 L4_CONSTEXPR L4_INLINE
00131 l4_timeout_t l4_timeout(l4_timeout_s snd, l4_timeout_s rcv) L4_NOTHROW;
00132
00140 L4_CONSTEXPR L4_INLINE
00141 void l4_snd_timeout(l4_timeout_s snd, l4_timeout_t *to) L4_NOTHROW;
00142
00150 L4_CONSTEXPR L4_INLINE
00151 void l4_rcv_timeout(l4_timeout_s rcv, l4_timeout_t *to) L4_NOTHROW;
00152
00161 L4_CONSTEXPR L4_INLINE
00162 l4_kernel_clock_t l4_timeout_rel_get(l4_timeout_s to) L4_NOTHROW;
00163
00164
00173 L4_CONSTEXPR L4_INLINE
00174 unsigned l4_timeout_is_absolute(l4_timeout_s to) L4_NOTHROW;
00175
00185 L4_CONSTEXPR L4_INLINE

```

```

00186 l4_kernel_clock_t l4_timeout_get(l4_kernel_clock_t cur, l4_timeout_s to) L4_NOTHROW;
00187
00195 L4_CONSTEXPR L4_INLINE
00196 l4_timeout_s l4_timeout_from_us(l4_uint64_t us) L4_NOTHROW;
00197
00198 /*
00199  * Implementation
00200  */
00201
00202 L4_CONSTEXPR L4_INLINE
00203 l4_timeout_t l4_ipc_timeout(unsigned snd_man, unsigned snd_exp,
00204                             unsigned rcv_man, unsigned rcv_exp) L4_NOTHROW
00205 {
00206     l4_uint16_t snd = (snd_man & 0x3ff) | ((snd_exp << 10) & 0x7c00);
00207     l4_uint16_t rcv = (rcv_man & 0x3ff) | ((rcv_exp << 10) & 0x7c00);
00208     return l4_timeout((l4_timeout_s){snd}, (l4_timeout_s){rcv});
00209 }
00210
00211
00212 L4_CONSTEXPR L4_INLINE
00213 l4_timeout_t l4_timeout(l4_timeout_s snd, l4_timeout_s rcv) L4_NOTHROW
00214 {
00215     return (l4_timeout_t){ ((l4_uint32_t){snd.t} << 16) | rcv.t };
00216 }
00217
00218
00219 L4_CONSTEXPR L4_INLINE
00220 void l4_snd_timeout(l4_timeout_s snd, l4_timeout_t *to) L4_NOTHROW
00221 {
00222     to->p.snd = snd;
00223 }
00224
00225
00226 L4_CONSTEXPR L4_INLINE
00227 void l4_rcv_timeout(l4_timeout_s rcv, l4_timeout_t *to) L4_NOTHROW
00228 {
00229     to->p.rcv = rcv;
00230 }
00231
00232
00233 L4_CONSTEXPR L4_INLINE
00234 l4_timeout_s l4_timeout_rel(unsigned man, unsigned exp) L4_NOTHROW
00235 {
00236     return (l4_timeout_s){(l4_uint16_t)((man & 0x3ff) | ((exp << 10) & 0x7c00))};
00237 }
00238
00239
00240 L4_CONSTEXPR L4_INLINE
00241 l4_kernel_clock_t l4_timeout_rel_get(l4_timeout_s to) L4_NOTHROW
00242 {
00243     if (to.t == 0)
00244         return ~0ULL;
00245     return (l4_kernel_clock_t)(to.t & 0x3ff) << ((to.t >> 10) & 0x1f);
00246 }
00247
00248
00249 L4_CONSTEXPR L4_INLINE
00250 unsigned l4_timeout_is_absolute(l4_timeout_s to) L4_NOTHROW
00251 {
00252     return to.t & 0x8000;
00253 }
00254
00255
00256 L4_CONSTEXPR L4_INLINE
00257 l4_kernel_clock_t l4_timeout_get(l4_kernel_clock_t cur, l4_timeout_s to) L4_NOTHROW
00258 {
00259     if (l4_timeout_is_absolute(to))
00260         return 0; /* We cannot retrieve the value ... */
00261     else
00262         return cur + l4_timeout_rel_get(to);
00263 }
00264
00265 L4_CONSTEXPR L4_INLINE
00266 l4_timeout_s l4_timeout_from_us(l4_uint64_t us) L4_NOTHROW
00267 {
00268     if (us == 0)
00269         return L4_IPC_TIMEOUT_0;
00270     else if (us == L4_TIMEOUT_US_NEVER || us > L4_TIMEOUT_US_MAX)
00271         return L4_IPC_TIMEOUT_NEVER;
00272     else
00273     {
00274         /* Here it is certain that at least one bit in 'us' is set. */
00275
00276         l4_uint16_t m = 0; // initialization required by constexpr, optimized away
00277         l4_uint16_t v = 0; // initialization required by constexpr, optimized away
00278         int e = (63 - __builtin_clzll(us)) - 9;
00279         if (e < 0)

```

```

00280     e = 0;
00281
00282     /* Here it is certain that '0 <= e <= 31' and '1 <= 2^e <= 2^31':
00283     * L4_TIMEOUT_US_MAX = 2^41-1 = 0x000001fffffffffff => e = 31.
00284     * Note: 2^41-1 (0x000001fffffffffff) > 1023*2^31 (0x00001ff800000000). */
00285
00286     m = us >> e;
00287
00288     /* Here it is certain that '1 <= m <= 1023. Consider the following cases:
00289     * o 1 <= us <= 1023: e = 0; 2^e = 1; 1 <= us/1 <= 1023
00290     * o 1024 <= us <= 2047: e = 1; 2^e = 2; 512 <= us/2 <= 1023
00291     * o 2048 <= us <= 4095: e = 2; 2^e = 4; 512 <= us/4 <= 1023
00292     * ...
00293     * o 2^31 <= us <= 2^32-1: e = 22; 512 <= us/2^22 <= 1023
00294     * o 2^40 <= us <= 2^41-1: e = 31; 512 <= us/2^31 <= 1023
00295     *
00296     * Dividing by (1<e) ensures that for all us < 2^41: m < 2^10.
00297     *
00298     * Maximum possible timeout using this format: L4_TIMEOUT_US_MAX = 2^41-1:
00299     * e = 31, m = 1023 => 2'196'875'771'904 us = 610h 14m 35s.
00300     */
00301
00302     /* Without introducing 'v' we had to type-cast the expression to
00303     * l4_uint16_t. This cannot be avoided by declaring m and e_pow_10 as
00304     * l4_uint16_t due to C++ integer promotion. */
00305     v = (e < 10) | m;
00306     return (l4_timeout_s){v};
00307 }
00308 }
00309
00310 #endif

```

## 17.443 I4/sys/\_\_typeinfo.h File Reference

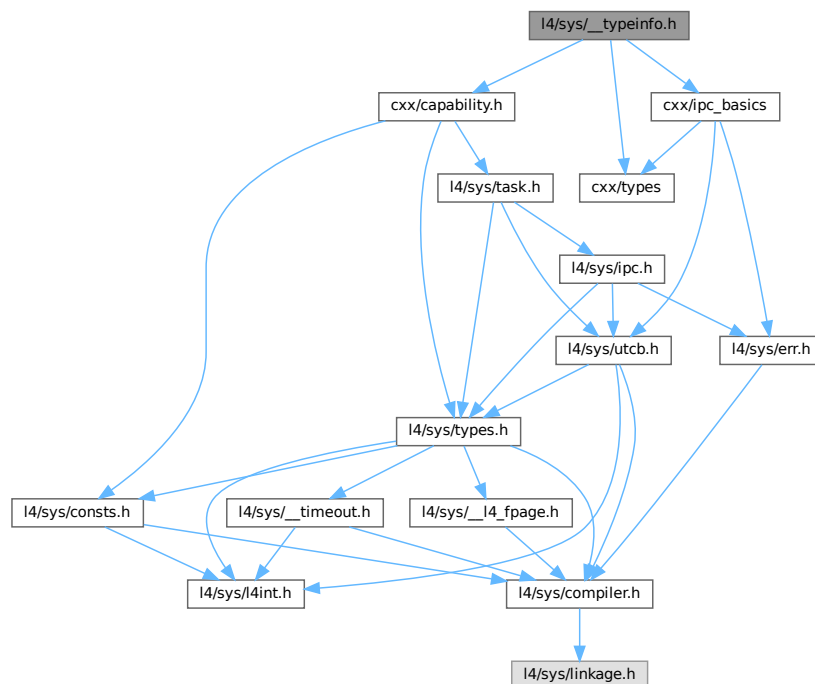
Type information handling.

```

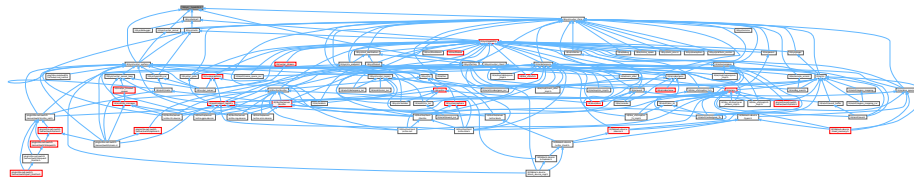
#include "cxx/types"
#include "cxx/ipc_basics"
#include "cxx/capability.h"

```

Include dependency graph for \_\_typeinfo.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [L4::Typeid::P\\_dispatch< LIST >](#)  
*Use for protocol based dispatch stage.*
- struct [L4::Typeid::Detail::Rpc\\_end](#)  
*Internal end-of-list marker.*
- struct [L4::Typeid::Detail::\\_Rpc< OPCODE, O, X >](#)  
*Empty list of RPCs.*
- struct [L4::Typeid::Detail::\\_Rpc< OPCODE, O, R, X... >](#)  
*Non-empty list of RPCs.*
- struct [L4::Typeid::Detail::\\_Rpc< OPCODE, O, R, X... >::Rpc< Y >](#)  
*Find the given RPC in the list.*
- struct [L4::Typeid::Detail::\\_Rpc< OPCODE, O, Default\\_op< R > >::Rpc< Y >](#)  
*Find the given RPC in the list.*
- struct [L4::Typeid::Raw\\_ipc< CLASS >](#)  
*RPCs list for passing raw incoming IPC to the server object.*
- struct [L4::Typeid::Rpc< RPCS >](#)  
*Standard list of RPCs of an interface.*
- struct [L4::Typeid::Rpc\\_code< OPCODE\\_TYPE >](#)  
*List of RPCs of an interface using a special opcode type.*
- struct [L4::Typeid::Rpc\\_code< OPCODE\\_TYPE >::F< RPCS >](#)
- struct [L4::Typeid::Rpc\\_nocode< OPERATION >](#)  
*List of RPCs of an interface using a single operation without an opcode.*
- struct [L4::Typeid::Rpc\\_sys< ARG >](#)  
*List of RPCs typically used for kernel interfaces.*
- struct [L4::Type\\_info](#)  
*Dynamic Type Information for [L4Re](#) Interfaces.*
- class [L4::Type\\_info::Demand](#)  
*Data type for expressing the needed receive buffers at the server-side of an interface.*
- struct [L4::Type\\_info::Demand\\_t< CAPS, FLAGS, MEM, PORTS >](#)  
*Template type statically describing demand of receive buffers.*
- struct [L4::Type\\_info::Demand\\_union\\_t< D1, D2 >](#)  
*Template type statically describing the combination of two [Demand](#) object.*
- struct [L4::Kobject\\_typeid< T >](#)  
*Meta object for handling access to type information of Kobjects.*
- struct [L4::Kobject\\_typeid< void >](#)  
*Minimalistic ID for void interface.*
- class [L4::Kobject\\_t< Derived, Base, PROTO, S\\_DEMAND >](#)  
*Helper class to create an [L4Re](#) interface class that is derived from a single base class.*
- class [L4::Kobject\\_2t< Derived, Base1, Base2, PROTO, S\\_DEMAND >](#)  
*Helper class to create an [L4Re](#) interface class that is derived from two base classes (see [L4::Kobject\\_t](#)).*
- struct [L4::Kobject\\_3t< Derived, Base1, Base2, Base3, PROTO, S\\_DEMAND >](#)  
*Helper class to create an [L4Re](#) interface class that is derived from three base classes (see [L4::Kobject\\_t](#)).*
- struct [L4::Proto\\_t< P >](#)  
*Data type for defining protocol numbers.*

## Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*
- namespace [L4::Typeid](#)  
*Definition of interface data-type helpers.*

## Typedefs

- typedef int [L4::Opcode](#)  
*Data type for RPC opcodes.*

## Enumerations

- enum { [L4::PROTO\\_ANY](#) = 0 , [L4::PROTO\\_EMPTY](#) = -19 }

## Functions

- template<typename T>  
[Type\\_info](#) const \* [L4::kobject\\_typeid](#) () noexcept  
*Get the [L4::Type\\_info](#) for the [L4Re](#) interface given in T.*

## 17.443.1 Detailed Description

Type information handling.

Definition in file [\\_\\_typeinfo.h](#).

## 17.444 [\\_\\_typeinfo.h](#)

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * Copyright (C) 2014-2017, 2019, 2022-2024 Kernkonzept GmbH.
00007  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00008  */
00009 /*
00010  * (c) 2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * License: see LICENSE.spdx (in this directory or the directories above)
00014  */
00015 #pragma once
00016 #pragma GCC system_header
00017
00018 #include "cxx/types"
00019 #include "cxx/ipc_basics"
00020 #include "cxx/capability.h"
00021
00022 #if defined(__GXX_RTTI) && !defined(L4_NO_RTTI)
00023 #   include <typeinfo>
00024     typedef std::type_info const *L4_std_type_info_ptr;
00025 #   define L4_KOBJECT_META_RTTI(type) (&typeid(type))
00026     inline char const *L4_kobject_type_name(L4_std_type_info_ptr n) noexcept
00027     { return n ? n->name() : 0; }
00028 #else
00029     typedef void const *L4_std_type_info_ptr;
00030 #   define L4_KOBJECT_META_RTTI(type) (0)

```

```

00031     inline char const *L4_kobject_type_name(L4_std_type_info_ptr) noexcept
00032     { return 0; }
00033 #endif
00034
00035 namespace L4 {
00036     typedef int Opcode;
00037     // internal max helpers
00038     namespace __I {
00039         // internal max of A nd B helper
00040         template< unsigned char A, unsigned char B>
00041         struct Max { enum { Res = A > B ? A : B }; };
00042     } // namespace __I
00043
00044     enum
00045     {
00047         PROTO_ANY = 0,
00049         PROTO_EMPTY = -19,
00050     };
00051
00052 namespace Typeid {
00071     using namespace L4::Types;
00072
00073     /*****
00074     template<long P, typename T>
00075     struct Iface
00076     {
00077         typedef Iface type;
00078         typedef T iface_type;
00079         enum { Proto = P };
00080     };
00081
00082     /*****
00083     struct Iface_list_end
00084     {
00085         typedef Iface_list_end type;
00086         static bool contains(long) noexcept { return false; }
00087     };
00088
00089     template<typename I, typename N = Iface_list_end>
00090     struct Iface_list
00091     {
00092         typedef Iface_list<I, N> type;
00093
00094         typedef typename I::iface_type iface_type;
00095         typedef N Next;
00096
00097         enum { Proto = I::Proto };
00098
00099         static bool contains(long proto) noexcept
00100         { return (proto == Proto) || Next::contains(proto); }
00101     };
00102
00103     // do not insert PROTO_EMPTY interfaces
00104     template<typename I, typename N>
00105     struct Iface_list<Iface<PROTO_EMPTY, I>, N> : N {};
00106
00107     // do not insert 'void' type interfaces
00108     template<long P, typename N>
00109     struct Iface_list<Iface<P, void>, N> : N {};
00110
00111     /*****
00112     * \internal
00113     * Test if an interface I is in list L
00114     * \tparam I Interface for lookup
00115     * \tparam L Iface_list for search
00116     */
00117     template< typename I, typename L >
00118     struct _In_list;
00119
00120     template< typename I >
00121     struct _In_list<I, Iface_list_end> : False {};
00122
00123     template< typename I, typename N >
00124     struct _In_list<I, Iface_list<I, N> > : True {};
00125
00126     template< typename I, typename I2, typename N >
00127     struct _In_list<I, Iface_list<I2, N> > : _In_list<I, typename N::type> {};
00128
00129     template<typename I, typename L>
00130     struct In_list : _In_list<typename I::type, typename L::type> {};
00131
00132
00133
00134
00135
00136
00137
00138
00139
00140
00141
00142
00143
00144
00145
00146
00147
00148
00149
00150
00151
00152
00153
00154
00155

```

```

00156
00157  /*****
00158  /*
00159  * \internal
00160  * Add Helper: add I to interface list L if ADD is true
00161  * \ingroup l4_cxx_ipc_internal
00162  */
00163  template< bool ADD, typename I, typename L>
00164  struct _Iface_list_add;
00165
00166  template< typename I, typename L>
00167  struct _Iface_list_add<false, I, L> : L {};
00168
00169  template< typename I, typename L>
00170  struct _Iface_list_add<true, I, L> : Iface_list<I, L> {};
00171
00172  /*
00173  * \internal
00174  * Add Helper: add I to interface list L if not already in L.
00175  * \ingroup l4_cxx_ipc_internal
00176  */
00177  template< typename I, typename L >
00178  struct Iface_list_add :
00179      _Iface_list_add<
00180          !In_list<I, typename L::type>::value, I, typename L::type>
00181      > {};
00182
00183  /*****
00184  /*
00185  * \internal
00186  * Helper: checking for a conflict between I1 and I2.
00187  * A conflict means I1 and I2 have the same protocol ID but a different
00188  * iface_type.
00189  */
00190  template< typename I1, typename I2 >
00191  struct __Iface_conflict : Bool<I1::Proto != PROTO_EMPTY && I1::Proto == I2::Proto> {};
00192
00193  template< typename I >
00194  struct __Iface_conflict<I, I> : False {};
00195
00196  /*
00197  * \internal
00198  * Helper: checking for a conflict between I and any interface in LIST.
00199  */
00200  template< typename I, typename LIST >
00201  struct _Iface_conflict;
00202
00203  template< typename I >
00204  struct _Iface_conflict<I, Iface_list_end> : False {};
00205
00206  template< typename I, typename I2, typename LIST >
00207  struct _Iface_conflict<I, Iface_list<I2, LIST> > :
00208      Bool<__Iface_conflict<I, I2>::value || _Iface_conflict<I, typename LIST::type>::value>
00209      > {};
00210
00211  template< typename I, typename LIST >
00212  struct Iface_conflict : _Iface_conflict<typename I::type, typename LIST::type> {};
00213
00214  /*****
00215  /*
00216  * \internal
00217  * Helper: merge two interface lists
00218  */
00219  template< typename L1, typename L2 >
00220  struct _Merge_list;
00221
00222  template< typename L >
00223  struct _Merge_list<Iface_list_end, L> : L {};
00224
00225  template< typename I, typename L1, typename L2 >
00226  struct _Merge_list<Iface_list<I, L1>, L2> :
00227      _Merge_list<typename L1::type, typename Iface_list_add<I, L2>::type> {};
00228
00229  template<typename L1, typename L2>
00230  struct Merge_list : _Merge_list<typename L1::type, typename L2::type> {};
00231
00232  /*****
00233  /*
00234  * \internal
00235  * check for conflicts among all interfaces in L1 with any interfaces in L2.
00236  */
00237  template< typename L1, typename L2 >
00238  struct _Conflict;
00239
00240  template< typename L >
00241  struct _Conflict<Iface_list_end, L> : False {};
00242
00243
00244
00245
00246

```



```

00247     template< typename I, typename L1, typename L2 >
00248     struct _Conflict<Iface_list<I, L1>, L2> :
00249         Bool<Iface_conflict<I, typename L2::type>::value
00250             || _Conflict<typename L1::type, typename L2::type>::value> {};
00251
00252     template< typename L1, typename L2 >
00253     struct Conflict : _Conflict<typename L1::type, typename L2::type> {};
00254
00255     // to be removed -----
00256     // p_dispatch code -- for legacy dispatch -----
00257     /*****
00258     /*
00259     * \internal
00260     * helper: Dispatch helper for calling server-side p_dispatch() functions.
00261     */
00262     template<typename LIST>
00263     struct _P_dispatch;
00264
00265     // No matching dispatcher found
00266     template<>
00267     struct _P_dispatch<Iface_list_end>
00268     {
00269         template< typename THIS, typename A1, typename A2 >
00270         static int f(THIS *, long, A1, A2 &) noexcept
00271         { return -L4_EBADPROTO; }
00272     };
00273
00274
00275     // call matching p_dispatch() function
00276     template< typename I, typename LIST >
00277     struct _P_dispatch<Iface_list<I, LIST> >
00278     {
00279         // special handling for the meta protocol, to avoid 'using' murx
00280         template< typename THIS, typename A1, typename A2 >
00281         static int _f(THIS self, A1, A2 &a2, True::type)
00282         {
00283             return self->dispatch_meta_request(a2);
00284         }
00285
00286         // normal p_dispatch() dispatching
00287         template< typename THIS, typename A1, typename A2 >
00288         static int _f(THIS self, A1 a1, A2 &a2, False::type)
00289         {
00290             return self->p_dispatch(reinterpret_cast<typename I::iface_type *>(0),
00291                                     a1, a2);
00292         }
00293
00294         // dispatch function with switch for meta protocol
00295         template< typename THIS, typename A1, typename A2 >
00296         static int f(THIS *self, long proto, A1 a1, A2 &a2)
00297         {
00298             if (I::Proto == proto)
00299                 return _f(self, a1, a2,
00300                           Bool<I::Proto == static_cast<long>(L4_PROTO_META)>());
00301
00302             return _P_dispatch<typename LIST::type>::f(self, proto, a1, a2);
00303         }
00304     };
00305
00306     template<typename LIST>
00307     struct P_dispatch : _P_dispatch<typename LIST::type> {};
00308
00309     // end: p_dispatch -----
00310     // end: to be removed -----
00311
00312     template<typename RPC> struct Default_op;
00313
00314     namespace Detail {
00315
00316     struct Rpcs_end
00317     {
00318         typedef void opcode_type;
00319         typedef Rpcs_end rpc;
00320         typedef Rpcs_end type;
00321     };
00322
00323     template<typename O1, typename O2, typename RPCS>
00324     struct _Rpc : _Rpc<typename RPCS::next::rpc, O2, typename RPCS::next>::type {};
00325
00326     template<typename O1, typename O2>
00327     struct _Rpc<O1, O2, Rpcs_end> {};
00328
00329     template<typename OP, typename RPCS>
00330     struct _Rpc<OP, OP, RPCS> : RPCS
00331     {
00332         typedef _Rpc type;
00333     };
00334
00335
00336
00337

```

```

00338 template<typename OP, typename RPCS>
00339 struct Rpc : _Rpc<typename RPCS::rpc, OP, RPCS> {};
00340
00341 template<typename T, unsigned CODE>
00342 struct _Get_opcode
00343 {
00344     template<bool, typename> struct Invalid_opcode {};
00345     template<typename X> struct Invalid_opcode<true, X>;
00346
00347 private:
00348     template<typename U, U> struct _chk;
00349     template<typename U> static long _opc(_chk<int, U::Opcode> *);
00350     template<typename U> static char _opc(...);
00351
00352     template<unsigned SZ, typename U>
00353     struct _Opc { enum { value = CODE }; };
00354
00355     template<typename U>
00356     struct _Opc<sizeof(long), U> { enum { value = U::Opcode }; };
00357
00358 public:
00359     enum { value = _Opc<sizeof(_opc<T>{0}), T::value };
00360     Invalid_opcode<(value < CODE), T> invalid_opcode;
00361 };
00362
00363 template<typename OPCODE, unsigned O, typename ...X>
00364 struct _Rpc : Rpc_end {};
00365
00366 template<typename OPCODE, unsigned O, typename R, typename ...X>
00367 struct _Rpc<OPCODE, O, R, X...>
00368 {
00369     typedef _Rpc type;
00370     typedef OPCODE opcode_type;
00371     typedef R rpc;
00372     typedef typename _Rpc<OPCODE, _Get_opcode<R, O>::value + 1, X...>::type next;
00373     enum { Opcode = _Get_opcode<R, O>::value };
00374     template<typename Y> struct Rpc : Typeid::Detail::Rpc<Y, _Rpc> {};
00375 };
00376
00377 template<typename OPCODE, unsigned O, typename R>
00378 struct _Rpc<OPCODE, O, Default_op<R> >
00379 {
00380     typedef _Rpc type;
00381     typedef void opcode_type;
00382     typedef R rpc;
00383     typedef Rpc_end next;
00384     enum { Opcode = -99 };
00385     template<typename Y> struct Rpc : Typeid::Detail::Rpc<Y, _Rpc> {};
00386 };
00387
00388 } // namespace Detail
00389
00390 template<typename CLASS>
00391 struct Raw_ipc
00392 {
00393     typedef Raw_ipc type;
00394     typedef Detail::Rpc_end next;
00395     typedef void opcode_type;
00396 };
00397
00398 template<typename ...RPCS>
00399 struct Rpc : Detail::_Rpc<L4::Opcode, 0, RPCS...> {};
00400
00401 template<typename OPCODE_TYPE>
00402 struct Rpc_code
00403 {
00404     template<typename ...RPCS>
00405     struct F : Detail::_Rpc<OPCODE_TYPE, 0, RPCS...> {};
00406 };
00407
00408 template<typename OPERATION>
00409 struct Rpc_nocode : Detail::_Rpc<void, 0, OPERATION> {};
00410
00411 template<typename ...ARG>
00412 struct Rpc_sys : Detail::_Rpc<l4_umword_t, 0, ARG...> {};
00413
00414 template<typename CLASS>
00415 struct Rights
00416 {
00417     unsigned rights;
00418     Rights(unsigned rights) noexcept : rights(rights) {}
00419     unsigned operator & (unsigned rhs) const noexcept { return rights & rhs; }
00420 };
00421
00422 } // namespace Typeid
00423
00424 struct L4_EXPORT Type_info

```

```

00500 {
00506     class L4_EXPORT Demand
00507     {
00508     private:
00510         static unsigned char max(unsigned char a, unsigned char b) noexcept
00511         { return a > b ? a : b; }
00512     public:
00514         unsigned char caps;
00515         unsigned char flags;
00516         unsigned char mem;
00517         unsigned char ports;
00518
00526         explicit
00527         Demand(unsigned char caps = 0, unsigned char flags = 0,
00528                unsigned char mem = 0, unsigned char ports = 0) noexcept
00529         : caps(caps), flags(flags), mem(mem), ports(ports) {}
00530
00532         bool no_demand() const noexcept
00533         { return caps == 0 && mem == 0 && ports == 0 && flags == 0; }
00534
00536         Demand operator | (Demand const &rhs) const noexcept
00537         {
00538             return Demand(max(caps, rhs.caps), flags | rhs.flags,
00539                            max(mem, rhs.mem), max(ports, rhs.ports));
00540         }
00541     };
00542
00551     template<unsigned char CAPS = 0, unsigned char FLAGS = 0,
00552              unsigned char MEM = 0, unsigned char PORTS = 0>
00553     struct Demand_t : Demand
00554     {
00555     enum
00556     {
00557         Caps = CAPS,
00558         Flags = FLAGS,
00559         Mem = MEM,
00560         Ports = PORTS
00561     };
00562     Demand_t() noexcept : Demand(CAPS, FLAGS, MEM, PORTS) {}
00563     };
00564
00572     template<typename D1, typename D2>
00573     struct Demand_union_t : Demand_t<__I::Max<D1::Caps, D2::Caps>::Res,
00574                                     D1::Flags | D2::Flags,
00575                                     __I::Max<D1::Mem, D2::Mem>::Res,
00576                                     __I::Max<D1::Ports, D2::Ports>::Res>
00577     {};
00578
00579     L4_std_type_info_ptr _type;
00580     Type_info const *const *_bases;
00581     unsigned _num_bases;
00582     long _proto;
00583
00584     L4_std_type_info_ptr type() const noexcept { return _type; }
00585     Type_info const *base(unsigned idx) const noexcept { return _bases[idx]; }
00586     unsigned num_bases() const noexcept { return _num_bases; }
00587     long proto() const noexcept { return _proto; }
00588     char const *name() const noexcept { return L4_kobject_type_name(type()); }
00589     bool has_proto(long proto) const noexcept
00590     {
00591         if (_proto && _proto == proto)
00592             return true;
00593
00594         if (!proto)
00595             return false;
00596
00597         for (unsigned i = 0; i < _num_bases; ++i)
00598             if (base(i)->has_proto(proto))
00599                 return true;
00600
00601         return false;
00602     }
00603 };
00604
00610 template<typename T> struct Kobject_typeid
00611 {
00622     typedef typename T::__Kobject_typeid::Demand Demand;
00623     typedef typename T::__Iface::iface_type Iface;
00624     typedef typename T::__Iface_list Iface_list;
00625
00630     static Type_info const *id() noexcept { return &T::__Kobject_typeid::_m; }
00631
00639     static Type_info::Demand demand() noexcept
00640     { return T::__Kobject_typeid::Demand(); }
00641
00642     // to be removed -----

```

```

00643 // p_dispatch -----
00659 template<typename THIS, typename A1, typename A2>
00660 static int proto_dispatch(THIS *self, long proto, A1 a1, A2 &a2)
00661 { return typeid::P_dispatch<typename T::__Iface_list>::f(self, proto, a1, a2); }
00662 // p_dispatch -----
00663 // end: to be removed -----
00664 };
00665
00667 template<> struct Kobject_typeid<void>
00668 {
00669     typedef Type_info::Demand_t<> Demand;
00670 };
00671
00680 template<typename T>
00681 inline
00682 Type_info const *kobject_typeid() noexcept
00683 { return Kobject_typeid<T>::id(); }
00684
00689 #define L4___GEN_TI(t...)
00690 Type_info const t::__Kobject_typeid::_m =
00691 {
00692     L4_KOBJECT_META_RTTI(Derived),
00693     &t::__Kobject_typeid::_b[0],
00694     sizeof(t::__Kobject_typeid::_b) / sizeof(t::__Kobject_typeid::_b[0]),
00695     PROTO
00696 }
00697
00702 #define L4___GEN_TI_MEMBERS(BASE_DEMAND...)
00703 private:
00704     template< typename T > friend struct Kobject_typeid;
00705 protected:
00706     struct __Kobject_typeid {
00707         typedef Type_info::Demand_union_t<S_DEMAND, BASE_DEMAND> Demand;
00708         static Type_info const *const _b[];
00709         static Type_info const _m;
00710     };
00711 public:
00712     static long const Protocol = PROTO;
00713     typedef L4::Typeid::Rights<Class> Rights;
00714
00743 template<
00744     typename Derived,
00745     typename Base,
00746     long PROTO = PROTO_ANY,
00747     typename S_DEMAND = Type_info::Demand_t<>
00748 >
00749 class Kobject_t : public Base
00750 {
00751 protected:
00753     typedef Derived Class;
00755     typedef Typeid::Iface<PROTO, Derived> __Iface;
00757     typedef Typeid::Merge_list<
00758         Typeid::Iface_list<__Iface>, typename Base::__Iface_list
00759     > __Iface_list;
00760
00762     static void __check_protocols__() noexcept
00763     {
00764         typedef Typeid::Iface_conflict<__Iface, typename Base::__Iface_list> Base_conflict;
00765         static_assert(!Base_conflict::value, "ambiguous protocol ID: protocol also used by Base");
00766     }
00767
00769     L4::Cap<Class> c() const noexcept { return L4::Cap<Class>(this->cap()); }
00770
00771     // Generate the remaining type information
00772     L4___GEN_TI_MEMBERS(typename Base::__Kobject_typeid::Demand)
00773 };
00774
00775
00777 template< typename Derived, typename Base, long PROTO, typename S_DEMAND>
00778 Type_info const *const
00779 Kobject_t<Derived, Base, PROTO, S_DEMAND>::
00780     __Kobject_typeid::_b[] = { &Base::__Kobject_typeid::_m };
00781
00782
00787 template< typename Derived, typename Base, long PROTO, typename S_DEMAND>
00788 L4___GEN_TI(Kobject_t<Derived, Base, PROTO, S_DEMAND>);
00789
00790
00820 template<
00821     typename Derived,
00822     typename Base1,
00823     typename Base2,
00824     long PROTO = PROTO_ANY,
00825     typename S_DEMAND = Type_info::Demand_t<>
00826 >
00827 class Kobject_2t : public Base1, public Base2
00828 {
00829 protected:

```

```

00831     typedef Derived Class;
00833     typedef Typeid::Iface<PROTO, Derived> __Iface;
00835     typedef Typeid::Merge_list<
00836         Typeid::Iface_list<__Iface>,
00837         Typeid::Merge_list<
00838             typename Base1::__Iface_list,
00839             typename Base2::__Iface_list
00840         >
00841     > __Iface_list;
00842
00844     static void __check_protocols__() noexcept
00845     {
00846         typedef typename Base1::__Iface_list Base1_proto_list;
00847         typedef typename Base2::__Iface_list Base2_proto_list;
00848
00849         typedef Typeid::Iface_conflict<__Iface, Base1_proto_list> Base1_conflict;
00850         typedef Typeid::Iface_conflict<__Iface, Base2_proto_list> Base2_conflict;
00851         static_assert(!Base1_conflict::value, "ambiguous protocol ID, also in Base1");
00852         static_assert(!Base2_conflict::value, "ambiguous protocol ID, also in Base2");
00853
00854         typedef Typeid::Conflict<Base1_proto_list, Base2_proto_list> Bases_conflict;
00855         static_assert(!Bases_conflict::value, "ambiguous protocol IDs in base classes");
00856     }
00857
00858     // disambiguate cap()
00859     l4_cap_idx_t cap() const noexcept
00860     { return Base1::cap(); }
00861
00863     L4::Cap<Class> c() const noexcept { return L4::Cap<Class>(this->cap()); }
00864
00865     L4__GEN_TI_MEMBERS(Type_info::Demand_union_t<
00866         typename Base1::__Kobject_typeid::Demand,
00867         typename Base2::__Kobject_typeid::Demand>
00868     )
00869
00870 public:
00871     // Provide non-ambiguous conversion to Kobject
00872     operator Kobject const & () const noexcept
00873     { return *static_cast<Base1 const *>(this); }
00874
00875     // Provide non-ambiguous access of dec_refcnt()
00876     l4_msgtag_t dec_refcnt(l4_mword_t diff, l4_utcb_t *utcb = l4_utcb())
00877     noexcept(noexcept(static_cast<Base1*>(nullptr)->dec_refcnt(diff, utcb)))
00878     { return Base1::dec_refcnt(diff, utcb); }
00879 };
00880
00881
00883 template< typename Derived, typename Base1, typename Base2,
00884     long PROTO, typename S_DEMAND >
00885 Type_info const *const
00886 Kobject_2t<Derived, Base1, Base2, PROTO, S_DEMAND>::__Kobject_typeid::b[] =
00887 {
00888     &Base1::__Kobject_typeid::_m,
00889     &Base2::__Kobject_typeid::_m
00890 };
00891
00892
00897 template< typename Derived, typename Base1, typename Base2,
00898     long PROTO, typename S_DEMAND >
00899 L4__GEN_TI(Kobject_2t<Derived, Base1, Base2, PROTO, S_DEMAND>);
00900
00901
00902 template<
00903     typename Derived,
00904     typename Base1,
00905     typename Base2,
00906     typename Base3,
00907     long PROTO = PROTO_ANY,
00908     typename S_DEMAND = Type_info::Demand_t<>
00909 >
00930 struct Kobject_3t : Base1, Base2, Base3
00931 {
00932 protected:
00934     typedef Derived Class;
00936     typedef Typeid::Iface<PROTO, Derived> __Iface;
00938     typedef Typeid::Merge_list<
00939         Typeid::Iface_list<__Iface>,
00940         Typeid::Merge_list<
00941             typename Base1::__Iface_list,
00942             Typeid::Merge_list<
00943                 typename Base2::__Iface_list,
00944                 typename Base3::__Iface_list
00945             >
00946         >
00947     > __Iface_list;
00948
00950     static void __check_protocols__() noexcept

```

```

00951 {
00952     typedef typename Base1::__Iface_list Base1_proto_list;
00953     typedef typename Base2::__Iface_list Base2_proto_list;
00954     typedef typename Base3::__Iface_list Base3_proto_list;
00955
00956     typedef Typeid::Iface_conflict<__Iface, Base1_proto_list> Base1_conflict;
00957     typedef Typeid::Iface_conflict<__Iface, Base2_proto_list> Base2_conflict;
00958     typedef Typeid::Iface_conflict<__Iface, Base3_proto_list> Base3_conflict;
00959
00960     static_assert(!Base1_conflict::value, "ambiguous protocol ID, also in Base1");
00961     static_assert(!Base2_conflict::value, "ambiguous protocol ID, also in Base2");
00962     static_assert(!Base3_conflict::value, "ambiguous protocol ID, also in Base3");
00963
00964     typedef Typeid::Conflict<Base1_proto_list, Base2_proto_list> Conflict_bases12;
00965     typedef Typeid::Conflict<Base1_proto_list, Base3_proto_list> Conflict_bases13;
00966     typedef Typeid::Conflict<Base2_proto_list, Base3_proto_list> Conflict_bases23;
00967
00968     static_assert(!Conflict_bases12::value, "ambiguous protocol IDs in base classes: Base1 and
Base2");
00969     static_assert(!Conflict_bases13::value, "ambiguous protocol IDs in base classes: Base1 and
Base3");
00970     static_assert(!Conflict_bases23::value, "ambiguous protocol IDs in base classes: Base2 and
Base3");
00971 }
00972
00973 // disambiguate cap()
00974 l4_cap_idx_t cap() const noexcept
00975 { return Base1::cap(); }
00976
00977 L4::Cap<Class> c() const noexcept { return L4::Cap<Class>(this->cap()); }
00978
00979 L4____GEN_TI_MEMBERS(Type_info::Demand_union_t<Type_info::Demand_union_t<
00980     typename Base1::__Kobject_typeid::Demand,
00981     typename Base2::__Kobject_typeid::Demand>,
00982     typename Base3::__Kobject_typeid::Demand>
00983 )
00984
00985 public:
00986 // Provide non-ambiguous conversion to Kobject
00987 operator Kobject const & () const noexcept
00988 { return *static_cast<Base1 const *>(this); }
00989
00990 // Provide non-ambiguous access of dec_refcnt()
00991 l4_msgtag_t dec_refcnt(l4_mword_t diff, l4_utcb_t *utcb = l4_utcb())
00992     noexcept (noexcept (static_cast<Base1*>(nullptr)->dec_refcnt(diff, utcb)))
00993 { return Base1::dec_refcnt(diff, utcb); }
00994
00995 };
00996
00997
00998 template< typename Derived, typename Base1, typename Base2, typename Base3,
00999     long PROTO, typename S_DEMAND >
01000 Type_info const *const
01001 Kobject_3t<Derived, Base1, Base2, Base3, PROTO, S_DEMAND>::__Kobject_typeid::__b[] =
01002 {
01003     &Base1::__Kobject_typeid::__m,
01004     &Base2::__Kobject_typeid::__m,
01005     &Base3::__Kobject_typeid::__m
01006 };
01007
01008
01009 template< typename Derived, typename Base1, typename Base2, typename Base3,
01010     long PROTO, typename S_DEMAND >
01011 L4____GEN_TI(Kobject_3t<Derived, Base1, Base2, Base3, PROTO, S_DEMAND>);
01012
01013
01014 #if __cplusplus >= 201103L
01015 namespace L4 {
01016
01017 template< typename ...T >
01018 struct Kobject_demand;
01019
01020 template<>
01021 struct Kobject_demand<> : Type_info::Demand_t<> {};
01022
01023 template<typename T>
01024 struct Kobject_demand<T> : Kobject_typeid<T>::Demand {};
01025
01026 template<typename T1, typename ...T2>
01027 struct Kobject_demand<T1, T2...> :
01028     Type_info::Demand_union_t<typename Kobject_typeid<T1>::Demand,
01029         Kobject_demand<T2...> >
01030 {};
01031
01032 namespace Typeid_xx {
01033
01034     template<typename ...LISTS>

```

```

01048     struct Merge_list;
01049
01050     template<typename L>
01051     struct Merge_list<L> : L {};
01052
01053     template<typename L1, typename L2>
01054     struct Merge_list<L1, L2> : Typeid::Merge_list<L1, L2> {};
01055
01056     template<typename L1, typename L2, typename ...LISTS>
01057     struct Merge_list<L1, L2, LISTS...> :
01058         Merge_list<typename Typeid::Merge_list<L1, L2>::type, LISTS...> {};
01059
01060     template< typename I, typename ...LIST >
01061     struct Iface_conflict;
01062
01063     template< typename I >
01064     struct Iface_conflict<I> : Typeid::False {};
01065
01066     template< typename I, typename L, typename ...LIST >
01067     struct Iface_conflict<I, L, LIST...> :
01068         Typeid::Bool<Typeid::Iface_conflict<typename I::type, typename L::type>::value
01069             || Iface_conflict<I, LIST...>::value>
01070     {};
01071
01072     template< typename ...LIST >
01073     struct Conflict;
01074
01075     template< typename L >
01076     struct Conflict<L> : Typeid::False {};
01077
01078     template< typename L1, typename L2, typename ...LIST >
01079     struct Conflict<L1, L2, LIST...> :
01080         Typeid::Bool<Typeid::Conflict<typename L1::type, typename L2::type>::value
01081             || Conflict<L1, LIST...>::value
01082             || Conflict<L2, LIST...>::value>
01083     {};
01084
01085     template< typename T >
01086     struct Is_demand
01087     {
01088         static long test(Type_info::Demand const *);
01089         static char test(...);
01090         enum { value = sizeof(test(static_cast<T*>(nullptr))) == sizeof(long) };
01091     };
01092
01093     template< typename T, typename ... >
01094     struct First : T { typedef T type; };
01095 } // Typeid
01096
01102 template< typename Derived, long PROTO, typename S_DEMAND, typename ...BASES>
01103 struct __Kobject_base : BASES...
01104 {
01105 protected:
01106     typedef Derived Class;
01107     typedef Typeid::Iface<PROTO, Derived> __Iface;
01108     typedef Typeid_xx::Merge_list<
01109         Typeid::Iface_list<__Iface>,
01110         typename BASES::__Iface_list...
01111     > __Iface_list;
01112
01113     static void __check_protocols__() noexcept
01114     {
01115         typedef Typeid_xx::Iface_conflict<__Iface, typename BASES::__Iface_list...> Conflict;
01116         static_assert(!Conflict::value, "ambiguous protocol ID, protocol also used in base class");
01117
01118         typedef Typeid_xx::Conflict<typename BASES::__Iface_list...> Base_conflict;
01119         static_assert(!Base_conflict::value, "ambiguous protocol IDs in base classes");
01120     }
01121
01122     // disambiguate cap()
01123     l4_cap_idx_t cap() const noexcept
01124     { return Typeid_xx::First<BASES...>::type::cap(); }
01125
01126     L4::Cap<Class> c() const noexcept { return L4::Cap<Class>(this->cap()); }
01127
01128     L4__GEN_TI_MEMBERS(Kobject_demand<BASES...>)
01129
01130 private:
01131     // This function returns the first base class (used below)
01132     template<typename B1, typename ...> struct Basel { typedef B1 type; };
01133
01134 public:
01135     // Provide non-ambiguous conversion to Kobject
01136     operator Kobject const & () const noexcept
01137     { return *static_cast<typename Basel<BASES...>::type const *>(this); }
01138
01139     // Provide non-ambiguous access of dec_refcnt()

```

```

01140     l4_msgtag_t dec_refcnt(l4_mword_t diff, l4_utcb_t *utcb = l4_utcb())
01141     noexcept(noexcept(static_cast<typename Base1<BASES...>::type *>(nullptr)
01142         ->dec_refcnt(diff, utcb)))
01143     { return Base1<BASES...>::type::dec_refcnt(diff, utcb); }
01144 };
01145
01146 template< typename Derived, long PROTO, typename S_DEMAND, typename ...BASES>
01147 Type_info const *const
01148 __Kobject_base<Derived, PROTO, S_DEMAND, BASES...>::__Kobject_typeid::b[] =
01149 {
01150     (&BASES::__Kobject_typeid::_m)...
01151 };
01152
01153 template< typename Derived, long PROTO, typename S_DEMAND, typename ...BASES>
01154 L4___GEN_TI(__Kobject_base<Derived, PROTO, S_DEMAND, BASES...>);
01155
01156 // Test if the there is a Demand argument to Kobject_x
01157 template< typename Derived, long PROTO, bool HAS_DEMAND, typename DEMAND, typename ...ARGS >
01158 struct __Kobject_x_proto;
01159
01160 // YES: pass it to __Kobject_base
01161 template< typename Derived, long PROTO, typename DEMAND, typename ...BASES>
01162 struct __Kobject_x_proto<Derived, PROTO, true, DEMAND, BASES...> :
01163     __Kobject_base<Derived, PROTO, DEMAND, BASES...> {};
01164
01165 // NO: pass it empty Type_info::Demand_t
01166 template< typename Derived, long PROTO, typename B1, typename ...BASES>
01167 struct __Kobject_x_proto<Derived, PROTO, false, B1, BASES...> :
01168     __Kobject_base<Derived, PROTO, Type_info::Demand_t<>, B1, BASES...> {};
01169
01170 template< long P = PROTO_EMPTY >
01171 struct Proto_t {};
01172
01173 template< typename Derived, typename ...ARGS >
01174 struct Kobject_x;
01175
01176 template< typename Derived, typename A, typename ...ARGS >
01177 struct Kobject_x<Derived, A, ARGS...> :
01178     __Kobject_x_proto<Derived, PROTO_ANY, Typeid_xx::Is_demand<A>::value, A, ARGS...>
01179 {};
01180
01181 template< typename Derived, long PROTO, typename A, typename ...ARGS >
01182 struct Kobject_x<Derived, Proto_t<PROTO>, A, ARGS...> :
01183     __Kobject_x_proto<Derived, PROTO, Typeid_xx::Is_demand<A>::value, A, ARGS...>
01184 {};
01185
01186 }
01187 #endif
01188 #undef L4___GEN_TI
01189 #undef L4___GEN_TI_MEMBERS

```

## 17.445 \_\_vcpu-arm.h

```

00001 /*
00002  * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00003  *
00004  * License: see LICENSE.spdx (in this directory or the directories above)
00005  */
00006 #pragma once
00007
00008 typedef struct l4_arm_vcpu_e_info_t
00009 {
00010     l4_uint8_t version; // must be 0
00011     l4_uint8_t gic_version;
00012     l4_uint8_t _rsvd0[2];
00013     l4_uint32_t features;
00014     l4_uint32_t _rsvd1[14];
00015     l4_umword_t user[8];
00016 } l4_arm_vcpu_e_info_t;
00017
00018 L4_INLINE void *l4_vcpu_e_ptr(void const *vcpu, unsigned id) L4_NOTHROW;
00019
00020 enum L4_vcpu_e_consts
00021 {
00022     L4_VCPU_E_NUM_LR = 4,
00023 };
00024
00025 L4_INLINE l4_arm_vcpu_e_info_t const *
00026 l4_vcpu_e_info(void const *vcpu) L4_NOTHROW;
00027
00028 L4_INLINE l4_umword_t *

```



```

00029 l4_vcpu_e_info_user(void *vcpu) L4_NOTHROW;
00030
00031 L4_INLINE l4_umword_t *
00032 l4_vcpu_e_info_user(void *vcpu) L4_NOTHROW
00033 {
00034     return ((l4_arm_vcpu_e_info_t *)l4_vcpu_e_info(vcpu))->user;
00035 }
00036
00037
00045 L4_INLINE l4_uint32_t
00046 l4_vcpu_e_read_32(void const *vcpu, unsigned id) L4_NOTHROW;
00047
00048 L4_INLINE l4_uint32_t
00049 l4_vcpu_e_read_32(void const *vcpu, unsigned id) L4_NOTHROW
00050 { return *(l4_uint32_t const *)l4_vcpu_e_ptr(vcpu, id); }
00051
00059 L4_INLINE void
00060 l4_vcpu_e_write_32(void *vcpu, unsigned id, l4_uint32_t val) L4_NOTHROW;
00061
00062 L4_INLINE void
00063 l4_vcpu_e_write_32(void *vcpu, unsigned id, l4_uint32_t val) L4_NOTHROW
00064 { *((l4_uint32_t *)l4_vcpu_e_ptr(vcpu, + id)) = val; }
00065
00073 L4_INLINE l4_uint64_t
00074 l4_vcpu_e_read_64(void const *vcpu, unsigned id) L4_NOTHROW;
00075
00076 L4_INLINE l4_uint64_t
00077 l4_vcpu_e_read_64(void const *vcpu, unsigned id) L4_NOTHROW
00078 { return *(l4_uint64_t const *)l4_vcpu_e_ptr(vcpu, id); }
00079
00087 L4_INLINE void
00088 l4_vcpu_e_write_64(void *vcpu, unsigned id, l4_uint64_t val) L4_NOTHROW;
00089
00090 L4_INLINE void
00091 l4_vcpu_e_write_64(void *vcpu, unsigned id, l4_uint64_t val) L4_NOTHROW
00092 { *((l4_uint64_t *)l4_vcpu_e_ptr(vcpu, id)) = val; }
00093
00101 L4_INLINE l4_umword_t
00102 l4_vcpu_e_read(void const *vcpu, unsigned id) L4_NOTHROW;
00103
00104 L4_INLINE l4_umword_t
00105 l4_vcpu_e_read(void const *vcpu, unsigned id) L4_NOTHROW
00106 { return *(l4_umword_t const *)l4_vcpu_e_ptr(vcpu, id); }
00107
00115 L4_INLINE void
00116 l4_vcpu_e_write(void *vcpu, unsigned id, l4_umword_t val) L4_NOTHROW;
00117
00118 L4_INLINE void
00119 l4_vcpu_e_write(void *vcpu, unsigned id, l4_umword_t val) L4_NOTHROW
00120 { *((l4_umword_t *)l4_vcpu_e_ptr(vcpu, id)) = val; }

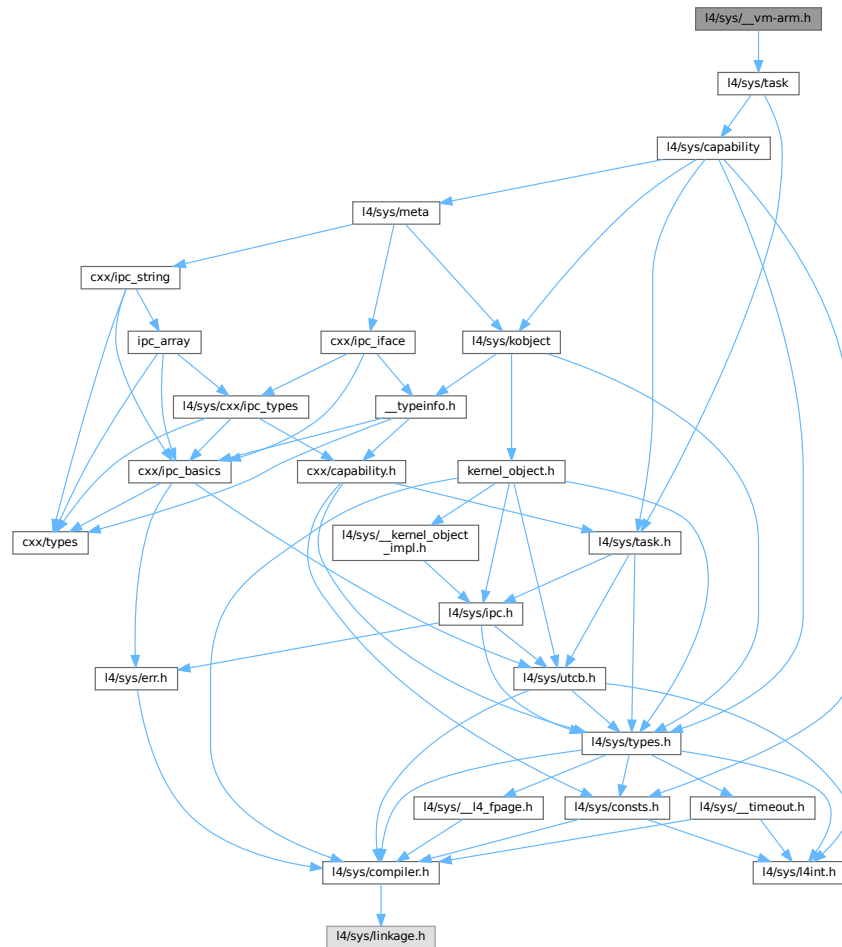
```

## 17.446 l4/sys/\_vm-arm.h File Reference

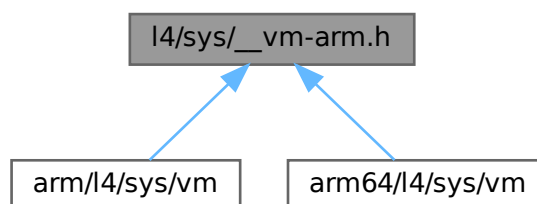
Virtualization interface.

```
#include <l4/sys/task>
```

Include dependency graph for `__vm-arm.h`:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4::Vm](#)

*Virtual machine host address space.*

## Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*

## 17.446.1 Detailed Description

Virtualization interface.

Definition in file [\\_\\_vm-arm.h](#).

## 17.447 \_\_vm-arm.h

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2018 Adam Lackorzynski <adam@l4re.org>
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00013 #include <l4/sys/task>
00014
00015 namespace L4 {
00016
00017 class Vm : public Kobject_t<Vm, Task, L4_PROTO_VM>
00018 {
00019 public:
00030     l4_msgtag_t vgicc_map(l4_fpage_t const vgicc_fpage,
00031                          l4_utcb_t *utcb = l4_utcb()) noexcept
00032     { return l4_task_vgicc_map_u(cap(), vgicc_fpage, utcb); }
00033
00034 protected:
00035     Vm();
00036
00037 private:
00038     Vm(Vm const &);
00039     void operator = (Vm const &);
00040 };
00041
00042 }
```

## 17.448 \_\_vm-svm.h

```
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00015 #include <l4/sys/types.h>
00016
00022
00023
00028 typedef struct l4_vm_svm_vmcb_control_area
00029 {
00030     l4_uint16_t intercept_rd_crX;
00031     l4_uint16_t intercept_wr_crX;
00032
00033     l4_uint16_t intercept_rd_drX;
00034     l4_uint16_t intercept_wr_drX;
00035
00036     l4_uint32_t intercept_exceptions;
```

```

00037
00038     14_uint32_t intercept_instruction0;
00039     14_uint32_t intercept_instruction1;
00040
00041     14_uint8_t _reserved0[40];
00042
00043     14_uint16_t pause_filter_threshold;
00044     14_uint16_t pause_filter_count;
00045
00046     14_uint64_t iopm_base_pa;
00047     14_uint64_t msrpm_base_pa;
00048     14_uint64_t tsc_offset;
00049     14_uint64_t guest_asid_tlb_ctl;
00050     14_uint64_t interrupt_ctl;
00051     14_uint64_t interrupt_shadow;
00052     14_uint64_t exitcode;
00053     14_uint64_t exitinfo1;
00054     14_uint64_t exitinfo2;
00055     14_uint64_t exitintinfo;
00056     14_uint64_t np_enable;
00057
00058     14_uint8_t _reserved1[16];
00059
00060     14_uint64_t eventinj;
00061     14_uint64_t n_cr3;
00062     14_uint64_t lbr_virtualization_enable;
00063     14_uint64_t clean_bits;
00064     14_uint64_t n_rip;
00065
00066     14_uint8_t _reserved2[816];
00067 } __attribute__((packed)) 14_vm_svm_vmcb_control_area_t;
00068
00073 typedef struct 14_vm_svm_vmcb_state_save_area_seg
00074 {
00075     14_uint16_t selector;
00076     14_uint16_t attrib;
00077     14_uint32_t limit;
00078     14_uint64_t base;
00079 } __attribute__((packed)) 14_vm_svm_vmcb_state_save_area_seg_t;
00080
00085 typedef struct 14_vm_svm_vmcb_state_save_area
00086 {
00087     struct 14_vm_svm_vmcb_state_save_area_seg es;
00088     struct 14_vm_svm_vmcb_state_save_area_seg cs;
00089     struct 14_vm_svm_vmcb_state_save_area_seg ss;
00090     struct 14_vm_svm_vmcb_state_save_area_seg ds;
00091     struct 14_vm_svm_vmcb_state_save_area_seg fs;
00092     struct 14_vm_svm_vmcb_state_save_area_seg gs;
00093     struct 14_vm_svm_vmcb_state_save_area_seg gdtr;
00094     struct 14_vm_svm_vmcb_state_save_area_seg ldtr;
00095     struct 14_vm_svm_vmcb_state_save_area_seg idtr;
00096     struct 14_vm_svm_vmcb_state_save_area_seg tr;
00097
00098     14_uint8_t _reserved0[43];
00099
00100     14_uint8_t cpl;
00101
00102     14_uint32_t _reserved1;
00103
00104     14_uint64_t efer;
00105
00106     14_uint8_t _reserved2[112];
00107
00108     14_uint64_t cr4;
00109     14_uint64_t cr3;
00110     14_uint64_t cr0;
00111     14_uint64_t dr7;
00112     14_uint64_t dr6;
00113     14_uint64_t rflags;
00114     14_uint64_t rip;
00115
00116     14_uint8_t _reserved3[88];
00117
00118     14_uint64_t rsp;
00119
00120     14_uint8_t _reserved4[24];
00121
00122     14_uint64_t rax;
00123     14_uint64_t star;
00124     14_uint64_t lstar;
00125     14_uint64_t cstar;
00126     14_uint64_t sfmask;
00127     14_uint64_t kernelgsbase;
00128     14_uint64_t sysenter_cs;
00129     14_uint64_t sysenter_esp;
00130     14_uint64_t sysenter_eip;
00131     14_uint64_t cr2;

```

```

00132
00133     l4_uint8_t _reserved5[32];
00134
00135     l4_uint64_t g_pat;
00136     l4_uint64_t dbgctl;
00137     l4_uint64_t br_from;
00138     l4_uint64_t br_to;
00139     l4_uint64_t lastexcpfrom;
00140     l4_uint64_t last_excpto;
00141
00142     // this field is _NOT_ part of the official VMCB specification
00143     // a (userlevel) VMM needs this for proper FPU state virtualization
00144     l4_uint64_t xcr0;
00145
00146     l4_uint8_t _reserved6[2400];
00147 } __attribute__((packed)) l4_vm_svm_vmc_b_state_save_area_t;
00148
00149
00154 typedef struct l4_vm_svm_vmc_b_t
00155 {
00156     l4_vm_svm_vmc_b_control_area_t control_area;
00157     l4_vm_svm_vmc_b_state_save_area_t state_save_area;
00158 } l4_vm_svm_vmc_b_t;

```

## 17.449 \_\_vm-vmx.h

```

00001
00006 /*
00007  * (c) 2010-2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013
00014 #pragma once
00015
00016 #include <l4/sys/vcpu.h>
00017
00023
00028 enum l4_vm_vmx_caps_regs
00029 {
00030     L4_VM_VMX_BASIC_REG = 0,
00031     L4_VM_VMX_TRUE_PINBASED_CTLs_REG = 1,
00032     L4_VM_VMX_TRUE_PROCBASED_CTLs_REG = 2,
00033     L4_VM_VMX_TRUE_EXIT_CTLs_REG = 3,
00034     L4_VM_VMX_TRUE_ENTRY_CTLs_REG = 4,
00035     L4_VM_VMX_MISC_REG = 5,
00036     L4_VM_VMX_CR0_FIXED0_REG = 6,
00037     L4_VM_VMX_CR0_FIXED1_REG = 7,
00038     L4_VM_VMX_CR4_FIXED0_REG = 8,
00039     L4_VM_VMX_CR4_FIXED1_REG = 9,
00040     L4_VM_VMX_VMCS_ENUM_REG = 10,
00041     L4_VM_VMX_PROCBASED_CTLs2_REG = 11,
00042     L4_VM_VMX_EPT_VPID_CAP_REG = 12,
00043     L4_VM_VMX_NESTED_REVISION = 13,
00044     L4_VM_VMX_NUM_CAPS_REGS
00045 };
00046
00051 enum l4_vm_vmx_dfl1_regs
00052 {
00053     L4_VM_VMX_PINBASED_CTLs_DFL1_REG = 0,
00054     L4_VM_VMX_PROCBASED_CTLs_DFL1_REG = 1,
00055     L4_VM_VMX_EXIT_CTLs_DFL1_REG = 2,
00056     L4_VM_VMX_ENTRY_CTLs_DFL1_REG = 3,
00057     L4_VM_VMX_NUM_DFL1_REGS
00058 };
00059
00069 enum l4_vm_vmx_sw_fields
00070 {
00078     L4_VM_VMX_VMCS_CR2 = 0x6880,
00080     L4_VM_VMX_VMCS_NAT_ARG0 = 0x6882,
00082     L4_VM_VMX_VMCS_NAT_ARG1 = 0x6884,
00084     L4_VM_VMX_VMCS_NAT_ARG2 = 0x6886,
00086     L4_VM_VMX_VMCS_NAT_ARG3 = 0x6888,
00088     L4_VM_VMX_VMCS_XCR0 = 0x2880,
00090     L4_VM_VMX_VMCS_MSR_SYSCALL_MASK = 0x2882,
00092     L4_VM_VMX_VMCS_MSR_LSTAR = 0x2884,
00094     L4_VM_VMX_VMCS_MSR_CSTAR = 0x2886,
00096     L4_VM_VMX_VMCS_MSR_TSC_AUX = 0x2888,
00098     L4_VM_VMX_VMCS_MSR_STAR = 0x288a,
00100     L4_VM_VMX_VMCS_MSR_KERNEL_GS_BASE = 0x288c,
00101 };

```

```

00102
00155 typedef struct l4_vmx_offset_table_t
00156 {
00157     l4_uint8_t offsets[4][4];
00158     l4_uint8_t limits[4][4];
00159     l4_uint8_t index_shifts[4];
00160     l4_uint8_t base_offset;
00161     l4_uint8_t size;
00162
00163     l4_uint8_t reserved[2];
00164 } l4_vmx_offset_table_t;
00165
00170 enum L4_vm_vmx_vmcs_sizes
00171 {
00173     L4_VM_VMX_VMCS_SIZE_VALUES = 2560,
00175     L4_VM_VMX_VMCS_SIZE_DIRTY_BITMAP = 320,
00176 };
00177
00205 typedef struct l4_vm_vmx_vcpu_vmcs_t
00206 {
00207     l4_uint64_t reserved0;
00208
00209     l4_uint64_t user_data;
00210     l4_uint32_t cr2_index;
00211     l4_uint8_t reserved1[4];
00212
00213     l4_cap_idx_t vmcs;
00214
00215     /*
00216      * Since the capability type size depends on the platform, we add a 32-bit
00217      * padding if necessary.
00218      */
00219
00220     #if L4_MWORD_BITS == 32
00221         l4_uint32_t padding0;
00222     #elif L4_MWORD_BITS == 64
00223         /* No padding needed. */
00224     #else
00225         #error Unsupported machine word size.
00226     #endif
00227
00228     l4_vmx_offset_table_t offset_table;
00229     l4_uint8_t reserved2[120];
00230
00231     l4_uint8_t values[L4_VM_VMX_VMCS_SIZE_VALUES];
00232     l4_uint8_t dirty_bitmap[L4_VM_VMX_VMCS_SIZE_DIRTY_BITMAP];
00233 } l4_vm_vmx_vcpu_vmcs_t;
00234
00239 typedef struct l4_vm_vmx_vcpu_infos_t
00240 {
00242     l4_uint64_t caps[L4_VM_VMX_NUM_CAPS_REGS];
00243
00246     l4_uint32_t dfll[L4_VM_VMX_NUM_DFL1_REGS];
00247 } l4_vm_vmx_vcpu_infos_t;
00248
00267 typedef struct l4_vm_vmx_vcpu_state_t
00268 {
00269     l4_vcpu_state_t vcpu_state;
00270     l4_uint8_t padding0[L4_VCPU_OFFSET_EXT_INFOS - sizeof(l4_vcpu_state_t)];
00271
00272     l4_vm_vmx_vcpu_infos_t infos;
00273     l4_uint8_t padding1[L4_VCPU_OFFSET_EXT_STATE - L4_VCPU_OFFSET_EXT_INFOS
00274                       - sizeof(l4_vm_vmx_vcpu_infos_t)];
00275
00276     l4_vm_vmx_vcpu_vmcs_t vmcs;
00277 } l4_vm_vmx_vcpu_state_t;
00278
00288 L4_INLINE
00289 l4_uint64_t
00290 l4_vm_vmx_get_caps(l4_vm_vmx_vcpu_state_t const *vcpu_state,
00291                   enum L4_vm_vmx_caps_regs caps_reg) L4_NOTHROW;
00292
00303 L4_INLINE
00304 l4_uint32_t
00305 l4_vm_vmx_get_caps_default1(l4_vm_vmx_vcpu_state_t const *vcpu_state,
00306                             enum L4_vm_vmx_dfll_regs dfll_reg) L4_NOTHROW;
00307
00315 L4_INLINE
00316 unsigned
00317 l4_vm_vmx_field_len(unsigned field) L4_NOTHROW;
00318
00326 L4_INLINE
00327 unsigned
00328 l4_vm_vmx_field_order(unsigned field) L4_NOTHROW;
00329
00344 L4_INLINE
00345 void *

```

```

00346 l4_vm_vmx_field_ptr(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field) L4_NOTHROW;
00347
00357 L4_INLINE
00358 void
00359 l4_vm_vmx_clear(l4_vm_vmx_vcpu_vmcs_t *vmcs,
00360                l4_vm_vmx_vcpu_vmcs_t *dest_vmcs) L4_NOTHROW;
00361
00371 L4_INLINE
00372 void
00373 l4_vm_vmx_ptr_load(l4_vm_vmx_vcpu_vmcs_t *vmcs,
00374                   l4_vm_vmx_vcpu_vmcs_t *src_vmcs) L4_NOTHROW;
00375
00391 L4_INLINE
00392 l4_uint32_t
00393 l4_vm_vmx_get_cr2_index(l4_vm_vmx_vcpu_vmcs_t const *vmcs) L4_NOTHROW;
00394
00404 L4_INLINE
00405 l4_umword_t
00406 l4_vm_vmx_read_nat(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field) L4_NOTHROW;
00407
00417 L4_INLINE
00418 l4_uint16_t
00419 l4_vm_vmx_read_16(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field) L4_NOTHROW;
00420
00430 L4_INLINE
00431 l4_uint32_t
00432 l4_vm_vmx_read_32(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field) L4_NOTHROW;
00433
00443 L4_INLINE
00444 l4_uint64_t
00445 l4_vm_vmx_read_64(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field) L4_NOTHROW;
00446
00456 L4_INLINE
00457 l4_uint64_t
00458 l4_vm_vmx_read(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field) L4_NOTHROW;
00459
00468 L4_INLINE
00469 void
00470 l4_vm_vmx_write_nat(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field,
00471                    l4_umword_t val) L4_NOTHROW;
00472
00481 L4_INLINE
00482 void
00483 l4_vm_vmx_write_16(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field,
00484                   l4_uint16_t val) L4_NOTHROW;
00485
00494 L4_INLINE
00495 void
00496 l4_vm_vmx_write_32(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field,
00497                   l4_uint32_t val) L4_NOTHROW;
00498
00507 L4_INLINE
00508 void
00509 l4_vm_vmx_write_64(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field,
00510                   l4_uint64_t val) L4_NOTHROW;
00511
00520 L4_INLINE
00521 void
00522 l4_vm_vmx_write(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field,
00523                l4_uint64_t val) L4_NOTHROW;
00524
00571 L4_INLINE
00572 void
00573 l4_vm_vmx_set_hw_vmcs(l4_vm_vmx_vcpu_vmcs_t *vmcs,
00574                      l4_cap_idx_t vmcs_cap) L4_NOTHROW;
00575
00585 L4_INLINE
00586 l4_cap_idx_t
00587 l4_vm_vmx_get_hw_vmcs(l4_vm_vmx_vcpu_vmcs_t *vmcs) L4_NOTHROW;
00588
00589 /* Implementations */
00590
00591 L4_INLINE
00592 unsigned
00593 l4_vm_vmx_field_len(unsigned field) L4_NOTHROW
00594 {
00595     return 1 « l4_vm_vmx_field_order(field);
00596 }
00597
00598 L4_INLINE
00599 unsigned
00600 l4_vm_vmx_field_order(unsigned field) L4_NOTHROW
00601 {
00602     unsigned size = (field » 13) & 0x03U;
00603
00604     switch (size)
00605     {

```

```

00606     case 0: return 1; /* 16 bits */
00607     case 1: return 3; /* 64 bits */
00608     case 2: return 2; /* 32 bits */
00609     case 3: return (sizeof(l4_umword_t) == 8) ? 3 : 2; /* Natural width */
00610     }
00611
00612     __builtin_trap();
00613 }
00614
00620 L4_INLINE
00621 unsigned
00622 l4_vm_vmx_field_offset(l4_vm_vmx_vcpu_vmcs_t const *vmcs,
00623                       unsigned field) L4_NOTHROW
00624 {
00625     unsigned index = field & 0x3feU;
00626     unsigned size = (field >> 13) & 0x03U;
00627     unsigned group = (field >> 10) & 0x03U;
00628
00629     unsigned shifted_index = index << vmcs->offset_table.index_shifts[size];
00630
00631     if (shifted_index >= (unsigned)vmcs->offset_table.limits[size][group] * 64)
00632         return ~0U;
00633
00634     return (unsigned)vmcs->offset_table.offsets[size][group] * 64
00635         + shifted_index;
00636 }
00637
00638 L4_INLINE
00639 void *
00640 l4_vm_vmx_field_ptr(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field) L4_NOTHROW
00641 {
00642     unsigned offset = l4_vm_vmx_field_offset(vmcs, field);
00643     if (offset == ~0U)
00644         return 0;
00645
00646     return (void *) (vmcs->values + offset);
00647 }
00648
00654 L4_INLINE
00655 void *
00656 l4_vm_vmx_field_ptr_offset(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field,
00657                           unsigned *offset) L4_NOTHROW
00658 {
00659     *offset = l4_vm_vmx_field_offset(vmcs, field);
00660     if (*offset == ~0U)
00661         return 0;
00662
00663     return (void *) (vmcs->values + *offset);
00664 }
00665
00671 L4_INLINE
00672 void
00673 l4_vm_vmx_offset_dirty(l4_vm_vmx_vcpu_vmcs_t *vmcs,
00674                      unsigned offset) L4_NOTHROW
00675 {
00676     vmcs->dirty_bitmap[offset / 8] |= 1U << (offset % 8);
00677 }
00678
00683 L4_INLINE
00684 void
00685 l4_vm_vmx_copy_values(l4_vm_vmx_vcpu_vmcs_t const *vmcs, l4_uint8_t *_dst,
00686                     l4_uint8_t const *_src) L4_NOTHROW
00687 {
00688     unsigned base_offset = vmcs->offset_table.base_offset * 64;
00689     unsigned size = vmcs->offset_table.size * 64;
00690
00691     void *dst = _dst + base_offset;
00692     void const *src = _src + base_offset;
00693     __builtin_memcpy(dst, src, size);
00694 }
00695
00696 L4_INLINE
00697 void
00698 l4_vm_vmx_clear(l4_vm_vmx_vcpu_vmcs_t *vmcs,
00699               l4_vm_vmx_vcpu_vmcs_t *dest_vmcs) L4_NOTHROW
00700 {
00701     l4_vm_vmx_vcpu_vmcs_t **current_vmcs_ptr
00702         = (l4_vm_vmx_vcpu_vmcs_t **)&vmcs->user_data;
00703
00704     if (*current_vmcs_ptr != dest_vmcs)
00705         return;
00706
00707     l4_vm_vmx_set_hw_vmcs(dest_vmcs, l4_vm_vmx_get_hw_vmcs(vmcs));
00708     l4_vm_vmx_copy_values(vmcs, dest_vmcs->values, vmcs->values);
00709
00710     /* Due to its size, the dirty bitmap is always copied in its entirety. */
00711     __builtin_memcpy(dest_vmcs->dirty_bitmap, vmcs->dirty_bitmap,

```



```

00712     L4_VM_VMX_VMCS_SIZE_DIRTY_BITMAP);
00713
00714     *current_vmcs_ptr = 0;
00715 }
00716
00717 L4_INLINE
00718 void
00719 l4_vm_vmx_ptr_load(l4_vm_vmx_vcpu_vmcs_t *vmcs,
00720                   l4_vm_vmx_vcpu_vmcs_t *src_vmcs) L4_NOTHROW
00721 {
00722     l4_vm_vmx_vcpu_vmcs_t **current_vmcs_ptr
00723     = (l4_vm_vmx_vcpu_vmcs_t **)&vmcs->user_data;
00724
00725     if (*current_vmcs_ptr == src_vmcs)
00726         return;
00727
00728     if (*current_vmcs_ptr && *current_vmcs_ptr != src_vmcs)
00729         l4_vm_vmx_clear(vmcs, *current_vmcs_ptr);
00730
00731     *current_vmcs_ptr = src_vmcs;
00732
00733     l4_vm_vmx_set_hw_vmcs(vmcs, l4_vm_vmx_get_hw_vmcs(src_vmcs));
00734     l4_vm_vmx_copy_values(vmcs, vmcs->values, src_vmcs->values);
00735
00736     /* Due to its size, the dirty bitmap is always compiled in its entirety. */
00737     __builtin_memcpy(vmcs->dirty_bitmap, src_vmcs->dirty_bitmap,
00738                     L4_VM_VMX_VMCS_SIZE_DIRTY_BITMAP);
00739 }
00740
00741 L4_INLINE
00742 l4_umword_t
00743 l4_vm_vmx_read_nat(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field) L4_NOTHROW
00744 {
00745     l4_umword_t *ptr = (l4_umword_t *)l4_vm_vmx_field_ptr(vmcs, field);
00746     if (!ptr)
00747         return 0;
00748
00749     return *ptr;
00750 }
00751
00752 L4_INLINE
00753 l4_uint16_t
00754 l4_vm_vmx_read_16(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field) L4_NOTHROW
00755 {
00756     l4_uint16_t *ptr = (l4_uint16_t *)l4_vm_vmx_field_ptr(vmcs, field);
00757     if (!ptr)
00758         return 0;
00759
00760     return *ptr;
00761 }
00762
00763 L4_INLINE
00764 l4_uint32_t
00765 l4_vm_vmx_read_32(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field) L4_NOTHROW
00766 {
00767     l4_uint32_t *ptr = (l4_uint32_t *)l4_vm_vmx_field_ptr(vmcs, field);
00768     if (!ptr)
00769         return 0;
00770
00771     return *ptr;
00772 }
00773
00774 L4_INLINE
00775 l4_uint64_t
00776 l4_vm_vmx_read_64(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field) L4_NOTHROW
00777 {
00778     l4_uint64_t *ptr = (l4_uint64_t *)l4_vm_vmx_field_ptr(vmcs, field);
00779     if (!ptr)
00780         return 0;
00781
00782     return *ptr;
00783 }
00784
00785 L4_INLINE
00786 l4_uint64_t
00787 l4_vm_vmx_read(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field) L4_NOTHROW
00788 {
00789     unsigned size = (field >> 13) & 0x03U;
00790
00791     switch (size)
00792     {
00793     case 0: return l4_vm_vmx_read_16(vmcs, field);
00794     case 1: return l4_vm_vmx_read_64(vmcs, field);
00795     case 2: return l4_vm_vmx_read_32(vmcs, field);
00796     case 3: return l4_vm_vmx_read_nat(vmcs, field);
00797     }
00798 }

```

```

00799  __builtin_trap();
00800 }
00801
00802 L4_INLINE
00803 void
00804 l4_vm_vmx_write_nat(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field,
00805                    l4_umword_t val) L4_NOTHROW
00806 {
00807     unsigned offset;
00808     l4_umword_t *ptr
00809         = (l4_umword_t *)l4_vm_vmx_field_ptr_offset(vmcs, field, &offset);
00810
00811     if ((ptr) && (*ptr != val))
00812     {
00813         *ptr = val;
00814         l4_vm_vmx_offset_dirty(vmcs, offset);
00815     }
00816 }
00817
00818 L4_INLINE
00819 void
00820 l4_vm_vmx_write_16(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field,
00821                   l4_uint16_t val) L4_NOTHROW
00822 {
00823     unsigned offset;
00824     l4_uint16_t *ptr
00825         = (l4_uint16_t *)l4_vm_vmx_field_ptr_offset(vmcs, field, &offset);
00826
00827     if ((ptr) && (*ptr != val))
00828     {
00829         *ptr = val;
00830         l4_vm_vmx_offset_dirty(vmcs, offset);
00831     }
00832 }
00833
00834 L4_INLINE
00835 void
00836 l4_vm_vmx_write_32(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field,
00837                   l4_uint32_t val) L4_NOTHROW
00838 {
00839     unsigned offset;
00840     l4_uint32_t *ptr
00841         = (l4_uint32_t *)l4_vm_vmx_field_ptr_offset(vmcs, field, &offset);
00842
00843     if ((ptr) && (*ptr != val))
00844     {
00845         *ptr = val;
00846         l4_vm_vmx_offset_dirty(vmcs, offset);
00847     }
00848 }
00849
00850 L4_INLINE
00851 void
00852 l4_vm_vmx_write_64(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field,
00853                   l4_uint64_t val) L4_NOTHROW
00854 {
00855     unsigned offset;
00856     l4_uint64_t *ptr
00857         = (l4_uint64_t *)l4_vm_vmx_field_ptr_offset(vmcs, field, &offset);
00858
00859     if ((ptr) && (*ptr != val))
00860     {
00861         *ptr = val;
00862         l4_vm_vmx_offset_dirty(vmcs, offset);
00863     }
00864 }
00865
00866 L4_INLINE
00867 void
00868 l4_vm_vmx_write(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field,
00869                 l4_uint64_t val) L4_NOTHROW
00870 {
00871     unsigned size = (field >> 13) & 0x03U;
00872
00873     switch (size)
00874     {
00875     case 0: l4_vm_vmx_write_16(vmcs, field, val); break;
00876     case 1: l4_vm_vmx_write_64(vmcs, field, val); break;
00877     case 2: l4_vm_vmx_write_32(vmcs, field, val); break;
00878     case 3: l4_vm_vmx_write_nat(vmcs, field, val); break;
00879     }
00880 }
00881
00882 L4_INLINE
00883 l4_uint64_t
00884 l4_vm_vmx_get_caps(l4_vm_vmx_vcpu_state_t const *vcpu_state,
00885                   enum L4_vm_vmx_caps_regs caps_reg) L4_NOTHROW

```

```

00886 {
00887     return vcpu_state->infos.caps[caps_reg];
00888 }
00889
00890 L4_INLINE
00891 l4_uint32_t
00892 l4_vm_vmx_get_caps_default1(l4_vm_vmx_vcpu_state_t const *vcpu_state,
00893                             enum l4_vm_vmx_dfll_regs dfll_reg) L4_NOTHROW
00894 {
00895     return vcpu_state->infos.dfll[dfll_reg];
00896 }
00897
00898 L4_INLINE
00899 l4_uint32_t
00900 l4_vm_vmx_get_cr2_index(l4_vm_vmx_vcpu_vmcs_t const *vmcs) L4_NOTHROW
00901 {
00902     return vmcs->cr2_index;
00903 }
00904
00905 L4_INLINE
00906 void
00907 l4_vm_vmx_set_hw_vmcs(l4_vm_vmx_vcpu_vmcs_t *vmcs,
00908                       l4_cap_idx_t vmcs_cap) L4_NOTHROW
00909 {
00910     vmcs->vmcs = vmcs_cap;
00911 }
00912
00913 L4_INLINE
00914 l4_cap_idx_t
00915 l4_vm_vmx_get_hw_vmcs(l4_vm_vmx_vcpu_vmcs_t *vmcs) L4_NOTHROW
00916 {
00917     return vmcs->vmcs & L4_CAP_MASK;
00918 }

```

## 17.450 l4/sys/arm\_smccc File Reference

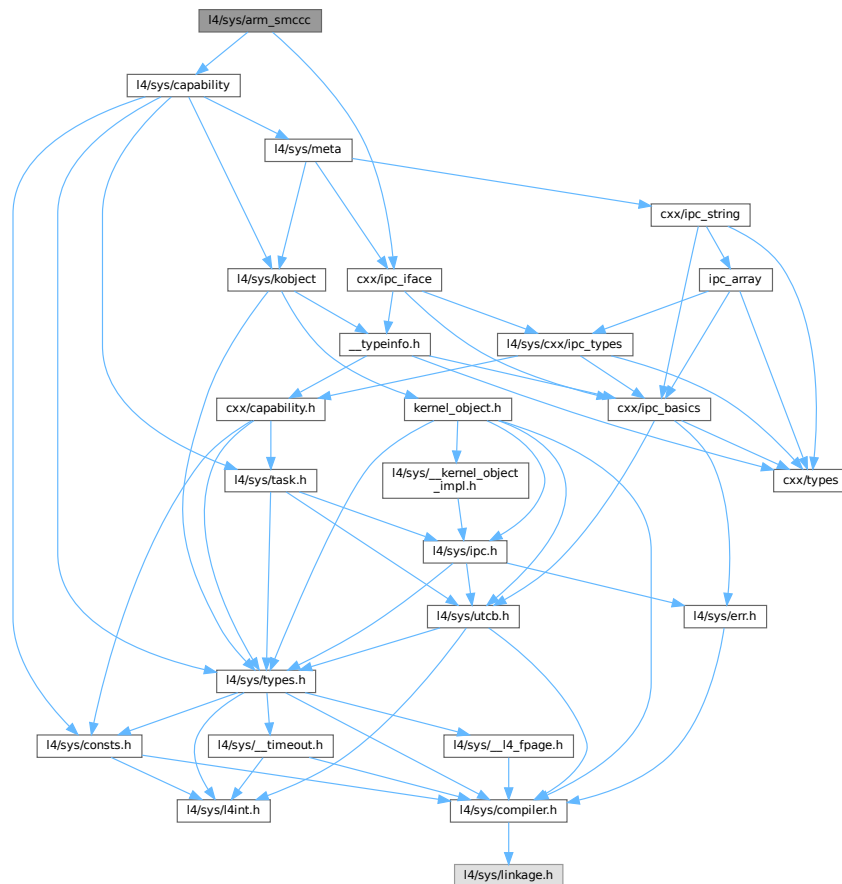
ARM secure monitor call functions.

```

#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_iface>

```

Include dependency graph for `arm_smccc`:



## Data Structures

- class [L4::Arm\\_smccc](#)

*Wrapper for function calls that follow the ARM SMC/HVC calling convention.*

## Namespaces

- namespace [L4](#)

*[L4](#) low-level kernel interface.*

## 17.450.1 Detailed Description

ARM secure monitor call functions.

Definition in file [arm\\_smccc](#).

## 17.451 arm\_smccc

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2018, 2022, 2024 Kernkonzept GmbH.
00004  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00012 #pragma once
00013
00014 #include <l4/sys/capability>
00015 #include <l4/sys/cxx/ipc_iface>
00016
00017 namespace L4 {
00018
00023 class L4_EXPORT Arm_smccc : public Kobject_0t<Arm_smccc, L4_PROTO_SMCCC>
00024 {
00025 public:
00064     L4_INLINE_RPC(l4_msgtag_t, call,
00065                   (l4_umword_t func, l4_umword_t in0, l4_umword_t in1,
00066                    l4_umword_t in2, l4_umword_t in3, l4_umword_t in4,
00067                    l4_umword_t in5, l4_umword_t *out0, l4_umword_t *out1,
00068                    l4_umword_t *out2, l4_umword_t *out3,
00069                    l4_umword_t client_id));
00070
00071     typedef L4::Typeid::Rpc_nocode<call_t> Rpccs;
00072 };
00073
00074 }

```

## 17.452 l4/sys/arm\_smccc.h File Reference

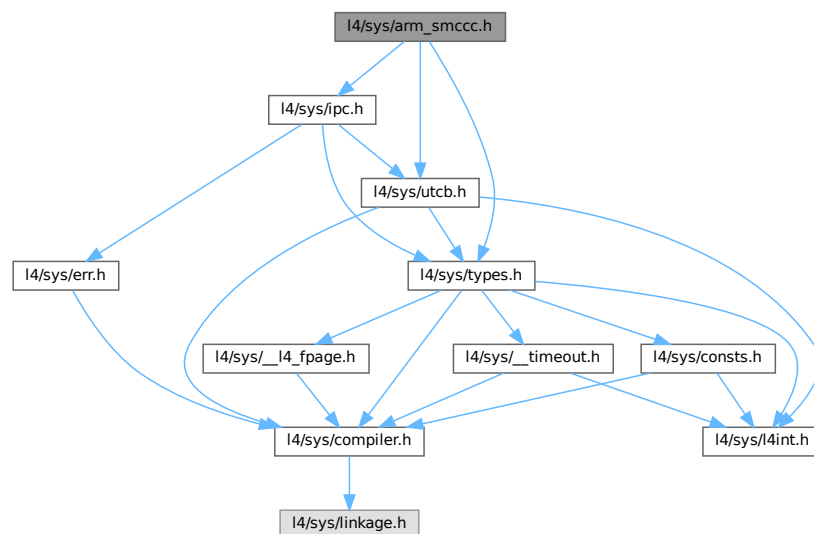
ARM secure monitor call functions.

```

#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>

```

Include dependency graph for arm\_smccc.h:



## Functions

- `l4_msgtag_t l4_arm_smccc_call (l4_cap_idx_t pfc, l4_umword_t func, l4_umword_t in0, l4_umword_t in1, l4_umword_t in2, l4_umword_t in3, l4_umword_t in4, l4_umword_t in5, l4_umword_t *out0, l4_umword_t *out1, l4_umword_t *out2, l4_umword_t *out3, l4_umword_t client_id) L4_NOTHROW`

C interface for calling the ARM secure monitor, see [L4::Arm\\_smccc::call\(\)](#) for the C++ interface.

### 17.452.1 Detailed Description

ARM secure monitor call functions.

Definition in file [arm\\_smccc.h](#).

### 17.452.2 Function Documentation

#### 17.452.2.1 l4\_arm\_smccc\_call()

```
l4_msgtag_t l4_arm_smccc_call (
    l4_cap_idx_t pfc,
    l4_umword_t func,
    l4_umword_t in0,
    l4_umword_t in1,
    l4_umword_t in2,
    l4_umword_t in3,
    l4_umword_t in4,
    l4_umword_t in5,
    l4_umword_t * out0,
    l4_umword_t * out1,
    l4_umword_t * out2,
    l4_umword_t * out3,
    l4_umword_t client_id) [inline]
```

C interface for calling the ARM secure monitor, see [L4::Arm\\_smccc::call\(\)](#) for the C++ interface.

## Parameters

	<i>pfc</i>	Capability of the SMC kernel object. The input parameters consist of a function identifier, 6 arguments and a client id. Results are returned in 4 output parameters.
	<i>func</i>	Function identifier. <ul style="list-style-type: none"> <li>• Bit 31 has to be set: This marks the call as <i>Fast Call</i>. <i>Yielding Calls</i> (bit 31 unset) are rejected by the kernel.</li> <li>• Bit 30 defines the calling convention:</li> <li>• Bit 30 == 1: 64-bit calling convention.</li> <li>• Bit 30 == 0: 32-bit calling convention.</li> <li>• Bits 24..29 determine the service call ID. The permitted IDs are set in the kernel configuration. By default only service IDs &gt;= 0x30000000 (<i>Trusted Application Calls</i> and <i>Trusted OS Calls</i>) are allowed.</li> </ul>

in	<i>in0</i>	First input parameter.
in	<i>in1</i>	Second input parameter.
in	<i>in2</i>	Third input parameter.
in	<i>in3</i>	Fourth input parameter.
in	<i>in4</i>	Fifth input parameter.
in	<i>in5</i>	Sixth input parameter.
out	<i>out0</i>	First output parameter.
out	<i>out1</i>	Second output parameter.
out	<i>out2</i>	Third output parameter.
out	<i>out3</i>	Fourth output parameter.
in	<i>client_id</i>	Client ID. According to the specification, this value might be ignored by certain functions.

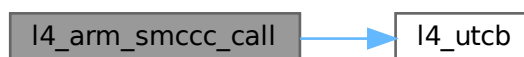
### Return values

<code>-L4_ENOSYS</code>	Either bit 31 of the function call not set or service ID outside the range permitted by kernel configuration.
<code>-L4_EINVAL</code>	Invalid number of parameters.
<code>&lt; 0</code>	Other <a href="#">L4</a> error.
<code>0</code>	Success.

Definition at line 42 of file [arm\\_smccc.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



## 17.453 arm\_smccc.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (C) 2018, 2022, 2024 Kernkonzept GmbH.
00003  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00011 #pragma once
00012
00013 #include <l4/sys/types.h>
00014 #include <l4/sys/utcb.h>

```

```

00015
00016 L4_INLINE l4_msgtag_t
00017 l4_arm_smccc_call(l4_cap_idx_t pfc, l4_umword_t func, l4_umword_t in0,
00018                  l4_umword_t in1, l4_umword_t in2, l4_umword_t in3,
00019                  l4_umword_t in4, l4_umword_t in5, l4_umword_t *out0,
00020                  l4_umword_t *out1, l4_umword_t *out2, l4_umword_t *out3,
00021                  l4_umword_t client_id) L4_NOTHROW;
00022
00023 L4_INLINE l4_msgtag_t
00024 l4_arm_smccc_call_u(l4_cap_idx_t pfc, l4_umword_t func, l4_umword_t in0,
00025                    l4_umword_t in1, l4_umword_t in2, l4_umword_t in3,
00026                    l4_umword_t in4, l4_umword_t in5, l4_umword_t *out0,
00027                    l4_umword_t *out1, l4_umword_t *out2, l4_umword_t *out3,
00028                    l4_umword_t client_id, l4_utcb_t *utcb) L4_NOTHROW;
00029
00030 /* IMPLEMENTATION ----- */
00031
00032 #include <l4/sys/ipc.h>
00033
00041 L4_INLINE l4_msgtag_t
00042 l4_arm_smccc_call(l4_cap_idx_t pfc, l4_umword_t func,
00043                  l4_umword_t in0, l4_umword_t in1,
00044                  l4_umword_t in2, l4_umword_t in3,
00045                  l4_umword_t in4, l4_umword_t in5,
00046                  l4_umword_t *out0, l4_umword_t *out1,
00047                  l4_umword_t *out2, l4_umword_t *out3,
00048                  l4_umword_t client_id) L4_NOTHROW
00049 {
00050     return l4_arm_smccc_call_u(pfc, func, in0, in1, in2, in3, in4, in5,
00051                                out0, out1, out2, out3, client_id, l4_utcb());
00052 }
00053
00054
00055 L4_INLINE l4_msgtag_t
00056 l4_arm_smccc_call_u(l4_cap_idx_t pfc, l4_umword_t func, l4_umword_t in0,
00057                    l4_umword_t in1, l4_umword_t in2, l4_umword_t in3,
00058                    l4_umword_t in4, l4_umword_t in5, l4_umword_t *out0,
00059                    l4_umword_t *out1, l4_umword_t *out2, l4_umword_t *out3,
00060                    l4_umword_t client_id, l4_utcb_t *utcb) L4_NOTHROW
00061 {
00062     l4_msgtag_t ret;
00063     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00064     v->mr[0] = func;
00065     v->mr[1] = in0;
00066     v->mr[2] = in1;
00067     v->mr[3] = in2;
00068     v->mr[4] = in3;
00069     v->mr[5] = in4;
00070     v->mr[6] = in5;
00071     v->mr[7] = client_id;
00072
00073     ret = l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_SMCCC, 8, 0, 0),
00074                     L4_IPC_NEVER);
00075
00076     if (l4_error(ret) >= 0)
00077     {
00078         *out0 = v->mr[0];
00079         *out1 = v->mr[1];
00080         *out2 = v->mr[2];
00081         *out3 = v->mr[3];
00082     }
00083
00084     return ret;
00085 }

```

## 17.454 amd64/l4/sys/cache.h File Reference

Cache functions.

### Functions

- int [l4\\_cache\\_clean\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
Cache clean a range in D-cache; writes back to PoC.
- int [l4\\_cache\\_flush\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
Cache flush a range; writes back to PoC.



- int [l4\\_cache\\_inv\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache invalidate a range; might write back to PoC.*
- int [l4\\_cache\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent between I-cache and D-cache; writes back to PoU.*
- int [l4\\_cache\\_dma\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent for use with external memory; writes back to PoC.*
- int [l4\\_cache\\_dma\\_coherent\\_full](#) (void) [L4\\_NOTHROW](#)  
*Make memory coherent for use with external memory; writes back to PoC.*

### 17.454.1 Detailed Description

Cache functions.

Definition in file [cache.h](#).

## 17.455 cache.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *      economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #ifndef __L4SYS__INCLUDE__ARCH_AMD64__CACHE_H__
00012 #define __L4SYS__INCLUDE__ARCH_AMD64__CACHE_H__
00013
00014 #include_next <l4/sys/cache.h>
00015
00016 L4_INLINE int
00017 l4_cache_clean_data(unsigned long start,
00018                    unsigned long end) L4_NOTHROW
00019 {
00020     (void)start; (void)end;
00021     return 0;
00022 }
00023
00024 L4_INLINE int
00025 l4_cache_flush_data(unsigned long start,
00026                    unsigned long end) L4_NOTHROW
00027 {
00028     (void)start; (void)end;
00029     return 0;
00030 }
00031
00032 L4_INLINE int
00033 l4_cache_inv_data(unsigned long start,
00034                  unsigned long end) L4_NOTHROW
00035 {
00036     (void)start; (void)end;
00037     return 0;
00038 }
00039
00040 L4_INLINE int
00041 l4_cache_coherent(unsigned long start,
00042                  unsigned long end) L4_NOTHROW
00043 {
00044     (void)start; (void)end;
00045     return 0;
00046 }
00047
00048 L4_INLINE int
00049 l4_cache_dma_coherent(unsigned long start,
00050                      unsigned long end) L4_NOTHROW
00051 {
00052     (void)start; (void)end;
00053     return 0;
00054 }
00055

```

```

00056 L4_INLINE int
00057 l4_cache_dma_coherent_full(void) L4_NOTHROW
00058 {
00059     return 0;
00060 }
00061
00062 #endif /* ! __L4SYS__INCLUDE__ARCH_AMD64__CACHE_H__ */

```

## 17.456 arm/l4/sys/cache.h File Reference

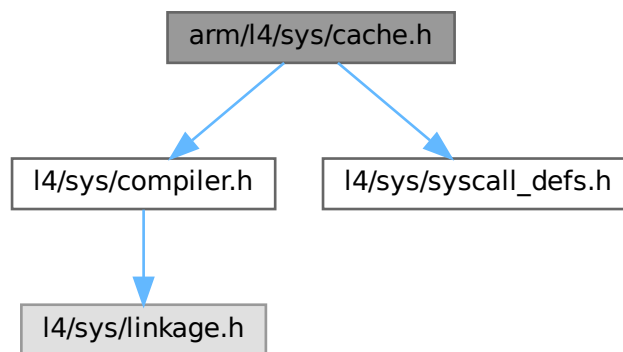
Cache functions.

```

#include <l4/sys/compiler.h>
#include <l4/sys/syscall_defs.h>

```

Include dependency graph for cache.h:



### Functions

- `int l4_cache_clean_data` (unsigned long start, unsigned long end) `L4_NOTHROW`  
Cache clean a range in D-cache; writes back to PoC.
- `int l4_cache_flush_data` (unsigned long start, unsigned long end) `L4_NOTHROW`  
Cache flush a range; writes back to PoC.
- `int l4_cache_inv_data` (unsigned long start, unsigned long end) `L4_NOTHROW`  
Cache invalidate a range; might write back to PoC.
- `int l4_cache_coherent` (unsigned long start, unsigned long end) `L4_NOTHROW`  
Make memory coherent between I-cache and D-cache; writes back to PoU.
- `int l4_cache_dma_coherent` (unsigned long start, unsigned long end) `L4_NOTHROW`  
Make memory coherent for use with external memory; writes back to PoC.
- `int l4_cache_dma_coherent_full` (void) `L4_NOTHROW`  
Make memory coherent for use with external memory; writes back to PoC.

## 17.456.1 Detailed Description

Cache functions.

Date

2007-11

Author

Adam Lackorzynski [adam@os.inf.tu-dresden.de](mailto:adam@os.inf.tu-dresden.de)

Definition in file [cache.h](#).

## 17.457 cache.h

[Go to the documentation of this file.](#)

```

00001
00009 /*
00010  * (c) 2007-2009 Author(s)
00011  *      economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * License: see LICENSE.spdx (in this directory or the directories above)
00014  */
00015 #ifndef __L4SYS__INCLUDE__ARCH_ARM__CACHE_H__
00016 #define __L4SYS__INCLUDE__ARCH_ARM__CACHE_H__
00017
00018 #include <l4/sys/compiler.h>
00019 #include <l4/sys/syscall_defs.h>
00020
00021 #include_next <l4/sys/cache.h>
00022
00026 L4_INLINE void
00027 l4_cache_op_arm_call(unsigned long op,
00028                     unsigned long start,
00029                     unsigned long end);
00030
00031 L4_INLINE void
00032 l4_cache_op_arm_call(unsigned long op,
00033                     unsigned long start,
00034                     unsigned long end)
00035 {
00036     register unsigned long _op    __asm__ ("r0") = op;
00037     register unsigned long _start __asm__ ("r1") = start;
00038     register unsigned long _end   __asm__ ("r2") = end;
00039
00040     __asm__ __volatile__
00041     ("@ l4_cache_op_arm_call(start) \n\t"
00042      "mov    r5, %[sc] \n\t"
00043      "blx    __l4_sys_syscall \n\t"
00044      "@ l4_cache_op_arm_call(end) \n\t"
00045      :
00046      "=r" (_op),
00047      "=r" (_start),
00048      "=r" (_end)
00049      :
00050      [sc] "i" (L4_SYSCALL_MEM_OP),
00051      "0" (_op),
00052      "1" (_start),
00053      "2" (_end)
00054      :
00055      "cc", "memory", "r5", "ip", "lr"
00056      );
00057 }
00058
00059 enum L4_mem_cache_ops
00060 {
00061     L4_MEM_CACHE_OP_CLEAN_DATA      = 0,
00062     L4_MEM_CACHE_OP_FLUSH_DATA     = 1,
00063     L4_MEM_CACHE_OP_INV_DATA       = 2,
00064     L4_MEM_CACHE_OP_COHERENT       = 3,
00065     L4_MEM_CACHE_OP_DMA_COHERENT   = 4,

```

```

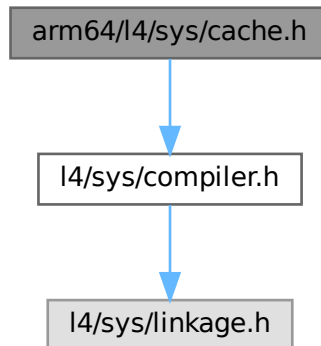
00066     L4_MEM_CACHE_OP_DMA_COHERENT_FULL = 5,
00067 };
00068
00069 L4_INLINE int
00070 l4_cache_clean_data(unsigned long start,
00071                    unsigned long end) L4_NOTHROW
00072 {
00073     l4_cache_op_arm_call(L4_MEM_CACHE_OP_CLEAN_DATA, start, end);
00074     return 0;
00075 }
00076
00077 L4_INLINE int
00078 l4_cache_flush_data(unsigned long start,
00079                    unsigned long end) L4_NOTHROW
00080 {
00081     l4_cache_op_arm_call(L4_MEM_CACHE_OP_FLUSH_DATA, start, end);
00082     return 0;
00083 }
00084
00085 L4_INLINE int
00086 l4_cache_inv_data(unsigned long start,
00087                  unsigned long end) L4_NOTHROW
00088 {
00089     l4_cache_op_arm_call(L4_MEM_CACHE_OP_INV_DATA, start, end);
00090     return 0;
00091 }
00092
00093 L4_INLINE int
00094 l4_cache_coherent(unsigned long start,
00095                  unsigned long end) L4_NOTHROW
00096 {
00097     l4_cache_op_arm_call(L4_MEM_CACHE_OP_COHERENT, start, end);
00098     return 0;
00099 }
00100
00101 L4_INLINE int
00102 l4_cache_dma_coherent(unsigned long start,
00103                      unsigned long end) L4_NOTHROW
00104 {
00105     l4_cache_op_arm_call(L4_MEM_CACHE_OP_DMA_COHERENT, start, end);
00106     return 0;
00107 }
00108
00109 L4_INLINE int
00110 l4_cache_dma_coherent_full(void) L4_NOTHROW
00111 {
00112     l4_cache_op_arm_call(L4_MEM_CACHE_OP_DMA_COHERENT_FULL, 0, 0);
00113     return 0;
00114 }
00115
00116 #endif /* ! __L4SYS__INCLUDE__ARCH_ARM__CACHE_H__ */

```

## 17.458 arm64/l4/sys/cache.h File Reference

Cache functions.

```
#include <l4/sys/compiler.h>  
Include dependency graph for cache.h:
```



## Functions

- int [l4\\_cache\\_clean\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache clean a range in D-cache; writes back to PoC.*
- int [l4\\_cache\\_flush\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache flush a range; writes back to PoC.*
- int [l4\\_cache\\_inv\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache invalidate a range; might write back to PoC.*
- int [l4\\_cache\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent between I-cache and D-cache; writes back to PoU.*
- int [l4\\_cache\\_dma\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent for use with external memory; writes back to PoC.*

### 17.458.1 Detailed Description

Cache functions.

#### Date

2007-11

#### Author

Adam Lackorzynski [adam@os.inf.tu-dresden.de](mailto:adam@os.inf.tu-dresden.de)

Definition in file [cache.h](#).

## 17.459 cache.h

[Go to the documentation of this file.](#)

```

00001
00009 /*
00010  * (c) 2007-2009 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * License: see LICENSE.spdx (in this directory or the directories above)
00014  */
00015 #ifndef __L4SYS__INCLUDE__ARCH_ARM__CACHE_H__
00016 #define __L4SYS__INCLUDE__ARCH_ARM__CACHE_H__
00017
00018 #include <l4/sys/compiler.h>
00019
00020 #include_next <l4/sys/cache.h>
00021
00022 L4_INLINE unsigned long __attribute__((pure, always_inline))
00023 l4_cache_arm_ctr(void);
00024
00025 L4_INLINE unsigned long __attribute__((pure, always_inline))
00026 l4_cache_arm_ctr(void)
00027 {
00028     unsigned long v;
00029     asm ("mrs %0, CTR_EL0" : "=r"(v));
00030     return v;
00031 }
00032
00033 L4_INLINE unsigned __attribute__((pure, always_inline))
00034 l4_cache_dmin_line(void);
00035
00036 L4_INLINE unsigned __attribute__((pure, always_inline))
00037 l4_cache_dmin_line(void)
00038 {
00039     return 4U < ((l4_cache_arm_ctr() > 16) & 0xf);
00040 }
00041
00042 #define L4_ARM_CACHE_LOOP(op)
00043     unsigned long step;
00044
00045     if (start > end)
00046         __builtin_unreachable();
00047
00048     step = l4_cache_dmin_line();
00049     start &= ~(step - 1);
00050     end = (end + step - 1) & ~(step - 1);
00051     for (; start != end; start += step)
00052         asm volatile (op " ", %0" : : "r"(start) : "memory");
00053     asm volatile ("dsb ish");
00054
00055
00056 L4_INLINE int
00057 l4_cache_clean_data(unsigned long start,
00058                     unsigned long end) L4_NOTHROW
00059 {
00060     L4_ARM_CACHE_LOOP("dc cvac");
00061     return 0;
00062 }
00063
00064 L4_INLINE int
00065 l4_cache_flush_data(unsigned long start,
00066                     unsigned long end) L4_NOTHROW
00067 {
00068     L4_ARM_CACHE_LOOP("dc civac");
00069     return 0;
00070 }
00071
00072 L4_INLINE int
00073 l4_cache_inv_data(unsigned long start,
00074                   unsigned long end) L4_NOTHROW
00075 {
00076     // DC IVAC is always privileged, use DC CIVAC instead
00077     L4_ARM_CACHE_LOOP("dc civac");
00078     return 0;
00079 }
00080
00081 L4_INLINE int
00082 l4_cache_coherent(unsigned long start,
00083                   unsigned long end) L4_NOTHROW
00084 {
00085     L4_ARM_CACHE_LOOP("dc cvau, %0; ic ivau");
00086     asm volatile ("isb");
00087     return 0;
00088 }
00089

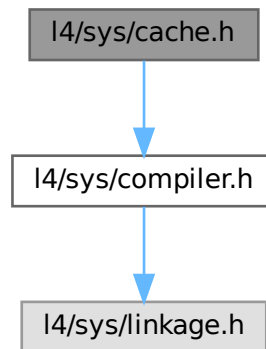
```

```
00090 L4_INLINE int
00091 l4_cache_dma_coherent(unsigned long start,
00092                       unsigned long end) L4_NOTHROW
00093 {
00094     L4_ARM_CACHE_LOOP("dc civac");
00095     return 0;
00096 }
00097
00098 #undef L4_ARM_CACHE_LOOP
00099
00100 #endif /* ! __L4SYS__INCLUDE__ARCH_ARM__CACHE_H__ */
```

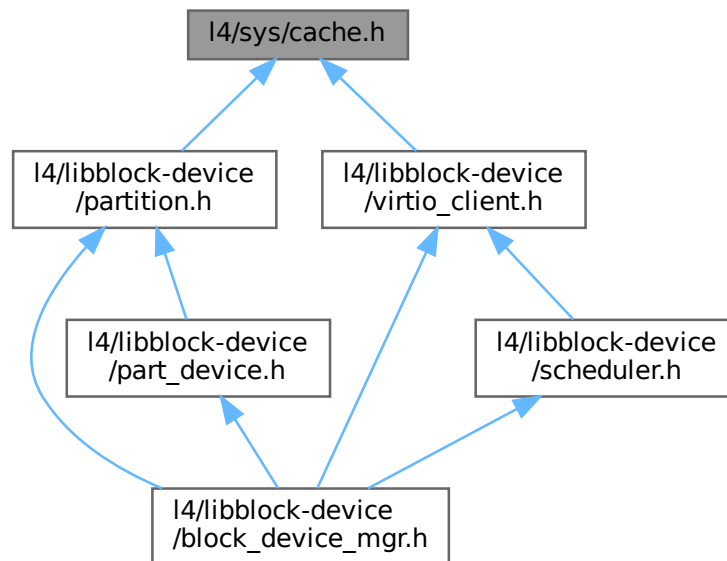
## 17.460 l4/sys/cache.h File Reference

Cache-consistency functions.

```
#include <l4/sys/compiler.h>
Include dependency graph for cache.h:
```



This graph shows which files directly or indirectly include this file:



## Functions

- [L4\\_BEGIN\\_DECLS](#) [int l4\\_cache\\_clean\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache clean a range in D-cache; writes back to PoC.*
- [int l4\\_cache\\_flush\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache flush a range; writes back to PoC.*
- [int l4\\_cache\\_inv\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache invalidate a range; might write back to PoC.*
- [int l4\\_cache\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent between I-cache and D-cache; writes back to PoU.*
- [int l4\\_cache\\_dma\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent for use with external memory; writes back to PoC.*
- [int l4\\_cache\\_dma\\_coherent\\_full](#) (void) [L4\\_NOTHROW](#)  
*Make memory coherent for use with external memory; writes back to PoC.*

## 17.460.1 Detailed Description

Cache-consistency functions.

Date

2007-11

Author

Adam Lackorzynski [adam@os.inf.tu-dresden.de](mailto:adam@os.inf.tu-dresden.de)

Definition in file [cache.h](#).



## 17.461 cache.h

[Go to the documentation of this file.](#)

```

00001
00010 /*
00011  * (c) 2007-2009 Author(s)
00012  *      economic rights: Technische Universität Dresden (Germany)
00013  *
00014  * License: see LICENSE.spdx (in this directory or the directories above)
00015  */
00016
00017 #ifndef __L4SYS__INCLUDE__CACHE_H__
00018 #define __L4SYS__INCLUDE__CACHE_H__
00019
00020 #include <14/sys/compiler.h>
00021
00036
00037 L4_BEGIN_DECLS
00038
00053 L4_INLINE int
00054 l4_cache_clean_data(unsigned long start,
00055                    unsigned long end) L4_NOTHROW;
00056
00071 L4_INLINE int
00072 l4_cache_flush_data(unsigned long start,
00073                    unsigned long end) L4_NOTHROW;
00074
00093 L4_INLINE int
00094 l4_cache_inv_data(unsigned long start,
00095                  unsigned long end) L4_NOTHROW;
00096
00108 L4_INLINE int
00109 l4_cache_coherent(unsigned long start,
00110                  unsigned long end) L4_NOTHROW;
00111
00123 L4_INLINE int
00124 l4_cache_dma_coherent(unsigned long start,
00125                      unsigned long end) L4_NOTHROW;
00126
00127 #if !defined(ARCH_arm64)
00132 L4_INLINE int
00133 l4_cache_dma_coherent_full(void) L4_NOTHROW;
00134 #endif
00135
00136 L4_END_DECLS
00137
00138 #endif /* ! __L4SYS__INCLUDE__CACHE_H__ */

```

## 17.462 x86/I4/sys/cache.h File Reference

Cache functions.

### Functions

- int [l4\\_cache\\_clean\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache clean a range in D-cache; writes back to PoC.*
- int [l4\\_cache\\_flush\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache flush a range; writes back to PoC.*
- int [l4\\_cache\\_inv\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache invalidate a range; might write back to PoC.*
- int [l4\\_cache\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent between I-cache and D-cache; writes back to PoU.*
- int [l4\\_cache\\_dma\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent for use with external memory; writes back to PoC.*
- int [l4\\_cache\\_dma\\_coherent\\_full](#) (void) [L4\\_NOTHROW](#)  
*Make memory coherent for use with external memory; writes back to PoC.*

## 17.462.1 Detailed Description

Cache functions.

Definition in file [cache.h](#).

## 17.463 cache.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *      economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #ifndef __L4SYS__INCLUDE__ARCH_X86__CACHE_H__
00012 #define __L4SYS__INCLUDE__ARCH_X86__CACHE_H__
00013
00014 #include_next <l4/sys/cache.h>
00015
00016 L4_INLINE int
00017 l4_cache_clean_data(unsigned long start,
00018                    unsigned long end) L4_NOTHROW
00019 {
00020     (void)start; (void)end;
00021     return 0;
00022 }
00023
00024 L4_INLINE int
00025 l4_cache_flush_data(unsigned long start,
00026                    unsigned long end) L4_NOTHROW
00027 {
00028     (void)start; (void)end;
00029     return 0;
00030 }
00031
00032 L4_INLINE int
00033 l4_cache_inv_data(unsigned long start,
00034                  unsigned long end) L4_NOTHROW
00035 {
00036     (void)start; (void)end;
00037     return 0;
00038 }
00039
00040 L4_INLINE int
00041 l4_cache_coherent(unsigned long start,
00042                  unsigned long end) L4_NOTHROW
00043 {
00044     (void)start; (void)end;
00045     return 0;
00046 }
00047
00048 L4_INLINE int
00049 l4_cache_dma_coherent(unsigned long start,
00050                      unsigned long end) L4_NOTHROW
00051 {
00052     (void)start; (void)end;
00053     return 0;
00054 }
00055
00056 L4_INLINE int
00057 l4_cache_dma_coherent_full(void) L4_NOTHROW
00058 {
00059     return 0;
00060 }
00061
00062 #endif /* ! __L4SYS__INCLUDE__ARCH_X86__CACHE_H__ */

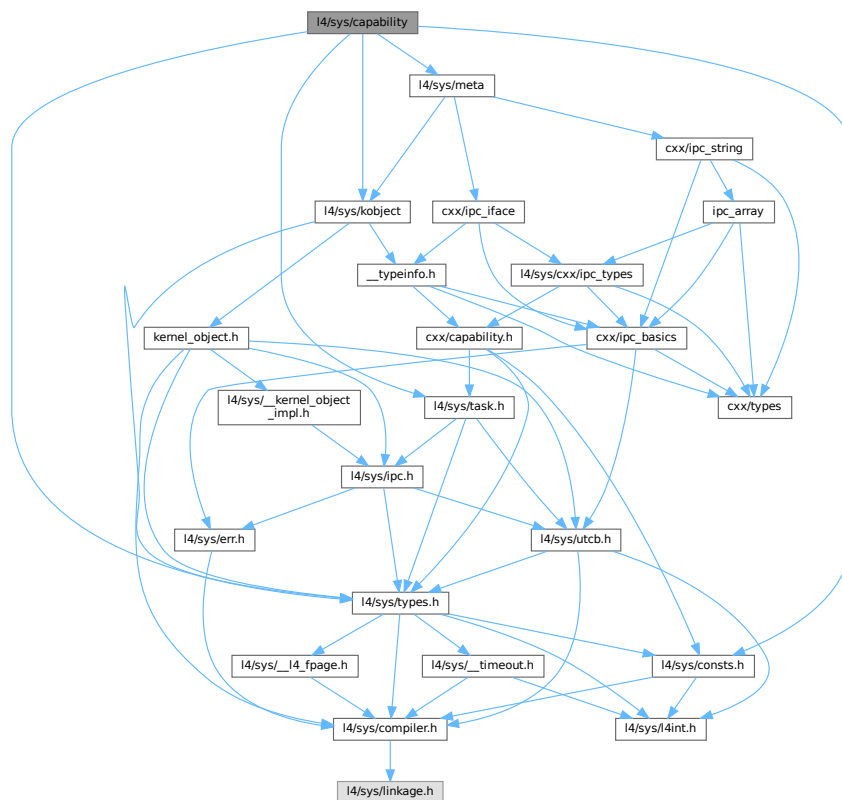
```

## 17.464 l4/sys/capability File Reference

[L4::Cap](#) related definitions.

```
#include <l4/sys/consts.h>
#include <l4/sys/types.h>
#include <l4/sys/kobject>
#include <l4/sys/task.h>
#include <l4/sys/meta>
```

Include dependency graph for capability:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*

## Macros

- `#define` [L4\\_DISABLE\\_COPY](#)(\_class)  
*Disable copy of a class.*

## Functions

- `template<typename T, typename F>`  
`Cap< T > L4::cap_dynamic_cast (Cap< F > const &c) noexcept`  
*dynamic\_cast for capabilities.*

### 17.464.1 Detailed Description

[L4::Cap](#) related definitions.

#### Author

Alexander Warg [alexander.warg@os.inf.tu-dresden.de](mailto:alexander.warg@os.inf.tu-dresden.de)

Definition in file [capability](#).

### 17.464.2 Macro Definition Documentation

#### 17.464.2.1 L4\_DISABLE\_COPY

```
#define L4_DISABLE_COPY(  
    _class)
```

#### Value:

```
public:
    _class(_class const &) = delete; \
    _class operator = (_class const &) = delete; \
private:
```

Disable copy of a class.

#### Parameters

<code>_class</code>	Name of the class that shall not have value copy semantics.
---------------------	-------------------------------------------------------------

The typical use of this is:

```
class Non_value
{
    L4_DISABLE_COPY(Non_value)

    ...
}
```

Definition at line 52 of file [capability](#).

## 17.465 capability

[Go to the documentation of this file.](#)

```

00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00009 /*
00010  * (c) 2008-2009,2015 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * License: see LICENSE.spdx (in this directory or the directories above)
00014  */
00015 #pragma once
00016
00017 #include <l4/sys/consts.h>
00018 #include <l4/sys/types.h>
00019 #include <l4/sys/kobject>
00020 #include <l4/sys/task.h>
00021
00022 namespace L4
00023 {
00024
00025 /* Forward declarations for our kernel object classes. */
00026 class Task;
00027 class Thread;
00028 class Thread_group;
00029 class Factory;
00030 class Irq;
00031 class Log;
00032 class Vm;
00033 class Vcpu_context;
00034 class Kobject;
00035
00051 #if __cplusplus >= 201103L
00052 # define L4_DISABLE_COPY(_class) \
00053     public: \
00054         _class(_class const &) = delete; \
00055         _class operator = (_class const &) = delete; \
00056     private:
00057 #else
00058 # define L4_DISABLE_COPY(_class) \
00059     private: \
00060         _class(_class const &); \
00061         _class operator = (_class const &);
00062 #endif
00063
00064
00065 #define L4_KOBJECT_DISABLE_COPY(_class) \
00066     protected: \
00067         _class(); \
00068         L4_DISABLE_COPY(_class)
00069
00070
00071 #define L4_KOBJECT(_class) L4_KOBJECT_DISABLE_COPY(_class)
00072
00073 inline l4_msgtag_t
00074 Cap_base::validate(Cap<Task> task, l4_utcb_t *u) const noexcept
00075 {
00076     return is_valid() ? l4_task_cap_valid_u(task.cap(), _c, u)
00077         : l4_msgtag(0, 0, 0, 0);
00078 }
00079
00080 inline l4_msgtag_t
00081 Cap_base::validate(l4_utcb_t *u) const noexcept
00082 {
00083     return is_valid() ? l4_task_cap_valid_u(L4_BASE_TASK_CAP, _c, u)
00084         : l4_msgtag(0, 0, 0, 0);
00085 }
00086
00087 }; // namespace L4
00088
00089 #include <l4/sys/meta>
00090
00091 namespace L4 {
00092
00114 template< typename T, typename F >
00115 inline
00116 Cap<T> cap_dynamic_cast(Cap<F> const &c) noexcept
00117 {
00118     if (!c.is_valid())
00119         return Cap<T>::Invalid;
00120
00121     Cap<Meta> mc = cap_reinterpret_cast<Meta>(c);
00122     Type_info const *m = kobject_typeid<T>();
00123     if (m->proto() && l4_error(mc->supports(m->proto())) > 0)
00124         return Cap<T>(c.cap());
00125

```

```

00126 // FIXME: use generic checker
00127 #if 0
00128 if (l4_error(mc->supports(T::kobject_proto())) > 0)
00129     return Cap<T>(c.cap());
00130 #endif
00131
00132 return Cap<T>::Invalid;
00133 }
00134
00135 }

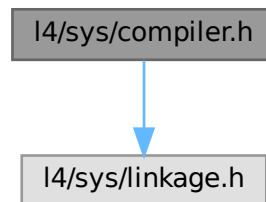
```

## 17.466 l4/sys/compiler.h File Reference

L4 compiler related defines.

```
#include <l4/sys/linkage.h>
```

Include dependency graph for compiler.h:



This graph shows which files directly or indirectly include this file:



### Macros

- **#define L4\_INLINE**  
*L4 Inline function attribute.*
- **#define L4\_ALWAYS\_INLINE**  
*Always inline a function.*
- **#define L4\_NOTHROW**  
*Mark a function declaration and definition as never throwing an exception.*
- **#define L4\_BEGIN\_DECLS**  
*Start section with C types and functions.*
- **#define L4\_END\_DECLS**  
*End section with C types and functions.*
- **#define L4\_CONSTEXPR**  
*Constexpr function attribute.*
- **#define L4\_NORETURN**

- Noreturn function attribute.*
- **#define L4\_NOINSTRUMENT**  
*No instrumentation function attribute.*
- **#define L4\_HIDDEN**  
*Attribute to mark functions, variables, and data types as being explicitly hidden from users of a library.*
- **#define L4\_EXPORT**  
*Attribute to mark functions, variables, and data types as being exported from a library.*
- **#define L4\_LIKELY(x)**  
*Expression is likely to execute.*
- **#define L4\_UNLIKELY(x)**  
*Expression is unlikely to execute.*
- **#define L4\_STICKY(x)**  
*Mark symbol sticky (even not there).*
- **#define L4\_DEPRECATED(s)**  
*Mark symbol deprecated.*
- **#define L4\_stringify\_helper(x)**  
*stringify helper.*
- **#define L4\_stringify(x)**  
*stringify.*

## Functions

- unsigned long **[l4\\_align\\_stack\\_for\\_direct\\_fncall](#)** (unsigned long stack)  
*Specify the desired alignment of the stack pointer.*
- void **[l4\\_barrier](#)** (void)  
*Memory barrier.*
- void **[l4\\_mb](#)** (void)  
*Memory barrier.*
- void **[l4\\_wmb](#)** (void)  
*Write memory barrier.*
- **[L4\\_NORETURN](#)** void **[l4\\_infinite\\_loop](#)** (void)  
*Infinite loop.*

## 17.466.1 Detailed Description

[L4](#) compiler related defines.

Definition in file [compiler.h](#).

## 17.467 compiler.h

[Go to the documentation of this file.](#)

```

00001 /*****
00002  */
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  * Alexander Warg <warg@os.inf.tu-dresden.de>,
00005  * Frank Mehnert <fm3@os.inf.tu-dresden.de>,
00006  * Jork Löser <jork@os.inf.tu-dresden.de>,
00007  * Ronald Aigner <ra3@os.inf.tu-dresden.de>
00008  * economic rights: Technische Universität Dresden (Germany)
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 /*****
00012  */
00013 #ifndef __L4_COMPILER_H__
00014 #define __L4_COMPILER_H__
00015
00016 #if !defined(__ASSEMBLY__) && !defined(__ASSEMBLER__)
00017
00018 #ifndef L4_INLINE
00019 #ifdef __cplusplus
00020 # if __OPTIMIZE__
00021 #   define L4_INLINE_STATIC static __inline__
00022 #   define L4_INLINE_EXTERN extern __inline__
00023 #   ifdef __GNUC_STDC_INLINE__
00024 #   define L4_INLINE L4_INLINE_STATIC
00025 #   else
00026 #   define L4_INLINE L4_INLINE_EXTERN
00027 #   endif
00028 # else /* ! __OPTIMIZE__ */
00029 #   define L4_INLINE static
00030 #   endif /* ! __OPTIMIZE__ */
00031 #else /* __cplusplus */
00032 # define L4_INLINE inline
00033 #endif /* __cplusplus */
00034 #elif defined DOXYGEN
00035 # define L4_INLINE inline
00036 #endif /* L4_INLINE */
00037 #define L4_ALWAYS_INLINE L4_INLINE __attribute__((__always_inline__))
00038
00039 #define L4_DECLARE_CONSTRUCTOR(func, prio) \
00040     static inline __attribute__((constructor(prio))) void func ## _ctor_func(void) { func(); }
00041
00042 #ifndef __cplusplus
00043 #define L4_NO_THROW __attribute__((nothrow))
00044 #define L4_NO_THROW
00045 #ifdef __BEGIN_DECLS
00046 #define L4_BEGIN_DECLS
00047 #define L4_END_DECLS
00048 #define L4_DEFAULT_PARAM(x)
00049 #else /* __cplusplus */
00050 # if __cplusplus >= 201103L
00051 #   define L4_NO_THROW noexcept
00052 #   else /* C++ < 11 */
00053 #   define L4_NO_THROW throw()
00054 #   endif
00055 #endif
00056 #define L4_BEGIN_DECLS extern "C" {
00057 #define L4_END_DECLS }
00058 # if !defined __BEGIN_DECLS || defined DOXYGEN
00059 #   define L4_BEGIN_DECLS extern "C" {
00060 #   endif
00061 # if !defined __END_DECLS || defined DOXYGEN
00062 #   define L4_END_DECLS }
00063 #   endif
00064 # define L4_DEFAULT_PARAM(x) = x
00065 #endif /* __cplusplus */
00066
00067 /* Deprecation hints during compile -- remove later (2025+) */
00068 #ifndef EXTERN_C
00069 #define EXTERN_C DO_NOT_USE_EXTERN_C_ANY_MORE
00070 #endif
00071 #ifndef EXTERN_C_BEGIN

```



```

00181 #define EXTERN_C_BEGIN DO_NOT_USE_EXTERN_C_BEGIN_ANY_MORE__USE_L4_BEGIN_DECLS
00182 #endif
00183 #ifndef EXTERN_C_END
00184 #define EXTERN_C_END DO_NOT_USE_EXTERN_C_END_ANY_MORE__USE_L4_END_DECLS
00185 #endif
00186
00191 #if defined __cplusplus && __cplusplus >= 201402L
00192 # define L4_CONSTEXPR constexpr
00193 #else
00194 # define L4_CONSTEXPR
00195 #endif
00196
00201 #define L4_NORETURN __attribute__((noreturn))
00202
00203 #define L4_PURE __attribute__((pure))
00204
00209 #define L4_NOINSTRUMENT __attribute__((no_instrument_function))
00210 #ifndef L4_HIDDEN
00211 # define L4_HIDDEN __attribute__((visibility("hidden")))
00212 #endif
00213 #if !defined L4_EXPORT || defined DOXYGEN
00214 # define L4_EXPORT __attribute__((visibility("default")))
00215 #endif
00216 #ifndef L4_EXPORT_TYPE
00217 # ifdef __cplusplus
00218 #   define L4_EXPORT_TYPE __attribute__((visibility("default")))
00219 # else
00220 #   define L4_EXPORT_TYPE
00221 # endif
00222 #endif
00223 #define L4_STRONG_ALIAS(name, aliasname) L4__STRONG_ALIAS(name, aliasname)
00224 #ifdef __clang__
00225 #define L4__STRONG_ALIAS(name, aliasname) \
00226     extern __typeof (name) aliasname __attribute__((alias (#name)));
00227 #else
00228 #define L4__STRONG_ALIAS(name, aliasname) \
00229     extern __typeof (name) aliasname __attribute__((alias (#name), copy(name)));
00230 #endif
00231
00239 #if defined(__i386__) || defined(__amd64__) || \
00240     defined(__arm__) || defined(__aarch64__) || \
00241     defined(__mips__) || defined(__riscv) || \
00242     defined(__powerpc__) || defined(__sparc__)
00243 # define L4_STACK_ALIGN __BIGGEST_ALIGNMENT__
00244 #else
00245 # error Define L4_STACK_ALIGN for this target!
00246 #endif
00247
00265 #if defined(__i386__) || defined(__amd64__)
00266 L4_INLINE unsigned long l4_align_stack_for_direct_fncall(unsigned long stack)
00267 {
00268     if ((stack & (L4_STACK_ALIGN - 1)) == (L4_STACK_ALIGN - sizeof(unsigned long)))
00269         return stack;
00270     return (stack & ~(L4_STACK_ALIGN)) - sizeof(unsigned long);
00271 }
00272 #else
00273 L4_INLINE unsigned long l4_align_stack_for_direct_fncall(unsigned long stack)
00274 {
00275     return stack & ~(L4_STACK_ALIGN);
00276 }
00277 #endif
00278
00279 #endif /* !__ASSEMBLY__ */
00280
00281 #include <l4/sys/linkage.h>
00282
00283 #define L4_LIKELY(x) __builtin_expect((x),1)
00284 #define L4_UNLIKELY(x) __builtin_expect((x),0)
00285
00286 /* Make sure that the function is not removed by optimization. Without the
00287  * "used" attribute, unreferenced static functions are removed. */
00288 #define L4_STICKY(x) __attribute__((used)) x
00289 #define L4_DEPRECATED(s) __attribute__((deprecated(s)))
00290
00291 #ifndef static_assert
00292 # if !defined(__cplusplus)
00293 #   define static_assert(x, y) _Static_assert(x, y)
00294 # elif __cplusplus < 201103L
00295 #   define static_assert(x, y) \
00296     extern int l4_static_assert[-(!x)] __attribute__((unused))
00297 # endif
00298 #endif
00299
00300 #define L4_stringify_helper(x) #x
00301 #define L4_stringify(x) L4_stringify_helper(x)
00302
00303 #ifdef __has_builtin

```

```

00304 #define L4_HAS_BUILTIN(def) __has_builtin(def)
00305 #else
00306 #define L4_HAS_BUILTIN(def) 0
00307 #endif
00308
00309 #ifndef __ASSEMBLER__
00313 L4_INLINE void l4_barrier(void);
00314
00318 L4_INLINE void l4_mb(void);
00319
00323 L4_INLINE void l4_wmb(void);
00324
00328 L4_INLINE L4_NORETURN void l4_infinite_loop(void);
00329
00330
00331 /* Implementations */
00332 L4_INLINE void l4_barrier(void)
00333 {
00334     __asm__ __volatile__ (" : : : \"memory\");
00335 }
00336
00337 L4_INLINE void l4_mb(void)
00338 {
00339     __asm__ __volatile__ (" : : : \"memory\");
00340 }
00341
00342 L4_INLINE void l4_wmb(void)
00343 {
00344     __asm__ __volatile__ (" : : : \"memory\");
00345 }
00346
00347 L4_INLINE L4_NORETURN void l4_infinite_loop(void)
00348 {
00349     while (1)
00350         l4_barrier();
00351 }
00352 #endif
00353
00355
00356 #endif /* !__L4_COMPILER_H__ */

```

## 17.468 amd64/l4/sys/consts.h File Reference

Common [L4](#) constants, AMD64 version.

### Macros

- `#define L4_PAGESHIFT 12`  
*Size of a page, log2-based.*
- `#define L4_SUPERPAGESHIFT 21`  
*Size of a large page, log2-based.*

### 17.468.1 Detailed Description

Common [L4](#) constants, AMD64 version.

Definition in file [consts.h](#).

## 17.469 consts.h

[Go to the documentation of this file.](#)

```

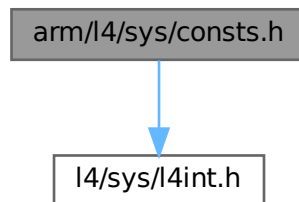
00001 /*****
00007 */
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00010 *      Björn Döbel <doebel@os.inf.tu-dresden.de>,
00011 *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00012 *      economic rights: Technische Universität Dresden (Germany)
00013 *
00014 * License: see LICENSE.spdx (in this directory or the directories above)
00015 */
00016 /*****
00017 #ifndef __L4SYS__INCLUDE__ARCH_AMD64__CONSTS_H__
00018 #define __L4SYS__INCLUDE__ARCH_AMD64__CONSTS_H__
00019
00024 #define L4_PAGESHIFT      12
00025
00030 #define L4_SUPERPAGESHIFT 21
00031
00032 #include_next <l4/sys/consts.h>
00033
00034 #endif /* ! __L4SYS__INCLUDE__ARCH_AMD64__CONSTS_H__ */

```

## 17.470 arm/l4/sys/consts.h File Reference

Common [L4](#) constants, arm version.

#include <l4/sys/l4int.h>  
 Include dependency graph for consts.h:



### Macros

- #define [L4\\_PAGESHIFT](#) 12  
*Size of a page, log2-based.*
- #define [L4\\_SUPERPAGESHIFT](#) 21  
*Size of a large page, log2-based.*

### 17.470.1 Detailed Description

Common [L4](#) constants, arm version.

Definition in file [consts.h](#).

## 17.471 consts.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Björn Döbel <doebel@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #ifndef _L4_SYS_CONSTS_H
00015 #define _L4_SYS_CONSTS_H
00016
00017 /* L4 includes */
00018 #include <l4/sys/l4int.h>
00026 #define L4_PAGESHIFT 12
00027
00031 #define L4_SUPERPAGESHIFT 21
00032
00034
00035 #include_next <l4/sys/consts.h>
00036
00037 #endif /* !_L4_SYS_CONSTS_H */

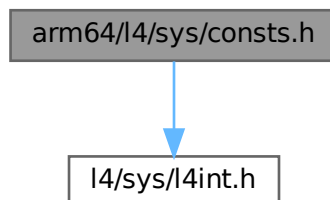
```

## 17.472 arm64/l4/sys/consts.h File Reference

Common [L4](#) constants, arm version.

`#include <l4/sys/l4int.h>`

Include dependency graph for consts.h:



### Macros

- `#define L4_PAGESHIFT 12`  
*Size of a page, log2-based.*
- `#define L4_SUPERPAGESHIFT 21`  
*Size of a large page, log2-based.*

### 17.472.1 Detailed Description

Common [L4](#) constants, arm version.

Definition in file [consts.h](#).

## 17.473 consts.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Björn Döbel <doebel@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #ifndef _L4_SYS_CONSTS_H
00015 #define _L4_SYS_CONSTS_H
00016
00017 /* L4 includes */
00018 #include <l4/sys/l4int.h>
00026 #define L4_PAGESHIFT 12
00027
00031 #define L4_SUPERPAGESHIFT 21
00032
00034
00035 #include_next <l4/sys/consts.h>
00036
00037 #endif /* !_L4_SYS_CONSTS_H */

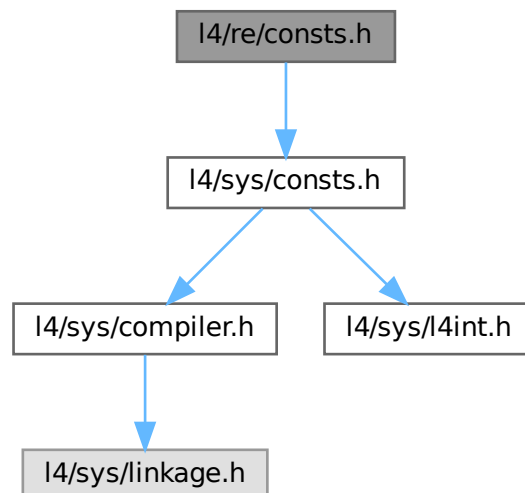
```

## 17.474 l4/re/consts.h File Reference

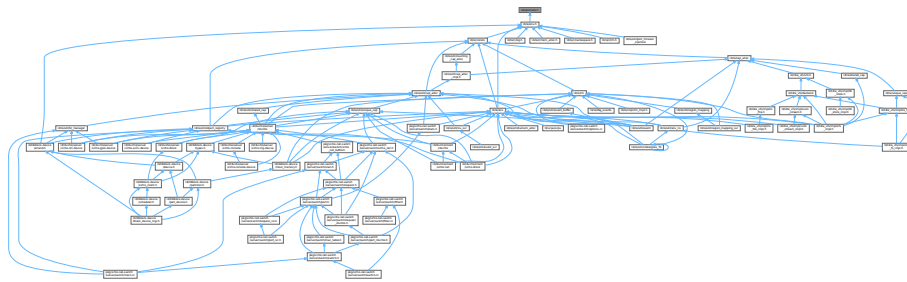
Constants.

```
#include <l4/sys/consts.h>
```

Include dependency graph for consts.h:



This graph shows which files directly or indirectly include this file:



## Enumerations

- `enum`  
*Defaults for local thread priorities.*

### 17.474.1 Detailed Description

Constants.

Definition in file [consts.h](#).

### 17.474.2 Enumeration Type Documentation

#### 17.474.2.1 anonymous enum

anonymous enum

Defaults for local thread priorities.

Priorities are to be seen as local. These are used by the loader and libpthread. They are to be understood as 'local', which means the actual priority of the thread (as seen by the kernel) is the base priority as defined by the scheduler plus the local priority.

Definition at line 28 of file [consts.h](#).

## 17.475 consts.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007  *      economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00013 #include <14/sys/consts.h>
00014
00015 enum
00016 {
00017     L4RE_THIS_TASK_CAP = 1UL << L4_CAP_SHIFT,
00018 };
00019
00028 enum
00029 {
00030     L4RE_MAIN_THREAD_PRIO = 2, /* Priority of the main thread */
00031 };
00032

```

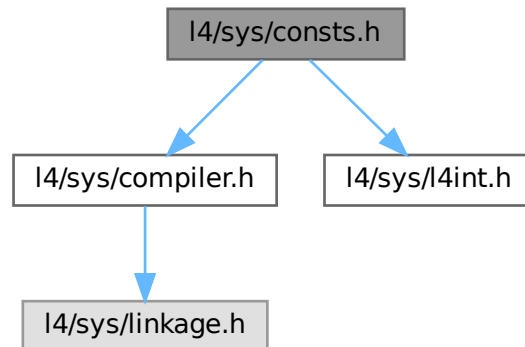
## 17.476 l4/sys/consts.h File Reference

Common constants.

```
#include <l4/sys/compiler.h>
```

```
#include <l4/sys/l4int.h>
```

Include dependency graph for consts.h:



This graph shows which files directly or indirectly include this file:



### Macros

- **#define L4\_PAGESIZE**  
*Minimal page size (in bytes).*
- **#define L4\_PAGEMASK**  
*Mask for the page number.*
- **#define L4\_LOG2\_PAGESIZE**  
*Number of bits used for page offset.*
- **#define L4\_SUPERPAGESIZE**  
*Size of a large page.*
- **#define L4\_SUPERPAGEMASK**  
*Mask for the number of a large page.*
- **#define L4\_LOG2\_SUPERPAGESIZE**  
*Number of bits used as offset for a large page.*
- **#define L4\_INVALID\_PTR** `((void *)L4_INVALID_ADDR)`  
*Invalid address as pointer type.*

## Enumerations

- enum [l4\\_syscall\\_flags\\_t](#) {  
[L4\\_SYSF\\_NONE](#) , [L4\\_SYSF\\_SEND](#) , [L4\\_SYSF\\_RECV](#) , [L4\\_SYSF\\_OPEN\\_WAIT](#) ,  
[L4\\_SYSF\\_REPLY](#) , [L4\\_SYSF\\_CALL](#) , [L4\\_SYSF\\_WAIT](#) , [L4\\_SYSF\\_SEND\\_AND\\_WAIT](#) ,  
[L4\\_SYSF\\_REPLY\\_AND\\_WAIT](#) }  
*Capability selector flags.*
- enum [l4\\_cap\\_consts\\_t](#) {  
[L4\\_CAP\\_SHIFT](#) , [L4\\_CAP\\_SIZE](#) = 1UL << [L4\\_CAP\\_SHIFT](#) , [L4\\_CAP\\_OFFSET](#) , [L4\\_CAP\\_MASK](#) ,  
[L4\\_INVALID\\_CAP](#) , [L4\\_INVALID\\_CAP\\_BIT](#) = 1UL << ([L4\\_CAP\\_SHIFT](#) - 1) }  
*Constants related to capability selectors.*
- enum [l4\\_unmap\\_flags\\_t](#) { [L4\\_FP\\_ALL\\_SPACES](#) , [L4\\_FP\\_DELETE\\_OBJ](#) , [L4\\_FP\\_OTHER\\_SPACES](#) }  
*Flags for the unmap operation.*
- enum [l4\\_msg\\_item\\_consts\\_t](#) {  
[L4\\_ITEM\\_MAP](#) = 8 , [L4\\_ITEM\\_CONT](#) = 1 , [L4\\_MAP\\_ITEM\\_GRANT](#) = 2 , [L4\\_MAP\\_ITEM\\_MAP](#) = 0 ,  
[L4\\_RCV\\_ITEM\\_FORWARD\\_MAPPINGS](#) = 1 , [L4\\_RCV\\_ITEM\\_SINGLE\\_CAP](#) = [L4\\_ITEM\\_MAP](#) | 2 ,  
[L4\\_RCV\\_ITEM\\_LOCAL\\_ID](#) = 4 }  
*Constants for message items.*
- enum [l4\\_buffer\\_desc\\_consts\\_t](#) { [L4\\_BDR\\_MEM\\_SHIFT](#) = 0 , [L4\\_BDR\\_IO\\_SHIFT](#) = 5 , [L4\\_BDR\\_OBJ\\_SHIFT](#)  
= 10 , [L4\\_BDR\\_OFFSET\\_MASK](#) = (1UL << 20) - 1 }  
*Constants for buffer descriptors.*
- enum [l4\\_default\\_caps\\_t](#) {  
[L4\\_BASE\\_TASK\\_CAP](#) , [L4\\_BASE\\_FACTORY\\_CAP](#) , [L4\\_BASE\\_THREAD\\_CAP](#) , [L4\\_BASE\\_PAGER\\_CAP](#) ,  
[L4\\_BASE\\_LOG\\_CAP](#) , [L4\\_BASE\\_ICU\\_CAP](#) , [L4\\_BASE\\_SCHEDULER\\_CAP](#) , [L4\\_BASE\\_IOMMU\\_CAP](#) ,  
[L4\\_BASE\\_DEBUGGER\\_CAP](#) , [L4\\_BASE\\_ARM\\_SMCCC\\_CAP](#) , [L4\\_BASE\\_CAPS\\_LAST\\_P1](#) , [L4\\_BASE\\_CAPS\\_LAST](#)  
= [L4\\_BASE\\_CAPS\\_LAST\\_P1](#) - 1 }  
*Default capabilities setup for the initial tasks.*
- enum [l4\\_addr\\_consts\\_t](#) { [L4\\_INVALID\\_ADDR](#) = ~0UL }  
*Address related constants.*

## Functions

- [l4\\_addr\\_t l4\\_trunc\\_page](#) ([l4\\_addr\\_t](#) address) [L4\\_NOTHROW](#)  
*Round an address down to the next lower page boundary.*
- [l4\\_addr\\_t l4\\_trunc\\_size](#) ([l4\\_addr\\_t](#) address, unsigned char bits) [L4\\_NOTHROW](#)  
*Round an address down to the next lower flexpage with size bits.*
- [l4\\_addr\\_t l4\\_round\\_page](#) ([l4\\_addr\\_t](#) address) [L4\\_NOTHROW](#)  
*Round address up to the next page.*
- [l4\\_addr\\_t l4\\_round\\_size](#) ([l4\\_addr\\_t](#) value, unsigned char bits) [L4\\_NOTHROW](#)  
*Round value up to the next alignment with bits size.*
- unsigned [l4\\_bytes\\_to\\_mwords](#) (unsigned size) [L4\\_NOTHROW](#)  
*Determine how many machine words ([l4\\_umword\\_t](#)) are required to store a buffer of 'size' bytes.*

## 17.476.1 Detailed Description

Common constants.

Definition in file [consts.h](#).



## 17.477 consts.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Björn Döbel <doebel@os.inf.tu-dresden.de>,
00010  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00011  *      economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * License: see LICENSE.spdx (in this directory or the directories above)
00014  */
00015 #ifndef __L4_SYS__INCLUDE__CONSTS_H__
00016 #define __L4_SYS__INCLUDE__CONSTS_H__
00017
00018 #include <l4/sys/compiler.h>
00019 #include <l4/sys/l4int.h>
00020
00050 enum l4_syscall_flags_t
00051 {
00056     L4_SYSF_NONE           = 0x00,
00057
00069     L4_SYSF_SEND           = 0x01,
00070
00080     L4_SYSF_RECV           = 0x02,
00081
00091     L4_SYSF_OPEN_WAIT     = 0x04,
00092
00100     L4_SYSF_REPLY          = 0x08,
00101
00108     L4_SYSF_CALL           = L4_SYSF_SEND | L4_SYSF_RECV,
00109
00116     L4_SYSF_WAIT           = L4_SYSF_OPEN_WAIT | L4_SYSF_RECV,
00117
00124     L4_SYSF_SEND_AND_WAIT = L4_SYSF_OPEN_WAIT | L4_SYSF_CALL,
00125
00132     L4_SYSF_REPLY_AND_WAIT = L4_SYSF_WAIT | L4_SYSF_SEND | L4_SYSF_REPLY
00133 };
00134
00139 enum l4_cap_consts_t
00140 {
00142     L4_CAP_SHIFT           = 12UL,
00144     L4_CAP_SIZE             = 1UL << L4_CAP_SHIFT,
00146     L4_CAP_OFFSET           = 1UL << L4_CAP_SHIFT,
00151     L4_CAP_MASK             = ~0UL << (L4_CAP_SHIFT - 1),
00153     L4_INVALID_CAP          = ~0UL << (L4_CAP_SHIFT - 1),
00154
00155     L4_INVALID_CAP_BIT      = 1UL << (L4_CAP_SHIFT - 1),
00156 };
00157
00158 enum l4_sched_consts_t
00159 {
00160     L4_SCHED_MIN_PRIO      = 1,
00161     L4_SCHED_MAX_PRIO      = 255,
00162 };
00163
00169 enum l4_unmap_flags_t
00170 {
00183     L4_FP_ALL_SPACES        = 0x80000000UL,
00184
00197     L4_FP_DELETE_OBJ        = 0xc0000000UL,
00198
00205     L4_FP_OTHER_SPACES      = 0x00UL
00206 };
00207
00212 enum l4_msg_item_consts_t
00213 {
00214     L4_ITEM_MAP              = 8,
00215
00220     L4_ITEM_CONT             = 1,
00221
00222     // send
00245     L4_MAP_ITEM_GRANT        = 2,
00246
00247     L4_MAP_ITEM_MAP          = 0,
00248
00249     // receive
00260     L4_RCV_ITEM_FORWARD_MAPPINGS = 1,
00261
00275     L4_RCV_ITEM_SINGLE_CAP   = L4_ITEM_MAP | 2,
00276
00296     L4_RCV_ITEM_LOCAL_ID     = 4,
00297 };
00298

```

```

00303 enum l4_buffer_desc_consts_t
00304 {
00305     L4_BDR_MEM_SHIFT    = 0,
00306     L4_BDR_IO_SHIFT     = 5,
00307     L4_BDR_OBJ_SHIFT    = 10,
00308     L4_BDR_OFFSET_MASK = (1UL << 20) - 1,
00309 };
00310
00324 enum l4_default_caps_t
00325 {
00327     L4_BASE_TASK_CAP      = 1UL << L4_CAP_SHIFT,
00329     L4_BASE_FACTORY_CAP   = 2UL << L4_CAP_SHIFT,
00331     L4_BASE_THREAD_CAP    = 3UL << L4_CAP_SHIFT,
00333     L4_BASE_PAGER_CAP     = 4UL << L4_CAP_SHIFT,
00347     L4_BASE_LOG_CAP       = 5UL << L4_CAP_SHIFT,
00349     L4_BASE_ICU_CAP       = 6UL << L4_CAP_SHIFT,
00351     L4_BASE_SCHEDULER_CAP = 7UL << L4_CAP_SHIFT,
00358     L4_BASE_IOMMU_CAP     = 8UL << L4_CAP_SHIFT,
00366     L4_BASE_DEBUGGER_CAP  = 10UL << L4_CAP_SHIFT,
00373     L4_BASE_ARM_SMCCC_CAP = 11UL << L4_CAP_SHIFT,
00374
00376     L4_BASE_CAPS_LAST_P1,
00378     L4_BASE_CAPS_LAST = L4_BASE_CAPS_LAST_P1 - 1
00379 };
00380
00391 #define L4_PAGESIZE      (1UL << L4_PAGESHIFT)
00392
00400 #define L4_PAGEMASK      (~(L4_PAGESIZE - 1))
00401
00409 #define L4_LOG2_PAGESIZE L4_PAGESHIFT
00410
00418 #define L4_SUPERPAGESIZE (1UL << L4_SUPERPAGESHIFT)
00419
00427 #define L4_SUPERPAGEMASK (~(L4_SUPERPAGESIZE - 1))
00428
00435 #define L4_LOG2_SUPERPAGESIZE L4_SUPERPAGESHIFT
00436
00447 L4_INLINE l4_addr_t l4_trunc_page(l4_addr_t address) L4_NOTHROW;
00448 L4_INLINE l4_addr_t l4_trunc_page(l4_addr_t address) L4_NOTHROW
00449 { return address & L4_PAGEMASK; }
00450
00458 L4_INLINE l4_addr_t l4_trunc_size(l4_addr_t address, unsigned char bits) L4_NOTHROW;
00459 L4_INLINE l4_addr_t l4_trunc_size(l4_addr_t address, unsigned char bits) L4_NOTHROW
00460 { return address & (~0UL << bits); }
00461
00472 L4_INLINE l4_addr_t l4_round_page(l4_addr_t address) L4_NOTHROW;
00473 L4_INLINE l4_addr_t l4_round_page(l4_addr_t address) L4_NOTHROW
00474 { return (address + L4_PAGESIZE - 1) & L4_PAGEMASK; }
00475
00483 L4_INLINE l4_addr_t l4_round_size(l4_addr_t value, unsigned char bits) L4_NOTHROW;
00484 L4_INLINE l4_addr_t l4_round_size(l4_addr_t value, unsigned char bits) L4_NOTHROW
00485 { return (value + (1UL << bits) - 1) & (~0UL << bits); }
00486
00495 L4_INLINE unsigned l4_bytes_to_mwords(unsigned size) L4_NOTHROW;
00496 L4_INLINE unsigned l4_bytes_to_mwords(unsigned size) L4_NOTHROW
00497 { return (size + sizeof(l4_umword_t) - 1) / sizeof(l4_umword_t); }
00498
00503 enum l4_addr_consts_t {
00505     L4_INVALID_ADDR = ~0UL
00506 };
00507
00512 #define L4_INVALID_PTR ((void *)L4_INVALID_ADDR)
00513
00514 #ifndef NULL
00515 #ifndef __cplusplus
00516 # define NULL ((void *)0)
00520 #elif __cplusplus >= 201103L
00521 # define NULL nullptr
00522 #else
00523 # define NULL 0
00524 #endif
00525 #endif
00526
00527 #endif /* ! __L4_SYS__INCLUDE__CONSTS_H__ */

```

## 17.478 x86/I4/sys/consts.h File Reference

Common [L4](#) constants, x86 version.

## Macros

- `#define L4_PAGESHIFT 12`  
*Size of a page log2-based.*
- `#define L4_SUPERPAGESHIFT 22`  
*Size of a large page log2-based.*

### 17.478.1 Detailed Description

Common [L4](#) constants, x86 version.

Definition in file [consts.h](#).

## 17.479 consts.h

[Go to the documentation of this file.](#)

```
00001 /*****
00007 */
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 *           Alexander Warg <warg@os.inf.tu-dresden.de>,
00010 *           Björn Döbel <doebel@os.inf.tu-dresden.de>,
00011 *           Lars Reuther <reuther@os.inf.tu-dresden.de>
00012 *           economic rights: Technische Universität Dresden (Germany)
00013 *
00014 * License: see LICENSE.spdx (in this directory or the directories above)
00015 */
00016 /*****
00017 #ifndef __L4SYS__INCLUDE__ARCH_X86__CONSTS_H__
00018 #define __L4SYS__INCLUDE__ARCH_X86__CONSTS_H__
00019
00024 #define L4_PAGESHIFT    12
00025
00030 #define L4_SUPERPAGESHIFT 22
00031
00032 #include_next <l4/sys/consts.h>
00033
00034 #endif /* ! __L4SYS__INCLUDE__ARCH_X86__CONSTS_H__ */
```

## 17.480 capability.h

```
00001
00002 #pragma once
00003
00004 #include <l4/sys/consts.h>
00005 #include <l4/sys/types.h>
00006 #include <l4/sys/task.h>
00007
00008 namespace L4 {
00009
00010 class Task;
00011 class Kobject;
00012
00013 template< typename T > class L4_EXPORT Cap;
00014
00025 class L4_EXPORT Cap_base
00026 {
00027 public:
00029     enum No_init_type
00030     {
00034         No_init
00035     };
00036
00040     enum Cap_type
00041     {
00042         Invalid = L4_INVALID_CAP
00043     };
00044
```

```

00049  l4_cap_idx_t cap() const noexcept { return _c; }
00050
00057  bool is_valid() const noexcept { return !(_c & L4_INVALID_CAP_BIT); }
00058
00062  int invalid_cap_error() const noexcept { return _c & ~L4_INVALID_CAP_BIT; }
00063
00064  explicit operator bool () const noexcept
00065  { return !(_c & L4_INVALID_CAP_BIT); }
00066
00074  l4_fpage_t fpage(unsigned rights = L4_CAP_FPAGE_RWS) const noexcept
00075  { return l4_obj_fpage(_c, 0, rights); }
00076
00086  l4_umword_t snd_base(unsigned grant = L4_MAP_ITEM_MAP,
00087                      l4_cap_idx_t base = L4_INVALID_CAP) const noexcept
00088  {
00089      if (base == L4_INVALID_CAP)
00090          base = _c;
00091      return l4_map_obj_control(base, grant);
00092  }
00093
00094
00098  bool operator == (Cap_base const &o) const noexcept
00099  { return _c == o._c; }
00100
00104  bool operator != (Cap_base const &o) const noexcept
00105  { return _c != o._c; }
00106
00120  inline l4_msgtag_t validate(l4_utcb_t *u = l4_utcb()) const noexcept;
00121
00136  inline l4_msgtag_t validate(Cap<Task> task,
00137                          l4_utcb_t *u = l4_utcb()) const noexcept;
00138
00142  void invalidate() noexcept { _c = L4_INVALID_CAP; }
00143 protected:
00149  explicit Cap_base(l4_cap_idx_t c) noexcept : _c(c) {}
00153  explicit Cap_base(Cap_type cap) noexcept : _c(cap) {}
00154
00160  explicit Cap_base(l4_default_caps_t cap) noexcept : _c(cap) {}
00161
00165  explicit Cap_base() noexcept {}
00166
00176  void move(Cap_base const &src) const
00177  {
00178      if (!is_valid() || !src.is_valid())
00179          return;
00180
00181      l4_task_map(L4_BASE_TASK_CAP, L4_BASE_TASK_CAP, src.fpage(L4_CAP_FPAGE_RWSD),
00182                snd_base(L4_MAP_ITEM_GRANT) | L4_FPAGE_C_OBJ_RIGHTS);
00183  }
00184
00192  void copy(Cap_base const &src) const
00193  {
00194      if (!is_valid() || !src.is_valid())
00195          return;
00196
00197      l4_task_map(L4_BASE_TASK_CAP, L4_BASE_TASK_CAP, src.fpage(L4_CAP_FPAGE_RWSD),
00198                snd_base() | L4_FPAGE_C_OBJ_RIGHTS);
00199  }
00200
00203  l4_cap_idx_t _c;
00204 };
00205
00206
00222 template< typename T >
00223 class L4_EXPORT Cap : public Cap_base
00224 {
00225 private:
00226     friend class L4::Kobject;
00227
00239     explicit Cap(T const *p) noexcept
00240     : Cap_base(reinterpret_cast<l4_cap_idx_t>(p)) {}
00241
00242 public:
00243
00250     template< typename From >
00251     static void check_convertible_from() noexcept
00252     {
00253         using To = T;
00254         [[maybe_unused]] To* t = static_cast<From*>(nullptr);
00255     }
00256
00263     template< typename From >
00264     static void check_castable_from() noexcept
00265     {
00266         using To = T;
00267         [[maybe_unused]] To *t = static_cast<To *>(static_cast<From *>(nullptr));
00268     }

```

```

00269
00274     template< typename O >
00275     Cap(Cap<O> const &o) noexcept : Cap_base(o.cap())
00276     { check_convertible_from<O>(); }
00277
00282     Cap(Cap_type cap) noexcept : Cap_base(cap) {}
00283
00288     Cap(l4_default_caps_t cap) noexcept : Cap_base(cap) {}
00289
00294     explicit Cap(l4_cap_idx_t idx = L4_INVALID_CAP) noexcept : Cap_base(idx) {}
00295
00299     explicit Cap(No_init_type) noexcept {}
00300
00307     Cap move(Cap const &src) const
00308     {
00309         Cap_base::move(src);
00310         return *this;
00311     }
00312
00317     Cap copy(Cap const &src) const
00318     {
00319         Cap_base::copy(src);
00320         return *this;
00321     }
00322
00326     T *operator -> () const noexcept { return reinterpret_cast<T*>(_c); }
00327 };
00328
00329
00340     template<
00341     class L4_EXPORT Cap<void> : public Cap_base
00342     {
00343     public:
00344
00345         explicit Cap(void const *p) noexcept
00346         : Cap_base(reinterpret_cast<l4_cap_idx_t>(p)) {}
00347
00351         Cap(Cap_type cap) noexcept : Cap_base(cap) {}
00352
00357         Cap(l4_default_caps_t cap) noexcept : Cap_base(cap) {}
00358
00363         explicit Cap(l4_cap_idx_t idx = L4_INVALID_CAP) noexcept : Cap_base(idx) {}
00364         explicit Cap(No_init_type) noexcept {}
00365
00366         template< typename From >
00367         static void check_convertible_from() noexcept {}
00368
00369         template< typename From >
00370         static void check_castable_from() noexcept {}
00371
00378         Cap move(Cap const &src) const
00379         {
00380             Cap_base::move(src);
00381             return *this;
00382         }
00383
00388         Cap copy(Cap const &src) const
00389         {
00390             Cap_base::copy(src);
00391             return *this;
00392         }
00393
00394         template< typename T >
00395         Cap(Cap<T> const &o) noexcept : Cap_base(o.cap()) {}
00396 };
00397
00414     template< typename T, typename F >
00415     inline
00416     Cap<T> cap_cast(Cap<F> const &c) noexcept
00417     {
00418         Cap<T>::template check_castable_from<F>();
00419         return Cap<T>(c.cap());
00420     }
00421
00422     // gracefully deal with L4::Kobject ambiguity
00423     template< typename T >
00424     inline
00425     Cap<T> cap_cast(Cap<L4::Kobject> const &c) noexcept
00426     {
00427         return Cap<T>(c.cap());
00428     }
00429
00445     template< typename T, typename F >
00446     inline
00447     Cap<T> cap_reinterpret_cast(Cap<F> const &c) noexcept
00448     {
00449         return Cap<T>(c.cap());

```

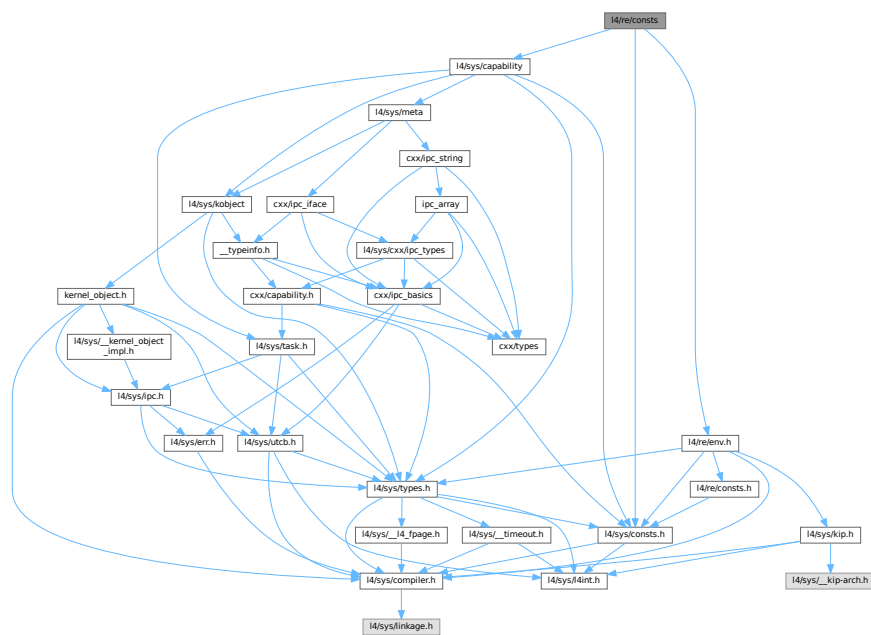
```
00450 }
00451
00452 }
```

## 17.481 l4/re/consts File Reference

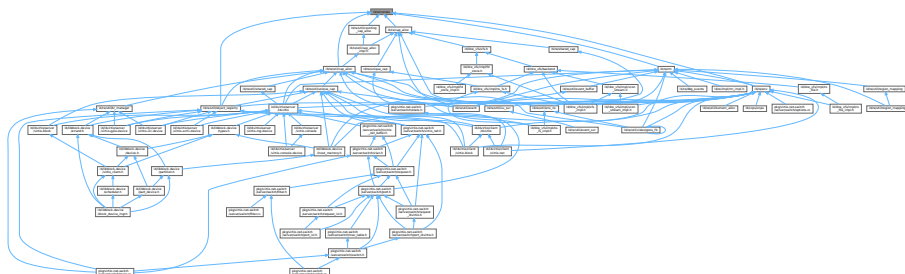
Constants.

```
#include <l4/sys/capability>
#include <l4/sys/consts.h>
#include <l4/re/env.h>
```

Include dependency graph for consts:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [L4Re](#)  
*L4Re C++ Interfaces.*

## 17.481.1 Detailed Description

Constants.

Definition in file [consts](#).

## 17.482 consts

[Go to the documentation of this file.](#)

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00015 #include <l4/sys/capability>
00016 #include <l4/sys/consts.h>
00017 #include <l4/re/env.h>
00018
00019 namespace L4Re {
00020     static L4::Cap<L4::Task>::Cap_type const This_task
00021     = static_cast<L4::Cap<L4::Task>::Cap_type>(L4RE_THIS_TASK_CAP);
00022 }
```

## 17.483 consts

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002
00003 #pragma once
00004
00005 #include <l4/sys/consts.h>
00006
00007 namespace L4 {
00008
00017     template<typename T>
00018     constexpr T trunc_order(T val, unsigned char order)
00019     {
00020         return val & ((~T(0)) << order);
00021     }
00022
00031     template<typename T>
00032     constexpr T round_order(T val, unsigned char order)
00033     {
00034         return (val + (T(1) << order) - T(1)) & ((~T(0)) << order);
00035     }
00036
00037     template<typename T>
00038     constexpr T trunc_page(T val)
00039     {
00040         return trunc_order(val, L4_PAGESHIFT);
00041     }
00042
00043     template<typename T>
00044     constexpr T round_page(T val)
00045     {
00046         return round_order(val, L4_PAGESHIFT);
00047     }
00048
00049     template<typename T>
00050     inline unsigned char
00051     max_order(unsigned char order, T addr,
00052              T min_addr, T max_addr,
00053              T hotspot = T(0))
00054     {
00055         while (order < 30 /* limit to 1GB flexpages */)
00056         {
00057             T mask;
00058             T base = trunc_order(addr, order + 1);
00059             if (base < min_addr)
```

```

00060         return order;
00061
00062         if (base + (T(1) < (order + 1)) - T(1) > max_addr - T(1))
00063             return order;
00064
00065         mask = ~((~T(0)) < (order + 1));
00066         if (hotspot == ~T(0) || ((addr ^ hotspot) & mask))
00067             break;
00068
00069         ++order;
00070     }
00071
00072     return order;
00073 }
00074
00075 }

```

## 17.484 ipc\_array

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "types"
00010 #include "ipc_basics"
00011 #include "ipc_types"
00012
00013 namespace L4 { namespace Ipc L4_EXPORT {
00014
00016     typedef unsigned short Array_len_default;
00017
00027     template<typename ELEM_TYPE, typename LEN_TYPE = Array_len_default >
00028     struct Array_ref
00029     {
00030         typedef ELEM_TYPE *ptr_type;
00031         typedef LEN_TYPE len_type;
00032
00033         len_type length;
00034         ptr_type data;
00035         Array_ref() = default;
00036         Array_ref(len_type length, ptr_type data)
00037             : length(length), data(data)
00038         {}
00039
00040         template<typename X> struct Non_const
00041         { typedef Array_ref<X, LEN_TYPE> type; };
00042
00043         template<typename X> struct Non_const<X const>
00044         { typedef Array_ref<X, LEN_TYPE> type; };
00045
00046         Array_ref(typename Non_const<ELEM_TYPE>::type const &other)
00047             : length(other.length), data(other.data)
00048         {}
00049
00050         Array_ref &operator = (typename Non_const<ELEM_TYPE>::type const &other)
00051         {
00052             this->length = other.length;
00053             this->data = other.data;
00054             return *this;
00055         }
00056     };
00057
00080     template<typename ELEM_TYPE, typename LEN_TYPE = Array_len_default>
00081     struct Array : Array_ref<ELEM_TYPE, LEN_TYPE>
00082     {
00084         Array() {}
00086         Array(LEN_TYPE length, ELEM_TYPE *data)
00087             : Array_ref<ELEM_TYPE, LEN_TYPE>(length, data)
00088         {}
00089
00090         template<typename X> struct Non_const
00091         { typedef Array<X, LEN_TYPE> type; };
00092
00093         template<typename X> struct Non_const<X const>
00094         { typedef Array<X, LEN_TYPE> type; };
00095
00097         Array(typename Non_const<ELEM_TYPE>::type const &other)
00098             : Array_ref<ELEM_TYPE, LEN_TYPE>(other.length, other.data)
00099         {}

```



```

00100
00101     Array &operator = (typename Non_const<ELEM_TYPE>::type const &other)
00102     {
00103         this->length = other.length;
00104         this->data = other.data;
00105         return *this;
00106     }
00107 };
00108
00122 template< typename ELEM_TYPE,
00123           typename LEN_TYPE = Array_len_default,
00124           LEN_TYPE MAX      = (L4_UTCB_GENERIC_DATA_SIZE *
00125                               sizeof(l4_umword_t)) / sizeof(ELEM_TYPE) >
00126 struct Array_in_buf
00127 {
00128     typedef Array_ref<ELEM_TYPE, LEN_TYPE> array;
00129     typedef Array_ref<ELEM_TYPE const, LEN_TYPE> const_array;
00130
00132     ELEM_TYPE data[MAX];
00134     LEN_TYPE length;
00135
00137     void copy_in(const_array a)
00138     {
00139         length = a.length;
00140         if (length > MAX)
00141             length = MAX;
00142
00143         for (LEN_TYPE i = 0; i < length; ++i)
00144             data[i] = a.data[i];
00145     }
00146
00148     Array_in_buf(const_array a) { copy_in(a); }
00150     Array_in_buf(array a) { copy_in(a); }
00151 };
00152
00153 // implementation details for transmission
00154 namespace Msg {
00155
00157 template<typename A, typename LEN>
00158 struct Elem< Array<A, LEN> >
00159 {
00161     typedef Array<A, LEN> arg_type;
00163     typedef Array_ref<A, LEN> svr_type;
00164     typedef svr_type svr_arg_type;
00165     enum { Is_optional = false };
00166 };
00167
00169 template<typename A, typename LEN>
00170 struct Elem< Array<A, LEN> & >
00171 {
00173     typedef Array<A, LEN> &arg_type;
00175     typedef Array_ref<A, LEN> svr_type;
00177     typedef svr_type &svr_arg_type;
00178     enum { Is_optional = false };
00179 };
00180
00182 template<typename A, typename LEN>
00183 struct Elem< Array_ref<A, LEN> & >
00184 {
00186     typedef Array_ref<A, LEN> &arg_type;
00188     typedef Array_ref<typename L4::Types::Remove_const<A>::type, LEN> svr_type;
00190     typedef svr_type &svr_arg_type;
00191     enum { Is_optional = false };
00192 };
00193
00194 template<typename A> struct Class<Array<A> > : Class<A>::type {};
00195 template<typename A> struct Class<Array_ref<A> > : Class<A>::type {};
00196
00197 namespace Detail {
00198
00199 template<typename A, typename LEN, typename ARRAY, bool REF>
00200 struct Clnt_val_ops_d_in : Clnt_noops<ARRAY>
00201 {
00202     using Clnt_noops<ARRAY>::to_msg;
00203     static int to_msg(char *msg, unsigned offset, unsigned limit,
00204                      ARRAY a, Dir_in, Cls_data)
00205     {
00206         offset = align_to<LEN>(offset);
00207         if (L4_UNLIKELY(!check_size<LEN>(offset, limit)))
00208             return -L4_MSGTOOLONG;
00209         *reinterpret_cast<LEN *>(msg + offset) = a.length;
00210         offset = align_to<A>(offset + sizeof(LEN));
00211         if (L4_UNLIKELY(!check_size<A>(offset, limit, a.length)))
00212             return -L4_MSGTOOLONG;
00213         typedef typename L4::Types::Remove_const<A>::type elem_type;
00214         elem_type *data = reinterpret_cast<elem_type*>(msg + offset);
00215

```

```

00216     // we do not correctly handle overlaps
00217     if (!REF || data != a.data)
00218     {
00219         for (LEN i = 0; i < a.length; ++i)
00220             data[i] = a.data[i];
00221     }
00222
00223     return offset + a.length * sizeof(A);
00224 }
00225 };
00226 } // namespace Detail
00227
00228 template<typename A, typename LEN>
00229 struct Clnt_val_ops<Array<A, LEN>, Dir_in, Cls_data> :
00230     Detail::Clnt_val_ops_d_in<A, LEN, Array<A, LEN>, false> {};
00231
00232 template<typename A, typename LEN>
00233 struct Clnt_val_ops<Array_ref<A, LEN>, Dir_in, Cls_data> :
00234     Detail::Clnt_val_ops_d_in<A, LEN, Array_ref<A, LEN>, true> {};
00235
00236 template<typename A, typename LEN, typename CLASS>
00237 struct Svr_val_ops<Array_ref<A, LEN>, Dir_in, CLASS> :
00238     Svr_noops<Array_ref<A, LEN>>
00239 {
00240     typedef Array_ref<A, LEN> svr_type;
00241
00242     using Svr_noops<svr_type>::to_svr;
00243     static int to_svr(char *msg, unsigned offset, unsigned limit,
00244                      svr_type &a, Dir_in, Cls_data)
00245     {
00246         offset = align_to<LEN>(offset);
00247         if (L4_UNLIKELY(!check_size<LEN>(offset, limit)))
00248             return -L4_MSGTOOSHORT;
00249         a.length = *reinterpret_cast<LEN*>(msg + offset);
00250         offset = align_to<A>(offset + sizeof(LEN));
00251         if (L4_UNLIKELY(!check_size<A>(offset, limit, a.length)))
00252             return -L4_MSGTOOSHORT;
00253         a.data = reinterpret_cast<A*>(msg + offset);
00254         return offset + a.length * sizeof(A);
00255     }
00256 };
00257
00258 template<typename A, typename LEN>
00259 struct Svr_xmit<Array<A, LEN>> : Svr_xmit<Array_ref<A, LEN>> {};
00260
00261 template<typename A, typename LEN>
00262 struct Clnt_val_ops<Array<A, LEN>, Dir_out, Cls_data> : Clnt_noops<Array<A, LEN>>
00263 {
00264     typedef Array<A, LEN> type;
00265
00266     using Clnt_noops<type>::from_msg;
00267     static int from_msg(char *msg, unsigned offset, unsigned limit, long,
00268                        type &a, Dir_out, Cls_data)
00269     {
00270         offset = align_to<LEN>(offset);
00271         if (L4_UNLIKELY(!check_size<LEN>(offset, limit)))
00272             return -L4_MSGTOOSHORT;
00273
00274         LEN l = *reinterpret_cast<LEN*>(msg + offset);
00275
00276         offset = align_to<A>(offset + sizeof(LEN));
00277         if (L4_UNLIKELY(!check_size<A>(offset, limit, l)))
00278             return -L4_MSGTOOSHORT;
00279
00280         A *data = reinterpret_cast<A*>(msg + offset);
00281
00282         if (l > a.length)
00283             l = a.length;
00284         else
00285             a.length = l;
00286
00287         for (unsigned i = 0; i < l; ++i)
00288             a.data[i] = data[i];
00289
00290         return offset + l * sizeof(A);
00291     };
00292 };
00293
00294 template<typename A, typename LEN>
00295 struct Clnt_val_ops<Array_ref<A, LEN>, Dir_out, Cls_data> :
00296     Clnt_noops<Array_ref<A, LEN>>
00297 {
00298     typedef Array_ref<A, LEN> type;
00299
00300     using Clnt_noops<type>::from_msg;
00301     static int from_msg(char *msg, unsigned offset, unsigned limit, long,
00302                        type &a, Dir_out, Cls_data)

```

```

00303 {
00304     offset = align_to<LEN>(offset);
00305     if (L4_UNLIKELY(!check_size<LEN>(offset, limit)))
00306         return -L4_EMSGTOOSHORT;
00307
00308     LEN l = *reinterpret_cast<LEN *>(msg + offset);
00309
00310     offset = align_to<A>(offset + sizeof(LEN));
00311     if (L4_UNLIKELY(!check_size<A>(offset, limit, 1)))
00312         return -L4_EMSGTOOSHORT;
00313
00314     a.data = reinterpret_cast<A*>(msg + offset);
00315     a.length = l;
00316     return offset + l * sizeof(A);
00317 };
00318 };
00319
00320 template<typename A, typename LEN, typename CLASS>
00321 struct Svr_val_ops<Array_ref<A, LEN>, Dir_out, CLASS> :
00322     Svr_noops<Array_ref<typename L4::Types::Remove_const<A>::type, LEN> &>
00323 {
00324     typedef typename L4::Types::Remove_const<A>::type elem_type;
00325     typedef Array_ref<elem_type, LEN> &svr_type;
00326
00327     using Svr_noops<svr_type>::to_svr;
00328     static int to_svr(char *msg, unsigned offset, unsigned limit,
00329                      svr_type a, Dir_out, Cls_data)
00330     {
00331         offset = align_to<LEN>(offset);
00332         if (L4_UNLIKELY(!check_size<LEN>(offset, limit)))
00333             return -L4_EMSGTOOLONG;
00334
00335         offset = align_to<A>(offset + sizeof(LEN));
00336         a.data = reinterpret_cast<elem_type *>(msg + offset);
00337         a.length = (limit - offset) / sizeof(A);
00338         return offset;
00339     }
00340
00341     using Svr_noops<svr_type>::from_svr;
00342     static int from_svr(char *msg, unsigned offset, unsigned limit, long,
00343                       svr_type a, Dir_out, Cls_data)
00344     {
00345         offset = align_to<LEN>(offset);
00346         if (L4_UNLIKELY(!check_size<LEN>(offset, limit)))
00347             return -L4_EMSGTOOLONG;
00348
00349         *reinterpret_cast<LEN *>(msg + offset) = a.length;
00350
00351         offset = align_to<A>(offset + sizeof(LEN));
00352         if (L4_UNLIKELY(!check_size<A>(offset, limit, a.length)))
00353             return -L4_EMSGTOOLONG;
00354
00355         return offset + a.length * sizeof(A);
00356     }
00357 };
00358
00359 template<typename A, typename LEN>
00360 struct Svr_xmit<Array<A, LEN> &> : Svr_xmit<Array_ref<A, LEN> &> {};
00361
00362 // Pointer to array is not implemented.
00363 template<typename A, typename LEN>
00364 struct Is_valid_rpc_type<Array_ref<A, LEN> *> : L4::Types::False {};
00365
00366 // Optional input arrays are not implemented.
00367 template<typename A, typename LEN>
00368 struct Is_valid_rpc_type<Opt<Array_ref<A, LEN> > > : L4::Types::False {};
00369
00370 } // namespace Msg
00371
00372 }

```

## 17.485 ipc\_basics

```

00001 // vi:set ft=c++ -- Mode: C++ --
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "types"
00010 #include <14/sys/utcb.h>

```

```

00011 #include <l4/sys/err.h>
00012
00013 namespace L4 {
00014
00016 namespace Ipc {
00017
00019 namespace Msg {
00020
00021 using L4::Types::True;
00022 using L4::Types::False;
00023
00030 constexpr unsigned long align_to(unsigned long bytes, unsigned long align) noexcept
00031 { return (bytes + align - 1) & ~(align - 1); }
00032
00039 template<typename T>
00040 constexpr unsigned long align_to(unsigned long bytes) noexcept
00041 { return align_to(bytes, __alignof(T)); }
00042
00052 template<typename T>
00053 constexpr bool check_size(unsigned offset, unsigned limit) noexcept
00054 {
00055     return offset + sizeof(T) <= limit;
00056 }
00057
00070 template<typename T, typename CTYPE>
00071 inline bool check_size(unsigned offset, unsigned limit, CTYPE cnt) noexcept
00072 {
00073     if (L4_UNLIKELY(sizeof(CTYPE) <= sizeof(unsigned) &&
00074                     ~0U / sizeof(T) <= static_cast<unsigned>(cnt)))
00075         return false;
00076
00077     if (L4_UNLIKELY(sizeof(CTYPE) > sizeof(unsigned) &&
00078                     static_cast<CTYPE>(~0U / sizeof(T)) <= cnt))
00079         return false;
00080
00081     return sizeof(T) * cnt <= limit - offset;
00082 }
00083
00084
00085 enum
00086 {
00088     Word_bytes = sizeof(l4_umword_t),
00090     Item_words = 2,
00092     Item_bytes = Word_bytes * Item_words,
00094     Mr_words   = L4_UTCB_GENERIC_DATA_SIZE,
00096     Mr_bytes   = Word_bytes * Mr_words,
00098     Br_bytes   = Word_bytes * L4_UTCB_GENERIC_BUFFERS_SIZE,
00099 };
00100
00101
00113 template<typename T>
00114 inline int msg_add(char *msg, unsigned offs, unsigned limit, T v) noexcept
00115 {
00116     offs = align_to<T>(offs);
00117     if (L4_UNLIKELY(!check_size<T>(offs, limit)))
00118         return -L4_EMMSGTOOLONG;
00119     *reinterpret_cast<typename L4::Types::Remove_const<T>::type *>(msg + offs) = v;
00120     return offs + sizeof(T);
00121 }
00122
00134 template<typename T>
00135 inline int msg_get(char *msg, unsigned offs, unsigned limit, T &v) noexcept
00136 {
00137     offs = align_to<T>(offs);
00138     if (L4_UNLIKELY(!check_size<T>(offs, limit)))
00139         return -L4_EMMSGTOOSHORT;
00140     v = *reinterpret_cast<T *>(msg + offs);
00141     return offs + sizeof(T);
00142 }
00143
00145 struct Dir_in { typedef Dir_in type;   typedef Dir_in dir; };
00147 struct Dir_out { typedef Dir_out type; typedef Dir_out dir; };
00148
00150 struct Cls_data { typedef Cls_data type;   typedef Cls_data cls; };
00152 struct Cls_item { typedef Cls_item type;   typedef Cls_item cls; };
00154 struct Cls_buffer { typedef Cls_buffer type; typedef Cls_buffer cls; };
00155
00156 // Typical combinations
00158 struct Do_in_data : Dir_in, Cls_data {};
00160 struct Do_out_data : Dir_out, Cls_data {};
00162 struct Do_in_items : Dir_in, Cls_item {};
00164 struct Do_out_items : Dir_out, Cls_item {};
00166 struct Do_rcv_buffers : Dir_in, Cls_buffer {};
00167
00168 // implementation details
00169 namespace Detail {
00170

```

```

00171 template<typename T> struct _Plain
00172 {
00173     typedef T type;
00174     static T deref(T v) noexcept { return v; }
00175 };
00176
00177 template<typename T> struct _Plain<T *>
00178 {
00179     typedef T type;
00180     static T &deref(T *v) noexcept { return *v; }
00181 };
00182
00183 template<typename T> struct _Plain<T &>
00184 {
00185     typedef T type;
00186     static T &deref(T &v) noexcept { return v; }
00187 };
00188
00189
00190 template<typename T> struct _Plain<T const *>
00191 {
00192     typedef T type;
00193     static T const &deref(T const *v) noexcept { return *v; }
00194 };
00195
00196 template<typename T> struct _Plain<T const &>
00197 {
00198     typedef T type;
00199     static T const &deref(T const &v) noexcept { return v; }
00200 };
00201
00202
00210 template<typename MTYPE, typename DIR, typename CLASS> struct Clnt_val_ops;
00211
00212 template<typename T> struct Clnt_noops
00213 {
00214     template<typename A, typename B>
00215     static constexpr int to_msg(char *, unsigned offset, unsigned, T, A, B) noexcept
00216     { return offset; }
00217
00218     template<typename A, typename B>
00219     static constexpr int from_msg(char *, unsigned offset, unsigned, long, T const &, A, B) noexcept
00220     { return offset; }
00221 };
00222
00223
00224 template<typename T> struct Svr_noops
00225 {
00226     template<typename A, typename B>
00227     static constexpr int from_svr(char *, unsigned offset, unsigned, long, T, A, B) noexcept
00228     { return offset; }
00229
00230     template<typename A, typename B>
00231     static constexpr int to_svr(char *, unsigned offset, unsigned, T, A, B) noexcept
00232     { return offset; }
00233 };
00234
00235
00236 template<typename MTYPE, typename CLASS>
00237 struct Clnt_val_ops<MTYPE, Dir_in, CLASS> : Clnt_noops<MTYPE>
00238 {
00239     using Clnt_noops<MTYPE>::to_msg;
00240     static int to_msg(char *msg, unsigned offset, unsigned limit,
00241                       MTYPE arg, Dir_in, CLASS) noexcept
00242     { return msg_add<MTYPE>(msg, offset, limit, arg); }
00243 };
00244
00245
00246
00247 template<typename MTYPE, typename CLASS>
00248 struct Clnt_val_ops<MTYPE, Dir_out, CLASS> : Clnt_noops<MTYPE>
00249 {
00250     using Clnt_noops<MTYPE>::from_msg;
00251     static int from_msg(char *msg, unsigned offset, unsigned limit, long,
00252                         MTYPE &arg, Dir_out, CLASS) noexcept
00253     { return msg_get<MTYPE>(msg, offset, limit, arg); }
00254 };
00255
00256
00264 template<typename MTYPE, typename DIR, typename CLASS> struct Svr_val_ops;
00265
00266 template<typename MTYPE, typename CLASS>
00267 struct Svr_val_ops<MTYPE, Dir_in, CLASS> : Svr_noops<MTYPE>
00268 {
00269     using Svr_noops<MTYPE>::to_svr;
00270     static int to_svr(char *msg, unsigned offset, unsigned limit,
00271                       MTYPE &arg, Dir_in, CLASS) noexcept
00272     { return msg_get<MTYPE>(msg, offset, limit, arg); }
00273 };
00274
00275
00276 template<typename MTYPE, typename CLASS>

```

```

00277 struct Svr_val_ops<MTYPE, Dir_out, CLASS> : Svr_noops<MTYPE>
00278 {
00279     using Svr_noops<MTYPE>::to_svr;
00280     static int to_svr(char *, unsigned offs, unsigned limit,
00281                     MTYPE &, Dir_out, CLASS) noexcept
00282     {
00283         offs = align_to<MTYPE>(offs);
00284         if (L4_UNLIKELY(!check_size<MTYPE>(offs, limit)))
00285             return -L4_MSGTOOLONG;
00286         return offs + sizeof(MTYPE);
00287     }
00288
00289     using Svr_noops<MTYPE>::from_svr;
00290     static int from_svr(char *msg, unsigned offset, unsigned limit, long,
00291                       MTYPE arg, Dir_out, CLASS) noexcept
00292     { return msg_add<MTYPE>(msg, offset, limit, arg); }
00293 };
00294
00295 template<typename T> struct Elem
00296 {
00297     typedef T arg_type;
00298     typedef T svr_type;
00300     typedef T svr_arg_type; // might by const & (depending on the size)
00301
00302     enum { Is_optional = false };
00303 };
00304
00305 template<typename T> struct Elem<T &>
00306 {
00307     typedef T &arg_type;
00308     typedef T svr_type;
00309     typedef T &svr_arg_type;
00310     enum { Is_optional = false };
00311 };
00312
00313 template<typename T> struct Elem<T const &>
00314 {
00315     typedef T const &arg_type;
00316     typedef T svr_type;
00317     // as the RPC uses a const reference we use it here too,
00318     // we could also use pass by value depending on the size
00319     typedef T const &svr_arg_type;
00320     enum { Is_optional = false };
00321 };
00322
00323 template<typename T> struct Elem<T *> : Elem<T &>
00324 {
00325     typedef T *arg_type;
00326 };
00327
00328 template<typename T> struct Elem<T const *> : Elem<T const &>
00329 {
00330     typedef T const *arg_type;
00331 };
00332
00333 template<typename T> struct Is_valid_rpc_type : L4::Types::True {};
00334
00335 // Static assertions outside functions work only properly from C++11
00336 // onwards. On earlier version make sure the compiler fails on an ugly
00337 // undefined struct instead.
00338 template<typename T, bool B> struct Error_invalid_rpc_parameter_used;
00339 template<typename T> struct Error_invalid_rpc_parameter_used<T, true> {};
00340
00341 #if __cplusplus >= 201103L
00342 template<typename T>
00343 struct _Elem : Elem<T>
00344 {
00345     static_assert(Is_valid_rpc_type<T>::value,
00346                 "L4::Rpc::Msg::_Elem<T>: type T is not a valid RPC parameter type.");
00347 };
00348 #else
00349 template<typename T>
00350 struct _Elem : Elem<T>, Error_invalid_rpc_parameter_used<T, Is_valid_rpc_type<T>::value>
00351 {};
00352 #endif
00353
00354 template<typename T> struct Class : Cls_data {};
00355 template<typename T> struct Direction : Dir_in {};
00356 template<typename T> struct Direction<T const &> : Dir_in {};
00357 template<typename T> struct Direction<T const *> : Dir_in {};
00358 template<typename T> struct Direction<T &> : Dir_out {};
00359 template<typename T> struct Direction<T *> : Dir_out {};
00360
00361 template<typename T> struct _Clnt_noops :

```

```

00370     Clnt_noops<typename Detail::_Plain<typename _Elem<T>::arg_type>::type>
00371 {};
00372
00373 namespace Detail {
00374
00375 template<typename T, typename DIR, typename CLASS>
00376 struct _Clnt_val_ops :
00377     Clnt_val_ops<typename Detail::_Plain<T>::type, DIR, CLASS> {};
00378
00379 template<typename T,
00380         typename ELEM = _Elem<T>,
00381         typename CLNT_OPS = _Clnt_val_ops<typename ELEM::arg_type,
00382         typename Direction<T>::type,
00383         typename Class<typename Detail::_Plain<T>::type>::type>
00384         >
00385 struct _Clnt_xmit : CLNT_OPS {};
00386
00387 template<typename T,
00388         typename ELEM = _Elem<T>,
00389         typename SVR_OPS = Svr_val_ops<typename ELEM::svr_type,
00390         typename Direction<T>::type,
00391         typename Class<typename Detail::_Plain<T>::type>::type>
00392         >
00393 struct _Svr_xmit : SVR_OPS {};
00394
00395 } //namespace Detail
00396 template<typename T> struct Clnt_xmit : Detail::_Clnt_xmit<T> {};
00397 template<typename T> struct Svr_xmit : Detail::_Svr_xmit<T> {};
00398
00399 }}} // namespace Msg, Ipc, L4
00400
00401

```

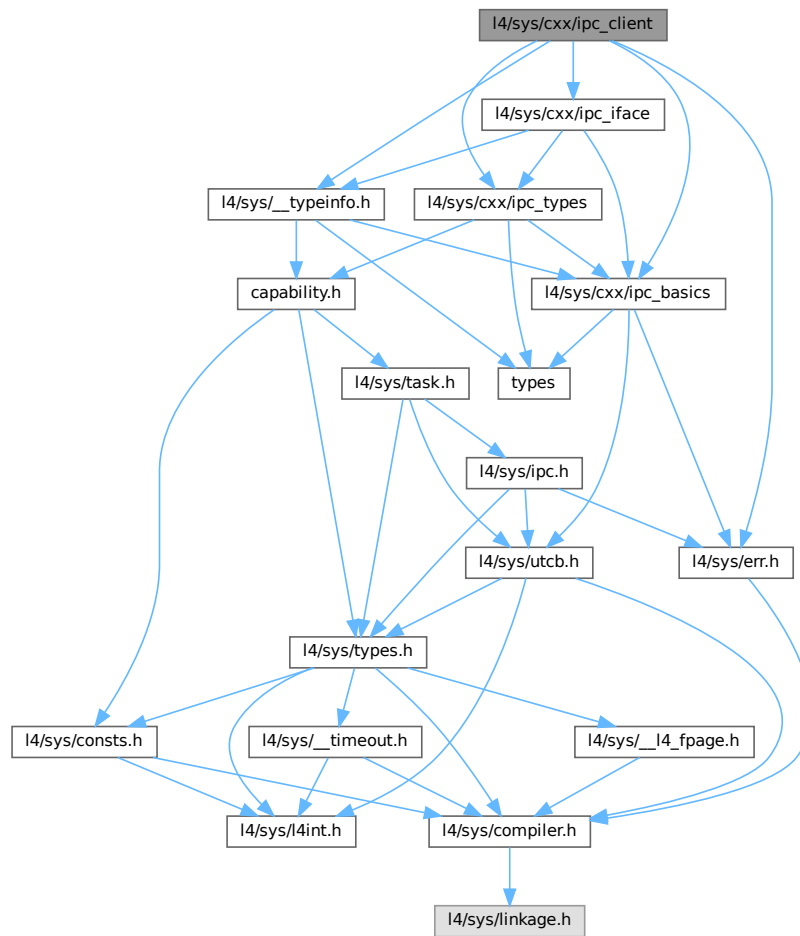
## 17.486 l4/sys/cxx/ipc\_client File Reference

```

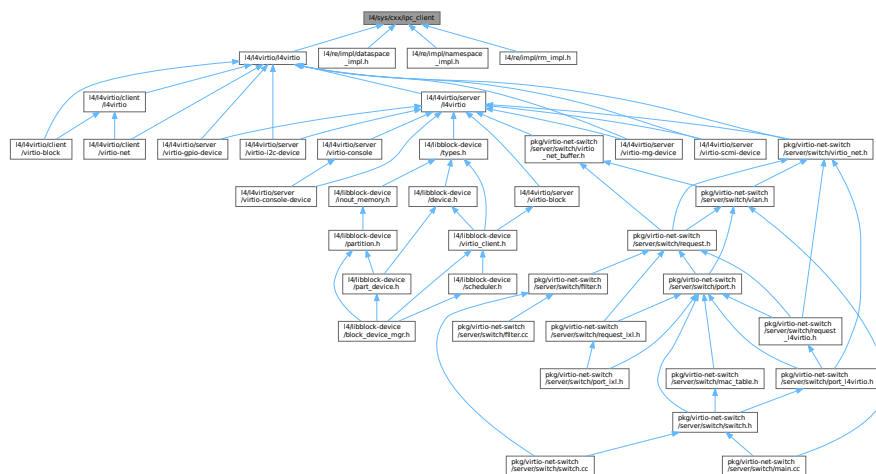
#include <l4/sys/cxx/ipc_basics>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/__typeinfo.h>
#include <l4/sys/err.h>

```

Include dependency graph for ipc\_client:



This graph shows which files directly or indirectly include this file:





## Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*
- namespace [L4::ipc](#)  
*IPC related functionality.*
- namespace [L4::ipc::Msg](#)  
*IPC Message related functionality.*

## Macros

- `#define L4_RPC_DEF(name)`  
*Generate the definition of an RPC stub.*

## 17.486.1 Macro Definition Documentation

### 17.486.1.1 L4\_RPC\_DEF

```
#define L4_RPC_DEF (
    name)
```

#### Value:

```
template struct L4::Ipcc::Msg::Rpc_call \
    <name##_t, name##_t::class_type, name##_t::ipc_type, name##_t::flags_type>
```

Generate the definition of an RPC stub.

#### Parameters

<i>name</i>	The fully qualified method name to be implemented, this means <code>class::method</code> .
-------------	--------------------------------------------------------------------------------------------

This macro generates the definition (implementation) for the given RPC interface method.

Definition at line 32 of file [ipc\\_client](#).

## 17.487 ipc\_client

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008 #pragma GCC system_header
00009
00010 #include <l4/sys/cxx/ipc_basics>
00011 #include <l4/sys/cxx/ipc_types>
00012 #include <l4/sys/cxx/ipc_iface>
00013 #include <l4/sys/__typeinfo.h>
00014 #include <l4/sys/err.h>
00015
00019
```

```

00020 namespace L4 { namespace Ipc { namespace Msg {
00021 //-----
00022
00032 #define L4_RPC_DEF(name) \
00033     template struct L4::Ipc::Msg::Rpc_call \
00034         <name##_t, name##_t::class_type, name##_t::ipc_type, name##_t::flags_type>
00035
00036
00038 //-----
00039 //Implementation of the RPC call
00040 template<typename OP, typename C, typename FLAGS, typename R, typename ...ARGS>
00041 R L4_EXPORT
00042 Rpc_call<OP, C, R (ARGS...), FLAGS>::
00043     call(L4::Cap<C> cap, typename _Elem<ARGS>::arg_type ...a, l4_utcb_t *utcb) noexcept
00044 {
00045     return Rpc_inline_call<OP, C, R (ARGS...), FLAGS>::call(cap, a..., utcb);
00046 }
00048
00049 } // namespace Msg
00050 } // namespace Ipc
00051 } // namespace L4
00052

```

## 17.488 ipc\_epiface

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014-2015 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008 #pragma GCC system_header
00009
00010 #include "capability.h"
00011 #include "ipc_server"
00012 #include "ipc_string"
00013 #include <l4/sys/types.h>
00014 #include <l4/sys/utcb.h>
00015 #include <l4/sys/__typeinfo.h>
00016 #include <l4/sys/meta>
00017 #include <l4/cxx/type_traits>
00018
00019 namespace L4 {
00020
00021 // forward for Irqep_t
00022 class Irq;
00023 class Rcv_endpoint;
00024
00025 namespace Ipc_svr {
00026
00027 class Timeout;
00028
00036 class Server_iface
00037 {
00038 private:
00039     Server_iface(Server_iface const &);
00040     Server_iface const &operator = (Server_iface const &);
00041
00042 public:
00044     typedef L4::Type_info::Demand Demand;
00045
00046     Server_iface(Server_iface &&) = delete;
00047     Server_iface &operator = (Server_iface &&) = delete;
00048
00050     Server_iface() {}
00051
00052     // Destroy the server interface
00053     virtual ~Server_iface() = 0;
00054
00064     virtual int alloc_buffer_demand(Demand const &demand) = 0;
00065
00074     virtual L4::Cap<void> get_rcv_cap(int index) const = 0;
00075
00084     virtual int realloc_rcv_cap(int index) = 0;
00085
00093     virtual int add_timeout(Timeout *timeout, l4_kernel_clock_t time) = 0;
00094
00100     virtual int remove_timeout(Timeout *timeout) = 0;
00101
00113     template<typename T>
00114     L4::Cap<T> rcv_cap(int index) const
00115     { return L4::cap_cast<T>(get_rcv_cap(index)); }

```

```

00116
00126 L4::Cap<void> rcv_cap(int index) const
00127 { return get_rcv_cap(index); }
00128 };
00129
00130 inline Server_iface::~Server_iface() {}
00131
00132 } // namespace Ipc_svr
00133
00145 struct Epiface
00146 {
00147     Epiface(Epiface const &) = delete;
00148     Epiface &operator = (Epiface const &) = delete;
00149
00151     typedef Ipc_svr::Server_iface Server_iface;
00153     typedef Ipc_svr::Server_iface::Demand Demand;
00154
00155     class Stored_cap : public Cap<void>
00156     {
00157     private:
00158         enum { Managed = 0x10 };
00159
00160     public:
00161         Stored_cap() = default;
00162         Stored_cap(Cap<void> const &c, bool managed = false)
00163             : Cap<void>((c.cap() & L4_CAP_MASK) | (managed ? Managed : 0))
00164         {
00165             static_assert (!(L4_CAP_MASK & Managed), "conflicting bits used...");
00166         }
00167
00168         bool managed() const { return cap() & Managed; }
00169     };
00170
00172     Epiface() : _data(0) {}
00173
00186     virtual l4_msgtag_t dispatch(l4_msgtag_t tag, unsigned rights,
00187                                 l4_utcb_t *utcb) = 0;
00188
00195     virtual Demand get_buffer_demand() const = 0; //{ return Demand(0); }
00196
00198     virtual ~Epiface() = 0;
00199
00206     Stored_cap obj_cap() const { return _cap; }
00207
00213     Server_iface *server_iface() const { return _data; }
00214
00224     int set_server(Server_iface *srv, Cap<void> cap, bool managed = false)
00225     {
00226         if ((srv && cap) || (!srv && !cap))
00227         {
00228             _data = srv;
00229             _cap = Stored_cap(cap, managed);
00230             return 0;
00231         }
00232
00233         return -L4_EINVAL;
00234     }
00235
00239     void set_obj_cap(Cap<void> const &cap) { _cap = cap; }
00240
00241 private:
00242     Server_iface *_data;
00243     Stored_cap _cap;
00244 };
00245
00246 inline Epiface::~Epiface() {}
00247
00255 template<typename RPC_IFACE, typename BASE = Epiface>
00256 struct Epiface_t0 : BASE
00257 {
00259     typedef RPC_IFACE Interface;
00260
00262     typename Type_info::Demand get_buffer_demand() const
00263     { return typename Kobject_typeid<RPC_IFACE>::Demand(); }
00264
00269     Cap<RPC_IFACE> obj_cap() const
00270     { return L4::cap_cast<RPC_IFACE>(BASE::obj_cap()); }
00271 };
00272
00280 template<typename Derived, typename BASE = Epiface,
00281         bool = cxx::is_polymorphic<BASE>::value>
00282 struct Irqep_t : Epiface_t0<void, BASE>
00283 {
00284     l4_msgtag_t dispatch(l4_msgtag_t, unsigned, l4_utcb_t *) final
00285     {
00286         static_cast<Derived*>(this)->handle_irq();
00287         return l4_msgtag(-L4_ENOREPLY, 0, 0, 0);

```

```

00288     }
00289
00294     Cap<L4::Irq> obj_cap() const
00295     { return L4::cap_cast<L4::Irq>(BASE::obj_cap()); }
00296 };
00297
00298 template<typename Derived, typename BASE>
00299 struct Irqep_t<Derived, BASE, false> : Epiface_t0<void, BASE>
00300 {
00301     l4_msgtag_t dispatch(l4_msgtag_t, unsigned, l4_utcb_t *)
00302     {
00303         static_cast<Derived*>(this)->handle_irq();
00304         return l4_msgtag(~L4_ENOREPLY, 0, 0, 0);
00305     }
00306
00311     Cap<L4::Irq> obj_cap() const
00312     { return L4::cap_cast<L4::Irq>(BASE::obj_cap()); }
00313 };
00314
00322 class Registry_iface
00323 {
00324 public:
00325     virtual ~Registry_iface() = 0;
00326
00339     virtual L4::Cap<void>
00340     register_obj(L4::Epiface *o, char const *service) = 0;
00341
00356     virtual L4::Cap<void>
00357     register_obj(L4::Epiface *o) = 0;
00358
00372     virtual L4::Cap<L4::Irq> register_irq_obj(L4::Epiface *o) = 0;
00373
00386     virtual L4::Cap<L4::Rcv_endpoint>
00387     register_obj(L4::Epiface *o, L4::Cap<L4::Rcv_endpoint> ep) = 0;
00388
00401     virtual void
00402     unregister_obj(L4::Epiface *o, bool unmap = true) = 0;
00403 };
00404
00405 inline Registry_iface::~Registry_iface() {}
00406
00407 namespace Ipc {
00408 namespace Detail {
00409 using namespace L4::Typeid;
00410
00411 template<typename IFACE>
00412 struct Meta_svr
00413 {
00414     {
00415         long op_num_interfaces(L4::Meta::Rights)
00416         { return 1; }
00417
00418         long op_interface(L4::Meta::Rights, l4_umword_t ifx, long &proto, L4::Ipc::String<char> &name)
00419         {
00420             if (ifx > 0)
00421                 return -L4_ERANGE;
00422             proto = L4::kobject_typeid<IFACE>()->proto();
00423             if (auto *n = L4::kobject_typeid<IFACE>()->name())
00424                 name.copy_in(n);
00425
00426             return 0;
00427         }
00428
00429         long op_supports(L4::Meta::Rights, l4_mword_t proto)
00430         { return L4::kobject_typeid<IFACE>()->has_proto(proto); }
00431     };
00432
00433     template<typename IFACE, typename LIST>
00434     struct _Dispatch;
00435
00436     // No match dispatcher found
00437     template<typename IFACE>
00438     struct _Dispatch<IFACE, Iface_list_end>
00439     {
00440         template< typename THIS, typename A1, typename A2 >
00441         static l4_msgtag_t f(THIS *, l4_msgtag_t, A1, A2 &)
00442         { return l4_msgtag(~L4_EBADPROTO, 0, 0, 0); }
00443     };
00444
00445     // call matching p_dispatch() function
00446     template<typename IFACE, typename I, typename LIST >
00447     struct _Dispatch<IFACE, Iface_list<I, LIST> >
00448     {
00449         // special handling for the meta protocol, to avoid 'using' murx
00450         template< typename THIS >
00451         static l4_msgtag_t _f(THIS *, l4_msgtag_t tag, unsigned r,
00452                               l4_utcb_t *utcb, True::type)

```

```

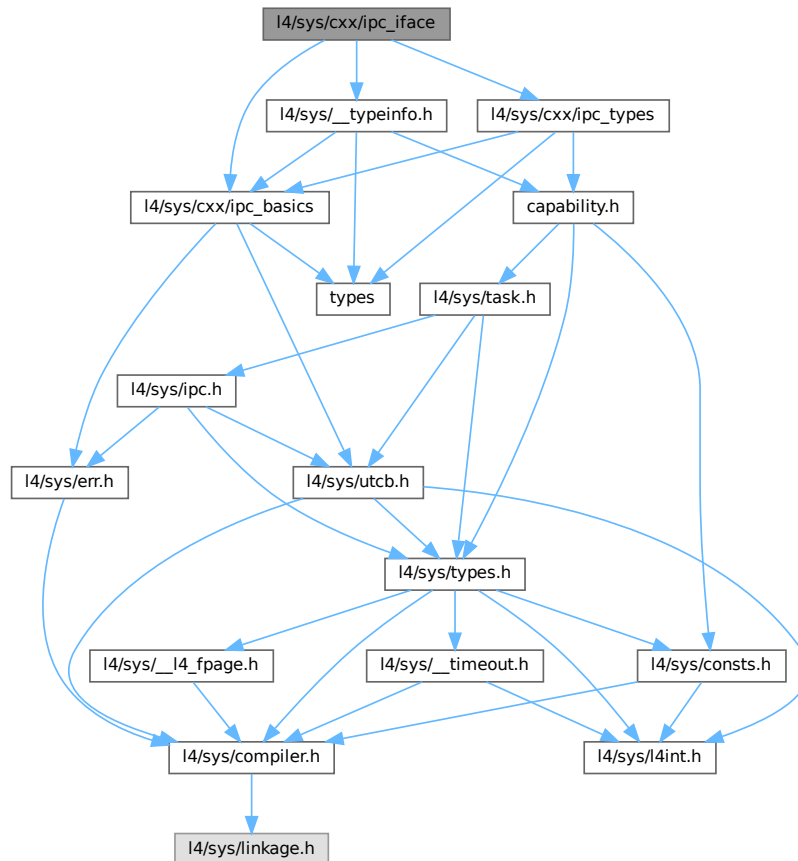
00453 {
00454     using L4::IpC::Msg::dispatch_call;
00455     typedef L4::Meta::Rpc Meta;
00456     typedef Meta_svr<IFACE> Msvr;
00457     return dispatch_call<Meta>(static_cast<Msvr*>(nullptr), utcb, tag, r);
00458 }
00459
00460 // normal dispatch to the op_<func> methods of \a self.
00461 template< typename THIS >
00462 static l4_msgtag_t _f(THIS *self, l4_msgtag_t t, unsigned r,
00463                      l4_utcb_t *utcb, False::type)
00464 {
00465     using L4::IpC::Msg::dispatch_call;
00466     return dispatch_call<typename I::iface_type::Rpc>(self, utcb, t, r);
00467 }
00468
00469 // dispatch function with switch for meta protocol
00470 template< typename THIS >
00471 static l4_msgtag_t f(THIS *self, l4_msgtag_t tag, unsigned r,
00472                     l4_utcb_t *utcb)
00473 {
00474     if (I::Proto == tag.label())
00475         return _f(self, tag, r, utcb,
00476                  Bool<I::Proto == static_cast<long>(L4_PROTO_META)>());
00477
00478     return _Dispatch<IFACE, typename LIST::type>::f(self, tag, r, utcb);
00479 }
00480 };
00481
00482 template<typename IFACE>
00483 struct Dispatch :
00484     _Dispatch<IFACE, typename L4::Kobject_typeid<IFACE>::Iface_list::type>
00485 {};
00486
00487 } // namespace Detail
00488
00489 template<typename EPIFACE>
00490 struct Dispatch : Detail::Dispatch<typename EPIFACE::Interface>
00491 {};
00492
00493 } // namespace IpC
00494
00501 template<typename Derived, typename IFACE, typename BASE = L4::Epiface,
00502         bool = cxx::is_polymorphic<BASE>::value>
00503 struct Epiface_t : Epiface_t0<IFACE, BASE>
00504 {
00505     l4_msgtag_t
00506     dispatch(l4_msgtag_t tag, unsigned rights, l4_utcb_t *utcb) final
00507     {
00508         typedef IpC::Dispatch<Derived> Dispatch;
00509         return Dispatch::f(static_cast<Derived*>(this), tag, rights, utcb);
00510     }
00511 };
00512
00513 template<typename Derived, typename IFACE, typename BASE>
00514 struct Epiface_t<Derived, IFACE, BASE, false> : Epiface_t0<IFACE, BASE>
00515 {
00516     l4_msgtag_t
00517     dispatch(l4_msgtag_t tag, unsigned rights, l4_utcb_t *utcb)
00518     {
00519         typedef IpC::Dispatch<Derived> Dispatch;
00520         return Dispatch::f(static_cast<Derived*>(this), tag, rights, utcb);
00521     }
00522 };
00523
00529 class Basic_registry
00530 {
00531 public:
00532     typedef Epiface Value;
00533     static Value *find(l4_umword_t label)
00534     { return reinterpret_cast<Value*>(label & ~3UL); }
00540
00554     static l4_msgtag_t dispatch(l4_msgtag_t tag, l4_umword_t label,
00555                                l4_utcb_t *utcb)
00556     {
00557         return find(label)->dispatch(tag, label, utcb);
00558     }
00559 };
00560
00561
00562 } // namespace L4

```

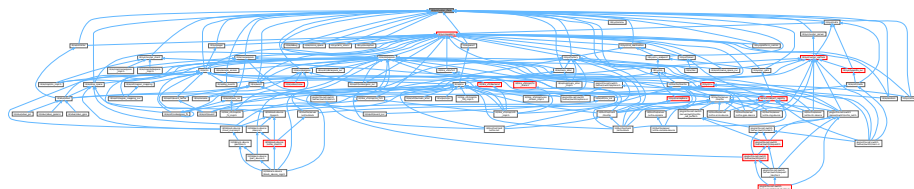
## 17.489 l4/sys/cxx/ipc\_iface File Reference

Interface Definition Language.

```
#include <l4/sys/cxx/ipc_basics>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/__typeinfo.h>
Include dependency graph for ipc_iface:
```



This graph shows which files directly or indirectly include this file:



### Data Structures

- struct [L4::ipc::Call](#)

- RPC attribute for a standard RPC call.*
  - struct [L4::lpc::Call\\_zero\\_send\\_timeout](#)
*RPC attribute for an RPC call, with zero send timeout.*
  - struct [L4::lpc::Call\\_t< RIGHTS >](#)
*RPC attribute for an RPC call with required rights.*
  - struct [L4::lpc::Send\\_only](#)
*RPC attribute for a send-only RPC.*

## Namespaces

- namespace [L4](#)
  - [L4](#) low-level kernel interface.
- namespace [L4::lpc](#)
  - IPC related functionality.
- namespace [L4::lpc::Msg](#)
  - IPC Message related functionality.

## Macros

- #define [L4\\_INLINE\\_RPC\\_NF](#)(res, name, args...)
  - Define an inline RPC call type (the type only, no callable).
- #define [L4\\_INLINE\\_RPC\\_NF\\_OP](#)(op, res, name, args...)
  - Define an inline RPC call type with specific opcode (the type only, no callable).
- #define [L4\\_INLINE\\_RPC](#)(res, name, args, attr...)
  - Define an inline RPC call (type and callable).
- #define [L4\\_INLINE\\_RPC\\_OP](#)(op, res, name, args, attr...)
  - Define an inline RPC call with specific opcode (type and callable).
- #define [L4\\_RPC\\_NF](#)(res, name, args...)
  - Define an RPC call type (the type only, no callable).
- #define [L4\\_RPC\\_NF\\_OP](#)(op, res, name, args...)
  - Define an RPC call type with specific opcode (the type only, no callable).
- #define [L4\\_RPC](#)(res, name, args, attr...)
  - Define an RPC call (type and callable).
- #define [L4\\_RPC\\_OP](#)(op, res, name, args, attr...)
  - Define an RPC call with specific opcode (type and callable).

## 17.489.1 Detailed Description

Interface Definition Language.

See also

[L4\\_RPC](#), [L4\\_INLINE\\_RPC](#), [L4::lpc::Call](#) [L4::lpc::Send\\_only](#), [L4::lpc::Msg::Rpc\\_call](#), [L4::lpc::Msg::Rpc\\_↵](#)  
[inline\\_call](#)

Definition in file [ipc\\_iface](#).

## 17.489.2 Macro Definition Documentation

### 17.489.2.1 L4\_INLINE\_RPC

```
#define L4_INLINE_RPC(
    res,
    name,
    args,
    attr...)
```

#### Value:

`res name args`

Define an inline RPC call (type and callable).

#### Parameters

<i>res</i>	The result type of the RPC call
<i>name</i>	The name of the function ( <code>name_t</code> is used for the type.)
<i>args</i>	The argument list of the RPC function.
<i>attr</i>	Optional RPC attributes ( <a href="#">L4::lpc::Call</a> , <a href="#">L4::lpc::Call_t</a> etc.).

#### Examples

[examples/clntsrv/src/shared.h](#).

Definition at line 482 of file [ipc\\_iface](#).

Referenced by [L4Re::Mem\\_alloc::info\(\)](#).

### 17.489.2.2 L4\_INLINE\_RPC\_NF

```
#define L4_INLINE_RPC_NF(
    res,
    name,
    args...)
```

#### Value:

```
struct name##_t : L4::Ipc::Msg::Rpc_inline_call<name##_t, Class, res args> \
{ \
    typedef L4::Ipc::Msg::Rpc_inline_call<name##_t, Class, res args> type; \
    L4_INLINE_RPC_SRV_FORWARD(name); \
}
```

Define an inline RPC call type (the type only, no callable).

#### Parameters

<i>res</i>	The result type of the RPC call
<i>name</i>	The name of the function ( <code>name_t</code> is used for the type.)
<i>args</i>	The argument list of the RPC function, and RPC attributes ( <a href="#">L4::lpc::Call</a> , <a href="#">L4::lpc::Call_t</a> etc.).

Stubs generated by this macro can be used explicitly in custom wrapper methods that need to use the underlying RPC code and provide some higher level abstraction, for example with default arguments or extra argument conversion.

Definition at line 453 of file [ipc\\_iface](#).



**17.489.2.3 L4\_INLINE\_RPC\_NF\_OP**

```
#define L4_INLINE_RPC_NF_OP (
    op,
    res,
    name,
    args...)

```

**Value:**

```
struct name##_t : L4::Ipc::Msg::Rpc_inline_call<name##_t, Class, res args> \
{ \
    typedef L4::Ipc::Msg::Rpc_inline_call<name##_t, Class, res args> type; \
    enum { Opcode = (op) }; \
    L4_INLINE_RPC_SRV_FORWARD(name); \
}

```

Define an inline RPC call type with specific opcode (the type only, no callable).

**Parameters**

<i>op</i>	The opcode number for this function
<i>res</i>	The result type of the RPC call
<i>name</i>	The name of the function ( <i>name_t</i> is used for the type.)
<i>args</i>	The argument list of the RPC function, and RPC attributes ( <a href="#">L4::Ipc::Call</a> , <a href="#">L4::Ipc::Call_t</a> etc.).

Stubs generated by this macro can be used explicitly in custom wrapper methods that need to use the underlying RPC code and provide some higher level abstraction, for example with default arguments or extra argument conversion.

Definition at line [466](#) of file [ipc\\_iface](#).

**17.489.2.4 L4\_INLINE\_RPC\_OP**

```
#define L4_INLINE_RPC_OP (
    op,
    res,
    name,
    args,
    attr...)

```

**Value:**

```
res name args
```

Define an inline RPC call with specific opcode (type and callable).

**Parameters**

<i>op</i>	The opcode number for this function
<i>res</i>	The result type of the RPC call
<i>name</i>	The name of the function ( <i>name_t</i> is used for the type.)
<i>args</i>	The argument list of the RPC function.
<i>attr</i>	Optional RPC attributes ( <a href="#">L4::Ipc::Call</a> , <a href="#">L4::Ipc::Call_t</a> etc.).

Definition at line [497](#) of file [ipc\\_iface](#).

### 17.489.2.5 L4\_RPC

```
#define L4_RPC(
    res,
    name,
    args,
    attr...)

```

#### Value:

res name args

Define an RPC call (type and callable).

#### Parameters

<i>res</i>	The result type of the RPC call
<i>name</i>	The name of the function ( <code>name_t</code> is used for the type.)
<i>args</i>	The argument list of the RPC function.
<i>attr</i>	Optional RPC attributes ( <a href="#">L4::lpc::Call</a> , <a href="#">L4::lpc::Call_t</a> etc.).

Definition at line 541 of file [ipc\\_iface](#).

Referenced by [L4Re::Dataspace::info\(\)](#).

### 17.489.2.6 L4\_RPC\_NF

```
#define L4_RPC_NF(
    res,
    name,
    args...)

```

#### Value:

```
struct name##_t : L4::Ipc::Msg::Rpc_call<name##_t, Class, res args>
{
    typedef L4::Ipc::Msg::Rpc_call<name##_t, Class, res args> type;
    L4_INLINE_RPC_SRV_FORWARD(name);
}

```

Define an RPC call type (the type only, no callable).

#### Parameters

<i>res</i>	The result type of the RPC call
<i>name</i>	The name of the function ( <code>name_t</code> is used for the type.)
<i>args</i>	The argument list of the RPC function, and RPC attributes ( <a href="#">L4::lpc::Call</a> , <a href="#">L4::lpc::Call_t</a> etc.).

Definition at line 510 of file [ipc\\_iface](#).

Referenced by [L4Re::Dataspace::info\(\)](#).

**17.489.2.7 L4\_RPC\_NF\_OP**

```
#define L4_RPC_NF_OP (
    op,
    res,
    name,
    args...)

```

**Value:**

```
struct name##_t : L4::Ipc::Msg::Rpc_call<name##_t, Class, res args>
{
    typedef L4::Ipc::Msg::Rpc_call<name##_t, Class, res args> type;
    enum { Opcode = (op) };
    L4_INLINE_RPC_SRV_FORWARD(name);
}

```

```
\\
\\
\\
\\

```

Define an RPC call type with specific opcode (the type only, no callable).

**Parameters**

<i>op</i>	The opcode number for this function
<i>res</i>	The result type of the RPC call
<i>name</i>	The name of the function ( <i>name_t</i> is used for the type.)
<i>args</i>	The argument list of the RPC function, and RPC attributes ( <a href="#">L4::Ipc::Call</a> , <a href="#">L4::Ipc::Call_t</a> etc.).

Definition at line 525 of file [ipc\\_iface](#).

**17.489.2.8 L4\_RPC\_OP**

```
#define L4_RPC_OP (
    op,
    res,
    name,
    args,
    attr...)

```

**Value:**

```
res name args
```

Define an RPC call with specific opcode (type and callable).

**Parameters**

<i>op</i>	The opcode number for this function
<i>res</i>	The result type of the RPC call
<i>name</i>	The name of the function ( <i>name_t</i> is used for the type.)
<i>args</i>	The argument list of the RPC function.
<i>attr</i>	Optional RPC attributes ( <a href="#">L4::Ipc::Call</a> , <a href="#">L4::Ipc::Call_t</a> etc.).

Definition at line 556 of file [ipc\\_iface](#).

## 17.490 ipc\_iface

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008 #pragma GCC system_header
00009
00010 #include <l4/sys/cxx/ipc_basics>
00011 #include <l4/sys/cxx/ipc_types>
00012 #include <l4/sys/__typeinfo.h>
00013
00022
00219
00220 // TODO: add some more documentation
00221 namespace L4 { namespace Ipc {
00222
00239 struct L4_EXPORT Call
00240 {
00241     enum { Is_call = true };
00242     enum { Rights = 0 };
00243     static l4_timeout_t timeout() { return L4_IPC_NEVER; }
00244 };
00245
00249 struct L4_EXPORT Call_zero_send_timeout : Call
00250 {
00251     static l4_timeout_t timeout() { return L4_IPC_SEND_TIMEOUT_0; }
00252 };
00253
00269 template<unsigned RIGHTS>
00270 struct L4_EXPORT Call_t : Call
00271 {
00272     enum { Rights = RIGHTS };
00273 };
00274
00287 struct L4_EXPORT Send_only
00288 {
00289     enum { Is_call = false };
00290     enum { Rights = 0 };
00291     static l4_timeout_t timeout() { return L4_IPC_NEVER; }
00292 };
00293
00294 namespace Msg {
00295
00306 template<typename OP, typename CLASS, typename SIG, typename FLAGS = Call>
00307 struct L4_EXPORT Rpc_inline_call;
00308
00313 template<typename OP, typename CLASS, typename FLAGS, typename R,
00314         typename ...ARGS>
00315 struct L4_EXPORT Rpc_inline_call<OP, CLASS, R (ARGS...), FLAGS>
00316 {
00317     template<typename T> struct Result { typedef T result_type; };
00318     enum
00319     {
00320         Return_tag = L4::Types::Same<R, l4_msgtag_t>::value
00321     };
00322
00324     typedef Rpc_inline_call type;
00326     typedef OP op_type;
00328     typedef CLASS class_type;
00330     typedef typename Result<R>::result_type result_type;
00332     typedef R ipc_type (ARGS...);
00334     typedef result_type func_type (typename _Elem<ARGS>::arg_type...);
00335
00337     typedef FLAGS flags_type;
00338
00339     template<typename RES>
00340     static typename L4::Types::Enable_if< Return_tag, RES >::type
00341     return_err(long err) noexcept { return l4_msgtag(err, 0, 0, 0); }
00342
00343     template<typename RES>
00344     static typename L4::Types::Enable_if< Return_tag, RES >::type
00345     return_ipc_err(l4_msgtag_t tag, l4_utcb_t const *) noexcept { return tag; }
00346
00347     template<typename RES>
00348     static typename L4::Types::Enable_if< Return_tag, RES >::type
00349     return_code(l4_msgtag_t tag) noexcept { return tag; }
00350
00351     template<typename RES>
00352     static typename L4::Types::Enable_if< !Return_tag, RES >::type
00353     return_err(long err) noexcept { return err; }

```

```

00354
00355     template<typename RES>
00356     static typename L4::Types::Enable_if< !Return_tag, RES >::type
00357     return_ipc_err(l4_msgtag_t, l4_utcb_t *utcb) noexcept
00358     { return l4_ipc_to_errno(l4_ipc_error_code(utcb)); }
00359
00360     template<typename RES>
00361     static typename L4::Types::Enable_if< !Return_tag, RES >::type
00362     return_code(l4_msgtag_t tag) noexcept { return tag.label(); }
00363
00364     static R call(L4::Cap<class_type> cap,
00365                  typename _Elem<ARGS>::arg_type ...a,
00366                  l4_utcb_t *utcb = l4_utcb()) noexcept;
00367 };
00368
00373 template<typename OP, typename CLASS, typename SIG, typename FLAGS = Call>
00374 struct L4_EXPORT Rpc_call;
00375
00383 template<typename IPC, typename SIG> struct _Call;
00384
00386 template<typename IPC, typename R, typename ...ARGS>
00387 struct _Call<IPC, R (ARGS...)>
00388 {
00389 public:
00390     typedef typename IPC::class_type class_type;
00391     typedef typename IPC::result_type result_type;
00392
00393 private:
00394     L4::Cap<class_type> cap() const noexcept
00395     {
00396         return L4::Cap<class_type>(reinterpret_cast<l4_cap_idx_t>(this)
00397                                     & L4_CAP_MASK);
00398     }
00399
00400 public:
00402     result_type operator () (ARGS ...a, l4_utcb_t *utcb = l4_utcb()) const noexcept
00403     { return IPC::call(cap(), a..., utcb); }
00404 };
00405
00412 template<typename IPC> struct Call : _Call<IPC, typename IPC::func_type> {};
00413
00418 template<typename OP,
00419           typename CLASS,
00420           typename FLAGS,
00421           typename R,
00422           typename ...ARGS>
00423 struct L4_EXPORT Rpc_call<OP, CLASS, R (ARGS...), FLAGS> :
00424     Rpc_inline_call<OP, CLASS, R (ARGS...), FLAGS>
00425 {
00426     static R call(L4::Cap<CLASS> cap,
00427                  typename _Elem<ARGS>::arg_type ...a,
00428                  l4_utcb_t *utcb = l4_utcb()) noexcept;
00429 };
00430
00431 #define L4_INLINE_RPC_SRV_FORWARD(name)
00432     template<typename OBJ> struct fwd
00433     {
00434         OBJ *o;
00435         fwd(OBJ *o) noexcept : o(o) {}
00436         template<typename ...ARGS> long call(ARGS ...a) noexcept (noexcept(o->op_##name(a...)))
00437         { return o->op_##name(a...); }
00438     }
00439
00440
00453 #define L4_INLINE_RPC_NF(res, name, args...)
00454     struct name##_t : L4::IpC::Msg::Rpc_inline_call<name##_t, Class, res args>
00455     {
00456         typedef L4::IpC::Msg::Rpc_inline_call<name##_t, Class, res args> type;
00457         L4_INLINE_RPC_SRV_FORWARD(name);
00458     }
00459
00466 #define L4_INLINE_RPC_NF_OP(op, res, name, args...)
00467     struct name##_t : L4::IpC::Msg::Rpc_inline_call<name##_t, Class, res args>
00468     {
00469         typedef L4::IpC::Msg::Rpc_inline_call<name##_t, Class, res args> type;
00470         enum { Opcode = (op) };
00471         L4_INLINE_RPC_SRV_FORWARD(name);
00472     }
00473
00474 #ifndef DOXYGEN
00482 #define L4_INLINE_RPC(res, name, args, attr...) res name args
00483 #else
00484 #define L4_INLINE_RPC(res, name, args...)
00485     L4_INLINE_RPC_NF(res, name, args); L4::IpC::Msg::Call<name##_t> name
00486 #endif
00487
00488 #ifndef DOXYGEN

```

```

00497 #define L4_INLINE_RPC_OP(op, res, name, args, attr...) res name args
00498 #else
00499 #define L4_INLINE_RPC_OP(op, res, name, args...) \
00500     L4_INLINE_RPC_NF_OP(op, res, name, args); L4::Rpc::Msg::Call<name##_t> name
00501 #endif
00502
00510 #define L4_RPC_NF(res, name, args...) \
00511     struct name##_t : L4::Rpc::Msg::Rpc_call<name##_t, Class, res args> \
00512     { \
00513         typedef L4::Rpc::Msg::Rpc_call<name##_t, Class, res args> type; \
00514         L4_INLINE_RPC_SRV_FORWARD(name); \
00515     }
00516
00525 #define L4_RPC_OP(op, res, name, args...) \
00526     struct name##_t : L4::Rpc::Msg::Rpc_call<name##_t, Class, res args> \
00527     { \
00528         typedef L4::Rpc::Msg::Rpc_call<name##_t, Class, res args> type; \
00529         enum { Opcode = (op) }; \
00530         L4_INLINE_RPC_SRV_FORWARD(name); \
00531     }
00532
00533 #ifdef DOXYGEN
00541 #define L4_RPC(res, name, args, attr...) res name args
00542 #else
00543 #define L4_RPC(res, name, args...) \
00544     L4_RPC_NF(res, name, args); L4::Rpc::Msg::Call<name##_t> name
00545 #endif
00546
00547 #ifdef DOXYGEN
00556 #define L4_RPC_OP(op, res, name, args, attr...) res name args
00557 #else
00558 #define L4_RPC_OP(op, res, name, args...) \
00559     L4_RPC_NF_OP(op, res, name, args); L4::Rpc::Msg::Call<name##_t> name
00560 #endif
00561
00562 namespace Detail {
00563
00572 template<typename ...ARGS>
00573 struct Buf
00574 {
00575 public:
00576     template<typename DIR>
00577     static constexpr int write(char *, int offset, int) noexcept
00578     { return offset; }
00579
00580     template<typename DIR>
00581     static constexpr int read(char *, int offset, int, long) noexcept
00582     { return offset; }
00583
00584     typedef void Base;
00585 };
00586
00587 template<typename A, typename ...M>
00588 struct Buf<A, M...> : Buf<M...>
00589 {
00590     typedef Buf<M...> Base;
00591
00592     typedef Clnt_xmit<A> xmit;
00593     typedef typename _Elem<A>::arg_type arg_type;
00594     typedef Detail::_Plain<arg_type> plain;
00595
00596     template<typename DIR>
00597     static int
00598     write(char *base, int offset, int limit,
00599           arg_type a, typename _Elem<M>::arg_type ...m) noexcept
00600     {
00601         offset = xmit::to_msg(base, offset, limit, plain::deref(a),
00602                               typename DIR::dir(), typename DIR::cls());
00603         return Base::template write<DIR>(base, offset, limit, m...);
00604     }
00605
00606     template<typename DIR>
00607     static int
00608     read(char *base, int offset, int limit, long ret,
00609          arg_type a, typename _Elem<M>::arg_type ...m) noexcept
00610     {
00611         int r = xmit::from_msg(base, offset, limit, ret, plain::deref(a),
00612                                typename DIR::dir(), typename DIR::cls());
00613         if (L4_LIKELY(r >= 0))
00614             return Base::template read<DIR>(base, r, limit, ret, m...);
00615
00616         if (_Elem<A>::Is_optional)
00617             return Base::template read<DIR>(base, offset, limit, ret, m...);
00618
00619         return r;
00620     }
00621 }

```

```

00621 };
00622
00623 template <typename ...ARGS> struct _Part
00624 {
00625     typedef Buf<ARGS...> Data;
00626
00627     template<typename DIR>
00628     static int write(void *b, int offset, int limit,
00629                     typename _Elem<ARGS>::arg_type ...m) noexcept
00630     {
00631         char *buf = static_cast<char *>(b);
00632         int r = Data::template write<DIR>(buf, offset, limit, m...);
00633         if (L4_LIKELY(r >= offset))
00634             return r - offset;
00635         return r;
00636     }
00637 }
00638
00639 template<typename DIR>
00640 static int read(void *b, int offset, int limit, long ret,
00641               typename _Elem<ARGS>::arg_type ...m) noexcept
00642 {
00643     char *buf = static_cast<char *>(b);
00644     int r = Data::template read<DIR>(buf, offset, limit, ret, m...);
00645     if (L4_LIKELY(r >= offset))
00646         return r - offset;
00647     return r;
00648 }
00649 };
00650
00651 template<typename IPC_TYPE, typename OPCODE = void>
00652 struct Part;
00653
00654 // The version without an op-code
00655 template<typename R, typename ...ARGS>
00656 struct Part<R (ARGS...), void> : _Part<ARGS...>
00657 {
00658     typedef Buf<ARGS...> Data;
00659
00660     // write arguments, skipping the dummy opcode
00661     template<typename DIR>
00662     static int write_op(void *b, int offset, int limit,
00663                       int /*placeholder for op*/,
00664                       typename _Elem<ARGS>::arg_type ...m) noexcept
00665     {
00666         char *buf = static_cast<char *>(b);
00667         int r = Data::template write<DIR>(buf, offset, limit, m...);
00668         if (L4_LIKELY(r >= offset))
00669             return r - offset;
00670         return r;
00671     }
00672 }
00673 };
00674
00675 // Message part with additional opcode
00676 template<typename OPCODE, typename R, typename ...ARGS>
00677 struct Part<R (ARGS...), OPCODE> : _Part<ARGS...>
00678 {
00679     typedef OPCODE opcode_type;
00680     typedef Buf<opcode_type, ARGS...> Data;
00681
00682     // write arguments, including the opcode
00683     template<typename DIR>
00684     static int write_op(void *b, int offset, int limit,
00685                       opcode_type op, typename _Elem<ARGS>::arg_type ...m) noexcept
00686     {
00687         char *buf = static_cast<char *>(b);
00688         int r = Data::template write<DIR>(buf, offset, limit, op, m...);
00689         if (L4_LIKELY(r >= offset))
00690             return r - offset;
00691         return r;
00692     }
00693 }
00694 };
00695
00696 } // namespace Detail
00697
00698 //-----
00699 // Implementation of the RPC call
00700 // TODO: Add support for timeout via special RPC argument
00701 // TODO: Add support for passing the UTCB pointer as argument
00702 //
00703 template<typename OP, typename CLASS, typename FLAGS, typename R,
00704         typename ...ARGS>
00705 inline R
00706 Rpc_inline_call<OP, CLASS, R (ARGS...), FLAGS>::
00707     call(L4::Cap<CLASS> cap,
00708         typename _Elem<ARGS>::arg_type ...a,
00709         l4_utcb_t *utcb) noexcept

```

```

00717 {
00718     using namespace Ipc::Msg;
00719
00720     typedef typename Kobject_typeid<CLASS>::Iface::Rpc Rpcs;
00721     typedef typename Rpcs::template Rpc<OP> Opt;
00722     typedef Detail::Part<ipc_type, typename Rpcs::opcode_type> Args;
00723
00724     l4_msg_regs_t *mrs = l4_utcb_mr_u(utcb);
00725
00726     // handle in-data part of the arguments
00727     int send_bytes =
00728         Args::template write_op<Do_in_data>(mrs->mr, 0, Mr_bytes,
00729                                             Opt::Opcode, a...);
00730
00731     if (L4_UNLIKELY(send_bytes < 0))
00732         return return_err<R>(send_bytes);
00733
00734     send_bytes = align_to<l4_umword_t>(send_bytes);
00735     int const send_words = send_bytes / Word_bytes;
00736     // write the in-items part of the message if there is one
00737     int item_bytes =
00738         Args::template write<Do_in_items>(&mrs->mr[send_words], 0,
00739                                           Mr_bytes - send_bytes, a...);
00740
00741     if (L4_UNLIKELY(item_bytes < 0))
00742         return return_err<R>(item_bytes);
00743
00744     int send_items = item_bytes / Item_bytes;
00745
00746     {
00747         // setup the receive buffers for the RPC call
00748         l4_buf_regs_t *brs = l4_utcb_br_u(utcb);
00749         // XXX: we currently support only one type of receive buffers per call
00750         brs->bdr = 0; // we always start at br[0]
00751
00752         // the limit leaves us at least one register for the zero terminator
00753         // add the buffers given as arguments to the buffer registers
00754         int bytes =
00755             Args::template write<Do_rcv_buffers>(brs->br, 0, Br_bytes - Word_bytes,
00756                                                  a...);
00757
00758         if (L4_UNLIKELY(bytes < 0))
00759             return return_err<R>(bytes);
00760
00761         brs->br[bytes / Word_bytes] = 0;
00762     }
00763
00764     // here we do the actual IPC -----
00765     l4_msgtag_t t;
00766     t = l4_msgtag(CLASS::Protocol, send_words, send_items, 0);
00767     // do the call (Q: do we need support for timeouts?)
00768     if (flags_type::Is_call)
00769         t = l4_ipc_call(cap.cap(), utcb, t, flags_type::timeout());
00770     else
00771     {
00772         t = l4_ipc_send(cap.cap(), utcb, t, flags_type::timeout());
00773         if (L4_UNLIKELY(t.has_error()))
00774             return return_ipc_err<R>(t, utcb);
00775
00776         return return_code<R>(l4_msgtag(0, 0, 0, t.flags()));
00777     }
00778
00779     // unmarshalling starts here -----
00780
00781     // bail out early in the case of an IPC error
00782     if (L4_UNLIKELY(t.has_error()))
00783         return return_ipc_err<R>(t, utcb);
00784
00785     // take the label as return value
00786     long r = t.label();
00787
00788     // bail out on negative error codes too
00789     if (L4_UNLIKELY(r < 0))
00790         return return_err<R>(r);
00791
00792     int const rcv_bytes = t.words() * Word_bytes;
00793
00794     // read the static out-data values to the arguments
00795     int err = Args::template read<Do_out_data>(mrs->mr, 0, rcv_bytes, r, a...);
00796
00797     int const item_limit = t.items() * Item_bytes;
00798
00799     if (L4_UNLIKELY(err < 0 || item_limit > Mr_bytes))
00800         return return_err<R>(-L4_EMSGTOOSHORT);
00801
00802     // read the static out-items to the arguments
00803

```



```

00804   err = Args::template read<Do_out_items>(&mrs->mr[t.words()], 0, item_limit,
00805                                           r, a...);
00806
00807   if (L4_UNLIKELY(err < 0))
00808       return return_err<R>(-L4_EMSTOOSHORT);
00809
00810   return return_code<R>(t);
00811 }
00812
00813 } // namespace Msg
00814 } // namespace Ipc
00815 } // namespace L4

```

## 17.491 ipc\_legacy

```

00001 // vi:set ft=cpp: -- Mode: C++ --
00002 /*
00003  * Copyright (C) 2015, 2017, 2024 Kernkonzept GmbH.
00004  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <l4/sys/cxx/ipc_epiface>
00011 #ifndef L4_RPC_DISABLE_LEGACY_DISPATCH
00012 #define L4_RPC_LEGACY_DISPATCH(IFACE)
00013     template<typename IOS>
00014     int dispatch(unsigned rights, IOS &ios)
00015     {
00016         typedef ::L4::Ipc::Detail::Dispatch<IFACE> Dispatch;
00017         l4_msgtag_t r = Dispatch::f(this, ios.tag(), rights, ios.utcb());
00018         ios.set_ipc_params(r);
00019         return r.label();
00020     }
00021
00022     template<typename IOS>
00023     int p_dispatch(IFACE *, unsigned rights, IOS &ios)
00024     {
00025         using ::L4::Ipc::Msg::dispatch_call;
00026         l4_msgtag_t r;
00027         r = dispatch_call<typename IFACE::Rpcs>(this, ios.utcb(),
00028                                                ios.tag(), rights);
00029         ios.set_ipc_params(r);
00030         return r.label();
00031     }
00032
00033 #define L4_RPC_LEGACY_USING(IFACE) \
00034     using IFACE::p_dispatch
00035
00036 #else
00037 #define L4_RPC_LEGACY_DISPATCH(IFACE)
00038 #define L4_RPC_LEGACY_USING(IFACE)
00039 #endif

```

## 17.492 ipc\_ret\_array

```

00001 // vi:set ft=cpp: -- Mode: C++ --
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "types"
00010 #include "ipc_basics"
00011
00012 namespace L4 { namespace Ipc L4_EXPORT {
00013
00014 // -----
00023 template<typename T> struct L4_EXPORT Ret_array
00024 {
00025     typedef T const **ptr_type;
00026
00027     T *value = nullptr;
00028     unsigned max = 0;
00029     Ret_array() {}

```

```

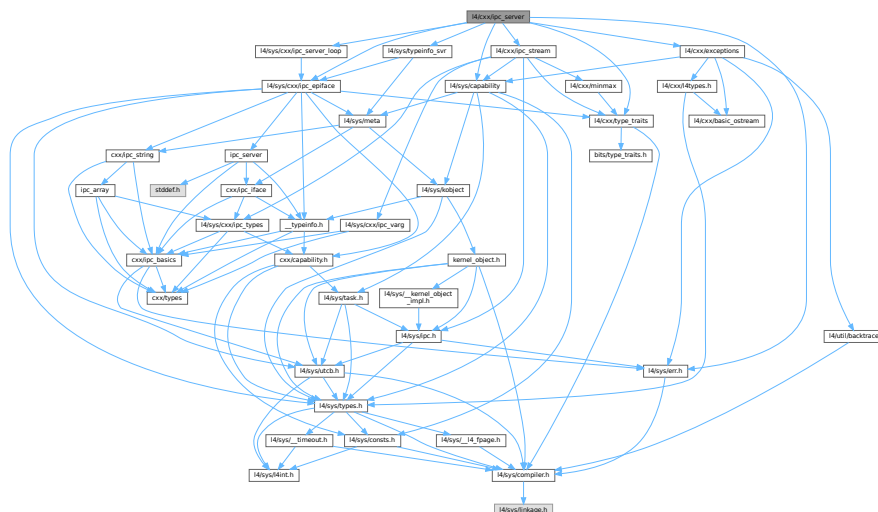
00030 Ret_array(T *v, unsigned max) : value(v), max(max) {}
00031 };
00032
00033 namespace Msg {
00034
00035 template<typename A> struct Elem< Ret_array<A> >
00036 {
00037     enum { Is_optional = false };
00038     typedef Ret_array<A> type;
00039     typedef typename type::ptr_type arg_type;
00040     typedef type svr_type;
00041     typedef type svr_arg_type;
00042 };
00043
00044 template<typename A>
00045 struct Is_valid_rpc_type<Ret_array<A> *> : L4::Types::False {};
00046 template<typename A>
00047 struct Is_valid_rpc_type<Ret_array<A> &> : L4::Types::False {};
00048 template<typename A>
00049 struct Is_valid_rpc_type<Ret_array<A> const &> : L4::Types::False {};
00050 template<typename A>
00051 struct Is_valid_rpc_type<Ret_array<A> const *> : L4::Types::False {};
00052
00053 template<typename A> struct Class< Ret_array<A> > : Class<A>::type {};
00054 template<typename A> struct Direction< Ret_array<A> > : Dir_out {};
00055
00056 template<typename A, typename CLASS>
00057 struct Clnt_val_ops<A const *, Dir_out, CLASS> : Clnt_noops<A const *>
00058 {
00059     using Clnt_noops<A const *>::from_msg;
00060     static int from_msg(char *msg, unsigned offset, unsigned limit, long ret,
00061                         A const *arg, Dir_out, Cls_data)
00062     {
00063         offset = align_to<A>(offset);
00064         arg = reinterpret_cast<A const *>(msg + offset);
00065         if (L4_UNLIKELY(!check_size<A>(offset, limit, ret)))
00066             return -1;
00067         return offset + ret * sizeof(A);
00068     }
00069 };
00070
00071
00072 template<typename A, typename CLASS>
00073 struct Svr_val_ops<Ret_array<A>, Dir_out, CLASS> :
00074     Svr_noops<Ret_array<A> >
00075 {
00076     typedef Ret_array<A> ret_array;
00077     using Svr_noops<ret_array>::from_svr;
00078     static int from_svr(char *, unsigned offset, unsigned limit, long ret,
00079                        ret_array const &, Dir_out, CLASS)
00080     {
00081         offset = align_to<A>(offset);
00082         if (L4_UNLIKELY(!check_size<A>(offset, limit, ret)))
00083             return -1;
00084         offset += sizeof(A) * ret;
00085         return offset;
00086     }
00087
00088     using Svr_noops<ret_array>::to_svr;
00089     static int to_svr(char *msg, unsigned offset, unsigned limit,
00090                      ret_array &arg, Dir_out, CLASS)
00091     {
00092         // there can be actually no limit check here, as this
00093         // is variably sized output array
00094         // FIXME: we could somehow makesure that this is the last
00095         // output value...
00096         offset = align_to<A>(offset);
00097         arg = ret_array(reinterpret_cast<A*>(msg + offset),
00098                         (limit - offset) / sizeof(A));
00099         // FIXME: we dont know the length of the array here so, cheat
00100         return offset;
00101     }
00102 };
00103 } // namespace Msg
00104
00105 }

```

## 17.493 I4/cxx/ipc\_server File Reference

IPC server loop.

Include dependency graph for ipc\_server:



- class `L4::Server_object`  
*Abstract server object to be used with `L4::Server` and `L4::Basic_registry`.*
- struct `L4::Server_object_t< IFACE, BASE >`  
*Base class (template) for server implementing server objects.*
- struct `L4::Server_object_x< Derived, IFACE, BASE >`  
*Helper class to implement `p_dispatch` based server objects.*
- struct `L4::Irq_handler_object`  
*`Server` object base class for handling IRQ messages.*

- namespace **L4**  
*L4 low-level kernel interface.*

Definition in file [ipc\\_server](#).

## 17.494 ipc\_server

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *           Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *           Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *           economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #pragma once
00015
00016 #include <l4/sys/capability>
00017 #include <l4/sys/typeinfo_svr>
00018 #include <l4/sys/err.h>
00019 #include <l4/cxx/ipc_stream>
00020 #include <l4/sys/cxx/ipc_epiface>
00021 #include <l4/sys/cxx/ipc_server_loop>
00022 #include <l4/cxx/type_traits>
00023 #include <l4/cxx/exceptions>
00024
00025 namespace L4 {
00026
00038 class Server_object : public Epiface
00039 {
00040 public:
00058     virtual int dispatch(unsigned long rights, Ipc::Iostream &ios) = 0;
00059
00060     l4_msgtag_t dispatch(l4_msgtag_t tag, unsigned rights, l4_utcb_t *utcb) override
00061     {
00062         L4::Ipc::Iostream ios(utcb);
00063         ios.tag() = tag;
00064         int r = dispatch(rights, ios);
00065         return ios.prepare_ipc(r);
00066     }
00067
00068     Cap<Kobject> obj_cap() const
00069     { return cap_cast<Kobject>(Epiface::obj_cap()); }
00070 };
00071
00079 template<typename IFACE, typename BASE = L4::Server_object>
00080 struct Server_object_t : BASE
00081 {
00083     typedef IFACE Interface;
00084
00086     typename BASE::Demand get_buffer_demand() const override
00087     { return typename L4::Kobject_typeid<IFACE>::Demand(); }
00088
00099     int dispatch_meta_request(L4::Ipc::Iostream &ios)
00100     { return L4::Util::handle_meta_request<IFACE>(ios); }
00101
00113     template<typename THIS>
00114     static int proto_dispatch(THIS *self, l4_umword_t rights, L4::Ipc::Iostream &ios)
00115     {
00116         l4_msgtag_t t;
00117         ios » t;
00118         return Kobject_typeid<IFACE>::proto_dispatch(self, t.label(), rights, ios);
00119     }
00120 };
00121
00142 template<typename Derived, typename IFACE, typename BASE = L4::Server_object>
00143 struct Server_object_x : Server_object_t<IFACE, BASE>
00144 {
00146     int dispatch(l4_umword_t r, L4::Ipc::Iostream &ios)
00147     {
00148         return Server_object_t<IFACE, BASE>::proto_dispatch(static_cast<Derived *>(this),
00149                                                             r, ios);
00150     }
00151 };
00152
00161 struct Irq_handler_object : Server_object_t<Kobject> {};
00162
00163 }

```

## 17.495 ipc\_server

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*

```

```

00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008 #pragma GCC system_header
00009
00010 #include <l4/sys/cxx/ipc_basics>
00011 #include <l4/sys/cxx/ipc_iface>
00012 #include <l4/sys/__typeinfo.h>
00013 #include <stddef.h>
00014
00015 namespace L4 {
00016 namespace Ipc {
00017 namespace Msg {
00018 namespace Detail {
00019
00020 template<typename T> struct Sizeof { enum { size = sizeof(T) }; };
00021 template<> struct Sizeof<void> { enum { size = 0 }; };
00022
00026 template<typename ...> struct Arg_pack
00027 {
00028     template<typename DIR>
00029     unsigned get(char *, unsigned offset, unsigned)
00030     { return offset; }
00031
00032     template<typename DIR>
00033     unsigned set(char *, unsigned offset, unsigned, long)
00034     { return offset; }
00035
00036     template<typename F, typename ...ARGS>
00037     long call(F f, ARGS ...args)
00038     { return f(args...); }
00039
00040     template<typename O, typename FUNC, typename ...ARGS>
00041     long obj_call(O *o, ARGS ...args)
00042     {
00043         typedef typename FUNC::template fwd<O> Fwd;
00044         return Fwd(o).template call<ARGS...>(args...);
00045         //return o->op_dispatch(args...);
00046     }
00047 };
00048
00052 template<typename T, typename SVR_TYPE, typename ...M>
00053 struct Svr_arg : Svr_xmit<T>, Arg_pack<M...>
00054 {
00055     typedef Arg_pack<M...> Base;
00056
00057     typedef SVR_TYPE svr_type;
00058     typedef typename _Elem<T>::svr_arg_type svr_arg_type;
00059
00060     svr_type v;
00061
00062     template<typename DIR>
00063     int get(char *msg, unsigned offset, unsigned limit)
00064     {
00065         typedef Svr_xmit<T> ct;
00066         int r = ct::to_svr(msg, offset, limit, this->v,
00067                             typename DIR::dir(), typename DIR::cls());
00068         if (L4_LIKELY(r >= 0))
00069             return Base::template get<DIR>(msg, r, limit);
00070
00071         if (_Elem<T>::Is_optional)
00072         {
00073             v = svr_type();
00074             return Base::template get<DIR>(msg, offset, limit);
00075         }
00076         return r;
00077     }
00078
00079     template<typename DIR>
00080     int set(char *msg, unsigned offset, unsigned limit, long ret)
00081     {
00082         typedef Svr_xmit<T> ct;
00083         int r = ct::from_svr(msg, offset, limit, ret, this->v,
00084                             typename DIR::dir(), typename DIR::cls());
00085         if (L4_UNLIKELY(r < 0))
00086             return r;
00087         return Base::template set<DIR>(msg, r, limit, ret);
00088     }
00089
00090     template<typename F, typename ...ARGS>
00091     long call(F f, ARGS ...args)
00092     {
00093         //As_arg<value_type> check;
00094         return Base::template
00095             call<F, ARGS..., svr_arg_type>(f, args..., this->v);

```

```

00096     }
00097
00098     template<typename O, typename FUNC, typename ...ARGS>
00099     long obj_call(O *o, ARGS ...args)
00100     {
00101         //As_arg<value_type> check;
00102         return Base::template
00103             obj_call<O,FUNC,  ARGS..., svr_arg_type>(o, args..., this->v);
00104     }
00105 };
00106
00107 template<typename T, typename ...M>
00108 struct Svr_arg<T, void, M...> : Arg_pack<M...>
00109 {
00110     typedef Arg_pack<M...> Base;
00111
00112     template<typename DIR>
00113     int get(char *msg, unsigned offset, unsigned limit)
00114     { return Base::template get<DIR>(msg, offset, limit); }
00115
00116     template<typename DIR>
00117     int set(char *msg, unsigned offset, unsigned limit, long ret)
00118     { return Base::template set<DIR>(msg, offset, limit, ret); }
00119
00120     template<typename F, typename ...ARGS>
00121     long call(F f, ARGS ...args)
00122     {
00123         return Base::template call<F, ARGS...>(f, args...);
00124     }
00125
00126     template<typename O, typename FUNC, typename ...ARGS>
00127     long obj_call(O *o, ARGS ...args)
00128     {
00129         return Base::template obj_call<O, FUNC, ARGS...>(o, args...);
00130     }
00131 };
00132
00133 template<typename A, typename ...M>
00134 struct Arg_pack<A, M...> : Svr_arg<A, typename _Elem<A>::svr_type, M...>
00135 {};
00136
00137 } // namespace Detail
00138
00139 //-----
00144 template<typename IPC_TYPE> struct Svr_arg_pack;
00145
00146 template<typename R, typename ...ARGS>
00147 struct Svr_arg_pack<R (ARGS...)> : Detail::Arg_pack<ARGS...>
00148 {
00149     typedef Detail::Arg_pack<ARGS...> Base;
00150     template<typename DIR>
00151     int get(void *msg, unsigned offset, unsigned limit)
00152     {
00153         char *buf = static_cast<char *>(msg);
00154         return Base::template get<DIR>(buf, offset, limit);
00155     }
00156
00157     template<typename DIR>
00158     int set(void *msg, unsigned offset, unsigned limit, long ret)
00159     {
00160         char *buf = static_cast<char *>(msg);
00161         return Base::template set<DIR>(buf, offset, limit, ret);
00162     }
00163 };
00164
00168 template<typename IPC_TYPE, typename O, typename ...ARGS>
00169 static l4_msgtag_t
00170 handle_svr_obj_call(O *o, l4_utcb_t *utcb, l4_msgtag_t tag, ARGS ...args)
00171 {
00172     typedef Svr_arg_pack<typename IPC_TYPE::rpc::ipc_type> Pack;
00173     enum
00174     {
00175         Do_reply = IPC_TYPE::rpc::flags_type::Is_call,
00176         Short_err = Do_reply ? -L4_EMMSGTOOSHORT : -L4_ENOREPLY,
00177     };
00178
00179     // XXX: send a reply or just do not reply in case of a cheating client
00180     if (L4_UNLIKELY(tag.words() + tag.items() * Item_words > Mr_words))
00181         return l4_msgtag(Short_err, 0, 0, 0);
00182
00183     // our whole arguments data structure
00184     Pack pack;
00185     l4_msg_regs_t *mrs = l4_utcb_mr_u(utcb);
00186
00187     int in_pos = Detail::Sizeof<typename IPC_TYPE::opcode_type>::size;
00188
00189     unsigned const in_bytes = tag.words() * Word_bytes;

```

```

00190
00191 in_pos = pack.template get<Do_in_data>(&mrs->mr[0], in_pos, in_bytes);
00192
00193 if (L4_UNLIKELY(in_pos < 0))
00194     return l4_msgtag(Short_err, 0, 0, 0);
00195
00196 if (L4_UNLIKELY(pack.template get<Do_out_data>(mrs->mr, 0, Mr_bytes) < 0))
00197     return l4_msgtag(Short_err, 0, 0, 0);
00198
00199
00200 in_pos = pack.template get<Do_in_items>(&mrs->mr[tag.words()], 0,
00201                                         tag.items() * Item_bytes);
00202
00203 if (L4_UNLIKELY(in_pos < 0))
00204     return l4_msgtag(Short_err, 0, 0, 0);
00205
00206 asm volatile ("": "=m" (mrs->mr));
00207
00208 // call the server function
00209 long ret = pack.template obj_call<0, typename IPC_TYPE::rpc, ARGS...>(o, args...);
00210
00211 if (!Do_reply)
00212     return l4_msgtag(-L4_ENOREPLY, 0, 0, 0);
00213
00214 // our convention says that negative return value means no
00215 // reply data
00216 if (L4_UNLIKELY(ret < 0))
00217     return l4_msgtag(ret, 0, 0, 0);
00218
00219 // reply with the reply data from the server function
00220 int bytes = pack.template set<Do_out_data>(mrs->mr, 0, Mr_bytes, ret);
00221 if (L4_UNLIKELY(bytes < 0))
00222     return l4_msgtag(-L4_MSGTOOLONG, 0, 0, 0);
00223
00224 unsigned words = (bytes + Word_bytes - 1) / Word_bytes;
00225 bytes = pack.template set<Do_out_items>(&mrs->mr[words], 0,
00226                                         Mr_bytes - words * Word_bytes,
00227                                         ret);
00228 if (L4_UNLIKELY(bytes < 0))
00229     return l4_msgtag(-L4_MSGTOOLONG, 0, 0, 0);
00230
00231 unsigned const items = bytes / Item_bytes;
00232 return l4_msgtag(ret, words, items, 0);
00233 }
00234
00235 //-----
00236
00237 template<typename RPCS, typename OPCODE_TYPE>
00238 struct Dispatch_call;
00239
00240 template<typename CLASS>
00241 struct Dispatch_call<L4::Typeid::Raw_ipc<CLASS>, void>
00242 {
00243     template<typename OBJ, typename ...ARGS>
00244     static l4_msgtag_t
00245     call(OBJ *o, l4_utcb_t *utcb, l4_msgtag_t tag, ARGS ...a)
00246     {
00247         return o->op_dispatch(utcb, tag, a...);
00248     }
00249 };
00250
00251 template<typename RPCS>
00252 struct Dispatch_call<RPCS, void>
00253 {
00254     constexpr static unsigned rmask()
00255     { return RPCS::rpc::flags_type::Rights & 3UL; }
00256
00257     template<typename OBJ, typename ...ARGS>
00258     static l4_msgtag_t
00259     call(OBJ *o, l4_utcb_t *utcb, l4_msgtag_t tag, unsigned rights, ARGS ...a)
00260     {
00261         if ((rights & rmask()) != rmask())
00262             return l4_msgtag(-L4_EPERM, 0, 0, 0);
00263
00264         typedef L4::Typeid::Rights<typename RPCS::rpc::class_type> Rights;
00265         return handle_svr_obj_call<RPCS>(o, utcb, tag,
00266                                         Rights(rights), a...);
00267     }
00268 };
00269
00270
00271 template<typename RPCS, typename OPCODE_TYPE>
00272 struct Dispatch_call
00273 {
00274     constexpr static unsigned rmask()
00275     { return RPCS::rpc::flags_type::Rights & 3UL; }
00276

```

```

00277 template<typename OBJ, typename ...ARGS>
00278 static l4_msgtag_t
00279 _call(OBJ *o, l4_utcb_t *utcb, l4_msgtag_t tag, unsigned rights, OPCODE_TYPE op, ARGS ...)
00280 {
00281     if (L4::Types::Same<typename RPCS::opcode_type, void>::value
00282         || RPCS::Opcode == op)
00283     {
00284         if ((rights & rmask()) != rmask())
00285             return l4_msgtag(-L4_EPERM, 0, 0, 0);
00286
00287         typedef L4::Typeid::Rights<typename RPCS::rpc::class_type> Rights;
00288         return handle_svr_obj_call<RPCS>(o, utcb, tag,
00289                                         Rights(rights), a...);
00290     }
00291     return Dispatch_call<typename RPCS::next, OPCODE_TYPE>::template
00292         _call<OBJ, ARGS...>(o, utcb, tag, rights, op, a...);
00293 }
00294
00295 template<typename OBJ, typename ...ARGS>
00296 static l4_msgtag_t
00297 call(OBJ *o, l4_utcb_t *utcb, l4_msgtag_t tag, unsigned rights, ARGS ...)
00298 {
00299     OPCODE_TYPE op;
00300     unsigned limit = tag.words() * Word_bytes;
00301     typedef Svr_xmit<OPCODE_TYPE> S;
00302     int err = S::to_svr(reinterpret_cast<char *>(l4_utcb_mr_u(utcb)->mr), 0,
00303                        limit, op, Dir_in(), Cls_data());
00304     if (L4_UNLIKELY(err < 0))
00305         return l4_msgtag(-L4_EMMSGTOOSHORT, 0, 0, 0);
00306
00307     return _call<OBJ, ARGS...>(o, utcb, tag, rights, op, a...);
00308 }
00309 };
00310
00311 template<>
00312 struct Dispatch_call<Typeid::Detail::Rpc_end, void>
00313 {
00314     template<typename OBJ, typename ...ARGS>
00315     static l4_msgtag_t
00316     _call(OBJ *, l4_utcb_t *, l4_msgtag_t, unsigned, int, ARGS ...)
00317     { return l4_msgtag(-L4_ENOSYS, 0, 0, 0); }
00318
00319     template<typename OBJ, typename ...ARGS>
00320     static l4_msgtag_t
00321     call(OBJ *, l4_utcb_t *, l4_msgtag_t, unsigned, ARGS ...)
00322     { return l4_msgtag(-L4_ENOSYS, 0, 0, 0); }
00323 };
00324
00325 template<typename OPCODE_TYPE>
00326 struct Dispatch_call<Typeid::Detail::Rpc_end, OPCODE_TYPE> :
00327     Dispatch_call<Typeid::Detail::Rpc_end, void> {};
00328
00329 template<typename RPCS, typename OBJ, typename ...ARGS>
00330 static l4_msgtag_t
00331 dispatch_call(OBJ *o, l4_utcb_t *utcb, l4_msgtag_t tag, unsigned rights, ARGS ...)
00332 {
00333     return Dispatch_call<typename RPCS::type, typename RPCS::opcode_type>::template
00334         call<OBJ, ARGS...>(o, utcb, tag, rights, a...);
00335 }
00336
00337 } // namespace Msg
00338 } // namespace Ipc
00339 } // namespace L4

```

## 17.496 ipc\_server\_loop

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2015, 2017, 2019, 2021-2024 Kernkonzept GmbH.
00004  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "ipc_epiface"
00011
00012 namespace L4 {
00013
00019
00031 namespace Ipc_svr {
00032
00046 enum Reply_mode

```



```

00047 {
00048     Reply_compound,
00049     Reply_separate
00050 };
00051
00052 struct Ignore_errors
00053 { static void error(l4_msgtag_t, l4_utcb_t *) {} };
00054
00055 struct Default_timeout
00056 { static l4_timeout_t timeout() { return L4_IPC_SEND_TIMEOUT_0; } };
00057
00058 struct Compound_reply
00059 {
00060     static Reply_mode before_reply(l4_msgtag_t, l4_utcb_t *)
00061     { return Reply_compound; }
00062 };
00063
00064 struct Default_setup_wait
00065 { static void setup_wait(l4_utcb_t *, Reply_mode) {} };
00066
00067 template< typename R >
00068 struct Direct_dispatch
00069 {
00070     R &r;
00071
00072     Direct_dispatch(R &r) : r(r) {}
00073
00074     l4_msgtag_t operator () (l4_msgtag_t tag, l4_umword_t obj, l4_utcb_t *utcb)
00075     { return r.dispatch(tag, obj, utcb); }
00076 };
00077
00078 template< typename R >
00079 struct Direct_dispatch<R*>
00080 {
00081     R *r;
00082
00083     Direct_dispatch(R *r) : r(r) {}
00084
00085     l4_msgtag_t operator () (l4_msgtag_t tag, l4_umword_t obj, l4_utcb_t *utcb)
00086     { return r->dispatch(tag, obj, utcb); }
00087 };
00088
00089 #ifdef __EXCEPTIONS
00090 template< typename R, typename Exc> // = L4::Runtime_error>
00091 struct Exc_dispatch : private Direct_dispatch<R>
00092 {
00093     Exc_dispatch(R r) : Direct_dispatch<R>(r) {}
00094
00095     l4_msgtag_t operator () (l4_msgtag_t tag, l4_umword_t obj, l4_utcb_t *utcb)
00096     {
00097         try
00098         {
00099             return Direct_dispatch<R>::operator () (tag, obj, utcb);
00100         }
00101         catch (Exc &e)
00102         {
00103             return l4_msgtag(e.err_no(), 0, 0, 0);
00104         }
00105         catch (int err)
00106         {
00107             return l4_msgtag(err, 0, 0, 0);
00108         }
00109         catch (long err)
00110         {
00111             return l4_msgtag(err, 0, 0, 0);
00112         }
00113     }
00114 };
00115
00116 template< typename R, typename Exc, typename Printer >
00117 struct Dbg_dispatch : private Direct_dispatch<R>
00118 {
00119     Dbg_dispatch(R r, Printer p) : Direct_dispatch<R>(r), _printer(p) {}
00120
00121     l4_msgtag_t operator () (l4_msgtag_t tag, l4_umword_t obj, l4_utcb_t *utcb)
00122     {
00123         try
00124         {
00125             return Direct_dispatch<R>::operator () (tag, obj, utcb);
00126         }
00127         catch (Exc &e)
00128         {
00129             _printer.printf("Error in handling IPC: %s: %s\n", e.str(),
00130                             e.extra_str());
00131             return l4_msgtag(e.err_no(), 0, 0, 0);
00132         }
00133         catch (int err)
00134         {
00135             return l4_msgtag(err, 0, 0, 0);
00136         }
00137     }
00138 };

```

```

00206     {
00207         _printer.printf("Error in handling IPC: %d (%s)\n", err,
00208             l4sys_errtostr(err));
00209         return l4_msgtag(err, 0, 0, 0);
00210     }
00211     catch (long err)
00212     {
00213         _printer.printf("Error in handling IPC: %ld (%s)\n", err,
00214             l4sys_errtostr(err));
00215         return l4_msgtag(err, 0, 0, 0);
00216     }
00217 }
00218
00219 Printer _printer;
00220 };
00221 #endif
00222
00233 class Br_manager_no_buffers : public Server_iface
00234 {
00235 public:
00240     int alloc_buffer_demand(Demand const &demand) override
00241     {
00242         if (!demand.no_demand())
00243             return -L4_ENOMEM;
00244         return L4_EOK;
00245     }
00246
00248     L4::Cap<void> get_rcv_cap(int) const override
00249     { return L4::Cap<void>::Invalid; }
00250
00252     int realloc_rcv_cap(int) override
00253     { return -L4_ENOMEM; }
00254
00256     int add_timeout(Timeout *, l4_kernel_clock_t) override
00257     { return -L4_ENOSYS; }
00258
00260     int remove_timeout(Timeout *) override
00261     { return -L4_ENOSYS; }
00262
00263 protected:
00265     unsigned first_free_br() const
00266     { return 1; }
00267
00269     void setup_wait(l4_utcb_t *utcb, L4::Ipc_svr::Reply_mode)
00270     {
00271         l4_buf_regs_t *br = l4_utcb_br_u(utcb);
00272         br->bdr = 0;
00273         br->br[0] = 0;
00274     }
00275 };
00276
00285 struct Default_loop_hooks :
00286     public Ignore_errors, public Default_timeout, public Compound_reply,
00287     Br_manager_no_buffers
00288 {};
00289
00290 }
00291
00306 template< typename LOOP_HOOKS = Ipc_svr::Default_loop_hooks >
00307 class Server :
00308     public LOOP_HOOKS
00309 {
00310 public:
00317     /* Internal note: After all users have been converted, remove this
00318        * constructor. Also remove the constructor below then. */
00319     explicit Server(l4_utcb_t *)
00320         L4_DEPRECATED("Do not specify the UTCB with the constructor. "
00321             "Supply it on the loop function if needed.")
00322     {}
00323
00327     /* Internal note: Remove this constructor when the above deprecated
00328        * constructor with the UTCB pointer is also removed. */
00329     Server() {}
00330
00337     template< typename DISPATCH >
00338     inline L4_NORETURN void internal_loop(DISPATCH dispatch, l4_utcb_t *);
00339
00343     template< typename R >
00344     inline L4_NORETURN void loop_noexc(R r, l4_utcb_t *u = l4_utcb())
00345     { internal_loop(Ipc_svr::Direct_dispatch<R>(r), u); }
00346
00347 #ifdef __EXCEPTIONS
00354     template< typename EXC, typename R >
00355     inline L4_NORETURN void loop(R r, l4_utcb_t *u = l4_utcb())
00356     {
00357         internal_loop(Ipc_svr::Exc_dispatch<R, EXC>(r), u);
00358         // function will never return

```

```

00359     }
00360
00367     template< typename EXC, typename R, typename Printer >
00368     inline L4_NORETURN void loop_dbg(R r, Printer p, l4_utcb_t *u = l4_utcb())
00369     {
00370         internal_loop(Ipc_svr::Dbg_dispatch<R, EXC, Printer>(r, p), u);
00371         // function will never return
00372     }
00373 #endif
00374 protected:
00376     inline l4_msgtag_t reply_n_wait(l4_msgtag_t reply, l4_umword_t *p, l4_utcb_t *);
00377 };
00378
00379 template< typename L >
00380 inline l4_msgtag_t
00381 Server<L>::reply_n_wait(l4_msgtag_t reply, l4_umword_t *p, l4_utcb_t *utcb)
00382 {
00383     if (reply.label() != -L4_ENOREPLY)
00384     {
00385         Ipc_svr::Reply_mode m = this->before_reply(reply, utcb);
00386         if (m == Ipc_svr::Reply_compound)
00387         {
00388             this->setup_wait(utcb, m);
00389             return l4_ipc_reply_and_wait(utcb, reply, p, this->timeout());
00390         }
00391         else
00392         {
00393             l4_msgtag_t res = l4_ipc_send(L4_INVALID_CAP | L4_SYSF_REPLY, utcb, reply, this->timeout());
00394             if (res.has_error())
00395                 return res;
00396         }
00397     }
00398     this->setup_wait(utcb, Ipc_svr::Reply_separate);
00399     return l4_ipc_wait(utcb, p, this->timeout());
00400 }
00401
00402 template< typename L >
00403 template< typename DISPATCH >
00404 inline L4_NORETURN void
00405 Server<L>::internal_loop(DISPATCH dispatch, l4_utcb_t *utcb)
00406 {
00407     l4_msgtag_t res;
00408     l4_umword_t p;
00409     l4_msgtag_t r = l4_msgtag(-L4_ENOREPLY, 0, 0, 0);
00410
00411     while (true)
00412     {
00413         res = reply_n_wait(r, &p, utcb);
00414         if (res.has_error())
00415         {
00416             this->error(res, utcb);
00417             r = l4_msgtag(-L4_ENOREPLY, 0, 0, 0);
00418             continue;
00419         }
00420
00421         r = dispatch(res, p, utcb);
00422     }
00423 }
00424
00425 } // namespace L4

```

## 17.497 ipc\_string

```

00001 // vi:set ft=c++: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "types"
00010 #include "ipc_basics"
00011 #include "ipc_array"
00012
00013 namespace L4 { namespace Ipc {
00014
00015     template<typename CHAR = char const, typename LEN = unsigned long>
00016     struct String : Array<CHAR, LEN>
00017     {
00018         static LEN strlength(CHAR *d) { LEN l = 0; while (d[l]) ++l; return l; }
00019         String() {}
00020         String(CHAR *d) : Array<CHAR, LEN>(strlength(d) + 1, d) {}
00021     };
00022 } }

```

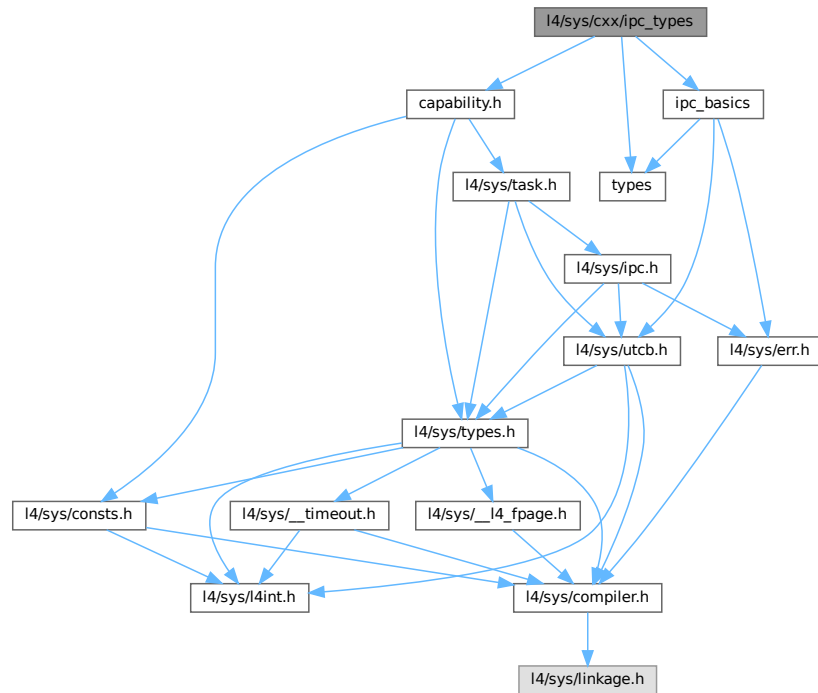
```

00021 String(LEN len, CHAR *d) : Array<CHAR, LEN>(len, d) {}
00022 void copy_in(CHAR const *s)
00023 {
00024     if (this->length < 1)
00025         return;
00026
00027     LEN i;
00028     for (i = 0; i < this->length - 1 && s[i]; ++i)
00029         this->data[i] = s[i];
00030     this->length = i + 1;
00031     this->data[i] = CHAR();
00032 }
00033 };
00034
00035 #if __cplusplus >= 201103L
00036 template< typename CHAR = char, typename LEN_TYPE = unsigned long,
00037           LEN_TYPE MAX = (L4_UTCB_GENERIC_DATA_SIZE *
00038                           sizeof(l4_umword_t)) / sizeof(CHAR) >
00039 using String_in_buf = Array_in_buf<CHAR, LEN_TYPE, MAX>;
00040 #endif
00041
00042 namespace Msg {
00043     template<typename A, typename LEN>
00044     struct Clnt_xmit< String<A, LEN> > : Clnt_xmit< Array<A, LEN> > {};
00045
00046     template<typename A, typename LEN, typename CLASS>
00047     struct Svr_val_ops< String<A, LEN>, Dir_in, CLASS >
00048     : Svr_val_ops< Array_ref<A, LEN>, Dir_in, CLASS >
00049     {
00050         typedef Svr_val_ops< Array_ref<A, LEN>, Dir_in, CLASS > Base;
00051         typedef typename Base::svr_type svr_type;
00052         using Base::to_svr;
00053         static int to_svr(char *msg, unsigned offset, unsigned limit,
00054                          svr_type &a, Dir_in dir, Cls_data cls)
00055         {
00056             int r = Base::to_svr(msg, offset, limit, a, dir, cls);
00057             if (r < 0)
00058                 return r;
00059
00060             // correct clients send at least the zero terminator
00061             if (a.length < 1)
00062                 return -L4_MSGTOOSHORT;
00063
00064             typedef typename L4::Types::Remove_const<A>::type elem_type;
00065             const_cast<elem_type*>(a.data)[a.length - 1] = A();
00066             return r;
00067         }
00068     };
00069
00070     template<typename A, typename LEN>
00071     struct Clnt_xmit<String<A, LEN> &> : Clnt_xmit<Array<A, LEN> &>
00072     {
00073         typedef Array<A, LEN> &type;
00074
00075         using Clnt_xmit<type>::from_msg;
00076         static int from_msg(char *msg, unsigned offset, unsigned limit, long ret,
00077                             Array<A, LEN> &a, Dir_out dir, Cls_data cls)
00078         {
00079             int r = Clnt_xmit<type>::from_msg(msg, offset, limit, ret, a, dir, cls);
00080             if (r < 0)
00081                 return r;
00082
00083             // check for a bad servers
00084             if (a.length < 1)
00085                 return -L4_MSGTOOSHORT;
00086
00087             a.data[a.length - 1] = A();
00088             return r;
00089         };
00090     };
00091
00092     template<typename A, typename LEN>
00093     struct Clnt_xmit<String<A, LEN> *> : Clnt_xmit<String<A, LEN> &> {};
00094
00095     template<typename A, typename LEN, typename CLASS>
00096     struct Svr_val_ops<String<A, LEN>, Dir_out, CLASS>
00097     : Svr_val_ops<Array_ref<A, LEN>, Dir_out, CLASS>
00098     {};
00099
00100     template<typename A, typename LEN>
00101     struct Is_valid_rpc_type<String<A, LEN> const *> : L4::Types::False {};
00102     template<typename A, typename LEN>
00103     struct Is_valid_rpc_type<String<A, LEN> const &> : L4::Types::False {};
00104
00105 } // namespace Msg
00106
00107 }

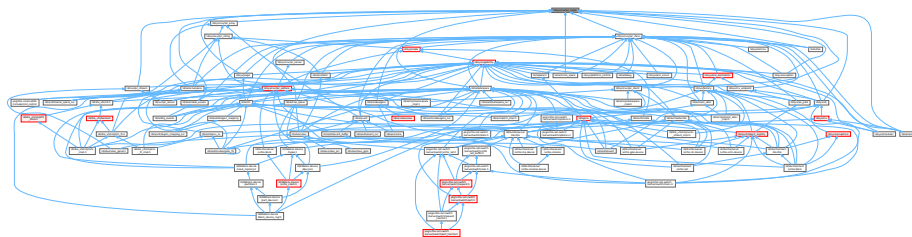
```

## 17.498 I4/sys/cxx/ipc\_types File Reference

```
#include "capability.h"
#include "types"
#include "ipc_basics"
Include dependency graph for ipc_types:
```



This graph shows which files directly or indirectly include this file:



### Data Structures

- struct [L4::ipc::In\\_out< T >](#)  
Mark an argument as in-out argument.
- struct [L4::ipc::As\\_value< T >](#)  
Pass the argument as plain data value.
- struct [L4::ipc::Opt< T >](#)  
Attribute for defining an optional RPC argument.

- class [L4::lpc::Small\\_buf](#)  
*A receive item for receiving a single object capability.*
- class [L4::lpc::Gen\\_fpage](#)  
*Generic RPC base for typed message items.*
- class [L4::lpc::Snd\\_fpage](#)  
*Send item or return item.*
- class [L4::lpc::Rcv\\_fpage](#)  
*Non-small receive item.*
- class [L4::lpc::Cap< T >](#)  
*Capability type for RPC interfaces (see [L4::Cap< T >](#)).*

## Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*
- namespace [L4::lpc](#)  
*IPC related functionality.*
- namespace [L4::lpc::Msg](#)  
*IPC Message related functionality.*

## Functions

- `template<typename T>`  
`Cap< T > L4::lpc::make_cap (L4::Cap< T > cap, unsigned rights) noexcept`  
*Make an [L4::lpc::Cap< T >](#) for the given capability and rights.*
- `template<typename T>`  
`Cap< T > L4::lpc::make_cap_rw (L4::Cap< T > cap) noexcept`  
*Make an [L4::lpc::Cap< T >](#) for the given capability with [L4\\_CAP\\_FPAGE\\_RW](#) rights.*
- `template<typename T>`  
`Cap< T > L4::lpc::make_cap_rws (L4::Cap< T > cap) noexcept`  
*Make an [L4::lpc::Cap< T >](#) for the given capability with [L4\\_CAP\\_FPAGE\\_RWS](#) rights.*
- `template<typename T>`  
`Cap< T > L4::lpc::make_cap_full (L4::Cap< T > cap) noexcept`  
*Make an [L4::lpc::Cap< T >](#) for the given capability with full fpage and object-specific rights.*

## 17.499 ipc\_types

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "capability.h"
00010 #include "types"
00011 #include "ipc_basics"
00012
00013 namespace L4 {
00014
00015     typedef int Opcode;
00016
00017     namespace Ipc {
```

```

00022
00031 template<typename T> struct L4_EXPORT Out;
00032
00033
00041 template<typename T> struct L4_EXPORT In_out
00042 {
00043     T v;
00044     In_out() {}
00045     In_out(T v) : v(v) {}
00046     operator T () const { return v; }
00047     operator T & () { return v; }
00048 };
00049
00050 namespace Msg {
00051 template<typename A> struct Elem< In_out<A *> > : Elem<A *> {};
00052
00053 template<typename A>
00054 struct Svr_xmit< In_out<A *> > : Svr_xmit<A *>, Svr_xmit<A const *>
00055 {
00056     using Svr_xmit<A *>::from_svr;
00057     using Svr_xmit<A const *>::to_svr;
00058 };
00059
00060 template<typename A>
00061 struct Clnt_xmit< In_out<A *> > : Clnt_xmit<A *>, Clnt_xmit<A const *>
00062 {
00063     using Clnt_xmit<A *>::from_msg;
00064     using Clnt_xmit<A const *>::to_msg;
00065 };
00066
00067 template<typename A>
00068 struct Is_valid_rpc_type< In_out<A *> > : Is_valid_rpc_type<A *> {};
00069 template<typename A>
00070 struct Is_valid_rpc_type< In_out<A const *> > : L4::Types::False {};
00071
00072 #ifdef CONFIG_ALLOW_REFS
00073 template<typename A> struct Elem< In_out<A &> > : Elem<A &> {};
00074
00075 template<typename A>
00076 struct Svr_xmit< In_out<A &> > : Svr_xmit<A &>, Svr_xmit<A const &>
00077 {
00078     using Svr_xmit<A &>::from_svr;
00079     using Svr_xmit<A const &>::to_svr;
00080 };
00081
00082 template<typename A>
00083 struct Clnt_xmit< In_out<A &> > : Clnt_xmit<A &>, Clnt_xmit<A const &>
00084 {
00085     using Clnt_xmit<A &>::from_msg;
00086     using Clnt_xmit<A const &>::to_msg;
00087 };
00088
00089 template<typename A>
00090 struct Is_valid_rpc_type< In_out<A &> > : Is_valid_rpc_type<A &> {};
00091 template<typename A>
00092 struct Is_valid_rpc_type< In_out<A const &> > : L4::Types::False {};
00093
00094 #else
00095
00096 template<typename A>
00097 struct Is_valid_rpc_type< In_out<A &> > : L4::Types::False {};
00098
00099 #endif
00100
00101 // Value types don't make sense for output.
00102 template<typename A>
00103 struct Is_valid_rpc_type< In_out<A > > : L4::Types::False {};
00104
00105 }
00106
00107
00116 template<typename T> struct L4_EXPORT As_value
00117 {
00118     typedef T value_type;
00119     T v;
00120     As_value() noexcept {}
00121     As_value(T v) noexcept : v(v) {}
00122     operator T () const noexcept { return v; }
00123     operator T & () noexcept { return v; }
00124 };
00125
00126 namespace Msg {
00127 template<typename T> struct Class< As_value<T> > : Cls_data {};
00128 template<typename T> struct Elem< As_value<T> > : Elem<T> {};
00129 template<typename T> struct Elem< As_value<T> *> : Elem<T *> {};
00130 }
00131

```

```

00132
00136 template<typename T> struct L4_EXPORT Opt
00137 {
00138     T _value;
00139     bool _valid;
00140
00142     Opt() noexcept : _valid(false) {}
00143
00145     Opt(T value) noexcept : _value(value), _valid(true) {}
00146
00148     Opt &operator = (T value) noexcept
00149     {
00150         this->_value = value;
00151         this->_valid = true;
00152         return *this;
00153     }
00154
00156     void set_valid(bool valid = true) noexcept { _valid = valid; }
00157
00159     T *operator -> () noexcept { return &this->_value; }
00161     T const *operator -> () const noexcept { return &this->_value; }
00163     T value() const noexcept { return this->_value; }
00165     T &value() noexcept { return this->_value; }
00167     bool is_valid() const noexcept { return this->_valid; }
00168 };
00169
00170 namespace Msg {
00171 template<typename T> struct Elem< Opt<T &> > : Elem<T &>
00172 {
00173     enum { Is_optional = true };
00174     typedef Opt<typename Elem<T &>::svr_type> &svr_arg_type;
00175     typedef Opt<typename Elem<T &>::svr_type> svr_type;
00176 };
00177
00178 template<typename T> struct Elem< Opt<T *> > : Elem<T *>
00179 {
00180     enum { Is_optional = true };
00181     typedef Opt<typename Elem<T *>::svr_type> &svr_arg_type;
00182     typedef Opt<typename Elem<T *>::svr_type> svr_type;
00183 };
00184
00185
00186
00187 template<typename T, typename CLASS>
00188 struct Svr_val_ops<Opt<T>, Dir_out, CLASS> : Svr_noops< Opt<T> >
00189 {
00190     typedef Opt<T> svr_type;
00191     typedef Svr_val_ops<T, Dir_out, CLASS> Native;
00192
00193     using Svr_noops< Opt<T> >::to_svr;
00194     static int to_svr(char *msg, unsigned offset, unsigned limit,
00195                     Opt<T> &arg, Dir_out, CLASS) noexcept
00196     {
00197         return Native::to_svr(msg, offset, limit, arg.value(), Dir_out(), CLASS());
00198     }
00199
00200     using Svr_noops< Opt<T> >::from_svr;
00201     static int from_svr(char *msg, unsigned offset, unsigned limit, long ret,
00202                     svr_type &arg, Dir_out, CLASS) noexcept
00203     {
00204         if (arg.is_valid())
00205             return Native::from_svr(msg, offset, limit, ret, arg.value(),
00206                                     Dir_out(), CLASS());
00207         return offset;
00208     }
00209 };
00210
00211 template<typename T> struct Elem< Opt<T> > : Elem<T>
00212 {
00213     enum { Is_optional = true };
00214     typedef Opt<T> arg_type;
00215 };
00216
00217 template<typename T> struct Elem< Opt<T const *> > : Elem<T const *>
00218 {
00219     enum { Is_optional = true };
00220     typedef Opt<T const *> arg_type;
00221 };
00222
00223 template<typename T>
00224 struct Is_valid_rpc_type< Opt<T const &> > : L4::Types::False {};
00225
00226 template<typename T, typename CLASS>
00227 struct Clnt_val_ops<Opt<T>, Dir_in, CLASS> : Clnt_noops< Opt<T> >
00228 {
00229     typedef Opt<T> arg_type;
00230     typedef Detail::_Clnt_val_ops<typename Elem<T>::arg_type, Dir_in, CLASS> Native;

```



```

00231
00232     using Clnt_noops< Opt<T> >::to_msg;
00233     static int to_msg(char *msg, unsigned offset, unsigned limit,
00234                       arg_type arg, Dir_in, CLASS) noexcept
00235     {
00236         if (arg.is_valid())
00237             return Native::to_msg(msg, offset, limit,
00238                                   Detail::_Plain<T>::deref(arg.value()),
00239                                   Dir_in(), CLASS());
00240         return offset;
00241     }
00242 };
00243
00244 template<typename T> struct Class< Opt<T> > :
00245     Class< typename Detail::_Plain<T>::type > {};
00246 template<typename T> struct Direction< Opt<T> > : Direction<T> {};
00247 }
00248
00257 class L4_EXPORT Small_buf
00258 {
00259 public:
00267     explicit Small_buf(L4::Cap<void> cap, unsigned long flags = 0) noexcept
00268         : _data(cap.cap() | L4_RCV_ITEM_SINGLE_CAP | flags) {}
00269
00274     explicit Small_buf(l4_cap_idx_t cap, unsigned long flags = 0) noexcept
00275         : _data(cap | L4_RCV_ITEM_SINGLE_CAP | flags) {}
00276
00278     l4_umword_t raw() const noexcept { return _data; }
00279 private:
00280     l4_umword_t _data;
00281 };
00282
00286 class L4_EXPORT Gen_fpage
00287 {
00288 public:
00290     enum Type
00291     {
00292         Special = L4_FPAGE_SPECIAL « 4,
00293         Memory  = L4_FPAGE_MEMORY  « 4,
00294         Io      = L4_FPAGE_IO      « 4,
00295         Obj     = L4_FPAGE_OBJ     « 4
00296     };
00297
00299     Gen_fpage(l4_umword_t base, l4_umword_t data) noexcept
00300         : _base(base), _data(data)
00301     {}
00302
00304     l4_umword_t data() const noexcept { return _data; }
00306     l4_umword_t base_x() const noexcept { return _base; }
00307
00308 protected:
00309     l4_umword_t _base;
00310     l4_umword_t _data;
00311 };
00312
00323 class Snd_fpage : public Gen_fpage
00324 {
00325 public:
00328     enum Map_type
00329     {
00330         Map    = L4_MAP_ITEM_MAP,
00331         Grant  = L4_MAP_ITEM_GRANT,
00332     };
00333
00336     enum Cacheopt
00337     {
00338         None    = 0,
00339         Cached  = L4_FPAGE_CACHEABLE « 4,
00340         Buffered = L4_FPAGE_BUFFERABLE « 4,
00341         Uncached = L4_FPAGE_UNCACHEABLE « 4
00342     };
00343
00346     enum Continue
00347     {
00348         Single   = 0,
00349         Last     = 0,
00350         More     = L4_ITEM_CONT,
00351         Compound = L4_ITEM_CONT,
00352     };
00353
00355     Snd_fpage(l4_umword_t base = 0, l4_umword_t data = 0) noexcept
00356         : Gen_fpage(base, data)
00357     {}
00358
00370     Snd_fpage(l4_fpage_t const &fp, l4_addr_t snd_base = 0,
00371               Map_type map_type = Map,
00372               Cacheopt cache = None, Continue cont = Last) noexcept

```

```

00373 : Gen_fpage(L4_ITEM_MAP | (snd_base & (~0UL << 12)) | l4_umword_t(map_type)
00374 | l4_umword_t(cache) | l4_umword_t(cont),
00375 fp.raw)
00376 {}
00377
00386 Snd_fpage(L4::Cap<void> cap, unsigned rights, Map_type map_type = Map) noexcept
00387 : Gen_fpage(L4_ITEM_MAP | l4_umword_t(map_type) | (rights & 0xf0),
00388 cap.fpage(rights).raw)
00389 {}
00390
00402 static Snd_fpage obj(l4_cap_idx_t base, int order,
00403 unsigned char rights,
00404 l4_addr_t snd_base = 0,
00405 Map_type map_type = Map,
00406 Continue cont = Last) noexcept
00407 {
00408 return Snd_fpage(l4_obj_fpage(base, order, rights), snd_base,
00409 map_type, None, cont);
00410 }
00411
00424 static Snd_fpage mem(l4_addr_t base, int order,
00425 unsigned char rights,
00426 l4_addr_t snd_base = 0,
00427 Map_type map_type = Map,
00428 Cacheopt cache = None, Continue cont = Last) noexcept
00429 {
00430 return Snd_fpage(l4_fpage(base, order, rights), snd_base, map_type, cache,
00431 cont);
00432 }
00433
00445 static Snd_fpage io(unsigned long base, int order,
00446 unsigned char rights,
00447 l4_addr_t snd_base = 0,
00448 Map_type map_type = Map,
00449 Continue cont = Last) noexcept
00450 {
00451 return Snd_fpage(l4_fpage_set_rights(l4_iofpage(base, order), rights),
00452 snd_base, map_type, None, cont);
00453 }
00454
00457 unsigned order() const noexcept { return (_data >> 6) & 0x3f; }
00458
00461 unsigned snd_order() const noexcept { return (_data >> 6) & 0x3f; }
00462
00465 unsigned rcv_order() const noexcept { return (_base >> 6) & 0x3f; }
00466
00469 l4_addr_t base() const noexcept { return _data & (~0UL << 12); }
00470
00473 l4_addr_t snd_base() const noexcept { return _base & (~0UL << 12); }
00474
00477 void snd_base(l4_addr_t b) noexcept { _base = (_base & ~(~0UL << 12)) | (b & (~0UL << 12)); }
00478
00480 bool is_valid() const noexcept { return _base & L4_ITEM_MAP; }
00481
00496 bool cap_received() const noexcept { return (_base & 0x3e) == 0x38; }
00512 bool id_received() const noexcept { return (_base & 0x3e) == 0x3c; }
00528 bool local_id_received() const noexcept { return (_base & 0x3e) == 0x3e; }
00535 bool is_compound() const noexcept { return _base & 1; }
00536 };
00537
00544 class Rcv_fpage : public Gen_fpage
00545 {
00546 public:
00550 Rcv_fpage() noexcept : Gen_fpage(0, 0), _rcv_task(L4_INVALID_CAP) {}
00551
00561 Rcv_fpage(l4_fpage_t const &fp, l4_addr_t snd_base = 0,
00562 l4_cap_idx_t rcv_task = L4_INVALID_CAP) noexcept
00563 : Gen_fpage(L4_ITEM_MAP | (snd_base & (~0UL << 12))
00564 | (l4_is_valid_cap(rcv_task) ? L4_RCV_ITEM_FORWARD_MAPPINGS : 0),
00565 fp.raw),
00566 _rcv_task(rcv_task)
00567 {}
00568
00578 static Rcv_fpage obj(l4_cap_idx_t base, int order, l4_addr_t snd_base = 0,
00579 L4::Cap<void> rcv_task = L4::Cap<void>::Invalid) noexcept
00580 {
00581 return Rcv_fpage(l4_obj_fpage(base, order, 0), snd_base,
00582 rcv_task.cap());
00583 }
00584
00594 static Rcv_fpage mem(l4_addr_t base, int order, l4_addr_t snd_base = 0,
00595 L4::Cap<void> rcv_task = L4::Cap<void>::Invalid) noexcept
00596 {
00597 return Rcv_fpage(l4_fpage(base, order, 0), snd_base, rcv_task.cap());
00598 }
00599
00609 static Rcv_fpage io(unsigned long base, int order, l4_addr_t snd_base = 0,

```

```

00610         L4::Cap<void> rcv_task = L4::Cap<void>::Invalid) noexcept
00611     {
00612         return Rcv_fpage(l4_iofpage(base, order), snd_base, rcv_task.cap());
00613     }
00614
00620     l4_cap_idx_t rcv_task() const { return _rcv_task; }
00621
00625     bool forward_mappings() const noexcept
00626     { return _base & L4_RCV_ITEM_FORWARD_MAPPINGS; }
00627
00628 protected:
00629     l4_cap_idx_t _rcv_task;
00630 };
00631
00632
00633 namespace Msg {
00634
00635     // Snd_fpage are out items
00636     template<> struct Class<L4::Ipc::Snd_fpage> : Cls_item {};
00637
00638     // Rcv_fpage are buffer items
00639     template<> struct Class<L4::Ipc::Rcv_fpage> : Cls_buffer {};
00640
00641     template<>
00642     struct Clnt_val_ops<L4::Ipc::Rcv_fpage, Dir_in, Cls_buffer>
00643     : Clnt_noops<L4::Ipc::Rcv_fpage>
00644     {
00645         using Clnt_noops<L4::Ipc::Rcv_fpage>::to_msg;
00646
00647         static int to_msg(char *msg, unsigned offs, unsigned limit,
00648             L4::Ipc::Rcv_fpage arg, Dir_in, Cls_buffer) noexcept
00649         {
00650             offs = align_to<l4_umword_t>(offs);
00651             unsigned words = arg.forward_mappings() ? 3 : 2;
00652             if (L4_UNLIKELY(!check_size<l4_umword_t>(offs, limit, words)))
00653                 return -L4_EMSGTOOLONG;
00654             auto *buf = reinterpret_cast<l4_umword_t*>(msg + offs);
00655             *buf++ = arg.base_x();
00656             *buf++ = arg.data();
00657             if (arg.forward_mappings())
00658                 *buf++ = arg.rcv_task();
00659             return offs + sizeof(l4_umword_t) * words;
00660         }
00661     };
00662
00663
00664     // Remove receive buffers from server-side arguments
00665     template<> struct Elem<L4::Ipc::Rcv_fpage>
00666     {
00667         typedef L4::Ipc::Rcv_fpage arg_type;
00668         typedef void svr_type;
00669         typedef void svr_arg_type;
00670         enum { Is_optional = false };
00671     };
00672
00673     // Small_buf are buffer items
00674     template<> struct Class<L4::Ipc::Small_buf> : Cls_buffer {};
00675
00676     // Remove receive buffers from server-side arguments
00677     template<> struct Elem<L4::Ipc::Small_buf>
00678     {
00679         typedef L4::Ipc::Small_buf arg_type;
00680         typedef void svr_type;
00681         typedef void svr_arg_type;
00682         enum { Is_optional = false };
00683     };
00684 } // namespace Msg
00685
00686 // L4::Cap<> handling
00687
00698 template<typename T> class Cap
00699 {
00700     template<typename O> friend class Cap;
00701     l4_umword_t _cap_n_rights;
00702
00703 public:
00704     enum
00705     {
00711         Rights_mask = 0xff,
00712
00717         Cap_mask    = L4_CAP_MASK
00718     };
00719
00721     template<typename O>
00722     Cap(Cap<O> const &o) noexcept : _cap_n_rights(o._cap_n_rights)
00723     {
00724         L4::Cap<T>::template check_convertible_from<O>();

```

```

00725     }
00726
00728     Cap(L4::Cap<T> cap) noexcept
00729     : _cap_n_rights((cap.cap() & Cap_mask) | (cap ? L4_CAP_FPAGE_R : 0))
00730     {}
00731
00733     template<typename O>
00734     Cap(L4::Cap<O> cap) noexcept
00735     : _cap_n_rights((cap.cap() & Cap_mask) | (cap ? L4_CAP_FPAGE_R : 0))
00736     {
00737         L4::Cap<T>::template check_convertible_from<O>();
00738     }
00739
00741     Cap() noexcept : _cap_n_rights(L4_INVALID_CAP) {}
00742
00750     Cap(L4::Cap<T> cap, unsigned char rights) noexcept
00751     : _cap_n_rights((cap.cap() & Cap_mask) | (rights & Rights_mask)) {}
00752
00758     static Cap from_ci(l4_cap_idx_t c) noexcept
00759     { return Cap(L4::Cap<T>(c & Cap_mask), c & Rights_mask); }
00760
00762     L4::Cap<T> cap() const noexcept
00763     { return L4::Cap<T>(_cap_n_rights & Cap_mask); }
00764
00766     unsigned rights() const noexcept
00767     { return _cap_n_rights & Rights_mask; }
00768
00770     L4::Ipc::Snd_fpage fpage() const noexcept
00771     { return L4::Ipc::Snd_fpage(cap(), rights()); }
00772
00774     bool is_valid() const noexcept
00775     { return !(_cap_n_rights & L4_INVALID_CAP_BIT); }
00776 };
00777
00784     template<typename T>
00785     Cap<T> make_cap(L4::Cap<T> cap, unsigned rights) noexcept
00786     { return Cap<T>(cap, rights); }
00787
00794     template<typename T>
00795     Cap<T> make_cap_rw(L4::Cap<T> cap) noexcept
00796     { return Cap<T>(cap, L4_CAP_FPAGE_RW); }
00797
00804     template<typename T>
00805     Cap<T> make_cap_rws(L4::Cap<T> cap) noexcept
00806     { return Cap<T>(cap, L4_CAP_FPAGE_RWS); }
00807
00823     template<typename T>
00824     Cap<T> make_cap_full(L4::Cap<T> cap) noexcept
00825     { return Cap<T>(cap, L4_CAP_FPAGE_RWS | L4_FPAGE_C_OBJ_RIGHTS); }
00826
00827     // caps are special the have an invalid representation
00828     template<typename T> struct L4_EXPORT Opt< Cap<T> >
00829     {
00830         Cap<T> _value;
00831         Opt() noexcept {}
00832         Opt(Cap<T> value) noexcept : _value(value) {}
00833         Opt(L4::Cap<T> value) noexcept : _value(value) {}
00834         Opt &operator = (Cap<T> value) noexcept
00835         { this->_value = value; return *this; }
00836         Opt &operator = (L4::Cap<T> value) noexcept
00837         { this->_value = value; return *this; }
00838
00839         Cap<T> value() const noexcept { return this->_value; }
00840         bool is_valid() const noexcept { return this->_value.is_valid(); }
00841     };
00842
00843
00844     namespace Msg {
00845         // prohibit L4::Cap as argument
00846         template<typename A>
00847         struct Is_valid_rpc_type< L4::Cap<A> > : L4::Types::False {};
00848
00849         template<typename A> struct Class< Cap<A> > : Cls_item {};
00850         template<typename A> struct Elem< Cap<A> >
00851         {
00852             enum { Is_optional = false };
00853             typedef Cap<A> arg_type;
00854             typedef L4::Ipc::Snd_fpage svr_type;
00855             typedef L4::Ipc::Snd_fpage svr_arg_type;
00856         };
00857
00858
00859         template<typename A, typename CLASS>
00860         struct Svr_val_ops<Cap<A>, Dir_in, CLASS> :
00861             Svr_val_ops<L4::Ipc::Snd_fpage, Dir_in, CLASS>
00862         {};
00863

```

```

00864 template<typename A, typename CLASS>
00865 struct Clnt_val_ops<Cap<A>, Dir_in, CLASS> :
00866     Clnt_noops< Cap<A> >
00867 {
00868     using Clnt_noops< Cap<A> >::to_msg;
00869
00870     static int to_msg(char *msg, unsigned offset, unsigned limit,
00871         Cap<A> arg, Dir_in, Cls_item) noexcept
00872     {
00873         // passing an invalid cap as mandatory argument is an error
00874         // XXX: This checks for a client calling error, we could
00875         //      also just ignore this for performance reasons and
00876         //      let the client fail badly (Alex: I'd prefer this)
00877         if (L4_UNLIKELY(!arg.is_valid()))
00878             return -L4_MSGMISSARG;
00879
00880         return msg_add(msg, offset, limit, arg.fpage());
00881     }
00882 };
00883
00884 template<typename A>
00885 struct Elem<Out<L4::Cap<A> > >
00886 {
00887     enum { Is_optional = false };
00888     typedef L4::Cap<A> arg_type;
00889     typedef Ipc::Cap<A> svr_type;
00890     typedef svr_type &svr_arg_type;
00891 };
00892
00893 template<typename A> struct Direction< Out< L4::Cap<A> > > : Dir_out {};
00894 template<typename A> struct Class< Out< L4::Cap<A> > > : Cls_item {};
00895
00896 template<typename A>
00897 struct Clnt_val_ops< L4::Cap<A>, Dir_out, Cls_item > :
00898     Clnt_noops< L4::Cap<A> >
00899 {
00900     using Clnt_noops< L4::Cap<A> >::to_msg;
00901     static int to_msg(char *msg, unsigned offset, unsigned limit,
00902         L4::Cap<A> arg, Dir_in, Cls_buffer) noexcept
00903     {
00904         if (L4_UNLIKELY(!arg.is_valid()))
00905             return -L4_MSGMISSARG; // no buffer inserted
00906         return msg_add(msg, offset, limit, Small_buf(arg));
00907     }
00908 };
00909
00910 template<typename A>
00911 struct Svr_val_ops< L4::Ipc::Cap<A>, Dir_out, Cls_item > :
00912     Svr_noops<Cap<A> &>
00913 {
00914     using Svr_noops<Cap<A> &>::from_svr;
00915     static int from_svr(char *msg, unsigned offset, unsigned limit, long,
00916         Cap<A> arg, Dir_out, Cls_item) noexcept
00917     {
00918         if (L4_UNLIKELY(!arg.is_valid()))
00919             // do not map anything
00920             return msg_add(msg, offset, limit, L4::Ipc::Snd_fpage(arg.cap(), 0));
00921
00922         return msg_add(msg, offset, limit, arg.fpage());
00923     }
00924 };
00925
00926 // prohibit a UTCB pointer as normal RPC argument
00927 template<> struct Is_valid_rpc_type<l4_utcb_t *> : L4::Types::False {};
00928
00929 } // namespace Msg
00930 } // namespace Ipc
00931 } // namespace L4

```

## 17.500 ipc\_varg

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008 #pragma GCC system_header
00009
00010 #include "types"
00011 #include "ipc_basics"
00012

```

```

00013 namespace L4 { namespace Ipc L4_EXPORT {
00014
00015 template< typename T, template <typename X> class B >
00016 struct Generic_va_type : B<T>
00017 {
00018     enum { Id = B<T>::Id };
00019     typedef B<T> ID;
00020     typedef T const &Ret_value;
00021     typedef T Value;
00022
00023     static Ret_value value(void const *d)
00024     { return *reinterpret_cast<Value const *>(d); }
00025
00026     static void const *addr_of(Value const &v) { return &v; }
00027
00028     static unsigned size(void const *) { return sizeof(T); }
00029
00030     static L4_varg_type unsigned_id()
00031     {
00032         return static_cast<L4_varg_type>(Id & ~L4_VARG_TYPE_SIGN);
00033     }
00034
00035     static L4_varg_type signed_id()
00036     {
00037         return static_cast<L4_varg_type>(Id | L4_VARG_TYPE_SIGN);
00038     }
00039
00040     static L4_varg_type id()
00041     {
00042         return static_cast<L4_varg_type>(Id);
00043     }
00044 };
00045
00046 template< typename T > struct Va_type_id;
00047 template<> struct Va_type_id<l4_umword_t> { enum { Id = L4_VARG_TYPE_UMWORD }; };
00048 template<> struct Va_type_id<l4_mword_t> { enum { Id = L4_VARG_TYPE_MWORD }; };
00049 template<> struct Va_type_id<l4_fpage_t> { enum { Id = L4_VARG_TYPE_FPAGE }; };
00050 template<> struct Va_type_id<void> { enum { Id = L4_VARG_TYPE_NIL }; };
00051 template<> struct Va_type_id<char const *> { enum { Id = L4_VARG_TYPE_STRING }; };
00052
00053 template< typename T > struct Va_type;
00054
00055 template<> struct Va_type<l4_umword_t> : Generic_va_type<l4_umword_t, Va_type_id> {};
00056 template<> struct Va_type<l4_mword_t> : Generic_va_type<l4_mword_t, Va_type_id> {};
00057 template<> struct Va_type<l4_fpage_t> : Generic_va_type<l4_fpage_t, Va_type_id> {};
00058
00059 template<> struct Va_type<void>
00060 {
00061     typedef void Ret_value;
00062     typedef void Value;
00063
00064     static void const *addr_of(void) { return 0; }
00065
00066     static void value(void const *) {}
00067     static L4_varg_type id() { return L4_VARG_TYPE_NIL; }
00068     static unsigned size(void const *) { return 0; }
00069 };
00070
00071 template<> struct Va_type<char const *>
00072 {
00073     typedef char const *Ret_value;
00074     typedef char const *Value;
00075
00076     static void const *addr_of(Value v) { return v; }
00077
00078     static L4_varg_type id() { return L4_VARG_TYPE_STRING; }
00079     static unsigned size(void const *s)
00080     {
00081         char const *_s = reinterpret_cast<char const *>(s);
00082         int l = 1;
00083         while (*_s)
00084         {
00085             ++_s; ++l;
00086         }
00087         return l;
00088     }
00089
00090     static Ret_value value(void const *d) { return static_cast<char const *>(d); }
00091 };
00092
00096 class Varg
00097 {
00098 private:
00099     enum { Direct_data = 0x8000 };
00100     l4_umword_t _tag;
00101     char const *_d;
00102

```

```

00103 public:
00104
00106     typedef l4_umword_t Tag;
00107
00109     L4_varg_type type() const { return static_cast<L4_varg_type>(_tag & 0xff); }
00114     unsigned length() const { return _tag >> 16; }
00116     Tag tag() const { return _tag & ~Direct_data; }
00118     void tag(Tag tag) { _tag = tag; }
00120     void data(char const *d) { _d = d; }
00121
00123     char const *data() const
00124     {
00125         if (_tag & Direct_data)
00126         {
00127             union T { char const *d; char v[sizeof(char const *)]; };
00128             return reinterpret_cast<T const *>(&_d)->v;
00129         }
00130         return _d;
00131     }
00132
00134 #if __cplusplus >= 201103L
00135     Varg() = default;
00136 #else
00137     Varg() {}
00138 #endif
00139
00141     Varg(L4_varg_type t, void const *v, int len)
00142     : _tag(t | (static_cast<l4_mword_t>(len) << 16)),
00143       _d(static_cast<char const *>(v))
00144     {}
00145
00146     static Varg nil() { return Varg(L4_VARG_TYPE_NIL, 0, 0); }
00147
00154     template< typename V >
00155     typename Va_type<V>::Ret_value value() const
00156     {
00157         if (_tag & Direct_data)
00158         {
00159             union X { char const *d; V v; };
00160             return reinterpret_cast<X const *>(&_d).v;
00161         }
00162         return Va_type<V>::value(_d);
00163     }
00164
00165
00166
00168     template< typename T >
00169     bool is_of() const { return Va_type<T>::id() == type(); }
00170
00172     bool is_nil() const { return is_of<void>(); }
00173
00175     bool is_of_int() const
00176     { return (type() & ~L4_VARG_TYPE_SIGN) == L4_VARG_TYPE_UMWORD; }
00177
00184     template< typename T >
00185     bool get_value(typename Va_type<T>::Value *v) const
00186     {
00187         if (!is_of<T>())
00188             return false;
00189
00190         *v = this->value<T>();
00191         return true;
00192     }
00193
00195     template< typename T >
00196     void set_value(void const *d)
00197     {
00198         typedef Va_type<T> Vt;
00199         _tag = Vt::id() | (Vt::size(d) << 16);
00200         _d = static_cast<char const *>(d);
00201     }
00202
00204     template<typename T>
00205     void set_direct_value(T val, typename L4::Types::Enable_if<sizeof(T) <= sizeof(char const *)>,
00206                          bool>::type = true)
00207     {
00208         static_assert(sizeof(T) <= sizeof(char const *), "direct Varg value too big");
00209         typedef Va_type<T> Vt;
00210         _tag = Vt::id() | (sizeof(T) << 16) | Direct_data;
00211         union X { char const *d; T v; };
00212         reinterpret_cast<X *>(&_d).v = val;
00213     }
00215     template<typename T> explicit
00216     Varg(T const *data) { set_value<T>(data); }
00218     Varg(char const *data) { set_value<char const *>(data); }
00219

```

```

00221     template<typename T> explicit
00222     Varg(T data, typename L4::Types::Enable_if<sizeof(T) <= sizeof(char const *)>, bool>::type = true)
00223     { set_direct_value<T>(data); }
00224 };
00225
00226
00227 template<typename T>
00228 class Varg_t : public Varg
00229 {
00230 public:
00231     typedef typename Va_type<T>::Value Value;
00232     explicit Varg_t(Value v) : Varg()
00233     { _data = v; set_value<T>(Va_type<T>::addr_of(_data)); }
00234
00235 private:
00236     Value _data;
00237 };
00238
00239 template<unsigned MAX = L4_UTCB_GENERIC_DATA_SIZE>
00240 class Varg_list;
00241
00253 class Varg_list_ref
00254 {
00255 private:
00256     template<unsigned T>
00257     friend class Varg_list;
00258
00260     class Iter_state
00261     {
00262     private:
00263         using M = l4_umword_t;
00264         using Mp = M const *;
00265         Mp _c;
00266         Mp _e;
00267
00269         Mp next_arg(Varg const &a) const
00270         {
00271             return _c + 1 + (Msg::align_to<M>(a.length()) / sizeof(M));
00272         }
00273
00274     public:
00276         Iter_state() : _c(nullptr), _e(nullptr) {}
00277
00279         Iter_state(Mp c, Mp e) : _c(c), _e(e)
00280         {}
00281
00283         bool valid() const
00284         { return _c && _c < _e; }
00285
00287         Mp begin() const { return _c; }
00288
00290         Mp end() const { return _e; }
00291
00296         Varg pop()
00297         {
00298             if (!valid())
00299                 return Varg::nil();
00300
00301             Varg a;
00302             a.tag(_c[0]);
00303             a.data(reinterpret_cast<char const *>(&_c[1]));
00304             _c = next_arg(a);
00305             if (_c > _e)
00306                 return Varg::nil();
00307
00308             return a;
00309         }
00310
00312         bool operator == (Iter_state const &o) const
00313         { return _c == o._c; }
00314
00316         bool operator != (Iter_state const &o) const
00317         { return _c != o._c; }
00318     };
00319
00320     Iter_state _s;
00321
00322 public:
00324     Varg_list_ref() = default;
00325
00332     Varg_list_ref(void const *start, void const *end)
00333     : _s(reinterpret_cast<l4_umword_t const *>(start),
00334         reinterpret_cast<l4_umword_t const *>(end))
00335     {}
00336
00338     class Iterator
00339     {

```



```

00340 private:
00341     Iter_state _s;
00342     Varg _a;
00343
00344 public:
00346     Iterator(Iter_state const &s)
00347     : _s(s)
00348     {
00349         _a = _s.pop();
00350     }
00351
00353     explicit operator bool () const
00354     { return !_a.is_nil(); }
00355
00357     Iterator &operator ++ ()
00358     {
00359         if (!_a.is_nil())
00360             _a = _s.pop();
00361
00362         return *this;
00363     }
00364
00366     Varg operator * () const
00367     { return _a; }
00368
00370     bool equals(Iterator const &o) const
00371     {
00372         if (_a.is_nil() && o._a.is_nil())
00373             return true;
00374
00375         return _s == o._s;
00376     }
00377
00378     bool operator == (Iterator const &o) const
00379     { return equals(o); }
00380
00381     bool operator != (Iterator const &o) const
00382     { return !equals(o); }
00383 };
00384
00386 Varg pop_front()
00387 { return _s.pop(); }
00388
00390 Varg next()
00391     L4_DEPRECATED("Use range for or pop_front.")
00392 { return _s.pop(); }
00393
00395 Iterator begin() const
00396 { return Iterator(_s); }
00397
00399 Iterator end() const
00400 { return Iterator(Iter_state()); }
00401 };
00402
00410 template<unsigned MAX>
00411 class Varg_list : public Varg_list_ref
00412 {
00413     l4_umword_t data[MAX];
00414     Varg_list(Varg_list const &);
00415
00416 public:
00418     Varg_list(Varg_list_ref const &r)
00419     {
00420         if (!r._s.valid())
00421             return;
00422
00423         l4_umword_t const *rs = r._s.begin();
00424         unsigned c = r._s.end() - rs;
00425         for (unsigned i = 0; i < c; ++i)
00426             data[i] = rs[i];
00427
00428         this->_s = Iter_state(data, data + c);
00429     }
00430 };
00431
00432 namespace Msg {
00433 template<> struct Elem<Varg const *>
00434 {
00435     typedef Varg const *arg_type;
00436     typedef Varg_list_ref svr_type;
00437     typedef Varg_list_ref svr_arg_type;
00438     enum { Is_optional = false };
00439 };
00440 };
00441
00442 template<> struct Is_valid_rpc_type<Varg> : L4::Types::False {};
00443 template<> struct Is_valid_rpc_type<Varg *> : L4::Types::False {};

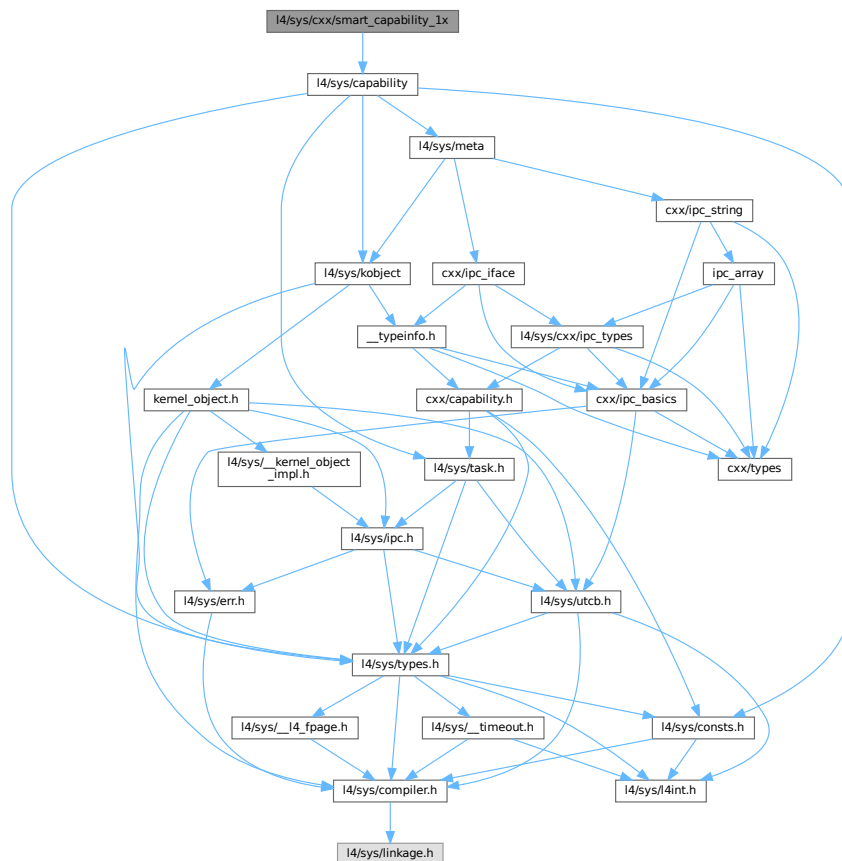
```

```

00444 template<> struct Is_valid_rpc_type<Varg &> : L4::Types::False {};
00445 template<> struct Is_valid_rpc_type<Varg const &> : L4::Types::False {};
00446
00447 template<> struct Direction<Varg const *> : Dir_in {};
00448 template<> struct Class<Varg const *> : Cls_data {};
00449
00450 template<typename DIR, typename CLASS>
00451 struct Clnt_val_ops<Varg, DIR, CLASS>;
00452
00453 template<>
00454 struct Clnt_val_ops<Varg, Dir_in, Cls_data> :
00455     Clnt_noops<Varg const &>
00456 {
00457     using Clnt_noops<Varg const &>::to_msg;
00458     static int to_msg(char *msg, unsigned offs, unsigned limit,
00459         Varg const &a, Dir_in, Cls_data)
00460     {
00461         for (Varg const *i = &a; i->tag(); ++i)
00462         {
00463             offs = align_to<l4_umword_t>(offs);
00464             if (L4_UNLIKELY(!check_size<l4_umword_t>(offs, limit)))
00465                 return -L4_MSGTOOLONG;
00466             *reinterpret_cast<l4_umword_t*>(msg + offs) = i->tag();
00467             offs += sizeof(l4_umword_t);
00468             if (L4_UNLIKELY(!check_size<char>(offs, limit, i->length())))
00469                 return -L4_MSGTOOLONG;
00470             char const *d = i->data();
00471             for (unsigned x = 0; x < i->length(); ++x)
00472                 msg[offs++] = *d++;
00473         }
00474         return offs;
00475     }
00476 };
00477
00478 template<>
00479 struct Svr_val_ops<Varg_list_ref, Dir_in, Cls_data> :
00480     Svr_noops<Varg_list_ref>
00481 {
00482     using Svr_noops<Varg_list_ref>::to_svr;
00483     static int to_svr(char *msg, unsigned offset, unsigned limit,
00484         Varg_list_ref &a, Dir_in, Cls_data)
00485     {
00486         unsigned start = align_to<l4_umword_t>(offset);
00487         unsigned offs;
00488         for (offs = start; offs < limit;)
00489         {
00490             unsigned noffs = align_to<l4_umword_t>(offs);
00491             if (L4_UNLIKELY(!check_size<l4_umword_t>(noffs, limit)))
00492                 break;
00493             offs = noffs;
00494             Varg arg;
00495             arg.tag(*reinterpret_cast<l4_umword_t*>(msg + offs));
00496             if (!arg.tag())
00497                 break;
00498             offs += sizeof(l4_umword_t);
00499             if (L4_UNLIKELY(!check_size<char>(offs, limit, arg.length())))
00500                 return -L4_MSGTOOLONG;
00501             offs += arg.length();
00502         }
00503         a = Varg_list_ref(msg + start, msg + align_to<l4_umword_t>(offs));
00504         return offs;
00505     }
00506 };
00507
00508
00509
00510
00511 }
00512 };
00513 }
00514 }

```

```
#include <l4/sys/capability>
Include dependency graph for smart_capability_1x:
```

[illegible]

- namespace **L4**  
*L4 low-level kernel interface.*

## 17.502 smart\_capability\_1x

[Go to the documentation of this file.](#)

```

00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <l4/sys/capability>
00011
00012 namespace L4 { namespace Detail {
00013
00014     template< typename T, typename IMPL >
00015     class Smart_cap_base : public Cap_base, protected IMPL
00016     {
00017     protected:
00018         template<typename X>
00019         static IMPL &impl(Smart_cap_base<X, IMPL> &o) { return o; }
00020
00021         template<typename X>
00022         static IMPL const &impl(Smart_cap_base<X, IMPL> const &o) { return o; }
00023
00024     public:
00025         template<typename X, typename I>
00026         friend class ::L4::Detail::Smart_cap_base;
00027
00028         Smart_cap_base(Smart_cap_base const &) = delete;
00029         Smart_cap_base &operator = (Smart_cap_base const &) = delete;
00030
00031         Smart_cap_base() noexcept : Cap_base(Invalid) {}
00032
00033         explicit Smart_cap_base(Cap_base::Cap_type t) noexcept
00034         : Cap_base(t)
00035         {}
00036
00037         template<typename O>
00038         explicit constexpr Smart_cap_base(Cap<O> c) noexcept
00039         : Cap_base(c.cap())
00040         {}
00041
00042         template<typename O>
00043         explicit constexpr Smart_cap_base(Cap<O> c, IMPL const &impl) noexcept
00044         : Cap_base(c.cap()), IMPL(impl)
00045         {}
00046
00047         Cap<T> release() noexcept
00048         {
00049             l4_cap_idx_t c = this->cap();
00050             IMPL::invalidate(*this);
00051             return Cap<T>(c);
00052         }
00053
00054         void reset()
00055         { IMPL::free(*this); }
00056
00057         Cap<T> operator -> () const noexcept { return Cap<T>(this->cap()); }
00058         Cap<T> get() const noexcept { return Cap<T>(this->cap()); }
00059         ~Smart_cap_base() noexcept { IMPL::free(*this); }
00060     };
00061
00062     template< typename T, typename IMPL >
00063     class Unique_cap_impl final : public Smart_cap_base<T, IMPL>
00064     {
00065     private:
00066         typedef Smart_cap_base<T, IMPL> Base;
00067
00068     public:
00069         using Base::Base;
00070         Unique_cap_impl() noexcept = default;
00071
00072         Unique_cap_impl(Unique_cap_impl &&o) noexcept
00073         : Base(o.release(), Base::impl(o))
00074         {}
00075
00076         template<typename O>
00077         Unique_cap_impl(Unique_cap_impl<O, IMPL> &&o) noexcept
00078         : Base(o.release(), Base::impl(o))
00079         { Cap<T>::template check_convertible_from<O>(); }
00080
00081         Unique_cap_impl &operator = (Unique_cap_impl &&o) noexcept

```

```

00087 {
00088     if (&o == this)
00089         return *this;
00090
00091     IMPL::free(*this);
00092     this->_c = o.release().cap();
00093     this->IMPL::operator = (Base::impl(o));
00094     return *this;
00095 }
00096
00097 template<typename O>
00098 Unique_cap_impl &operator = (Unique_cap_impl<O, IMPL> &o) noexcept
00099 {
00100     Cap<T>::template check_convertible_from<O>();
00101
00102     IMPL::free(*this);
00103     this->_c = o.release().cap();
00104     this->IMPL::operator = (Base::impl(o));
00105     return *this;
00106 }
00107 };
00108
00109 template<typename T, typename IMPL>
00110 class Shared_cap_impl final : public Smart_cap_base<T, IMPL>
00111 {
00112 private:
00113     typedef Smart_cap_base<T, IMPL> Base;
00114
00115 public:
00116     using Base::Base;
00117     Shared_cap_impl() noexcept = default;
00118
00119     Shared_cap_impl(Shared_cap_impl &o) noexcept
00120     : Base(o.release(), Base::impl(o))
00121     {}
00122
00123     template<typename O>
00124     Shared_cap_impl(Shared_cap_impl<O, IMPL> &o) noexcept
00125     : Base(o.release(), Base::impl(o))
00126     { Cap<T>::template check_convertible_from<O>(); }
00127
00128     Shared_cap_impl &operator = (Shared_cap_impl &o) noexcept
00129     {
00130         if (&o == this)
00131             return *this;
00132
00133         IMPL::free(*this);
00134         this->_c = o.release().cap();
00135         this->IMPL::operator = (Base::impl(o));
00136         return *this;
00137     }
00138
00139     template<typename O>
00140     Shared_cap_impl &operator = (Shared_cap_impl<O, IMPL> &o) noexcept
00141     {
00142         Cap<T>::template check_convertible_from<O>();
00143
00144         IMPL::free(*this);
00145         this->_c = o.release().cap();
00146         this->IMPL::operator = (Base::impl(o));
00147         return *this;
00148     }
00149
00150     Shared_cap_impl(Shared_cap_impl const &o) noexcept
00151     : Base()
00152     {
00153         this->IMPL::operator = (Base::impl(o));
00154         this->_c = IMPL::copy(o).cap();
00155     }
00156
00157     template<typename O>
00158     Shared_cap_impl(Shared_cap_impl<O, IMPL> const &o) noexcept
00159     : Base()
00160     {
00161         Cap<T>::template check_convertible_from<O>();
00162         this->IMPL::operator = (Base::impl(o));
00163         this->_c = IMPL::copy(o).cap();
00164     }
00165
00166     Shared_cap_impl &operator = (Shared_cap_impl const &o) noexcept
00167     {
00168         if (&o == this)
00169             return *this;
00170
00171         IMPL::free(*this);
00172         this->IMPL::operator = (static_cast<IMPL const &>(o));
00173         this->_c = this->IMPL::copy(o).cap();

```

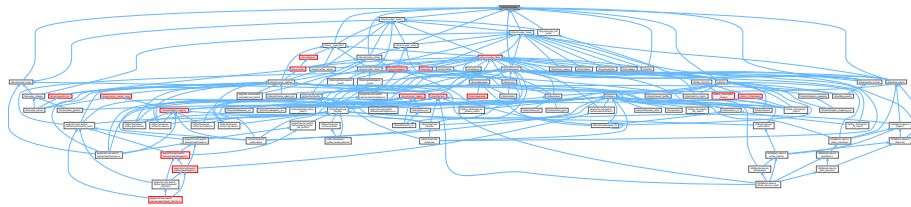
```

00174     return *this;
00175 }
00176
00177 template<typename O>
00178 Shared_cap_impl &operator = (Shared_cap_impl<O, IMPL> const &o) noexcept
00179 {
00180     Cap<T>::template check_convertible_from<O>();
00181     IMPL::free(*this);
00182     this->IMPL::operator = (static_cast<IMPL const &>(o));
00183     this->_c = this->IMPL::copy(o).cap();
00184     return *this;
00185 }
00186 };
00187
00188 }} // L4::Detail

```

## 17.503 l4/sys/cxx/types File Reference

This graph shows which files directly or indirectly include this file:



### Data Structures

- class [L4::Types::Flags< BITS\\_ENUM, UNDERLYING >](#)  
*Template for defining typical [Flags](#) bitmaps.*
- struct [L4::Types::Int\\_for\\_type< T >](#)  
*Metafunction to get an integral type of the same size as *T*.*
- struct [L4::Types::Flags\\_ops\\_t< DT >](#)  
*Mixin class to define a set of friend bitwise operators on *DT*.*
- struct [L4::Types::Flags\\_t< DT, T >](#)  
*Template type to define a flags type with bitwise operations.*
- struct [L4::Types::Bool< V >](#)  
*Boolean meta type.*
- struct [L4::Types::False](#)  
*[False](#) meta value.*
- struct [L4::Types::True](#)  
*[True](#) meta value.*
- struct [L4::Types::Same< A, B >](#)  
*Compare two data types for equality.*

### Namespaces

- namespace [L4](#)  
*[L4](#) low-level kernel interface.*
- namespace [L4::Types](#)  
*[L4](#) basic type helpers for C++.*

## Macros

- `#define L4_TYPES_FLAGS_OPS_DEF(T)`  
*Helper macro to define a set of bitwise operators on an enum type.*

### 17.503.1 Macro Definition Documentation

#### 17.503.1.1 L4\_TYPES\_FLAGS\_OPS\_DEF

```
#define L4_TYPES_FLAGS_OPS_DEF(  
    T)
```

#### Value:

```
friend constexpr T operator ~ (T f)
{
    return T(~static_cast<typename L4::Types::Int_for_type<T>::type>(f));
}

friend constexpr T operator | (T l, T r)
{
    return T(static_cast<typename L4::Types::Int_for_type<T>::type>(l)
        | static_cast<typename L4::Types::Int_for_type<T>::type>(r));
}

friend constexpr T operator & (T l, T r)
{
    return T(static_cast<typename L4::Types::Int_for_type<T>::type>(l)
        & static_cast<typename L4::Types::Int_for_type<T>::type>(r));
}
```

Helper macro to define a set of bitwise operators on an enum type.

This allows to use the enum type as bitmask type with '&', '|', and '~' operators that keep the enum type as result.

Definition at line 195 of file [types](#).

## 17.504 types

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008
00009
00010 #pragma once
00011
00012 // very simple type traits for basic L4 functions, for a more complete set
00013 // use <l4/cxx/type_traits> or the standard <type_traits>.
00014
00015 namespace L4 {
00016
00020 namespace Types {
00021
00051     template<typename BITS_ENUM, typename UNDERLYING = unsigned long>
00052     class Flags
00053     {
00054     public:
00056         typedef UNDERLYING value_type;
00058         typedef BITS_ENUM bits_enum_type;
00060         typedef Flags<BITS_ENUM, UNDERLYING> type;
00061
00062     private:
00063         value_type _v;
```

```

00064     explicit Flags(value_type v) : _v(v) {}
00065
00066 public:
00067     enum None_type { None };
00068
00069     Flags(None_type) : _v(0) {}
00070
00071     Flags() : _v(0) {}
00072
00073     Flags(BITS_ENUM e) : _v((value_type{1}) << e) {}
00074
00075     static type from_raw(value_type v) { return type(v); }
00076
00077     explicit operator bool () const
00078     { return _v != 0; }
00079
00080     bool operator ! () const { return _v == 0; }
00081
00082     friend type operator | (type lhs, type rhs)
00083     { return type(lhs._v | rhs._v); }
00084
00085     friend type operator | (type lhs, bits_enum_type rhs)
00086     { return lhs | type(rhs); }
00087
00088     friend type operator & (type lhs, type rhs)
00089     { return type(lhs._v & rhs._v); }
00090
00091     friend type operator & (type lhs, bits_enum_type rhs)
00092     { return lhs & type(rhs); }
00093
00094     type &operator |= (type rhs) { _v |= rhs._v; return *this; }
00095     type &operator |= (bits_enum_type rhs) { return operator |= (type(rhs)); }
00096
00097     type &operator &= (type rhs) { _v &= rhs._v; return *this; }
00098     type &operator &= (bits_enum_type rhs) { return operator &= (type(rhs)); }
00099
00100     type operator ~ () const { return type(~_v); }
00101
00102     type &clear(bits_enum_type flag) { return operator &= (~type(flag)); }
00103
00104     value_type as_value() const { return _v; }
00105 };
00106
00107 template<unsigned SIZE, bool = true> struct Int_for_size;
00108
00109 template<> struct Int_for_size<sizeof(unsigned char), true>
00110 { typedef unsigned char type; };
00111
00112 template<> struct Int_for_size<sizeof(unsigned short),
00113                             (sizeof(unsigned short) > sizeof(unsigned char))>
00114 { typedef unsigned short type; };
00115
00116 template<> struct Int_for_size<sizeof(unsigned),
00117                             (sizeof(unsigned) > sizeof(unsigned short))>
00118 { typedef unsigned type; };
00119
00120 template<> struct Int_for_size<sizeof(unsigned long),
00121                             (sizeof(unsigned long) > sizeof(unsigned))>
00122 { typedef unsigned long type; };
00123
00124 template<> struct Int_for_size<sizeof(unsigned long long),
00125                             (sizeof(unsigned long long) > sizeof(unsigned long))>
00126 { typedef unsigned long long type; };
00127
00128 template<typename T> struct Int_for_type
00129 {
00130     typedef typename Int_for_size<sizeof(T)>::type type;
00131 };
00132
00133 #define L4_TYPES_FLAGS_OPS_DEF(T)
00134     friend constexpr T operator ~ (T f)
00135     {
00136         return T(~static_cast<typename L4::Types::Int_for_type<T>::type>(f));
00137     }
00138
00139     friend constexpr T operator | (T l, T r)
00140     {
00141         return T(static_cast<typename L4::Types::Int_for_type<T>::type>(l)
00142                 | static_cast<typename L4::Types::Int_for_type<T>::type>(r));
00143     }
00144
00145     friend constexpr T operator & (T l, T r)
00146     {
00147         return T(static_cast<typename L4::Types::Int_for_type<T>::type>(l)
00148                 & static_cast<typename L4::Types::Int_for_type<T>::type>(r));
00149     }
00150
00151

```



```

00219     template<typename DT>
00220     struct Flags_ops_t
00221     {
00223         friend constexpr DT operator | (DT l, DT r)
00224         { return DT(l.raw | r.raw); }
00225
00227         friend constexpr DT operator & (DT l, DT r)
00228         { return DT(l.raw & r.raw); }
00229
00231         friend constexpr bool operator == (DT l, DT r)
00232         { return l.raw == r.raw; }
00233
00235         friend constexpr bool operator != (DT l, DT r)
00236         { return l.raw != r.raw; }
00237
00239         DT operator |= (DT r)
00240         {
00241             static_cast<DT *>(this)->raw |= r.raw;
00242             return *static_cast<DT *>(this);
00243         }
00244
00246         DT operator &= (DT r)
00247         {
00248             static_cast<DT *>(this)->raw &= r.raw;
00249             return *static_cast<DT *>(this);
00250         }
00251
00253         explicit constexpr operator bool () const
00254         {
00255             return static_cast<DT const *>(this)->raw != 0;
00256         }
00257
00259         constexpr DT operator ~ () const
00260         { return DT(~static_cast<DT const *>(this)->raw); }
00261     };
00262
00271     template<typename DT, typename T>
00272     struct Flags_t : Flags_ops_t<Flags_t<DT, T>
00273     {
00275         T raw;
00277         Flags_t() = default;
00279         explicit constexpr Flags_t(T f) : raw(f) {}
00280     };
00281
00282
00288     template< bool V > struct Bool
00289     {
00290         typedef Bool<V> type;
00291         enum { value = V };
00292     };
00293
00296     struct False : Bool<false> {};
00297
00300     struct True : Bool<true> {};
00301
00302     /*****/
00311     template<typename A, typename B>
00312     struct Same : False {};
00313
00314     template<typename A>
00315     struct Same<A, A> : True {};
00316
00317     template<bool EXP, typename T = void> struct Enable_if {};
00318     template<typename T> struct Enable_if<true, T> { typedef T type; };
00319
00320     template<typename T1, typename T2, typename T = void>
00321     struct Enable_if_same : Enable_if<Same<T1, T2>::value, T> {};
00322
00323     template<typename T> struct Remove_const { typedef T type; };
00324     template<typename T> struct Remove_const<T const> { typedef T type; };
00325     template<typename T> struct Remove_volatile { typedef T type; };
00326     template<typename T> struct Remove_volatile<T volatile> { typedef T type; };
00327     template<typename T> struct Remove_cv
00328     { typedef typename Remove_const<typename Remove_volatile<T>::type>::type type; };
00329
00330     template<typename T> struct Remove_pointer { typedef T type; };
00331     template<typename T> struct Remove_pointer<T*> { typedef T type; };
00332     template<typename T> struct Remove_reference { typedef T type; };
00333     template<typename T> struct Remove_reference<T&> { typedef T type; };
00334     template<typename T> struct Remove_pr { typedef T type; };
00335     template<typename T> struct Remove_pr<T&> { typedef T type; };
00336     template<typename T> struct Remove_pr<T*> { typedef T type; };
00337 } // Types
00338 } // L4

```

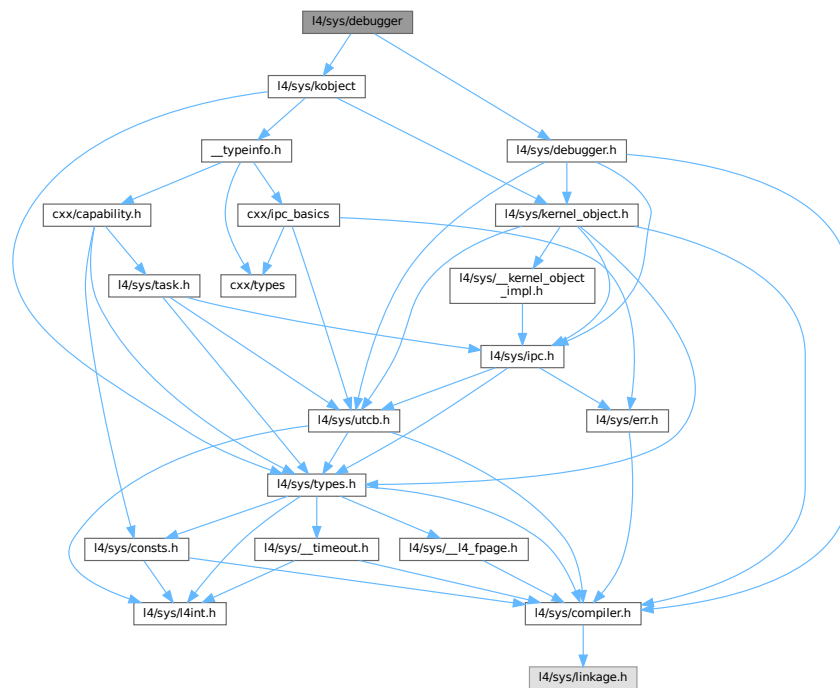
## 17.505 I4/sys/debugger File Reference

The debugger interface specifies common debugging related definitions.

```
#include <l4/sys/debugger.h>
```

```
#include <l4/sys/kobject>
```

Include dependency graph for debugger:



### Data Structures

- class [L4::Debugger](#)  
*C++ kernel debugger API.*

### Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*

### 17.505.1 Detailed Description

The debugger interface specifies common debugging related definitions.

Definition in file [debugger](#).

## 17.506 debugger

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2010-2011 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00015 #include <l4/sys/debugger.h>
00016 #include <l4/sys/kobject>
00017
00018 namespace L4 {
00019
00042 class Debugger : public Kobject_t<Debugger, Kobject, L4_PROTO_DEBUGGER>
00043 {
00044 public:
00045     enum
00046     {
00047         Switch_log_on   = L4_DEBUGGER_SWITCH_LOG_ON,
00048         Switch_log_off  = L4_DEBUGGER_SWITCH_LOG_OFF,
00049     };
00050
00059     l4_msgtag_t set_object_name(const char *name,
00060                                l4_utcb_t *utcb = l4_utcb()) noexcept
00061     { return l4_debugger_set_object_name_u(cap(), name, utcb); }
00062
00071     unsigned long global_id(l4_utcb_t *utcb = l4_utcb()) noexcept
00072     { return l4_debugger_global_id_u(cap(), utcb); }
00073
00083     unsigned long kobj_to_id(l4_addr_t kobjp,
00084                              l4_utcb_t *utcb = l4_utcb()) noexcept
00085     { return l4_debugger_kobj_to_id_u(cap(), kobjp, utcb); }
00086
00097     long query_log_typeid(const char *name, unsigned idx,
00098                           l4_utcb_t *utcb = l4_utcb()) noexcept
00099     { return l4_debugger_query_log_typeid_u(cap(), name, idx, utcb); }
00100
00116     long query_log_name(unsigned idx,
00117                          char *name, unsigned namelen,
00118                          char *shortname, unsigned shortnamelen,
00119                          l4_utcb_t *utcb = l4_utcb()) noexcept
00120     {
00121         return l4_debugger_query_log_name_u(cap(), idx, name, namelen,
00122                                             shortname, shortnamelen, utcb);
00123     }
00124
00133     l4_msgtag_t switch_log(const char *name, unsigned on_off,
00134                             l4_utcb_t *utcb = l4_utcb()) noexcept
00135     { return l4_debugger_switch_log_u(cap(), name, on_off, utcb); }
00136
00148     l4_msgtag_t get_object_name(unsigned id, char *name, unsigned size,
00149                                  l4_utcb_t *utcb = l4_utcb()) noexcept
00150     { return l4_debugger_get_object_name_u(cap(), id, name, size, utcb); }
00151
00161     l4_msgtag_t add_image_info(l4_addr_t base, const char *name,
00162                                l4_utcb_t *utcb = l4_utcb()) noexcept
00163     { return l4_debugger_add_image_info_u(cap(), base, name, utcb); }
00164 };
00165 }

```

## 17.507 l4/sys/debugger.h File Reference

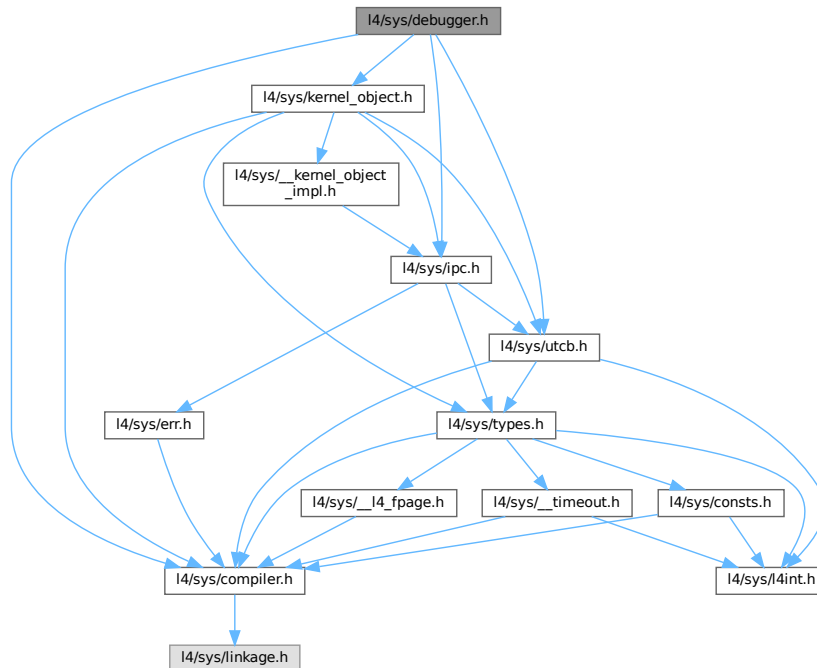
Debugger related definitions.

```

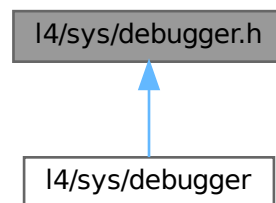
#include <l4/sys/compiler.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>

```

```
#include <l4/sys/kernel_object.h>
Include dependency graph for debugger.h:
```



This graph shows which files directly or indirectly include this file:



## Functions

- [l4\\_msgtag\\_t l4\\_debugger\\_set\\_object\\_name](#) ([l4\\_cap\\_idx\\_t](#) cap, const char \*name) [L4\\_NOTHROW](#)  
Set the name of a kernel object.
- [l4\\_msgtag\\_t l4\\_debugger\\_get\\_object\\_name](#) ([l4\\_cap\\_idx\\_t](#) cap, unsigned id, char \*name, unsigned size) [L4\\_NOTHROW](#)  
Get name of the kernel object with Id id.
- unsigned long [l4\\_debugger\\_global\\_id](#) ([l4\\_cap\\_idx\\_t](#) cap) [L4\\_NOTHROW](#)  
Get the globally unique ID of the object behind a capability.

- unsigned long [l4\\_debugger\\_kobj\\_to\\_id](#) ([l4\\_cap\\_idx\\_t](#) cap, [l4\\_addr\\_t](#) kobjp) [L4\\_NOTHROW](#)  
*Get the globally unique ID of the object behind the kobject pointer.*
- long [l4\\_debugger\\_query\\_log\\_typeid](#) ([l4\\_cap\\_idx\\_t](#) cap, const char \*name, unsigned idx) [L4\\_NOTHROW](#)  
*Query the log-id for a log type.*
- long [l4\\_debugger\\_query\\_log\\_name](#) ([l4\\_cap\\_idx\\_t](#) cap, unsigned idx, char \*name, unsigned namelen, char \*shortname, unsigned shortnamelen) [L4\\_NOTHROW](#)  
*Query the name of a log type given the ID.*
- [l4\\_msgtag\\_t](#) [l4\\_debugger\\_switch\\_log](#) ([l4\\_cap\\_idx\\_t](#) cap, const char \*name, int on\_off) [L4\\_NOTHROW](#)  
*Set or unset log.*
- [l4\\_msgtag\\_t](#) [l4\\_debugger\\_add\\_image\\_info](#) ([l4\\_cap\\_idx\\_t](#) cap, [l4\\_addr\\_t](#) base, const char \*name) [L4\\_NOTHROW](#)  
*Add loaded image information for a task.*

## 17.507.1 Detailed Description

Debugger related definitions.

Definition in file [debugger.h](#).

## 17.508 debugger.h

[Go to the documentation of this file.](#)

```

00001 #pragma once
00007 /*
00008  * (c) 2008-2011 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014
00015 #include <l4/sys/compiler.h>
00016 #include <l4/sys/utcb.h>
00017 #include <l4/sys/ipc.h>
00018
00031
00042 L4_INLINE l4_msgtag_t
00043 l4_debugger_set_object_name(l4_cap_idx_t cap, const char *name) L4_NOTHROW;
00044
00048 L4_INLINE l4_msgtag_t
00049 l4_debugger_set_object_name_u(l4_cap_idx_t cap, const char *name, l4_utcb_t *utcb) L4_NOTHROW;
00050
00063 L4_INLINE l4_msgtag_t
00064 l4_debugger_get_object_name(l4_cap_idx_t cap, unsigned id,
00065                             char *name, unsigned size) L4_NOTHROW;
00066
00070 L4_INLINE l4_msgtag_t
00071 l4_debugger_get_object_name_u(l4_cap_idx_t cap, unsigned id,
00072                               char *name, unsigned size,
00073                               l4_utcb_t *utcb) L4_NOTHROW;
00074
00086 L4_INLINE unsigned long
00087 l4_debugger_global_id(l4_cap_idx_t cap) L4_NOTHROW;
00088
00092 L4_INLINE unsigned long
00093 l4_debugger_global_id_u(l4_cap_idx_t cap, l4_utcb_t *utcb) L4_NOTHROW;
00094
00107 L4_INLINE unsigned long
00108 l4_debugger_kobj_to_id(l4_cap_idx_t cap, l4_addr_t kobjp) L4_NOTHROW;
00109
00113 L4_INLINE unsigned long
00114 l4_debugger_kobj_to_id_u(l4_cap_idx_t cap, l4_addr_t kobjp, l4_utcb_t *utcb) L4_NOTHROW;
00115
00128 L4_INLINE long
00129 l4_debugger_query_log_typeid(l4_cap_idx_t cap, const char *name,
00130                              unsigned idx) L4_NOTHROW;
00131

```

```

00135 L4_INLINE long
00136 l4_debugger_query_log_typeid_u(l4_cap_idx_t cap, const char *name,
00137                                unsigned idx, l4_utcb_t *utcb) L4_NOTHROW;
00138
00155 L4_INLINE long
00156 l4_debugger_query_log_name(l4_cap_idx_t cap, unsigned idx,
00157                             char *name, unsigned namelen,
00158                             char *shortname, unsigned shortnamelen) L4_NOTHROW;
00159
00163 L4_INLINE long
00164 l4_debugger_query_log_name_u(l4_cap_idx_t cap, unsigned idx,
00165                               char *name, unsigned namelen,
00166                               char *shortname, unsigned shortnamelen,
00167                               l4_utcb_t *utcb) L4_NOTHROW;
00168
00179 L4_INLINE l4_msgtag_t
00180 l4_debugger_switch_log(l4_cap_idx_t cap, const char *name,
00181                        int on_off) L4_NOTHROW;
00182
00186 L4_INLINE l4_msgtag_t
00187 l4_debugger_switch_log_u(l4_cap_idx_t cap, const char *name, int on_off,
00188                          l4_utcb_t *utcb) L4_NOTHROW;
00189
00200 L4_INLINE l4_msgtag_t
00201 l4_debugger_add_image_info(l4_cap_idx_t cap, l4_addr_t base,
00202                            const char *name) L4_NOTHROW;
00203
00207 L4_INLINE l4_msgtag_t
00208 l4_debugger_add_image_info_u(l4_cap_idx_t cap, l4_addr_t base, const char *name,
00209                              l4_utcb_t *utcb) L4_NOTHROW;
00210
00211 enum
00212 {
00213     L4_DEBUGGER_NAME_SET_OP      = 0UL,
00214     L4_DEBUGGER_GLOBAL_ID_OP    = 1UL,
00215     L4_DEBUGGER_KOBJ_TO_ID_OP   = 2UL,
00216     L4_DEBUGGER_QUERY_LOG_TYPEID_OP = 3UL,
00217     L4_DEBUGGER_SWITCH_LOG_OP   = 4UL,
00218     L4_DEBUGGER_NAME_GET_OP     = 5UL,
00219     L4_DEBUGGER_QUERY_LOG_NAME_OP = 6UL,
00220     L4_DEBUGGER_ADD_IMAGE_INFO_OP = 7UL,
00221 };
00222
00223 enum
00224 {
00225     L4_DEBUGGER_SWITCH_LOG_ON = 1,
00226     L4_DEBUGGER_SWITCH_LOG_OFF = 0,
00227 };
00228
00229 /* IMPLEMENTATION -----*/
00230
00231 #include <l4/sys/kernel_object.h>
00232
00246 L4_INLINE unsigned
00247 __strcpy_maxlen(char *dst, char const *src, unsigned maxlen)
00248 {
00249     unsigned i;
00250     if (!maxlen)
00251         return 0;
00252
00253     for (i = 0; i < maxlen - 1 && src[i]; ++i)
00254         dst[i] = src[i];
00255     dst[i] = '\0';
00256
00257     return i + 1;
00258 }
00259
00260 L4_INLINE l4_msgtag_t
00261 l4_debugger_set_object_name_u(l4_cap_idx_t cap,
00262                               const char *name, l4_utcb_t *utcb) L4_NOTHROW
00263 {
00264     unsigned i;
00265     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_NAME_SET_OP;
00266     i = __strcpy_maxlen((char *)&l4_utcb_mr_u(utcb)->mr[1], name,
00267                         (L4_UTCB_GENERIC_DATA_SIZE - 2) * sizeof(l4_umword_t));
00268     i = l4_bytes_to_mwords(i);
00269     return l4_invoke_debugger(cap, l4_msgtag(0, 1 + i, 0, 0), utcb);
00270 }
00271
00272 L4_INLINE unsigned long
00273 l4_debugger_global_id_u(l4_cap_idx_t cap, l4_utcb_t *utcb) L4_NOTHROW
00274 {
00275     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_GLOBAL_ID_OP;
00276     if (l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 1, 0, 0), utcb), utcb))
00277         return ~0UL;
00278     return l4_utcb_mr_u(utcb)->mr[0];
00279 }

```

```

00280
00281 L4_INLINE unsigned long
00282 l4_debugger_kobj_to_id_u(l4_cap_idx_t cap, l4_addr_t kobjp, l4_utcb_t *utcb) L4_NOTHROW
00283 {
00284     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_KOBJ_TO_ID_OP;
00285     l4_utcb_mr_u(utcb)->mr[1] = kobjp;
00286     if (l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 2, 0, 0), utcb), utcb))
00287         return ~0UL;
00288     return l4_utcb_mr_u(utcb)->mr[0];
00289 }
00290
00291 L4_INLINE long
00292 l4_debugger_query_log_typeid_u(l4_cap_idx_t cap, const char *name,
00293                                unsigned idx,
00294                                l4_utcb_t *utcb) L4_NOTHROW
00295 {
00296     unsigned i;
00297     long e;
00298     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_QUERY_LOG_TYPEID_OP;
00299     l4_utcb_mr_u(utcb)->mr[1] = idx;
00300     i = __strcpy_maxlen((char *)&l4_utcb_mr_u(utcb)->mr[2], name, 32);
00301     i = l4_bytes_to_mwords(i);
00302     e = l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 2 + i, 0, 0), utcb), utcb);
00303     if (e < 0)
00304         return e;
00305     return l4_utcb_mr_u(utcb)->mr[0];
00306 }
00307
00308 L4_INLINE long
00309 l4_debugger_query_log_name_u(l4_cap_idx_t cap, unsigned idx,
00310                             char *name, unsigned namelen,
00311                             char *shortname, unsigned shortnamelen,
00312                             l4_utcb_t *utcb) L4_NOTHROW
00313 {
00314     long e;
00315     char const *n;
00316     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_QUERY_LOG_NAME_OP;
00317     l4_utcb_mr_u(utcb)->mr[1] = idx;
00318     e = l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 2, 0, 0), utcb), utcb);
00319     if (e < 0)
00320         return e;
00321     n = (char const *)&l4_utcb_mr_u(utcb)->mr[0];
00322     __strcpy_maxlen(name, n, namelen);
00323     __strcpy_maxlen(shortname, n + __builtin_strlen(n) + 1, shortnamelen);
00324     return 0;
00325 }
00326
00327
00328 L4_INLINE l4_msgtag_t
00329 l4_debugger_switch_log_u(l4_cap_idx_t cap, const char *name, int on_off,
00330                          l4_utcb_t *utcb) L4_NOTHROW
00331 {
00332     unsigned i;
00333     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_SWITCH_LOG_OP;
00334     l4_utcb_mr_u(utcb)->mr[1] = on_off;
00335     i = __strcpy_maxlen((char *)&l4_utcb_mr_u(utcb)->mr[2], name, 32);
00336     i = l4_bytes_to_mwords(i);
00337     return l4_invoke_debugger(cap, l4_msgtag(0, 2 + i, 0, 0), utcb);
00338 }
00339
00340 L4_INLINE l4_msgtag_t
00341 l4_debugger_get_object_name_u(l4_cap_idx_t cap, unsigned id,
00342                              char *name, unsigned size,
00343                              l4_utcb_t *utcb) L4_NOTHROW
00344 {
00345     l4_msgtag_t t;
00346     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_NAME_GET_OP;
00347     l4_utcb_mr_u(utcb)->mr[1] = id;
00348     t = l4_invoke_debugger(cap, l4_msgtag(0, 2, 0, 0), utcb);
00349     __strcpy_maxlen(name, (char const *)&l4_utcb_mr_u(utcb)->mr[0], size);
00350     return t;
00351 }
00352
00353 L4_INLINE l4_msgtag_t
00354 l4_debugger_add_image_info_u(l4_cap_idx_t cap, l4_addr_t base,
00355                             const char *name, l4_utcb_t *utcb) L4_NOTHROW
00356 {
00357     unsigned i;
00358     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_ADD_IMAGE_INFO_OP;
00359     l4_utcb_mr_u(utcb)->mr[1] = base;
00360     i = __strcpy_maxlen((char *)&l4_utcb_mr_u(utcb)->mr[2], name,
00361                        (L4_UTCB_GENERIC_DATA_SIZE - 3) * sizeof(l4_umword_t));
00362     i = l4_bytes_to_mwords(i);
00363     return l4_invoke_debugger(cap, l4_msgtag(0, 2 + i, 0, 0), utcb);
00364 }
00365
00366

```

```

00367 L4_INLINE l4_msgtag_t
00368 l4_debugger_set_object_name(l4_cap_idx_t cap,
00369                             const char *name) L4_NOTHROW
00370 {
00371     return l4_debugger_set_object_name_u(cap, name, l4_utcb());
00372 }
00373
00374 L4_INLINE unsigned long
00375 l4_debugger_global_id(l4_cap_idx_t cap) L4_NOTHROW
00376 {
00377     return l4_debugger_global_id_u(cap, l4_utcb());
00378 }
00379
00380 L4_INLINE unsigned long
00381 l4_debugger_kobj_to_id(l4_cap_idx_t cap, l4_addr_t kobjp) L4_NOTHROW
00382 {
00383     return l4_debugger_kobj_to_id_u(cap, kobjp, l4_utcb());
00384 }
00385
00386 L4_INLINE long
00387 l4_debugger_query_log_typeid(l4_cap_idx_t cap, const char *name,
00388                             unsigned idx) L4_NOTHROW
00389 {
00390     return l4_debugger_query_log_typeid_u(cap, name, idx, l4_utcb());
00391 }
00392
00393 L4_INLINE long
00394 l4_debugger_query_log_name(l4_cap_idx_t cap, unsigned idx,
00395                           char *name, unsigned namelen,
00396                           char *shortname, unsigned shortnamelen) L4_NOTHROW
00397 {
00398     return l4_debugger_query_log_name_u(cap, idx, name, namelen,
00399                                         shortname, shortnamelen, l4_utcb());
00400 }
00401
00402 L4_INLINE l4_msgtag_t
00403 l4_debugger_switch_log(l4_cap_idx_t cap, const char *name,
00404                       int on_off) L4_NOTHROW
00405 {
00406     return l4_debugger_switch_log_u(cap, name, on_off, l4_utcb());
00407 }
00408
00409 L4_INLINE l4_msgtag_t
00410 l4_debugger_get_object_name(l4_cap_idx_t cap, unsigned id,
00411                           char *name, unsigned size) L4_NOTHROW
00412 {
00413     return l4_debugger_get_object_name_u(cap, id, name, size, l4_utcb());
00414 }
00415
00416 L4_INLINE l4_msgtag_t
00417 l4_debugger_add_image_info(l4_cap_idx_t cap, l4_addr_t base,
00418                           const char *name) L4_NOTHROW
00419 {
00420     return l4_debugger_add_image_info_u(cap, base, name, l4_utcb());
00421 }

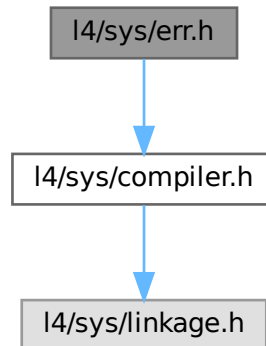
```

## 17.509 l4/sys/err.h File Reference

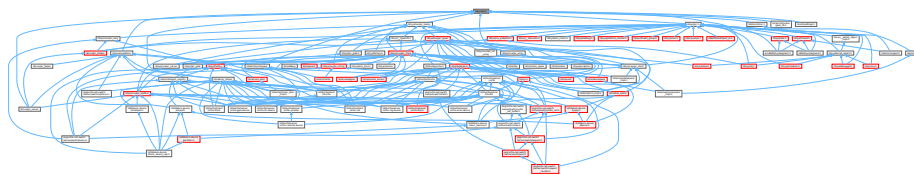
Error codes.



```
#include <l4/sys/compiler.h>
Include dependency graph for err.h:
```



This graph shows which files directly or indirectly include this file:



## Enumerations

```
enum l4_error_code_t {
    L4_EOK = 0, L4_EPERM = 1, L4_ENOENT = 2, L4_EIO = 5,
    L4_ENXIO = 6, L4_E2BIG = 7, L4_EAGAIN = 11, L4_ENOMEM = 12,
    L4_EACCESS = 13, L4_EFAULT = 14, L4_EBUSY = 16, L4_EEXIST = 17,
    L4_ENODEV = 19, L4_ENOTDIR = 20, L4_EINVAL = 22, L4_ENOSPC = 28,
    L4_ERANGE = 34, L4_ENAMETOOLONG = 36, L4_ENOSYS = 38, L4_EBADPROTO = 39,
    L4_EADDRNOTAVAIL = 99, L4_ERRNOMAX = 100, L4_ENOREPLY = 1000, L4_MSGTOOSHORT =
    1001,
    L4_MSGTOOLONG = 1002, L4_MSGMISSARG = 1003, L4_EIPC_LO = 2000, L4_EIPC_HI = 2000 +
    0x1f }
```

*L4 error codes.*

## 17.509.1 Detailed Description

Error codes.

Definition in file [err.h](#).

## 17.510 err.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 #include <l4/sys/compiler.h>
00015
00023
00030 enum l4_error_code_t
00031 {
00032     L4_EOK                = 0,
00033     L4_EPERM              = 1,
00034     L4_ENOENT             = 2,
00035     L4_EIO                = 5,
00036     L4_ENXIO              = 6,
00037     L4_E2BIG              = 7,
00038     L4_EAGAIN             = 11,
00039     L4_ENOMEM             = 12,
00040     L4_EACCESS            = 13,
00041     L4_EFAULT             = 14,
00042     L4_EBUSY              = 16,
00043     L4_EEXIST             = 17,
00044     L4_ENODEV             = 19,
00045     L4_ENOTDIR            = 20,
00046     L4_EINVAL            = 22,
00047     L4_ENOSPC            = 28,
00048     L4_ERANGE             = 34,
00049     L4_ENAMETOOLONG       = 36,
00050     L4_ENOSYS            = 38,
00051     L4_EBADPROTO          = 39,
00052     L4_EADDRNOTAVAIL      = 99,
00053     L4_ERRNOMAX           = 100,
00054
00055     L4_ENOREPLY           = 1000,
00056     L4_MSGTOOSHORT        = 1001,
00057     L4_MSGTOOLONG         = 1002,
00058     L4_MSGMISSARG         = 1003,
00059
00060     L4_EIPC_LO            = 2000,
00061     L4_EIPC_HI            = 2000 + 0x1f,
00062 };
00063
00064 L4_BEGIN_DECLS
00065 L4_CV char const *l4sys_errtostr(long err) L4_NOTHROW;
00066 L4_END_DECLS
00067
00068

```

## 17.511 l4/sys/exception File Reference

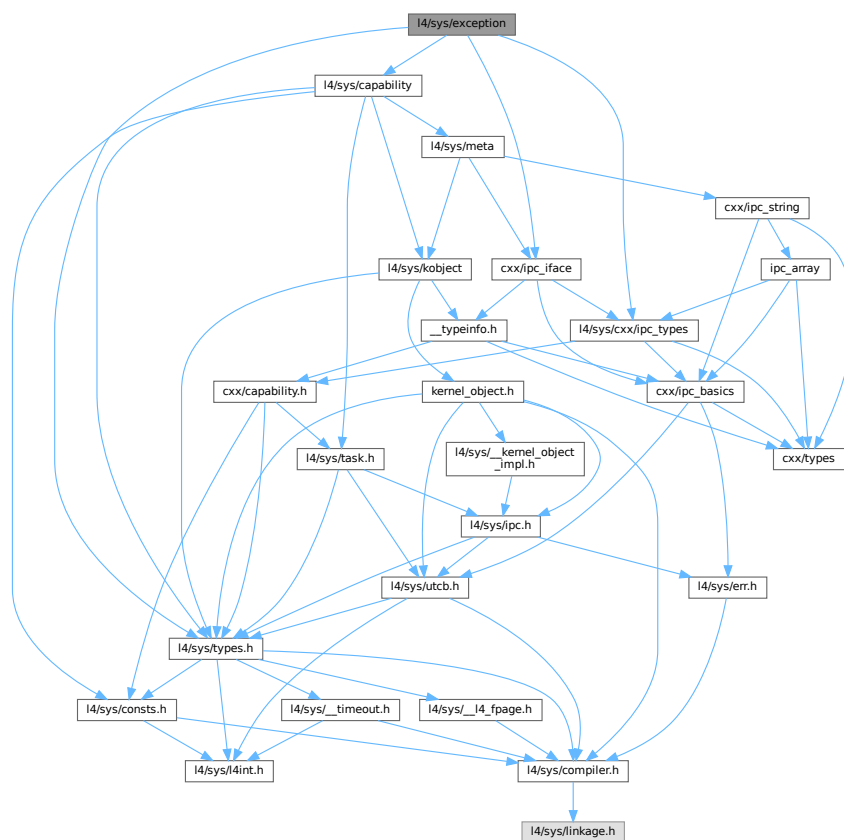
Exception C++ interface.

```

#include <l4/sys/capability>
#include <l4/sys/types.h>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_iface>

```

Include dependency graph for exception:



## Data Structures

- class [L4::Exception](#)  
*Exception interface.*

## Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*

### 17.511.1 Detailed Description

Exception C++ interface.

Definition in file [exception](#).

## 17.512 exception

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011
00012 #pragma once
00013
00014 #include <l4/sys/capability>
00015 #include <l4/sys/types.h>
00016 #include <l4/sys/cxx/ipc_types>
00017 #include <l4/sys/cxx/ipc_iface>
00018
00019 namespace L4 {
00020
00031 class L4_EXPORT Exception :
00032     public Kobject_0t<Exception, L4_PROTO_EXCEPTION>
00033 {
00034 public:
00035     // TODO: pass a reference/pointer to the UTCB not copy the regs
00046     L4_INLINE_RPC(
00047         l4_msgtag_t, exception, (L4::Ipc::In_out<l4_exc_regs_t *> regs,
00048                                 L4::Ipc::Rcv_fpage rwin,
00049                                 L4::Ipc::Opt<L4::Ipc::Snd_fpage &> fp));
00050
00051     typedef L4::Typeid::Rpc_nocode<exception_t> Rpcs;
00052 };
00053
00054 }

```

## 17.513 l4/sys/factory File Reference

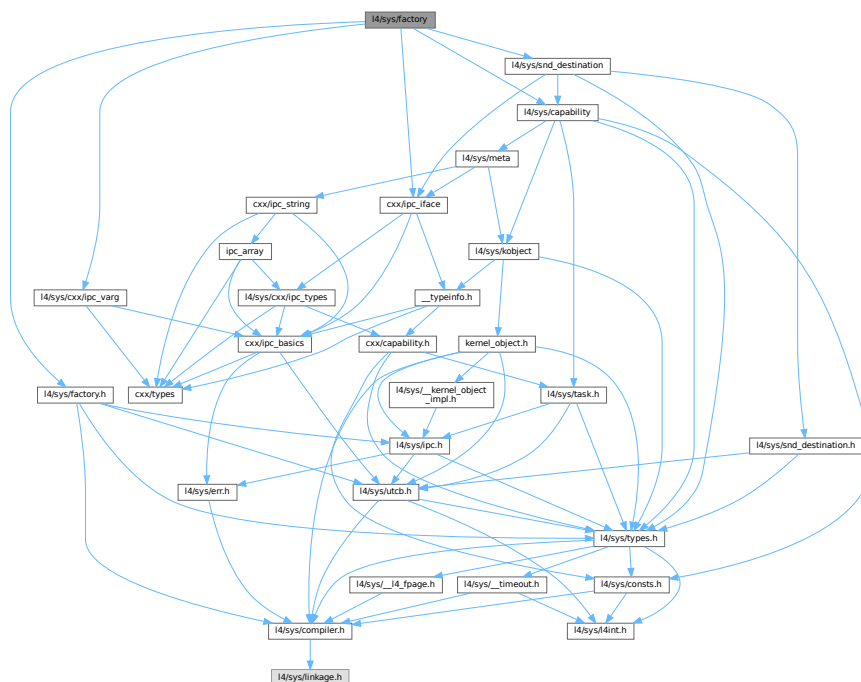
Common factory related definitions.

```

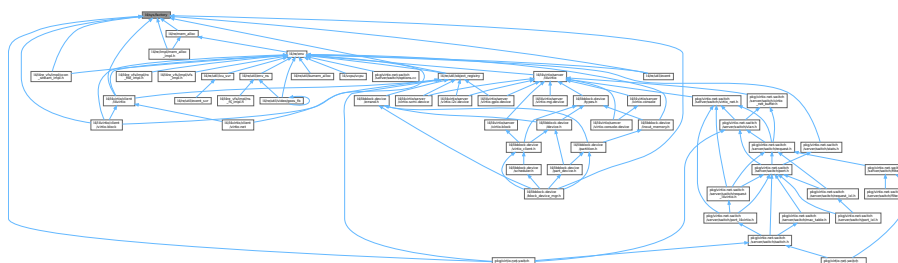
#include <l4/sys/factory.h>
#include <l4/sys/capability>
#include <l4/sys/snd_destination>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/cxx/ipc_varg>

```

Include dependency graph for factory:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `L4::Factory`  
*C++ Factory interface, see `Factory` for the C interface.*
- struct `L4::Factory::Nil`  
*Special type to add a void argument into the factory create stream.*
- struct `L4::Factory::Lstr`  
*Special type to add a pascal string into the factory create stream.*
- class `L4::Factory::S`  
*Stream class for the `create()` argument stream.*

## Namespaces

- namespace **L4**  
*L4 low-level kernel interface.*

## 17.513.1 Detailed Description

Common factory related definitions.

Definition in file [factory](#).

## 17.514 factory

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013
00014 #pragma once
00015
00016 #include <l4/sys/factory.h>
00017 #include <l4/sys/capability>
00018 #include <l4/sys/snd_destination>
00019 #include <l4/sys/cxx/ipc_iface>
00020 #include <l4/sys/cxx/ipc_varg>
00021
00022 namespace L4 {
00023
00038 class Factory : public Kobject_t<Factory, Kobject, L4_PROTO_FACTORY>
00039 {
00040 public:
00041
00042     typedef l4_mword_t Proto;
00043
00047     struct Nil {};
00048
00054     struct Lstr
00055     {
00059         char const *s;
00060
00064         unsigned len;
00065
00070         Lstr(char const *s, unsigned len) noexcept : s(s), len(len) {}
00071     };
00072
00079     class S
00080     {
00081     private:
00082         l4_utcb_t *u;
00083         l4_msgtag_t t;
00084         l4_cap_idx_t f;
00085
00086         template<typename T>
00087         static T &&_move(T &c) { return static_cast<T &&>(c); }
00088
00089     public:
00090         S(S const &) = delete;
00091         S &operator = (S const &) &= delete;
00092
00098         S(S &&o) noexcept
00099         : u(o.u), t(o.t), f(o.f)
00100         { o.t.raw = 0; }
00101
00102         S &operator = (S &&o) & noexcept
00103         {
00104             u = o.u;
00105             t = o.t;
00106             f = o.f;
00107             o.t.raw = 0;
00108             return *this;
00109         }
00110
00125         S(l4_cap_idx_t f, long obj, L4::Cap<void> target,
00126           l4_utcb_t *utcb) noexcept
00127         : u(utcb), t(l4_factory_create_start_u(obj, target.cap(), u)), f(f)
00128         {}
00129
00139         ~S() noexcept

```

```

00140     {
00141         if (t.raw)
00142     l4_factory_create_commit_u(f, t, u);
00143     }
00144
00157     operator l4_msgtag_t () noexcept
00158     {
00159         l4_msgtag_t r = l4_factory_create_commit_u(f, t, u);
00160         t.raw = 0;
00161         return r;
00162     }
00163
00169     void put(l4_mword_t i) noexcept
00170     {
00171         l4_factory_create_add_int_u(i, &t, u);
00172     }
00173
00179     void put(l4_umword_t i) noexcept
00180     {
00181         l4_factory_create_add_uint_u(i, &t, u);
00182     }
00183
00191     void put(char const *s) & noexcept
00192     {
00193         l4_factory_create_add_str_u(s, &t, u);
00194     }
00195
00205     void put(Lstr const &s) & noexcept
00206     {
00207         l4_factory_create_add_lstr_u(s.s, s.len, &t, u);
00208     }
00209
00213     void put(Nil) & noexcept
00214     {
00215         l4_factory_create_add_nil_u(&t, u);
00216     }
00217
00223     void put(l4_fpage_t d) & noexcept
00224     {
00225         l4_factory_create_add_fpage_u(d, &t, u);
00226     }
00227
00236     template<typename T>
00237     S &operator « (T const &d) & noexcept
00238     {
00239         put(d);
00240         return *this;
00241     }
00242
00251     template<typename T>
00252     S &&operator « (T const &d) && noexcept
00253     {
00254         put(d);
00255         return _move(*this);
00256     }
00257 };
00258
00259 public:
00260
00292     S create(Cap<void> target, long obj, l4_utcb_t *utcb = l4_utcb()) noexcept
00293     {
00294         return S(cap(), obj, target, utcb);
00295     }
00296
00328     template<typename OBJ>
00329     S create(Cap<OBJ> target, l4_utcb_t *utcb = l4_utcb()) noexcept
00330     {
00331         return S(cap(), OBJ::Protocol, target, utcb);
00332     }
00333
00334     L4_INLINE_RPC_NF(
00335         l4_msgtag_t, create, (L4::Ipc::Out<L4::Cap<void> > target, l4_mword_t obj,
00336                             L4::Ipc::Varg const *args),
00337         L4::Ipc::Call_t<L4_CAP_FPAGE_S>);
00338
00370     l4_msgtag_t create_task(Cap<Task> const & target_cap,
00371                             l4_fpage_t *utcb_area,
00372                             l4_utcb_t *utcb = l4_utcb()) noexcept
00373     { return l4_factory_create_task_u(cap(), target_cap.cap(), utcb_area, utcb); }
00374
00404     l4_msgtag_t create_factory(Cap<Factory> const &target_cap,
00405                                unsigned long limit,
00406                                l4_utcb_t *utcb = l4_utcb()) noexcept
00407     { return l4_factory_create_factory_u(cap(), target_cap.cap(), limit, utcb); }
00408
00440     l4_msgtag_t create_gate(Cap<void> const &target_cap,

```

```

00441         Cap<Snd_destination> const &snd_dst_cap,
00442         l4_umword_t label,
00443         l4_utcb_t *utcb = l4_utcb()) noexcept
00444     {
00445         return l4_factory_create_gate_u(cap(), target_cap.cap(), snd_dst_cap.cap(),
00446                                         label, utcb);
00447     }
00448
00475 l4_msgtag_t create_thread_group(Cap<Thread_group> const &target_cap,
00476                                unsigned policy,
00477                                l4_utcb_t *utcb = l4_utcb()) noexcept
00478     {
00479         return l4_factory_create_thread_group_u(cap(), target_cap.cap(),
00480                                                 policy, utcb);
00481     }
00482
00483 typedef L4::Typeid::Rpc_nocode<create_t> Rpccs;
00484 };
00485
00486 }

```

## 17.515 l4/sys/factory.h File Reference

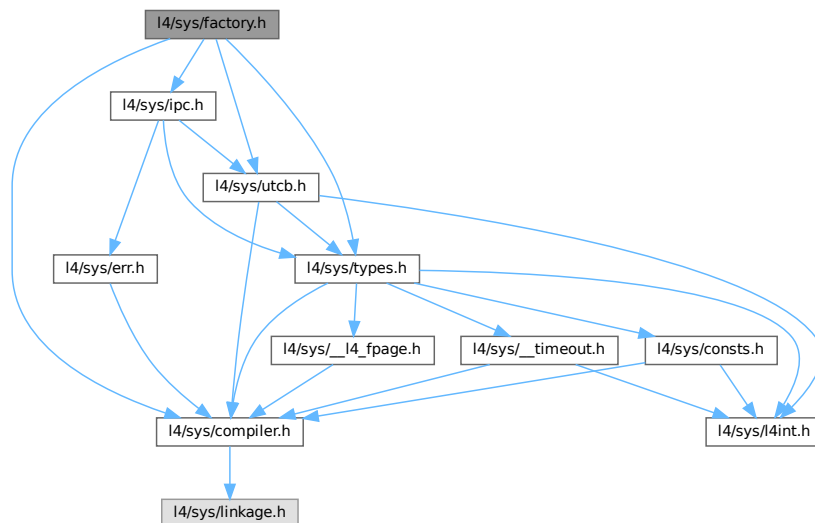
Common factory related definitions.

```

#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>

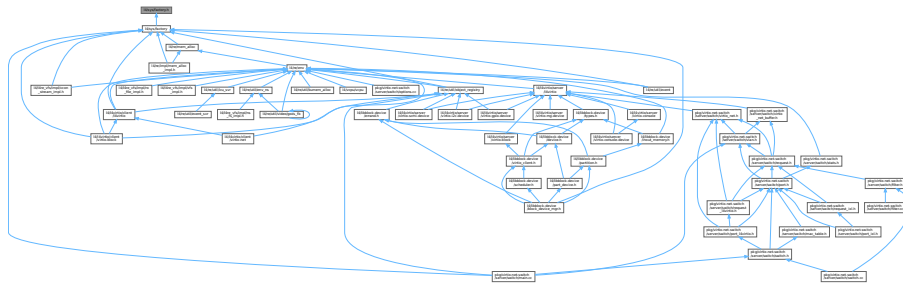
```

Include dependency graph for factory.h:





This graph shows which files directly or indirectly include this file:



## Functions

- [l4\\_msgtag\\_t l4\\_factory\\_create\\_task](#) ([l4\\_cap\\_idx\\_t](#) factory, [l4\\_cap\\_idx\\_t](#) target\_cap, [l4\\_fpage\\_t](#) \*utcb\_area) [L4\\_NOTHROW](#)  
Create a new task.
- [l4\\_msgtag\\_t l4\\_factory\\_create\\_thread](#) ([l4\\_cap\\_idx\\_t](#) factory, [l4\\_cap\\_idx\\_t](#) target\_cap) [L4\\_NOTHROW](#)  
Create a new thread.
- [l4\\_msgtag\\_t l4\\_factory\\_create\\_factory](#) ([l4\\_cap\\_idx\\_t](#) factory, [l4\\_cap\\_idx\\_t](#) target\_cap, unsigned long limit) [L4\\_NOTHROW](#)  
Create a new factory.
- [l4\\_msgtag\\_t l4\\_factory\\_create\\_gate](#) ([l4\\_cap\\_idx\\_t](#) factory, [l4\\_cap\\_idx\\_t](#) target\_cap, [l4\\_cap\\_idx\\_t](#) snd\_dst\_↔ cap, [l4\\_umword\\_t](#) label) [L4\\_NOTHROW](#)  
Create a new IPC gate, optionally bound to a send destination (a thread or thread group).
- [l4\\_msgtag\\_t l4\\_factory\\_create\\_irq](#) ([l4\\_cap\\_idx\\_t](#) factory, [l4\\_cap\\_idx\\_t](#) target\_cap) [L4\\_NOTHROW](#)  
Create a new IRQ sender.
- [l4\\_msgtag\\_t l4\\_factory\\_create\\_vm](#) ([l4\\_cap\\_idx\\_t](#) factory, [l4\\_cap\\_idx\\_t](#) target\_cap) [L4\\_NOTHROW](#)  
Create a new virtual machine.
- [l4\\_msgtag\\_t l4\\_factory\\_create\\_vcpu\\_context](#) ([l4\\_cap\\_idx\\_t](#) factory, [l4\\_cap\\_idx\\_t](#) target\_cap) [L4\\_NOTHROW](#)  
Create a new vCPU context.
- [l4\\_msgtag\\_t l4\\_factory\\_create\\_thread\\_group](#) ([l4\\_cap\\_idx\\_t](#) factory, [l4\\_cap\\_idx\\_t](#) target\_cap, unsigned policy) [L4\\_NOTHROW](#)  
Create a new thread group.
- [l4\\_msgtag\\_t l4\\_factory\\_create](#) ([l4\\_cap\\_idx\\_t](#) factory, long obj, [l4\\_cap\\_idx\\_t](#) target) [L4\\_NOTHROW](#)  
Create a new object.

## 17.515.1 Detailed Description

Common factory related definitions.

Definition in file [factory.h](#).

## 17.516 factory.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Björn Döbel <doebel@os.inf.tu-dresden.de>,
00010  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>,
00011  *      Henning Schild <hschild@os.inf.tu-dresden.de>
00012  *      economic rights: Technische Universität Dresden (Germany)
00013  *
00014  * License: see LICENSE.spdx (in this directory or the directories above)
00015  */
00016 #pragma once
00017
00018 #include <l4/sys/compiler.h>
00019 #include <l4/sys/types.h>
00020 #include <l4/sys/utcb.h>
00021
00050
00056
00086 L4_INLINE l4_msgtag_t
00087 l4_factory_create_task(l4_cap_idx_t factory, l4_cap_idx_t target_cap,
00088                      l4_fpage_t *utcb_area) L4_NOTHROW;
00089
00094 L4_INLINE l4_msgtag_t
00095 l4_factory_create_task_u(l4_cap_idx_t factory, l4_cap_idx_t target_cap,
00096                       l4_fpage_t *utcb_area, l4_utcb_t *utcb) L4_NOTHROW;
00097
00116 L4_INLINE l4_msgtag_t
00117 l4_factory_create_thread(l4_cap_idx_t factory,
00118                        l4_cap_idx_t target_cap) L4_NOTHROW;
00119
00124 L4_INLINE l4_msgtag_t
00125 l4_factory_create_thread_u(l4_cap_idx_t factory,
00126                          l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW;
00127
00153 L4_INLINE l4_msgtag_t
00154 l4_factory_create_factory(l4_cap_idx_t factory, l4_cap_idx_t target_cap,
00155                          unsigned long limit) L4_NOTHROW;
00156
00161 L4_INLINE l4_msgtag_t
00162 l4_factory_create_factory_u(l4_cap_idx_t factory, l4_cap_idx_t target_cap,
00163                           unsigned long limit, l4_utcb_t *utcb) L4_NOTHROW;
00164
00194 L4_INLINE l4_msgtag_t
00195 l4_factory_create_gate(l4_cap_idx_t factory,
00196                      l4_cap_idx_t target_cap,
00197                      l4_cap_idx_t snd_dst_cap, l4_umword_t label) L4_NOTHROW;
00198
00203 L4_INLINE l4_msgtag_t
00204 l4_factory_create_gate_u(l4_cap_idx_t factory,
00205                        l4_cap_idx_t target_cap,
00206                        l4_cap_idx_t snd_dst_cap, l4_umword_t label,
00207                        l4_utcb_t *utcb) L4_NOTHROW;
00208
00225 L4_INLINE l4_msgtag_t
00226 l4_factory_create_irq(l4_cap_idx_t factory,
00227                     l4_cap_idx_t target_cap) L4_NOTHROW;
00228
00233 L4_INLINE l4_msgtag_t
00234 l4_factory_create_irq_u(l4_cap_idx_t factory,
00235                       l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW;
00236
00255 L4_INLINE l4_msgtag_t
00256 l4_factory_create_vm(l4_cap_idx_t factory,
00257                    l4_cap_idx_t target_cap) L4_NOTHROW;
00258
00278 L4_INLINE l4_msgtag_t
00279 l4_factory_create_vcpu_context(l4_cap_idx_t factory,
00280                              l4_cap_idx_t target_cap) L4_NOTHROW;
00281
00309 L4_INLINE l4_msgtag_t
00310 l4_factory_create_thread_group(l4_cap_idx_t factory,
00311                              l4_cap_idx_t target_cap,
00312                              unsigned policy) L4_NOTHROW;
00313
00318 L4_INLINE l4_msgtag_t
00319 l4_factory_create_vm_u(l4_cap_idx_t factory,
00320                      l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW;
00321
00326 L4_INLINE l4_msgtag_t
00327 l4_factory_create_vcpu_context_u(l4_cap_idx_t factory,
00328                                 l4_cap_idx_t target_cap,

```

```

00329             l4_utcb_t *utcb) L4_NOTHROW;
00330
00335 L4_INLINE l4_msgtag_t
00336 l4_factory_create_thread_group_u(l4_cap_idx_t factory,
00337             l4_cap_idx_t target_cap,
00338             unsigned policy,
00339             l4_utcb_t *u) L4_NOTHROW;
00340
00345 L4_INLINE l4_msgtag_t
00346 l4_factory_create_start_u(long obj, l4_cap_idx_t target,
00347             l4_utcb_t *utcb) L4_NOTHROW;
00348
00353 L4_INLINE int
00354 l4_factory_create_add_fpage_u(l4_fpage_t d, l4_msgtag_t *tag,
00355             l4_utcb_t *utcb) L4_NOTHROW;
00356
00361 L4_INLINE int
00362 l4_factory_create_add_int_u(l4_mword_t d, l4_msgtag_t *tag,
00363             l4_utcb_t *utcb) L4_NOTHROW;
00364
00369 L4_INLINE int
00370 l4_factory_create_add_uint_u(l4_umword_t d, l4_msgtag_t *tag,
00371             l4_utcb_t *utcb) L4_NOTHROW;
00372
00377 L4_INLINE int
00378 l4_factory_create_add_str_u(char const *s, l4_msgtag_t *tag,
00379             l4_utcb_t *utcb) L4_NOTHROW;
00380
00385 L4_INLINE int
00386 l4_factory_create_add_lstr_u(char const *s, unsigned len, l4_msgtag_t *tag,
00387             l4_utcb_t *utcb) L4_NOTHROW;
00388
00393 L4_INLINE int
00394 l4_factory_create_add_nil_u(l4_msgtag_t *tag, l4_utcb_t *utcb) L4_NOTHROW;
00395
00400 L4_INLINE l4_msgtag_t
00401 l4_factory_create_commit_u(l4_cap_idx_t factory, l4_msgtag_t tag,
00402             l4_utcb_t *utcb) L4_NOTHROW;
00403
00408 L4_INLINE l4_msgtag_t
00409 l4_factory_create_u(l4_cap_idx_t factory, long obj, l4_cap_idx_t target,
00410             l4_utcb_t *utcb) L4_NOTHROW;
00411
00412
00430 L4_INLINE l4_msgtag_t
00431 l4_factory_create(l4_cap_idx_t factory, long obj,
00432             l4_cap_idx_t target) L4_NOTHROW;
00433
00434 /* IMPLEMENTATION -----*/
00435
00436 #include <l4/sys/ipc.h>
00437
00438 L4_INLINE l4_msgtag_t
00439 l4_factory_create_task_u(l4_cap_idx_t factory,
00440             l4_cap_idx_t target_cap, l4_fpage_t *utcb_area,
00441             l4_utcb_t *u) L4_NOTHROW
00442 {
00443     l4_msgtag_t t;
00444     t = l4_factory_create_start_u(L4_PROTO_TASK, target_cap, u);
00445     l4_factory_create_add_fpage_u(*utcb_area, &t, u);
00446     t = l4_factory_create_commit_u(factory, t, u);
00447     if (!l4_msgtag_has_error(t))
00448     {
00449         l4_msg_regs_t *v = l4_utcb_mr_u(u);
00450         utcb_area->raw = v->mr[0];
00451     }
00452     return t;
00453 }
00454
00455 L4_INLINE l4_msgtag_t
00456 l4_factory_create_thread_u(l4_cap_idx_t factory,
00457             l4_cap_idx_t target_cap, l4_utcb_t *u) L4_NOTHROW
00458 {
00459     return l4_factory_create_u(factory, L4_PROTO_THREAD, target_cap, u);
00460 }
00461
00462 L4_INLINE l4_msgtag_t
00463 l4_factory_create_factory_u(l4_cap_idx_t factory,
00464             l4_cap_idx_t target_cap, unsigned long limit,
00465             l4_utcb_t *u) L4_NOTHROW
00466 {
00467     l4_msgtag_t t;
00468     t = l4_factory_create_start_u(L4_PROTO_FACTORY, target_cap, u);
00469     l4_factory_create_add_uint_u(limit, &t, u);
00470     return l4_factory_create_commit_u(factory, t, u);
00471 }
00472

```

```

00473 L4_INLINE l4_msgtag_t
00474 l4_factory_create_gate_u(l4_cap_idx_t factory,
00475                          l4_cap_idx_t target_cap,
00476                          l4_cap_idx_t snd_dst_cap, l4_umword_t label,
00477                          l4_utcb_t *u) L4_NOTHROW
00478 {
00479     l4_msgtag_t t;
00480     l4_msg_regs_t *v;
00481     int items = 0;
00482     t = l4_factory_create_start_u(0, target_cap, u);
00483     l4_factory_create_add_uint_u(label, &t, u);
00484     v = l4_utcb_mr_u(u);
00485     if (!(snd_dst_cap & L4_INVALID_CAP_BIT))
00486     {
00487         items = 1;
00488         v->mr[3] = l4_map_obj_control(0,0);
00489         v->mr[4] = l4_obj_fpage(snd_dst_cap, 0, L4_CAP_FPAGE_RWS).raw;
00490     }
00491     t = l4_msgtag(l4_msgtag_label(t), l4_msgtag_words(t), items, l4_msgtag_flags(t));
00492     return l4_factory_create_commit_u(factory, t, u);
00493 }
00494
00495 L4_INLINE l4_msgtag_t
00496 l4_factory_create_irq_u(l4_cap_idx_t factory,
00497                        l4_cap_idx_t target_cap, l4_utcb_t *u) L4_NOTHROW
00498 {
00499     return l4_factory_create_u(factory, L4_PROTO_IRQ_SENDER, target_cap, u);
00500 }
00501
00502 L4_INLINE l4_msgtag_t
00503 l4_factory_create_vm_u(l4_cap_idx_t factory,
00504                       l4_cap_idx_t target_cap,
00505                       l4_utcb_t *u) L4_NOTHROW
00506 {
00507     return l4_factory_create_u(factory, L4_PROTO_VM, target_cap, u);
00508 }
00509
00510 L4_INLINE l4_msgtag_t
00511 l4_factory_create_vcpu_context_u(l4_cap_idx_t factory,
00512                                 l4_cap_idx_t target_cap,
00513                                 l4_utcb_t *u) L4_NOTHROW
00514 {
00515     return l4_factory_create_u(factory, L4_PROTO_VCPU_CONTEXT, target_cap, u);
00516 }
00517
00518 L4_INLINE l4_msgtag_t
00519 l4_factory_create_thread_group_u(l4_cap_idx_t factory,
00520                                 l4_cap_idx_t target_cap,
00521                                 unsigned policy,
00522                                 l4_utcb_t *u) L4_NOTHROW
00523 {
00524     l4_msgtag_t t = l4_factory_create_start_u(L4_PROTO_THREAD_GROUP, target_cap, u);
00525     l4_factory_create_add_uint_u(policy, &t, u);
00526     return l4_factory_create_commit_u(factory, t, u);
00527 }
00528
00529
00530 L4_INLINE l4_msgtag_t
00531 l4_factory_create_task(l4_cap_idx_t factory,
00532                       l4_cap_idx_t target_cap, l4_fpage_t *utcb_area) L4_NOTHROW
00533 {
00534     return l4_factory_create_task_u(factory, target_cap, utcb_area, l4_utcb());
00535 }
00536
00537 L4_INLINE l4_msgtag_t
00538 l4_factory_create_thread(l4_cap_idx_t factory,
00539                          l4_cap_idx_t target_cap) L4_NOTHROW
00540 {
00541     return l4_factory_create_thread_u(factory, target_cap, l4_utcb());
00542 }
00543
00544 L4_INLINE l4_msgtag_t
00545 l4_factory_create_factory(l4_cap_idx_t factory,
00546                          l4_cap_idx_t target_cap, unsigned long limit) L4_NOTHROW
00547 {
00548     return l4_factory_create_factory_u(factory, target_cap, limit, l4_utcb());
00549 }
00550
00551
00552 L4_INLINE l4_msgtag_t
00553 l4_factory_create_gate(l4_cap_idx_t factory,
00554                       l4_cap_idx_t target_cap,
00555                       l4_cap_idx_t snd_dst_cap, l4_umword_t label) L4_NOTHROW
00556 {
00557     return l4_factory_create_gate_u(factory, target_cap, snd_dst_cap, label, l4_utcb());
00558 }
00559

```

```

00560 L4_INLINE l4_msgtag_t
00561 l4_factory_create_irq(l4_cap_idx_t factory,
00562                       l4_cap_idx_t target_cap) L4_NOTHROW
00563 {
00564     return l4_factory_create_irq_u(factory, target_cap, l4_utcb());
00565 }
00566
00567 L4_INLINE l4_msgtag_t
00568 l4_factory_create_vm(l4_cap_idx_t factory,
00569                     l4_cap_idx_t target_cap) L4_NOTHROW
00570 {
00571     return l4_factory_create_vm_u(factory, target_cap, l4_utcb());
00572 }
00573
00574 L4_INLINE l4_msgtag_t
00575 l4_factory_create_vcpu_context(l4_cap_idx_t factory,
00576                               l4_cap_idx_t target_cap) L4_NOTHROW
00577 {
00578     return l4_factory_create_vcpu_context_u(factory, target_cap, l4_utcb());
00579 }
00580
00581 L4_INLINE l4_msgtag_t
00582 l4_factory_create_thread_group(l4_cap_idx_t factory,
00583                               l4_cap_idx_t target_cap,
00584                               unsigned policy) L4_NOTHROW
00585 {
00586     return l4_factory_create_thread_group_u(factory, target_cap, policy, l4_utcb());
00587 }
00588
00589
00590 L4_INLINE l4_msgtag_t
00591 l4_factory_create_start_u(long obj, l4_cap_idx_t target_cap,
00592                          l4_utcb_t *u) L4_NOTHROW
00593 {
00594     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00595     l4_buf_regs_t *b = l4_utcb_br_u(u);
00596     v->mr[0] = obj;
00597     b->bdr = 0;
00598     b->br[0] = target_cap | L4_RCV_ITEM_SINGLE_CAP;
00599     return l4_msgtag(L4_PROTO_FACTORY, 1, 0, 0);
00600 }
00601
00602 L4_INLINE int
00603 l4_factory_create_add_fpage_u(l4_fpage_t d, l4_msgtag_t *tag,
00604                              l4_utcb_t *u) L4_NOTHROW
00605 {
00606     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00607     int w = l4_msgtag_words(*tag);
00608     if (w + 2 > L4_UTCB_GENERIC_DATA_SIZE)
00609         return 0;
00610     v->mr[w] = L4_VARG_TYPE_FPAGE | (sizeof(l4_fpage_t) << 16);
00611     v->mr[w + 1] = d.raw;
00612     w += 2;
00613     tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00614     return 1;
00615 }
00616
00617 L4_INLINE int
00618 l4_factory_create_add_int_u(l4_mword_t d, l4_msgtag_t *tag,
00619                           l4_utcb_t *u) L4_NOTHROW
00620 {
00621     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00622     int w = l4_msgtag_words(*tag);
00623     if (w + 2 > L4_UTCB_GENERIC_DATA_SIZE)
00624         return 0;
00625     v->mr[w] = L4_VARG_TYPE_MWORD | (sizeof(l4_mword_t) << 16);
00626     v->mr[w + 1] = d;
00627     w += 2;
00628     tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00629     return 1;
00630 }
00631
00632 L4_INLINE int
00633 l4_factory_create_add_uint_u(l4_umword_t d, l4_msgtag_t *tag,
00634                             l4_utcb_t *u) L4_NOTHROW
00635 {
00636     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00637     int w = l4_msgtag_words(*tag);
00638     if (w + 2 > L4_UTCB_GENERIC_DATA_SIZE)
00639         return 0;
00640     v->mr[w] = L4_VARG_TYPE_UMWORD | (sizeof(l4_umword_t) << 16);
00641     v->mr[w + 1] = d;
00642     w += 2;
00643     tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00644     return 1;
00645 }
00646

```

```

00647 L4_INLINE int
00648 l4_factory_create_add_str_u(char const *s, l4_msgtag_t *tag,
00649                             l4_utcb_t *u) L4_NOTHROW
00650 {
00651     return l4_factory_create_add_lstr_u(s, __builtin_strlen(s) + 1, tag, u);
00652 }
00653
00654 L4_INLINE int
00655 l4_factory_create_add_lstr_u(char const *s, unsigned len, l4_msgtag_t *tag,
00656                             l4_utcb_t *u) L4_NOTHROW
00657 {
00658
00659     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00660     unsigned w = l4_msgtag_words(*tag);
00661     char *c;
00662     unsigned i;
00663
00664     if (w + 1 + l4_bytes_to_mwords(len) > L4_UTCB_GENERIC_DATA_SIZE)
00665         return 0;
00666
00667     v->mr[w] = L4_VARG_TYPE_STRING | (len << 16);
00668     c = (char*)&v->mr[w + 1];
00669     for (i = 0; i < len; ++i)
00670         *c++ = *s++;
00671
00672     w = w + 1 + l4_bytes_to_mwords(len);
00673
00674     tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00675     return 1;
00676 }
00677
00678 L4_INLINE int
00679 l4_factory_create_add_nil_u(l4_msgtag_t *tag, l4_utcb_t *utcb) L4_NOTHROW
00680 {
00681     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00682     int w = l4_msgtag_words(*tag);
00683     v->mr[w] = L4_VARG_TYPE_NIL;
00684     ++w;
00685     tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00686     return 1;
00687 }
00688
00689
00690 L4_INLINE l4_msgtag_t
00691 l4_factory_create_commit_u(l4_cap_idx_t factory, l4_msgtag_t tag,
00692                             l4_utcb_t *u) L4_NOTHROW
00693 {
00694     return l4_ipc_call(factory, u, tag, L4_IPC_NEVER);
00695 }
00696
00697 L4_INLINE l4_msgtag_t
00698 l4_factory_create_u(l4_cap_idx_t factory, long obj, l4_cap_idx_t target,
00699                     l4_utcb_t *utcb) L4_NOTHROW
00700 {
00701     l4_msgtag_t t = l4_factory_create_start_u(obj, target, utcb);
00702     return l4_factory_create_commit_u(factory, t, utcb);
00703 }
00704
00705
00706 L4_INLINE l4_msgtag_t
00707 l4_factory_create(l4_cap_idx_t factory, long obj,
00708                   l4_cap_idx_t target) L4_NOTHROW
00709 {
00710     return l4_factory_create_u(factory, obj, target, l4_utcb());
00711 }

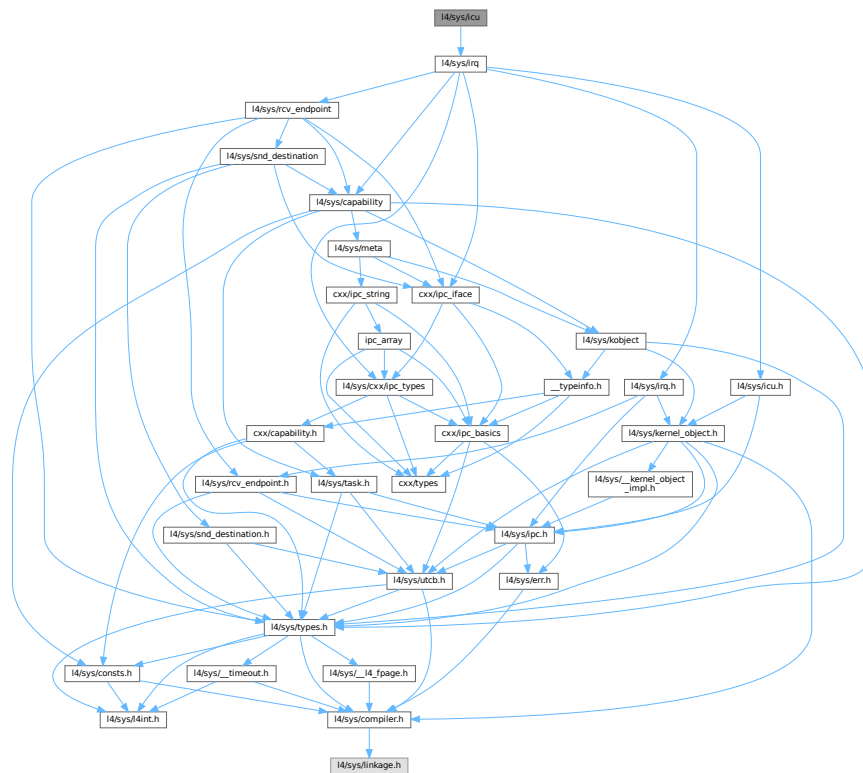
```

## 17.517 I4/sys/icu File Reference

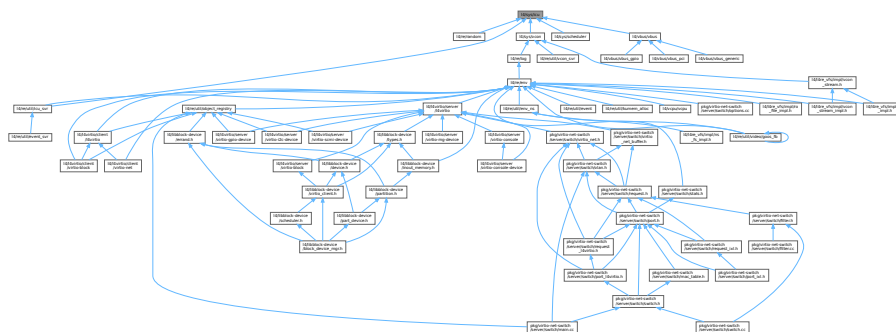
Interrupt controller.

```
#include <l4/sys/irq>
```

Include dependency graph for icu:



This graph shows which files directly or indirectly include this file:



## 17.517.1 Detailed Description

Interrupt controller.

Definition in file [icu](#).

## 17.518 icu

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00007 /*
00008  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00015 #include <l4/sys/irq>
```

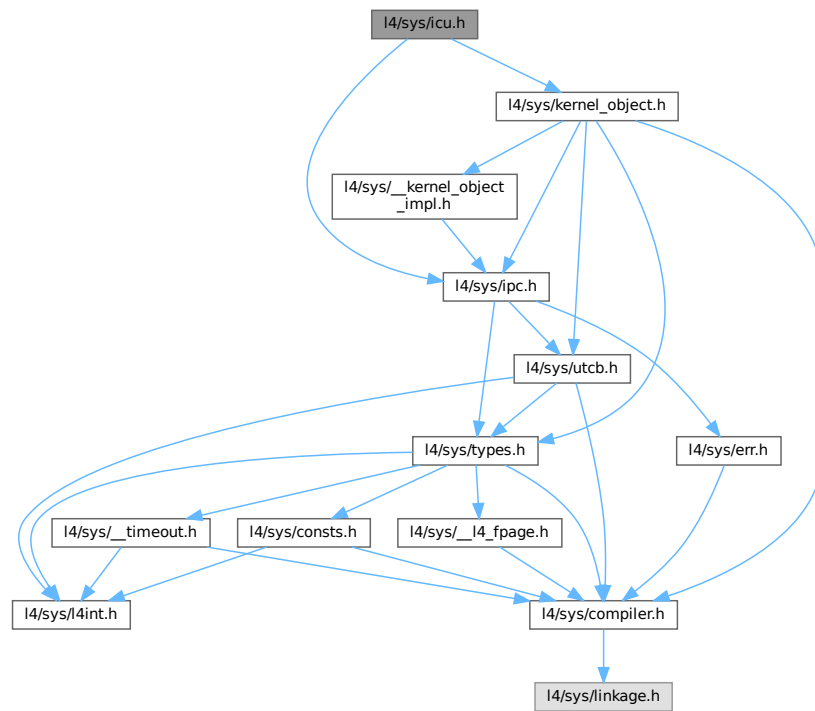
## 17.519 l4/sys/icu.h File Reference

Interrupt controller.

```
#include <l4/sys/kernel_object.h>
```

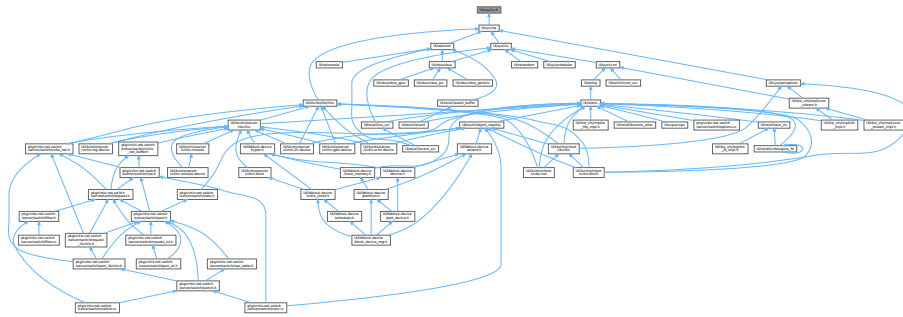
```
#include <l4/sys/ipc.h>
```

Include dependency graph for icu.h:





This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [l4\\_icu\\_info\\_t](#)  
*Info structure for an ICU.*
- struct [l4\\_icu\\_msi\\_info\\_t](#)  
*Info to use for a specific MSI.*

## Typedefs

- typedef struct [l4\\_icu\\_info\\_t](#) [l4\\_icu\\_info\\_t](#)  
*Info structure for an ICU.*
- typedef struct [l4\\_icu\\_msi\\_info\\_t](#) [l4\\_icu\\_msi\\_info\\_t](#)  
*Info to use for a specific MSI.*

## Enumerations

- enum [L4\\_icu\\_flags](#) { [L4\\_ICU\\_FLAG\\_MSI](#) }  
*Flags for IRQ numbers used for the ICU.*
- enum [L4\\_irq\\_mode](#) {  
[L4\\_IRQ\\_F\\_NONE](#) = 0 , [L4\\_IRQ\\_F\\_SET\\_MODE](#) = 0x1 , [L4\\_IRQ\\_F\\_LEVEL](#) = 0x2 , [L4\\_IRQ\\_F\\_EDGE](#) = 0x0 ,  
[L4\\_IRQ\\_F\\_POS](#) = 0x0 , [L4\\_IRQ\\_F\\_NEG](#) = 0x4 , [L4\\_IRQ\\_F\\_BOTH](#) = 0x8 , [L4\\_IRQ\\_F\\_LEVEL\\_HIGH](#) =  
[L4\\_IRQ\\_F\\_SET\\_MODE](#) | [L4\\_IRQ\\_F\\_LEVEL](#) | [L4\\_IRQ\\_F\\_POS](#) ,  
[L4\\_IRQ\\_F\\_LEVEL\\_LOW](#) = [L4\\_IRQ\\_F\\_SET\\_MODE](#) | [L4\\_IRQ\\_F\\_LEVEL](#) | [L4\\_IRQ\\_F\\_NEG](#) , [L4\\_IRQ\\_F\\_POS\\_EDGE](#)  
= [L4\\_IRQ\\_F\\_SET\\_MODE](#) | [L4\\_IRQ\\_F\\_EDGE](#) | [L4\\_IRQ\\_F\\_POS](#) , [L4\\_IRQ\\_F\\_NEG\\_EDGE](#) = [L4\\_IRQ\\_F\\_](#)  
[SET\\_MODE](#) | [L4\\_IRQ\\_F\\_EDGE](#) | [L4\\_IRQ\\_F\\_NEG](#) , [L4\\_IRQ\\_F\\_BOTH\\_EDGE](#) = [L4\\_IRQ\\_F\\_SET\\_MODE](#) |  
[L4\\_IRQ\\_F\\_EDGE](#) | [L4\\_IRQ\\_F\\_BOTH](#) ,  
[L4\\_IRQ\\_F\\_MASK](#) = 0xf , [L4\\_IRQ\\_F\\_SET\\_WAKEUP](#) = 0x10 , [L4\\_IRQ\\_F\\_CLEAR\\_WAKEUP](#) = 0x20 }  
*Interrupt attributes.*
- enum [L4\\_icu\\_opcode](#) {  
[L4\\_ICU\\_OP\\_BIND](#) , [L4\\_ICU\\_OP\\_UNBIND](#) , [L4\\_ICU\\_OP\\_INFO](#) , [L4\\_ICU\\_OP\\_MSI\\_INFO](#) ,  
[L4\\_ICU\\_OP\\_UNMASK](#) , [L4\\_ICU\\_OP\\_MASK](#) , [L4\\_ICU\\_OP\\_SET\\_MODE](#) }  
*Opcodes to the ICU interface.*

## Functions

- [l4\\_msgtag\\_t l4\\_icu\\_bind](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_cap\\_idx\\_t](#) irq) [L4\\_NOTHROW](#)  
*Bind an interrupt line of an interrupt controller to an interrupt object.*
- [l4\\_msgtag\\_t l4\\_icu\\_bind\\_u](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_cap\\_idx\\_t](#) irq, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Bind an interrupt line of an interrupt controller to an interrupt object.*
- [l4\\_msgtag\\_t l4\\_icu\\_unbind](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_cap\\_idx\\_t](#) irq) [L4\\_NOTHROW](#)  
*Remove binding of an interrupt line from the interrupt controller object.*
- [l4\\_msgtag\\_t l4\\_icu\\_unbind\\_u](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_cap\\_idx\\_t](#) irq, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Remove binding of an interrupt line from the interrupt controller object.*
- [l4\\_msgtag\\_t l4\\_icu\\_set\\_mode](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_umword\\_t](#) mode) [L4\\_NOTHROW](#)  
*Set interrupt mode.*
- [l4\\_msgtag\\_t l4\\_icu\\_set\\_mode\\_u](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_umword\\_t](#) mode, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Set interrupt mode.*
- [l4\\_msgtag\\_t l4\\_icu\\_info](#) ([l4\\_cap\\_idx\\_t](#) icu, [l4\\_icu\\_info\\_t](#) \*info) [L4\\_NOTHROW](#)  
*Get information about the ICU features.*
- [l4\\_msgtag\\_t l4\\_icu\\_info\\_u](#) ([l4\\_cap\\_idx\\_t](#) icu, [l4\\_icu\\_info\\_t](#) \*info, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Get information about the ICU features.*
- [l4\\_msgtag\\_t l4\\_icu\\_msi\\_info](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_uint64\\_t](#) source, [l4\\_icu\\_msi\\_info\\_t](#) \*msi\_info) [L4\\_NOTHROW](#)  
*Get MSI info about IRQ.*
- [l4\\_msgtag\\_t l4\\_icu\\_msi\\_info\\_u](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_uint64\\_t](#) source, [l4\\_icu\\_msi\\_info\\_t](#) \*msi\_info, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Get MSI info about IRQ.*
- [l4\\_msgtag\\_t l4\\_icu\\_unmask](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_umword\\_t](#) \*label, [l4\\_timeout\\_t](#) to) [L4\\_NOTHROW](#)  
*Unmask an IRQ line.*
- [l4\\_msgtag\\_t l4\\_icu\\_unmask\\_u](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_umword\\_t](#) \*label, [l4\\_timeout\\_t](#) to, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Unmask the given interrupt line.*
- [l4\\_msgtag\\_t l4\\_icu\\_mask](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_umword\\_t](#) \*label, [l4\\_timeout\\_t](#) to) [L4\\_NOTHROW](#)  
*Mask an IRQ line.*
- [l4\\_msgtag\\_t l4\\_icu\\_mask\\_u](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_umword\\_t](#) \*label, [l4\\_timeout\\_t](#) to, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Mask an IRQ line.*

### 17.519.1 Detailed Description

Interrupt controller.

Definition in file [icu.h](#).

## 17.520 icu.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #pragma once
00015
00016 #include <l4/sys/kernel_object.h>
00017 #include <l4/sys/ipc.h>
00018
00047
00048
00053 enum L4_icu_flags
00054 {
00062     L4_ICU_FLAG_MSI = 0x80000000,
00063 };
00064
00065
00070 enum L4_irq_mode
00071 {
00073     L4_IRQ_F_NONE           = 0,
00074     L4_IRQ_F_SET_MODE      = 0x1,
00075     L4_IRQ_F_LEVEL         = 0x2,
00076     L4_IRQ_F_EDGE         = 0x0,
00077     L4_IRQ_F_POS           = 0x0,
00078     L4_IRQ_F_NEG           = 0x4,
00079     L4_IRQ_F_BOTH          = 0x8,
00080     L4_IRQ_F_LEVEL_HIGH   = L4_IRQ_F_SET_MODE | L4_IRQ_F_LEVEL | L4_IRQ_F_POS,
00081     L4_IRQ_F_LEVEL_LOW    = L4_IRQ_F_SET_MODE | L4_IRQ_F_LEVEL | L4_IRQ_F_NEG,
00082     L4_IRQ_F_POS_EDGE     = L4_IRQ_F_SET_MODE | L4_IRQ_F_EDGE | L4_IRQ_F_POS,
00083     L4_IRQ_F_NEG_EDGE     = L4_IRQ_F_SET_MODE | L4_IRQ_F_EDGE | L4_IRQ_F_NEG,
00084     L4_IRQ_F_BOTH_EDGE    = L4_IRQ_F_SET_MODE | L4_IRQ_F_EDGE | L4_IRQ_F_BOTH,
00085     L4_IRQ_F_MASK         = 0xf,
00086
00088     L4_IRQ_F_SET_WAKEUP    = 0x10,
00089     L4_IRQ_F_CLEAR_WAKEUP = 0x20,
00090 };
00091
00092
00097 enum L4_icu_opcode
00098 {
00104     L4_ICU_OP_BIND = 0,
00105
00111     L4_ICU_OP_UNBIND = 1,
00112
00118     L4_ICU_OP_INFO = 2,
00119
00125     L4_ICU_OP_MSI_INFO = 3,
00126
00132     L4_ICU_OP_UNMASK = 4,
00133
00139     L4_ICU_OP_MASK = 5,
00140
00146     L4_ICU_OP_SET_MODE = 6,
00147 };
00148
00149 enum L4_icu_ctl_op
00150 {
00151     L4_ICU_CTL_UNMASK = 0,
00152     L4_ICU_CTL_MASK = 1
00153 };
00154
00155
00163 typedef struct l4_icu_info_t
00164 {
00170     unsigned features;
00171
00175     unsigned nr_irqs;
00176
00180     unsigned nr_msis;
00181 } l4_icu_info_t;
00182
00184 typedef struct l4_icu_msi_info_t
00185 {
00187     l4_uint64_t msi_addr;
00189     l4_uint32_t msi_data;
00190 } l4_icu_msi_info_t;
00191

```

```

00219 L4_INLINE l4_msgtag_t
00220 l4_icu_bind(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq) L4_NOTHROW;
00221
00222 L4_INLINE l4_msgtag_t
00223 l4_icu_bind_u(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq,
00230 l4_utcb_t *utcb) L4_NOTHROW;
00231
00242 L4_INLINE l4_msgtag_t
00243 l4_icu_unbind(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq) L4_NOTHROW;
00244
00251 L4_INLINE l4_msgtag_t
00252 l4_icu_unbind_u(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq,
00253 l4_utcb_t *utcb) L4_NOTHROW;
00254
00265 L4_INLINE l4_msgtag_t
00266 l4_icu_set_mode(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t mode) L4_NOTHROW;
00267
00274 L4_INLINE l4_msgtag_t
00275 l4_icu_set_mode_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t mode,
00276 l4_utcb_t *utcb) L4_NOTHROW;
00277
00286 L4_INLINE l4_msgtag_t
00287 l4_icu_info(l4_cap_idx_t icu, l4_icu_info_t *info) L4_NOTHROW;
00288
00295 L4_INLINE l4_msgtag_t
00296 l4_icu_info_u(l4_cap_idx_t icu, l4_icu_info_t *info,
00297 l4_utcb_t *utcb) L4_NOTHROW;
00298
00305 L4_INLINE l4_msgtag_t
00306 l4_icu_msi_info(l4_cap_idx_t icu, unsigned irqnum, l4_uint64_t source,
00307 l4_icu_msi_info_t *msi_info) L4_NOTHROW;
00308
00315 L4_INLINE l4_msgtag_t
00316 l4_icu_msi_info_u(l4_cap_idx_t icu, unsigned irqnum, l4_uint64_t source,
00317 l4_icu_msi_info_t *msi_info, l4_utcb_t *utcb) L4_NOTHROW;
00318
00319
00337 L4_INLINE l4_msgtag_t
00338 l4_icu_unmask(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00339 l4_timeout_t to) L4_NOTHROW;
00340
00347 L4_INLINE l4_msgtag_t
00348 l4_icu_unmask_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00349 l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW;
00350
00368 L4_INLINE l4_msgtag_t
00369 l4_icu_mask(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00370 l4_timeout_t to) L4_NOTHROW;
00371
00378 L4_INLINE l4_msgtag_t
00379 l4_icu_mask_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00380 l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW;
00381
00385 L4_INLINE l4_msgtag_t
00386 l4_icu_control_u(l4_cap_idx_t icu, unsigned irqnum, unsigned op,
00387 l4_umword_t *label, l4_timeout_t to,
00388 l4_utcb_t *utcb) L4_NOTHROW;
00389
00390
00391 /*****
00392 * Implementations
00393 */
00394
00395 L4_INLINE l4_msgtag_t
00396 l4_icu_bind_u(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq,
00397 l4_utcb_t *utcb) L4_NOTHROW
00398 {
00399     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00400     m->mr[0] = L4_ICU_OP_BIND;
00401     m->mr[1] = irqnum;
00402     m->mr[2] = l4_map_obj_control(0, 0);
00403     m->mr[3] = l4_obj_fpage(irq, 0, L4_CAP_FPAGE_RWS).raw;
00404     return l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 2, 1, 0), L4_IPC_NEVER);
00405 }
00406
00407 L4_INLINE l4_msgtag_t
00408 l4_icu_unbind_u(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq,
00409 l4_utcb_t *utcb) L4_NOTHROW
00410 {
00411     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00412     m->mr[0] = L4_ICU_OP_UNBIND;
00413     m->mr[1] = irqnum;
00414     m->mr[2] = l4_map_obj_control(0, 0);
00415     m->mr[3] = l4_obj_fpage(irq, 0, L4_CAP_FPAGE_RWS).raw;
00416     return l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 2, 1, 0), L4_IPC_NEVER);
00417 }
00418

```

```

00419 L4_INLINE l4_msgtag_t
00420 l4_icu_info_u(l4_cap_idx_t icu, l4_icu_info_t *info,
00421              l4_utcb_t *utcb) L4_NOTHROW
00422 {
00423     l4_msgtag_t res;
00424     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00425     m->mr[0] = L4_ICU_OP_INFO;
00426     res = l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 1, 0, 0), L4_IPC_NEVER);
00427     info->features = m->mr[0];
00428     info->nr_irqs = m->mr[1];
00429     info->nr_msis = m->mr[2];
00430     return res;
00431 }
00432
00433 L4_INLINE l4_msgtag_t
00434 l4_icu_msi_info_u(l4_cap_idx_t icu, unsigned irqnum, l4_uint64_t source,
00435                  l4_icu_msi_info_t *msi_info, l4_utcb_t *utcb) L4_NOTHROW
00436 {
00437     l4_msgtag_t res;
00438     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00439     m->mr[0] = L4_ICU_OP_MSI_INFO;
00440     m->mr[1] = irqnum;
00441     m->mr64[l4_utcb_mr64_idx(2)] = source;
00442     res = l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ,
00443                                           2 + 1 * sizeof(l4_uint64_t)
00444                                           / sizeof(l4_umword_t),
00445                                           0, 0), L4_IPC_NEVER);
00446     if (L4_UNLIKELY(l4_msgtag_has_error(res)))
00447         return res;
00448
00449     if (L4_UNLIKELY(l4_msgtag_words(res) * sizeof(l4_umword_t) < sizeof(*msi_info)))
00450         return res;
00451
00452     __builtin_memcpy(msi_info, &m->mr[0], sizeof(*msi_info));
00453     return res;
00454 }
00455
00456 L4_INLINE l4_msgtag_t
00457 l4_icu_set_mode_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t mode,
00458                  l4_utcb_t *utcb) L4_NOTHROW
00459 {
00460     l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);
00461     mr->mr[0] = L4_ICU_OP_SET_MODE;
00462     mr->mr[1] = irqnum;
00463     mr->mr[2] = mode;
00464     return l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 3, 0, 0), L4_IPC_NEVER);
00465 }
00466
00467 L4_INLINE l4_msgtag_t
00468 l4_icu_control_u(l4_cap_idx_t icu, unsigned irqnum, unsigned op,
00469                 l4_umword_t *label, l4_timeout_t to,
00470                 l4_utcb_t *utcb) L4_NOTHROW
00471 {
00472     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00473     m->mr[0] = L4_ICU_OP_UNMASK + op;
00474     m->mr[1] = irqnum;
00475     if (label)
00476         return l4_ipc_send_and_wait(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 2, 0, 0),
00477                                     label, to);
00478     else
00479         return l4_ipc_send(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 2, 0, 0), to);
00480 }
00481
00482 L4_INLINE l4_msgtag_t
00483 l4_icu_mask_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00484              l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW
00485 { return l4_icu_control_u(icu, irqnum, L4_ICU_CTL_MASK, label, to, utcb); }
00486
00487 L4_INLINE l4_msgtag_t
00488 l4_icu_unmask_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00489                l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW
00490 { return l4_icu_control_u(icu, irqnum, L4_ICU_CTL_UNMASK, label, to, utcb); }
00491
00492
00493
00494
00495 L4_INLINE l4_msgtag_t
00496 l4_icu_bind(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq) L4_NOTHROW
00497 { return l4_icu_bind_u(icu, irqnum, irq, l4_utcb()); }
00498
00499 L4_INLINE l4_msgtag_t
00500 l4_icu_unbind(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq) L4_NOTHROW
00501 { return l4_icu_unbind_u(icu, irqnum, irq, l4_utcb()); }
00502
00503 L4_INLINE l4_msgtag_t
00504 l4_icu_info(l4_cap_idx_t icu, l4_icu_info_t *info) L4_NOTHROW
00505 { return l4_icu_info_u(icu, info, l4_utcb()); }

```

```

00506
00507 L4_INLINE l4_msgtag_t
00508 l4_icu_msi_info(l4_cap_idx_t icu, unsigned irqnum, l4_uint64_t source,
00509                l4_icu_msi_info_t *msi_info) L4_NOTHROW
00510 { return l4_icu_msi_info_u(icu, irqnum, source, msi_info, l4_utcb()); }
00511
00512 L4_INLINE l4_msgtag_t
00513 l4_icu_unmask(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00514              l4_timeout_t to) L4_NOTHROW
00515 { return l4_icu_control_u(icu, irqnum, L4_ICU_CTL_UNMASK, label, to, l4_utcb()); }
00516
00517 L4_INLINE l4_msgtag_t
00518 l4_icu_mask(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00519             l4_timeout_t to) L4_NOTHROW
00520 { return l4_icu_control_u(icu, irqnum, L4_ICU_CTL_MASK, label, to, l4_utcb()); }
00521
00522 L4_INLINE l4_msgtag_t
00523 l4_icu_set_mode(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t mode) L4_NOTHROW
00524 {
00525     return l4_icu_set_mode_u(icu, irqnum, mode, l4_utcb());
00526 }

```

## 17.521 iommu

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /* \file
00003  * IO-MMU interface description.
00004  */
00005 #pragma once
00006
00007 #include <l4/sys/cxx/ipc_iface>
00008
00009 namespace L4 {
00021 class Iommu :
00022     public Kobject_x<Iommu, Proto_t<L4_PROTO_IOMMU>, Type_info::Demand_t<1> >
00023 {
00024 public:
00037     L4_INLINE_RPC(
00038         l4_msgtag_t, bind, (l4_uint64_t src_id, Ipc::Cap<Task> dma_space));
00039
00050     L4_INLINE_RPC(
00051         l4_msgtag_t, unbind, (l4_uint64_t src_id, Ipc::Cap<Task> dma_space));
00052
00053     typedef Typeid::Rpc_code<l4_umword_t>::F<bind_t, unbind_t> Rpcs;
00054 };
00055
00056 }

```

## 17.522 ipc.h

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Frank Mehnert <fm3@os.inf.tu-dresden.de>,
00010  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00011  *      economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * License: see LICENSE.spdx (in this directory or the directories above)
00014  */
00015 #ifndef __L4SYS__INCLUDE__ARCH_AMD64__L4API_L4F__IPC_H__
00016 #define __L4SYS__INCLUDE__ARCH_AMD64__L4API_L4F__IPC_H__
00017
00018 #include_next <l4/sys/ipc.h>
00019
00020 L4_INLINE l4_msgtag_t
00021 l4_ipc(l4_cap_idx_t dest, l4_utcb_t *utcb,
00022        l4_umword_t flags,
00023        l4_umword_t slabel,
00024        l4_msgtag_t tag,
00025        l4_umword_t *rlabel,
00026        l4_timeout_t timeout) L4_NOTHROW
00027 {
00028     l4_umword_t dummy, dummy2;
00029     register l4_umword_t to __asm__("r8") = timeout.raw;
00030
00031     (void)utcb;
00032 }

```

```

00033     __asm__ __volatile__
00034     ("syscall"
00035      :
00036      "=d" (dummy2),
00037      "=S" (slabel),
00038      "=D" (dummy),
00039      "=a" (tag.raw)
00040      :
00041      "S" (slabel),
00042      "r" (to),
00043      "a" (tag.raw),
00044      "d" (dest | flags)
00045      :
00046      "memory", "cc", "rcx", "r11", "r15"
00047      );
00048
00049     if (rlabel)
00050         *rlabel = slabel;
00051
00052     return tag;
00053 }
00054
00055 #endif /* ! __L4SYS__INCLUDE__ARCH_AMD64__L4API_L4F__IPC_H__ */

```

## 17.523 ipc.h

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00015 #include_next <l4/sys/ipc.h>
00016
00017 #ifdef __GNUC__
00018
00019 #include <l4/sys/compiler.h>
00020 #include <l4/sys/syscall_defs.h>
00021
00022 L4_INLINE l4_msgtag_t
00023 l4_ipc(l4_cap_idx_t dest, l4_utcb_t *utcb,
00024        l4_umword_t flags,
00025        l4_umword_t slabel,
00026        l4_msgtag_t tag,
00027        l4_umword_t *rlabel,
00028        l4_timeout_t timeout) L4_NOTHROW
00029 {
00030     register l4_umword_t _dest    __asm__("r2") = dest | flags;
00031     register l4_umword_t _timeout __asm__("r3") = timeout.raw;
00032     register l4_umword_t _tag     __asm__("r0") = tag.raw;
00033     register l4_umword_t _label   __asm__("r4") = slabel;
00034     (void)utcb;
00035
00036     __asm__ __volatile__
00037     (" .type __l4_sys_syscall, #function\n"
00038      "mov r5, %[sc]          \n"
00039      "blx __l4_sys_syscall  \n"
00040      :
00041      "+r" (_dest),
00042      "+r" (_timeout),
00043      "+r" (_label),
00044      "+r" (_tag)
00045      :
00046      [sc] "i" (L4_SYSCALL_INVOKE)
00047      :
00048      "cc", "memory", "r5", "ip", "lr");
00049
00050     if (rlabel)
00051         *rlabel = _label;
00052     tag.raw = _tag;
00053
00054     return tag;
00055 }
00056
00057 #endif // __GNUC__

```

## 17.524 ipc.h

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00015 #include_next <l4/sys/ipc.h>
00016
00017 #ifdef __GNUC__
00018
00019 #include <l4/sys/compiler.h>
00020
00021 L4_BEGIN_DECLS
00022
00023 struct __l4_sys_syscall_res
00024 {
00025     l4_mword_t tag;
00026     l4_umword_t label;
00027 };
00028
00029 extern struct __l4_sys_syscall_res
00030 __l4_sys_syscall(l4_mword_t tag,
00031                 l4_umword_t slabel,
00032                 l4_umword_t dest,
00033                 l4_umword_t timeout) L4_NOTHROW;
00034
00035 L4_END_DECLS
00036
00037 L4_INLINE l4_msgtag_t
00038 l4_ipc(l4_cap_idx_t dest, l4_utcb_t *utcb,
00039        l4_umword_t flags,
00040        l4_umword_t slabel,
00041        l4_msgtag_t tag,
00042        l4_umword_t *rlabel,
00043        l4_timeout_t timeout) L4_NOTHROW
00044 {
00045     // No need for memory clobbers. The compiler has to assume that all global
00046     // data is read/written because __l4_sys_syscall is implemented in a
00047     // different translation unit.
00048     struct __l4_sys_syscall_res res
00049         = __l4_sys_syscall(tag.raw, slabel, dest | flags, timeout.raw);
00050
00051     (void)utcb;
00052
00053     if (rlabel)
00054         *rlabel = res.label;
00055     tag.raw = res.tag;
00056
00057     return tag;
00058 }
00059
00060 #endif // __GNUC__

```

## 17.525 l4/sys/ipc.h File Reference

Common IPC interface.

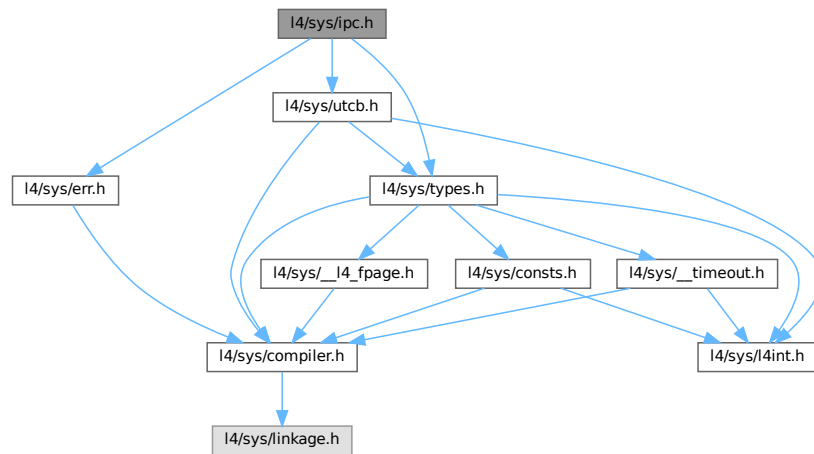
```

#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/err.h>

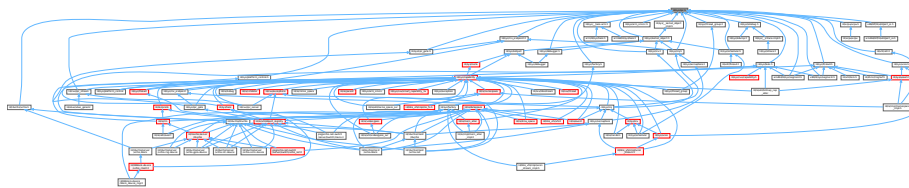
```



Include dependency graph for ipc.h:



This graph shows which files directly or indirectly include this file:



## Enumerations

- enum `l4_ipc_tcr_error_t` {  
`L4_IPC_ERROR_MASK` = 0x1F , `L4_IPC_SND_ERR_MASK` = 0x01 , `L4_IPC_ENOT_EXISTENT` = 0x04 ,  
`L4_IPC_RETIMEOUT` = 0x03 ,  
`L4_IPC_SETIMEOUT` = 0x02 , `L4_IPC_RECANCELED` = 0x07 , `L4_IPC_SECANCELED` = 0x06 ,  
`L4_IPC_REMAPFAILED` = 0x11 ,  
`L4_IPC_SEMAPFAILED` = 0x10 , `L4_IPC_RESNDPFTO` = 0x0b , `L4_IPC_SESNDPFTO` = 0x0a ,  
`L4_IPC_RERCVPFTO` = 0x0d ,  
`L4_IPC_SERCVPFTO` = 0x0c , `L4_IPC_REABORTED` = 0x0f , `L4_IPC_SEABORTED` = 0x0e ,  
`L4_IPC_REMSGCUT` = 0x09 ,  
`L4_IPC_SEMSGCUT` = 0x08 }

*Error codes in the error TCR.*

## Functions

- `l4_umword_t l4_ipc_error(l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW`  
*Get the IPC error code for an IPC operation.*
- `long l4_error(l4_msgtag_t tag) L4_NOTHROW`  
*Get IPC error code if any or message tag label otherwise for an IPC call.*
- `int l4_ipc_is_snd_error(l4_utcb_t *utcb) L4_NOTHROW`  
*Returns whether an error occurred in send phase of an invocation.*

- `int l4_ipc_is_rcv_error (l4_utcb_t *utcb) L4_NOTHROW`  
Returns whether an error occurred in receive phase of an invocation.
- `int l4_ipc_error_code (l4_utcb_t *utcb) L4_NOTHROW`  
Get the error condition of the last invocation from the TCR.
- `long l4_ipc_to_errno (unsigned long ipc_error_code) L4_NOTHROW`  
Get a negative error code for the given IPC error code.
- `l4_msgtag_t l4_ipc_send (l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag, l4_timeout_t timeout) L4_NOTHROW`  
Send a message to an object (do **not** wait for a reply).
- `l4_msgtag_t l4_ipc_wait (l4_utcb_t *utcb, l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW`  
Wait for an incoming message from any possible sender.
- `l4_msgtag_t l4_ipc_receive (l4_cap_idx_t object, l4_utcb_t *utcb, l4_timeout_t timeout) L4_NOTHROW`  
Wait for a message from a specific source.
- `l4_msgtag_t l4_ipc_call (l4_cap_idx_t object, l4_utcb_t *utcb, l4_msgtag_t tag, l4_timeout_t timeout) L4_NOTHROW`  
Object call (usual invocation).
- `l4_msgtag_t l4_ipc_reply_and_wait (l4_utcb_t *utcb, l4_msgtag_t tag, l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW`  
Reply and wait operation (uses the reply capability).
- `l4_msgtag_t l4_ipc_send_and_wait (l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag, l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW`  
Send a message and do an open wait.
- `l4_msgtag_t l4_ipc (l4_cap_idx_t dest, l4_utcb_t *utcb, l4_umword_t flags, l4_umword_t slabel, l4_msgtag_t tag, l4_umword_t *rlabel, l4_timeout_t timeout) L4_NOTHROW`  
Generic L4 object invocation.
- `l4_msgtag_t l4_ipc_sleep (l4_timeout_t timeout) L4_NOTHROW`  
Sleep for an amount of time.
- `l4_msgtag_t l4_ipc_sleep_ms (l4_uint32_t ms) L4_NOTHROW`  
Sleep for a certain amount of milliseconds.
- `l4_msgtag_t l4_ipc_sleep_us (l4_uint64_t us) L4_NOTHROW`  
Sleep for a certain amount of microseconds.
- `int l4_sndfpage_add (l4_fpage_t const snd_fpage, unsigned long snd_base, l4_msgtag_t *tag) L4_NOTHROW`  
Add a flexpage to be sent to the UTCB.

## 17.525.1 Detailed Description

Common IPC interface.

Definition in file [ipc.h](#).

## 17.525.2 Function Documentation

### 17.525.2.1 l4\_ipc\_to\_errno()

```
long l4_ipc_to_errno (
    unsigned long ipc_error_code) [inline]
```

Get a negative error code for the given IPC error code.

#### Parameters

<i>ipc_error_code</i>	IPC error code as delivered by the kernel. (or returned by the <a href="#">l4_ipc_error_code()</a> function).
-----------------------	---------------------------------------------------------------------------------------------------------------

**Returns**

negative error code in the range of [L4\\_EIPC\\_LO](#) to [L4\\_EIPC\\_HI](#).

Definition at line 561 of file [ipc.h](#).

References [L4\\_EIPC\\_LO](#), [L4\\_INLINE](#), and [L4\\_NOTHROW](#).

## 17.526 ipc.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Björn Döbel <doebel@os.inf.tu-dresden.de>,
00010  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00011  *      economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * License: see LICENSE.spdx (in this directory or the directories above)
00014  */
00015 #ifndef __L4SYS__INCLUDE__L4API_FIASCO__IPC_H__
00016 #define __L4SYS__INCLUDE__L4API_FIASCO__IPC_H__
00017
00018 #include <l4/sys/types.h>
00019 #include <l4/sys/utcb.h>
00020 #include <l4/sys/err.h>
00021
00061
00062 /***** IPC result checking *****/
00063 *** IPC result checking
00064 *****/
00065
00073
00081 enum l4_ipc_tcr_error_t
00082 {
00083     L4_IPC_ERROR_MASK          = 0x1F,
00084     L4_IPC_SND_ERR_MASK       = 0x01,
00085
00086     L4_IPC_ENOT_EXISTENT      = 0x04,
00089     L4_IPC_RETIMEOUT          = 0x03,
00092     L4_IPC_SETIMEOUT          = 0x02,
00095     L4_IPC_RECANCELED         = 0x07,
00098     L4_IPC_SECANCELED         = 0x06,
00101     L4_IPC_REMAPFAILED        = 0x11,
00105     L4_IPC_SEMAPFAILED        = 0x10,
00108     L4_IPC_RESNDPFTO          = 0x0b,
00112     L4_IPC_SESNDPFTO          = 0x0a,
00116     L4_IPC_RERCVPFTO          = 0x0d,
00120     L4_IPC_SERCVPFTO          = 0x0c,
00124     L4_IPC_REABORTED          = 0x0f,
00127     L4_IPC_SEABORTED          = 0x0e,
00136     L4_IPC_REMSGCUT           = 0x09,
00137
00143     L4_IPC_SEMSGCUT           = 0x08,
00144 };
00145
00146
00157 L4_INLINE l4_umword_t
00158 l4_ipc_error(l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW;
00159
00160
00177 L4_INLINE long
00178 l4_error(l4_msgtag_t tag) L4_NOTHROW;
00179
00180 L4_INLINE long
00181 l4_error_u(l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW;
00182
00183 /***** IPC results *****/
00184 *** IPC results
00185 *****/

```

```

00186
00196 L4_INLINE int l4_ipc_is_snd_error(l4_utcb_t *utcb) L4_NOTHROW;
00197
00207 L4_INLINE int l4_ipc_is_rcv_error(l4_utcb_t *utcb) L4_NOTHROW;
00208
00218 L4_INLINE int l4_ipc_error_code(l4_utcb_t *utcb) L4_NOTHROW;
00219
00226 L4_INLINE long l4_ipc_to_errno(unsigned long ipc_error_code) L4_NOTHROW;
00227
00228
00229 /*****
00230 *** IPC calls
00231 *****/
00232
00261 L4_INLINE l4_msgtag_t
00262 l4_ipc_send(l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag,
00263             l4_timeout_t timeout) L4_NOTHROW;
00264
00265
00292 L4_INLINE l4_msgtag_t
00293 l4_ipc_wait(l4_utcb_t *utcb, l4_umword_t *label,
00294             l4_timeout_t timeout) L4_NOTHROW;
00295
00296
00321 L4_INLINE l4_msgtag_t
00322 l4_ipc_receive(l4_cap_idx_t object, l4_utcb_t *utcb,
00323               l4_timeout_t timeout) L4_NOTHROW;
00324
00355 L4_INLINE l4_msgtag_t
00356 l4_ipc_call(l4_cap_idx_t object, l4_utcb_t *utcb, l4_msgtag_t tag,
00357             l4_timeout_t timeout) L4_NOTHROW;
00358
00359
00385 L4_INLINE l4_msgtag_t
00386 l4_ipc_reply_and_wait(l4_utcb_t *utcb, l4_msgtag_t tag,
00387                      l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW;
00388
00417 L4_INLINE l4_msgtag_t
00418 l4_ipc_send_and_wait(l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag,
00419                     l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW;
00420
00426
00427 #if 0
00438 L4_INLINE l4_msgtag_t
00439 l4_ipc_wait_next_period(l4_utcb_t *utcb,
00440                        l4_umword_t *label,
00441                        l4_timeout_t timeout);
00442
00443 #endif
00444
00464 L4_ALWAYS_INLINE l4_msgtag_t
00465 l4_ipc(l4_cap_idx_t dest,
00466        l4_utcb_t *utcb,
00467        l4_umword_t flags,
00468        l4_umword_t slabel,
00469        l4_msgtag_t tag,
00470        l4_umword_t *rlabel,
00471        l4_timeout_t timeout) L4_NOTHROW;
00472
00489 L4_INLINE l4_msgtag_t
00490 l4_ipc_sleep(l4_timeout_t timeout) L4_NOTHROW;
00491
00508 L4_INLINE l4_msgtag_t
00509 l4_ipc_sleep_ms(l4_uint32_t ms) L4_NOTHROW;
00510
00527 L4_INLINE l4_msgtag_t
00528 l4_ipc_sleep_us(l4_uint64_t us) L4_NOTHROW;
00529
00544 L4_INLINE int
00545 l4_sndfpage_add(l4_fpage_t const snd_fpage, unsigned long snd_base,
00546                l4_msgtag_t *tag) L4_NOTHROW;
00547
00548 /*
00549 * \internal
00550 * \ingroup l4_ipc_api
00551 */
00552 L4_INLINE int
00553 l4_sndfpage_add_u(l4_fpage_t const snd_fpage, unsigned long snd_base,
00554                  l4_msgtag_t *tag, l4_utcb_t *utcb) L4_NOTHROW;
00555
00556
00557 /*****
00558 * Implementations
00559 *****/
00560
00561 L4_INLINE long l4_ipc_to_errno(unsigned long ipc_error_code) L4_NOTHROW
00562 { return -(L4_EIPC_LO + ipc_error_code); }

```

```

00563
00564 L4_INLINE l4_msgtag_t
00565 l4_ipc_call(l4_cap_idx_t object, l4_utcb_t *utcb, l4_msgtag_t tag,
00566             l4_timeout_t timeout) L4_NOTHROW
00567 {
00568     return l4_ipc(object, utcb, L4_SYSF_CALL, 0, tag, 0, timeout);
00569 }
00570
00571 L4_INLINE l4_msgtag_t
00572 l4_ipc_reply_and_wait(l4_utcb_t *utcb, l4_msgtag_t tag,
00573                      l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW
00574 {
00575     return l4_ipc(L4_INVALID_CAP, utcb, L4_SYSF_REPLY_AND_WAIT, 0, tag, label, timeout);
00576 }
00577
00578 L4_INLINE l4_msgtag_t
00579 l4_ipc_send_and_wait(l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag,
00580                     l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW
00581 {
00582     return l4_ipc(dest, utcb, L4_SYSF_SEND_AND_WAIT, 0, tag, label, timeout);
00583 }
00584
00585 L4_INLINE l4_msgtag_t
00586 l4_ipc_send(l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag,
00587             l4_timeout_t timeout) L4_NOTHROW
00588 {
00589     return l4_ipc(dest, utcb, L4_SYSF_SEND, 0, tag, 0, timeout);
00590 }
00591
00592 L4_INLINE l4_msgtag_t
00593 l4_ipc_wait(l4_utcb_t *utcb, l4_umword_t *label,
00594             l4_timeout_t timeout) L4_NOTHROW
00595 {
00596     l4_msgtag_t t;
00597     t.raw = 0;
00598     return l4_ipc(L4_INVALID_CAP, utcb, L4_SYSF_WAIT, 0, t, label, timeout);
00599 }
00600
00601 L4_INLINE l4_msgtag_t
00602 l4_ipc_receive(l4_cap_idx_t object, l4_utcb_t *utcb,
00603               l4_timeout_t timeout) L4_NOTHROW
00604 {
00605     l4_msgtag_t t;
00606     t.raw = 0;
00607     return l4_ipc(object, utcb, L4_SYSF_RECV, 0, t, 0, timeout);
00608 }
00609
00610 L4_INLINE l4_msgtag_t
00611 l4_ipc_sleep(l4_timeout_t timeout) L4_NOTHROW
00612 { return l4_ipc_receive(L4_INVALID_CAP, NULL, timeout); }
00613
00614 L4_INLINE l4_msgtag_t
00615 l4_ipc_sleep_ms(l4_uint32_t ms) L4_NOTHROW
00616 {
00617     l4_uint64_t us = ms * 1000ULL; // cannot overflow because ms < 2^32
00618     return l4_ipc_sleep(l4_timeout(L4_IPC_TIMEOUT_NEVER, l4_timeout_from_us(us)));
00619 }
00620
00621 L4_INLINE l4_msgtag_t
00622 l4_ipc_sleep_us(l4_uint64_t us) L4_NOTHROW
00623 {
00624     return l4_ipc_sleep(l4_timeout(L4_IPC_TIMEOUT_NEVER,
00625                                   l4_timeout_from_us(us)));
00626 }
00627
00628 L4_INLINE l4_umword_t
00629 l4_ipc_error(l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW
00630 {
00631     if (L4_LIKELY(!l4_msgtag_has_error(tag)))
00632         return 0;
00633     return l4_utcb_tcr_u(utcb)->error & L4_IPC_ERROR_MASK;
00634 }
00635
00636 L4_INLINE long
00637 l4_error_u(l4_msgtag_t tag, l4_utcb_t *u) L4_NOTHROW
00638 {
00639     if (L4_UNLIKELY(l4_msgtag_has_error(tag)))
00640         return l4_ipc_to_errno(l4_utcb_tcr_u(u)->error & L4_IPC_ERROR_MASK);
00641
00642     return l4_msgtag_label(tag);
00643 }
00644
00645 L4_INLINE long
00646 l4_error(l4_msgtag_t tag) L4_NOTHROW
00647 {
00648     return l4_error_u(tag, l4_utcb());
00649 }

```

```

00650
00651
00652 L4_INLINE int l4_ipc_is_snd_error(l4_utcb_t *u) L4_NOTHROW
00653 { return (l4_utcb_tcr_u(u)->error & 1) == 0; }
00654
00655 L4_INLINE int l4_ipc_is_rcv_error(l4_utcb_t *u) L4_NOTHROW
00656 { return l4_utcb_tcr_u(u)->error & 1; }
00657
00658 L4_INLINE int l4_ipc_error_code(l4_utcb_t *u) L4_NOTHROW
00659 { return l4_utcb_tcr_u(u)->error & L4_IPC_ERROR_MASK; }
00660
00661
00662 /*
00663  * \internal
00664  * \ingroup l4_ipc_api
00665  */
00666 L4_INLINE int
00667 l4_sndfpage_add_u(l4_fpage_t const snd_fpage, unsigned long snd_base,
00668                  l4_msgtag_t *tag, l4_utcb_t *utcb) L4_NOTHROW
00669 {
00670     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00671     int i = l4_msgtag_words(*tag) + 2 * l4_msgtag_items(*tag);
00672
00673     if (i >= L4_UTCB_GENERIC_DATA_SIZE - 1)
00674         return -L4_ENOMEM;
00675
00676     v->mr[i] = snd_base | L4_ITEM_MAP | L4_ITEM_CONT;
00677     v->mr[i + 1] = snd_fpage.raw;
00678
00679     *tag = l4_msgtag(l4_msgtag_label(*tag), l4_msgtag_words(*tag),
00680                    l4_msgtag_items(*tag) + 1, l4_msgtag_flags(*tag));
00681     return 0;
00682 }
00683
00684 L4_INLINE int
00685 l4_sndfpage_add(l4_fpage_t const snd_fpage, unsigned long snd_base,
00686                l4_msgtag_t *tag) L4_NOTHROW
00687 {
00688     return l4_sndfpage_add_u(snd_fpage, snd_base, tag, l4_utcb());
00689 }
00690
00691
00692 #endif /* ! __L4SYS__INCLUDE__L4API_FIASCO__IPC_H__ */

```

## 17.527 x86/i4f/i4/sys/ipc.h File Reference

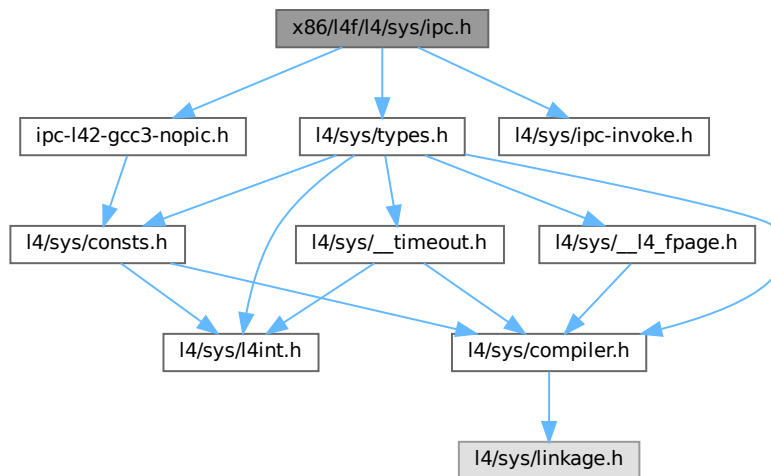
L4 IPC System Calls, x86.

```

#include <l4/sys/types.h>
#include <l4/sys/ipc-invoke.h>
#include "ipc-l42-gcc3-nopic.h"

```

Include dependency graph for ipc.h:



### 17.527.1 Detailed Description

[L4 IPC System Calls, x86.](#)

Definition in file [ipc.h](#).

## 17.528 ipc.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Lars Reuther <reuther@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #ifndef __L4_IPC_H__
00015 #define __L4_IPC_H__
00016
00017 #include <l4/sys/types.h>
00018
00019 #include_next <l4/sys/ipc.h>
00020
00021 /*****
00022  *** Implementation
00023  *****/
00024
00025 #include <l4/sys/ipc-invoke.h>
00026 #include "ipc-l42-gcc3-nopic.h"
00027
00028 #endif /* !__L4_IPC_H__ */

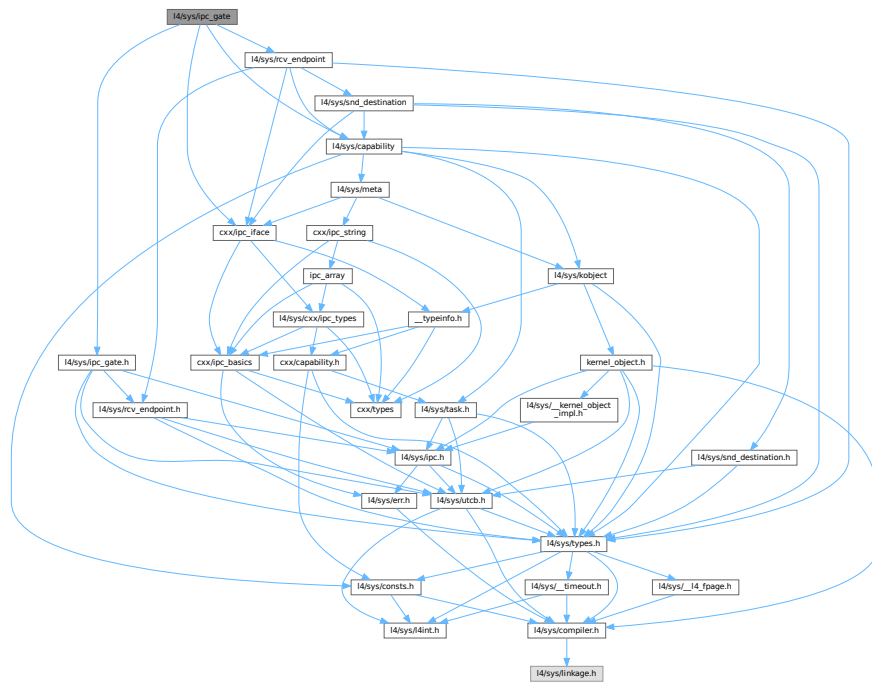
```

## 17.529 l4/sys/ipc\_gate File Reference

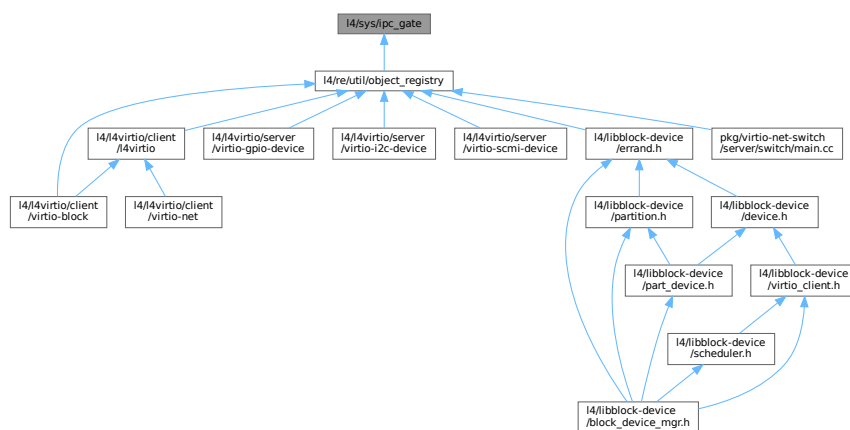
The C++ IPC gate interface.

```
#include <l4/sys/ipc_gate.h>
#include <l4/sys/capability>
#include <l4/sys/rcv_endpoint>
#include <l4/sys/cxx/ipc_iface>
```

Include dependency graph for ipc\_gate:



This graph shows which files directly or indirectly include this file:



### Data Structures

- class [L4::ipc\\_gate](#)

The C++ IPC gate interface, see [IPC-Gate API](#) for the C interface.



## Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*

## 17.529.1 Detailed Description

The C++ IPC gate interface.

Definition in file [ipc\\_gate](#).

## 17.530 ipc\_gate

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2009-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00015 #include <l4/sys/ipc_gate.h>
00016 #include <l4/sys/capability>
00017 #include <l4/sys/rcv_endpoint>
00018 #include <l4/sys/cxx/ipc_iface>
00019
00020 namespace L4 {
00021
00022 class Thread;
00023
00085 class L4_EXPORT Ipc_gate :
00086     public Kobject_t<Ipc_gate, Rcv_endpoint, L4_PROTO_KOBJECT,
00087         Type_info::Demand_t<1> >
00088 {
00089 public:
00103     L4_INLINE_RPC_OP(L4_IPC_GATE_GET_INFO_OP,
00104         l4_msgtag_t, get_infos, (l4_umword_t *label));
00105
00106     typedef L4::Typeid::Rpc_sys<bind_thread_t, get_infos_t> Rpc_sys;
00107 };
00108
00109 }

```

## 17.531 l4/sys/ipc\_gate.h File Reference

The C IPC gate interface, see [L4::ipc\\_gate](#) for the C++ interface.

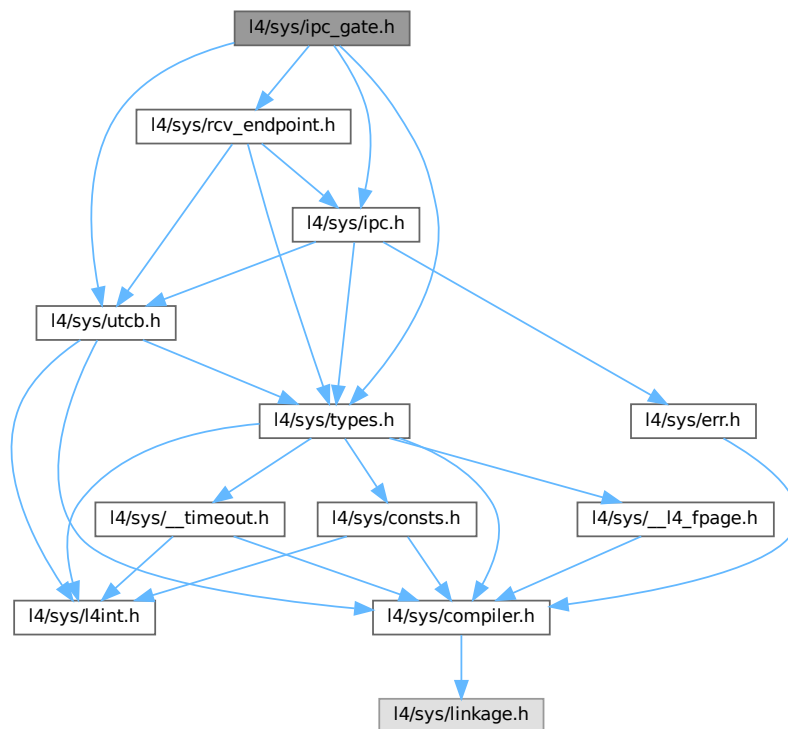
```

#include <l4/sys/utcb.h>
#include <l4/sys/types.h>
#include <l4/sys/rcv_endpoint.h>

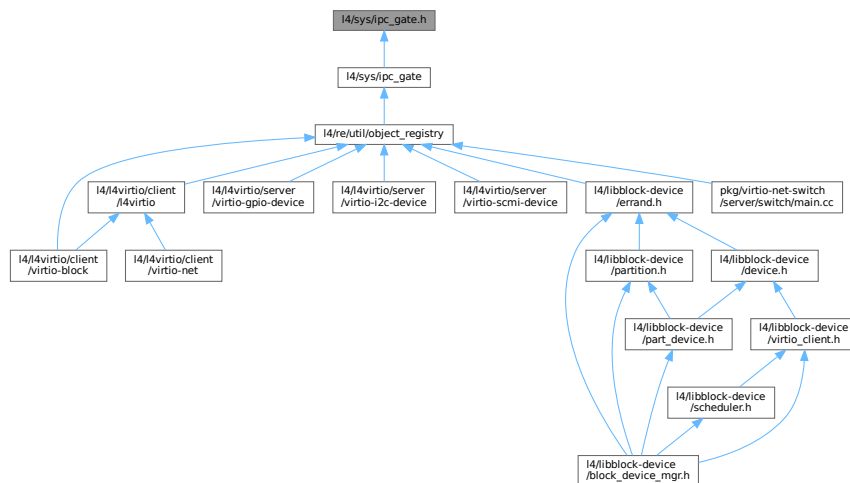
```

```
#include <l4/sys/ipc.h>
```

Include dependency graph for ipc\_gate.h:



This graph shows which files directly or indirectly include this file:



## Enumerations

- enum [L4\\_ipc\\_gate\\_ops](#) { [L4\\_IPC\\_GATE\\_BIND\\_OP](#) = 0x10 , [L4\\_IPC\\_GATE\\_GET\\_INFO\\_OP](#) = 0x11 }
- Operations on the IPC-gate.

## Functions

- [l4\\_msgtag\\_t l4\\_ipc\\_gate\\_get\\_infos](#) ([l4\\_cap\\_idx\\_t](#) gate, [l4\\_umword\\_t](#) \*label)

*Get information about the IPC-gate.*

### 17.531.1 Detailed Description

The C IPC gate interface, see [L4::ipc\\_gate](#) for the C++ interface.

IPC gates are used to create secure communication channels between protection domains. An IPC gate can be created using the [Factory](#) interface.

Depending on the permissions of the capability used, an IPC gate forwards IPC to the [Thread](#) the IPC gate is *bound* to (cf. [l4\\_rcv\\_ep\\_bind\\_thread\(\)](#) and [l4\\_rcv\\_ep\\_bind\\_snd\\_destination\(\)](#)). If the capability has the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission, only IPC using a protocol different from the [L4\\_PROTO\\_KOBJECT](#) protocol is forwarded. Without the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission, all IPC is forwarded. The latter is the usual case for a client in a client/server scenario. When not bound to a thread or thread group yet, the forwarded IPC blocks until the IPC gate is bound to a thread or thread group, or the IPC times out.

Forwarded IPC is always forwarded to the userland of the thread the IPC gate is bound to, either directly or indirectly using a thread group. That means, the [Thread](#) interface of that thread is not accessible via an IPC gate. The [IPC-Gate API](#) of an IPC gate is only accessible if the capability used has the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission (cf. previous paragraph). Conversely that means, if the capability used lacks the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission, [IPC-Gate API](#) calls are forwarded to the thread or thread group the IPC gate is bound to instead of being processed by the IPC gate itself. In a client/server scenario, a client should only get IPC gate capabilities without [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission so the client cannot tamper with the IPC gate.

When binding an IPC gate to a thread or thread group, a user-defined, kernel protected, machine-word sized payload called the IPC gate's *label* is assigned to the IPC gate (note that the two least significant bits of the label must be zero; cf. [l4\\_rcv\\_ep\\_bind\\_thread\(\)](#) and [l4\\_rcv\\_ep\\_bind\\_snd\\_destination\(\)](#)). When a send-only IPC or call IPC is forwarded via an IPC gate, the label provided by the sender is ignored and replaced by the IPC gate's label where the two least significant bits are set to the [L4\\_CAP\\_FPAGE\\_S](#) and [L4\\_CAP\\_FPAGE\\_W](#) permissions of the capability used. The replaced label is only visible to the thread the IPC gate is bound to upon receive (or to the selected thread from the thread group the IPC gate is bound to). However, the configured label of an IPC gate can also be queried via [l4\\_ipc\\_gate\\_get\\_infos\(\)](#) if the capability used has the [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#) permission.

When deleting an IPC gate or when unbinding it from a thread or thread group, the label of IPC already in flight won't be changed. To ensure that no IPC from this IPC gate is received by a thread with an unexpected label, [l4\\_thread\\_modify\\_sender\\_start\(\)](#) shall be used to change the labels of every pending IPC to that gate. This is also required if the label of an already bound IPC gate is changed. It is not necessary after binding the IPC gate to a thread or thread group for the first time.

When binding a currently bound IPC gate to a new thread or thread group, the same label should be used that was used with the old thread. Otherwise the old and the new thread need to synchronize to avoid IPC messages with unexpected labels.

## Include File

```
#include <l4/sys/ipc_gate.h>
```

For the C++ interface refer to the [L4::ipc\\_gate](#) documentation.

## See also

[Object Invocation](#)

Definition in file [ipc\\_gate.h](#).

## 17.532 ipc\_gate.h

[Go to the documentation of this file.](#)

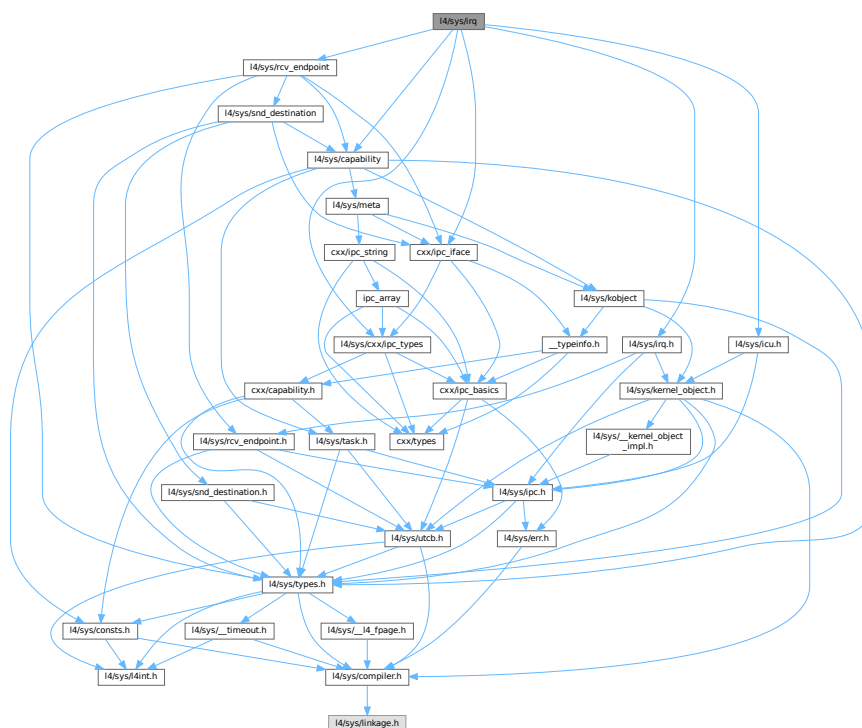
```

00001 /*
00002  * (c) 2009-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 #include <l4/sys/utcb.h>
00012 #include <l4/sys/types.h>
00013 #include <l4/sys/rcv_endpoint.h>
00014
00015 L4_INLINE l4_msgtag_t
00016 l4_ipc_gate_get_infos(l4_cap_idx_t gate, l4_umword_t *label);
00017
00018 L4_INLINE l4_msgtag_t
00019 l4_ipc_gate_get_infos_u(l4_cap_idx_t gate, l4_umword_t *label, l4_utcb_t *utcb);
00020
00021 enum L4_ipc_gate_ops
00022 {
00023     L4_IPC_GATE_BIND_OP      = 0x10,
00024     L4_IPC_GATE_GET_INFO_OP = 0x11,
00025 };
00026
00027 /* IMPLEMENTATION -----*/
00028
00029 #include <l4/sys/ipc.h>
00030
00031 L4_INLINE l4_msgtag_t
00032 l4_ipc_gate_bind_thread_u(l4_cap_idx_t gate,
00033                          l4_cap_idx_t thread, l4_umword_t label,
00034                          l4_utcb_t *utcb)
00035 {
00036     return l4_rcv_ep_bind_thread_u(gate, thread, label, utcb);
00037 }
00038
00039 L4_INLINE l4_msgtag_t
00040 l4_ipc_gate_bind_snd_destination_u(l4_cap_idx_t gate,
00041                                   l4_cap_idx_t snd_dst, l4_umword_t label,
00042                                   l4_utcb_t *utcb)
00043 {
00044     return l4_rcv_ep_bind_snd_destination_u(gate, snd_dst, label, utcb);
00045 }
00046
00047 L4_INLINE l4_msgtag_t
00048 l4_ipc_gate_get_infos_u(l4_cap_idx_t gate, l4_umword_t *label, l4_utcb_t *utcb)
00049 {
00050     l4_msgtag_t tag;
00051     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00052     m->mr[0] = L4_IPC_GATE_GET_INFO_OP;
00053     tag = l4_ipc_call(gate, utcb, l4_msgtag(L4_PROTO_KOBJECT, 1, 0, 0),
00054                     L4_IPC_NEVER);
00055     if (!l4_msgtag_has_error(tag) && l4_msgtag_label(tag) >= 0)
00056         *label = m->mr[0];
00057     return tag;
00058 }
00059
00060 L4_INLINE l4_msgtag_t
00061 l4_ipc_gate_bind_thread(l4_cap_idx_t gate, l4_cap_idx_t thread,
00062                        l4_umword_t label)
00063 {
00064     return l4_rcv_ep_bind_thread_u(gate, thread, label, l4_utcb());
00065 }
00066
00067 L4_INLINE l4_msgtag_t
00068 l4_ipc_gate_bind_snd_destination(l4_cap_idx_t gate, l4_cap_idx_t snd_dst,
00069                                l4_umword_t label)
00070 {
00071     return l4_rcv_ep_bind_snd_destination_u(gate, snd_dst, label, l4_utcb());
00072 }
00073
00074 L4_INLINE l4_msgtag_t
00075 l4_ipc_gate_get_infos(l4_cap_idx_t gate, l4_umword_t *label)
00076 {
00077     return l4_ipc_gate_get_infos_u(gate, label, l4_utcb());
00078 }

```

### 17.533 l4/sys/irq File Reference

```
#include <l4/sys/icu.h>
#include <l4/sys/irq.h>
#include <l4/sys/capability>
#include <l4/sys/rcv_endpoint>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/cxx/ipc_types>
Include dependency graph for irq:
```



The diagram is a complex network graph with numerous nodes and edges. The nodes are represented by rectangular boxes, many of which contain text labels. The edges are blue lines connecting the nodes, forming a dense web of relationships. The network is organized into several clusters or communities, with some nodes acting as central hubs. The labels on the nodes include names of individuals (e.g., "John Doe", "Jane Smith"), organizations (e.g., "ABC Company", "XYZ Corporation"), and locations (e.g., "New York City", "Los Angeles"). The overall structure suggests a highly interconnected system, possibly representing a social network, a corporate hierarchy, or a geographical network.

## Data Structures

- class [L4::Irq\\_eoi](#)  
*Interface for sending an unmask message to an object.*
- struct [L4::Triggerable](#)  
*Interface that allows an object to be triggered by some source.*
- class [L4::Irq](#)  
*C++ [Irq](#) interface, see [IRQs](#) for the C interface.*
- class [L4::Icu](#)  
*C++ [Icu](#) interface, see [Interrupt controller](#) for the C interface.*
- class [L4::Icu::Info](#)  
*This class encapsulates information about an ICU.*

## Namespaces

- namespace [L4](#)  
*[L4](#) low-level kernel interface.*

## 17.533.1 Detailed Description

C++ Irq interface.

Definition in file [irq](#).

## 17.534 irq

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013
00014 #pragma once
00015
00016 #include <l4/sys/icu.h>
00017 #include <l4/sys/irq.h>
00018 #include <l4/sys/capability>
00019 #include <l4/sys/rcv_endpoint>
00020 #include <l4/sys/cxx/ipc_iface>
00021 #include <l4/sys/cxx/ipc_types>
00022
00023 namespace L4 {
00024
00037 class Irq_eoi : public Kobject_0t<Irq_eoi, L4::PROTO_EMPTY>
00038 {
00039 public:
00064   l4_msgtag_t unmask(unsigned irqnum, l4_umword_t *label = 0,
00065                       l4_timeout_t to = L4_IPC_NEVER,
00066                       l4_utcb_t *utcb = l4_utcb()) noexcept
00067   {
00068       return l4_icu_control_u(cap(), irqnum, L4_ICU_CTL_UNMASK, label, to, utcb);
00069   }
00070 };
00071
00079 struct Triggerable : Kobject_t<Triggerable, Irq_eoi, L4_PROTO_IRQ>
00080 {
00091   l4_msgtag_t trigger(l4_utcb_t *utcb = l4_utcb()) noexcept
00092   { return l4_irq_trigger_u(cap(), utcb); }
00093 };

```

```

00094
00120 class Irq : public Kobject_2t<Irq, Triggerable, Rcv_endpoint, L4_PROTO_IRQ_SENDER>
00121 {
00122 public:
00123     using Triggerable::unmask;
00124
00158     l4_msgtag_t bind_vcpu(L4::Cap<Thread> const &thread, l4_umword_t cfg,
00159                         l4_utcb_t *utcb = l4_utcb()) noexcept
00160     { return l4_irq_bind_vcpu_u(cap(), thread.cap(), cfg, utcb); }
00161
00176     l4_msgtag_t detach(l4_utcb_t *utcb = l4_utcb()) noexcept
00177     { return l4_irq_detach_u(cap(), utcb); }
00178
00179
00191     l4_msgtag_t receive(l4_timeout_t timeout = L4_IPC_NEVER,
00192                       l4_utcb_t *utcb = l4_utcb()) noexcept
00193     { return l4_irq_receive_u(cap(), timeout, utcb); }
00194
00204     l4_msgtag_t wait(l4_umword_t *label, l4_timeout_t timeout = L4_IPC_NEVER,
00205                    l4_utcb_t *utcb = l4_utcb()) noexcept
00206     { return unmask(-1, label, timeout, utcb); }
00207
00221     l4_msgtag_t unmask(l4_utcb_t *utcb = l4_utcb()) noexcept
00222     { return unmask(-1, 0, L4_IPC_NEVER, utcb); }
00223 };
00224
00250 class Icu :
00251     public Kobject_t<Icu, Irq_eoi, L4_PROTO_IRQ,
00252                   Type_info::Demand_t<1> >
00253 {
00254 public:
00255     enum Mode
00256     {
00257         F_none           = L4_IRQ_F_NONE,
00258         F_level_high     = L4_IRQ_F_LEVEL_HIGH,
00259         F_level_low      = L4_IRQ_F_LEVEL_LOW,
00260         F_pos_edge       = L4_IRQ_F_POS_EDGE,
00261         F_neg_edge       = L4_IRQ_F_NEG_EDGE,
00262         F_both_edge      = L4_IRQ_F_BOTH_EDGE,
00263         F_mask           = L4_IRQ_F_MASK,
00264
00265         F_set_wakeup     = L4_IRQ_F_SET_WAKEUP,
00266         F_clear_wakeup   = L4_IRQ_F_CLEAR_WAKEUP,
00267     };
00268
00269     enum Flags
00270     {
00271         F_msi = L4_ICU_FLAG_MSI
00272     };
00273
00277     class Info : public l4_icu_info_t
00278     {
00279     public:
00281         bool supports_msi() const noexcept { return features & F_msi; }
00282     };
00283
00310     l4_msgtag_t bind(unsigned irqnum, L4::Cap<Triggerable> irq,
00311                    l4_utcb_t *utcb = l4_utcb()) noexcept
00312     { return l4_icu_bind_u(cap(), irqnum, irq.cap(), utcb); }
00313
00314     L4_RPC_NF_OP(
00315         L4_ICU_OP_BIND,
00316         l4_msgtag_t, bind, (l4_umword_t irqnum, Ipc::Cap<Irq> irq)
00317     );
00318
00328     l4_msgtag_t unbind(unsigned irqnum, L4::Cap<Triggerable> irq,
00329                      l4_utcb_t *utcb = l4_utcb()) noexcept
00330     { return l4_icu_unbind_u(cap(), irqnum, irq.cap(), utcb); }
00331
00332     L4_RPC_NF_OP(
00333         L4_ICU_OP_UNBIND,
00334         l4_msgtag_t, unbind, (l4_umword_t irqnum, Ipc::Cap<Irq> irq)
00335     );
00336
00345     l4_msgtag_t info(l4_icu_info_t *info, l4_utcb_t *utcb = l4_utcb()) noexcept
00346     { return l4_icu_info_u(cap(), info, utcb); }
00347
00348     struct _Info { l4_umword_t features, nr_irqs, nr_msis; };
00349     L4_RPC_NF_OP(L4_ICU_OP_INFO, l4_msgtag_t, info, (_Info *info));
00350
00363     L4_INLINE_RPC_OP(L4_ICU_OP_MSI_INFO,
00364                    l4_msgtag_t, msi_info, (l4_umword_t irqnum, l4_uint64_t source,
00365                    l4_icu_msi_info_t *msi_info));
00366
00370     l4_msgtag_t control(unsigned irqnum, unsigned op, l4_umword_t *label,
00371                      l4_timeout_t to, l4_utcb_t *utcb = l4_utcb()) noexcept
00372     { return l4_icu_control_u(cap(), irqnum, op, label, to, utcb); }

```

```

00373
00393  l4_msgtag_t mask(unsigned irqnum,
00394                  l4_umword_t *label = 0,
00395                  l4_timeout_t to = L4_IPC_NEVER,
00396                  l4_utcb_t *utcb = l4_utcb()) noexcept
00397  { return l4_icu_mask_u(cap(), irqnum, label, to, utcb); }
00398
00399  L4_RPC_NF_OP(
00400      L4_ICU_OP_MASK,
00401      l4_msgtag_t, mask, (l4_umword_t irqnum),
00402      L4::Ipc::Send_only
00403  );
00404
00405
00406  L4_RPC_NF_OP(
00407      L4_ICU_OP_UNMASK,
00408      l4_msgtag_t, unmask, (l4_umword_t irqnum),
00409      L4::Ipc::Send_only
00410  );
00411
00421  l4_msgtag_t set_mode(unsigned irqnum, l4_umword_t mode,
00422                      l4_utcb_t *utcb = l4_utcb()) noexcept
00423  { return l4_icu_set_mode_u(cap(), irqnum, mode, utcb); }
00424
00425  L4_RPC_NF_OP(
00426      L4_ICU_OP_SET_MODE,
00427      l4_msgtag_t, set_mode, (l4_umword_t irqnum, l4_umword_t mode)
00428  );
00429
00430  typedef L4::Typeid::Rpcsys<
00431      bind_t, unbind_t, info_t, msi_info_t, unmask_t, mask_t, set_mode_t
00432  > Rpcsys;
00433 };
00434
00435 }

```

## 17.535 l4/sys/kdebug.h File Reference

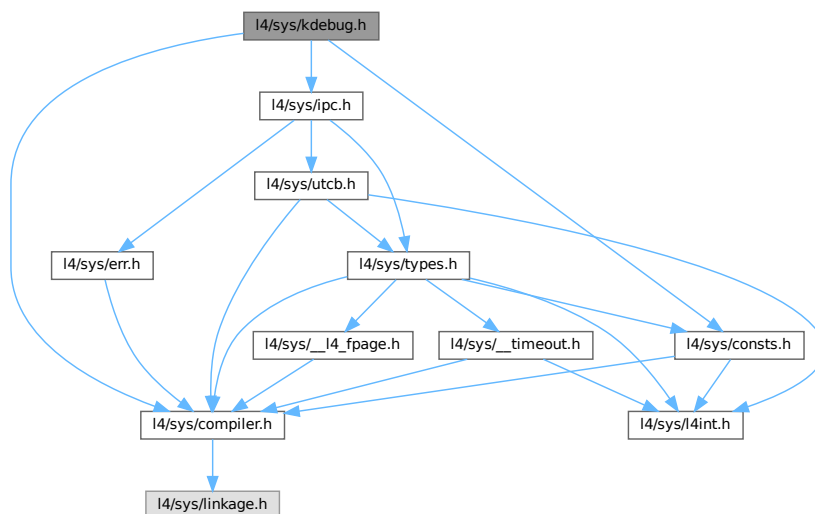
Functionality for invoking the kernel debugger.

```

#include <l4/sys/compiler.h>
#include <l4/sys/consts.h>
#include <l4/sys/ipc.h>

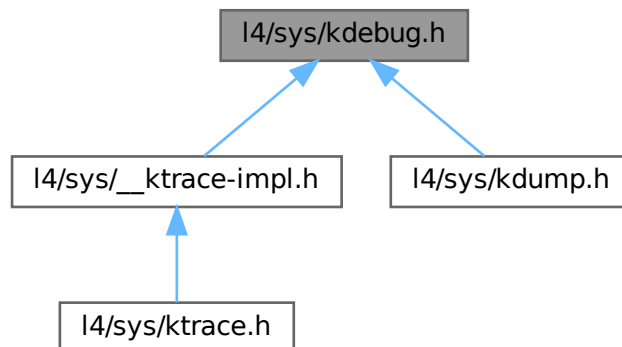
```

Include dependency graph for kdebug.h:





This graph shows which files directly or indirectly include this file:



## Enumerations

- enum [l4\\_kdebug\\_group\\_t](#)  
*Opcode groups for operations that can be invoked on the base debugger capability.*
- enum [l4\\_kdebug\\_ops\\_t](#)  
*Op-codes for operations that can be invoked on the base debugger capability.*

## Functions

- void [enter\\_kdebug](#) (char const \*text) [L4\\_NOTHROW](#)  
*Enter the kernel debugger.*
- [l4\\_msgtag\\_t \\_\\_kdebug\\_op](#) (unsigned op) [L4\\_NOTHROW](#)  
*Invoke a nullary operation on the base debugger capability.*
- [l4\\_msgtag\\_t \\_\\_kdebug\\_text](#) (unsigned op, char const \*text, unsigned len) [L4\\_NOTHROW](#)  
*Invoke a text output operation on the base debugger capability.*
- [l4\\_msgtag\\_t \\_\\_kdebug\\_3\\_text](#) (unsigned op, char const \*text, unsigned len, [l4\\_umword\\_t](#) v1, [l4\\_umword\\_t](#) v2, [l4\\_umword\\_t](#) v3) [L4\\_NOTHROW](#)  
*Invoke a text output operation with 3 additional machine word arguments on the base debugger capability.*
- [l4\\_msgtag\\_t \\_\\_kdebug\\_op\\_1](#) (unsigned op, [l4\\_mword\\_t](#) val) [L4\\_NOTHROW](#)  
*Invoke an unary operation on the base debugger capability.*
- void [outnstring](#) (char const \*text, unsigned len)  
*Output a fixed-length string via the kernel debugger.*
- void [outstring](#) (char const \*text)  
*Output a string via the kernel debugger.*
- void [outchar](#) (char c)  
*Output a single character via the kernel debugger.*
- void [outumword](#) ([l4\\_umword\\_t](#) number)  
*Output a hexadecimal unsigned machine word via the kernel debugger.*
- void [outhex64](#) ([l4\\_uint64\\_t](#) number)  
*Output a 64-bit unsigned hexadecimal number via the kernel debugger.*
- void [outhex32](#) ([l4\\_uint32\\_t](#) number)

- *Output a 32-bit unsigned hexadecimal number via the kernel debugger.*  
void [outhex20](#) ([l4\\_uint32\\_t](#) number)
- *Output a 20-bit unsigned hexadecimal number via the kernel debugger.*  
void [outhex16](#) ([l4\\_uint16\\_t](#) number)
- *Output a 16-bit unsigned hexadecimal number via the kernel debugger.*  
void [outhex12](#) ([l4\\_uint16\\_t](#) number)
- *Output a 12-bit unsigned hexadecimal number via the kernel debugger.*  
void [outhex8](#) ([l4\\_uint8\\_t](#) number)
- *Output an 8-bit unsigned hexadecimal number via the kernel debugger.*  
void [outdec](#) ([l4\\_mword\\_t](#) number)
- *Output a decimal unsigned machine word via the kernel debugger.*

### 17.535.1 Detailed Description

Functionality for invoking the kernel debugger.

Definition in file [kdebug.h](#).

### 17.535.2 Enumeration Type Documentation

#### 17.535.2.1 [l4\\_kdebug\\_ops\\_t](#)

enum [l4\\_kdebug\\_ops\\_t](#)

Op-codes for operations that can be invoked on the base debugger capability.

See also [\\_\\_ktrace-impl.h](#) for additional op-codes.

Definition at line 44 of file [kdebug.h](#).

### 17.535.3 Function Documentation

#### 17.535.3.1 [\\_\\_kdebug\\_3\\_text\(\)](#)

```
l4\_msgtag\_t __kdebug_3_text (
    unsigned op,
    char const * text,
    unsigned len,
    l4\_umword\_t v1,
    l4\_umword\_t v2,
    l4\_umword\_t v3) [inline]
```

Invoke a text output operation with 3 additional machine word arguments on the base debugger capability.

#### Parameters

---

<i>op</i>	Text output operation code from <a href="#">l4_kdebug_ops_t</a> or a value above 0x200 used by the kernel trace buffer implementation ( <a href="#">__ktrace-impl.h</a> ).
<i>text</i>	Output string.
<i>len</i>	Length of the output string. The maximum length is limited to <a href="#">L4_UTCB_GENERIC_DATA_SIZE</a> - 5 machine words. Output strings longer than this limit will be cropped.
<i>v1</i>	First machine word argument.
<i>v2</i>	Second machine word argument.
<i>v3</i>	Third machine word argument.

### Return values

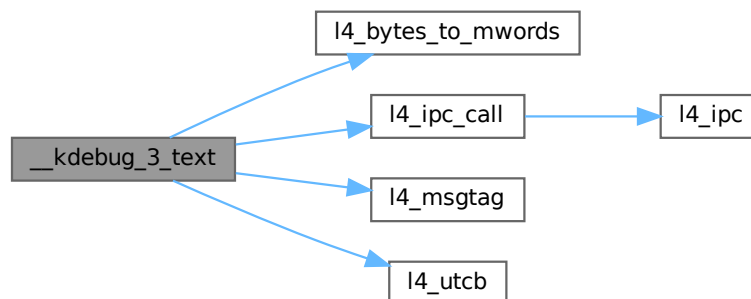
<i>Message</i>	tag returned from the IPC on the base debugger capability.
----------------	------------------------------------------------------------

Definition at line 139 of file [kdebug.h](#).

References [L4\\_BASE\\_DEBUGGER\\_CAP](#), [l4\\_bytes\\_to\\_mwords\(\)](#), [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_NOTHROW](#), [L4\\_PROTO\\_DEBUGGER](#), [l4\\_utcb\(\)](#), and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [fiasco\\_tbuf\\_log\\_3val\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 17.535.3.2 `__kdebug_op()`

```
l4_msgtag_t __kdebug_op (
    unsigned op) [inline]
```

Invoke a nullary operation on the base debugger capability.

#### Parameters

<i>op</i>	Nullary operation code from <a href="#">l4_kdebug_ops_t</a> or a value above 0x200 used by the kernel trace buffer implementation ( <a href="#">__ktrace-impl.h</a> ).
-----------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------

#### Return values

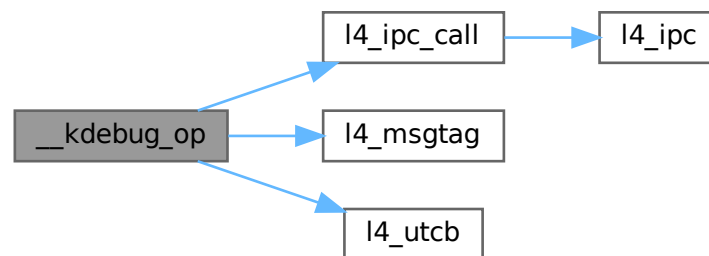
<i>Message</i>	tag returned from the IPC on the base debugger capability.
----------------	------------------------------------------------------------

Definition at line 68 of file [kdebug.h](#).

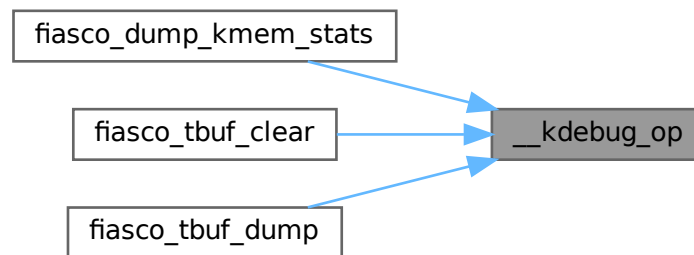
References [L4\\_BASE\\_DEBUGGER\\_CAP](#), [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_NOTHROW](#), [L4\\_PROTO\\_DEBUGGER](#), [l4\\_utcb\(\)](#), and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [fiasco\\_dump\\_kmem\\_stats\(\)](#), [fiasco\\_tbuf\\_clear\(\)](#), and [fiasco\\_tbuf\\_dump\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 17.535.3.3 \_\_kdebug\_op\_1()

```

l4_msgtag_t __kdebug_op_1 (
    unsigned op,
    l4_mword_t val)  [inline]
  
```

Invoke an unary operation on the base debugger capability.

#### Parameters

<i>op</i>	Unary operation code from <a href="#">l4_kdebug_ops_t</a> or a value above 0x200 used by the kernel trace buffer implementation ( <a href="#">__ktrace-impl.h</a> ).
<i>val</i>	Machine word argument to the unary operation.

#### Return values

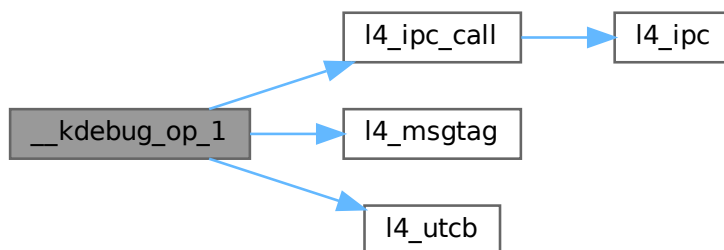
<i>Message</i>	tag returned from the IPC on the base debugger capability.
----------------	------------------------------------------------------------

Definition at line 176 of file [kdebug.h](#).

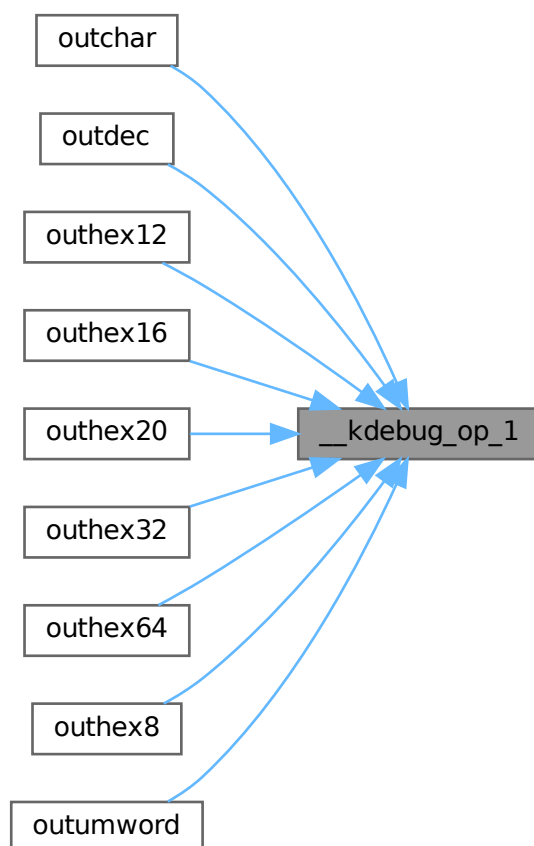
References [L4\\_BASE\\_DEBUGGER\\_CAP](#), [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_NOTHROW](#), [L4\\_PROTO\\_DEBUGGER](#), [l4\\_utcb\(\)](#), and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [outchar\(\)](#), [outdec\(\)](#), [outhex12\(\)](#), [outhex16\(\)](#), [outhex20\(\)](#), [outhex32\(\)](#), [outhex64\(\)](#), [outhex8\(\)](#), and [outumword\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



17.535.3.4 `__kdebug_text()`

```
l4_msgtag_t __kdebug_text (
    unsigned op,
    char const * text,
    unsigned len) [inline]
```

Invoke a text output operation on the base debugger capability.

**Parameters**

<i>op</i>	Text output operation code from <a href="#">l4_kdebug_ops_t</a> or a value above 0x200 used by the kernel trace buffer implementation ( <a href="#">__ktrace-impl.h</a> ).
<i>text</i>	Output string.
<i>len</i>	Length of the output string. The maximum length is limited to <a href="#">L4_UTCB_GENERIC_DATA_SIZE</a> - 2 machine words. Output strings longer than this limit will be cropped.

**Return values**

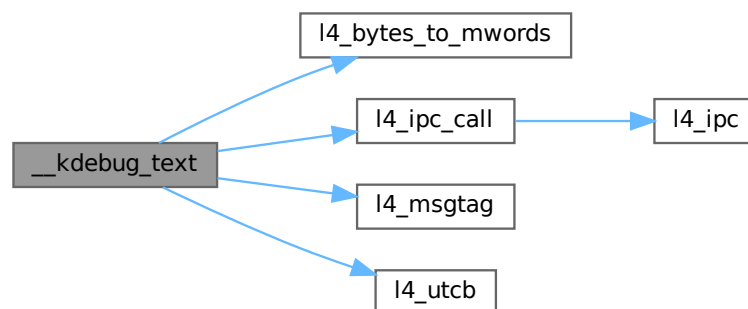
<i>Message</i>	tag returned from the IPC on the base debugger capability.
----------------	------------------------------------------------------------

Definition at line 98 of file [kdebug.h](#).

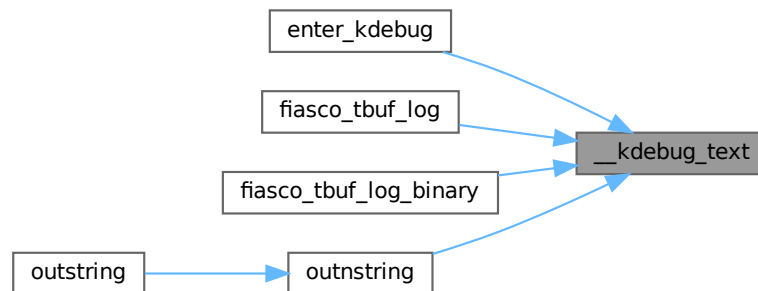
References [L4\\_BASE\\_DEBUGGER\\_CAP](#), [l4\\_bytes\\_to\\_mwords\(\)](#), [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_NOTHROW](#), [L4\\_PROTO\\_DEBUGGER](#), [l4\\_utcb\(\)](#), and [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [enter\\_kdebug\(\)](#), [fiasco\\_tbuf\\_log\(\)](#), [fiasco\\_tbuf\\_log\\_binary\(\)](#), and [outnstring\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 17.535.3.5 enter\_kdebug()

```
void enter_kdebug (
    char const * text) [inline]
```

Enter the kernel debugger.

#### Parameters

<i>text</i>	Optional message displayed by the kernel debugger when entered.
-------------	-----------------------------------------------------------------

Enter the kernel debugger, if configured. An optional message can be passed to the kernel debugger which is printed upon the entering of the debugger.

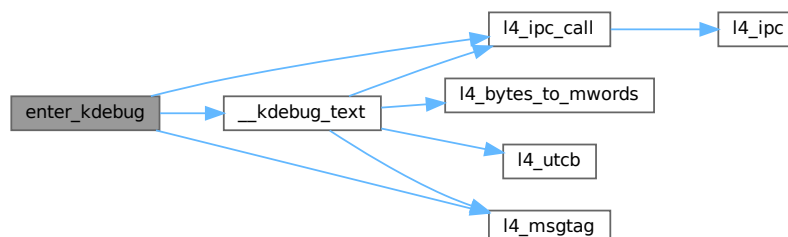
#### Examples

[examples/sys/singlestep/main.c](#).

Definition at line 204 of file [kdebug.h](#).

References [\\_\\_kdebug\\_text\(\)](#), [L4\\_BASE\\_DEBUGGER\\_CAP](#), [L4\\_INLINE](#), [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_msgtag\(\)](#), [L4\\_NOTHROW](#), and [L4\\_PROTO\\_DEBUGGER](#).

Here is the call graph for this function:





### 17.535.3.6 outchar()

```
void outchar (  
    char c) [inline]
```

Output a single character via the kernel debugger.

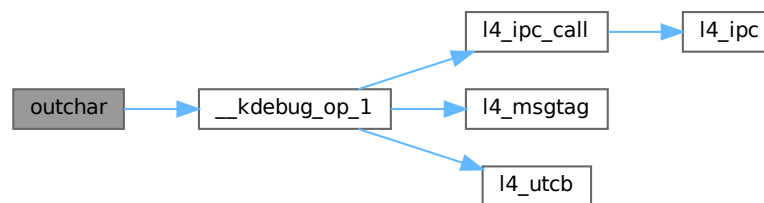
#### Parameters

<i>c</i>	Output character.
----------	-------------------

Definition at line 245 of file [kdebug.h](#).

References [\\_\\_kdebug\\_op\\_1\(\)](#), and [L4\\_INLINE](#).

Here is the call graph for this function:



### 17.535.3.7 outdec()

```
void outdec (  
    l4_mword_t number) [inline]
```

Output a decimal unsigned machine word via the kernel debugger.

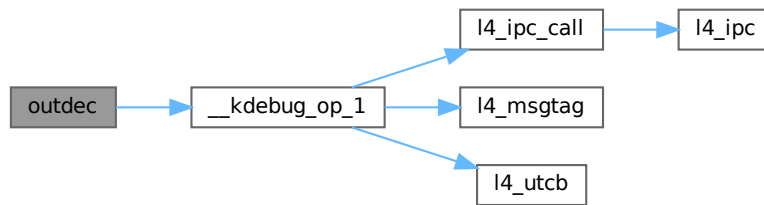
#### Parameters

<i>number</i>	Output machine word.
---------------	----------------------

Definition at line 334 of file [kdebug.h](#).

References [\\_\\_kdebug\\_op\\_1\(\)](#), and [L4\\_INLINE](#).

Here is the call graph for this function:



### 17.535.3.8 outhex12()

```
void outhex12 (
    14_uint16_t number) [inline]
```

Output a 12-bit unsigned hexadecimal number via the kernel debugger.

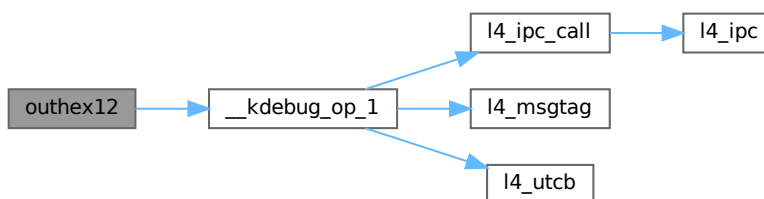
#### Parameters

<i>number</i>	Output 12-bit number. Only the 12 LSB bits are used.
---------------	------------------------------------------------------

Definition at line 314 of file [kdebug.h](#).

References [\\_\\_kdebug\\_op\\_1\(\)](#), and [L4\\_INLINE](#).

Here is the call graph for this function:



### 17.535.3.9 outhex16()

```
void outhex16 (
    14_uint16_t number) [inline]
```

Output a 16-bit unsigned hexadecimal number via the kernel debugger.

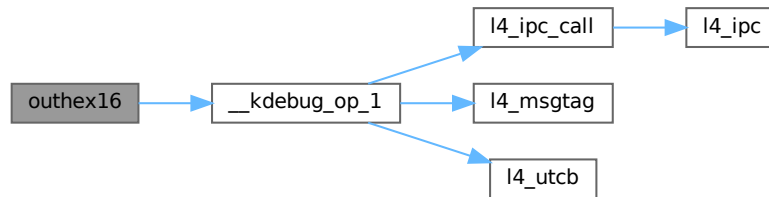
#### Parameters

<i>number</i>	Output 16-bit number.
---------------	-----------------------

Definition at line 304 of file [kdebug.h](#).

References [\\_\\_kdebug\\_op\\_1\(\)](#), and [L4\\_INLINE](#).

Here is the call graph for this function:



### 17.535.3.10 outhex20()

```
void outhex20 (
    14_uint32_t number) [inline]
```

Output a 20-bit unsigned hexadecimal number via the kernel debugger.

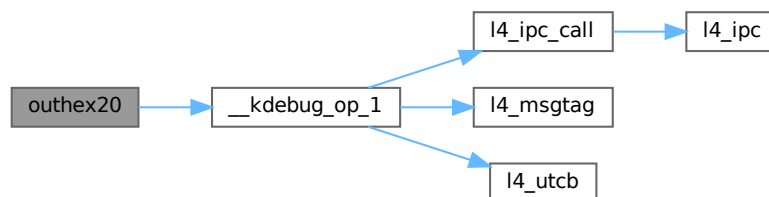
#### Parameters

<i>number</i>	Output 20-bit number. Only the 20 LSB bits are used.
---------------	------------------------------------------------------

Definition at line 294 of file [kdebug.h](#).

References [\\_\\_kdebug\\_op\\_1\(\)](#), and [L4\\_INLINE](#).

Here is the call graph for this function:



**17.535.3.11 outhex32()**

```
void outhex32 (  
    14_uint32_t number)  [inline]
```

Output a 32-bit unsigned hexadecimal number via the kernel debugger.

**Parameters**

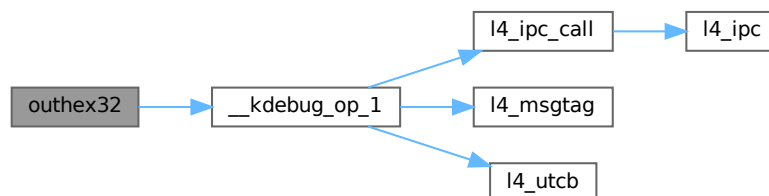
---

<i>number</i>	Output 32-bit number.
---------------	-----------------------

Definition at line 284 of file [kdebug.h](#).

References [\\_\\_kdebug\\_op\\_1\(\)](#), and [L4\\_INLINE](#).

Here is the call graph for this function:



### 17.535.3.12 outhex64()

```
void outhex64 (  
    l4_uint64_t number) [inline]
```

Output a 64-bit unsigned hexadecimal number via the kernel debugger.

#### Parameters

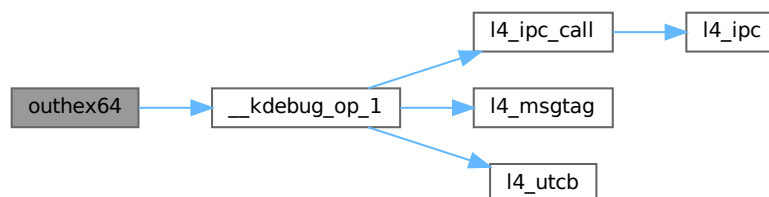
<i>number</i>	Output 64-bit number.
---------------	-----------------------

The two 32-bit halves are printed non-atomically.

Definition at line 273 of file [kdebug.h](#).

References [\\_\\_kdebug\\_op\\_1\(\)](#), and [L4\\_INLINE](#).

Here is the call graph for this function:



**17.535.3.13 outhex8()**

```
void outhex8 (
    l4_uint8_t number) [inline]
```

Output an 8-bit unsigned hexadecimal number via the kernel debugger.

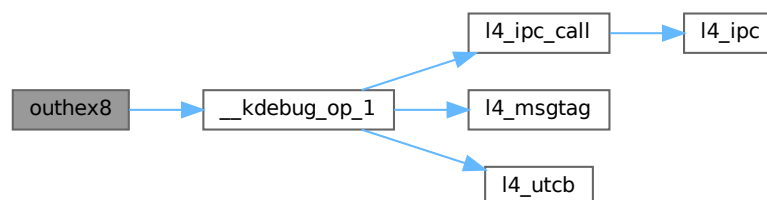
**Parameters**

<i>number</i>	Output 8-bit number.
---------------	----------------------

Definition at line 324 of file [kdebug.h](#).

References [\\_\\_kdebug\\_op\\_1\(\)](#), and [L4\\_INLINE](#).

Here is the call graph for this function:

**17.535.3.14 outnstring()**

```
void outnstring (
    char const * text,
    unsigned len) [inline]
```

Output a fixed-length string via the kernel debugger.

**Parameters**

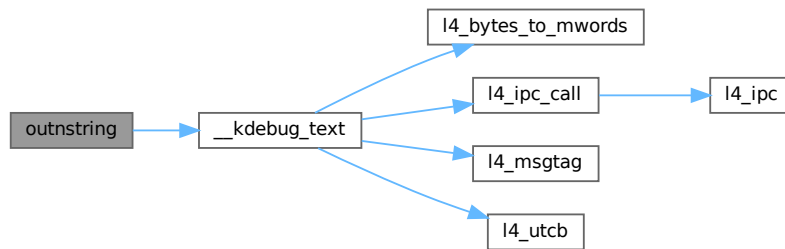
<i>text</i>	Beginning of the output string.
<i>len</i>	Length of the output string. The maximum length is limited to <a href="#">L4_UTCB_GENERIC_DATA_SIZE</a> - 2 machine words. Output strings longer than this limit will be cropped.

Definition at line 226 of file [kdebug.h](#).

References [\\_\\_kdebug\\_text\(\)](#), and [L4\\_INLINE](#).

Referenced by [outstring\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 17.535.3.15 outstring()

```
void outstring (
    char const * text) [inline]
```

Output a string via the kernel debugger.

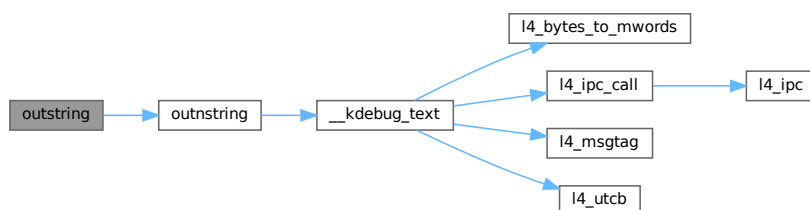
#### Parameters

<i>text</i>	Beginning of the output string. The maximum length of the output string is limited to <a href="#">L4_UTCB_GENERIC_DATA_SIZE</a> - 2 machine words. Output strings longer than this limit will be cropped.
-------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Definition at line 237 of file [kdebug.h](#).

References [L4\\_INLINE](#), and [outnstring\(\)](#).

Here is the call graph for this function:



### 17.535.3.16 outumword()

```
void outumword (
    l4_umword_t number) [inline]
```

Output a hexadecimal unsigned machine word via the kernel debugger.

#### Parameters

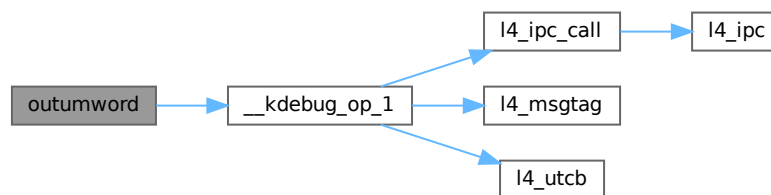
<i>number</i>	Output machine word.
---------------	----------------------

If the machine word is 64 bits long, it is printed non-atomically as two 32-bit numbers.

Definition at line 258 of file [kdebug.h](#).

References [\\_\\_kdebug\\_op\\_1\(\)](#), and [L4\\_INLINE](#).

Here is the call graph for this function:



## 17.536 kdebug.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #ifndef __KDEBUG_H__
00011 #define __KDEBUG_H__
00012
00013 #include <l4/sys/compiler.h>
00014 #include <l4/sys/consts.h>
00015 #include <l4/sys/ipc.h>
00016
00017 L4_INLINE void
00018 enter_kdebug(char const *text) L4_NOTHROW;
00019
00020 enum l4_kdebug_group_t
00021 {
00022     L4_KDEBUG_GROUP_JDB    = 0x000,
00023     L4_KDEBUG_GROUP_KOBJ   = 0x100, // see __kernel_object_impl.h
00024     L4_KDEBUG_GROUP_TRACE  = 0x200, // see __ktrace-impl.h
00025     L4_KDEBUG_GROUP_COV    = 0x400,
00026     L4_KDEBUG_GROUP_DUMP   = 0x500, // see kdump.h
00027 }
```



```

00038 };
00039
00044 enum l4_kdebug_ops_t
00045 {
00046     L4_KDEBUG_ENTER      = L4_KDEBUG_GROUP_JDB + 0,
00047     L4_KDEBUG_OUTCHAR    = L4_KDEBUG_GROUP_JDB + 1,
00048     L4_KDEBUG_OUTNSTRING = L4_KDEBUG_GROUP_JDB + 2,
00049     L4_KDEBUG_OUTHEX32   = L4_KDEBUG_GROUP_JDB + 3,
00050     L4_KDEBUG_OUTHEX20   = L4_KDEBUG_GROUP_JDB + 4,
00051     L4_KDEBUG_OUTHEX16   = L4_KDEBUG_GROUP_JDB + 5,
00052     L4_KDEBUG_OUTHEX12   = L4_KDEBUG_GROUP_JDB + 6,
00053     L4_KDEBUG_OUTHEX8    = L4_KDEBUG_GROUP_JDB + 7,
00054     L4_KDEBUG_OUTDEC     = L4_KDEBUG_GROUP_JDB + 8,
00055 };
00056
00057
00067 L4_INLINE l4_msgtag_t
00068 __kdebug_op(unsigned op) L4_NOTHROW
00069 {
00070     l4_msgtag_t res;
00071     l4_utcb_t *u = l4_utcb();
00072     l4_msg_regs_t *mr = l4_utcb_mr_u(u);
00073     l4_umword_t mr0 = mr->mr[0];
00074
00075     mr->mr[0] = op;
00076     res = l4_ipc_call(L4_BASE_DEBUGGER_CAP, u,
00077                     l4_msgtag(L4_PROTO_DEBUGGER, 1, 0, 0),
00078                     L4_IPC_NEVER);
00079     mr->mr[0] = mr0;
00080     return res;
00081 }
00082
00097 L4_INLINE l4_msgtag_t
00098 __kdebug_text(unsigned op, char const *text, unsigned len) L4_NOTHROW
00099 {
00100     l4_msg_regs_t store;
00101     l4_msgtag_t res;
00102     l4_utcb_t *u = l4_utcb();
00103     l4_msg_regs_t *mr = l4_utcb_mr_u(u);
00104
00105     if (len > (sizeof(store) - (2 * sizeof(l4_umword_t))))
00106         len = sizeof(store) - (2 * sizeof(l4_umword_t));
00107
00108     __builtin_memcpy(&store, mr, sizeof(store));
00109     mr->mr[0] = op;
00110     mr->mr[1] = len;
00111     __builtin_memcpy(&mr->mr[2], text, len);
00112     res = l4_ipc_call(L4_BASE_DEBUGGER_CAP, u,
00113                     l4_msgtag(L4_PROTO_DEBUGGER,
00114                             l4_bytes_to_mwords(len) + 2, 0, 0),
00115                     L4_IPC_NEVER);
00116     __builtin_memcpy(mr, &store, sizeof(*mr));
00117     return res;
00118 }
00119
00138 L4_INLINE l4_msgtag_t
00139 __kdebug_3_text(unsigned op, char const *text, unsigned len,
00140                l4_umword_t v1, l4_umword_t v2, l4_umword_t v3) L4_NOTHROW
00141 {
00142     l4_msg_regs_t store;
00143     l4_msgtag_t res;
00144     l4_utcb_t *u = l4_utcb();
00145     l4_msg_regs_t *mr = l4_utcb_mr_u(u);
00146
00147     if (len > (sizeof(store) - (5 * sizeof(l4_umword_t))))
00148         len = sizeof(store) - (5 * sizeof(l4_umword_t));
00149
00150     __builtin_memcpy(&store, mr, sizeof(store));
00151     mr->mr[0] = op;
00152     mr->mr[1] = v1;
00153     mr->mr[2] = v2;
00154     mr->mr[3] = v3;
00155     mr->mr[4] = len;
00156     __builtin_memcpy(&mr->mr[5], text, len);
00157     res = l4_ipc_call(L4_BASE_DEBUGGER_CAP, u,
00158                     l4_msgtag(L4_PROTO_DEBUGGER,
00159                             l4_bytes_to_mwords(len) + 5, 0, 0),
00160                     L4_IPC_NEVER);
00161     __builtin_memcpy(mr, &store, sizeof(*mr));
00162     return res;
00163 }
00164
00175 L4_INLINE l4_msgtag_t
00176 __kdebug_op_l(unsigned op, l4_mword_t val) L4_NOTHROW
00177 {
00178     l4_umword_t m[2];
00179     l4_msgtag_t res;

```

```

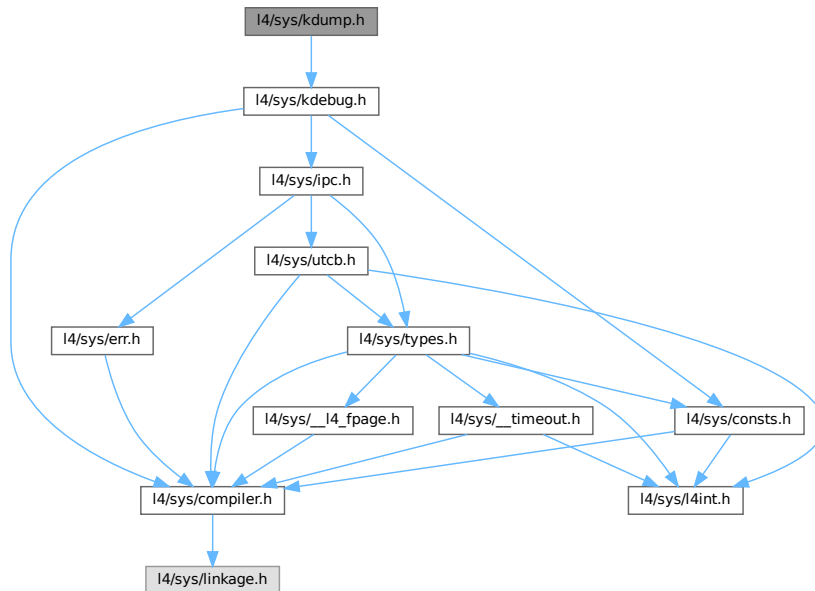
00180     l4_utcb_t *u = l4_utcb();
00181     l4_msg_regs_t *mr = l4_utcb_mr_u(u);
00182
00183     m[0] = mr->mr[0];
00184     m[1] = mr->mr[1];
00185     mr->mr[0] = op;
00186     mr->mr[1] = val;
00187     res = l4_ipc_call(L4_BASE_DEBUGGER_CAP, u,
00188                     l4_msgtag(L4_PROTO_DEBUGGER, 2, 0, 0),
00189                     L4_IPC_NEVER);
00190     mr->mr[0] = m[0];
00191     mr->mr[1] = m[1];
00192     return res;
00193 }
00194
00204 L4_INLINE void enter_kdebug(char const *text) L4_NOTHROW
00205 {
00206     /* special case, enter without any text and use of the UTCB */
00207     if (!text)
00208     {
00209         l4_ipc_call(L4_BASE_DEBUGGER_CAP, 0,
00210                     l4_msgtag(L4_PROTO_DEBUGGER, 0, 0, 0),
00211                     L4_IPC_NEVER);
00212         return;
00213     }
00214     __kdebug_text(L4_KDEBUG_ENTER, text, __builtin_strlen(text));
00215 }
00216
00226 L4_INLINE void outnstring(char const *text, unsigned len)
00227 { __kdebug_text(L4_KDEBUG_OUTNSTRING, text, len); }
00228
00237 L4_INLINE void outstring(char const *text)
00238 { outnstring(text, __builtin_strlen(text)); }
00239
00245 L4_INLINE void outchar(char c)
00246 {
00247     __kdebug_op_1(L4_KDEBUG_OUTCHAR, c);
00248 }
00249
00258 L4_INLINE void outumword(l4_umword_t number)
00259 {
00260     if (sizeof(l4_umword_t) == sizeof(l4_uint64_t))
00261         __kdebug_op_1(L4_KDEBUG_OUTHEX32, (l4_uint64_t)number >> 32);
00262     __kdebug_op_1(L4_KDEBUG_OUTHEX32, number);
00263 }
00264
00273 L4_INLINE void outhex64(l4_uint64_t number)
00274 {
00275     __kdebug_op_1(L4_KDEBUG_OUTHEX32, number >> 32);
00276     __kdebug_op_1(L4_KDEBUG_OUTHEX32, number);
00277 }
00278
00284 L4_INLINE void outhex32(l4_uint32_t number)
00285 {
00286     __kdebug_op_1(L4_KDEBUG_OUTHEX32, number);
00287 }
00288
00294 L4_INLINE void outhex20(l4_uint32_t number)
00295 {
00296     __kdebug_op_1(L4_KDEBUG_OUTHEX20, number);
00297 }
00298
00304 L4_INLINE void outhex16(l4_uint16_t number)
00305 {
00306     __kdebug_op_1(L4_KDEBUG_OUTHEX16, number);
00307 }
00308
00314 L4_INLINE void outhex12(l4_uint16_t number)
00315 {
00316     __kdebug_op_1(L4_KDEBUG_OUTHEX12, number);
00317 }
00318
00324 L4_INLINE void outhex8(l4_uint8_t number)
00325 {
00326     __kdebug_op_1(L4_KDEBUG_OUTHEX8, number);
00327 }
00328
00334 L4_INLINE void outdec(l4_mword_t number)
00335 {
00336     __kdebug_op_1(L4_KDEBUG_OUTDEC, number);
00337 }
00338
00339 #endif // __KDEBUG_H__

```

## 17.537 l4/sys/kdump.h File Reference

Functionality for dumping kernel information.

```
#include <l4/sys/kdebug.h>
Include dependency graph for kdump.h:
```



### Functions

- long [fiasco\\_dump\\_kmem\\_stats](#) (void)  
*Dump kernel memory statistics on console.*

### 17.537.1 Detailed Description

Functionality for dumping kernel information.

Definition in file [kdump.h](#).

### 17.537.2 Function Documentation

#### 17.537.2.1 fiasco\_dump\_kmem\_stats()

```
long fiasco_dump_kmem_stats (
    void ) [inline]
```

Dump kernel memory statistics on console.

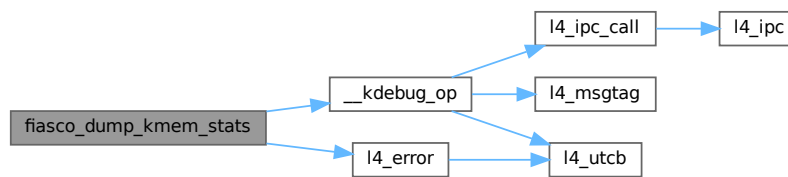
#### Return values

0	Success.
-L4_ENOSYS	Not implemented by kernel.

Definition at line 38 of file [kdump.h](#).

References [\\_\\_kdebug\\_op\(\)](#), and [l4\\_error\(\)](#).

Here is the call graph for this function:



## 17.538 kdump.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (C) 2024 Kernkonzept GmbH.
00003  * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010
00014
00025
00026 #include <l4/sys/kdebug.h>
00027
00034 L4_INLINE long
00035 fiasco_dump_kmem_stats(void);
00036
00037 L4_INLINE long
00038 fiasco_dump_kmem_stats(void)
00039 {
00040     enum { DUMP_KMEM_STATS = L4_KDEBUG_GROUP_DUMP + 0x00 };
00041     return l4_error(__kdebug_op(DUMP_KMEM_STATS));
00042 }

```

## 17.539 l4/sys/kernel\_object.h File Reference

Kernel object system calls.

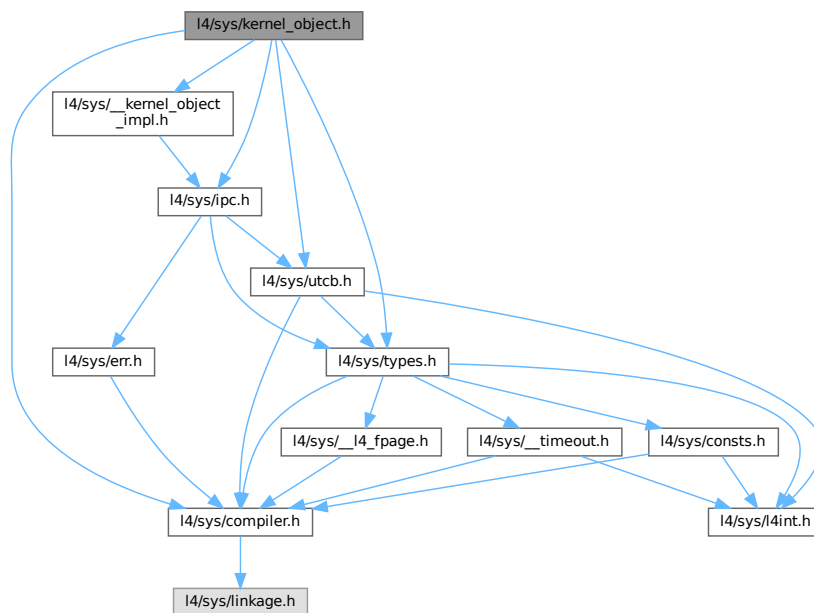
```

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>
#include <l4/sys/utcb.h>
#include <l4/sys/__kernel_object_impl.h>

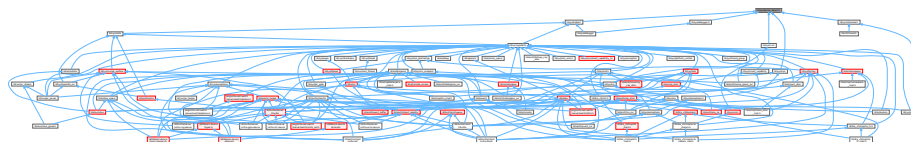
```

```
#include <l4/sys/ipc.h>
```

Include dependency graph for kernel\_object.h:



This graph shows which files directly or indirectly include this file:



## 17.539.1 Detailed Description

Kernel object system calls.

Definition in file [kernel\\_object.h](#).

## 17.540 kernel\_object.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *      Björn Döbel <doebel@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #ifndef __L4SYS_KERNEL_OBJECT_H__
00014 #define __L4SYS_KERNEL_OBJECT_H__

```

```

00015
00016 #include <l4/sys/types.h>
00017 #include <l4/sys/compiler.h>
00018 #include <l4/sys/utcb.h>
00019
00027
00038 L4_INLINE l4_msgtag_t
00039 l4_invoke_debugger(l4_cap_idx_t obj, l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW;
00040
00041
00042 /*****
00043  * Implementation
00044  *****/
00045
00046 #include <l4/sys/__kernel_object_impl.h>
00047 #include <l4/sys/ipc.h>
00048
00049 enum L4_kobject_op {
00050     L4_KOBJECT_OP_DEC_REFCNT = 0,
00051     L4_KOBJECT_OP_REGISTER_IRQ,
00052 };
00053
00054 L4_INLINE l4_msgtag_t
00055 l4_kobject_dec_refcnt_u(l4_cap_idx_t obj, l4_mword_t diff, l4_utcb_t *u) L4_NOTHROW;
00056
00057 L4_INLINE l4_msgtag_t
00058 l4_kobject_dec_refcnt(l4_cap_idx_t obj, l4_mword_t diff) L4_NOTHROW;
00059
00060 L4_INLINE l4_msgtag_t
00061 l4_kobject_dec_refcnt_u(l4_cap_idx_t obj, l4_mword_t diff, l4_utcb_t *u) L4_NOTHROW
00062 {
00063     l4_msg_regs_t *m = l4_utcb_mr_u(u);
00064     m->mr[0] = L4_KOBJECT_OP_DEC_REFCNT;
00065     m->mr[1] = diff;
00066     return l4_ipc_call(obj, u, l4_msgtag(L4_PROTO_KOBJECT, 2, 0, 0), L4_IPC_NEVER);
00067 }
00068
00069 L4_INLINE l4_msgtag_t
00070 l4_kobject_dec_refcnt(l4_cap_idx_t obj, l4_mword_t diff) L4_NOTHROW
00071 {
00072     return l4_kobject_dec_refcnt_u(obj, diff, l4_utcb());
00073 }
00074
00075 #endif /* ! __L4SYS__KERNEL_OBJECT_H__ */

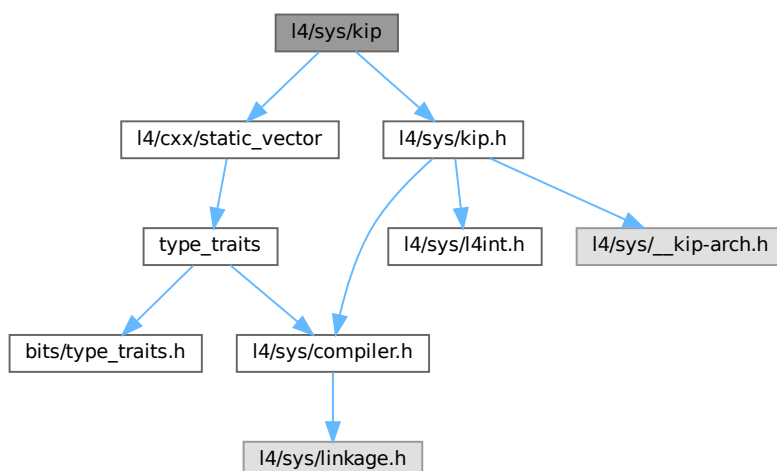
```

## 17.541 l4/sys/kip File Reference

```
#include <l4/cxx/static_vector>
```

```
#include <l4/sys/kip.h>
```

Include dependency graph for kip:



## Data Structures

- class [L4::Kip::Mem\\_desc](#)

*Memory descriptors stored in the kernel interface page.*

## Namespaces

- namespace [L4](#)

*[L4](#) low-level kernel interface.*

## 17.541.1 Detailed Description

L4::Kip class, memory descriptors.

### Author

Alexander Warg [alexander.warg@os.inf.tu-dresden.de](mailto:alexander.warg@os.inf.tu-dresden.de)

Definition in file [kip](#).

## 17.542 kip

[Go to the documentation of this file.](#)

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00010 /*
00011  * (c) 2008-2009 Author(s)
00012  *      economic rights: Technische Universität Dresden (Germany)
00013  *
00014  * License: see LICENSE.spdx (in this directory or the directories above)
00015  */
00016 #pragma once
00017
00018 #include <l4/cxx/static_vector>
00019 #include <l4/sys/kip.h>
00020
00021 /* C++ version of memory descriptors */
00022
00031
00032 namespace L4
00033 {
00034     namespace Kip
00035     {
00042         class Mem_desc
00043         {
00044         public:
00048             enum Mem_type
00049             {
00050                 Undefined      = 0x0,
00051                 Conventional   = 0x1,
00052                 Reserved       = 0x2,
00053                 Dedicated      = 0x3,
00054                 Shared         = 0x4,
00055
00056                 Info           = 0xd,
00057                 Bootloader     = 0xe,
00058                 Arch           = 0xf
00059             };
00060
00064             enum Info_sub_type
00065             {
00066                 Info_acpi_rsdp = 0,
00067
00068                 Reserved_kernel = 0,
00069                 Reserved_heap   = 1,
00070                 Reserved_mmio   = 2,
```

```

00071     };
00072
00076     enum Arch_sub_type_common
00077     {
00078         Arch_acpi_tables = 3,
00079         Arch_acpi_nvs    = 4,
00080     };
00081
00082 private:
00083     unsigned long _l, _h;
00084
00085 public:
00093     static Mem_desc *first(l4_kernel_info_t *kip) noexcept
00094     {
00095         return reinterpret_cast<Mem_desc *>(reinterpret_cast<char *>(kip) + kip->mem_descs);
00096     }
00097
00098     static Mem_desc const *first(l4_kernel_info_t const *kip) noexcept
00099     {
00100         char const *addr = reinterpret_cast<char const *>(kip) + kip->mem_descs;
00101         return reinterpret_cast<Mem_desc const *>(addr);
00102     }
00103
00111     static unsigned long count(l4_kernel_info_t const *kip) noexcept
00112     {
00113         return kip->mem_descs_num;
00114     }
00115
00122     static void count(l4_kernel_info_t *kip, unsigned count) noexcept
00123     {
00124         kip->mem_descs_num = count;
00125     }
00126
00132     static inline cxx::static_vector<Mem_desc const> all(l4_kernel_info_t const *kip)
00133     {
00134         return cxx::static_vector<Mem_desc const>(Mem_desc::first(kip),
00135                                                    Mem_desc::count(kip));
00136     }
00137
00143     static inline cxx::static_vector<Mem_desc> all(l4_kernel_info_t *kip)
00144     {
00145         return cxx::static_vector<Mem_desc>(Mem_desc::first(kip),
00146                                              Mem_desc::count(kip));
00147     }
00148
00162     Mem_desc(unsigned long start, unsigned long end,
00163              Mem_type t, unsigned char st = 0, bool virt = false,
00164              bool eager = false) noexcept
00165     : _l((start & ~0x3ffUL) | (t & 0x0f) | ((st << 4) & 0x0f0)
00166         | (virt ? 0x0200 : 0x0) | (eager ? 0x100 : 0x0)), _h(end | 0x3ffUL)
00167     {}
00168
00174     unsigned long start() const noexcept { return _l & ~0x3ffUL; }
00175
00181     unsigned long end() const noexcept { return _h | 0x3ffUL; }
00182
00188     unsigned long size() const noexcept { return end() + 1 - start(); }
00189
00195     Mem_type type() const noexcept
00196     {
00197         return static_cast<Mem_type>(_l & 0x0f);
00198     }
00199
00205     unsigned char sub_type() const noexcept { return (_l >> 4) & 0x0f; }
00206
00213     unsigned is_virtual() const noexcept { return _l & 0x200; }
00214
00219     unsigned eager_map() const noexcept { return _l & 0x100; }
00220
00233     void set(unsigned long start, unsigned long end,
00234             Mem_type t, unsigned char st = 0, bool virt = false,
00235             bool eager = false) noexcept
00236     {
00237         _l = (start & ~0x3ffUL) | (t & 0x0f) | ((st << 4) & 0x0f0)
00238             | (virt ? 0x0200 : 0x0) | (eager ? 0x0100 : 0x0);
00239
00240         _h = end | 0x3ffUL;
00241     }
00242
00243 };
00244 };
00245 };

```



## 17.543 kobject

```

00001 // vi:set ft=c++: -*- Mode: C++ -*-
00002 /* \file
00003  * Kobject C++ interface.
00004  */
00005 /*
00006  * Copyright (C) 2015-2017, 2019, 2021, 2023-2024 Kernkonzept GmbH.
00007  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00013 #include "kernel_object.h"
00014 #include "types.h"
00015 #include "__typeinfo.h"
00016
00017 namespace L4 {
00018
00036 class L4_EXPORT Kobject
00037 {
00038 private:
00039     Kobject();
00040     Kobject(Kobject const &);
00041     Kobject &operator = (Kobject const &);
00042
00043     template<typename T> friend struct Kobject_typeid;
00044
00045 protected:
00046     typedef Typeid::Iface<L4_PROTO_META, Kobject> __Iface;
00047     typedef Typeid::Iface_list<__Iface> __Iface_list;
00048
00055     struct __Kobject_typeid
00056     {
00057         typedef Type_info::Demand_t<> Demand;
00058         static Type_info const _m;
00059     };
00060
00069     l4_cap_idx_t cap() const noexcept { return _c(); }
00070
00071 private:
00072
00077     l4_cap_idx_t _c() const noexcept
00078     { return reinterpret_cast<l4_cap_idx_t>(this) & L4_CAP_MASK; }
00079
00080 public:
00100     l4_msgtag_t dec_refcnt(l4_mword_t diff, l4_utcb_t *utcb = l4_utcb())
00101     { return l4_kobject_dec_refcnt_u(cap(), diff, utcb); }
00102 };
00103
00104 template<typename Derived, long PROTO = L4::PROTO_ANY,
00105         typename S_DEMAND = Type_info::Demand_t<> >
00106 struct Kobject_0t : Kobject_t<Derived, L4::Kobject, PROTO, S_DEMAND> {};
00107
00108 }
00109

```

## 17.544 l4/sys/ktrace.h File Reference

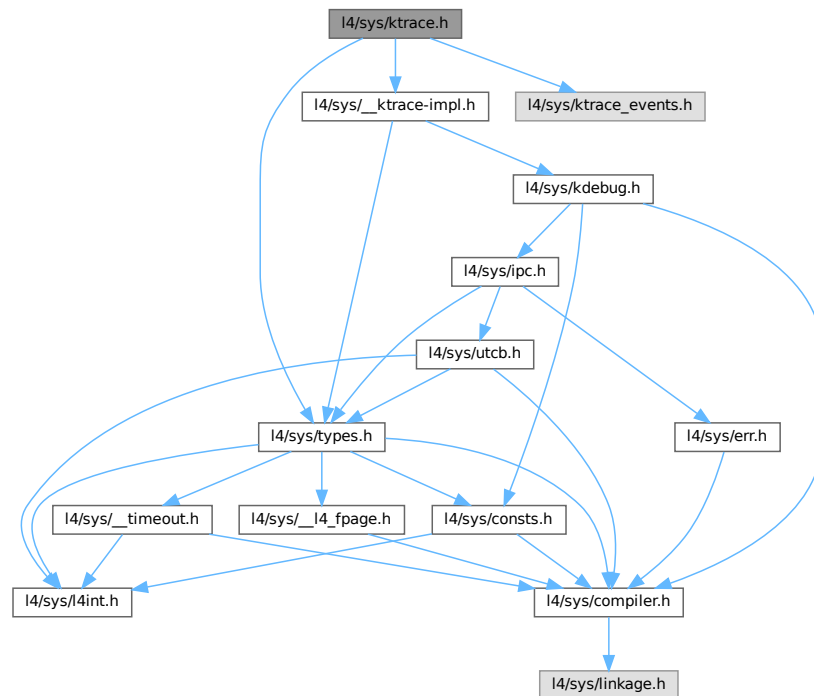
[L4](#) kernel event tracing.

```

#include <l4/sys/types.h>
#include <l4/sys/ktrace_events.h>
#include <l4/sys/__ktrace-impl.h>

```

Include dependency graph for `ktrace.h`:



## Functions

- `l4_umword_t fiasco_tbuf_log` (const char \*text)  
Create new trace-buffer entry with describing <text>.
- `l4_umword_t fiasco_tbuf_log_3val` (const char \*text, `l4_umword_t` v1, `l4_umword_t` v2, `l4_umword_t` v3)  
Create new trace-buffer entry with describing <text> and three additional values.
- `l4_umword_t fiasco_tbuf_log_binary` (const unsigned char \*data)  
Create new trace-buffer entry with binary data.
- void `fiasco_tbuf_clear` (void)  
Clear trace-buffer.
- void `fiasco_tbuf_dump` (void)  
Dump trace-buffer to kernel console.

### 17.544.1 Detailed Description

[L4](#) kernel event tracing.

Definition in file [ktrace.h](#).

## 17.545 ktrace.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *          Björn Döbel <doebel@os.inf.tu-dresden.de>,
00009  *          Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *          2015 Adam Lackorzynski <adam@l4re.org>
00011  *          economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * License: see LICENSE.spdx (in this directory or the directories above)
00014  */
00015 /*****
00016 #ifndef __L4_KTRACE_H__
00017 #define __L4_KTRACE_H__
00018
00019 #include <l4/sys/types.h>
00020 #include <l4/sys/ktrace_events.h>
00021
00033
00041 L4_INLINE l4_umword_t
00042 fiasco_tbuf_log(const char *text);
00043
00055 L4_INLINE l4_umword_t
00056 fiasco_tbuf_log_3val(const char *text, l4_umword_t v1, l4_umword_t v2, l4_umword_t v3);
00057
00065 L4_INLINE l4_umword_t
00066 fiasco_tbuf_log_binary(const unsigned char *data);
00067
00072 L4_INLINE void
00073 fiasco_tbuf_clear(void);
00074
00079 L4_INLINE void
00080 fiasco_tbuf_dump(void);
00081
00082 #include <l4/sys/__ktrace-impl.h>
00083
00084 #endif

```

## 17.546 amd64/l4/sys/l4int.h File Reference

Fixed sized integer types, AMD64 version.

### Macros

- `#define L4_MWORD_BITS 64`  
*Size of machine words in bits.*

### Typedefs

- `typedef unsigned long l4_size_t`  
*Unsigned size type.*
- `typedef signed long l4_ssize_t`  
*Signed size type.*

### 17.546.1 Detailed Description

Fixed sized integer types, AMD64 version.

Definition in file [l4int.h](#).

## 17.547 l4int.h

[Go to the documentation of this file.](#)

```

00001 /*****
00007 */
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 *      Alexander Warg <warg@os.inf.tu-dresden.de>
00010 *      economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * License: see LICENSE.spdx (in this directory or the directories above)
00013 */
00014 #pragma once
00015
00016 #include_next <l4/sys/l4int.h>
00017
00022
00023 #define L4_MWORD_BITS          64
00024
00025 typedef unsigned long          l4_size_t;
00026 typedef signed long           l4_ssize_t;
00028

```

## 17.548 arm/l4/sys/l4int.h File Reference

Fixed sized integer types, arm version.

### Macros

- `#define L4_MWORD_BITS 32`  
*Size of machine words in bits.*

### Typedefs

- `typedef unsigned int l4_size_t`  
*Unsigned size type.*
- `typedef signed int l4_ssize_t`  
*Signed size type.*

### 17.548.1 Detailed Description

Fixed sized integer types, arm version.

Definition in file [l4int.h](#).

## 17.549 l4int.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009 *      economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * License: see LICENSE.spdx (in this directory or the directories above)
00012 */
00013 #pragma once
00014
00015 #include_next <l4/sys/l4int.h>
00016
00021
00022 #define L4_MWORD_BITS          32
00023
00024 typedef unsigned int          l4_size_t;
00025 typedef signed int           l4_ssize_t;
00027

```

## 17.550 arm64/l4/sys/l4int.h File Reference

Fixed sized integer types, arm version.

### Macros

- `#define L4_MWORD_BITS 64`  
*Size of machine words in bits.*

### Typedefs

- `typedef unsigned long l4_size_t`  
*Unsigned size type.*
- `typedef signed long l4_ssize_t`  
*Signed size type.*

### 17.550.1 Detailed Description

Fixed sized integer types, arm version.

Definition in file [l4int.h](#).

## 17.551 l4int.h

[Go to the documentation of this file.](#)

```

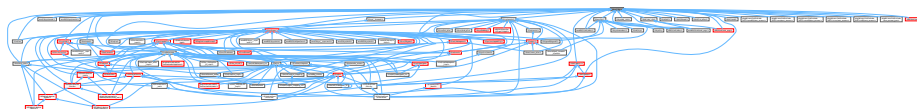
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00015 #include_next <l4/sys/l4int.h>
00016
00021
00022 #define L4_MWORD_BITS          64
00023
00024 typedef unsigned long          l4_size_t;
00025 typedef signed long           l4_ssize_t;
00027

```

## 17.552 l4/sys/l4int.h File Reference

Fixed sized integer types, generic version.

This graph shows which files directly or indirectly include this file:



## Typedefs

- typedef signed char **I4\_int8\_t**  
*Signed 8bit value.*
- typedef unsigned char **I4\_uint8\_t**  
*Unsigned 8bit value.*
- typedef signed short int **I4\_int16\_t**  
*Signed 16bit value.*
- typedef unsigned short int **I4\_uint16\_t**  
*Unsigned 16bit value.*
- typedef signed int **I4\_int32\_t**  
*Signed 32bit value.*
- typedef unsigned int **I4\_uint32\_t**  
*Unsigned 32bit value.*
- typedef signed long long **I4\_int64\_t**  
*Signed 64bit value.*
- typedef unsigned long long **I4\_uint64\_t**  
*Unsigned 64bit value.*
- typedef unsigned long **I4\_addr\_t**  
*Address type.*
- typedef signed long **I4\_mword\_t**  
*Signed machine word.*
- typedef unsigned long **I4\_umword\_t**  
*Unsigned machine word.*
- typedef [I4\\_uint64\\_t](#) **I4\_cpu\_time\_t**  
*CPU clock type.*
- typedef [I4\\_uint64\\_t](#) **I4\_kernel\_clock\_t**  
*Kernel clock type.*

### 17.552.1 Detailed Description

Fixed sized integer types, generic version.

Definition in file [I4int.h](#).

## 17.553 I4int.h

[Go to the documentation of this file.](#)

```

00001
00007
00013 /*
00014  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00015  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00016  *      economic rights: Technische Universität Dresden (Germany)
00017  *
00018  * License: see LICENSE.spdx (in this directory or the directories above)
00019  */
00020 #ifndef __L4_SYS_L4INT_H__
00021 #define __L4_SYS_L4INT_H__
00022
00023 /* fixed sized data types */
00024 typedef signed char      I4_int8_t;
00025 typedef unsigned char    I4_uint8_t;
00026 typedef signed short int I4_int16_t;
00027 typedef unsigned short int I4_uint16_t;
00028 typedef signed int       I4_int32_t;
```

```

00029 typedef unsigned int      l4_uint32_t;
00030 typedef signed long long    l4_int64_t;
00031 typedef unsigned long long  l4_uint64_t;
00032
00033 /* some common data types */
00034 typedef unsigned long       l4_addr_t;
00035
00036
00037 typedef signed long         l4_mword_t;
00040 typedef unsigned long      l4_umword_t;
00047 typedef l4_uint64_t l4_cpu_time_t;
00048
00053 typedef l4_uint64_t l4_kernel_clock_t;
00054
00055 #endif /* !__L4_SYS_L4INT_H__ */

```

## 17.554 x86/l4/sys/l4int.h File Reference

Fixed sized integer types, x86 version.

### Macros

- **#define L4\_MWORD\_BITS 32**  
*Size of machine words in bits.*

### Typedefs

- typedef unsigned int **l4\_size\_t**  
*Unsigned size type.*
- typedef signed int **l4\_ssize\_t**  
*Signed size type.*

### 17.554.1 Detailed Description

Fixed sized integer types, x86 version.

Definition in file [l4int.h](#).

## 17.555 l4int.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00015 #include_next <l4/sys/l4int.h>
00016
00021
00022 #define L4_MWORD_BITS      32
00023
00024 typedef unsigned int      l4_size_t;
00025 typedef signed int        l4_ssize_t;
00027

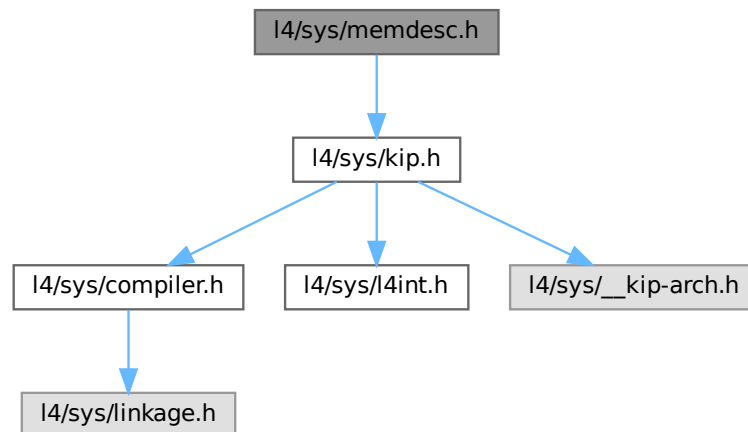
```

## 17.556 l4/sys/memdesc.h File Reference

Memory description functions.

```
#include <l4/sys/kip.h>
```

Include dependency graph for memdesc.h:



### Data Structures

- struct `l4_kernel_info_mem_desc_t`  
*Memory descriptor data structure.*

### Typedefs

- typedef struct `l4_kernel_info_mem_desc_t` `l4_kernel_info_mem_desc_t`  
*Memory descriptor data structure.*

### Enumerations

- enum `l4_mem_type_t` {  
`l4_mem_type_undefined` = 0x0 , `l4_mem_type_conventional` = 0x1 , `l4_mem_type_reserved` = 0x2 ,  
`l4_mem_type_dedicated` = 0x3 ,  
`l4_mem_type_shared` = 0x4 , `l4_mem_type_info` = 0xd , `l4_mem_type_bootloader` = 0xe , `l4_mem_type_archspecific`  
= 0xf }  
*Type of a memory descriptor.*
- enum `l4_mem_info_sub_type_t` { `l4_mem_info_acpi_rsdp` = 0 , `l4_mem_reserved_kernel` = 0 ,  
`l4_mem_reserved_heap` = 1 , `l4_mem_reserved_mmio` = 2 }  
*Memory sub types for l4\_mem\_type\_info descriptors.*
- enum `l4_mem_archspecific_sub_type_common_t` { `l4_mem_archspecific_acpi_tables` = 3 , `l4_mem_archspecific_acpi_nvs`  
= 4 }  
*Memory sub types for l4\_mem\_type\_archspecific descriptors.*



## Functions

- [l4\\_kernel\\_info\\_mem\\_desc\\_t \\* l4\\_kernel\\_info\\_get\\_mem\\_descs](#) ([l4\\_kernel\\_info\\_t](#) \*kip) [L4\\_NOTHROW](#)  
*Get pointer to memory descriptors from KIP.*
- [unsigned l4\\_kernel\\_info\\_get\\_num\\_mem\\_descs](#) ([l4\\_kernel\\_info\\_t](#) \*kip) [L4\\_NOTHROW](#)  
*Get number of memory descriptors in KIP.*
- [void l4\\_kernel\\_info\\_set\\_mem\\_desc](#) ([l4\\_kernel\\_info\\_mem\\_desc\\_t](#) \*md, [l4\\_addr\\_t](#) start, [l4\\_addr\\_t](#) end, unsigned type, unsigned virt, unsigned sub\_type) [L4\\_NOTHROW](#)  
*Populate a memory descriptor.*
- [l4\\_umword\\_t l4\\_kernel\\_info\\_get\\_mem\\_desc\\_start](#) ([l4\\_kernel\\_info\\_mem\\_desc\\_t](#) \*md) [L4\\_NOTHROW](#)  
*Get start address of the region described by the memory descriptor.*
- [l4\\_umword\\_t l4\\_kernel\\_info\\_get\\_mem\\_desc\\_end](#) ([l4\\_kernel\\_info\\_mem\\_desc\\_t](#) \*md) [L4\\_NOTHROW](#)  
*Get end address of the region described by the memory descriptor.*
- [l4\\_umword\\_t l4\\_kernel\\_info\\_get\\_mem\\_desc\\_type](#) ([l4\\_kernel\\_info\\_mem\\_desc\\_t](#) \*md) [L4\\_NOTHROW](#)  
*Get type of the memory region.*
- [l4\\_umword\\_t l4\\_kernel\\_info\\_get\\_mem\\_desc\\_subtype](#) ([l4\\_kernel\\_info\\_mem\\_desc\\_t](#) \*md) [L4\\_NOTHROW](#)  
*Get sub-type of memory region.*
- [l4\\_umword\\_t l4\\_kernel\\_info\\_get\\_mem\\_desc\\_is\\_virtual](#) ([l4\\_kernel\\_info\\_mem\\_desc\\_t](#) \*md) [L4\\_NOTHROW](#)  
*Get virtual flag of the memory descriptor.*

### 17.556.1 Detailed Description

Memory description functions.

Definition in file [memdesc.h](#).

## 17.557 memdesc.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2007-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #ifndef __L4SYS_MEMDESC_H__
00014 #define __L4SYS_MEMDESC_H__
00015
00016 #include <l4/sys/kip.h>
00017
00028
00033 enum l4_mem_type_t
00034 {
00035     l4_mem_type_undefined    = 0x0,
00036     l4_mem_type_conventional = 0x1,
00037     l4_mem_type_reserved     = 0x2,
00038     l4_mem_type_dedicated    = 0x3,
00039     l4_mem_type_shared       = 0x4,
00040
00041     l4_mem_type_info         = 0xd,
00042     l4_mem_type_bootloader   = 0xe,
00043     l4_mem_type_archspecific = 0xf,
00044 };
00045
00050 enum l4_mem_info_sub_type_t
00051 {
00052     l4_mem_info_acpi_rsdp = 0,
00053
00054     l4_mem_reserved_kernel = 0,
00055     l4_mem_reserved_heap   = 1,

```

```

00056     l4_mem_reserved_mmio    = 2,
00057 };
00058
00063 enum l4_mem_archspecific_sub_type_common_t
00064 {
00065     l4_mem_archspecific_acpi_tables = 3,
00066     l4_mem_archspecific_acpi_nvs    = 4,
00067 };
00068
00069
00077 typedef struct l4_kernel_info_mem_desc_t
00078 {
00080     l4_umword_t l;
00082     l4_umword_t h;
00083 } l4_kernel_info_mem_desc_t;
00084
00085
00090 L4_INLINE
00091 l4_kernel_info_mem_desc_t *
00092 l4_kernel_info_get_mem_descs(l4_kernel_info_t *kip) L4_NOTHROW;
00093
00100 L4_INLINE
00101 unsigned
00102 l4_kernel_info_get_num_mem_descs(l4_kernel_info_t *kip) L4_NOTHROW;
00103
00115 L4_INLINE
00116 void
00117 l4_kernel_info_set_mem_desc(l4_kernel_info_mem_desc_t *md,
00118                             l4_addr_t start,
00119                             l4_addr_t end,
00120                             unsigned type,
00121                             unsigned virt,
00122                             unsigned sub_type) L4_NOTHROW;
00123
00130 L4_INLINE
00131 l4_umword_t
00132 l4_kernel_info_get_mem_desc_start(l4_kernel_info_mem_desc_t *md) L4_NOTHROW;
00133
00140 L4_INLINE
00141 l4_umword_t
00142 l4_kernel_info_get_mem_desc_end(l4_kernel_info_mem_desc_t *md) L4_NOTHROW;
00143
00150 L4_INLINE
00151 l4_umword_t
00152 l4_kernel_info_get_mem_desc_type(l4_kernel_info_mem_desc_t *md) L4_NOTHROW;
00153
00163 L4_INLINE
00164 l4_umword_t
00165 l4_kernel_info_get_mem_desc_subtype(l4_kernel_info_mem_desc_t *md) L4_NOTHROW;
00166
00173 L4_INLINE
00174 l4_umword_t
00175 l4_kernel_info_get_mem_desc_is_virtual(l4_kernel_info_mem_desc_t *md) L4_NOTHROW;
00176
00177 /*****
00178  * Implementations
00179  *****/
00180
00181 L4_INLINE
00182 l4_kernel_info_mem_desc_t *
00183 l4_kernel_info_get_mem_descs(l4_kernel_info_t *kip) L4_NOTHROW
00184 {
00185     return (l4_kernel_info_mem_desc_t *) ((l4_addr_t)kip + kip->mem_descs);
00186 }
00187
00188 L4_INLINE
00189 unsigned
00190 l4_kernel_info_get_num_mem_descs(l4_kernel_info_t *kip) L4_NOTHROW
00191 {
00192     return kip->mem_descs_num;
00193 }
00194
00195 L4_INLINE
00196 void
00197 l4_kernel_info_set_mem_desc(l4_kernel_info_mem_desc_t *md,
00198                             l4_addr_t start,
00199                             l4_addr_t end,
00200                             unsigned type,
00201                             unsigned virt,
00202                             unsigned sub_type) L4_NOTHROW
00203 {
00204     md->l = (start & ~0x3fffUL) | (type & 0x0f) | ((sub_type << 4) & 0x0f0);
00205     | (virt ? 0x200 : 0x0);
00206     md->h = end;
00207 }
00208
00209

```

```

00210 L4_INLINE
00211 l4_umword_t
00212 l4_kernel_info_get_mem_desc_start(l4_kernel_info_mem_desc_t *md) L4_NOTHROW
00213 {
00214     return md->l & ~0x3ffUL;
00215 }
00216
00217 L4_INLINE
00218 l4_umword_t
00219 l4_kernel_info_get_mem_desc_end(l4_kernel_info_mem_desc_t *md) L4_NOTHROW
00220 {
00221     return md->h | 0x3ffUL;
00222 }
00223
00224 L4_INLINE
00225 l4_umword_t
00226 l4_kernel_info_get_mem_desc_type(l4_kernel_info_mem_desc_t *md) L4_NOTHROW
00227 {
00228     return md->l & 0xf;
00229 }
00230
00231 L4_INLINE
00232 l4_umword_t
00233 l4_kernel_info_get_mem_desc_subtype(l4_kernel_info_mem_desc_t *md) L4_NOTHROW
00234 {
00235     return (md->l & 0xf0) >> 4;
00236 }
00237
00238 L4_INLINE
00239 l4_umword_t
00240 l4_kernel_info_get_mem_desc_is_virtual(l4_kernel_info_mem_desc_t *md) L4_NOTHROW
00241 {
00242     return md->l & 0x200;
00243 }
00244
00245 #endif /* ! __L4SYS__MEMDESC_H__ */

```

## 17.558 meta

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002
00003 #pragma once
00004
00005 #include <l4/sys/meta>
00006 #include <l4/sys/typeinfo_svr>
00007
00008 namespace L4Re { namespace Util {
00009     using L4::Util::handle_meta_request;
00010 }}

```

## 17.559 l4/sys/meta File Reference

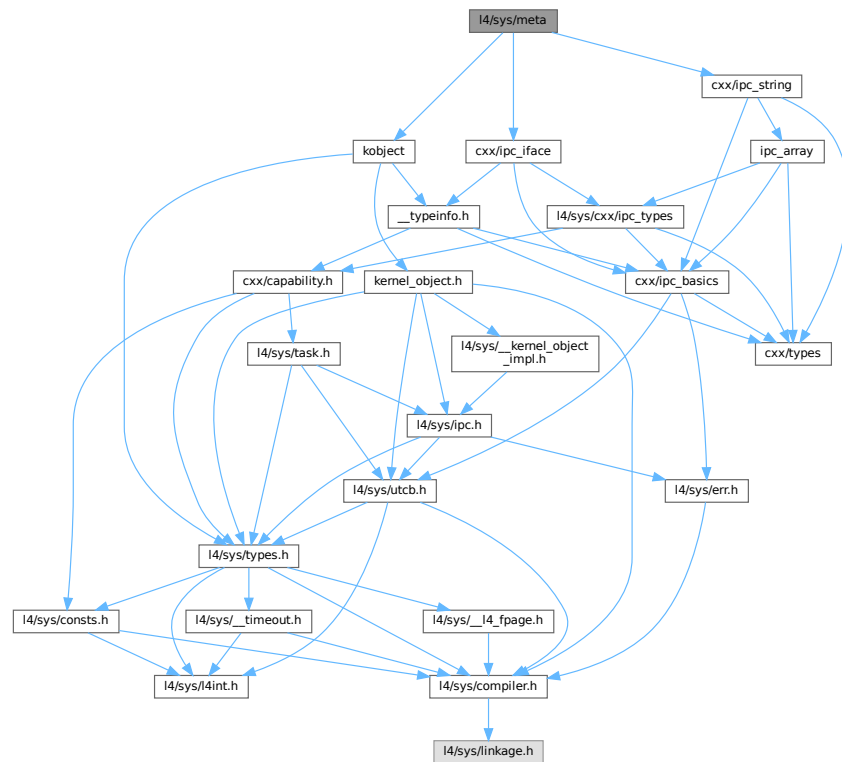
Meta interface for getting dynamic type information about objects behind capabilities.

```

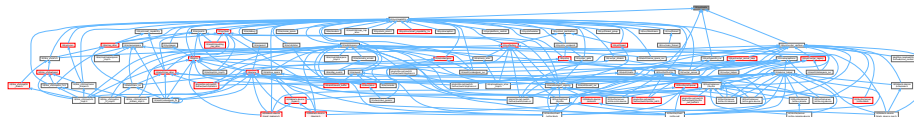
#include "kobject"
#include "cxx/ipc_iface"
#include "cxx/ipc_string"

```

Include dependency graph for meta:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4::Meta](#)

*Meta* interface that shall be implemented by each [L4Re](#) object and gives access to the dynamic type information for [L4Re](#) objects.

## Namespaces

- namespace [L4](#)

*L4* low-level kernel interface.

## 17.559.1 Detailed Description

Meta interface for getting dynamic type information about objects behind capabilities.

Definition in file [meta](#).

## 17.560 meta

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 #include "kobject"
00012 #include "cxx/ipc_iface"
00013 #include "cxx/ipc_string"
00014
00015 namespace L4 {
00016
00017 class Meta : public Kobject_t<Meta, Kobject, L4_PROTO_META>
00018 {
00019 public:
00020     L4_INLINE_RPC(l4_msgtag_t, num_interfaces, ());
00021     L4_INLINE_RPC(l4_msgtag_t, interface, (l4_umword_t idx, long *proto,
00022                                           L4::Ipc::String<char> *name));
00023     L4_INLINE_RPC(l4_msgtag_t, supports, (l4_mword_t protocol));
00024     typedef L4::Typeid::Rpc<num_interfaces_t, interface_t, supports_t> Rpc;
00025 };
00026
00027 }

```

## 17.561 l4/sys/obj\_info.h File Reference

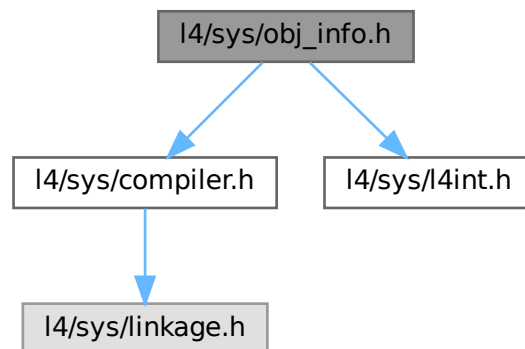
Debugger related functions.

```

#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>

```

Include dependency graph for obj\_info.h:



## Functions

- [l4\\_msgtag\\_t](#) [l4\\_debugger\\_query\\_obj\\_infos](#) ([l4\\_cap\\_idx\\_t](#) cap, [l4\\_addr\\_t](#) ku\_mem\_addr, [l4\\_size\\_t](#) ku\_mem\_size, [l4\\_umword\\_t](#) skip, [l4\\_umword\\_t](#) \*result\_cnt, [l4\\_umword\\_t](#) \*result\_all) [L4\\_NOTHROW](#)

*Retrieve information from the kernel about all objects in the mapping database and write data to the passed KU memory.*

### 17.561.1 Detailed Description

Debugger related functions.

#### Attention

This API is subject to change!

Definition in file [obj\\_info.h](#).

### 17.561.2 Function Documentation

#### 17.561.2.1 [l4\\_debugger\\_query\\_obj\\_infos\(\)](#)

```
l4\_msgtag\_t l4\_debugger\_query\_obj\_infos (
    l4\_cap\_idx\_t cap,
    l4\_addr\_t ku_mem_addr,
    l4\_size\_t ku_mem_size,
    l4\_umword\_t skip,
    l4\_umword\_t * result_cnt,
    l4\_umword\_t * result_all) [inline]
```

Retrieve information from the kernel about all objects in the mapping database and write data to the passed KU memory.

#### Parameters

	<i>cap</i>	Capability of the debugger object.
	<i>ku_mem_addr</i>	Address of the KU memory for writing the information.
	<i>ku_mem_size</i>	Size of the KU memory to writing the information.
	<i>skip</i>	Number of objects to skip.
out	<i>result_cnt</i>	Number of objects in the mapping database.
out	<i>result_all</i>	Number of objects written to the KU memory.

**Note**

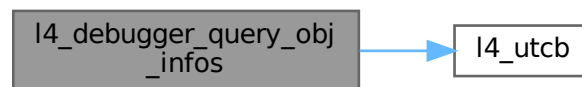
The kernel will only write a number of object information which fits to the passed KU memory. To retrieve missing object information, repeat the call and adapt the `skip` parameter accordingly.

If this system call is performed several times, the number of kernel objects might have changed in the meantime.

Definition at line 176 of file [obj\\_info.h](#).

References [L4\\_NOTHROW](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**17.562 obj\_info.h**

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (C) 2023 Kernkonzept GmbH.
00003  * Author(s): Frank Mehnert <frank.mehnert@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00014
00015 #pragma once
00016
00017 #include <l4/sys/compiler.h>
00018 #include <l4/sys/l4int.h>
00019
00020 struct L4_kobj_info
00021 {
00022     // Type_mapping: See Jdb_mapdb::info_obj_mapping().
00023     struct Mapping
00024     {
00025         enum { Type = 0 };
00026         l4_uint64_t mapping_ptr;
00027         char space_name[16];
00028         l4_uint32_t cap_idx;
00029         l4_uint16_t entry_rights;
00030         l4_uint16_t entry_flags;
00031         l4_uint64_t entry_ptr;
00032     };
00033
00034     // Type_thread: See Jdb_tcb::info_kobject().
00035     struct Thread
00036     {
00037         enum { Type = 1 };
00038         bool is_kernel;
00039         bool is_current;
00040         bool in_ready_list;
00041         bool is_kernel_task;
00042         l4_uint32_t home_cpu;
00043         l4_uint32_t current_cpu;
00044         l4_int64_t ref_cnt;
00045         l4_uint64_t space_id;
00046     };
00047
00048     // Type_space: See Jdb_space::info_kobject().

```

```

00049 struct Space
00050 {
00051     enum { Type = 2 };
00052     bool is_kernel;
00053     l4_int64_t ref_cnt;
00054 };
00055
00056 // Type_vm: See Jdb_vm::info_kobject().
00057 struct Vm
00058 {
00059     enum { Type = 3 };
00060     l4_uint64_t utcb;
00061     l4_uint64_t pc;
00062 };
00063
00064 // Type_ipc_gate: See Jdb_ipc_gate::info_kobject().
00065 struct Ipc_gate
00066 {
00067     enum { Type = 4 };
00068     l4_uint64_t label;
00069     l4_uint64_t thread_id;
00070 };
00071
00072 // Type_irq: See Jdb_kobject_irq::info_kobject().
00073 struct Irq_sender
00074 {
00075     enum { Type = 5 };
00076     char chip_type[10];
00077     l4_uint16_t flags;
00078     l4_uint32_t pin;
00079     l4_uint64_t label;
00080     l4_uint64_t target_id;
00081     l4_int64_t queued;
00082 };
00083
00084 // Type_irq: See Jdb_kobject_irq::info_kobject().
00085 struct Irq_semaphore
00086 {
00087     enum { Type = 6 };
00088     char chip_type[10];
00089     l4_uint16_t flags;
00090     l4_uint32_t pin;
00091     l4_uint64_t sender_id;
00092     l4_uint64_t target_id;
00093     l4_int64_t queued;
00094 };
00095
00096 // Type_factory: See Jdb_factory::info_kobject().
00097 struct Factory
00098 {
00099     enum { Type = 7 };
00100     l4_uint64_t current;
00101     l4_uint64_t limit;
00102 };
00103
00104 struct Jdb          { enum { Type = 8 }; };
00105 struct Scheduler    { enum { Type = 9 }; };
00106 struct Vlog         { enum { Type = 10 }; };
00107 struct Pfc          { enum { Type = 11 }; };
00108 struct Dmar_space   { enum { Type = 12 }; };
00109 struct Iommu        { enum { Type = 13 }; };
00110 struct Smmu         { enum { Type = 14 }; };
00111
00112 l4_uint64_t type;5;
00113 l4_uint64_t id;59;
00114 l4_uint64_t mapping_ptr;
00115 l4_uint64_t ref_cnt;
00116 union
00117 {
00118     Thread thread;
00119     Space space;
00120     Vm vm;
00121     Ipc_gate ipc_gate;
00122     Irq_sender irq_sender;
00123     Irq_semaphore irq_semaphore;
00124     Factory factory;
00125     Mapping mapping;
00126     l4_uint64_t raw[5];
00127 };
00128 };
00129
00130 static_assert(sizeof(L4_kobj_info) == 64, "Size of Jobb_info");
00131
00150 L4_INLINE l4_msgtag_t
00151 l4_debugger_query_obj_infos(l4_cap_idx_t cap, l4_addr_t ku_mem_addr,
00152                             l4_size_t ku_mem_size, l4_umword_t skip,
00153                             l4_umword_t *result_cnt, l4_umword_t *result_all)

```



```

00154                                     L4_NOTHROW;
00155
00156 L4_INLINE l4_msgtag_t
00157 l4_debugger_query_obj_infos_u(l4_cap_idx_t cap, l4_addr_t ku_mem_addr,
00158                               l4_size_t ku_mem_size, l4_umword_t skip,
00159                               l4_umword_t *result_cnt, l4_umword_t *result_all,
00160                               l4_utcb_t *utcb) L4_NOTHROW
00161 {
00162     l4_utcb_mr()->mr[0] = 16;
00163     l4_utcb_mr()->mr[1] = ku_mem_addr;
00164     l4_utcb_mr()->mr[2] = ku_mem_size;
00165     l4_utcb_mr()->mr[3] = skip;
00166
00167     l4_msgtag_t tag = l4_invoke_debugger(cap, l4_msgtag(0, 4, 0, 0), utcb);
00168
00169     *result_cnt = l4_utcb_mr()->mr[0];
00170     *result_all = l4_utcb_mr()->mr[1];
00171
00172     return tag;
00173 }
00174
00175 L4_INLINE l4_msgtag_t
00176 l4_debugger_query_obj_infos(l4_cap_idx_t cap, l4_addr_t ku_mem_addr,
00177                             l4_size_t ku_mem_size, l4_umword_t skip,
00178                             l4_umword_t *result_cnt, l4_umword_t *result_all)
00179                             L4_NOTHROW
00180 {
00181     return l4_debugger_query_obj_infos_u(cap, ku_mem_addr, ku_mem_size, skip,
00182                                           result_cnt, result_all, l4_utcb());
00183 }

```

## 17.563 l4/sys/pager File Reference

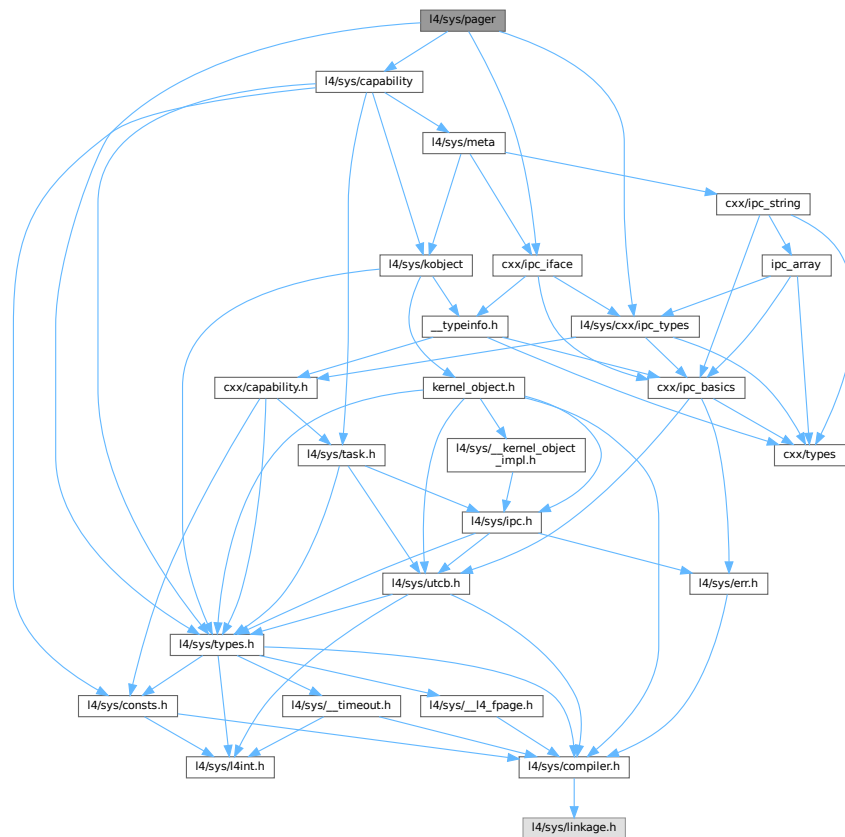
Pager and lo\_pager C++ interface.

```

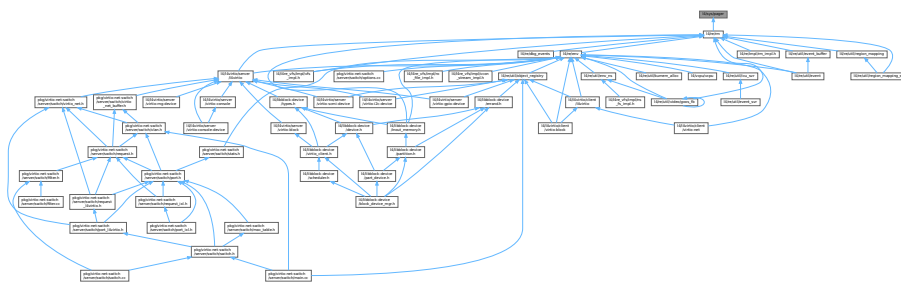
#include <l4/sys/capability>
#include <l4/sys/types.h>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_iface>

```

Include dependency graph for pager:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `L4::lo_pager`  
*lo\_pager* interface.
- class `L4::Pager`  
*Pager* interface including the *lo\_pager* interface.

## Namespaces

- namespace **L4**  
*L4 low-level kernel interface.*

## 17.563.1 Detailed Description

Pager and Io\_pager C++ interface.

Definition in file [pager](#).

## 17.564 pager

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <l4/sys/capability>
00011 #include <l4/sys/types.h>
00012 #include <l4/sys/cxx/ipc_types>
00013 #include <l4/sys/cxx/ipc_iface>
00014
00015 namespace L4 {
00016
00017 class L4_EXPORT Io_pager :
00018     public Kobject_0t<Io_pager, L4_PROTO_IO_PAGE_FAULT>
00019 {
00020 public:
00021     L4_INLINE_RPC(
00022         l4_msgtag_t, io_page_fault, (l4_fpage_t io_pfa, l4_umword_t pc,
00023                                     L4::Ipc::Rcv_fpage rwin,
00024                                     L4::Ipc::Opt<L4::Ipc::Snd_fpage &> fp));
00025
00026     typedef L4::Typeid::Rpc_nocode<io_page_fault_t> Rpcs;
00027 };
00028
00029 class L4_EXPORT Pager :
00030     public Kobject_t<Pager, Io_pager, L4_PROTO_PAGE_FAULT>
00031 {
00032 public:
00033     L4_INLINE_RPC(
00034         l4_msgtag_t, page_fault, (l4_umword_t pfa, l4_umword_t pc,
00035                                   L4::Ipc::Rcv_fpage rwin,
00036                                   L4::Ipc::Opt<L4::Ipc::Snd_fpage &> fp));
00037
00038     typedef L4::Typeid::Rpc_nocode<page_fault_t> Rpcs;
00039 };
00040
00041 }
00042

```

## 17.565 l4/sys/platform\_control File Reference

Platform control object.

```

#include <l4/sys/capability>
#include <l4/sys/platform_control.h>

```



## 17.566 platform\_control

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Steffen Liebergeld <steffen.liebergeld@kernkonzept.com>
00004  *      Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00005  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00006  *      economic rights: Technische Universität Dresden (Germany)
00007  *
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010
00011 #pragma once
00012
00013 #include <l4/sys/capability>
00014 #include <l4/sys/platform_control.h>
00015 #include <l4/sys/cxx/ipc_iface>
00016
00017 namespace L4 {
00018
00019 class L4_EXPORT Platform_control
00020 : public Kobject_t<Platform_control, Kobject, L4_PROTO_PLATFORM_CTL>
00021 {
00022 public:
00023     L4_INLINE_RPC_OP(L4_PLATFORM_CTL_SYS_SUSPEND_OP,
00024                     l4_msgtag_t, system_suspend, (l4_umword_t extras));
00025
00026     L4_INLINE_RPC_OP(L4_PLATFORM_CTL_SYS_SHUTDOWN_OP,
00027                     l4_msgtag_t, system_shutdown, (l4_umword_t reboot));
00028
00029     L4_INLINE_RPC_OP(L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP,
00030                     l4_msgtag_t, cpu_allow_shutdown,
00031                     (l4_umword_t phys_id, l4_umword_t enable));
00032
00033     L4_INLINE_RPC_OP(L4_PLATFORM_CTL_CPU_ENABLE_OP,
00034                     l4_msgtag_t, cpu_enable, (l4_umword_t phys_id));
00035
00036     L4_INLINE_RPC_OP(L4_PLATFORM_CTL_CPU_DISABLE_OP,
00037                     l4_msgtag_t, cpu_disable, (l4_umword_t phys_id));
00038
00039     typedef L4::Typeid::Rpcs_sys<system_suspend_t, system_shutdown_t,
00040                                 cpu_allow_shutdown_t, cpu_enable_t,
00041                                 cpu_disable_t> Rpcs;
00042 };
00043
00044 }
00045
00046
00047
00048

```

## 17.567 platform\_control.h

```

00001 /*
00002  * Copyright (C) 2024 Kernkonzept GmbH.
00003  * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include_next <l4/sys/platform_control.h>
00010 #include <l4/sys/__platform_control-arm.h>

```

## 17.568 platform\_control.h

```

00001 /*
00002  * Copyright (C) 2024 Kernkonzept GmbH.
00003  * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include_next <l4/sys/platform_control.h>
00010 #include <l4/sys/__platform_control-arm.h>

```

## 17.569 I4/sys/platform\_control.h File Reference

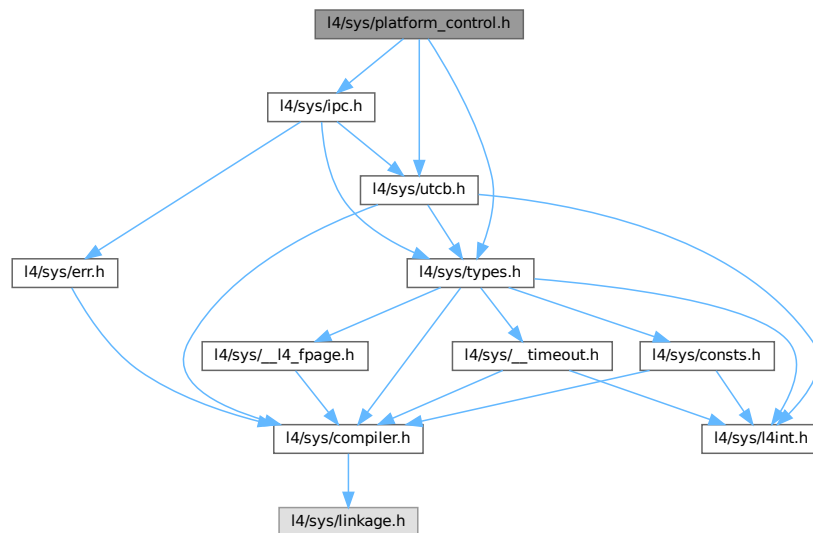
Platform control object.

```
#include <l4/sys/types.h>
```

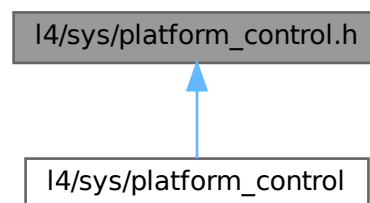
```
#include <l4/sys/utcb.h>
```

```
#include <l4/sys/ipc.h>
```

Include dependency graph for platform\_control.h:



This graph shows which files directly or indirectly include this file:



### Enumerations

- enum [L4\\_platform\\_ctl\\_ops](#) {  
[L4\\_PLATFORM\\_CTL\\_SYS\\_SUSPEND\\_OP](#) = 0UL , [L4\\_PLATFORM\\_CTL\\_SYS\\_SHUTDOWN\\_OP](#) = 1UL ,  
[L4\\_PLATFORM\\_CTL\\_CPU\\_ALLOW\\_SHUTDOWN\\_OP](#) = 2UL , [L4\\_PLATFORM\\_CTL\\_CPU\\_ENABLE\\_OP](#) = 3UL ,  
[L4\\_PLATFORM\\_CTL\\_CPU\\_DISABLE\\_OP](#) = 4UL , [L4\\_PLATFORM\\_CTL\\_SET\\_TASK\\_ASID\\_OP](#) = 0x10UL }

*Operations on platform-control objects.*

- enum [L4\\_platform\\_ctl\\_proto](#) { [L4\\_PROTO\\_PLATFORM\\_CTL](#) = 0 }

*Predefined protocol type for messages to platform-control objects.*

## Functions

- [l4\\_msgtag\\_t l4\\_platform\\_ctl\\_system\\_suspend \(l4\\_cap\\_idx\\_t pfc, l4\\_umword\\_t extras\) L4\\_NOTHROW](#)  
*Enter suspend to RAM.*
- [l4\\_msgtag\\_t l4\\_platform\\_ctl\\_system\\_shutdown \(l4\\_cap\\_idx\\_t pfc, l4\\_umword\\_t reboot\) L4\\_NOTHROW](#)  
*Shutdown or reboot the system.*
- [l4\\_msgtag\\_t l4\\_platform\\_ctl\\_cpu\\_allow\\_shutdown \(l4\\_cap\\_idx\\_t pfc, l4\\_umword\\_t phys\\_id, l4\\_umword\\_t enable\) L4\\_NOTHROW](#)  
*Allow a CPU to be shut down.*
- [l4\\_msgtag\\_t l4\\_platform\\_ctl\\_cpu\\_enable \(l4\\_cap\\_idx\\_t pfc, l4\\_umword\\_t phys\\_id\) L4\\_NOTHROW](#)  
*Enable an offline CPU.*
- [l4\\_msgtag\\_t l4\\_platform\\_ctl\\_cpu\\_disable \(l4\\_cap\\_idx\\_t pfc, l4\\_umword\\_t phys\\_id\) L4\\_NOTHROW](#)  
*Disable an online CPU.*

## 17.569.1 Detailed Description

Platform control object.

Definition in file [platform\\_control.h](#).

## 17.570 platform\_control.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00007  *
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010
00011 #pragma once
00012
00013 #include <l4/sys/types.h>
00014 #include <l4/sys/utcb.h>
00015
00030
00031
00049 L4_INLINE l4_msgtag_t
00050 l4_platform_ctl_system_suspend(l4_cap_idx_t pfc,
00051                               l4_umword_t extras) L4_NOTHROW;
00052
00056 L4_INLINE l4_msgtag_t
00057 l4_platform_ctl_system_suspend_u(l4_cap_idx_t pfc,
00058                                 l4_umword_t extras,
00059                                 l4_utcb_t *utcb) L4_NOTHROW;
00060
00061
00070 L4_INLINE l4_msgtag_t
00071 l4_platform_ctl_system_shutdown(l4_cap_idx_t pfc,
00072                                 l4_umword_t reboot) L4_NOTHROW;
00073
00077 L4_INLINE l4_msgtag_t
00078 l4_platform_ctl_system_shutdown_u(l4_cap_idx_t pfc,
00079                                   l4_umword_t reboot,
00080                                   l4_utcb_t *utcb) L4_NOTHROW;
00081
00091 L4_INLINE l4_msgtag_t
00092 l4_platform_ctl_cpu_allow_shutdown(l4_cap_idx_t pfc,
00093                                   l4_umword_t phys_id,
00094                                   l4_umword_t enable) L4_NOTHROW;
00095
00099 L4_INLINE l4_msgtag_t
00100 l4_platform_ctl_cpu_allow_shutdown_u(l4_cap_idx_t pfc,
00101                                     l4_umword_t phys_id,
00102                                     l4_umword_t enable,
00103                                     l4_utcb_t *utcb) L4_NOTHROW;
00104
00114 L4_INLINE l4_msgtag_t

```

```

00115 l4_platform_ctl_cpu_enable(l4_cap_idx_t pfc,
00116                             l4_umword_t phys_id) L4_NOTHROW;
00117
00121 L4_INLINE l4_msgtag_t
00122 l4_platform_ctl_cpu_enable_u(l4_cap_idx_t pfc,
00123                             l4_umword_t phys_id,
00124                             l4_utcb_t *utcb) L4_NOTHROW;
00125
00136 L4_INLINE l4_msgtag_t
00137 l4_platform_ctl_cpu_disable(l4_cap_idx_t pfc,
00138                             l4_umword_t phys_id) L4_NOTHROW;
00139
00143 L4_INLINE l4_msgtag_t
00144 l4_platform_ctl_cpu_disable_u(l4_cap_idx_t pfc,
00145                               l4_umword_t phys_id,
00146                               l4_utcb_t *utcb) L4_NOTHROW;
00147 /* ends l4_platform_control_api group */
00149
00150
00159 enum L4_platform_ctl_ops
00160 {
00161     L4_PLATFORM_CTL_SYS_SUSPEND_OP      = 0UL,
00162     L4_PLATFORM_CTL_SYS_SHUTDOWN_OP     = 1UL,
00163     L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP = 2UL,
00164     L4_PLATFORM_CTL_CPU_ENABLE_OP       = 3UL,
00165     L4_PLATFORM_CTL_CPU_DISABLE_OP      = 4UL,
00166
00167     L4_PLATFORM_CTL_SET_TASK_ASID_OP     = 0x10UL,
00168 };
00169
00174 enum L4_platform_ctl_proto
00175 {
00181     L4_PROTO_PLATFORM_CTL = 0
00182 };
00183
00184 /* IMPLEMENTATION -----*/
00185
00186 #include <l4/sys/ipc.h>
00187
00188 L4_INLINE l4_msgtag_t
00189 l4_platform_ctl_system_suspend_u(l4_cap_idx_t pfc,
00190                                  l4_umword_t extras,
00191                                  l4_utcb_t *utcb) L4_NOTHROW
00192 {
00193     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00194     v->mr[0] = L4_PLATFORM_CTL_SYS_SUSPEND_OP;
00195     v->mr[1] = extras;
00196     return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 2, 0, 0),
00197                       L4_IPC_NEVER);
00198 }
00199
00200 L4_INLINE l4_msgtag_t
00201 l4_platform_ctl_system_shutdown_u(l4_cap_idx_t pfc,
00202                                   l4_umword_t reboot,
00203                                   l4_utcb_t *utcb) L4_NOTHROW
00204 {
00205     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00206     v->mr[0] = L4_PLATFORM_CTL_SYS_SHUTDOWN_OP;
00207     v->mr[1] = reboot;
00208     return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 2, 0, 0),
00209                       L4_IPC_NEVER);
00210 }
00211
00212
00213 L4_INLINE l4_msgtag_t
00214 l4_platform_ctl_system_suspend(l4_cap_idx_t pfc,
00215                                l4_umword_t extras) L4_NOTHROW
00216 {
00217     return l4_platform_ctl_system_suspend_u(pfc, extras, l4_utcb());
00218 }
00219
00220 L4_INLINE l4_msgtag_t
00221 l4_platform_ctl_system_shutdown(l4_cap_idx_t pfc,
00222                                 l4_umword_t reboot) L4_NOTHROW
00223 {
00224     return l4_platform_ctl_system_shutdown_u(pfc, reboot, l4_utcb());
00225 }
00226
00227 L4_INLINE l4_msgtag_t
00228 l4_platform_ctl_cpu_allow_shutdown_u(l4_cap_idx_t pfc,
00229                                       l4_umword_t phys_id,
00230                                       l4_umword_t enable,
00231                                       l4_utcb_t *utcb) L4_NOTHROW
00232 {
00233     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00234     v->mr[0] = L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP;
00235     v->mr[1] = phys_id;

```



```

00236     v->mr[2] = enable;
00237     return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 3, 0, 0),
00238                       L4_IPC_NEVER);
00239 }
00240
00241 L4_INLINE l4_msgtag_t
00242 l4_platform_ctl_cpu_allow_shutdown(l4_cap_idx_t pfc,
00243                                   l4_umword_t phys_id,
00244                                   l4_umword_t enable) L4_NOTHROW
00245 {
00246     return l4_platform_ctl_cpu_allow_shutdown_u(pfc, phys_id, enable, l4_utcb());
00247 }
00248
00249 L4_INLINE l4_msgtag_t
00250 l4_platform_ctl_cpu_enable_u(l4_cap_idx_t pfc,
00251                             l4_umword_t phys_id,
00252                             l4_utcb_t *utcb) L4_NOTHROW
00253 {
00254     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00255     v->mr[0] = L4_PLATFORM_CTL_CPU_ENABLE_OP;
00256     v->mr[1] = phys_id;
00257     return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 2, 0, 0),
00258                       L4_IPC_NEVER);
00259 }
00260
00261 L4_INLINE l4_msgtag_t
00262 l4_platform_ctl_cpu_disable_u(l4_cap_idx_t pfc,
00263                              l4_umword_t phys_id,
00264                              l4_utcb_t *utcb) L4_NOTHROW
00265 {
00266     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00267     v->mr[0] = L4_PLATFORM_CTL_CPU_DISABLE_OP;
00268     v->mr[1] = phys_id;
00269     return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 2, 0, 0),
00270                       L4_IPC_NEVER);
00271 }
00272
00273 L4_INLINE l4_msgtag_t
00274 l4_platform_ctl_cpu_enable(l4_cap_idx_t pfc,
00275                           l4_umword_t phys_id) L4_NOTHROW
00276 {
00277     return l4_platform_ctl_cpu_enable_u(pfc, phys_id, l4_utcb());
00278 }
00279
00280 L4_INLINE l4_msgtag_t
00281 l4_platform_ctl_cpu_disable(l4_cap_idx_t pfc,
00282                             l4_umword_t phys_id) L4_NOTHROW
00283 {
00284     return l4_platform_ctl_cpu_disable_u(pfc, phys_id, l4_utcb());
00285 }

```

## 17.571 l4/sys/rcv\_endpoint File Reference

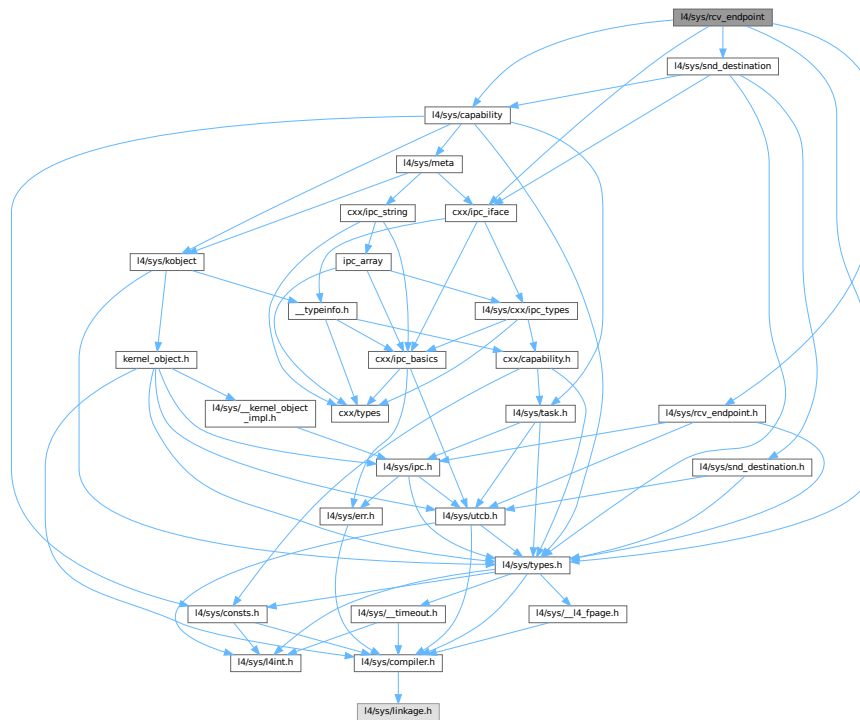
The C++ Receive endpoint interface.

```

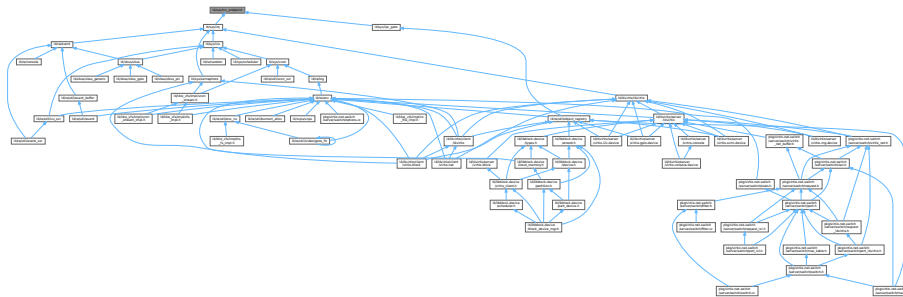
#include <l4/sys/rcv_endpoint.h>
#include <l4/sys/snd_destination>
#include <l4/sys/types.h>
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_iface>

```

Include dependency graph for `rcv_endpoint`:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4::Rcv\\_endpoint](#)  
*Interface for kernel objects that allow to receive IPC from them.*

## Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*

## 17.571.1 Detailed Description

The C++ Receive endpoint interface.

Definition in file [rcv\\_endpoint](#).

## 17.572 rcv\_endpoint

[Go to the documentation of this file.](#)

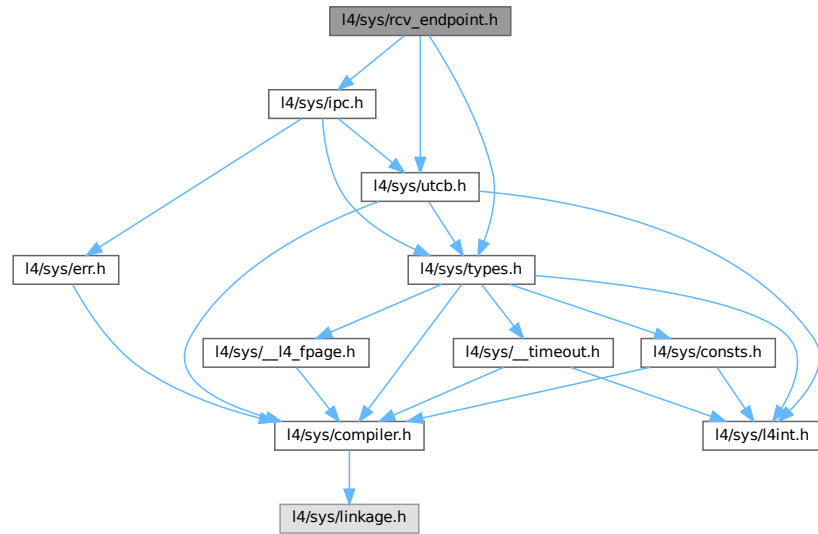
```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00013 #include <l4/sys/rcv_endpoint.h>
00014 #include <l4/sys/snd_destination>
00015 #include <l4/sys/types.h>
00016 #include <l4/sys/capability>
00017 #include <l4/sys/cxx/ipc_iface>
00018
00019 namespace L4 {
00020
00021 class Thread;
00022
00030 class L4_EXPORT Rcv_endpoint :
00031     public Kobject_t<Rcv_endpoint, Kobject, L4_PROTO_KOBJECT,
00032         Type_info::Demand_t<1> >
00033 {
00034 public:
00066     L4_INLINE_RPC_OP(L4_RCV_EP_BIND_OP,
00067         l4_msgtag_t, bind_thread, (Ipc::Cap<Thread> t, l4_umword_t label));
00068
00101     l4_msgtag_t bind_snd_destination(Cap<Snd_destination> snd_dst, l4_umword_t label)
00102     {
00103         return l4_rcv_ep_bind_snd_destination(cap(), snd_dst.cap(), label);
00104     }
00105
00106     typedef L4::Typeid::Rpc_sys<bind_thread_t> Rpc;
00107 };
00108
00109 }
```

## 17.573 l4/sys/rcv\_endpoint.h File Reference

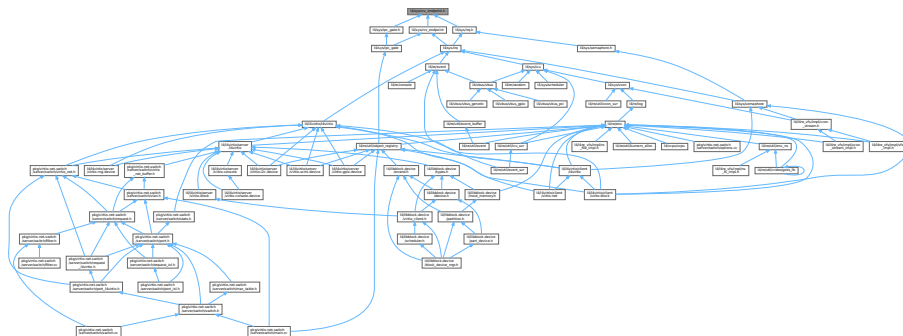
Receive endpoint C interface.

```
#include <l4/sys/utcb.h>
#include <l4/sys/types.h>
#include <l4/sys/ipc.h>
```

Include dependency graph for `rcv_endpoint.h`:



This graph shows which files directly or indirectly include this file:



## Enumerations

- enum `L4_rcv_ep_ops` { `L4_RCV_EP_BIND_OP` = 0x10 }  
Receive endpoint operations.

## Functions

- `l4_msgtag_t l4_rcv_ep_bind_thread` (`l4_cap_idx_t` ep, `l4_cap_idx_t` thread, `l4_umword_t` label)  
Bind the IPC receive endpoint to a thread.
- `l4_msgtag_t l4_rcv_ep_bind_snd_destination` (`l4_cap_idx_t` ep, `l4_cap_idx_t` snd\_dst, `l4_umword_t` label)  
Bind the IPC receive endpoint to a send destination (a thread).

## 17.573.1 Detailed Description

Receive endpoint C interface.

Definition in file [rcv\\_endpoint.h](#).

## 17.573.2 Enumeration Type Documentation

### 17.573.2.1 L4\_rcv\_ep\_ops

enum [L4\\_rcv\\_ep\\_ops](#)

Receive endpoint operations.

#### Enumerator

L4_RCV_EP_BIND_OP	Bind to thread or thread group.
-------------------	---------------------------------

Definition at line 93 of file [rcv\\_endpoint.h](#).

## 17.574 rcv\_endpoint.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00007  *
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010 #pragma once
00011
00012 #include <l4/sys/utcb.h>
00013 #include <l4/sys/types.h>
00014
00045 L4_INLINE l4_msgtag_t
00046 l4_rcv_ep_bind_thread(l4_cap_idx_t ep, l4_cap_idx_t thread,
00047                      l4_umword_t label);
00048
00076 L4_INLINE l4_msgtag_t
00077 l4_rcv_ep_bind_snd_destination(l4_cap_idx_t ep, l4_cap_idx_t snd_dst,
00078                               l4_umword_t label);
00079
00084 L4_INLINE l4_msgtag_t
00085 l4_rcv_ep_bind_thread_u(l4_cap_idx_t ep, l4_cap_idx_t thread,
00086                        l4_umword_t label, l4_utcb_t *utcb);
00087
00088 L4_INLINE l4_msgtag_t
00089 l4_rcv_ep_bind_snd_destination_u(l4_cap_idx_t ep, l4_cap_idx_t snd_dst,
00090                                 l4_umword_t label, l4_utcb_t *utcb);
00091
00093 enum L4_rcv_ep_ops
00094 {
00095     L4_RCV_EP_BIND_OP = 0x10,
00096 };
00097
00098 /* IMPLEMENTATION -----*/
00099
00100 #include <l4/sys/ipc.h>
00101
00102 L4_INLINE l4_msgtag_t
00103 l4_rcv_ep_bind_thread_u(l4_cap_idx_t ep,
00104                        l4_cap_idx_t thread, l4_umword_t label,
00105                        l4_utcb_t *utcb)
00106 {
00107     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);

```

```

00108     m->mr[0] = L4_RCV_EP_BIND_OP;
00109     m->mr[1] = label;
00110     m->mr[2] = l4_map_obj_control(0, 0);
00111     m->mr[3] = l4_obj_fpage(thread, 0, L4_CAP_FPAGE_RWS).raw;
00112     return l4_ipc_call(ep, utcb, l4_msgtag(L4_PROTO_KOBJECT, 2, 1, 0),
00113                       L4_IPC_NEVER);
00114 }
00115
00116 L4_INLINE l4_msgtag_t
00117 l4_rcv_ep_bind_snd_destination_u(l4_cap_idx_t ep,
00118                                  l4_cap_idx_t snd_dst, l4_umword_t label,
00119                                  l4_utcb_t *utcb)
00120 {
00121     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00122     m->mr[0] = L4_RCV_EP_BIND_OP;
00123     m->mr[1] = label;
00124     m->mr[2] = l4_map_obj_control(0, 0);
00125     m->mr[3] = l4_obj_fpage(snd_dst, 0, L4_CAP_FPAGE_RWS).raw;
00126     return l4_ipc_call(ep, utcb, l4_msgtag(L4_PROTO_KOBJECT, 2, 1, 0),
00127                       L4_IPC_NEVER);
00128 }
00129
00130 L4_INLINE l4_msgtag_t
00131 l4_rcv_ep_bind_thread(l4_cap_idx_t ep, l4_cap_idx_t thread,
00132                       l4_umword_t label)
00133 {
00134     return l4_rcv_ep_bind_thread_u(ep, thread, label, l4_utcb());
00135 }
00136
00137 L4_INLINE l4_msgtag_t
00138 l4_rcv_ep_bind_snd_destination(l4_cap_idx_t ep, l4_cap_idx_t snd_dst,
00139                                 l4_umword_t label)
00140 {
00141     return l4_rcv_ep_bind_snd_destination_u(ep, snd_dst, label, l4_utcb());
00142 }

```

## 17.575 l4/sys/scheduler File Reference

Scheduler object functions.

```

#include <l4/sys/icu>
#include <l4/sys/scheduler.h>
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_iface>

```

- class [L4::Scheduler](#)  
C++ interface of the [Scheduler](#) kernel object, see [Scheduler](#) for the C interface.

- namespace L4  
*L4 low-level kernel interface.*

Scheduler object functions.

Generated for L4Re by Doxygen

## 17.576 scheduler

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include <l4/sys/icu>
00012 #include <l4/sys/scheduler.h>
00013 #include <l4/sys/capability>
00014 #include <l4/sys/cxx/ipc_iface>
00015
00016 namespace L4 {
00017
00018 class L4_EXPORT Scheduler :
00019     public Kobject_t<Scheduler, Icu, L4_PROTO_SCHEDULER,
00020         Type_info::Demand_t<1> >
00021 {
00022 public:
00023     // ABI function for 'info' call
00024     L4_INLINE_RPC_NF_OP(L4_SCHEDULER_INFO_OP,
00025         l4_msgtag_t, info, (l4_umword_t gran_offset, l4_umword_t *map,
00026             l4_umword_t *cpu_max, l4_umword_t *sched_classes));
00027
00028     l4_msgtag_t info(l4_umword_t *cpu_max, l4_sched_cpu_set_t *cpus,
00029         l4_umword_t *sched_classes = nullptr,
00030         l4_utcb_t *utcb = l4_utcb()) const noexcept
00031     {
00032         l4_umword_t max = 0;
00033         l4_umword_t sc = 0;
00034         l4_msgtag_t t =
00035             info_t::call(c(), cpus->gran_offset, &cpus->map, &max, &sc, utcb);
00036         if (cpu_max)
00037             *cpu_max = max;
00038         if (sched_classes)
00039             *sched_classes = sc;
00040         return t;
00041     }
00042
00043     L4_INLINE_RPC_OP(L4_SCHEDULER_RUN_THREAD_OP,
00044         l4_msgtag_t, run_thread, (Ipc::Cap<Thread> thread, l4_sched_param_t const &sp));
00045
00046     L4_INLINE_RPC_OP(L4_SCHEDULER_IDLE_TIME_OP,
00047         l4_msgtag_t, idle_time, (l4_sched_cpu_set_t const &cpus,
00048             l4_kernel_clock_t *us));
00049
00050     bool is_online(l4_umword_t cpu, l4_utcb_t *utcb = l4_utcb()) const noexcept
00051     { return l4_scheduler_is_online_u(cap(), cpu, utcb); }
00052
00053     typedef L4::Typeid::Rpcsys<info_t, run_thread_t, idle_time_t> Rpcsys;
00054 };
00055
00056 }
```

## 17.577 l4/sys/semaphore File Reference

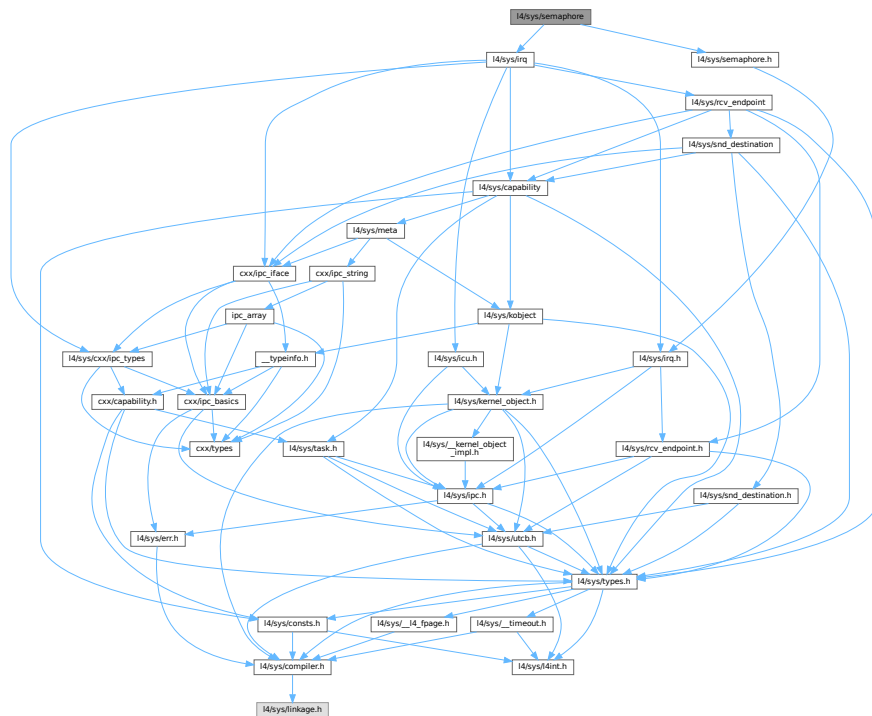
Semaphore class definition.

```

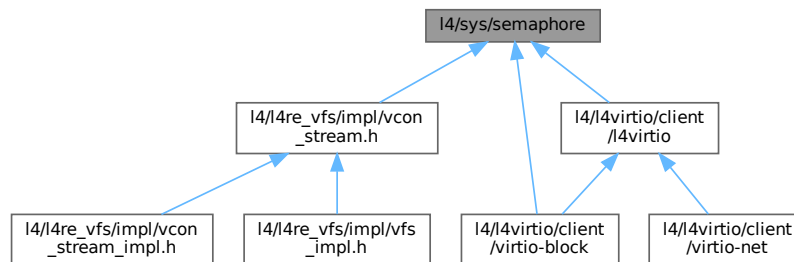
#include <l4/sys/irq>
#include <l4/sys/semaphore.h>
```



Include dependency graph for semaphore:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [L4::Semaphore](#)

*C++ Kernel-provided semaphore interface, see [Kernel-provided semaphore](#) for the C interface.*

## Namespaces

- namespace [L4](#)

*[L4](#) low-level kernel interface.*

## 17.577.1 Detailed Description

Semaphore class definition.

Definition in file [semaphore](#).

## 17.578 semaphore

[Go to the documentation of this file.](#)

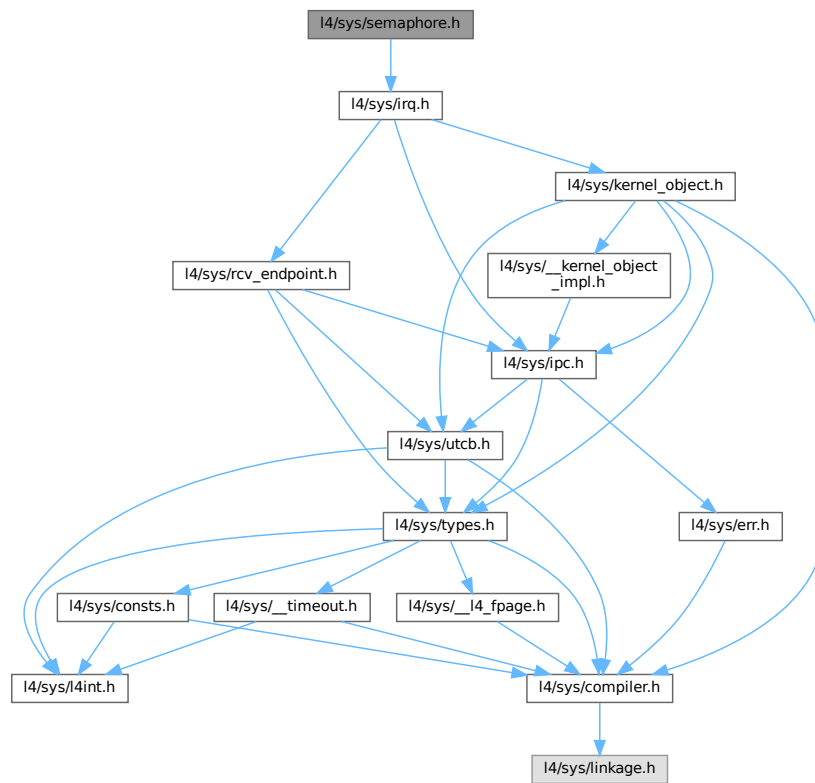
```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2015 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011
00012 #pragma once
00013
00014 #include <l4/sys/irq>
00015 #include <l4/sys/semaphore.h>
00016
00017 namespace L4 {
00018
00051 struct Semaphore : Kobject_t<Semaphore, Triggerable, L4_PROTO_SEMAPHORE>
00052 {
00067     l4_msgtag_t up(l4_utcb_t *utcb = l4_utcb()) noexcept
00068     { return trigger(utcb); }
00069
00089     l4_msgtag_t down(l4_timeout_t timeout = L4_IPC_NEVER,
00090                     l4_utcb_t *utcb = l4_utcb()) noexcept
00091     { return l4_semaphore_down_u(cap(), timeout, utcb); }
00092 };
00093
00094 }
```

## 17.579 l4/sys/semaphore.h File Reference

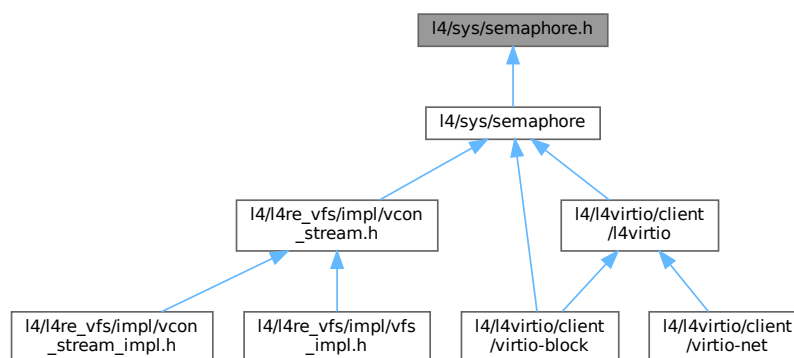
C semaphore interface.

```
#include <l4/sys/irq.h>
```

Include dependency graph for semaphore.h:



This graph shows which files directly or indirectly include this file:



## Functions

- [l4\\_msgtag\\_t l4\\_semaphore\\_up \(l4\\_cap\\_idx\\_t sem\) L4\\_NOTHROW](#)  
Semaphore up operation (wrapper for trigger()).
- [l4\\_msgtag\\_t l4\\_semaphore\\_down \(l4\\_cap\\_idx\\_t sem, l4\\_timeout\\_t timeout\) L4\\_NOTHROW](#)  
Semaphore down operation.

## 17.579.1 Detailed Description

C semaphore interface.

Definition in file [semaphore.h](#).

## 17.580 semaphore.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2015 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011
00012 #pragma once
00013
00014 #include <l4/sys/irq.h>
00015
00024
00025 enum L4_semaphore_op
00026 {
00027     L4_SEMAPHORE_OP_DOWN    = 0,
00028     // semaphore up is IRQ_OP_TRIGGER with IRQ/Triggerable protocol
00029 };
00030
00044 L4_INLINE l4_msgtag_t
00045 l4_semaphore_up(l4_cap_idx_t sem) L4_NOTHROW
00046 {
00047     return l4_irq_trigger(sem);
00048 }
00049
00053 L4_INLINE l4_msgtag_t
00054 l4_semaphore_up_u(l4_cap_idx_t sem, l4_utcb_t *utcb) L4_NOTHROW
00055 {
00056     return l4_irq_trigger_u(sem, utcb);
00057 }
00058
00078 L4_INLINE l4_msgtag_t
00079 l4_semaphore_down(l4_cap_idx_t sem, l4_timeout_t timeout) L4_NOTHROW;
00080
00084 L4_INLINE l4_msgtag_t
00085 l4_semaphore_down_u(l4_cap_idx_t sem, l4_timeout_t to,
00086                    l4_utcb_t *utcb) L4_NOTHROW;
00087
00088
00089 L4_INLINE l4_msgtag_t
00090 l4_semaphore_down_u(l4_cap_idx_t sem, l4_timeout_t to,
00091                    l4_utcb_t *utcb) L4_NOTHROW
00092 {
00093     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00094     m->mr[0] = L4_SEMAPHORE_OP_DOWN;
00095     return l4_ipc_call(sem, utcb, l4_msgtag(L4_PROTO_SEMAPHORE, 1, 0, 0), to);
00096 }
00097
00098
00099 L4_INLINE l4_msgtag_t
00100 l4_semaphore_down(l4_cap_idx_t sem, l4_timeout_t to) L4_NOTHROW
00101 {
00102     return l4_semaphore_down_u(sem, to, l4_utcb());
00103 }
00104

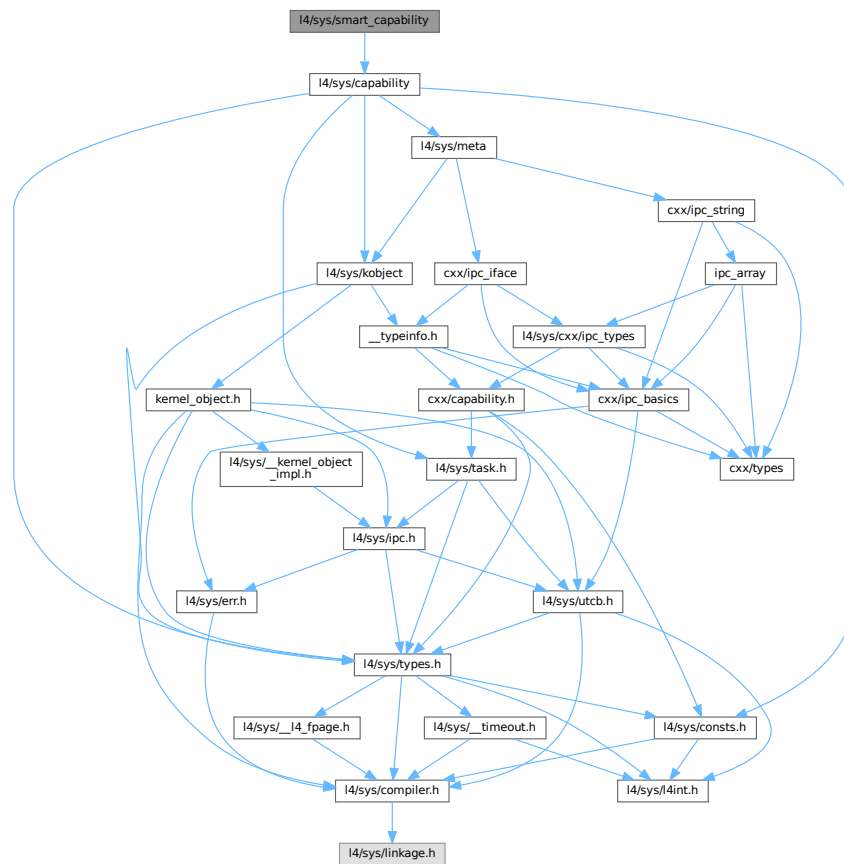
```

## 17.581 l4/sys/smart\_capability File Reference

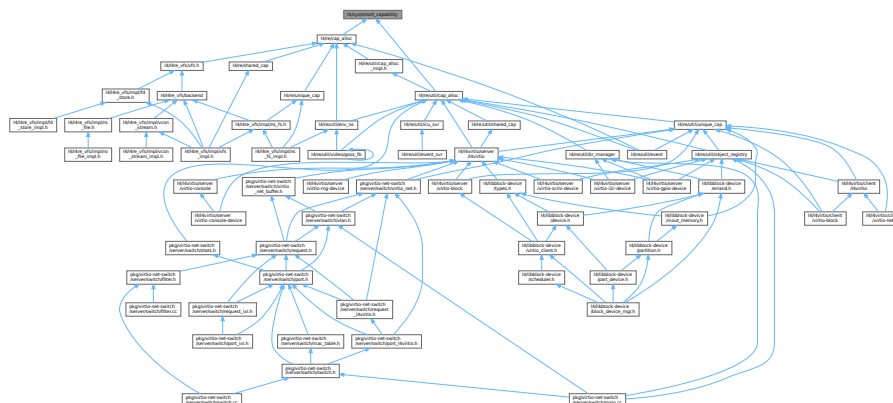
L4::Capability class.

```
#include <l4/sys/capability>
```

Include dependency graph for smart\_capability:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4::Smart\\_cap< T, SMART >](#)  
*Smart capability class.*

## Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*

## Functions

- `template<typename T, typename F, typename SMART>`  
`Smart_cap< T, SMART > L4::cap_cast (Smart_cap< F, SMART > const &c) noexcept`  
*static\_cast for (smart) capabilities.*
- `template<typename T, typename F, typename SMART>`  
`Smart_cap< T, SMART > L4::cap_reinterpret_cast (Smart_cap< F, SMART > const &c) noexcept`  
*reinterpret\_cast for (smart) capabilities.*

## 17.581.1 Detailed Description

L4::Capability class.

### Author

Alexander Warg [alexander.warg@os.inf.tu-dresden.de](mailto:alexander.warg@os.inf.tu-dresden.de)

Definition in file [smart\\_capability](#).

## 17.582 smart\_capability

[Go to the documentation of this file.](#)

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00009 /*
00010  * (c) 2008-2009 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * License: see LICENSE.spdx (in this directory or the directories above)
00014  */
00015 #pragma once
00016
00017 #include <l4/sys/capability>
00018
00019 namespace L4 {
00020
00021 template< typename T, typename SMART >
00022 class Smart_cap : public Cap_base, private SMART
00023 {
00024 public:
00025     SMART const &smart() const noexcept { return *this; }
00026
00027     void _delete() noexcept
00028     {
00029         SMART::free(const_cast<Smart_cap<T, SMART>&>(*this));
00030     }
00031
00032     Cap<T> release() const noexcept
00033     {
00034         l4_cap_idx_t r = cap();
00035         SMART::invalidate(const_cast<Smart_cap<T, SMART>&>(*this));
00036
00037         return Cap<T>(r);
00038     }
00039
00040     void reset() noexcept
00041     {
00042         _c = L4_INVALID_CAP;
00043     }
00044 }
```

```

00047     }
00048
00049     Smart_cap() noexcept : Cap_base(Invalid) {}
00050
00051     Smart_cap(Cap_base::Cap_type t) noexcept : Cap_base(t) {}
00052
00061     template< typename O >
00062     Smart_cap(Cap<O> const &p) noexcept : Cap_base(p.cap())
00063     { Cap<T>::template check_convertible_from<O>(); }
00064
00065     template< typename O >
00066     Smart_cap(Cap<O> const &p, SMART const &smart) noexcept
00067     : Cap_base(p.cap()), SMART(smart)
00068     { Cap<T>::template check_convertible_from<O>(); }
00069
00070     template< typename O >
00071     Smart_cap(Smart_cap<O, SMART> const &o) noexcept
00072     : Cap_base(SMART::copy(o)), SMART(o.smart())
00073     { Cap<T>::template check_convertible_from<O>(); }
00074
00075     Smart_cap(Smart_cap const &o) noexcept
00076     : Cap_base(SMART::copy(o)), SMART(o.smart())
00077     { }
00078
00079     template< typename O >
00080     Smart_cap(typename Cap<O>::Cap_type cap) noexcept : Cap_base(cap)
00081     { Cap<T>::template check_convertible_from<O>(); }
00082
00083     void operator = (typename Cap<T>::Cap_type cap) noexcept
00084     {
00085         _delete();
00086         _c = cap;
00087     }
00088
00089     template< typename O >
00090     void operator = (Smart_cap<O, SMART> const &o) noexcept
00091     {
00092         _delete();
00093         _c = this->SMART::copy(o).cap();
00094         this->SMART::operator = (o.smart());
00095         // return *this;
00096     }
00097
00098     Smart_cap const &operator = (Smart_cap const &o) noexcept
00099     {
00100         if (&o == this)
00101             return *this;
00102
00103         _delete();
00104         _c = this->SMART::copy(o).cap();
00105         this->SMART::operator = (o.smart());
00106         return *this;
00107     }
00108
00109 #if __cplusplus >= 201103L
00110     template< typename O >
00111     Smart_cap(Smart_cap<O, SMART> &&o) noexcept
00112     : Cap_base(o.release()), SMART(o.smart())
00113     { Cap<T>::template check_convertible_from<O>(); }
00114
00115     Smart_cap(Smart_cap &&o) noexcept
00116     : Cap_base(o.release()), SMART(o.smart())
00117     { }
00118
00119     template< typename O >
00120     void operator = (Smart_cap<O, SMART> &&o) noexcept
00121     {
00122         _delete();
00123         _c = o.release().cap();
00124         this->SMART::operator = (o.smart());
00125         // return *this;
00126     }
00127
00128     Smart_cap const &operator = (Smart_cap &&o) noexcept
00129     {
00130         if (&o == this)
00131             return *this;
00132
00133         _delete();
00134         _c = o.release().cap();
00135         this->SMART::operator = (o.smart());
00136         return *this;
00137     }
00138 #endif
00139
00143     Cap<T> operator -> () const noexcept { return Cap<T>(_c); }
00144

```

```

00145     Cap<T> get() const noexcept { return Cap<T>(_c); }
00146
00147     ~Smart_cap() noexcept { _delete(); }
00148 };
00149
00150 template< typename T >
00151 class Weak_cap : public Cap_base
00152 {
00153 public:
00154     Weak_cap() noexcept : Cap_base(Invalid) {}
00155
00156     template< typename O >
00157     Weak_cap(typename Cap<O>::Cap_type t) noexcept : Cap_base(t)
00158     { Cap<T>::template check_convertible_from<O>(); }
00159
00160     template< typename O, typename S >
00161     Weak_cap(Smart_cap<O, S> const &c) noexcept : Cap_base(c.cap())
00162     { Cap<T>::template check_convertible_from<O>(); }
00163
00164     Weak_cap(Weak_cap const &o) noexcept : Cap_base(o) {}
00165
00166     template< typename O >
00167     Weak_cap(Weak_cap<O> const &o) noexcept : Cap_base(o)
00168     { Cap<T>::template check_convertible_from<O>(); }
00169
00170 };
00171
00172 namespace Cap_traits {
00173     template< typename T1, typename T2 >
00174     struct Type { enum { Equal = false }; };
00175
00176     template< typename T1 >
00177     struct Type<T1,T1> { enum { Equal = true }; };
00178 };
00179
00190 template< typename T, typename F, typename SMART >
00191 inline
00192 Smart_cap<T, SMART> cap_cast(Smart_cap<F, SMART> const &c) noexcept
00193 {
00194     Cap<T>::template check_castable_from<F>();
00195     return Smart_cap<T, SMART>(Cap<T>(SMART::copy(c).cap()));
00196 }
00197
00198
00209 template< typename T, typename F, typename SMART >
00210 inline
00211 Smart_cap<T, SMART> cap_reinterpret_cast(Smart_cap<F, SMART> const &c) noexcept
00212 {
00213     return Smart_cap<T, SMART>(Cap<T>(SMART::copy(c).cap()));
00214 }
00215
00216
00217 }

```

## 17.583 l4/sys/snd\_destination File Reference

The C++ Sender destination interface.

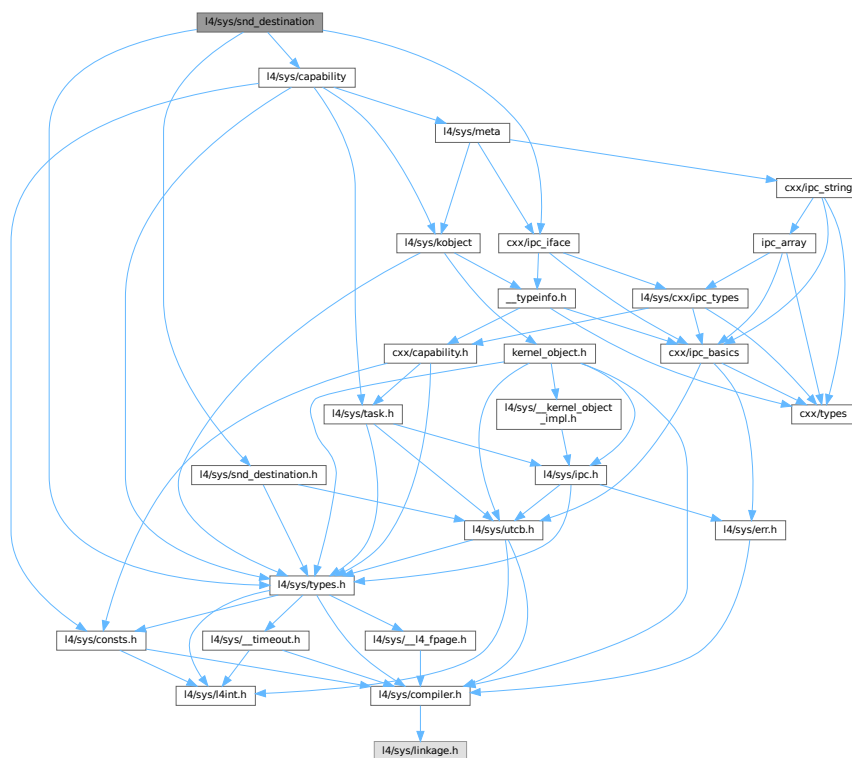
```

#include <l4/sys/snd_destination.h>
#include <l4/sys/types.h>
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_iface>

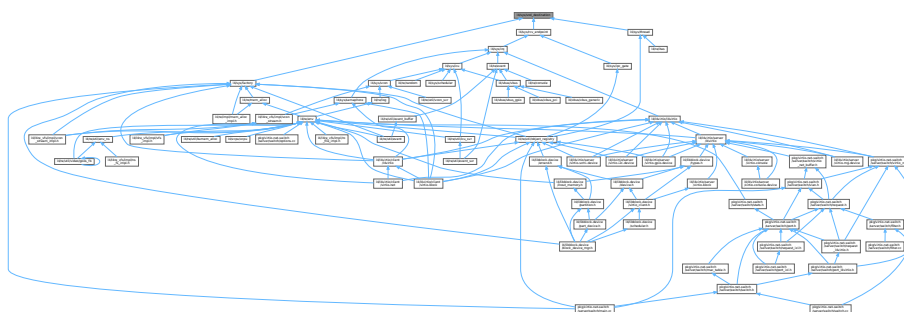
```



Include dependency graph for `snd_destination`:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*

## 17.583.1 Detailed Description

The C++ Sender destination interface.

Definition in file [snd\\_destination](#).

## 17.584 snd\_destination

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2025 Frank Mehnert <frank.mehnert@kernkonzept.com>
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00013 #include <l4/sys/snd_destination.h>
00014 #include <l4/sys/types.h>
00015 #include <l4/sys/capability>
00016 #include <l4/sys/cxx/ipc_iface>
00017
00018 namespace L4 {
00019
00020 class L4_EXPORT Snd_destination :
00021     public Kobject_t<Snd_destination, Kobject, L4_PROTO_KOBJECT>
00022 {
00023 };
00024
00025 }

```

## 17.585 l4/sys/snd\_destination.h File Reference

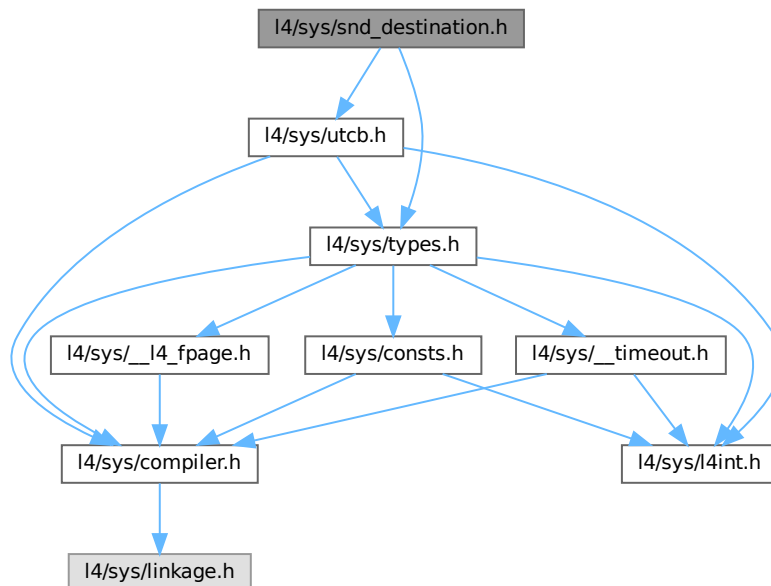
Sender destination endpoint C interface.

```

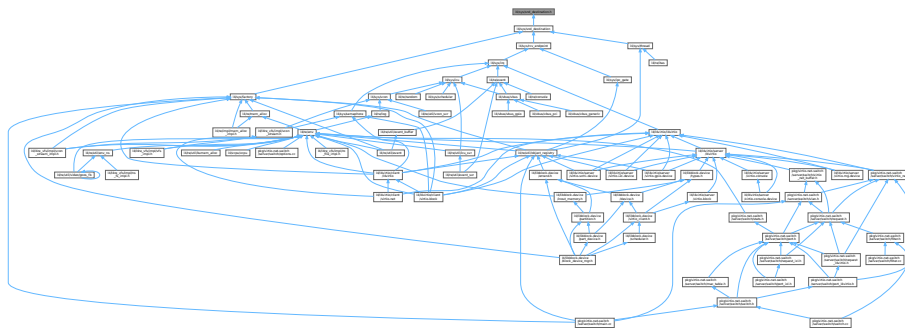
#include <l4/sys/utcb.h>
#include <l4/sys/types.h>

```

Include dependency graph for snd\_destination.h:



This graph shows which files directly or indirectly include this file:



## 17.585.1 Detailed Description

Sender destination endpoint C interface.

Definition in file [snd\\_destination.h](#).

## 17.586 snd\_destination.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2025 Frank Mehnert <frank.mehnert@kernkonzept.com>
00007  *
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010 #pragma once
00011
00012 #include <l4/sys/utcb.h>
00013 #include <l4/sys/types.h>

```

## 17.587 l4/sys/task File Reference

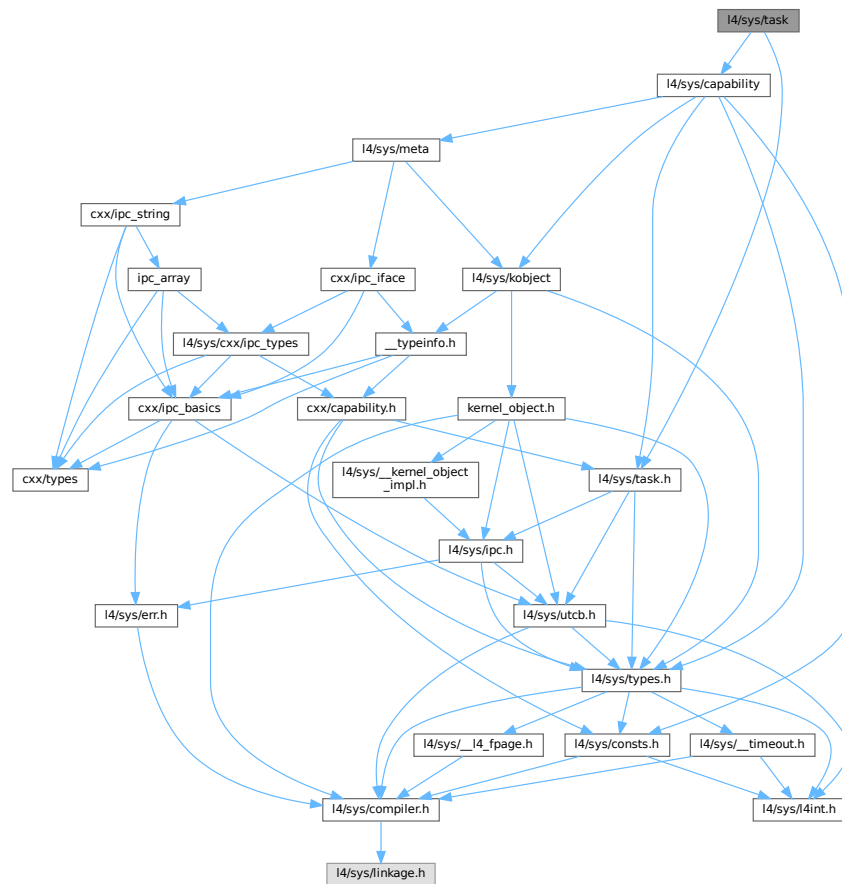
Common task related definitions.

```

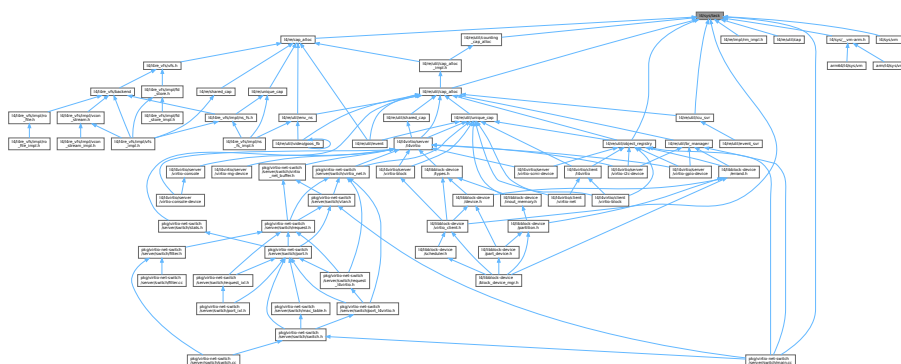
#include <l4/sys/task.h>
#include <l4/sys/capability>

```

Include dependency graph for task:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4::Task](#)

*C++ interface of the [Task](#) kernel object, see [Task](#) for the C interface.*

## Namespaces

- namespace [L4](#)  
*[L4](#) low-level kernel interface.*

### 17.587.1 Detailed Description

Common task related definitions.

Definition in file [task](#).

## 17.588 task

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 #include <l4/sys/task.h>
00013 #include <l4/sys/capability>
00014
00015 namespace L4 {
00016
00017 class Task :
00018 public Kobject<Task, Kobject, L4_PROTO_TASK,
00019         Type_info::Demand_t<2> >
00020 {
00021 public:
00022     l4_msgtag_t map(Cap<Task> const &src_task,
00023                     l4_fpage_t const &snd_fpage, l4_umword_t snd_base,
00024                     l4_utcb_t *utcb = l4_utcb()) noexcept
00025     { return l4_task_map_u(cap(), src_task.cap(), snd_fpage, snd_base, utcb); }
00026
00027     l4_msgtag_t unmap(l4_fpage_t const &fpage,
00028                       l4_umword_t map_mask,
00029                       l4_utcb_t *utcb = l4_utcb()) noexcept
00030     { return l4_task_unmap_u(cap(), fpage, map_mask, utcb); }
00031
00032     l4_msgtag_t unmap_batch(l4_fpage_t const *fpages,
00033                             unsigned num_fpages,
00034                             l4_umword_t map_mask,
00035                             l4_utcb_t *utcb = l4_utcb()) noexcept
00036     { return l4_task_unmap_batch_u(cap(), fpages, num_fpages, map_mask, utcb); }
00037
00038     l4_msgtag_t delete_obj(L4::Cap<void> obj,
00039                            l4_utcb_t *utcb = l4_utcb()) noexcept
00040     { return l4_task_delete_obj_u(cap(), obj.cap(), utcb); }
00041
00042     l4_msgtag_t release_cap(L4::Cap<void> cap,
00043                             l4_utcb_t *utcb = l4_utcb()) noexcept
00044     { return l4_task_release_cap_u(this->cap(), cap.cap(), utcb); }
00045
00046     l4_msgtag_t cap_valid(Cap<void> const &cap,
00047                           l4_utcb_t *utcb = l4_utcb()) noexcept
00048     { return l4_task_cap_valid_u(this->cap(), cap.cap(), utcb); }
00049
00050     l4_msgtag_t cap_equal(Cap<void> const &cap_a,
00051                           Cap<void> const &cap_b,
00052                           l4_utcb_t *utcb = l4_utcb()) noexcept
00053     { return l4_task_cap_equal_u(cap(), cap_a.cap(), cap_b.cap(), utcb); }
00054
00055     l4_msgtag_t add_ku_mem(l4_fpage_t *fpage,
00056                           l4_utcb_t *utcb = l4_utcb()) noexcept
00057     { return l4_task_add_ku_mem_u(cap(), fpage, utcb); }
00058 };
00059 }
```

## 17.589 task.h

```

00001 /*
00002  * (c) 2018 Adam Lackorzynski <adam@l4re.org>
00003  *
00004  * License: see LICENSE.spdx (in this directory or the directories above)
00005  */
00006 #pragma once
00007
00008 #include_next <l4/sys/task.h>
00009 #include <l4/sys/__task-arm.h>

```

## 17.590 task.h

```

00001 /*
00002  * (c) 2018 Adam Lackorzynski <adam@l4re.org>
00003  *
00004  * License: see LICENSE.spdx (in this directory or the directories above)
00005  */
00006 #pragma once
00007
00008 #include_next <l4/sys/task.h>
00009 #include <l4/sys/__task-arm.h>

```

## 17.591 l4/sys/task.h File Reference

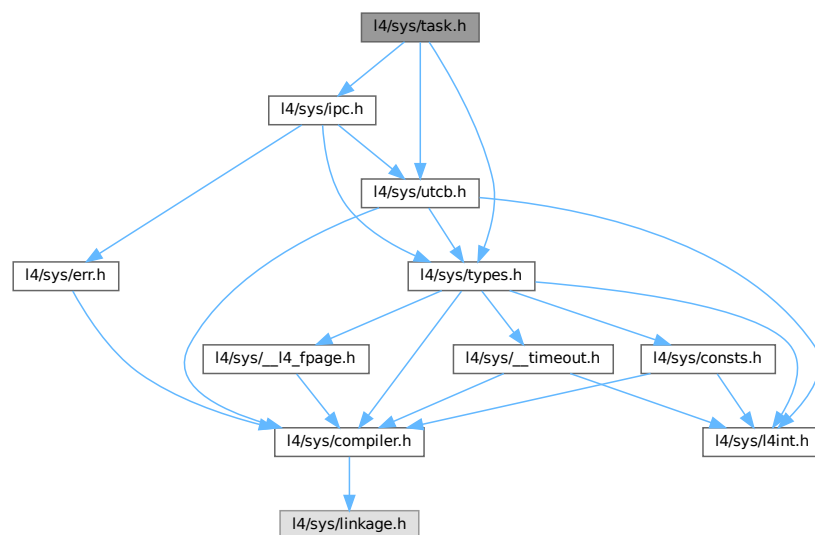
Common task related definitions.

```
#include <l4/sys/types.h>
```

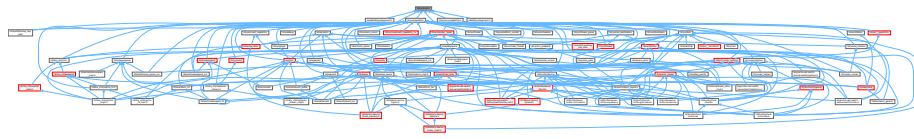
```
#include <l4/sys/utcb.h>
```

```
#include <l4/sys/ipc.h>
```

Include dependency graph for task.h:



This graph shows which files directly or indirectly include this file:



## Enumerations

- enum `L4_task_ops` {  
`L4_TASK_MAP_OP` = 0UL , `L4_TASK_UNMAP_OP` = 1UL , `L4_TASK_CAP_INFO_OP` = 2UL ,  
`L4_TASK_ADD_KU_MEM_OP` = 3UL ,  
`L4_TASK_LDT_SET_X86_OP` = 0x11UL , `L4_TASK_MAP_VGICC_ARM_OP` = 0x12UL }

*Operations on task objects.*

## Functions

- `l4_msgtag_t l4_task_map` (`l4_cap_idx_t` dst\_task, `l4_cap_idx_t` src\_task, `l4_fpage_t` snd\_fpage, `l4_umword_t` snd\_base) `L4_NOTHROW`  
*Map resources available in the source task to a destination task.*
- `l4_msgtag_t l4_task_unmap` (`l4_cap_idx_t` task, `l4_fpage_t` fpage, `l4_umword_t` map\_mask) `L4_NOTHROW`  
*Revoke rights from the task.*
- `l4_msgtag_t l4_task_unmap_batch` (`l4_cap_idx_t` task, `l4_fpage_t` const \*fpages, unsigned num\_fpages, `l4_umword_t` map\_mask) `L4_NOTHROW`  
*Revoke rights from a task.*
- `l4_msgtag_t l4_task_delete_obj` (`l4_cap_idx_t` task, `l4_cap_idx_t` obj) `L4_NOTHROW`  
*Release capability and delete object.*
- `l4_msgtag_t l4_task_release_cap` (`l4_cap_idx_t` task, `l4_cap_idx_t` cap) `L4_NOTHROW`  
*Release object capability.*
- `l4_msgtag_t l4_task_cap_valid` (`l4_cap_idx_t` task, `l4_cap_idx_t` cap) `L4_NOTHROW`  
*Check whether a capability is present (refers to an object).*
- `l4_msgtag_t l4_task_cap_equal` (`l4_cap_idx_t` task, `l4_cap_idx_t` cap\_a, `l4_cap_idx_t` cap\_b) `L4_NOTHROW`  
*Test whether two capabilities point to the same object with the same permissions (only considering selected permissions).*
- `l4_msgtag_t l4_task_add_ku_mem` (`l4_cap_idx_t` task, `l4_fpage_t` \*ku\_mem) `L4_NOTHROW`  
*Add kernel-user memory.*

### 17.591.1 Detailed Description

Common task related definitions.

Definition in file [task.h](#).

## 17.592 task.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *      Björn Döbel <doebel@os.inf.tu-dresden.de>,
00009  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #pragma once
00015 #include <l4/sys/types.h>
00016 #include <l4/sys/utcb.h>
00017
00031
00079 L4_INLINE l4_msgtag_t
00080 l4_task_map(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
00081            l4_fpage_t snd_fpage, l4_umword_t snd_base) L4_NOTHROW;
00082
00086 L4_INLINE l4_msgtag_t
00087 l4_task_map_u(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
00088              l4_fpage_t snd_fpage, l4_umword_t snd_base, l4_utcb_t *utcb) L4_NOTHROW;
00089
00134 L4_INLINE l4_msgtag_t
00135 l4_task_unmap(l4_cap_idx_t task, l4_fpage_t fpage,
00136              l4_umword_t map_mask) L4_NOTHROW;
00137
00141 L4_INLINE l4_msgtag_t
00142 l4_task_unmap_u(l4_cap_idx_t task, l4_fpage_t fpage,
00143                l4_umword_t map_mask, l4_utcb_t *utcb) L4_NOTHROW;
00144
00164 L4_INLINE l4_msgtag_t
00165 l4_task_unmap_batch(l4_cap_idx_t task, l4_fpage_t const *fpages,
00166                    unsigned num_fpages, l4_umword_t map_mask) L4_NOTHROW;
00167
00171 L4_INLINE l4_msgtag_t
00172 l4_task_unmap_batch_u(l4_cap_idx_t task, l4_fpage_t const *fpages,
00173                      unsigned num_fpages, l4_umword_t map_mask,
00174                      l4_utcb_t *u) L4_NOTHROW;
00175
00197 L4_INLINE l4_msgtag_t
00198 l4_task_delete_obj(l4_cap_idx_t task, l4_cap_idx_t obj) L4_NOTHROW;
00199
00203 L4_INLINE l4_msgtag_t
00204 l4_task_delete_obj_u(l4_cap_idx_t task, l4_cap_idx_t obj,
00205                     l4_utcb_t *u) L4_NOTHROW;
00206
00225 L4_INLINE l4_msgtag_t
00226 l4_task_release_cap(l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW;
00227
00231 L4_INLINE l4_msgtag_t
00232 l4_task_release_cap_u(l4_cap_idx_t task, l4_cap_idx_t cap,
00233                      l4_utcb_t *u) L4_NOTHROW;
00234
00235
00253 L4_INLINE l4_msgtag_t
00254 l4_task_cap_valid(l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW;
00255
00259 L4_INLINE l4_msgtag_t
00260 l4_task_cap_valid_u(l4_cap_idx_t task, l4_cap_idx_t cap, l4_utcb_t *utcb) L4_NOTHROW;
00261
00289 L4_INLINE l4_msgtag_t
00290 l4_task_cap_equal(l4_cap_idx_t task, l4_cap_idx_t cap_a,
00291                 l4_cap_idx_t cap_b) L4_NOTHROW;
00292
00296 L4_INLINE l4_msgtag_t
00297 l4_task_add_ku_mem_u(l4_cap_idx_t task, l4_fpage_t *ku_mem,
00298                    l4_utcb_t *u) L4_NOTHROW;
00299
00326 L4_INLINE l4_msgtag_t
00327 l4_task_add_ku_mem(l4_cap_idx_t task, l4_fpage_t *ku_mem) L4_NOTHROW;
00328
00329
00333 L4_INLINE l4_msgtag_t
00334 l4_task_cap_equal_u(l4_cap_idx_t task, l4_cap_idx_t cap_a,
00335                    l4_cap_idx_t cap_b, l4_utcb_t *utcb) L4_NOTHROW;
00336
00341 enum L4_task_ops
00342 {
00343     L4_TASK_MAP_OP          = 0UL,
00344     L4_TASK_UNMAP_OP       = 1UL,
00345     L4_TASK_CAP_INFO_OP    = 2UL,

```



```

00346     L4_TASK_ADD_KU_MEM_OP      = 3UL,
00347     L4_TASK_LDT_SET_X86_OP     = 0x11UL,
00348     L4_TASK_MAP_VGICC_ARM_OP  = 0x12UL,
00349 };
00350
00351
00352 /* IMPLEMENTATION ----- */
00353
00354 #include <l4/sys/ipc.h>
00355
00356
00357 L4_INLINE l4_msgtag_t
00358 l4_task_map_u(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
00359              l4_fpage_t snd_fpage, l4_umword_t snd_base, l4_utcb_t *u) L4_NOTHROW
00360 {
00361     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00362     v->mr[0] = L4_TASK_MAP_OP;
00363     v->mr[3] = l4_map_obj_control(0,0);
00364     v->mr[4] = l4_obj_fpage(src_task, 0, L4_CAP_FPAGE_RWS).raw;
00365     v->mr[1] = snd_base;
00366     v->mr[2] = snd_fpage.raw;
00367     return l4_ipc_call(dst_task, u, l4_msgtag(L4_PROTO_TASK, 3, 1, 0), L4_IPC_NEVER);
00368 }
00369
00370 L4_INLINE l4_msgtag_t
00371 l4_task_unmap_u(l4_cap_idx_t task, l4_fpage_t fpage,
00372                l4_umword_t map_mask, l4_utcb_t *u) L4_NOTHROW
00373 {
00374     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00375     v->mr[0] = L4_TASK_UNMAP_OP;
00376     v->mr[1] = map_mask;
00377     v->mr[2] = fpage.raw;
00378     return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 3, 0, 0), L4_IPC_NEVER);
00379 }
00380
00381 L4_INLINE l4_msgtag_t
00382 l4_task_unmap_batch_u(l4_cap_idx_t task, l4_fpage_t const *fpages,
00383                      unsigned num_fpages, l4_umword_t map_mask,
00384                      l4_utcb_t *u) L4_NOTHROW
00385 {
00386     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00387     v->mr[0] = L4_TASK_UNMAP_OP;
00388     v->mr[1] = map_mask;
00389     __builtin_memcpy(&v->mr[2], fpages, num_fpages * sizeof(l4_fpage_t));
00390     return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2 + num_fpages, 0, 0), L4_IPC_NEVER);
00391 }
00392
00393 L4_INLINE l4_msgtag_t
00394 l4_task_cap_valid_u(l4_cap_idx_t task, l4_cap_idx_t cap, l4_utcb_t *u) L4_NOTHROW
00395 {
00396     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00397     v->mr[0] = L4_TASK_CAP_INFO_OP;
00398     v->mr[1] = cap & ~1UL;
00399     return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2, 0, 0), L4_IPC_NEVER);
00400 }
00401
00402 L4_INLINE l4_msgtag_t
00403 l4_task_cap_equal_u(l4_cap_idx_t task, l4_cap_idx_t cap_a,
00404                    l4_cap_idx_t cap_b, l4_utcb_t *u) L4_NOTHROW
00405 {
00406     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00407     v->mr[0] = L4_TASK_CAP_INFO_OP;
00408     v->mr[1] = cap_a;
00409     v->mr[2] = cap_b;
00410     return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 3, 0, 0), L4_IPC_NEVER);
00411 }
00412
00413 L4_INLINE l4_msgtag_t
00414 l4_task_add_ku_mem_u(l4_cap_idx_t task, l4_fpage_t *ku_mem,
00415                     l4_utcb_t *u) L4_NOTHROW
00416 {
00417     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00418     l4_msgtag_t ret;
00419     v->mr[0] = L4_TASK_ADD_KU_MEM_OP;
00420     v->mr[1] = ku_mem->raw;
00421     ret = l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2, 0, 0), L4_IPC_NEVER);
00422     if (!l4_msgtag_has_error(ret))
00423     {
00424         l4_msg_regs_t *v = l4_utcb_mr_u(u);
00425         ku_mem->raw = v->mr[0];
00426     }
00427     return ret;
00428 }
00429
00430
00431
00432 L4_INLINE l4_msgtag_t

```

```

00433 l4_task_map(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
00434             l4_fpage_t snd_fpage, l4_umword_t snd_base) L4_NOTHROW
00435 {
00436     return l4_task_map_u(dst_task, src_task, snd_fpage, snd_base, l4_utcb());
00437 }
00438
00439 L4_INLINE l4_msgtag_t
00440 l4_task_unmap(l4_cap_idx_t task, l4_fpage_t fpage,
00441             l4_umword_t map_mask) L4_NOTHROW
00442 {
00443     return l4_task_unmap_u(task, fpage, map_mask, l4_utcb());
00444 }
00445
00446 L4_INLINE l4_msgtag_t
00447 l4_task_unmap_batch(l4_cap_idx_t task, l4_fpage_t const *fpages,
00448                   unsigned num_fpages, l4_umword_t map_mask) L4_NOTHROW
00449 {
00450     return l4_task_unmap_batch_u(task, fpages, num_fpages, map_mask,
00451                                 l4_utcb());
00452 }
00453
00454 L4_INLINE l4_msgtag_t
00455 l4_task_delete_obj_u(l4_cap_idx_t task, l4_cap_idx_t obj,
00456                    l4_utcb_t *u) L4_NOTHROW
00457 {
00458     return l4_task_unmap_u(task, l4_obj_fpage(obj, 0, L4_CAP_FPAGE_RWSD),
00459                           L4_FP_DELETE_OBJ, u);
00460 }
00461
00462 L4_INLINE l4_msgtag_t
00463 l4_task_delete_obj(l4_cap_idx_t task, l4_cap_idx_t obj) L4_NOTHROW
00464 {
00465     return l4_task_delete_obj_u(task, obj, l4_utcb());
00466 }
00467
00468 L4_INLINE l4_msgtag_t
00469 l4_task_release_cap_u(l4_cap_idx_t task, l4_cap_idx_t cap,
00470                     l4_utcb_t *u) L4_NOTHROW
00471 {
00472     return l4_task_unmap_u(task, l4_obj_fpage(cap, 0, L4_CAP_FPAGE_RWSD),
00473                           L4_FP_ALL_SPACES, u);
00474 }
00475
00476 L4_INLINE l4_msgtag_t
00477 l4_task_release_cap(l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW
00478 {
00479     return l4_task_release_cap_u(task, cap, l4_utcb());
00480 }
00481
00482 L4_INLINE l4_msgtag_t
00483 l4_task_cap_valid(l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW
00484 {
00485     return l4_task_cap_valid_u(task, cap, l4_utcb());
00486 }
00487
00488 L4_INLINE l4_msgtag_t
00489 l4_task_cap_equal(l4_cap_idx_t task, l4_cap_idx_t cap_a,
00490                 l4_cap_idx_t cap_b) L4_NOTHROW
00491 {
00492     return l4_task_cap_equal_u(task, cap_a, cap_b, l4_utcb());
00493 }
00494
00495 L4_INLINE l4_msgtag_t
00496 l4_task_add_ku_mem(l4_cap_idx_t task, l4_fpage_t *ku_mem) L4_NOTHROW
00497 {
00498     return l4_task_add_ku_mem_u(task, ku_mem, l4_utcb());
00499 }
00500

```

## 17.593 l4/cxx/thread File Reference

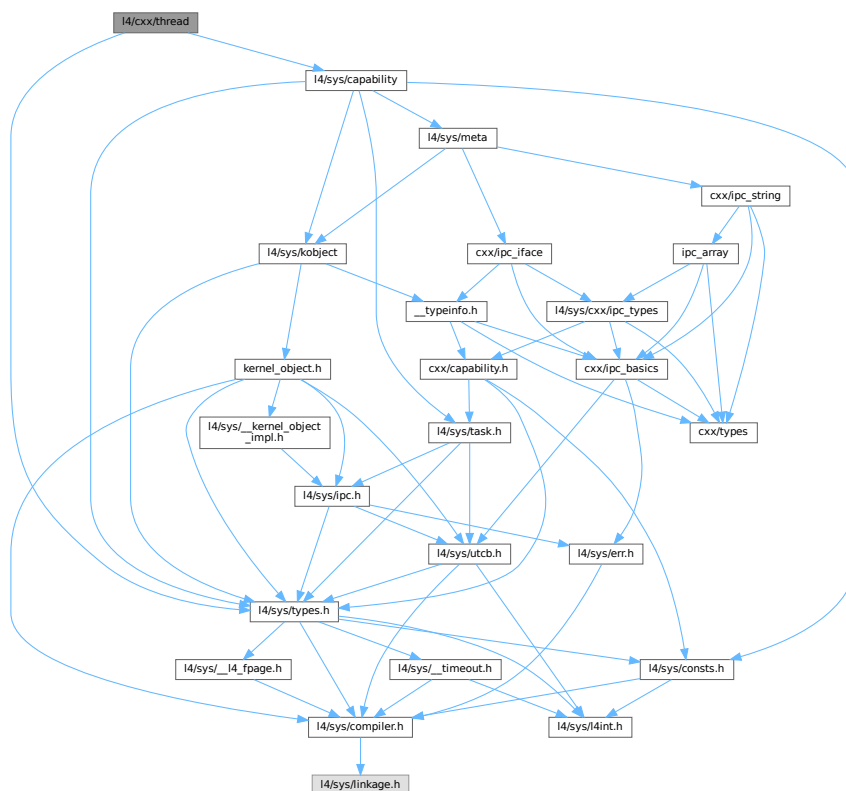
Thread implementation.

```

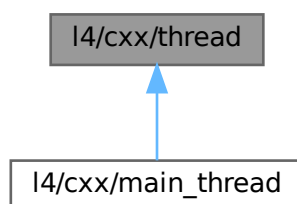
#include <l4/sys/capability>
#include <l4/sys/types.h>

```

Include dependency graph for thread:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace [cxx](#)  
Our C++ library.

## 17.593.1 Detailed Description

Thread implementation.

Definition in file [thread](#).

## 17.594 thread

[Go to the documentation of this file.](#)

```

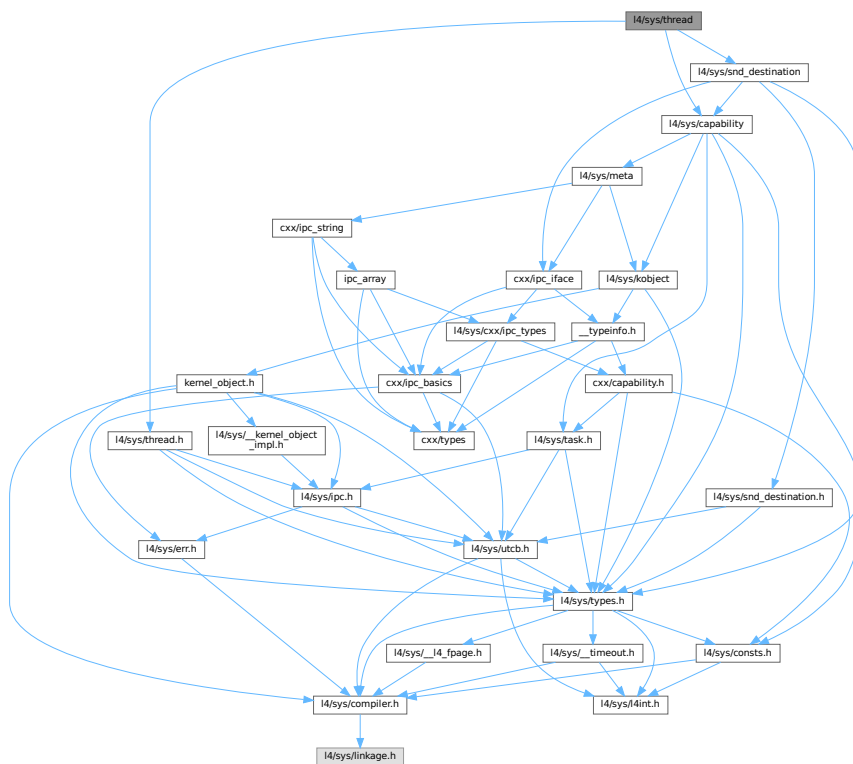
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU Lesser General Public License 2.1.
00007  * Please see the COPYING-LGPL-2.1 file for details.
00008  */
00009
00010 #ifndef CXX_THREAD_H__
00011 #define CXX_THREAD_H__
00012
00013 #include <l4/sys/capability>
00014 #include <l4/sys/types.h>
00015
00016 namespace cxx {
00017
00018     class Thread
00019     {
00020     public:
00021
00022         enum State
00023         {
00024             Dead      = 0,
00025             Running   = 1,
00026             Stopped   = 2,
00027         };
00028
00029         Thread(bool initiate);
00030         Thread(void *stack);
00031         Thread(void *stack, L4::Cap<L4::Thread> const &cap);
00032         virtual ~Thread();
00033         void execute() asm ("L4_Thread_execute");
00034         virtual void run() = 0;
00035         virtual void shutdown() asm ("L4_Thread_shutdown");
00036         void start();
00037         void stop();
00038
00039         L4::Cap<L4::Thread> self() const throw()
00040         { return _cap; }
00041
00042         State state() const
00043         { return _state; }
00044
00045         static void start_cxx_thread(Thread *_this)
00046             asm ("L4_Thread_start_cxx_thread");
00047
00048         static void kill_cxx_thread(Thread *_this)
00049             asm ("L4_Thread_kill_cxx_thread");
00050
00051         static void set_pager(L4::Cap<void>const &p) throw()
00052         { _pager = p; }
00053
00054     private:
00055         int create();
00056
00057         L4::Cap<L4::Thread> _cap;
00058         State _state;
00059
00060     protected:
00061         void *_stack;
00062
00063     private:
00064         static L4::Cap<void> _pager;
00065         static L4::Cap<void> _master;
00066     };
00067
00068 };
00069
00070 #endif /* CXX_THREAD_H__ */
00071

```

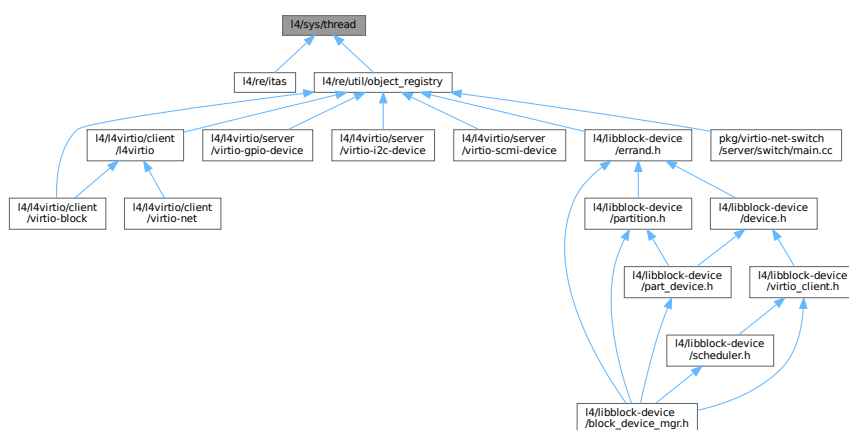
## 17.595 l4/sys/thread File Reference

Common thread related definitions.

Include dependency graph for thread:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class L4::Thread

C++ [L4](#) kernel thread interface, see [Thread](#) for the C interface.

- class [L4::Thread::Attr](#)

[Thread](#) attributes used for [control\(\)](#).

- class [L4::Thread::Modify\\_senders](#)

[Class](#) wrapping a list of rules which modify the sender label of IPC messages inbound to this thread.

## Namespaces

- namespace [L4](#)

[L4](#) low-level kernel interface.

## 17.595.1 Detailed Description

Common thread related definitions.

Definition in file [thread](#).

## 17.596 thread

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=c++: -- Mode: C++ --
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013
00014 #pragma once
00015
00016 #include <l4/sys/capability>
00017 #include <l4/sys/snd_destination>
00018 #include <l4/sys/thread.h>
00019
00020 namespace L4 {
00021
00022 class Thread :
00023     public Kobject_t<Thread, Snd_destination, L4_PROTO_THREAD,
00024         Type_info::Demand_t<1> >
00025 {
00026 public:
00027     l4_msgtag_t ex_regs(l4_addr_t ip, l4_addr_t sp,
00028         l4_umword_t flags,
00029         l4_utcb_t *utcb = l4_utcb()) noexcept
00030     { return l4_thread_ex_regs_u(cap(), ip, sp, flags, utcb); }
00031
00032     l4_msgtag_t ex_regs(l4_addr_t *ip, l4_addr_t *sp,
00033         l4_umword_t *flags,
00034         l4_utcb_t *utcb = l4_utcb()) noexcept
00035     { return l4_thread_ex_regs_ret_u(cap(), ip, sp, flags, utcb); }
00036
00037     class Attr
00038     {
00039     private:
00040         friend class L4::Thread;
00041         l4_utcb_t *_u;
00042
00043     public:
00044         explicit Attr(l4_utcb_t *utcb = l4_utcb()) noexcept : _u(utcb)
00045         { l4_thread_control_start_u(utcb); }
00046
00047         void pager(Cap<void> const &pager) noexcept
00048         { l4_thread_control_pager_u(pager.cap(), _u); }
00049
00050         Cap<void> pager() noexcept

```

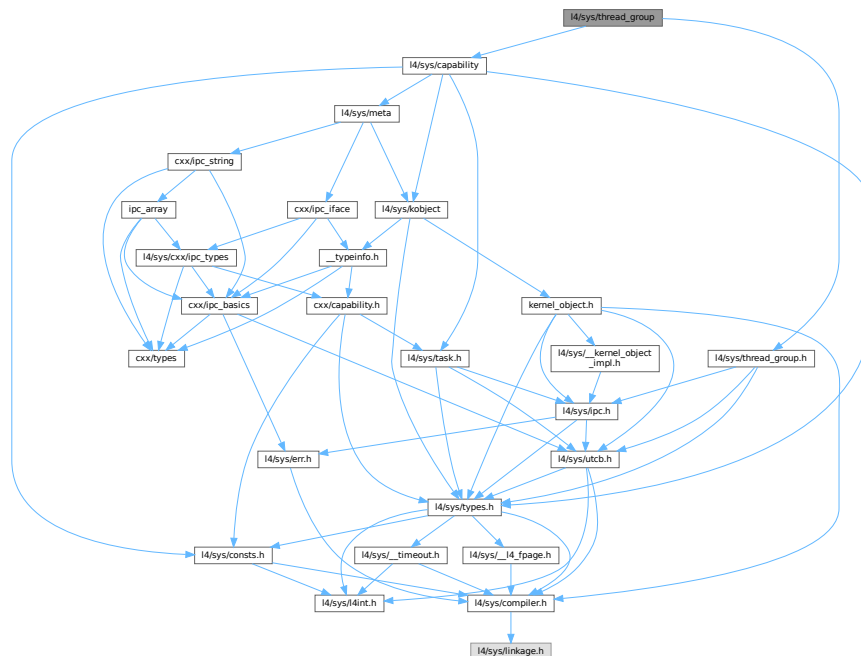
```

00171     { return Cap<void>(l4_utcb_mr_u(_u)->mr[1]); }
00172
00180 void exc_handler(Cap<void> const &exc_handler) noexcept
00181 { l4_thread_control_exc_handler_u(exc_handler.cap(), _u); }
00182
00189 Cap<void> exc_handler() noexcept
00190 { return Cap<void>(l4_utcb_mr_u(_u)->mr[2]); }
00191
00218 void bind(l4_utcb_t *thread_utcb, Cap<Task> const &task) noexcept
00219 { l4_thread_control_bind_u(thread_utcb, task.cap(), _u); }
00220
00224 void alien(int on) noexcept
00225 { l4_thread_control_alien_u(_u, on); }
00226 };
00227
00243 l4_msgtag_t control(Attr const &attr) noexcept
00244 { return l4_thread_control_commit_u(cap(), attr._u); }
00245
00253 l4_msgtag_t switch_to(l4_utcb_t *utcb = l4_utcb()) noexcept
00254 { return l4_thread_switch_u(cap(), utcb); }
00255
00264 l4_msgtag_t stats_time(l4_kernel_clock_t *us,
00265                        l4_utcb_t *utcb = l4_utcb()) noexcept
00266 { return l4_thread_stats_time_u(cap(), us, utcb); }
00267
00283 l4_msgtag_t vcpu_resume_start(l4_utcb_t *utcb = l4_utcb()) noexcept
00284 { return l4_thread_vcpu_resume_start_u(utcb); }
00285
00334 l4_msgtag_t vcpu_resume_commit(l4_msgtag_t tag,
00335                                l4_utcb_t *utcb = l4_utcb()) noexcept
00336 { return l4_thread_vcpu_resume_commit_u(cap(), tag, utcb); }
00337
00358 l4_msgtag_t vcpu_control(l4_addr_t vcpu_state, l4_utcb_t *utcb = l4_utcb())
00359     noexcept
00360 { return l4_thread_vcpu_control_u(cap(), vcpu_state, utcb); }
00361
00398 l4_msgtag_t vcpu_control_ext(l4_addr_t ext_vcpu_state,
00399                              l4_utcb_t *utcb = l4_utcb()) noexcept
00400 { return l4_thread_vcpu_control_ext_u(cap(), ext_vcpu_state, utcb); }
00401
00427 l4_msgtag_t register_del_irq(Cap<Irq> irq, l4_utcb_t *u = l4_utcb()) noexcept
00428 { return l4_thread_register_del_irq_u(cap(), irq.cap(), u); }
00429
00448 class Modify_senders
00449 {
00450 private:
00451     friend class Thread;
00452     l4_utcb_t *utcb;
00453     unsigned cnt;
00454
00455 public:
00456     explicit Modify_senders(l4_utcb_t *u = l4_utcb()) noexcept
00457         : utcb(u), cnt(1)
00458     {
00459         l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_MODIFY_SENDER_OP;
00460     }
00461
00481 int add(l4_umword_t match_mask, l4_umword_t match,
00482         l4_umword_t del_bits, l4_umword_t add_bits) noexcept
00483 {
00484     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00485     if (cnt >= L4_UTCB_GENERIC_DATA_SIZE - 4)
00486         return -L4_ENOMEM;
00487     m->mr[cnt++] = match_mask;
00488     m->mr[cnt++] = match;
00489     m->mr[cnt++] = del_bits;
00490     m->mr[cnt++] = add_bits;
00491     return 0;
00492 }
00493 };
00494
00525 l4_msgtag_t modify_senders(Modify_senders const &todo) noexcept
00526 {
00527     return l4_ipc_call(cap(), todo.utcb, l4_msgtag(L4_PROTO_THREAD, todo.cnt, 0, 0), L4_IPC_NEVER);
00528 }
00529
00553 l4_msgtag_t register_doorbell_irq(Cap<Irq> irq, l4_utcb_t *u = l4_utcb()) noexcept
00554 { return l4_thread_register_doorbell_irq_u(cap(), irq.cap(), u); }
00555 };
00556 }

```

## 17.597 l4/sys/thread\_group File Reference

```
#include <l4/sys/capability>
#include <l4/sys/thread_group.h>
Include dependency graph for thread_group:
```



### Data Structures

- class [L4::Thread\\_group](#)  
C++ [L4](#) kernel thread group interface, see [Thread groups](#) for the C interface.

### Namespaces

- namespace [L4](#)  
[L4](#) low-level kernel interface.

## 17.598 thread\_group

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2022-2025 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00005  *             Frank Mehnert <frank.mehnert@kernkonzept.com>
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00012 #pragma once
00013
00014 #include <l4/sys/capability>
00015 #include <l4/sys/thread_group.h>
```



```

00016
00017 namespace L4 {
00018
00034 class L4_EXPORT Thread_group :
00035     public Kobject_t<Thread_group, Snd_destination, L4_PROTO_THREAD_GROUP,
00036         Type_info::Demand_t<1>
00037 {
00038 public:
00053     l4_msgtag_t add(Cap<Thread> thread, l4_utcb_t *utcb = l4_utcb()) noexcept
00054     { return l4_thread_group_add_u(cap(), thread.cap(), utcb); }
00055
00067     l4_msgtag_t remove(Cap<Thread> thread, l4_utcb_t *utcb = l4_utcb()) noexcept
00068     { return l4_thread_group_remove_u(cap(), thread.cap(), utcb); }
00069 };
00070
00071 }

```

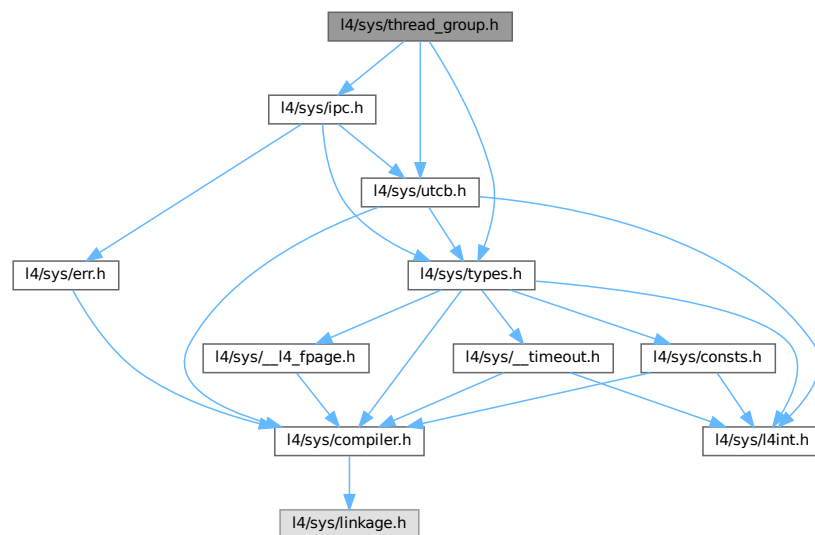
## 17.599 l4/sys/thread\_group.h File Reference

```
#include <l4/sys/types.h>
```

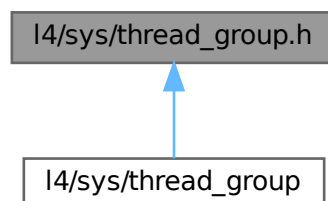
```
#include <l4/sys/utcb.h>
```

```
#include <l4/sys/ipc.h>
```

Include dependency graph for thread\_group.h:



This graph shows which files directly or indirectly include this file:



## Functions

- [l4\\_msgtag\\_t l4\\_thread\\_group\\_add \(l4\\_cap\\_idx\\_t tg, l4\\_cap\\_idx\\_t thread\) L4\\_NOTHROW](#)  
*Add thread to a thread group.*
- [l4\\_msgtag\\_t l4\\_thread\\_group\\_remove \(l4\\_cap\\_idx\\_t tg, l4\\_cap\\_idx\\_t thread\) L4\\_NOTHROW](#)  
*Remove thread from a thread group.*

## 17.600 thread\_group.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (C) 2022-2025 Kernkonzept GmbH.
00003  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00004  *           Frank Mehnert <frank.mehnert@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00011 #pragma once
00012
00013 #include <l4/sys/types.h>
00014 #include <l4/sys/utcb.h>
00015
00033
00034 enum l4_thread_group_ops
00035 {
00036     L4_THREAD_GROUP_ADD_OP = 2UL,
00037     L4_THREAD_GROUP_REMOVE_OP = 3UL,
00038 };
00039
00040 enum l4_thread_group_policy
00041 {
00042     L4_THREAD_GROUP_POLICY_STRICT_CORE_LOCAL = 0,
00043     L4_THREAD_GROUP_POLICY_SOFT_CORE_LOCAL = 1,
00044 };
00045
00052 L4_INLINE l4_msgtag_t
00053 l4_thread_group_add(l4_cap_idx_t tg,
00054                    l4_cap_idx_t thread) L4_NOTHROW;
00055
00059 L4_INLINE l4_msgtag_t
00060 l4_thread_group_add_u(l4_cap_idx_t tg,
00061                      l4_cap_idx_t thread,
00062                      l4_utcb_t *utcb) L4_NOTHROW;
00063
00070 L4_INLINE l4_msgtag_t
00071 l4_thread_group_remove(l4_cap_idx_t tg,
00072                       l4_cap_idx_t thread) L4_NOTHROW;
00073
00077 L4_INLINE l4_msgtag_t
00078 l4_thread_group_remove_u(l4_cap_idx_t tg,
00079                          l4_cap_idx_t thread,
00080                          l4_utcb_t *utcb) L4_NOTHROW;
00081
00082
00083 /* IMPLEMENTATION ----- */
00084
00085 #include <l4/sys/ipc.h>
00086
00087 L4_INLINE l4_msgtag_t
00088 l4_thread_group_add_u(l4_cap_idx_t tg,
00089                      l4_cap_idx_t thread,
00090                      l4_utcb_t *utcb) L4_NOTHROW
00091 {
00092     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00093     v->mr[0] = L4_THREAD_GROUP_ADD_OP;
00094     v->mr[1] = l4_map_obj_control(0, 0);
00095     v->mr[2] = l4_obj_fpage(thread, 0, L4_CAP_FPAGE_RWS).raw;
00096     return l4_ipc_call(tg, utcb,
00097                       l4_msgtag(L4_PROTO_THREAD_GROUP, 1, 1, 0), L4_IPC_NEVER);
00098 }
00099
00100 L4_INLINE l4_msgtag_t
00101 l4_thread_group_remove_u(l4_cap_idx_t tg,
00102                          l4_cap_idx_t thread,
00103                          l4_utcb_t *utcb) L4_NOTHROW
00104 {
00105     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);

```

```

00106 v->mr[0] = L4_THREAD_GROUP_REMOVE_OP;
00107 v->mr[1] = l4_map_obj_control(0, 0);
00108 v->mr[2] = l4_obj_fpage(thread, 0, L4_CAP_FPAGE_RWS).raw;
00109 return l4_ipc_call(tg, utcb,
00110                  l4_msgtag(L4_PROTO_THREAD_GROUP, 1, 1, 0), L4_IPC_NEVER);
00111 }
00112
00113 L4_INLINE l4_msgtag_t
00114 l4_thread_group_add(l4_cap_idx_t tg,
00115                   l4_cap_idx_t thread) L4_NOTHROW
00116 {
00117     return l4_thread_group_add_u(tg, thread, l4_utcb());
00118 }
00119
00120 L4_INLINE l4_msgtag_t
00121 l4_thread_group_remove(l4_cap_idx_t tg,
00122                      l4_cap_idx_t thread) L4_NOTHROW
00123 {
00124     return l4_thread_group_remove_u(tg, thread, l4_utcb());
00125 }

```

## 17.601 l4/sys/typeinfo\_svr File Reference

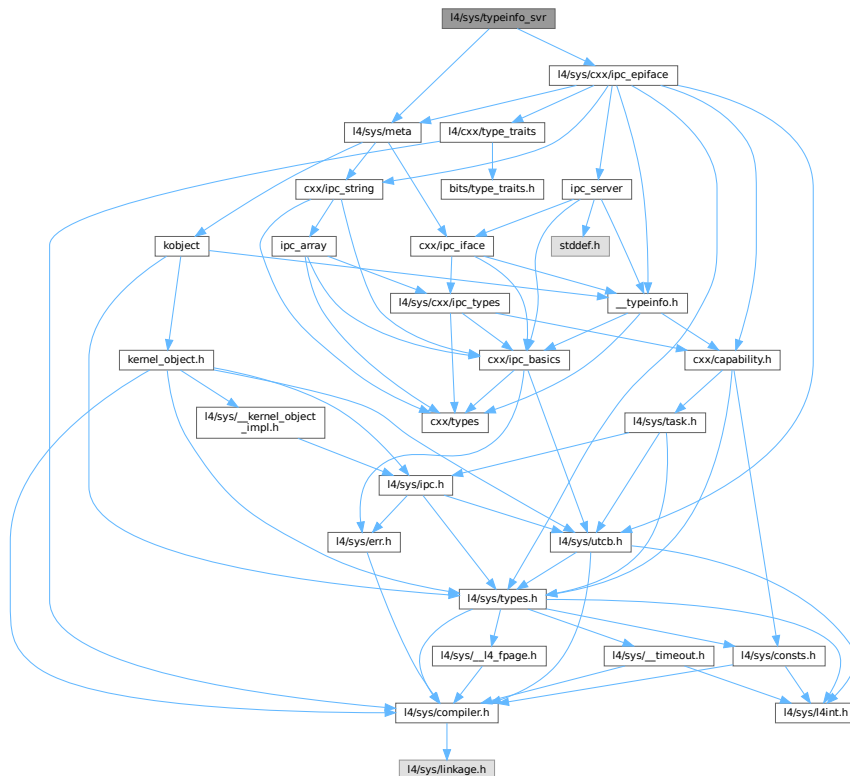
Type information server template.

```

#include <l4/sys/meta>
#include <l4/sys/cxx/ipc_epiface>

```

Include dependency graph for typeinfo\_svr:



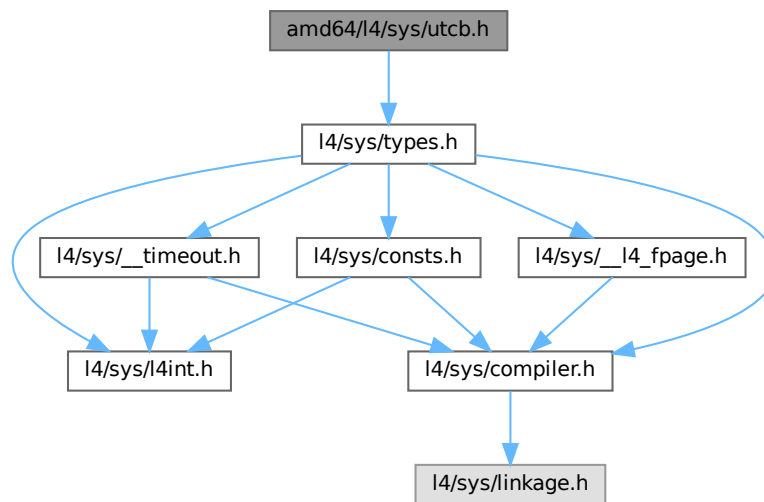


## 17.603 amd64/l4/sys/utcb.h File Reference

UTCB definitions for AMD64.

```
#include <l4/sys/types.h>
```

Include dependency graph for utcb.h:



### Data Structures

- struct [l4\\_exc\\_regs\\_t](#)  
*UTCB structure for exceptions.*

### Typedefs

- typedef struct `l4_exc_regs_t` [l4\\_exc\\_regs\\_t](#)  
*UTCB structure for exceptions.*

### Enumerations

- enum [L4\\_utcb\\_consts\\_amd64](#)  
*UTCB constants for AMD64.*

## Functions

- [l4\\_umword\\_t l4\\_utcb\\_exc\\_pc](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#)  
Access function to get the program counter of the exception state.
- void [l4\\_utcb\\_exc\\_pc\\_set](#) ([l4\\_exc\\_regs\\_t](#) \*u, [l4\\_addr\\_t](#) pc) [L4\\_NOTHROW](#)  
Set the program counter register in the exception state.
- [l4\\_umword\\_t l4\\_utcb\\_exc\\_typeval](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#)  
Get the value out of an exception UTCB that describes the type of exception.
- int [l4\\_utcb\\_exc\\_is\\_pf](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#)  
Check whether an exception IPC is a page fault.
- [l4\\_addr\\_t l4\\_utcb\\_exc\\_pfa](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#)  
Function to get the L4 style page fault address out of an exception.
- int [l4\\_utcb\\_exc\\_is\\_ex\\_regs\\_exception](#) ([l4\\_exc\\_regs\\_t](#) const \*u) [L4\\_NOTHROW](#)  
Check whether an exception IPC was triggered via [l4\\_thread\\_ex\\_regs\(\)](#).

### 17.603.1 Detailed Description

UTCB definitions for AMD64.

Definition in file [utcb.h](#).

## 17.604 utcb.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 /*****
00014  #ifndef __L4_SYS__INCLUDE__ARCH_AMD64__UTCB_H__
00015  #define __L4_SYS__INCLUDE__ARCH_AMD64__UTCB_H__
00016
00017  #include <l4/sys/types.h>
00018
00023
00028 enum l4_utcb_consts_amd64
00029 {
00030     L4_UTCB_EXCEPTION_REGS_SIZE      = 26,
00031     L4_UTCB_GENERIC_DATA_SIZE        = 63,
00032     L4_UTCB_GENERIC_BUFFERS_SIZE     = 58,
00033
00034     L4_UTCB_MSG_REGS_OFFSET          = 0,
00035     L4_UTCB_BUF_REGS_OFFSET          = 64 * sizeof(l4_umword_t),
00036     L4_UTCB_THREAD_REGS_OFFSET       = 123 * sizeof(l4_umword_t),
00037
00038     L4_UTCB_INHERIT_FPU               = 1UL < 24,
00039     L4_UTCB_OFFSET                   = 1024,
00040 };
00041
00046 typedef struct l4_exc_regs_t
00047 {
00048     l4_umword_t r15;
00049     l4_umword_t r14;
00050     l4_umword_t r13;
00051     l4_umword_t r12;
00052     l4_umword_t r11;
00053     l4_umword_t r10;
00054     l4_umword_t r9;
00055     l4_umword_t r8;
00056     l4_umword_t rdi;
00057     l4_umword_t rsi;
00058     l4_umword_t rbp;

```

```

00059  l4_umword_t pfa;
00060  l4_umword_t rbx;
00061  l4_umword_t rdx;
00062  l4_umword_t rcx;
00063  l4_umword_t rax;
00064
00065  l4_umword_t trapno;
00066  l4_umword_t err;
00067  l4_umword_t ip;
00068  l4_umword_t dummy1;
00069  l4_umword_t flags;
00070  l4_umword_t sp;
00071  l4_umword_t ss;
00072  l4_umword_t fs_base;
00073  l4_umword_t gs_base;
00074  l4_uint16_t ds, es, fs, gs;
00075 } l4_exc_regs_t;
00076
00077
00078 #include_next <l4/sys/utcb.h>
00079
00080 /*
00081  * =====
00082  * Implementations.
00083  */
00084
00085 L4_INLINE l4_utcb_t *l4_utcb_direct(void) L4_NOTHROW
00086 {
00087     l4_utcb_t *res;
00088     __asm__ ( "mov %%gs:0, %0 \n" : "=r"(res));
00089     return res;
00090 }
00091
00092 L4_INLINE l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u) L4_NOTHROW
00093 {
00094     return u->ip;
00095 }
00096
00097 L4_INLINE void l4_utcb_exc_pc_set(l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW
00098 {
00099     u->ip = pc;
00100 }
00101
00102 L4_INLINE l4_umword_t l4_utcb_exc_typeval(l4_exc_regs_t const *u) L4_NOTHROW
00103 {
00104     return u->trapno;
00105 }
00106
00107 L4_INLINE int l4_utcb_exc_is_pf(l4_exc_regs_t const *u) L4_NOTHROW
00108 {
00109     return u->trapno == 14;
00110 }
00111
00112 L4_INLINE l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u) L4_NOTHROW
00113 {
00114     return (u->pfa & ~7UL) | (u->err & 2);
00115 }
00116
00117 L4_INLINE int l4_utcb_exc_is_ex_regs_exception(l4_exc_regs_t const *u) L4_NOTHROW
00118 {
00119     return l4_utcb_exc_typeval(u) == 0xff;
00120 }
00121
00122 #endif /* ! __L4_SYS__INCLUDE__ARCH_AMD64__UTCB_H__ */

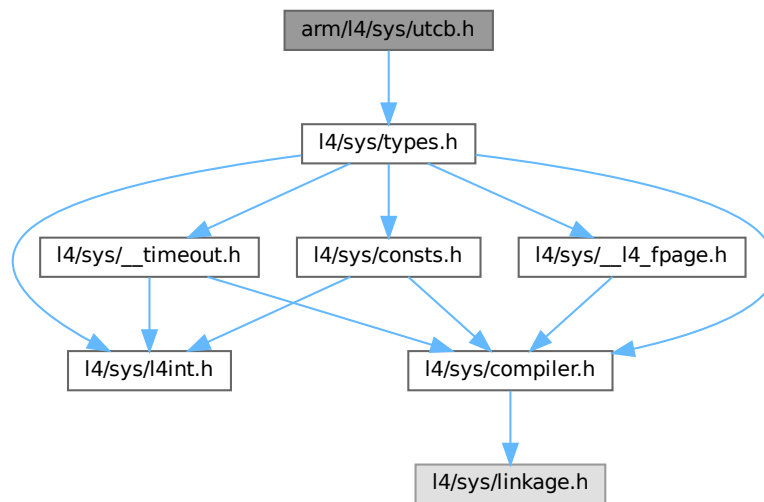
```

## 17.605 arm/l4/sys/utcb.h File Reference

UTCB definitions for ARM.

```
#include <l4/sys/types.h>
```

Include dependency graph for utcb.h:



## Data Structures

- struct `l4_exc_regs_t`  
*UTCB structure for exceptions.*

## Typedefs

- typedef struct `l4_exc_regs_t` `l4_exc_regs_t`  
*UTCB structure for exceptions.*

## Enumerations

- enum `L4_utcb_consts_arm`  
*UTCB constants for ARM.*

## Functions

- `l4_umword_t l4_utcb_exc_pc (l4_exc_regs_t const *u) L4_NOTHROW`  
*Access function to get the program counter of the exception state.*
- `void l4_utcb_exc_pc_set (l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW`  
*Set the program counter register in the exception state.*
- `l4_umword_t l4_utcb_exc_typeval (l4_exc_regs_t const *u) L4_NOTHROW`  
*Get the value out of an exception UTCB that describes the type of exception.*
- `int l4_utcb_exc_is_pf (l4_exc_regs_t const *u) L4_NOTHROW`  
*Check whether an exception IPC is a page fault.*
- `l4_addr_t l4_utcb_exc_pfa (l4_exc_regs_t const *u) L4_NOTHROW`  
*Function to get the L4 style page fault address out of an exception.*
- `int l4_utcb_exc_is_ex_regs_exception (l4_exc_regs_t const *u) L4_NOTHROW`  
*Check whether an exception IPC was triggered via `l4_thread_ex_regs()`.*



## 17.605.1 Detailed Description

UTCB definitions for ARM.

Definition in file [utcb.h](#).

## 17.606 utcb.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #ifndef __L4_SYS__INCLUDE__ARCH_ARM__UTCB_H__
00014 #define __L4_SYS__INCLUDE__ARCH_ARM__UTCB_H__
00015
00016 #include <l4/sys/types.h>
00017
00022
00027 typedef struct l4_exc_regs_t
00028 {
00029     l4_umword_t pfa;
00030     l4_umword_t err;
00031
00032     l4_umword_t r[13];
00033     l4_umword_t sp;
00034     l4_umword_t ulr;
00035     l4_umword_t _dummy1;
00036     l4_umword_t pc;
00037     l4_umword_t cpsr;
00038     l4_umword_t tpidruro;
00039     l4_umword_t tpidrurw;
00040 } l4_exc_regs_t;
00041
00047 enum L4_utcb_consts_arm
00048 {
00049     L4_UTCB_EXCEPTION_REGS_SIZE    = sizeof(l4_exc_regs_t) / sizeof(l4_umword_t),
00050     L4_UTCB_GENERIC_DATA_SIZE      = 63,
00051     L4_UTCB_GENERIC_BUFFERS_SIZE   = 58,
00052
00053     L4_UTCB_MSG_REGS_OFFSET        = 0,
00054     L4_UTCB_BUF_REGS_OFFSET        = 64 * sizeof(l4_umword_t),
00055     L4_UTCB_THREAD_REGS_OFFSET     = 123 * sizeof(l4_umword_t),
00056
00057     L4_UTCB_INHERIT_FPU            = 1UL < 24,
00058
00059     L4_UTCB_OFFSET                 = 512,
00060 };
00061
00062 #include_next <l4/sys/utcb.h>
00063
00064 /*
00065  * =====
00066  * Implementations.
00067  */
00068
00069 #ifdef __GNUC__
00070 L4_INLINE l4_utcb_t *l4_utcb_direct(void) L4_NOTHROW
00071 {
00072     #if defined(__ARM_ARCH) && __ARM_ARCH >= 7
00073         l4_utcb_t *utcb;
00074         __asm__ ("mrc p15, 0, %0, c13, c0, 2" : "=r" (utcb)); // TPIDRURW
00075     #else
00076         register l4_utcb_t *utcb __asm__ ("r0");
00077         __asm__ ("mov lr, pc\n"
00078                 "mvn pc, #0xff\n"
00079                 : "=r"(utcb) : : "lr"); // write 0xfffff00 to pc
00080     #endif
00081     return utcb;
00082 }
00083 #endif
00084
00085 L4_INLINE l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u) L4_NOTHROW
00086 {

```

```

00087     return u->pc;
00088 }
00089
00090 L4_INLINE void l4_utcb_exc_pc_set(l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW
00091 {
00092     u->pc = pc;
00093 }
00094
00095 L4_INLINE l4_umword_t l4_utcb_exc_typeval(l4_exc_regs_t const *u) L4_NOTHROW
00096 {
00097     return u->err >> 26;
00098 }
00099
00100 L4_INLINE int l4_utcb_exc_is_pf(l4_exc_regs_t const *u) L4_NOTHROW
00101 {
00102     return ((u->err >> 26) & 0x30) == 0x20;
00103 }
00104
00105 L4_INLINE l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u) L4_NOTHROW
00106 {
00107     return (u->pfa & ~7UL) | ((u->err >> 5) & 2);
00108 }
00109
00110 L4_INLINE int l4_utcb_exc_is_ex_regs_exception(l4_exc_regs_t const *u) L4_NOTHROW
00111 {
00112     return l4_utcb_exc_typeval(u) == 0x3e;
00113 }
00114
00115 #endif /* ! __L4_SYS__INCLUDE__ARCH_ARM__UTCB_H__ */

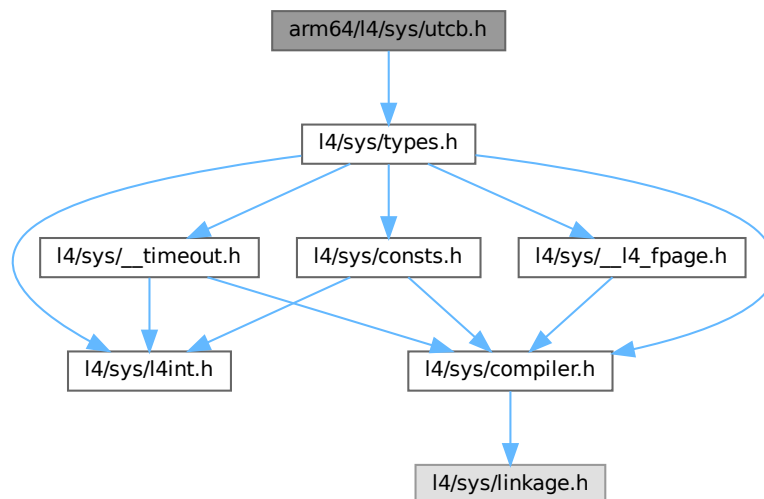
```

## 17.607 arm64/l4/sys/utcb.h File Reference

UTCB definitions for ARM64.

```
#include <l4/sys/types.h>
```

Include dependency graph for utcb.h:



### Data Structures

- struct `l4_exc_regs_t`  
*UTCB structure for exceptions.*

## Typedefs

- typedef struct l4\_exc\_regs\_t l4\_exc\_regs\_t  
*UTCB structure for exceptions.*

## Enumerations

- enum L4\_utcb\_consts\_arm64  
*UTCB constants for ARM64.*

## Functions

- l4\_umword\_t l4\_utcb\_exc\_pc (l4\_exc\_regs\_t const \*u) L4\_NOTHROW  
*Access function to get the program counter of the exception state.*
- void l4\_utcb\_exc\_pc\_set (l4\_exc\_regs\_t \*u, l4\_addr\_t pc) L4\_NOTHROW  
*Set the program counter register in the exception state.*
- l4\_umword\_t l4\_utcb\_exc\_typeval (l4\_exc\_regs\_t const \*u) L4\_NOTHROW  
*Get the value out of an exception UTCB that describes the type of exception.*
- int l4\_utcb\_exc\_is\_pf (l4\_exc\_regs\_t const \*u) L4\_NOTHROW  
*Check whether an exception IPC is a page fault.*
- l4\_addr\_t l4\_utcb\_exc\_pfa (l4\_exc\_regs\_t const \*u) L4\_NOTHROW  
*Function to get the L4 style page fault address out of an exception.*
- int l4\_utcb\_exc\_is\_ex\_regs\_exception (l4\_exc\_regs\_t const \*u) L4\_NOTHROW  
*Check whether an exception IPC was triggered via l4\_thread\_ex\_regs().*

## 17.607.1 Detailed Description

UTCB definitions for ARM64.

Definition in file [utcb.h](#).

## 17.608 utcb.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #ifndef __L4_SYS__INCLUDE__ARCH_ARM64__UTCB_H__
00014 #define __L4_SYS__INCLUDE__ARCH_ARM64__UTCB_H__
00015
00016 #include <l4/sys/types.h>
00017
00022
00027 typedef struct l4_exc_regs_t
00028 {
00029     l4_umword_t eret_work;
00030     l4_umword_t r[31];
00031     l4_umword_t reserved;
00032     l4_umword_t err;
00033
00034     l4_umword_t pfa;
00035     l4_umword_t sp;

```

```

00036 union { l4_umword_t ip; l4_umword_t pc; }; /* aliases for PC */
00037 union { l4_umword_t flags; l4_umword_t pstate; }; /* aliases for PSTATE (PSR) */
00038 l4_umword_t tpidruro;
00039 l4_umword_t tpidrurw;
00040 } l4_exc_regs_t;
00041
00047 enum L4_utcb_consts_arm64
00048 {
00049     L4_UTCB_EXCEPTION_REGS_SIZE    = sizeof(l4_exc_regs_t) / sizeof(l4_umword_t),
00050     L4_UTCB_GENERIC_DATA_SIZE      = 63,
00051     L4_UTCB_GENERIC_BUFFERS_SIZE   = 58,
00052
00053     L4_UTCB_MSG_REGS_OFFSET        = 0,
00054     L4_UTCB_BUF_REGS_OFFSET        = 64 * sizeof(l4_umword_t),
00055     L4_UTCB_THREAD_REGS_OFFSET     = 123 * sizeof(l4_umword_t),
00056
00057     L4_UTCB_INHERIT_FPU            = 1UL < 24,
00058
00059     L4_UTCB_OFFSET                 = 1024,
00060 };
00061
00062 #include_next <l4/sys/utcb.h>
00063
00064 /*
00065  * =====
00066  * Implementations.
00067  */
00068
00069 #ifdef __GNUC__
00070 L4_INLINE l4_utcb_t *l4_utcb_direct(void) L4_NOTHROW
00071 {
00072     l4_utcb_t *utcb;
00073     __asm__ ("mrs %0, TPIDRRO_EL0" : "=r" (utcb));
00074     return utcb;
00075 }
00076 #endif
00077
00078 L4_INLINE l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u) L4_NOTHROW
00079 {
00080     return u->pc;
00081 }
00082
00083 L4_INLINE void l4_utcb_exc_pc_set(l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW
00084 {
00085     u->pc = pc;
00086 }
00087
00088 L4_INLINE l4_umword_t l4_utcb_exc_typeval(l4_exc_regs_t const *u) L4_NOTHROW
00089 {
00090     return u->err >> 26;
00091 }
00092
00093 L4_INLINE int l4_utcb_exc_is_pf(l4_exc_regs_t const *u) L4_NOTHROW
00094 {
00095     return ((u->err >> 26) & 0x30) == 0x20;
00096 }
00097
00098 L4_INLINE l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u) L4_NOTHROW
00099 {
00100     return (u->pfa & ~7UL) | ((u->err >> 5) & 2);
00101 }
00102
00103 L4_INLINE int l4_utcb_exc_is_ex_regs_exception(l4_exc_regs_t const *u) L4_NOTHROW
00104 {
00105     return (u->err >> 26) == 0x3e;
00106 }
00107
00108 #endif /* ! __L4_SYS__INCLUDE__ARCH_ARM64__UTCB_H__ */

```

## 17.609 l4/sys/utcb.h File Reference

UTCB definitions.

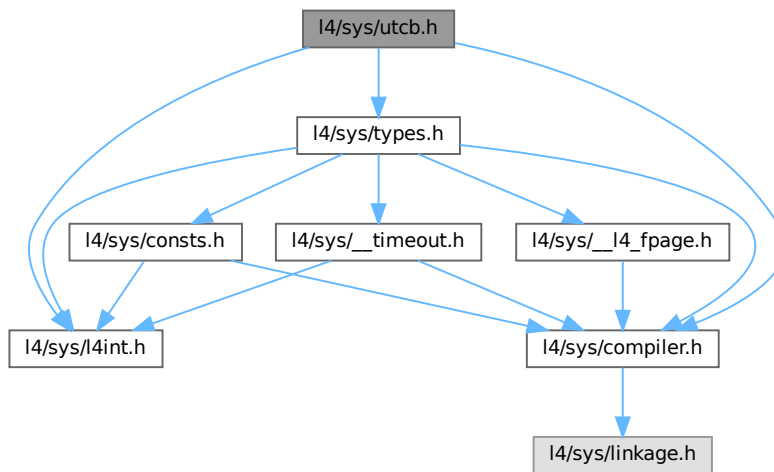
```

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>

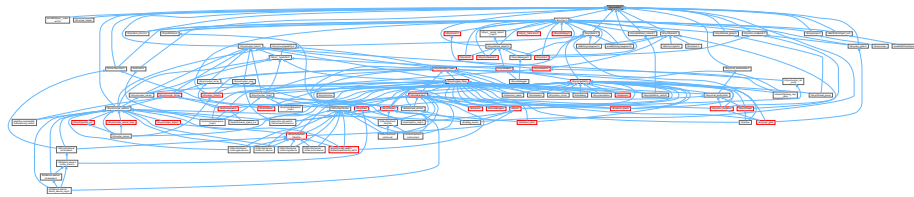
```

```
#include <l4/sys/l4int.h>
```

Include dependency graph for utcb.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- union `I4_msg_regs_t`  
*Encapsulation of the message-register block in the UTCB.*
- struct `I4_buf_regs_t`  
*Encapsulation of the buffer-registers block in the UTCB.*
- struct `I4_thread_regs_t`  
*Encapsulation of the thread-control-register block of the UTCB.*

## Typedefs

- typedef struct `I4_utcb_t` `I4_utcb_t`  
*Opaque type for the UTCB.*
- typedef union `I4_msg_regs_t` `I4_msg_regs_t`  
*Encapsulation of the message-register block in the UTCB.*
- typedef struct `I4_buf_regs_t` `I4_buf_regs_t`  
*Encapsulation of the buffer-registers block in the UTCB.*
- typedef struct `I4_thread_regs_t` `I4_thread_regs_t`  
*Encapsulation of the thread-control-register block of the UTCB.*

## Functions

- `l4_utcb_t * l4_utcb` (void) `L4_NOTHROW` `L4_PURE`  
*Get the UTCB address.*
- `l4_msg_regs_t * l4_utcb_mr` (void) `L4_NOTHROW` `L4_PURE`  
*Get the message-register block of a UTCB.*
- `l4_buf_regs_t * l4_utcb_br` (void) `L4_NOTHROW` `L4_PURE`  
*Get the buffer-register block of a UTCB.*
- `l4_thread_regs_t * l4_utcb_tcr` (void) `L4_NOTHROW` `L4_PURE`  
*Get the thread-control-register block of a UTCB.*
- `l4_exc_regs_t * l4_utcb_exc` (void) `L4_NOTHROW` `L4_PURE`  
*Get the message-register block of a UTCB (for an exception IPC).*
- `l4_umword_t l4_utcb_exc_pc` (`l4_exc_regs_t` const \*u) `L4_NOTHROW` `L4_PURE`  
*Access function to get the program counter of the exception state.*
- `void l4_utcb_exc_pc_set` (`l4_exc_regs_t` \*u, `l4_addr_t` pc) `L4_NOTHROW`  
*Set the program counter register in the exception state.*
- `unsigned long l4_utcb_exc_typeval` (`l4_exc_regs_t` const \*u) `L4_NOTHROW` `L4_PURE`  
*Get the value out of an exception UTCB that describes the type of exception.*
- `int l4_utcb_exc_is_pf` (`l4_exc_regs_t` const \*u) `L4_NOTHROW` `L4_PURE`  
*Check whether an exception IPC is a page fault.*
- `l4_addr_t l4_utcb_exc_pfa` (`l4_exc_regs_t` const \*u) `L4_NOTHROW` `L4_PURE`  
*Function to get the L4 style page fault address out of an exception.*
- `int l4_utcb_exc_is_ex_regs_exception` (`l4_exc_regs_t` const \*u) `L4_NOTHROW` `L4_PURE`  
*Check whether an exception IPC was triggered via `l4_thread_ex_regs()`.*
- `void l4_utcb_inherit_fpu` (int switch\_on) `L4_NOTHROW`  
*Enable or disable inheritance of FPU state to receiver.*
- `l4_timeout_s l4_timeout_abs` (`l4_kernel_clock_t` pint, int br) `L4_NOTHROW`  
*Set an absolute timeout.*
- `unsigned l4_utcb_mr64_idx` (unsigned idx) `L4_NOTHROW`  
*Get index into 64bit message registers alias from native-sized index.*

## 17.609.1 Detailed Description

UTCB definitions.

Definition in file `utcb.h`.

## 17.610 utcb.h

[Go to the documentation of this file.](#)

```

00001 /*****
00002  */
00003 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004 *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00005 *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00006 *      economic rights: Technische Universität Dresden (Germany)
00007 *
00008 * License: see LICENSE.spdx (in this directory or the directories above)
00009 */
00010 /*****
00011 #ifndef _L4_SYS_UTCB_H
00012 #define _L4_SYS_UTCB_H
00013 #include <l4/sys/types.h>

```

```

00020 #include <l4/sys/compiler.h>
00021 #include <l4/sys/l4int.h>
00022
00047
00056 typedef struct l4_utcb_t l4_utcb_t;
00057
00062
00067 typedef union l4_msg_regs_t
00068 {
00069     l4_umword_t mr[L4_UTCB_GENERIC_DATA_SIZE];
00070     l4_uint64_t mr64[L4_UTCB_GENERIC_DATA_SIZE / (sizeof(l4_uint64_t)/sizeof(l4_umword_t))];
00071 } l4_msg_regs_t;
00072
00082 typedef struct l4_buf_regs_t
00083 {
00085     l4_umword_t bdr;
00086
00088     l4_umword_t br[L4_UTCB_GENERIC_BUFFERS_SIZE];
00089 } l4_buf_regs_t;
00090
00099 typedef struct l4_thread_regs_t
00100 {
00106     l4_umword_t error;
00107
00120     l4_umword_t free_marker;
00121
00123     l4_umword_t user[3];
00124 } l4_thread_regs_t;
00125
00126 L4_BEGIN_DECLS
00127
00138 L4_CV l4_utcb_t *l4_utcb_wrap(void) L4_NOTHROW L4_PURE;
00139
00145 L4_INLINE l4_utcb_t *l4_utcb_direct(void) L4_NOTHROW L4_PURE;
00146
00151 L4_INLINE l4_utcb_t *l4_utcb(void) L4_NOTHROW L4_PURE;
00152
00158 L4_INLINE l4_msg_regs_t *l4_utcb_mr(void) L4_NOTHROW L4_PURE;
00159
00164 L4_INLINE l4_msg_regs_t *l4_utcb_mr_u(l4_utcb_t *u) L4_NOTHROW L4_PURE;
00165
00172 L4_INLINE l4_buf_regs_t *l4_utcb_br(void) L4_NOTHROW L4_PURE;
00173
00178 L4_INLINE l4_buf_regs_t *l4_utcb_br_u(l4_utcb_t *u) L4_NOTHROW L4_PURE;
00179
00185 L4_INLINE l4_thread_regs_t *l4_utcb_tcr(void) L4_NOTHROW L4_PURE;
00186
00191 L4_INLINE l4_thread_regs_t *l4_utcb_tcr_u(l4_utcb_t *u) L4_NOTHROW L4_PURE;
00192
00198
00205 L4_INLINE l4_exc_regs_t *l4_utcb_exc(void) L4_NOTHROW L4_PURE;
00206
00211 L4_INLINE l4_exc_regs_t *l4_utcb_exc_u(l4_utcb_t *u) L4_NOTHROW L4_PURE;
00212
00220 L4_INLINE l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u) L4_NOTHROW L4_PURE;
00221
00230 L4_INLINE void l4_utcb_exc_pc_set(l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW;
00231
00236 L4_INLINE unsigned long l4_utcb_exc_typeval(l4_exc_regs_t const *u) L4_NOTHROW L4_PURE;
00237
00247 L4_INLINE int l4_utcb_exc_is_pf(l4_exc_regs_t const *u) L4_NOTHROW L4_PURE;
00248
00253 L4_INLINE l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u) L4_NOTHROW L4_PURE;
00254
00255
00266 L4_INLINE int l4_utcb_exc_is_ex_regs_exception(l4_exc_regs_t const *u) L4_NOTHROW L4_PURE;
00267
00272 L4_INLINE void l4_utcb_inherit_fpu(int switch_on) L4_NOTHROW;
00273
00277 L4_INLINE void l4_utcb_inherit_fpu_u(l4_utcb_t *u, int switch_on) L4_NOTHROW;
00278
00293 L4_INLINE
00294 l4_timeout_s l4_timeout_abs_u(l4_kernel_clock_t pint, int br,
00295                               l4_utcb_t *utcb) L4_NOTHROW;
00309 L4_INLINE
00310 l4_timeout_s l4_timeout_abs(l4_kernel_clock_t pint, int br) L4_NOTHROW;
00311
00319 L4_INLINE
00320 unsigned l4_utcb_mr64_idx(unsigned idx) L4_NOTHROW;
00321
00322 /*****
00323  * Implementations
00324  *****/
00325
00326 L4_INLINE l4_msg_regs_t *l4_utcb_mr_u(l4_utcb_t *u) L4_NOTHROW
00327 { return (l4_msg_regs_t*)((char*)u + L4_UTCB_MSG_REGS_OFFSET); }
00328

```

```

00329 L4_INLINE l4_buf_regs_t *l4_utcb_br_u(l4_utcb_t *u) L4_NOTHROW
00330 { return (l4_buf_regs_t*)((char*)u + L4_UTCB_BUF_REGS_OFFSET); }
00331
00332 L4_INLINE l4_thread_regs_t *l4_utcb_tcr_u(l4_utcb_t *u) L4_NOTHROW
00333 { return (l4_thread_regs_t*)((char*)u + L4_UTCB_THREAD_REGS_OFFSET); }
00334
00335 L4_INLINE l4_exc_regs_t *l4_utcb_exc_u(l4_utcb_t *u) L4_NOTHROW
00336 { return (l4_exc_regs_t*)((char*)u + L4_UTCB_MSG_REGS_OFFSET); }
00337
00338 L4_INLINE void l4_utcb_inherit_fpu_u(l4_utcb_t *u, int switch_on) L4_NOTHROW
00339 {
00340     if (switch_on)
00341         l4_utcb_br_u(u)->bdr |= L4_UTCB_INHERIT_FPU;
00342     else
00343         l4_utcb_br_u(u)->bdr &= ~L4_UTCB_INHERIT_FPU;
00344 }
00345
00346 L4_INLINE l4_utcb_t *l4_utcb(void) L4_NOTHROW
00347 {
00348     #ifdef L4SYS_USE_UTCB_WRAP
00349         return l4_utcb_wrap();
00350     #else
00351         return l4_utcb_direct();
00352     #endif
00353 }
00354
00355
00356
00357
00358 L4_INLINE l4_msg_regs_t *l4_utcb_mr(void) L4_NOTHROW
00359 { return l4_utcb_mr_u(l4_utcb()); }
00360
00361 L4_INLINE l4_buf_regs_t *l4_utcb_br(void) L4_NOTHROW
00362 { return l4_utcb_br_u(l4_utcb()); }
00363
00364 L4_INLINE l4_thread_regs_t *l4_utcb_tcr(void) L4_NOTHROW
00365 { return l4_utcb_tcr_u(l4_utcb()); }
00366
00367 L4_INLINE l4_exc_regs_t *l4_utcb_exc(void) L4_NOTHROW
00368 { return l4_utcb_exc_u(l4_utcb()); }
00369
00370 L4_INLINE void l4_utcb_inherit_fpu(int switch_on) L4_NOTHROW
00371 { l4_utcb_inherit_fpu_u(l4_utcb(), switch_on); }
00372
00373 L4_INLINE
00374 l4_timeout_s l4_timeout_abs_u(l4_kernel_clock_t val, int pos,
00375                               l4_utcb_t *utcb) L4_NOTHROW
00376 {
00377     union T
00378     {
00379         l4_kernel_clock_t t;
00380         l4_umword_t m[sizeof(l4_kernel_clock_t)/sizeof(l4_umword_t)];
00381     };
00382     l4_timeout_s to;
00383     to.t = 0x8000 | pos;
00384     ((union T*)(l4_utcb_br_u(utcb)->br + pos))->t = val;
00385     return to;
00386 }
00387
00388 L4_INLINE
00389 l4_timeout_s l4_timeout_abs(l4_kernel_clock_t val, int pos) L4_NOTHROW
00390 { return l4_timeout_abs_u(val, pos, l4_utcb()); }
00391
00392 L4_INLINE unsigned l4_utcb_mr64_idx(unsigned idx) L4_NOTHROW
00393 { return idx / (sizeof(l4_uint64_t) / sizeof(l4_umword_t)); }
00394
00395 L4_END_DECLS
00396
00397 #endif /* ! _L4_SYS_UTCB_H */

```

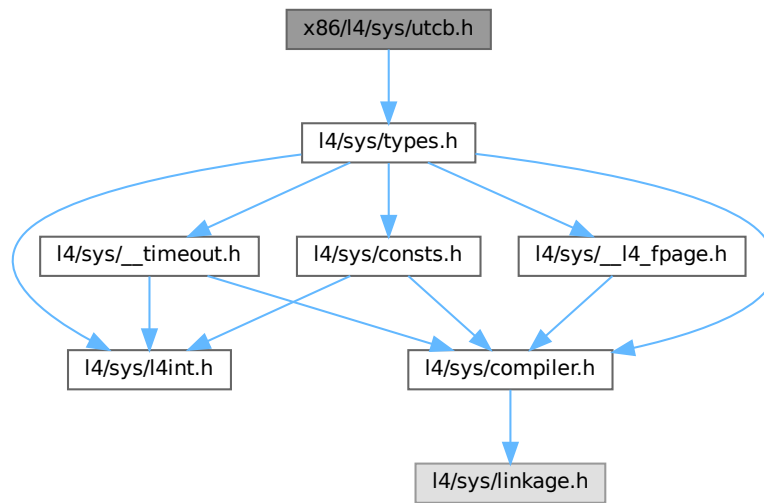
## 17.611 x86/i4/sys/utcb.h File Reference

UTCB definitions for x86.



```
#include <l4/sys/types.h>
```

Include dependency graph for utcb.h:



## Data Structures

- struct `l4_exc_regs_t`  
*UTCB structure for exceptions.*

## Typedefs

- typedef struct `l4_exc_regs_t` `l4_exc_regs_t`  
*UTCB structure for exceptions.*

## Enumerations

- enum `L4_utcb_consts_x86` {  
`L4_UTCB_EXCEPTION_REGS_SIZE = 19` , `L4_UTCB_GENERIC_DATA_SIZE = 63` , `L4_UTCB_GENERIC_BUFFERS_SIZE = 58` , `L4_UTCB_MSG_REGS_OFFSET = 0` ,  
`L4_UTCB_BUF_REGS_OFFSET = 64 * sizeof(l4_umword_t)` , `L4_UTCB_THREAD_REGS_OFFSET = 123 * sizeof(l4_umword_t)` , `L4_UTCB_INHERIT_FPU = 1UL << 24` , `L4_UTCB_OFFSET = 512` }  
*UTCB constants for x86.*

## Functions

- `l4_umword_t l4_utcb_exc_pc (l4_exc_regs_t const *u)` `L4_NOTHROW`  
*Access function to get the program counter of the exception state.*
- `void l4_utcb_exc_pc_set (l4_exc_regs_t *u, l4_addr_t pc)` `L4_NOTHROW`  
*Set the program counter register in the exception state.*
- `l4_umword_t l4_utcb_exc_typeval (l4_exc_regs_t const *u)` `L4_NOTHROW`

*Get the value out of an exception UTCB that describes the type of exception.*

- `int l4_utcb_exc_is_pf (l4_exc_regs_t const *u) L4_NOTHROW`

*Check whether an exception IPC is a page fault.*

- `l4_addr_t l4_utcb_exc_pfa (l4_exc_regs_t const *u) L4_NOTHROW`

*Function to get the L4 style page fault address out of an exception.*

- `int l4_utcb_exc_is_ex_regs_exception (l4_exc_regs_t const *u) L4_NOTHROW`

*Check whether an exception IPC was triggered via `l4_thread_ex_regs()`.*

## 17.611.1 Detailed Description

UTCB definitions for x86.

Definition in file `utcb.h`.

## 17.612 utcb.h

[Go to the documentation of this file.](#)

```

00001 /*****
00002  */
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 /*****
00010  */
00011 #ifndef __L4_SYS__INCLUDE__ARCH_X86__UTCB_H__
00012 #define __L4_SYS__INCLUDE__ARCH_X86__UTCB_H__
00013
00014 #include <l4/sys/types.h>
00015
00016 enum l4_utcb_consts_x86
00017 {
00018     L4_UTCB_EXCEPTION_REGS_SIZE    = 19,
00019     L4_UTCB_GENERIC_DATA_SIZE      = 63,
00020     L4_UTCB_GENERIC_BUFFERS_SIZE   = 58,
00021     L4_UTCB_MSG_REGS_OFFSET        = 0,
00022     L4_UTCB_BUF_REGS_OFFSET        = 64 * sizeof(l4_umword_t),
00023     L4_UTCB_THREAD_REGS_OFFSET     = 123 * sizeof(l4_umword_t),
00024     L4_UTCB_INHERIT_FPU            = 1UL < 24,
00025     L4_UTCB_OFFSET                 = 512,
00026 };
00027
00028 typedef struct l4_exc_regs_t
00029 {
00030     l4_umword_t es;
00031     l4_umword_t ds;
00032     l4_umword_t gs;
00033     l4_umword_t fs;
00034
00035     l4_umword_t edi;
00036     l4_umword_t esi;
00037     l4_umword_t ebp;
00038     l4_umword_t pfa;
00039     l4_umword_t ebx;
00040     l4_umword_t edx;
00041     l4_umword_t ecx;
00042     l4_umword_t eax;
00043
00044     l4_umword_t trapno;
00045     l4_umword_t err;
00046 }

```

```

00080  l4_umword_t ip;
00081  l4_umword_t dummy1;
00082  l4_umword_t flags;
00083  l4_umword_t sp;
00084  l4_umword_t ss;
00085 } l4_exc_regs_t;
00086
00087 #include_next <l4/sys/utcb.h>
00088
00089 /*
00090  * =====
00091  * Implementations.
00092  */
00093
00094 L4_INLINE l4_utcb_t *l4_utcb_direct(void) L4_NOTHROW
00095 {
00096     l4_utcb_t *utcb;
00097     __asm__ ("mov %%fs:0, %0" : "=r" (utcb));
00098     return utcb;
00099 }
00100
00101 L4_INLINE l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u) L4_NOTHROW
00102 {
00103     return u->ip;
00104 }
00105
00106 L4_INLINE void l4_utcb_exc_pc_set(l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW
00107 {
00108     u->ip = pc;
00109 }
00110
00111 L4_INLINE void l4_utcb_exc_sp_set(l4_exc_regs_t *u, l4_addr_t sp) L4_NOTHROW
00112 {
00113     u->sp = sp;
00114 }
00115
00116 L4_INLINE l4_umword_t l4_utcb_exc_typeval(l4_exc_regs_t const *u) L4_NOTHROW
00117 {
00118     return u->trapno;
00119 }
00120
00121 L4_INLINE int l4_utcb_exc_is_pf(l4_exc_regs_t const *u) L4_NOTHROW
00122 {
00123     return u->trapno == 14;
00124 }
00125
00126 L4_INLINE l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u) L4_NOTHROW
00127 {
00128     return (u->pfa & ~7UL) | (u->err & 2);
00129 }
00130
00131 L4_INLINE int l4_utcb_exc_is_ex_regs_exception(l4_exc_regs_t const *u) L4_NOTHROW
00132 {
00133     return l4_utcb_exc_typeval(u) == 0xff;
00134 }
00135
00136 #endif /* ! __L4_SYS__INCLUDE__ARCH_X86__UTCB_H__ */

```

## 17.613 l4/sys/vcon File Reference

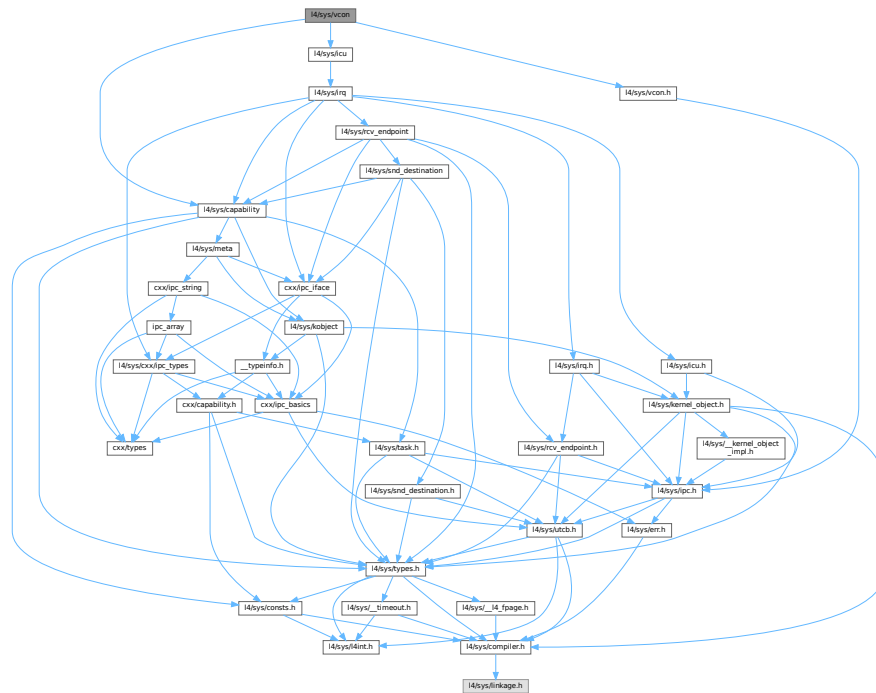
C++ Virtual console interface.

```

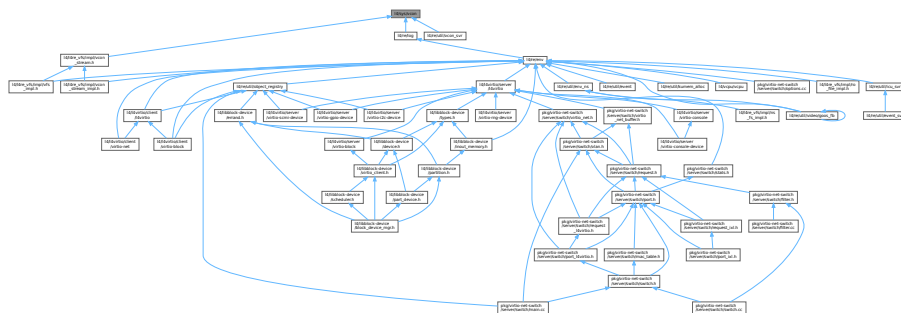
#include <l4/sys/icu>
#include <l4/sys/vcon.h>
#include <l4/sys/capability>

```

Include dependency graph for vcon:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4::Vcon](#)  
*C++ L4 Vcon interface, see [Virtual Console](#) for the C interface.*

## Namespaces

- namespace **L4**  
*L4 low-level kernel interface.*

## 17.613.1 Detailed Description

C++ Virtual console interface.

Definition in file [vcon](#).

## 17.614 vcon

[Go to the documentation of this file.](#)

```

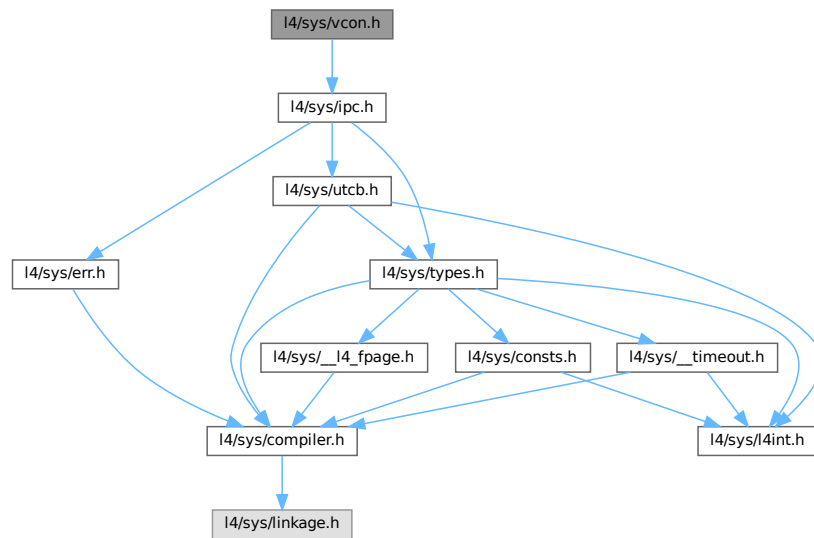
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00005  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00006  *      economic rights: Technische Universität Dresden (Germany)
00007  *
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010 #pragma once
00011
00012 #include <l4/sys/icu>
00013 #include <l4/sys/vcon.h>
00014 #include <l4/sys/capability>
00015
00016 namespace L4 {
00017
00018 class Vcon :
00019     public Kobject_t<Vcon, Icu, L4_PROTO_LOG>
00020 {
00021 public:
00022     l4_msgtag_t
00023     send(char const *buf, unsigned size, l4_utcb_t *utcb = l4_utcb()) const noexcept
00024     { return l4_vcon_send_u(cap(), buf, size, utcb); }
00025
00026     long
00027     write(char const *buf, unsigned size, l4_utcb_t *utcb = l4_utcb()) const noexcept
00028     { return l4_vcon_write_u(cap(), buf, size, utcb); }
00029
00030     int
00031     read(char *buf, unsigned size, l4_utcb_t *utcb = l4_utcb()) const noexcept
00032     { return l4_vcon_read_u(cap(), buf, size, utcb); }
00033
00034     int
00035     read_with_flags(char *buf, unsigned size, l4_utcb_t *utcb = l4_utcb()) const noexcept
00036     { return l4_vcon_read_with_flags_u(cap(), buf, size, utcb); }
00037
00038     l4_msgtag_t
00039     set_attr(l4_vcon_attr_t const *attr, l4_utcb_t *utcb = l4_utcb()) const noexcept
00040     { return l4_vcon_set_attr_u(cap(), attr, utcb); }
00041
00042     l4_msgtag_t
00043     get_attr(l4_vcon_attr_t *attr, l4_utcb_t *utcb = l4_utcb()) const noexcept
00044     { return l4_vcon_get_attr_u(cap(), attr, utcb); }
00045
00046     typedef L4::Typeid::Raw_ipc<Vcon> Rpcs;
00047 };
00048
00049 }
```

## 17.615 l4/sys/vcon.h File Reference

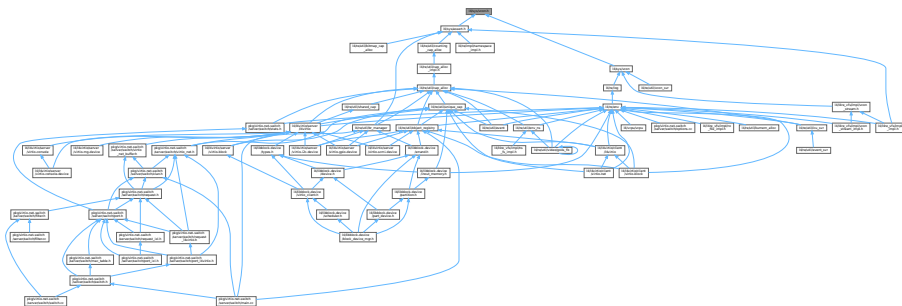
Virtual console interface.

```
#include <l4/sys/ipc.h>
```

Include dependency graph for vcon.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `l4_vcon_attr_t`  
*Vcon attribute structure.*

## Typedefs

- typedef struct `l4_vcon_attr_t` `l4_vcon_attr_t`  
*Vcon attribute structure.*

## Enumerations

- enum `L4_vcon_size_consts` { `L4_VCON_WRITE_SIZE` = (L4\_UTCB\_GENERIC\_DATA\_SIZE - 2) \* sizeof(l4\_umword\_t) , `L4_VCON_READ_SIZE` = (L4\_UTCB\_GENERIC\_DATA\_SIZE - 1) \* sizeof(l4\_umword\_t) }  
Size constants.
- enum `L4_vcon_read_flags` { `L4_VCON_READ_SIZE_MASK` = 0x3ffffff , `L4_VCON_READ_STAT_BREAK` = 1 << 30 , `L4_VCON_READ_STAT_DONE` = 1 << 31 }  
Vcon read flags.
- enum `L4_vcon_i_flags` { `L4_VCON_INLCR` = 000100 , `L4_VCON_IGNCR` = 000200 , `L4_VCON_ICRNL` = 000400 }  
Input flags.
- enum `L4_vcon_o_flags` { `L4_VCON_ONLCR` = 000004 , `L4_VCON_OCRNL` = 000010 , `L4_VCON_ONLRET` = 000040 }  
Output flags.
- enum `L4_vcon_l_flags` { `L4_VCON_ICANON` = 000002 , `L4_VCON_ECHO` = 000010 }  
Local flags.
- enum `L4_vcon_ops` { `L4_VCON_WRITE_OP` = 0UL , `L4_VCON_READ_OP` = 1UL , `L4_VCON_SET_ATTR_OP` = 2UL , `L4_VCON_GET_ATTR_OP` = 3UL }  
Operations on vcon objects.

## Functions

- `l4_msgtag_t l4_vcon_send` (`l4_cap_idx_t` vcon, char const \*buf, unsigned size) `L4_NOTHROW`  
Send data to virtual console.
- `l4_msgtag_t l4_vcon_send_u` (`l4_cap_idx_t` vcon, char const \*buf, unsigned size, `l4_utcb_t` \*utcb) `L4_NOTHROW`  
Send data to *this* virtual console.
- `long l4_vcon_write` (`l4_cap_idx_t` vcon, char const \*buf, unsigned size) `L4_NOTHROW`  
Write data to virtual console.
- `long l4_vcon_write_u` (`l4_cap_idx_t` vcon, char const \*buf, unsigned size, `l4_utcb_t` \*utcb) `L4_NOTHROW`  
Write data to *this* virtual console.
- `int l4_vcon_read` (`l4_cap_idx_t` vcon, char \*buf, unsigned size) `L4_NOTHROW`  
Read data from virtual console.
- `int l4_vcon_read_u` (`l4_cap_idx_t` vcon, char \*buf, unsigned size, `l4_utcb_t` \*utcb) `L4_NOTHROW`  
Read data from *this* virtual console.
- `int l4_vcon_read_with_flags` (`l4_cap_idx_t` vcon, char \*buf, unsigned size) `L4_NOTHROW`  
Read data from virtual console, extended version including flags.
- `l4_msgtag_t l4_vcon_set_attr` (`l4_cap_idx_t` vcon, `l4_vcon_attr_t` const \*attr) `L4_NOTHROW`  
Set attributes of a Vcon.
- `l4_msgtag_t l4_vcon_set_attr_u` (`l4_cap_idx_t` vcon, `l4_vcon_attr_t` const \*attr, `l4_utcb_t` \*utcb) `L4_NOTHROW`  
Set the attributes of *this* virtual console.
- `l4_msgtag_t l4_vcon_get_attr` (`l4_cap_idx_t` vcon, `l4_vcon_attr_t` \*attr) `L4_NOTHROW`  
Get attributes of a Vcon.
- `l4_msgtag_t l4_vcon_get_attr_u` (`l4_cap_idx_t` vcon, `l4_vcon_attr_t` \*attr, `l4_utcb_t` \*utcb) `L4_NOTHROW`  
Get attributes of *this* virtual console.
- `void l4_vcon_set_attr_raw` (`l4_vcon_attr_t` \*attr) `L4_NOTHROW`  
Set terminal attributes to disable all special processing.

## 17.615.1 Detailed Description

Virtual console interface.

Definition in file [vcon.h](#).

## 17.615.2 Enumeration Type Documentation

### 17.615.2.1 L4\_vcon\_read\_flags

```
enum L4_vcon_read_flags
```

Vcon read flags.

#### Enumerator

L4_VCON_READ_SIZE_MASK	Size mask.
L4_VCON_READ_STAT_BREAK	Break condition flag.
L4_VCON_READ_STAT_DONE	Done condition flag.

Definition at line 170 of file [vcon.h](#).

## 17.616 vcon.h

[Go to the documentation of this file.](#)

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00015 #include <l4/sys/ipc.h>
00016
00039
00056 L4_INLINE l4_msgtag_t
00057 l4_vcon_send(l4_cap_idx_t vcon, char const *buf, unsigned size) L4_NOTHROW;
00058
00065 L4_INLINE l4_msgtag_t
00066 l4_vcon_send_u(l4_cap_idx_t vcon, char const *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW;
00067
00079 L4_INLINE long
00080 l4_vcon_write(l4_cap_idx_t vcon, char const *buf, unsigned size) L4_NOTHROW;
00081
00088 L4_INLINE long
00089 l4_vcon_write_u(l4_cap_idx_t vcon, char const *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW;
00090
00095 enum L4_vcon_size_consts
00096 {
00098     L4_VCON_WRITE_SIZE = (L4_UTCB_GENERIC_DATA_SIZE - 2) * sizeof(l4_umword_t),
00100     L4_VCON_READ_SIZE  = (L4_UTCB_GENERIC_DATA_SIZE - 1) * sizeof(l4_umword_t),
00101 };
00102
00119 L4_INLINE int
00120 l4_vcon_read(l4_cap_idx_t vcon, char *buf, unsigned size) L4_NOTHROW;
00121
00128 L4_INLINE int
00129 l4_vcon_read_u(l4_cap_idx_t vcon, char *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW;
```



```

00130
00157 L4_INLINE int
00158 l4_vcon_read_with_flags(l4_cap_idx_t vcon, char *buf, unsigned size) L4_NOTHROW;
00159
00163 L4_INLINE int
00164 l4_vcon_read_with_flags_u(l4_cap_idx_t vcon, char *buf, unsigned size,
00165                           l4_utcb_t *utcb) L4_NOTHROW;
00166
00170 enum L4_vcon_read_flags
00171 {
00172     L4_VCON_READ_SIZE_MASK = 0x3fffffff,
00173     L4_VCON_READ_STAT_BREAK = 1 << 30,
00174     L4_VCON_READ_STAT_DONE = 1 << 31,
00175 };
00176
00187 typedef struct l4_vcon_attr_t
00188 {
00189     l4_umword_t i_flags;
00190     l4_umword_t o_flags;
00191     l4_umword_t l_flags;
00192
00193 #ifdef __cplusplus
00200     inline void set_raw();
00201 #endif
00202 } l4_vcon_attr_t;
00203
00208 enum L4_vcon_i_flags
00209 {
00210     L4_VCON_INLCR = 000100,
00211     L4_VCON_IGNCR = 000200,
00212     L4_VCON_ICRNL = 000400,
00213 };
00214
00219 enum L4_vcon_o_flags
00220 {
00221     L4_VCON_ONLCR = 000004,
00222     L4_VCON_OCRNL = 000010,
00223     L4_VCON_ONLRET = 000040,
00224 };
00225
00230 enum L4_vcon_l_flags
00231 {
00232     L4_VCON_ICANON = 000002,
00233     L4_VCON_ECHO = 000010,
00234 };
00235
00244 L4_INLINE l4_msgtag_t
00245 l4_vcon_set_attr(l4_cap_idx_t vcon, l4_vcon_attr_t const *attr) L4_NOTHROW;
00246
00253 L4_INLINE l4_msgtag_t
00254 l4_vcon_set_attr_u(l4_cap_idx_t vcon, l4_vcon_attr_t const *attr,
00255                   l4_utcb_t *utcb) L4_NOTHROW;
00256
00265 L4_INLINE l4_msgtag_t
00266 l4_vcon_get_attr(l4_cap_idx_t vcon, l4_vcon_attr_t *attr) L4_NOTHROW;
00267
00274 L4_INLINE l4_msgtag_t
00275 l4_vcon_get_attr_u(l4_cap_idx_t vcon, l4_vcon_attr_t *attr,
00276                   l4_utcb_t *utcb) L4_NOTHROW;
00277
00283 L4_INLINE void
00284 l4_vcon_set_attr_raw(l4_vcon_attr_t *attr) L4_NOTHROW;
00285
00286
00291 enum L4_vcon_ops
00292 {
00293     L4_VCON_WRITE_OP = 0UL,
00294     L4_VCON_READ_OP = 1UL,
00295     L4_VCON_SET_ATTR_OP = 2UL,
00296     L4_VCON_GET_ATTR_OP = 3UL,
00297 };
00298
00299 /***** Implementations *****/
00300
00301 L4_INLINE l4_msgtag_t
00302 l4_vcon_send_u(l4_cap_idx_t vcon, char const *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW
00303 {
00304     l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);
00305     mr->mr[0] = L4_VCON_WRITE_OP;
00306     mr->mr[1] = size;
00307     __builtin_memcpy(&mr->mr[2], buf, size);
00308     return l4_ipc_send(vcon, utcb,
00309                       l4_msgtag(L4_PROTO_LOG, 2 + l4_bytes_to_mwords(size),
00310                                0, L4_MSGTAG_SCHEDULE),
00311                       L4_IPC_NEVER);
00312 }
00313

```

```

00314 L4_INLINE l4_msgtag_t
00315 l4_vcon_send(l4_cap_idx_t vcon, char const *buf, unsigned size) L4_NOTHROW
00316 {
00317     return l4_vcon_send_u(vcon, buf, size, l4_utcb());
00318 }
00319
00320 L4_INLINE long
00321 l4_vcon_write_u(l4_cap_idx_t vcon, char const *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW
00322 {
00323     l4_msgtag_t t;
00324
00325     if (size > L4_VCON_WRITE_SIZE)
00326         size = L4_VCON_WRITE_SIZE;
00327
00328     t = l4_vcon_send_u(vcon, buf, size, utcb);
00329     if (l4_msgtag_has_error(t))
00330         return l4_error(t);
00331
00332     return (long) size;
00333 }
00334
00335 L4_INLINE long
00336 l4_vcon_write(l4_cap_idx_t vcon, char const *buf, unsigned size) L4_NOTHROW
00337 {
00338     return l4_vcon_write_u(vcon, buf, size, l4_utcb());
00339 }
00340
00341 L4_INLINE int
00342 l4_vcon_read_with_flags_u(l4_cap_idx_t vcon, char *buf, unsigned size,
00343                          l4_utcb_t *utcb) L4_NOTHROW
00344 {
00345     int ret;
00346     unsigned r;
00347     l4_msg_regs_t *mr;
00348
00349     mr = l4_utcb_mr_u(utcb);
00350     mr->mr[0] = (size << 16) | L4_VCON_READ_OP;
00351
00352     ret = l4_error_u(l4_ipc_call(vcon, utcb,
00353                                l4_msgtag(L4_PROTO_LOG, 1, 0, 0),
00354                                L4_IPC_NEVER),
00355                    utcb);
00356     if (ret < 0)
00357         return ret;
00358
00359     r = mr->mr[0] & L4_VCON_READ_SIZE_MASK;
00360
00361     if (!(mr->mr[0] & L4_VCON_READ_STAT_DONE)) // !eof
00362         ret = size + 1;
00363     else if (r < size)
00364         ret = r;
00365     else
00366         ret = size;
00367
00368     if (L4_LIKELY(buf != NULL))
00369         __builtin_memcpy(buf, &mr->mr[1], r < size ? r : size);
00370
00371     return ret | (mr->mr[0] & ~(L4_VCON_READ_STAT_DONE | L4_VCON_READ_SIZE_MASK));
00372 }
00373
00374 L4_INLINE int
00375 l4_vcon_read_with_flags(l4_cap_idx_t vcon, char *buf, unsigned size) L4_NOTHROW
00376 {
00377     return l4_vcon_read_with_flags_u(vcon, buf, size, l4_utcb());
00378 }
00379
00380 L4_INLINE int
00381 l4_vcon_read_u(l4_cap_idx_t vcon, char *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW
00382 {
00383     int r = l4_vcon_read_with_flags_u(vcon, buf, size, utcb);
00384     if (r < 0)
00385         return r;
00386
00387     return r & L4_VCON_READ_SIZE_MASK;
00388 }
00389
00390 L4_INLINE int
00391 l4_vcon_read(l4_cap_idx_t vcon, char *buf, unsigned size) L4_NOTHROW
00392 {
00393     return l4_vcon_read_u(vcon, buf, size, l4_utcb());
00394 }
00395
00396 L4_INLINE l4_msgtag_t
00397 l4_vcon_set_attr_u(l4_cap_idx_t vcon, l4_vcon_attr_t const *attr,
00398                   l4_utcb_t *utcb) L4_NOTHROW
00399 {
00400     l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);

```

```

00401
00402     mr->mr[0] = L4_VCON_SET_ATTR_OP;
00403     __builtin_memcpy(&mr->mr[1], attr, sizeof(*attr));
00404
00405     return l4_ipc_call(vcon, utcb,
00406                       l4_msgtag(L4_PROTO_LOG, 4, 0, 0),
00407                       L4_IPC_NEVER);
00408 }
00409
00410 L4_INLINE l4_msgtag_t
00411 l4_vcon_set_attr(l4_cap_idx_t vcon, l4_vcon_attr_t const *attr) L4_NOTHROW
00412 {
00413     return l4_vcon_set_attr_u(vcon, attr, l4_utcb());
00414 }
00415
00416 L4_INLINE l4_msgtag_t
00417 l4_vcon_get_attr_u(l4_cap_idx_t vcon, l4_vcon_attr_t *attr,
00418                  l4_utcb_t *utcb) L4_NOTHROW
00419 {
00420     l4_msgtag_t res;
00421     l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);
00422
00423     mr->mr[0] = L4_VCON_GET_ATTR_OP;
00424
00425     res = l4_ipc_call(vcon, utcb,
00426                      l4_msgtag(L4_PROTO_LOG, 1, 0, 0),
00427                      L4_IPC_NEVER);
00428     if (l4_error_u(res, utcb) >= 0)
00429         __builtin_memcpy(attr, &mr->mr[1], sizeof(*attr));
00430
00431     return res;
00432 }
00433
00434 L4_INLINE l4_msgtag_t
00435 l4_vcon_get_attr(l4_cap_idx_t vcon, l4_vcon_attr_t *attr) L4_NOTHROW
00436 {
00437     return l4_vcon_get_attr_u(vcon, attr, l4_utcb());
00438 }
00439
00440 L4_INLINE void
00441 l4_vcon_set_attr_raw(l4_vcon_attr_t *attr) L4_NOTHROW
00442 {
00443     attr->i_flags = 0;
00444     attr->o_flags = 0;
00445     attr->l_flags = 0;
00446 }
00447
00448 #ifdef __cplusplus
00449 inline void
00450 l4_vcon_attr_t::set_raw()
00451 { l4_vcon_set_attr_raw(this); }
00452 #endif

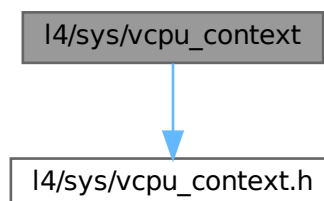
```

## 17.617 l4/sys/vcpu\_context File Reference

Hardware vCPU context interface.

```
#include <l4/sys/vcpu_context.h>
```

Include dependency graph for vcpu\_context:



## Namespaces

- namespace [L4](#)  
*[L4](#) low-level kernel interface.*

### 17.617.1 Detailed Description

Hardware vCPU context interface.

Definition in file [vcpu\\_context](#).

## 17.618 vcpu\_context

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006
00007 #pragma once
00008
00009 #include <l4/sys/vcpu_context.h>
00010
00011 namespace L4 {
00012
00013 class Vcpu_context :
00014     public Kobject_t<Vcpu_context, Kobject, L4_PROTO_VCPU_CONTEXT>
00015 {
00016 public:
00017     Vcpu_context(Vcpu_context const &) = delete;
00018     void operator = (Vcpu_context const &) = delete;
00019
00020 protected:
00021     Vcpu_context();
00022 };
00023
00024 };
```

## 17.619 vcpu\_context.h

```
00001
00006
00007 #pragma once
```

## 17.620 vm

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2018 Adam Lackorzynski <adam@l4re.org>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/sys/__vm-arm.h>
```

## 17.621 vm

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2018 Adam Lackorzynski <adam@l4re.org>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/sys/__vm-arm.h>
```

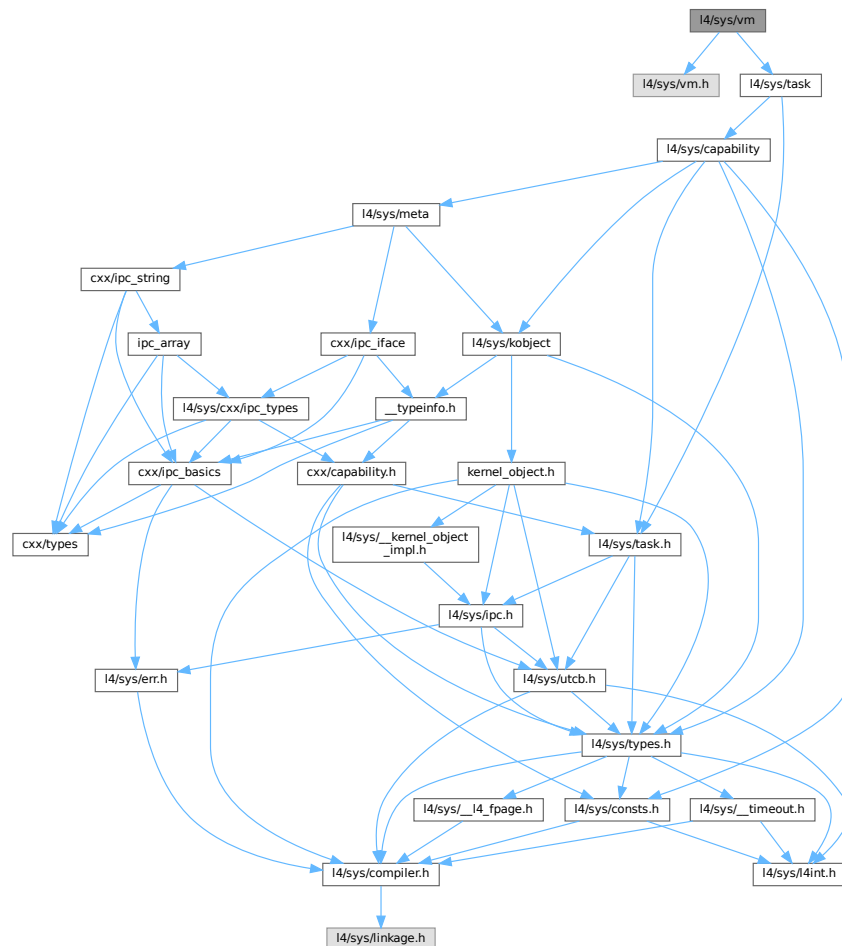
## 17.622 I4/sys/vm File Reference

Virtualization interface.

```
#include <l4/sys/vm.h>
```

```
#include <l4/sys/task>
```

Include dependency graph for vm:



### Data Structures

- class [L4::Vm](#)  
*Virtual machine host address space.*

### Namespaces

- namespace [L4](#)  
*L4 low-level kernel interface.*

### 17.622.1 Detailed Description

Virtualization interface.

Definition in file [vm](#).

## 17.623 vm

[Go to the documentation of this file.](#)

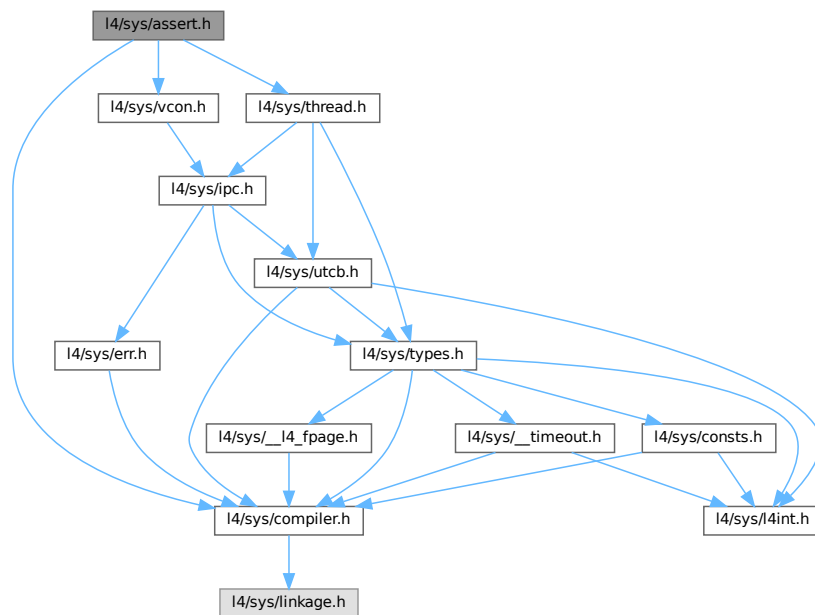
```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013
00014 #pragma once
00015
00016 #include <l4/sys/vm.h>
00017 #include <l4/sys/task>
00018
00019 namespace L4 {
00020
00021     class Vm : public Kobject_t<Vm, Task, L4_PROTO_VM>
00022     {
00023     protected:
00024         Vm();
00025
00026     private:
00027         Vm(Vm const &);
00028         void operator = (Vm const &);
00029     };
00030
00031 };
```

## 17.624 l4/sys/assert.h File Reference

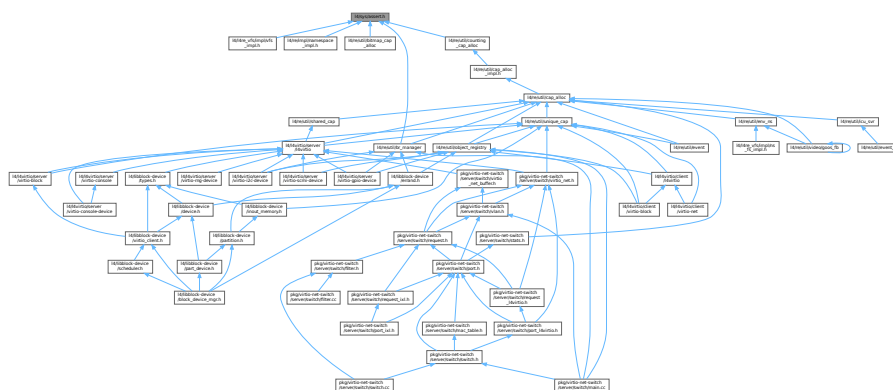
Low-level assert implementation.

```
#include <l4/sys/compiler.h>
#include <l4/sys/thread.h>
#include <l4/sys/vcon.h>
```

Include dependency graph for assert.h:



This graph shows which files directly or indirectly include this file:



## Macros

- `#define l4_assert(expr)`  
*Low-level assert.*

## 17.624.1 Detailed Description

Low-level assert implementation.

Definition in file [assert.h](#).

## 17.624.2 Macro Definition Documentation

### 17.624.2.1 l4\_assert

```
#define l4_assert(  
    expr)
```

#### Value:

```
l4_assert_fn(!(expr), __FILE__ ":" L4_stringify(__LINE__) ": Assertion \"" \
    L4_stringify(expr) "\" failed.\n")
```

Low-level assert.

#### Parameters

<i>expr</i>	Expression to be evaluate for the assertion.
-------------	----------------------------------------------

This assertion is a low-level implementation that directly uses kernel primitives. Only use [l4\\_assert\(\)](#) when the standard `assert()` functionality is not available.

Definition at line 32 of file [assert.h](#).

Referenced by [L4Re::Util::Cap\\_alloc\\_base::free\(\)](#), [L4Re::Util::Counting\\_cap\\_alloc< COUNTERTYPE, Dbg >::free\(\)](#), [L4Re::Util::Counting\\_cap\\_alloc< COUNTERTYPE, Dbg >::release\(\)](#), [L4Re::Util::Br\\_manager::set\\_rcv\\_cap\\_flags\(\)](#), and [Block\\_device::Device\\_mgr< DEV, FACTORY, SCHEDULER >::shutdown\\_event\(\)](#).

## 17.625 assert.h

[Go to the documentation of this file.](#)

```
00001
00005 /*
00006  * (c) 2015 Adam Lackorzynski <adam@l4re.org>
00007  *
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010 #pragma once
00011
00012 #ifndef NDEBUG
00013
00014 #define l4_assert(x) do { } while (0)
00015 #define l4_check(x) do { (void)(x); } while (0)
00016
00017 #else
00018
00019 #include <l4/sys/compiler.h>
00020 #include <l4/sys/thread.h>
00021 #include <l4/sys/vcon.h>
00022
00032 #define l4_assert(expr) \
00033     l4_assert_fn(!(expr), __FILE__ ":" L4_stringify(__LINE__) ": Assertion \"" \
00034         L4_stringify(expr) "\" failed.\n")
00035
00036 #define l4_check(expr) l4_assert(expr)
00037
00041 L4_ALWAYS_INLINE
00042 void l4_assert_fn(unsigned expr, const char *text) L4_NOTHROW;
00043
00047 L4_INLINE L4_NORETURN
00048 void l4_assert_abort(const char *text) L4_NOTHROW;
00049
00050
00051 /* IMPLEMENTATION -----*/
00052
00053 L4_INLINE L4_NORETURN
00054 void l4_assert_abort(const char *text) L4_NOTHROW
```



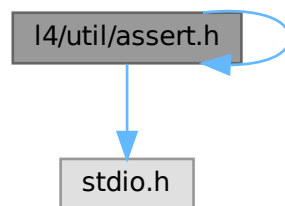
```
00055 {
00056     l4_vcon_write(L4_BASE_LOG_CAP, text, __builtin_strlen(text));
00057     for (;;)
00058         l4_thread_ex_regs(L4_INVALID_CAP, ~0UL, ~0UL,
00059                           L4_THREAD_EX_REGS_TRIGGER_EXCEPTION);
00060 }
00061
00062 L4_ALWAYS_INLINE
00063 void l4_assert_fn(unsigned expr, const char *text) L4_NOTHROW
00064 {
00065     if (L4_LIKELY(expr))
00066         return;
00067     l4_assert_abort(text);
00068 }
00069
00070
00071 #endif /* NDEBUG */
```

## 17.626 l4/util/assert.h File Reference

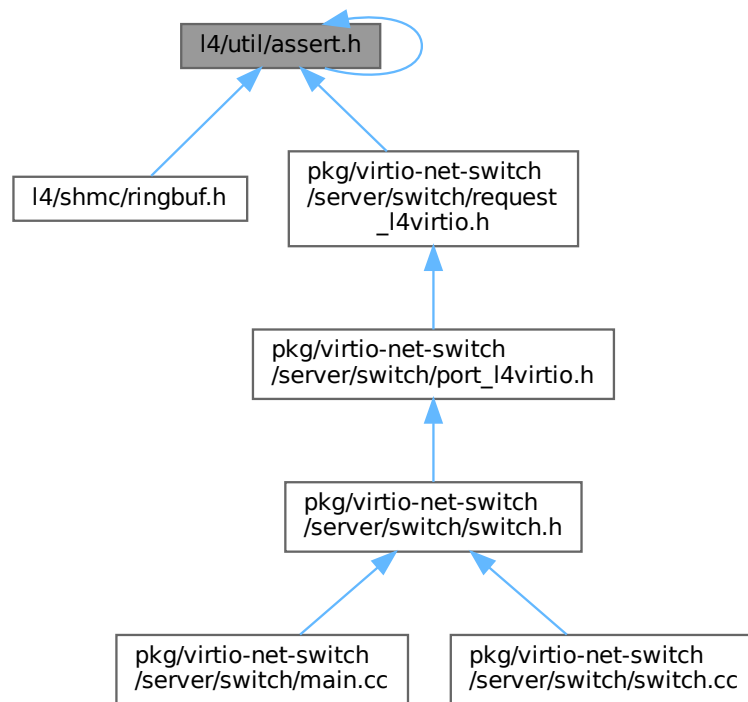
Some useful assert-style macros.

```
#include <stdio.h>
#include <assert.h>
```

Include dependency graph for assert.h:



This graph shows which files directly or indirectly include this file:



### 17.626.1 Detailed Description

Some useful assert-style macros.

#### Date

09/2009

#### Author

Bjoern Doebel [doebel@tudos.org](mailto:doebel@tudos.org)

Definition in file [assert.h](#).

## 17.627 assert.h

[Go to the documentation of this file.](#)

```

00001  /*****
00009  */
00010  * (c) 2009 Author(s)
00011  *   economic rights: Technische Universität Dresden (Germany)
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */

```

```

00014
00015 /*****
00016 #pragma once
00017
00018 #ifndef NDEBUG
00019
00020 #define DO_NOTHING          do {} while (0)
00021 #define ASSERT_ASSERT(x)    DO_NOTHING
00022 #define ASSERT_VALID(c)     DO_NOTHING
00023 #define ASSERT_EQUAL(a,b)   DO_NOTHING
00024 #define ASSERT_NOT_EQUAL(a,b) DO_NOTHING
00025 #define ASSERT_LOWER_EQ(a,b) DO_NOTHING
00026 #define ASSERT_GREATER_EQ(a,b) DO_NOTHING
00027 #define ASSERT_BETWEEN(a,b,c) DO_NOTHING
00028 #define ASSERT_IPC_OK(i)    DO_NOTHING
00029 #define ASSERT_OK(e)         do { (void)e; } while (0)
00030 #define ASSERT_NOT_NULL(p)   DO_NOTHING
00031 #ifndef assert
00032 #define assert(cond)          DO_NOTHING
00033 #endif
00034
00035 #else // NDEBUG
00036
00037 #ifndef ASSERT_PRINTF
00038 #include <stdio.h>
00039 #define ASSERT_PRINTF printf
00040 #endif
00041 #ifndef ASSERT_ASSERT
00042 #include <assert.h>
00043 #define ASSERT_ASSERT(x) assert(x)
00044 #endif
00045
00046 #define ASSERT_VALID(cap) \
00047     do { \
00048         typeof(cap) _cap = cap; \
00049         if (!l4_is_invalid_cap(_cap)) { \
00050             ASSERT_PRINTF("%s: Cap invalid.\n", __func__); \
00051             ASSERT_ASSERT(!l4_is_invalid_cap(_cap)); \
00052         } \
00053     } while (0)
00054
00055 #define ASSERT_EQUAL(a, b) \
00056     do { \
00057         typeof(a) _a = a; \
00058         typeof(b) _b = b; \
00059         if (_a != _b) { \
00060             ASSERT_PRINTF("%s:\n", __func__); \
00061             ASSERT_PRINTF("    "#a" (%lx) != "#b" (%lx)\n", (unsigned long)_a, (unsigned long)_b); \
00062             ASSERT_ASSERT(_a == _b); \
00063         } \
00064     } while (0)
00065
00066 #define ASSERT_NOT_EQUAL(a, b) \
00067     do { \
00068         typeof(a) _a = a; \
00069         typeof(b) _b = b; \
00070         if (_a == _b) { \
00071             ASSERT_PRINTF("%s:\n", __func__); \
00072             ASSERT_PRINTF("    "#a" (%lx) == "#b" (%lx)\n", (unsigned long)_a, (unsigned long)_b); \
00073             ASSERT_ASSERT(_a != _b); \
00074         } \
00075     } while (0)
00076
00077 #define ASSERT_LOWER_EQ(val, max) \
00078     do { \
00079         typeof(val) _val = val; \
00080         typeof(max) _max = max; \
00081         if (_val > _max) { \
00082             ASSERT_PRINTF("%s:\n", __func__); \
00083             ASSERT_PRINTF("    "#val" (%lx) > "#max" (%lx)\n", (unsigned long)_val, (unsigned long)_max); \
00084             ASSERT_ASSERT(_val <= _max); \
00085         } \
00086     } while (0)
00087
00088 #define ASSERT_GREATER_EQ(val, min) \
00089     do { \
00090         typeof(val) _val = val; \
00091         typeof(min) _min = min; \
00092         if (_val < _min) { \
00093             ASSERT_PRINTF("%s:\n", __func__); \
00094             ASSERT_PRINTF("    "#val" (%lx) < "#min" (%lx)\n", (unsigned long)_val, (unsigned long)_min); \
00095             ASSERT_ASSERT(_val >= _min); \
00096         } \
00097     } while (0)
00098
00099
00100

```

```

00101     } while (0)
00102
00103
00104 #define ASSERT_BETWEEN(val, min, max) \
00105     ASSERT_LOWER_EQ((val), (max)); \
00106     ASSERT_GREATER_EQ((val), (min));
00107
00108
00109 #define ASSERT_IPC_OK(msgtag) \
00110     do { \
00111         int _r = l4_ipc_error(msgtag, l4_utcb()); \
00112         if (_r) { \
00113             ASSERT_PRINTF("%s: IPC Error: %lx\n", __func__, _r); \
00114             ASSERT_ASSERT(_r == 0); \
00115         } \
00116     } while (0)
00117
00118 #define ASSERT_OK(val)          ASSERT_EQUAL((val), 0)
00119 #define ASSERT_NOT_NULL(ptr)    ASSERT_NOT_EQUAL((ptr), (void *)0)
00120
00121 #endif // NDEBUG

```

## 17.628 atomic.h

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00015 #include <l4/sys/compiler.h>
00016
00017 L4_BEGIN_DECLS
00018
00019 long int
00020 l4_atomic_add(volatile long int* mem, long int offset) L4_NOTHROW L4_LONG_CALL;
00021
00022 long int
00023 l4_atomic_xchg(volatile long int* mem, long int newval) L4_NOTHROW L4_LONG_CALL;
00024
00025 long int
00026 l4_atomic_cmpxchg(volatile long int* mem, long int oldval, long int newval) L4_NOTHROW L4_LONG_CALL;
00027
00028 L4_END_DECLS

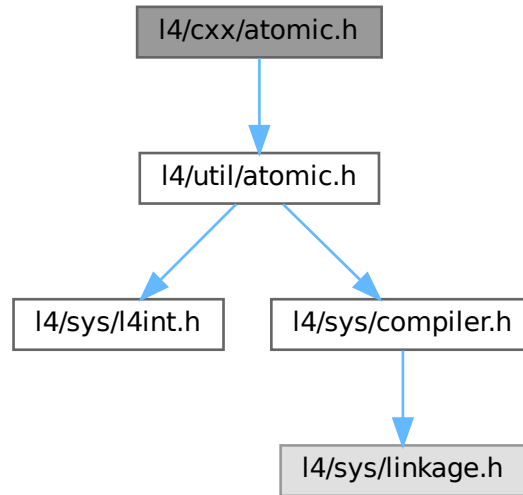
```

## 17.629 l4/cxx/atomic.h File Reference

Atomic template.

```
#include <l4/util/atomic.h>
```

Include dependency graph for atomic.h:



## Namespaces

- namespace `L4`  
*L4 low-level kernel interface.*

## 17.629.1 Detailed Description

Atomic template.

Definition in file [atomic.h](#).

## 17.630 atomic.h

[Go to the documentation of this file.](#)

```

00001
00002 /*
00003  * (c) 2004-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010 #include <l4/util/atomic.h>
00011
00012 extern "C" void ____error_compare_and_swap_does_not_support_3_bytes____();
00013 extern "C" void ____error_compare_and_swap_does_not_support_more_than_4_bytes____();
00014
00015 namespace L4
00016 {

```

```

00021  template< typename X >
00022  inline int compare_and_swap(X volatile *dst, X old_val, X new_val)
00023  {
00024      switch (sizeof(X))
00025      {
00026          case 1:
00027              return l4util_cmpxchg8((l4_uint8_t volatile*)dst, old_val, new_val);
00028          case 2:
00029              return l4util_cmpxchg16((l4_uint16_t volatile *)dst, old_val, new_val);
00030          case 3: ____error_compare_and_swap_does_not_support_3_bytes____();
00031          case 4:
00032              return l4util_cmpxchg32((l4_uint32_t volatile*)dst, old_val, new_val);
00033          default:
00034              ____error_compare_and_swap_does_not_support_more_than_4_bytes____();
00035      }
00036      return 0;
00037  }
00038 }

```

## 17.631 l4/util/atomic.h File Reference

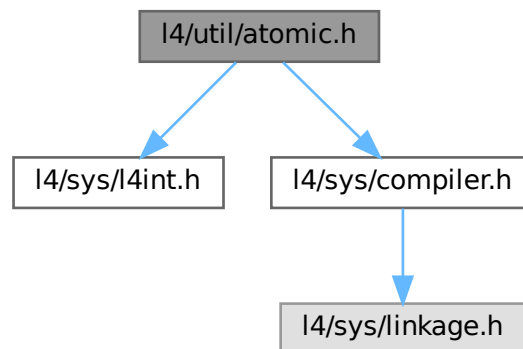
atomic operations header and generic implementations

```

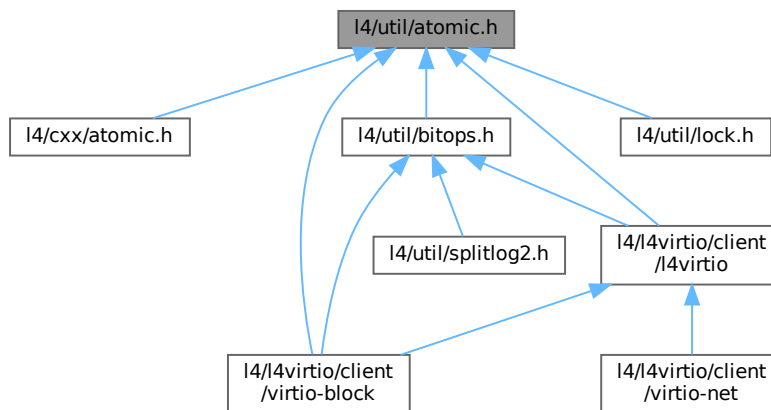
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for atomic.h:



This graph shows which files directly or indirectly include this file:



## Functions

- `int l4util_cmpxchg32` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` cmp\_val, `l4_uint32_t` new\_val)  
*Atomic compare and exchange (32 bit version).*
- `int l4util_cmpxchg16` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` cmp\_val, `l4_uint16_t` new\_val)  
*Atomic compare and exchange (16 bit version).*
- `int l4util_cmpxchg8` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` cmp\_val, `l4_uint8_t` new\_val)  
*Atomic compare and exchange (8 bit version).*
- `int l4util_cmpxchg` (volatile `l4_umword_t` \*dest, `l4_umword_t` cmp\_val, `l4_umword_t` new\_val)  
*Atomic compare and exchange (machine wide fields).*
- `l4_uint32_t l4util_xchg32` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` val)  
*Atomic exchange (32 bit version).*
- `l4_uint16_t l4util_xchg16` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)  
*Atomic exchange (16 bit version).*
- `l4_uint8_t l4util_xchg8` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)  
*Atomic exchange (8 bit version).*
- `l4_umword_t l4util_xchg` (volatile `l4_umword_t` \*dest, `l4_umword_t` val)  
*Atomic exchange (machine wide fields).*
- `void l4util_atomic_add` (volatile long \*dest, long val)  
*Atomic add.*
- `void l4util_atomic_inc` (volatile long \*dest)  
*Atomic increment.*

## Atomic add/sub/and/or (8,16,32 bit version) without result

- `void l4util_add8` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)
- `void l4util_add16` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)
- `void l4util_add32` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` val)
- `void l4util_sub8` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)
- `void l4util_sub16` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)
- `void l4util_sub32` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` val)
- `void l4util_and8` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)
- `void l4util_and16` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)

- void [l4util\\_and32](#) (volatile [l4\\_uint32\\_t](#) \*dest, [l4\\_uint32\\_t](#) val)
- void [l4util\\_or8](#) (volatile [l4\\_uint8\\_t](#) \*dest, [l4\\_uint8\\_t](#) val)
- void [l4util\\_or16](#) (volatile [l4\\_uint16\\_t](#) \*dest, [l4\\_uint16\\_t](#) val)
- void [l4util\\_or32](#) (volatile [l4\\_uint32\\_t](#) \*dest, [l4\\_uint32\\_t](#) val)

#### Atomic add/sub/and/or operations (8,16,32 bit) with result

- [l4\\_uint8\\_t l4util\\_add8\\_res](#) (volatile [l4\\_uint8\\_t](#) \*dest, [l4\\_uint8\\_t](#) val)
- [l4\\_uint16\\_t l4util\\_add16\\_res](#) (volatile [l4\\_uint16\\_t](#) \*dest, [l4\\_uint16\\_t](#) val)
- [l4\\_uint32\\_t l4util\\_add32\\_res](#) (volatile [l4\\_uint32\\_t](#) \*dest, [l4\\_uint32\\_t](#) val)
- [l4\\_uint8\\_t l4util\\_sub8\\_res](#) (volatile [l4\\_uint8\\_t](#) \*dest, [l4\\_uint8\\_t](#) val)
- [l4\\_uint16\\_t l4util\\_sub16\\_res](#) (volatile [l4\\_uint16\\_t](#) \*dest, [l4\\_uint16\\_t](#) val)
- [l4\\_uint32\\_t l4util\\_sub32\\_res](#) (volatile [l4\\_uint32\\_t](#) \*dest, [l4\\_uint32\\_t](#) val)
- [l4\\_uint8\\_t l4util\\_and8\\_res](#) (volatile [l4\\_uint8\\_t](#) \*dest, [l4\\_uint8\\_t](#) val)
- [l4\\_uint16\\_t l4util\\_and16\\_res](#) (volatile [l4\\_uint16\\_t](#) \*dest, [l4\\_uint16\\_t](#) val)
- [l4\\_uint32\\_t l4util\\_and32\\_res](#) (volatile [l4\\_uint32\\_t](#) \*dest, [l4\\_uint32\\_t](#) val)
- [l4\\_uint8\\_t l4util\\_or8\\_res](#) (volatile [l4\\_uint8\\_t](#) \*dest, [l4\\_uint8\\_t](#) val)
- [l4\\_uint16\\_t l4util\\_or16\\_res](#) (volatile [l4\\_uint16\\_t](#) \*dest, [l4\\_uint16\\_t](#) val)
- [l4\\_uint32\\_t l4util\\_or32\\_res](#) (volatile [l4\\_uint32\\_t](#) \*dest, [l4\\_uint32\\_t](#) val)

#### Atomic inc/dec (8,16,32 bit) without result

- void [l4util\\_inc8](#) (volatile [l4\\_uint8\\_t](#) \*dest)
- void [l4util\\_inc16](#) (volatile [l4\\_uint16\\_t](#) \*dest)
- void [l4util\\_inc32](#) (volatile [l4\\_uint32\\_t](#) \*dest)
- void [l4util\\_dec8](#) (volatile [l4\\_uint8\\_t](#) \*dest)
- void [l4util\\_dec16](#) (volatile [l4\\_uint16\\_t](#) \*dest)
- void [l4util\\_dec32](#) (volatile [l4\\_uint32\\_t](#) \*dest)

#### Atomic inc/dec (8,16,32 bit) with result

- [l4\\_uint8\\_t l4util\\_inc8\\_res](#) (volatile [l4\\_uint8\\_t](#) \*dest)
- [l4\\_uint16\\_t l4util\\_inc16\\_res](#) (volatile [l4\\_uint16\\_t](#) \*dest)
- [l4\\_uint32\\_t l4util\\_inc32\\_res](#) (volatile [l4\\_uint32\\_t](#) \*dest)
- [l4\\_uint8\\_t l4util\\_dec8\\_res](#) (volatile [l4\\_uint8\\_t](#) \*dest)
- [l4\\_uint16\\_t l4util\\_dec16\\_res](#) (volatile [l4\\_uint16\\_t](#) \*dest)
- [l4\\_uint32\\_t l4util\\_dec32\\_res](#) (volatile [l4\\_uint32\\_t](#) \*dest)

## 17.631.1 Detailed Description

atomic operations header and generic implementations

Date

10/20/2000

Author

Lars Reuther [reuther@os.inf.tu-dresden.de](mailto:reuther@os.inf.tu-dresden.de), Jork Loeser [jork@os.inf.tu-dresden.de](mailto:jork@os.inf.tu-dresden.de)

Definition in file [atomic.h](#).



## 17.632 atomic.h

[Go to the documentation of this file.](#)

```

00001 /*****
00010 */
00011 * (c) 2000-2009 Author(s)
00012 *     economic rights: Technische Universität Dresden (Germany)
00013 * License: see LICENSE.spdx (in this directory or the directories above)
00014 */
00015
00016 /*****
00017 #ifndef __L4UTIL__INCLUDE__ATOMIC_H__
00018 #define __L4UTIL__INCLUDE__ATOMIC_H__
00019
00020 #include <l4/sys/l4int.h>
00021 #include <l4/sys/compiler.h>
00022
00023 /*****
00024 *** Prototypes
00025 *****/
00026
00027 L4_BEGIN_DECLS
00028
00029
00030
00031 #if __SIZEOF_LONG__ == 8
00032
00033
00034 L4_INLINE int
00035 l4util_cmpxchg64(volatile l4_uint64_t * dest,
00036                 l4_uint64_t cmp_val, l4_uint64_t new_val);
00037
00038 #endif
00039
00040 L4_INLINE int
00041 l4util_cmpxchg32(volatile l4_uint32_t * dest,
00042                 l4_uint32_t cmp_val, l4_uint32_t new_val);
00043
00044 L4_INLINE int
00045 l4util_cmpxchg16(volatile l4_uint16_t * dest,
00046                 l4_uint16_t cmp_val, l4_uint16_t new_val);
00047
00048 L4_INLINE int
00049 l4util_cmpxchg8(volatile l4_uint8_t * dest,
00050                l4_uint8_t cmp_val, l4_uint8_t new_val);
00051
00052 L4_INLINE int
00053 l4util_cmpxchg(volatile l4_umword_t * dest,
00054               l4_umword_t cmp_val, l4_umword_t new_val);
00055
00056 L4_INLINE l4_uint32_t
00057 l4util_xchg32(volatile l4_uint32_t * dest, l4_uint32_t val);
00058
00059 L4_INLINE l4_uint16_t
00060 l4util_xchg16(volatile l4_uint16_t * dest, l4_uint16_t val);
00061
00062 L4_INLINE l4_uint8_t
00063 l4util_xchg8(volatile l4_uint8_t * dest, l4_uint8_t val);
00064
00065 L4_INLINE l4_umword_t
00066 l4util_xchg(volatile l4_umword_t * dest, l4_umword_t val);
00067
00068
00069 L4_INLINE void
00070 l4util_add8(volatile l4_uint8_t *dest, l4_uint8_t val);
00071 L4_INLINE void
00072 l4util_add16(volatile l4_uint16_t *dest, l4_uint16_t val);
00073 L4_INLINE void
00074 l4util_add32(volatile l4_uint32_t *dest, l4_uint32_t val);
00075 L4_INLINE void
00076 l4util_sub8(volatile l4_uint8_t *dest, l4_uint8_t val);
00077 L4_INLINE void
00078 l4util_sub16(volatile l4_uint16_t *dest, l4_uint16_t val);
00079 L4_INLINE void
00080 l4util_sub32(volatile l4_uint32_t *dest, l4_uint32_t val);
00081 L4_INLINE void
00082 l4util_and8(volatile l4_uint8_t *dest, l4_uint8_t val);
00083 L4_INLINE void
00084 l4util_and16(volatile l4_uint16_t *dest, l4_uint16_t val);
00085 L4_INLINE void
00086 l4util_and32(volatile l4_uint32_t *dest, l4_uint32_t val);
00087 L4_INLINE void
00088 l4util_or8(volatile l4_uint8_t *dest, l4_uint8_t val);
00089 L4_INLINE void
00090 l4util_or16(volatile l4_uint16_t *dest, l4_uint16_t val);
00091 L4_INLINE void
00092 l4util_or32(volatile l4_uint32_t *dest, l4_uint32_t val);

```

```

00220
00222
00229 L4_INLINE l4_uint8_t
00230 l4util_add8_res(volatile l4_uint8_t *dest, l4_uint8_t val);
00232 L4_INLINE l4_uint16_t
00233 l4util_add16_res(volatile l4_uint16_t *dest, l4_uint16_t val);
00235 L4_INLINE l4_uint32_t
00236 l4util_add32_res(volatile l4_uint32_t *dest, l4_uint32_t val);
00238 L4_INLINE l4_uint8_t
00239 l4util_sub8_res(volatile l4_uint8_t *dest, l4_uint8_t val);
00241 L4_INLINE l4_uint16_t
00242 l4util_sub16_res(volatile l4_uint16_t *dest, l4_uint16_t val);
00244 L4_INLINE l4_uint32_t
00245 l4util_sub32_res(volatile l4_uint32_t *dest, l4_uint32_t val);
00247 L4_INLINE l4_uint8_t
00248 l4util_and8_res(volatile l4_uint8_t *dest, l4_uint8_t val);
00250 L4_INLINE l4_uint16_t
00251 l4util_and16_res(volatile l4_uint16_t *dest, l4_uint16_t val);
00253 L4_INLINE l4_uint32_t
00254 l4util_and32_res(volatile l4_uint32_t *dest, l4_uint32_t val);
00256 L4_INLINE l4_uint8_t
00257 l4util_or8_res(volatile l4_uint8_t *dest, l4_uint8_t val);
00259 L4_INLINE l4_uint16_t
00260 l4util_or16_res(volatile l4_uint16_t *dest, l4_uint16_t val);
00262 L4_INLINE l4_uint32_t
00263 l4util_or32_res(volatile l4_uint32_t *dest, l4_uint32_t val);
00265
00267
00272 L4_INLINE void
00273 l4util_inc8(volatile l4_uint8_t *dest);
00275 L4_INLINE void
00276 l4util_inc16(volatile l4_uint16_t *dest);
00278 L4_INLINE void
00279 l4util_inc32(volatile l4_uint32_t *dest);
00281 L4_INLINE void
00282 l4util_dec8(volatile l4_uint8_t *dest);
00284 L4_INLINE void
00285 l4util_dec16(volatile l4_uint16_t *dest);
00287 L4_INLINE void
00288 l4util_dec32(volatile l4_uint32_t *dest);
00290
00292
00298 L4_INLINE l4_uint8_t
00299 l4util_inc8_res(volatile l4_uint8_t *dest);
00301 L4_INLINE l4_uint16_t
00302 l4util_inc16_res(volatile l4_uint16_t *dest);
00304 L4_INLINE l4_uint32_t
00305 l4util_inc32_res(volatile l4_uint32_t *dest);
00307 L4_INLINE l4_uint8_t
00308 l4util_dec8_res(volatile l4_uint8_t *dest);
00310 L4_INLINE l4_uint16_t
00311 l4util_dec16_res(volatile l4_uint16_t *dest);
00313 L4_INLINE l4_uint32_t
00314 l4util_dec32_res(volatile l4_uint32_t *dest);
00316
00324 L4_INLINE void
00325 l4util_atomic_add(volatile long *dest, long val);
00326
00333 L4_INLINE void
00334 l4util_atomic_inc(volatile long *dest);
00335
00336 L4_END_DECLS
00337
00338 /*****
00339  * IMPLEMENTATION
00340  *****/
00341
00342 #if __SIZEOF_LONG__ == 8
00343
00344 L4_INLINE int
00345 l4util_cmpxchg64(volatile l4_uint64_t * dest,
00346                  l4_uint64_t cmp_val, l4_uint64_t new_val)
00347 {
00348     return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00349                                         __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00350 }
00351
00352 #endif
00353
00354 L4_INLINE int
00355 l4util_cmpxchg32(volatile l4_uint32_t * dest,
00356                  l4_uint32_t cmp_val, l4_uint32_t new_val)
00357 {
00358     return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00359                                         __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00360 }
00361

```

```

00362 L4_INLINE int
00363 l4util_cmpxchg16(volatile l4_uint16_t * dest,
00364                  l4_uint16_t cmp_val, l4_uint16_t new_val)
00365 {
00366     return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00367                                         __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00368 }
00369
00370 L4_INLINE int
00371 l4util_cmpxchg8(volatile l4_uint8_t * dest,
00372                 l4_uint8_t cmp_val, l4_uint8_t new_val)
00373 {
00374     return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00375                                         __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00376 }
00377
00378 L4_INLINE int
00379 l4util_cmpxchg(volatile l4_umword_t * dest,
00380                l4_umword_t cmp_val, l4_umword_t new_val)
00381 {
00382     return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00383                                         __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00384 }
00385
00386 L4_INLINE l4_uint32_t
00387 l4util_xchg32(volatile l4_uint32_t * dest, l4_uint32_t val)
00388 {
00389     return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
00390 }
00391
00392 L4_INLINE l4_uint16_t
00393 l4util_xchg16(volatile l4_uint16_t * dest, l4_uint16_t val)
00394 {
00395     return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
00396 }
00397
00398 L4_INLINE l4_uint8_t
00399 l4util_xchg8(volatile l4_uint8_t * dest, l4_uint8_t val)
00400 {
00401     return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
00402 }
00403
00404 L4_INLINE l4_umword_t
00405 l4util_xchg(volatile l4_umword_t * dest, l4_umword_t val)
00406 {
00407     return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
00408 }
00409
00410 L4_INLINE void
00411 l4util_inc8(volatile l4_uint8_t *dest)
00412 { __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }
00413
00414 L4_INLINE void
00415 l4util_inc16(volatile l4_uint16_t *dest)
00416 { __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }
00417
00418 L4_INLINE void
00419 l4util_inc32(volatile l4_uint32_t *dest)
00420 { __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }
00421
00422 L4_INLINE void
00423 l4util_atomic_inc(volatile long *dest)
00424 { __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }
00425
00426 L4_INLINE void
00427 l4util_dec8(volatile l4_uint8_t *dest)
00428 { __atomic_fetch_sub(dest, 1, __ATOMIC_SEQ_CST); }
00429
00430 L4_INLINE void
00431 l4util_dec16(volatile l4_uint16_t *dest)
00432 { __atomic_fetch_sub(dest, 1, __ATOMIC_SEQ_CST); }
00433
00434 L4_INLINE void
00435 l4util_dec32(volatile l4_uint32_t *dest)
00436 { __atomic_fetch_sub(dest, 1, __ATOMIC_SEQ_CST); }
00437
00438
00439 L4_INLINE l4_uint8_t
00440 l4util_inc8_res(volatile l4_uint8_t *dest)
00441 { return __atomic_add_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00442
00443 L4_INLINE l4_uint16_t
00444 l4util_inc16_res(volatile l4_uint16_t *dest)
00445 { return __atomic_add_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00446
00447 L4_INLINE l4_uint32_t
00448 l4util_inc32_res(volatile l4_uint32_t *dest)

```

```

00449 { return __atomic_add_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00450
00451 L4_INLINE l4_uint8_t
00452 l4util_dec8_res(volatile l4_uint8_t *dest)
00453 { return __atomic_sub_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00454
00455 L4_INLINE l4_uint16_t
00456 l4util_dec16_res(volatile l4_uint16_t *dest)
00457 { return __atomic_sub_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00458
00459 L4_INLINE l4_uint32_t
00460 l4util_dec32_res(volatile l4_uint32_t *dest)
00461 { return __atomic_sub_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00462
00463 L4_INLINE l4_umword_t
00464 l4util_dec_res(volatile l4_umword_t *dest)
00465 { return __atomic_sub_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00466
00467 L4_INLINE void
00468 l4util_add8(volatile l4_uint8_t *dest, l4_uint8_t val)
00469 { __atomic_fetch_add(dest, val, __ATOMIC_SEQ_CST); }
00470
00471 L4_INLINE void
00472 l4util_add16(volatile l4_uint16_t *dest, l4_uint16_t val)
00473 { __atomic_fetch_add(dest, val, __ATOMIC_SEQ_CST); }
00474
00475 L4_INLINE void
00476 l4util_add32(volatile l4_uint32_t *dest, l4_uint32_t val)
00477 { __atomic_fetch_add(dest, val, __ATOMIC_SEQ_CST); }
00478
00479 L4_INLINE void
00480 l4util_atomic_add(volatile long *dest, long val)
00481 { __atomic_fetch_add(dest, val, __ATOMIC_SEQ_CST); }
00482
00483 L4_INLINE void
00484 l4util_sub8(volatile l4_uint8_t *dest, l4_uint8_t val)
00485 { __atomic_fetch_sub(dest, val, __ATOMIC_SEQ_CST); }
00486
00487 L4_INLINE void
00488 l4util_sub16(volatile l4_uint16_t *dest, l4_uint16_t val)
00489 { __atomic_fetch_sub(dest, val, __ATOMIC_SEQ_CST); }
00490
00491 L4_INLINE void
00492 l4util_sub32(volatile l4_uint32_t *dest, l4_uint32_t val)
00493 { __atomic_fetch_sub(dest, val, __ATOMIC_SEQ_CST); }
00494
00495 L4_INLINE void
00496 l4util_and8(volatile l4_uint8_t *dest, l4_uint8_t val)
00497 { __atomic_fetch_and(dest, val, __ATOMIC_SEQ_CST); }
00498
00499 L4_INLINE void
00500 l4util_and16(volatile l4_uint16_t *dest, l4_uint16_t val)
00501 { __atomic_fetch_and(dest, val, __ATOMIC_SEQ_CST); }
00502
00503 L4_INLINE void
00504 l4util_and32(volatile l4_uint32_t *dest, l4_uint32_t val)
00505 { __atomic_fetch_and(dest, val, __ATOMIC_SEQ_CST); }
00506
00507 L4_INLINE void
00508 l4util_or8(volatile l4_uint8_t *dest, l4_uint8_t val)
00509 { __atomic_fetch_or(dest, val, __ATOMIC_SEQ_CST); }
00510
00511 L4_INLINE void
00512 l4util_or16(volatile l4_uint16_t *dest, l4_uint16_t val)
00513 { __atomic_fetch_or(dest, val, __ATOMIC_SEQ_CST); }
00514
00515 L4_INLINE void
00516 l4util_or32(volatile l4_uint32_t *dest, l4_uint32_t val)
00517 { __atomic_fetch_or(dest, val, __ATOMIC_SEQ_CST); }
00518
00519 L4_INLINE l4_uint8_t
00520 l4util_add8_res(volatile l4_uint8_t *dest, l4_uint8_t val)
00521 { return __atomic_add_fetch(dest, val, __ATOMIC_SEQ_CST); }
00522
00523 L4_INLINE l4_uint16_t
00524 l4util_add16_res(volatile l4_uint16_t *dest, l4_uint16_t val)
00525 { return __atomic_add_fetch(dest, val, __ATOMIC_SEQ_CST); }
00526
00527 L4_INLINE l4_uint32_t
00528 l4util_add32_res(volatile l4_uint32_t *dest, l4_uint32_t val)
00529 { return __atomic_add_fetch(dest, val, __ATOMIC_SEQ_CST); }
00530
00531 L4_INLINE l4_uint8_t
00532 l4util_sub8_res(volatile l4_uint8_t *dest, l4_uint8_t val)
00533 { return __atomic_sub_fetch(dest, val, __ATOMIC_SEQ_CST); }
00534
00535 L4_INLINE l4_uint16_t

```

```

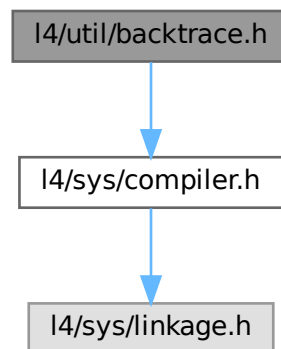
00536 l4util_sub16_res(volatile l4_uint16_t *dest, l4_uint16_t val)
00537 { return __atomic_sub_fetch(dest, val, __ATOMIC_SEQ_CST); }
00538
00539 L4_INLINE l4_uint32_t
00540 l4util_sub32_res(volatile l4_uint32_t *dest, l4_uint32_t val)
00541 { return __atomic_sub_fetch(dest, val, __ATOMIC_SEQ_CST); }
00542
00543 L4_INLINE l4_uint8_t
00544 l4util_and8_res(volatile l4_uint8_t *dest, l4_uint8_t val)
00545 { return __atomic_and_fetch(dest, val, __ATOMIC_SEQ_CST); }
00546
00547 L4_INLINE l4_uint16_t
00548 l4util_and16_res(volatile l4_uint16_t *dest, l4_uint16_t val)
00549 { return __atomic_and_fetch(dest, val, __ATOMIC_SEQ_CST); }
00550
00551 L4_INLINE l4_uint32_t
00552 l4util_and32_res(volatile l4_uint32_t *dest, l4_uint32_t val)
00553 { return __atomic_and_fetch(dest, val, __ATOMIC_SEQ_CST); }
00554
00555 L4_INLINE l4_uint8_t
00556 l4util_or8_res(volatile l4_uint8_t *dest, l4_uint8_t val)
00557 { return __atomic_or_fetch(dest, val, __ATOMIC_SEQ_CST); }
00558
00559 L4_INLINE l4_uint16_t
00560 l4util_or16_res(volatile l4_uint16_t *dest, l4_uint16_t val)
00561 { return __atomic_or_fetch(dest, val, __ATOMIC_SEQ_CST); }
00562
00563 L4_INLINE l4_uint32_t
00564 l4util_or32_res(volatile l4_uint32_t *dest, l4_uint32_t val)
00565 { return __atomic_or_fetch(dest, val, __ATOMIC_SEQ_CST); }
00566
00567 #endif /* ! __L4UTIL__INCLUDE__ATOMIC_H__ */

```

## 17.633 l4/util/backtrace.h File Reference

Backtrace.

#include <l4/sys/compiler.h>  
 Include dependency graph for backtrace.h:





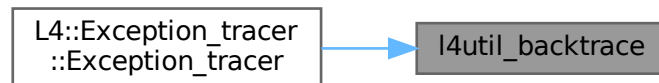
**Returns**

Number of entries

References [L4\\_END\\_DECLS](#).

Referenced by [L4::Exception\\_tracer::Exception\\_tracer\(\)](#).

Here is the caller graph for this function:

**17.634 backtrace.h**

[Go to the documentation of this file.](#)

```

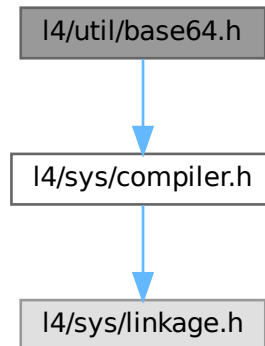
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *           Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *           economic rights: Technische Universität Dresden (Germany)
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00013 #include <l4/sys/compiler.h>
00014
00015 L4_BEGIN_DECLS
00016
00024 int l4util_backtrace(void **pc_array, int max_len);
00025
00026 L4_END_DECLS
  
```

**17.635 l4/util/base64.h File Reference**

base 64 encoding and decoding functions adapted from Bob Trower 08/04/01

```
#include <l4/sys/compiler.h>
```

Include dependency graph for base64.h:



## Functions

- void **base64\_encode** (const char \*infile, unsigned int in\_size, char \*\*outfile)  
*base-64-encode string infile*
- void **base64\_decode** (const char \*infile, unsigned int in\_size, char \*\*outfile)  
*decode base-64-encoded string infile*

### 17.635.1 Detailed Description

base 64 encoding and decoding functions adapted from Bob Trower 08/04/01

#### Date

04/26/2002

#### Author

Joerg Nothnagel [jn6@os.inf.tu-dresden.de](mailto:jn6@os.inf.tu-dresden.de)

Definition in file [base64.h](#).



## 17.636 base64.h

[Go to the documentation of this file.](#)

```

00001
00009 /*
00010  * (c) 2008-2009 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014
00015 #ifndef B64_EN_DECODE
00016 #define B64_EN_DECODE
00017
00018 #include <l4/sys/compiler.h>
00019
00020 L4_BEGIN_DECLS
00021
00027
00038 L4_CV void base64_encode( const char *infile, unsigned int in_size, char **outfile);
00039
00050 L4_CV void base64_decode(const char *infile, unsigned int in_size, char **outfile);
00051
00052 L4_END_DECLS
00053
00055 #endif //B64_EN_DECODE

```

## 17.637 l4/util/bitops.h File Reference

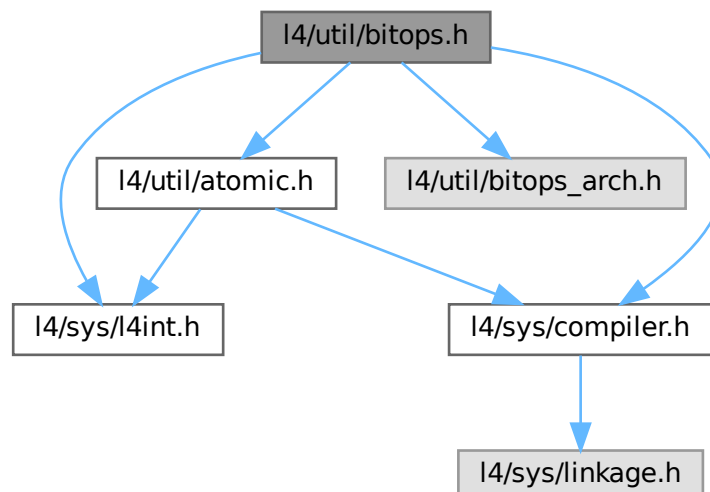
bit manipulation functions

```

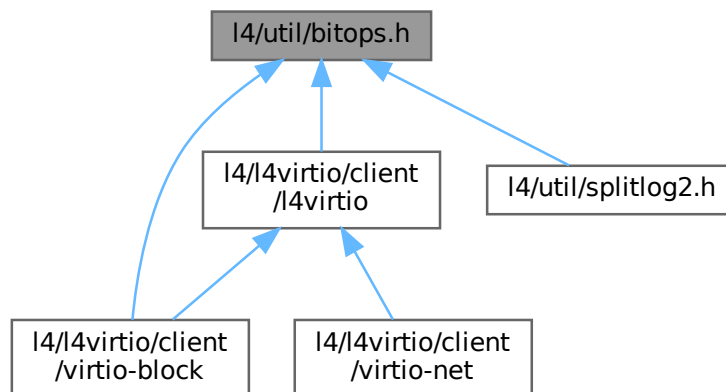
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>
#include <l4/util/bitops_arch.h>
#include <l4/util/atomic.h>

```

Include dependency graph for bitops.h:



This graph shows which files directly or indirectly include this file:



## Macros

- `#define I4util_test_and_clear_bit(b, dest)`  
*define some more usual names*

## Functions

- void `I4util_set_bit` (int b, volatile `I4_umword_t` \*dest)  
*Set bit in memory.*
- void `I4util_clear_bit` (int b, volatile `I4_umword_t` \*dest)  
*Clear bit in memory.*
- void `I4util_complement_bit` (int b, volatile `I4_umword_t` \*dest)  
*Complement bit in memory.*
- int `I4util_test_bit` (int b, const volatile `I4_umword_t` \*dest)  
*Test bit (return value of bit).*
- int `I4util_bts` (int b, volatile `I4_umword_t` \*dest)  
*Bit test and set.*
- int `I4util_btr` (int b, volatile `I4_umword_t` \*dest)  
*Bit test and reset.*
- int `I4util_btc` (int b, volatile `I4_umword_t` \*dest)  
*Bit test and complement.*
- int `I4util_bsr` (`I4_umword_t` word)  
*Bit scan reverse.*
- int `I4util_bsf` (`I4_umword_t` word)  
*Bit scan forward.*
- int `I4util_find_first_set_bit` (const void \*dest, `I4_size_t` size)  
*Find the first set bit in a memory region.*
- int `I4util_find_first_zero_bit` (const void \*dest, `I4_size_t` size)  
*Find the first zero bit in a memory region.*
- int `I4util_next_power2` (unsigned long val)  
*Find the next power of 2 for a given number.*

## 17.637.1 Detailed Description

bit manipulation functions

### Date

07/03/2001

### Author

Lars Reuther [reuther@os.inf.tu-dresden.de](mailto:reuther@os.inf.tu-dresden.de)

Definition in file [bitops.h](#).

## 17.638 bitops.h

[Go to the documentation of this file.](#)

```

00001  /*****
00009  */
00010  * (c) 2000-2009 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014
00015  /*****
00016  #ifndef __L4UTIL__INCLUDE__BITOPS_H__
00017  #define __L4UTIL__INCLUDE__BITOPS_H__
00018
00019  /* L4 includes */
00020  #include <l4/sys/l4int.h>
00021  #include <l4/sys/compiler.h>
00022
00024  #define l4util_test_and_clear_bit(b, dest)  l4util_btr(b, dest)
00025  #define l4util_test_and_set_bit(b, dest)    l4util_bts(b, dest)
00026  #define l4util_test_and_change_bit(b, dest) l4util_btc(b, dest)
00027  #define l4util_log2(word)                  l4util_bsr(word)
00028
00029  /*****
00030  *** Prototypes
00031  *****/
00032
00033  L4_BEGIN_DECLS
00034
00039
00047  L4_INLINE void
00048  l4util_set_bit(int b, volatile l4_umword_t * dest);
00049
00057  L4_INLINE void
00058  l4util_clear_bit(int b, volatile l4_umword_t * dest);
00059
00067  L4_INLINE void
00068  l4util_complement_bit(int b, volatile l4_umword_t * dest);
00069
00079  L4_INLINE int
00080  l4util_test_bit(int b, const volatile l4_umword_t * dest);
00081
00093  L4_INLINE int
00094  l4util_bts(int b, volatile l4_umword_t * dest);
00095
00107  L4_INLINE int
00108  l4util_btr(int b, volatile l4_umword_t * dest);
00109
00121  L4_INLINE int
00122  l4util_btc(int b, volatile l4_umword_t * dest);
00123
00135  L4_INLINE int
00136  l4util_bsr(l4_umword_t word);
00137
00149  L4_INLINE int
00150  l4util_bsf(l4_umword_t word);
00151
00163  L4_INLINE int

```

```

00164 l4util_find_first_set_bit(const void * dest, l4_size_t size);
00165
00177 L4_INLINE int
00178 l4util_find_first_zero_bit(const void * dest, l4_size_t size);
00179
00180
00189 L4_INLINE int
00190 l4util_next_power2(unsigned long val);
00191
00192 L4_END_DECLS
00193
00194 /*****
00195  *** Implementation of specific version
00196  *****/
00197
00198 #include <l4/util/bitops_arch.h>
00199
00200 /*****
00201  *** Generic implementations
00202  *****/
00203
00204 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_SET_BIT
00205 #include <l4/util/atomic.h>
00206 L4_INLINE void
00207 l4util_set_bit(int b, volatile l4_umword_t * dest)
00208 {
00209     l4_umword_t oldval, newval;
00210
00211     dest += b / (sizeof(*dest) * 8); /* advance dest to the proper element */
00212     b    &= sizeof(*dest) * 8 - 1; /* modulo; cut off all upper bits */
00213
00214     do
00215     {
00216         oldval = *dest;
00217         newval = oldval | (1UL << b);
00218     }
00219     while (!l4util_cmpxchg(dest, oldval, newval));
00220 }
00221 #endif
00222
00223 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_CLEAR_BIT
00224 #include <l4/util/atomic.h>
00225 L4_INLINE void
00226 l4util_clear_bit(int b, volatile l4_umword_t * dest)
00227 {
00228     l4_umword_t oldval, newval;
00229
00230     dest += b / (sizeof(*dest) * 8);
00231     b    &= sizeof(*dest) * 8 - 1;
00232
00233     do
00234     {
00235         oldval = *dest;
00236         newval = oldval & ~(1UL << b);
00237     }
00238     while (!l4util_cmpxchg(dest, oldval, newval));
00239 }
00240 #endif
00241
00242 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_TEST_BIT
00243 L4_INLINE int
00244 l4util_test_bit(int b, const volatile l4_umword_t * dest)
00245 {
00246     dest += b / (sizeof(*dest) * 8);
00247     b    &= sizeof(*dest) * 8 - 1;
00248
00249     return (*dest >> b) & 1;
00250 }
00251 #endif
00252
00253 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_SET
00254 #include <l4/util/atomic.h>
00255 L4_INLINE int
00256 l4util_bts(int b, volatile l4_umword_t * dest)
00257 {
00258     l4_umword_t oldval, newval;
00259
00260     dest += b / (sizeof(*dest) * 8);
00261     b    &= sizeof(*dest) * 8 - 1;
00262
00263     do
00264     {
00265         oldval = *dest;
00266         newval = oldval | (1UL << b);
00267     }
00268     while (!l4util_cmpxchg(dest, oldval, newval));
00269

```

```

00270  /* Return old bit */
00271  return (oldval >> b) & 1;
00272 }
00273 #endif
00274
00275 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_RESET
00276 #include <l4/util/atomic.h>
00277 L4_INLINE int
00278 l4util_btr(int b, volatile l4_umword_t * dest)
00279 {
00280     l4_umword_t oldval, newval;
00281
00282     dest += b / (sizeof(*dest) * 8);
00283     b    &= sizeof(*dest) * 8 - 1;
00284
00285     do
00286     {
00287         oldval = *dest;
00288         newval = oldval & ~(1UL << b);
00289     }
00290     while (!l4util_cmpxchg(dest, oldval, newval));
00291
00292     /* Return old bit */
00293     return (oldval >> b) & 1;
00294 }
00295 #endif
00296
00297 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_REVERSE
00298 L4_INLINE int
00299 l4util_bsr(l4_umword_t word)
00300 {
00301     int i;
00302
00303     if (!word)
00304         return -1;
00305
00306     for (i = 8 * sizeof(word) - 1; i >= 0; i--)
00307         if ((1UL << i) & word)
00308             return i;
00309
00310     __builtin_unreachable();
00311 }
00312 #endif
00313
00314 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_FORWARD
00315 L4_INLINE int
00316 l4util_bsf(l4_umword_t word)
00317 {
00318     unsigned int i;
00319
00320     if (!word)
00321         return -1;
00322
00323     for (i = 0; i < sizeof(word) * 8; i++)
00324         if ((1UL << i) & word)
00325             return i;
00326
00327     __builtin_unreachable();
00328 }
00329 #endif
00330
00331 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_ZERO_BIT
00332 L4_INLINE int
00333 l4util_find_first_zero_bit(const void * dest, l4_size_t size)
00334 {
00335     l4_size_t i, j;
00336     unsigned long *v = (unsigned long*)dest;
00337
00338     if (!size)
00339         return 0;
00340
00341     size = (size + 31) & ~0x1f; /* Grmbl: adapt to x86 implementation... */
00342
00343     for (i = j = 0; i < size; i++, j++)
00344     {
00345         if (j >= sizeof(*v) * 8)
00346         {
00347             j = 0;
00348             v++;
00349         }
00350         if (!((1UL << j) & *v))
00351             return i;
00352     }
00353     return size + 1;
00354 }
00355 #endif
00356

```

```

00357 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_COMPLEMENT_BIT
00358 L4_INLINE void
00359 l4util_complement_bit(int b, volatile l4_umword_t * dest)
00360 {
00361     dest += b / (sizeof(*dest) * 8);
00362     b    &= sizeof(*dest) * 8 - 1;
00363     *dest ^= 1UL << b;
00364 }
00365 #endif
00366
00367 /*
00368 * Adapted from:
00369 * http://en.wikipedia.org/wiki/Power_of_two#Algorithm_to_find_the_next-highest_power_of_two
00370 */
00371 L4_INLINE int
00372 l4util_next_power2(unsigned long val)
00373 {
00374     unsigned i;
00375     if (val == 0)
00376         return 1;
00377     val--;
00378     for (i=1; i < sizeof(unsigned long)*8; i++)
00379         val = val | val >> i;
00380     return val+1;
00381 }
00382
00383 /* Non-implemented version, catch with a linker warning */
00384 extern int __this_l4util_bitops_function_is_not_implemented_for_this_arch__sorry(void);
00385
00386 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_COMPLEMENT
00387 L4_INLINE int
00388 l4util_btc(int b, volatile l4_umword_t * dest)
00389 { (void)b; (void)dest; __this_l4util_bitops_function_is_not_implemented_for_this_arch__sorry(); return 0; }
00390 #endif
00391
00392 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_SET_BIT
00393 L4_INLINE int
00394 l4util_find_first_set_bit(const void * dest, l4_size_t size)
00395 { (void)dest; (void)size; __this_l4util_bitops_function_is_not_implemented_for_this_arch__sorry(); return 0; }
00396 #endif
00397
00398 #endif /* ! __L4UTIL__INCLUDE__BITOPS_H__ */

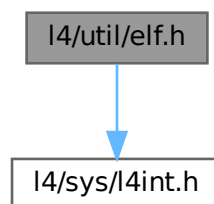
```

## 17.639 l4/util/elf.h File Reference

ELF definition.

```
#include <l4/sys/l4int.h>
```

Include dependency graph for elf.h:



## Data Structures

- struct [Elf32\\_Ehdr](#)  
*ELF32 header.*
- struct [Elf64\\_Ehdr](#)  
*ELF64 header.*
- struct [Elf32\\_Shdr](#)  
*ELF32 section header.*
- struct [Elf64\\_Shdr](#)  
*ELF64 section header.*
- struct [Elf32\\_Phdr](#)  
*ELF32 program header.*
- struct [Elf64\\_Phdr](#)  
*ELF64 program header.*
- struct [Elf32\\_Dyn](#)  
*ELF32 dynamic entry.*
- struct [Elf64\\_Dyn](#)  
*ELF64 dynamic entry.*
- struct [Elf32\\_Rel](#)  
*ELF32 relocation entry w/o addend.*
- struct [Elf32\\_Rela](#)  
*ELF32 relocation entry w/ addend.*
- struct [Elf64\\_Rel](#)  
*ELF64 relocation entry w/o addend.*
- struct [Elf64\\_Rela](#)  
*ELF64 relocation entry w/ addend.*
- struct [Elf32\\_Sym](#)  
*ELF32 symbol table entry.*
- struct [Elf64\\_Sym](#)  
*ELF64 symbol table entry.*
- struct [Elf32\\_Auxv](#)  
*Auxiliary vector (32-bit).*
- struct [Elf64\\_Auxv](#)  
*Auxiliary vector (64-bit).*

## Macros

- #define [ElfW](#)(type)  
*Use 64 or 32 bits types depending on the target architecture.*
- #define [ELF32\\_R\\_SYM](#)(i)  
*Symbol table index.*
- #define [ELF32\\_R\\_TYPE](#)(i)
- #define [ELF32\\_R\\_INFO](#)(s, t)  
*Create info from symbol table index + type.*
- #define [ELF64\\_R\\_SYM](#)(i)  
*Symbol table index.*
- #define [ELF64\\_R\\_TYPE](#)(i)
- #define [ELF64\\_R\\_INFO](#)(s, t)  
*Create info from symbol table index + type.*
- #define [ELF32\\_ST\\_BIND](#)(i)
- #define [ELF32\\_ST\\_TYPE](#)(i)

- `#define ELF32_ST_INFO(b, t)`  
*Make info from bind + type.*
- `#define ELF64_ST_BIND(i)`
- `#define ELF64_ST_TYPE(i)`
- `#define ELF64_ST_INFO(b, t)`  
*Make info from bind + type.*

## Typedefs

- `typedef struct Elf32_Auxv Elf32_Auxv`  
*Auxiliary vector (32-bit).*
- `typedef struct Elf64_Auxv Elf64_Auxv`  
*Auxiliary vector (64-bit).*

## ELF types

- `typedef l4\_uint32\_t Elf32_Addr`  
*size 4 align 4*
- `typedef l4\_uint32\_t Elf32_Off`  
*size 4 align 4*
- `typedef l4\_uint16\_t Elf32_Half`  
*size 2 align 2*
- `typedef l4\_uint32\_t Elf32_Word`  
*size 4 align 4*
- `typedef l4\_int32\_t Elf32_Sword`  
*size 4 align 4*
- `typedef l4\_uint64\_t Elf64_Addr`  
*size 8 align 8*
- `typedef l4\_uint64\_t Elf64_Off`  
*size 8 align 8*
- `typedef l4\_uint16\_t Elf64_Half`  
*size 2 align 2*
- `typedef l4\_uint32\_t Elf64_Word`  
*size 4 align 4*
- `typedef l4\_int32\_t Elf64_Sword`  
*size 4 align 4*
- `typedef l4\_uint64\_t Elf64_Xword`  
*size 8 align 8*
- `typedef l4\_int64\_t Elf64_Sxword`  
*size 8 align 8*

## Enumerations

- `enum { EI_NIDENT = 16 }`
- `enum Elf_ETs {`  
`ET\_NONE = 0 , ET\_REL = 1 , ET\_EXEC = 2 , ET\_DYN = 3 ,`  
`ET\_CORE = 4 , ET\_LOPROC = 0xff00 , ET\_HIPROC = 0xffff }`  
*Object file type.*



- enum `Elf_EMs` {  
`EM_NONE` = 0 , `EM_M32` = 1 , `EM_SPARC` = 2 , `EM_386` = 3 ,  
`EM_68K` = 4 , `EM_88K` = 5 , `EM_860` = 7 , `EM_MIPS` = 8 ,  
`EM_MIPS_RS4_BE` = 10 , `EM_SPARC64` = 11 , `EM_PARISC` = 15 , `EM_VPP500` = 17 ,  
`EM_SPARC32PLUS` = 18 , `EM_960` = 19 , `EM_PPC` = 20 , `EM_V800` = 36 ,  
`EM_FR20` = 37 , `EM_RH32` = 38 , `EM_RCE` = 39 , `EM_ARM` = 40 ,  
`EM_ALPHA` = 41 , `EM_SH` = 42 , `EM_SPARCV9` = 43 , `EM_TRICORE` = 44 ,  
`EM_ARC` = 45 , `EM_H8_300` = 46 , `EM_H8_300H` = 47 , `EM_H8S` = 48 ,  
`EM_H8_500` = 49 , `EM_IA_64` = 50 , `EM_MIPS_X` = 51 , `EM_COLDFIRE` = 52 ,  
`EM_68HC12` = 53 , `EM_X86_64` = 62 , `EM_PDSP` = 63 , `EM_FX66` = 66 ,  
`EM_ST9PLUS` = 67 , `EM_ST7` = 68 , `EM_68HC16` = 69 , `EM_68HC11` = 70 ,  
`EM_68HC08` = 71 , `EM_68HC05` = 72 , `EM_SVX` = 73 , `EM_ST19` = 74 ,  
`EM_VAX` = 75 , `EM_CRIS` = 76 , `EM_JAVELIN` = 77 , `EM_FIREPATH` = 78 ,  
`EM_ZSP` = 79 , `EM_MMIX` = 80 , `EM_HUANY` = 81 , `EM_PRISM` = 82 ,  
`EM_AVR` = 83 , `EM_FR30` = 84 , `EM_D10V` = 85 , `EM_D30V` = 86 ,  
`EM_V850` = 87 , `EM_M32R` = 88 , `EM_MN10300` = 89 , `EM_MN10200` = 90 ,  
`EM_PJ` = 91 , `EM_OPENRISC` = 92 , `EM_ARC_A5` = 93 , `EM_XTENSA` = 94 ,  
`EM_ALTERA_NIOS2` = 113 , `EM_AARCH64` = 183 , `EM_TILEPRO` = 188 , `EM_MICROBLAZE` = 189 ,  
`EM_TILEGX` = 191 , `EM_RISCV` = 243 , **`EM_NUM`** = 244 }  
*Required architecture.*
- enum `Elf_EVs` { `EV_NONE` = 0 , `EV_CURRENT` = 1 }  
*Object file version.*
- enum `Elf_EIs` {  
`EI_MAG0` = 0 , `EI_MAG1` = 1 , `EI_MAG2` = 2 , `EI_MAG3` = 3 ,  
`EI_CLASS` = 4 , `EI_DATA` = 5 , `EI_VERSION` = 6 , `EI_OSABI` = 7 ,  
`EI_ABIVERSION` = 8 , `EI_PAD` = 9 }  
*Identification Indices.*
- enum `Elf_MAGs` { `ELFMAG0` = 0x7f , `ELFMAG1` = 'E' , `ELFMAG2` = 'L' , `ELFMAG3` = 'F' }  
*Magic number.*
- enum `Elf_CIASSs` { `ELFCLASSNONE` = 0 , `ELFCLASS32` = 1 , `ELFCLASS64` = 2 , `ELFCLASSNUM` = 3 }  
*File class or capacity.*
- enum `Elf_DATAs` { `ELFDATANONE` = 0 , `ELFDATA2LSB` = 1 , `ELFDATA2MSB` = 2 , `ELFDATANUM` = 3 }  
*Data encoding.*
- enum `Elf_OSABIs` {  
`ELFOSABI_NONE` = 0 , `ELFOSABI_SYSV` = 0 , `ELFOSABI_HPUX` = 1 , `ELFOSABI_NETBSD` = 2 ,  
`ELFOSABI_LINUX` = 3 , `ELFOSABI_SOLARIS` = 6 , `ELFOSABI_AIX` = 7 , `ELFOSABI_IRIX` = 8 ,  
`ELFOSABI_FREEBSD` = 9 , `ELFOSABI_TRU64` = 10 , `ELFOSABI_MODESTO` = 11 , `ELFOSABI_OPENBSD`  
= 12 ,  
`ELFOSABI_ARM` = 97 , `ELFOSABI_STANDALONE` = 255 }  
*Identify operating system and ABI to which the object is targeted.*
- enum `Elf_SHNs` {  
`SHN_UNDEF` = 0 , `SHN_LORESERVE` = 0xff00 , `SHN_LOPROC` = 0xff00 , `SHN_HIPROC` = 0xff1f ,  
`SHN_ABS` = 0xffff1 , `SHN_COMMON` = 0xffff2 , `SHN_HIRESERVE` = 0xffff }  
*Special section indexes.*
- enum `Elf_SHTs` {  
`SHT_NULL` = 0 , `SHT_PROGBITS` = 1 , `SHT_SYMTAB` = 2 , `SHT_STRTAB` = 3 ,  
`SHT_RELA` = 4 , `SHT_HASH` = 5 , `SHT_DYNAMIC` = 6 , `SHT_NOTE` = 7 ,  
`SHT_NOBITS` = 8 , `SHT_REL` = 9 , `SHT_SHLIB` = 10 , `SHT_DYNSYM` = 11 ,  
`SHT_INIT_ARRAY` = 14 , `SHT_FINI_ARRAY` = 15 , `SHT_PREINIT_ARRAY` = 16 , `SHT_GROUP` = 17 ,  
`SHT_SYMTAB_SHNDX` = 18 , `SHT_NUM` = 19 , `SHT_LOOS` = 0x60000000 , `SHT_HIOS` = 0x6fffffff ,  
`SHT_LOPROC` = 0x70000000 , `SHT_HIPROC` = 0x7fffffff , `SHT_LOUSER` = 0x80000000 , `SHT_HIUSER` =  
0xffffffff }  
*Section type.*
- enum `Elf_SHFs` {  
`SHF_WRITE` = 0x1 , `SHF_ALLOC` = 0x2 , `SHF_EXECINSTR` = 0x4 , `SHF_MERGE` = 0x10 ,  
`SHF_STRINGS` = 0x20 , `SHF_INFO_LINK` = 0x40 , `SHF_OS_NONCONFORMING` = 0x100 , `SHF_GROUP`

= 0x200 ,  
 SHF\_TLS = 0x400 , SHF\_MASKOS = 0x0ff00000 , SHF\_MASKPROC = 0xf0000000 }

*Section attribute flags.*

- enum Elf\_PTs {  
 PT\_NULL = 0 , PT\_LOAD = 1 , PT\_DYNAMIC = 2 , PT\_INTERP = 3 ,  
 PT\_NOTE = 4 , PT\_SHLIB = 5 , PT\_PHDR = 6 , PT\_TLS = 7 ,  
 PT\_NUM = 8 , PT\_LOOS = 0x60000000 , PT\_HIOS = 0x6ffffff , PT\_LOPROC = 0x70000000 ,  
 PT\_HIPROC = 0x7ffffff , PT\_GNU\_EH\_FRAME = PT\_LOOS + 0x474e550 , PT\_GNU\_STACK = PT\_LOOS  
 + 0x474e551 , PT\_GNU\_RELRO = PT\_LOOS + 0x474e552 ,  
 PT\_L4\_STACK = PT\_LOOS + 0x12 , PT\_L4\_AUX = PT\_LOOS + 0x14 }

*Segment types.*

- enum ELF\_PFs {  
 PF\_X = 0x1 , PF\_W = 0x2 , PF\_R = 0x4 , PF\_MASKOS = 0x0ff00000 ,  
 PF\_MASKPROC = 0x7ffffff }

*Segment permissions.*

- enum Elf\_NTscore {  
 NT\_PRSTATUS = 1 , NT\_FPREGSET = 2 , NT\_PRPSINFO = 3 , NT\_PRXREG = 4 ,  
 NT\_TASKSTRUCT = 4 , NT\_PLATFORM = 5 , NT\_AUXV = 6 , NT\_GWINDOWS = 7 ,  
 NT\_ASRS = 8 , NT\_PSTATUS = 10 , NT\_PSINFO = 13 , NT\_PRCRED = 14 ,  
 NT\_UTSNAME = 15 , NT\_LWPSTATUS = 16 , NT\_LWPSINFO = 17 , NT\_PRFPXREG = 20 }

*Legal values for note segment descriptor types for core files.*

- enum Elf\_NTscobj { NT\_VERSION = 1 }

*Legal values for the note segment descriptor types for object files.*

- enum Elf\_DTsc {  
 DT\_NULL = 0 , DT\_NEEDED = 1 , DT\_PLTRELSZ = 2 , DT\_PLTGOT = 3 ,  
 DT\_HASH = 4 , DT\_STRTAB = 5 , DT\_SYMTAB = 6 , DT\_RELA = 7 ,  
 DT\_RELASZ = 8 , DT\_RELAENT = 9 , DT\_STRSZ = 10 , DT\_SYMENT = 11 ,  
 DT\_INIT = 12 , DT\_FINI = 13 , DT\_SONAME = 14 , DT\_RPATH = 15 ,  
 DT\_SYMBOLIC = 16 , DT\_REL = 17 , DT\_RELSZ = 18 , DT\_RELENT = 19 ,  
 DT\_PTRREL = 20 , DT\_DEBUG = 21 , DT\_TEXTREL = 22 , DT\_JMPREL = 23 ,  
 DT\_BIND\_NOW = 24 , DT\_INIT\_ARRAY = 25 , DT\_FINI\_ARRAY = 26 , DT\_INIT\_ARRAYSZ = 27 ,  
 DT\_FINI\_ARRAYSZ = 28 , DT\_RUNPATH = 29 , DT\_FLAGS = 30 , DT\_ENCODING = 32 ,  
 DT\_PREINIT\_ARRAY = 32 , DT\_PREINIT\_ARRAYSZ = 33 , DT\_NUM = 34 , DT\_LOOS = 0x6000000d ,  
 DT\_HIOS = 0x6ffffff0 , DT\_LOPROC = 0x70000000 , DT\_HIPROC = 0x7ffffff }

*Dynamic Array Tags.*

- enum Elf\_DFsc {  
 DF\_ORIGIN = 0x00000001 , DF\_SYMBOLIC = 0x00000002 , DF\_TEXTREL = 0x00000004 ,  
 DF\_BIND\_NOW = 0x00000008 ,  
 DF\_STATIC\_TLS = 0x00000010 }

*Values of Elf32\_Dyn.d\_un.d\_val, Elf64\_Dyn.d\_un.d\_val in the DT\_FLAGS entry.*

- enum Elf\_DF1sc {  
 DF\_1\_NOW = 0x00000001 , DF\_1\_GLOBAL = 0x00000002 , DF\_1\_GROUP = 0x00000004 ,  
 DF\_1\_NODELETE = 0x00000008 ,  
 DF\_1\_LOADFLTR = 0x00000010 , DF\_1\_INITFIRST = 0x00000020 , DF\_1\_NOOPEN = 0x00000040 ,  
 DF\_1\_ORIGIN = 0x00000080 ,  
 DF\_1\_DIRECT = 0x00000100 , DF\_1\_TRANS = 0x00000200 , DF\_1\_INTERPOSE = 0x00000400 ,  
 DF\_1\_NODEFLIB = 0x00000800 ,  
 DF\_1\_NODUMP = 0x00001000 , DF\_1\_CONFALT = 0x00002000 , DF\_1\_ENDFILTEE = 0x00004000 ,  
 DF\_1\_DISPRELDNE = 0x00008000 ,  
 DF\_1\_DISPRELPND = 0x00010000 }

*State flags selectable in the Elf32\_Dyn.d\_un.d\_val / Elf64\_Dyn.d\_un.d\_val element of the DT\_FLAGS\_1 entry in the dynamic section.*

- enum Elf\_DTF1sc

*Flags for the feature selection in DT\_FEATURE\_1.*

- enum Elf\_DF\_P1sc { DF\_P1\_LAZYLOAD = 0x00000001 , DF\_P1\_GROUPPERM = 0x00000002 }

*Flags in the DT\_POSFLAG\_1 entry effecting only the next DT\_\* entry.*

- enum `Elf_R_386_s` {  
`R_386_NONE` = 0 , `R_386_32` = 1 , `R_386_PC32` = 2 , `R_386_GOT32` = 3 ,  
`R_386_PLT32` = 4 , `R_386_COPY` = 5 , `R_386_GLOB_DAT` = 6 , `R_386_JMP_SLOT` = 7 ,  
`R_386_RELATIVE` = 8 , `R_386_GOTOFF` = 9 , `R_386_GOTPC` = 10 , `R_386_32PLT` = 11 ,  
`R_386_TLS_TPOFF` = 14 , `R_386_TLS_IE` = 15 , `R_386_TLS_GOTIE` = 16 , `R_386_TLS_LE` = 17 ,  
`R_386_TLS_GD` = 18 , `R_386_TLS_LDM` = 19 , `R_386_16` = 20 , `R_386_PC16` = 21 ,  
`R_386_8` = 22 , `R_386_PC8` = 23 , `R_386_TLS_GD_32` = 24 , `R_386_TLS_GD_PUSH` = 25 ,  
`R_386_TLS_GD_CALL` = 26 , `R_386_TLS_GD_POP` = 27 , `R_386_TLS_LDM_32` = 28 , `R_386_TLS_LDM_PUSH`  
= 29 ,  
`R_386_TLS_LDM_CALL` = 30 , `R_386_TLS_LDM_POP` = 31 , `R_386_TLS_LDO_32` = 32 , `R_386_TLS_IE_32`  
= 33 ,  
`R_386_TLS_LE_32` = 34 , `R_386_TLS_DTPMOD32` = 35 , `R_386_TLS_DTPOFF32` = 36 , `R_386_TLS_TPOFF32`  
= 37 ,  
`R_386_NUM` = 38 }  
*Relocation types (processor specific).*
- enum `Elf_EF_ARM_s` { }  
*ARM specific declarations.*
- enum `Elf_STT_ARM_s`  
*Additional symbol types for Thumb.*
- enum `Elf_SHF_s_ARM` { `SHF_ARM_ENTRYSECT` = 0x10000000 , `SHF_ARM_COMDEF` = 0x80000000 }  
*ARM-specific values for `Elf32_Shdr.sh_flags` / `Elf64_Shdr.sh_flags`.*
- enum `Elf_ARM_SBs` { `PF_ARM_SB` = 0x10000000 }  
*ARM-specific program header flags.*
- enum `Elf_R_ARM_s` {  
`R_ARM_NONE` = 0 , `R_ARM_PC24` = 1 , `R_ARM_ABS32` = 2 , `R_ARM_REL32` = 3 ,  
`R_ARM_PC13` = 4 , `R_ARM_ABS16` = 5 , `R_ARM_ABS12` = 6 , `R_ARM_THM_ABS5` = 7 ,  
`R_ARM_ABS8` = 8 , `R_ARM_SBREL32` = 9 , `R_ARM_THM_PC22` = 10 , `R_ARM_THM_PC8` = 11 ,  
`R_ARM_AMP_VCALL9` = 12 , `R_ARM_SWI24` = 13 , `R_ARM_THM_SWI8` = 14 , `R_ARM_XPC25` = 15 ,  
`R_ARM_THM_XPC22` = 16 , `R_ARM_COPY` = 20 , `R_ARM_GLOB_DAT` = 21 , `R_ARM_JUMP_SLOT` = 22 ,  
`R_ARM_RELATIVE` = 23 , `R_ARM_GOTOFF` = 24 , `R_ARM_GOTPC` = 25 , `R_ARM_GOT32` = 26 ,  
`R_ARM_PLT32` = 27 , `R_ARM_ALU_PCREL_7_0` = 32 , `R_ARM_ALU_PCREL_15_8` = 33 , `R_ARM_↵`  
`ALU_PCREL_23_15` = 34 ,  
`R_ARM_LDR_SBREL_11_0` = 35 , `R_ARM_ALU_SBREL_19_12` = 36 , `R_ARM_ALU_SBREL_27_20` =  
37 , `R_ARM_GNU_VTENTRY` = 100 ,  
`R_ARM_GNU_VTINHERIT` = 101 , `R_ARM_THM_PC11` = 102 , `R_ARM_THM_PC9` = 103 , `R_ARM_↵`  
`RXPC25` = 249 ,  
`R_ARM_RSBREL32` = 250 , `R_ARM_THM_RPC22` = 251 , `R_ARM_RREL32` = 252 , `R_ARM_RABS22` =  
253 ,  
`R_ARM_RPC24` = 254 , `R_ARM_RBASE` = 255 , `R_ARM_NUM` = 256 }  
*ARM relocations.*
- enum `Elf_R_AARCH64_s` { `R_AARCH64_NONE` = 0 , `R_AARCH64_RELATIVE` = 1027 }  
*AARCH64 relocations.*
- enum `Elf_R_X86_64_s` {  
`R_X86_64_NONE` = 0 , `R_X86_64_64` = 1 , `R_X86_64_PC32` = 2 , `R_X86_64_GOT32` = 3 ,  
`R_X86_64_PLT32` = 4 , `R_X86_64_COPY` = 5 , `R_X86_64_GLOB_DAT` = 6 , `R_X86_64_JUMP_SLOT` = 7 ,  
`R_X86_64_RELATIVE` = 8 , `R_X86_64_GOTPCREL` = 9 , `R_X86_64_32` = 10 , `R_X86_64_32S` = 11 ,  
`R_X86_64_16` = 12 , `R_X86_64_PC16` = 13 , `R_X86_64_8` = 14 , `R_X86_64_PC8` = 15 ,  
`R_X86_64_DTPMOD64` = 16 , `R_X86_64_DTPOFF64` = 17 , `R_X86_64_TPOFF64` = 18 , `R_X86_64_TLSD`  
= 19 ,  
`R_X86_64_TLSD` = 20 , `R_X86_64_DTPOFF32` = 21 , `R_X86_64_GOTTPOFF` = 22 , `R_X86_64_TPOFF32`  
= 23 ,  
`R_X86_64_NUM` = 24 }  
*AMD x86-64 relocations.*
- enum `Elf_STNs`  
*Symbol Table Entry.*

- enum `Elf_STBs` {  
`STB_LOCAL` = 0 , `STB_GLOBAL` = 1 , `STB_WEAK` = 2 , `STB_LOOS` = 10 ,  
`STB_HIOS` = 12 , `STB_LOPROC` = 13 , `STB_HIPROC` = 15 }  
*Symbol Binding.*
- enum `Elf_STTs` {  
`STT_NOTYPE` = 0 , `STT_OBJECT` = 1 , `STT_FUNC` = 2 , `STT_SECTION` = 3 ,  
`STT_FILE` = 4 , `STT_LOOS` = 10 , `STT_HIOS` = 12 , `STT_LOPROC` = 13 ,  
`STT_HIPROC` = 15 }  
*Symbol Types.*
- enum `Elf_ATs` {  
`AT_NULL` = 0 , `AT_IGNORE` = 1 , `AT_EXECFD` = 2 , `AT_PHDR` = 3 ,  
`AT_PHENT` = 4 , `AT_PHNUM` = 5 , `AT_PAGESZ` = 6 , `AT_BASE` = 7 ,  
`AT_FLAGS` = 8 , `AT_ENTRY` = 9 , `AT_NOTELF` = 10 , `AT_UID` = 11 ,  
`AT_EUID` = 12 , `AT_GID` = 13 , `AT_EGID` = 14 , `AT_L4_AUX` = 0xf0 ,  
`AT_L4_ENV` = 0xf1 }  
*Legal values for `Elf32_Auxv.atype` / `Elf64_Auxv.atype`.*

## 17.639.1 Detailed Description

ELF definition.

Date

08/18/2000

Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de) Alexander Warg [aw11@os.inf.tu-dresden.de](mailto:aw11@os.inf.tu-dresden.de)

Many structs from "Executable and Linkable Format (ELF)", Portable Formats Specification, Version 1.1 and "↔ System V Application Binary Interface - DRAFT - April 29, 1998" The Santa Cruz Operation, Inc. (see [http↔://www.sco.com/developer/gabi/contents.html](http://www.sco.com/developer/gabi/contents.html))

Definition in file [elf.h](#).

## 17.640 elf.h

[Go to the documentation of this file.](#)

```

00001
00019 /*
00020  * (c) 2008-2009 Author(s)
00021  *      economic rights: Technische Universität Dresden (Germany)
00022  * License: see LICENSE.spdx (in this directory or the directories above)
00023  */
00024
00025 /* (c) 2003-2006 Technische Universitaet Dresden
00026  * License: see LICENSE.spdx (in this directory or the directories above) */
00027
00028 #pragma once
00029
00030 #include <14/sys/l4int.h>
00031
00037
00042 typedef l4_uint32_t      Elf32_Addr;
00043 typedef l4_uint32_t      Elf32_Off;
00044 typedef l4_uint16_t      Elf32_Half;
00045 typedef l4_uint32_t      Elf32_Word;
00046 typedef l4_int32_t       Elf32_Sword;
```

```

00047 typedef l4_uint64_t      Elf64_Addr;
00048 typedef l4_uint64_t      Elf64_Off;
00049 typedef l4_uint16_t      Elf64_Half;
00050 typedef l4_uint32_t      Elf64_Word;
00051 typedef l4_int32_t       Elf64_Sword;
00052 typedef l4_uint64_t      Elf64_Xword;
00053 typedef l4_int64_t       Elf64_Sxword;
00055
00060 #if L4_MWORD_BITS == 64
00061 # define ElfW(type)      _ElfW(Elf, 64, type)
00062 #else
00063 # define ElfW(type)      _ElfW(Elf, 32, type)
00064 #endif
00065 #define _ElfW(e,w,t)      __ElfW(e, w, _##t)
00066 #define __ElfW(e,w,t)     e##w##t
00067
00068 #if defined(ARCH_x86)
00069 # define L4_ARCH_EI_DATA      ELFDATA2LSB
00070 # define L4_ARCH_E_MACHINE    EM_386
00071 # define L4_ARCH_EI_CLASS     ELFCLASS32
00072 #elif defined(ARCH_amd64)
00073 # define L4_ARCH_EI_DATA      ELFDATA2LSB
00074 # define L4_ARCH_E_MACHINE    EM_X86_64
00075 # define L4_ARCH_EI_CLASS     ELFCLASS64
00076 #elif defined(ARCH_arm)
00077 # define L4_ARCH_EI_DATA      ELFDATA2LSB
00078 # define L4_ARCH_E_MACHINE    EM_ARM
00079 # define L4_ARCH_EI_CLASS     ELFCLASS32
00080 #elif defined(ARCH_arm64)
00081 # define L4_ARCH_EI_DATA      ELFDATA2LSB
00082 # define L4_ARCH_E_MACHINE    EM_AARCH64
00083 # define L4_ARCH_EI_CLASS     ELFCLASS64
00084 #elif defined(ARCH_ppc32)
00085 # define L4_ARCH_EI_DATA      ELFDATA2MSB
00086 # define L4_ARCH_E_MACHINE    EM_PPC
00087 # define L4_ARCH_EI_CLASS     ELFCLASS32
00088 #elif defined(ARCH_sparc)
00089 # define L4_ARCH_EI_DATA      ELFDATA2MSB
00090 # define L4_ARCH_E_MACHINE    EM_SPARC
00091 # define L4_ARCH_EI_CLASS     ELFCLASS32
00092 #elif defined(ARCH_mips)
00093 # define L4_ARCH_EI_DATA      ELFDATA2LSB
00094 # define L4_ARCH_E_MACHINE    EM_MIPS
00095 # ifdef __mips64
00096 #   define L4_ARCH_EI_CLASS     ELFCLASS64
00097 # else
00098 #   define L4_ARCH_EI_CLASS     ELFCLASS32
00099 # endif
00100 #elif defined(ARCH_riscv)
00101 # define L4_ARCH_EI_DATA      ELFDATA2LSB
00102 # define L4_ARCH_E_MACHINE    EM_RISCV
00103 # if __riscv_xlen == 64
00104 #   define L4_ARCH_EI_CLASS     ELFCLASS64
00105 # else
00106 #   define L4_ARCH_EI_CLASS     ELFCLASS32
00107 # endif
00108 #else
00109 # warning elf.h: Unsupported build architecture!
00110 #endif
00111
00112
00114
00116
00117 enum
00118 {
00119     EI_NIDENT                = 16,
00120 };
00121
00125 typedef struct
00126 {
00127     unsigned char e_ident[EI_NIDENT];
00128     Elf32_Half    e_type;
00129     Elf32_Half    e_machine;
00130     Elf32_Word    e_version;
00131     Elf32_Addr    e_entry;
00132     Elf32_Off     e_phoff;
00133     Elf32_Off     e_shoff;
00134     Elf32_Word    e_flags;
00135     Elf32_Half    e_ehsize;
00136     Elf32_Half    e_phentsize;
00137     Elf32_Half    e_phnum;
00138     Elf32_Half    e_shentsize;
00139     Elf32_Half    e_shnum;
00140     Elf32_Half    e_shstrndx;
00141 } Elf32_Ehdr;
00142
00146 typedef struct

```

```

00147 {
00148     unsigned char e_ident[EI_NIDENT];
00149     Elf64_Half    e_type;
00150     Elf64_Half    e_machine;
00151     Elf64_Word    e_version;
00152     Elf64_Addr    e_entry;
00153     Elf64_Off     e_phoff;
00154     Elf64_Off     e_shoff;
00155     Elf64_Word    e_flags;
00156     Elf64_Half    e_ehsize;
00157     Elf64_Half    e_phentsize;
00158     Elf64_Half    e_phnum;
00159     Elf64_Half    e_shentsize;
00160     Elf64_Half    e_shnum;
00161     Elf64_Half    e_shstrndx;
00162 } Elf64_Ehdr;
00163
00164 enum Elf_ETs
00165 {
00170     ET_NONE          = 0,
00171     ET_REL           = 1,
00172     ET_EXEC          = 2,
00173     ET_DYN           = 3,
00174     ET_CORE          = 4,
00175     ET_LOPROC        = 0xff00,
00176     ET_HIPROC        = 0xffff,
00177 };
00178
00183 enum Elf_EMs
00184 {
00185     EM_NONE          = 0,
00186     EM_M32           = 1,
00187     EM_SPARC         = 2,
00188     EM_386           = 3,
00189     EM_68K           = 4,
00190     EM_88K           = 5,
00191     EM_860           = 7,
00192     EM_MIPS          = 8,
00193     EM_MIPS_RS4_BE   = 10,
00194     EM_SPARC64       = 11,
00195     EM_PARISC        = 15,
00196     EM_VPP500        = 17,
00197     EM_SPARC32PLUS   = 18,
00198     EM_960           = 19,
00199     EM_PPC           = 20,
00200     EM_V800          = 36,
00201     EM_FR20          = 37,
00202     EM_RH32          = 38,
00203     EM_RCE           = 39,
00204     EM_ARM           = 40,
00205     EM_ALPHA         = 41,
00206     EM_SH            = 42,
00207     EM_SPARCV9       = 43,
00208     EM_TRICORE       = 44,
00209     EM_ARC           = 45,
00210     EM_H8_300        = 46,
00211     EM_H8_300H       = 47,
00212     EM_H8S           = 48,
00213     EM_H8_500        = 49,
00214     EM_IA_64         = 50,
00215     EM_MIPS_X        = 51,
00216     EM_COLDFIRE      = 52,
00217     EM_68HC12        = 53,
00218     EM_X86_64        = 62,
00219     EM_PDSP          = 63,
00220     EM_FX66          = 66,
00221     EM_ST9PLUS       = 67,
00222     EM_ST7           = 68,
00223     EM_68HC16        = 69,
00224     EM_68HC11        = 70,
00225     EM_68HC08        = 71,
00226     EM_68HC05        = 72,
00227     EM_SVX           = 73,
00228     EM_ST19          = 74,
00229     EM_VAX           = 75,
00230     EM_CRIS          = 76,
00231     EM_JAVELIN       = 77,
00232     EM_FIREPATH      = 78,
00233     EM_ZSP           = 79,
00234     EM_MMIX          = 80,
00235     EM_HUANY         = 81,
00236     EM_PRISM         = 82,
00237     EM_AVR           = 83,
00238     EM_FR30          = 84,
00239     EM_D10V          = 85,
00240     EM_D30V          = 86,
00241     EM_V850          = 87,

```

```

00242     EM_M32R                = 88,
00243     EM_MN10300             = 89,
00244     EM_MN10200            = 90,
00245     EM_PJ                  = 91,
00246     EM_OPENRISC           = 92,
00247     EM_ARC_A5             = 93,
00248     EM_XTENSA              = 94,
00249     EM_ALTERA_NIOS2        = 113,
00250     EM_AARCH64             = 183,
00251     EM_TILEPRO             = 188,
00252     EM_MICROBLAZE          = 189,
00253     EM_TILEGX              = 191,
00254     EM_RISCV               = 243,
00255     EM_NUM                 = 244,
00256 };
00257
00258 #if 0
00259 #define EM_ALPHA            0x9026 /* interim value used by Linux until the
00260                                   committee comes up with a final number */
00261 #define EM_S390             0xA390 /* interim value used for IBM S390 */
00262 #endif
00263
00266 enum Elf_EVs
00267 {
00268     EV_NONE                = 0,
00269     EV_CURRENT              = 1,
00270 };
00271
00274 enum Elf_EIs
00275 {
00276     EI_MAG0                = 0,
00277     EI_MAG1                = 1,
00278     EI_MAG2                = 2,
00279     EI_MAG3                = 3,
00280     EI_CLASS               = 4,
00281     EI_DATA                = 5,
00282     EI_VERSION             = 6,
00283     EI_OSABI               = 7,
00284     EI_ABIVERSION          = 8,
00285     EI_PAD                 = 9,
00286 };
00287
00289 enum Elf_MAGs
00290 {
00291     ELFMAG0                = 0x7f,
00292     ELFMAG1                = 'E',
00293     ELFMAG2                = 'L',
00294     ELFMAG3                = 'F',
00295 };
00296
00298 enum Elf_CLASSES
00299 {
00300     ELFCLASSNONE           = 0,
00301     ELFCLASS32             = 1,
00302     ELFCLASS64             = 2,
00303     ELFCLASSNUM            = 3,
00304 };
00305
00307 enum Elf_DATAs
00308 {
00309     ELFDATANONE            = 0,
00310     ELFDATA2LSB            = 1,
00311     ELFDATA2MSB            = 2,
00312     ELFDATANUM             = 3,
00313 };
00314
00316 enum Elf_OSABIs
00317 {
00318     ELFOSABI_NONE          = 0,
00319     ELFOSABI_SYSV           = 0,
00320     ELFOSABI_HPUX          = 1,
00321     ELFOSABI_NETBSD        = 2,
00322     ELFOSABI_LINUX         = 3,
00323     ELFOSABI_SOLARIS       = 6,
00324     ELFOSABI_AIX           = 7,
00325     ELFOSABI_IRIX          = 8,
00326     ELFOSABI_FREEBSD       = 9,
00327     ELFOSABI_TRU64         = 10,
00328     ELFOSABI_MODESTO       = 11,
00329     ELFOSABI_OPENBSD       = 12,
00330     ELFOSABI_ARM           = 97,
00331     ELFOSABI_STANDALONE    = 255,
00332 };
00333
00335 enum Elf_SHNs
00336 {
00337     SHN_UNDEF              = 0,

```

```

00338 SHN_LORESERVE      = 0xff00,
00339 SHN_LOPROC          = 0xff00,
00340 SHN_HIPROC          = 0xff1f,
00341 SHN_ABS              = 0xffff1,
00342 SHN_COMMON           = 0xffff2,
00343 SHN_HIRESERVE       = 0xffff,
00344 };
00345
00347 typedef struct
00348 {
00349     Elf32_Word      sh_name;
00350     Elf32_Word      sh_type;
00351     Elf32_Word      sh_flags;
00352     Elf32_Addr      sh_addr;
00353     Elf32_Off       sh_offset;
00354     Elf32_Word      sh_size;
00355     Elf32_Word      sh_link;
00356     Elf32_Word      sh_info;
00357     Elf32_Word      sh_addralign;
00358     Elf32_Word      sh_entsize;
00359 } Elf32_Shdr;
00360
00362 typedef struct
00363 {
00364     Elf64_Word      sh_name;
00365     Elf64_Word      sh_type;
00366     Elf64_Xword     sh_flags;
00367     Elf64_Addr      sh_addr;
00368     Elf64_Off       sh_offset;
00369     Elf64_Xword     sh_size;
00370     Elf64_Word      sh_link;
00371     Elf64_Word      sh_info;
00372     Elf64_Xword     sh_addralign;
00373     Elf64_Xword     sh_entsize;
00374 } Elf64_Shdr;
00375
00377 enum Elf_SHTs
00378 {
00379     SHT_NULL          = 0,
00380     SHT_PROGBITS      = 1,
00381     SHT_SYMTAB        = 2,
00382     SHT_STRTAB        = 3,
00383     SHT_RELA          = 4,
00384     SHT_HASH          = 5,
00385     SHT_DYNAMIC        = 6,
00386     SHT_NOTE          = 7,
00387     SHT_NOBITS        = 8,
00388     SHT_REL           = 9,
00389     SHT_SHLIB         = 10,
00390     SHT_DYNSYM        = 11,
00391     SHT_INIT_ARRAY    = 14,
00392     SHT_FINI_ARRAY    = 15,
00393     SHT_PREINIT_ARRAY = 16,
00394     SHT_GROUP          = 17,
00395     SHT_SYMTAB_SHNDX  = 18,
00396     SHT_NUM            = 19,
00397     SHT_LOOS          = 0x60000000,
00398     SHT_HIOS          = 0x6fffffff,
00399     SHT_LOPROC        = 0x70000000,
00400     SHT_HIPROC        = 0x7fffffff,
00401     SHT_LOUSER        = 0x80000000,
00402     SHT_HIUSER        = 0xffffffff,
00403 };
00404
00406 enum Elf_SHFs
00407 {
00408     SHF_WRITE          = 0x1,
00409     SHF_ALLOC          = 0x2,
00410     SHF_EXECINSTR      = 0x4,
00411     SHF_MERGE          = 0x10,
00412     SHF_STRINGS        = 0x20,
00413     SHF_INFO_LINK      = 0x40,
00414     SHF_LINK_ORDER     = 0x80,
00415     SHF_OS_NONCONFORMING = 0x100,
00417     SHF_GROUP          = 0x200,
00418     SHF_TLS            = 0x400,
00419     SHF_MASKOS         = 0x0ff00000,
00420     SHF_MASKPROC       = 0xf0000000,
00421 };
00422
00423
00425 typedef struct
00426 {
00427     Elf32_Word      p_type;
00428     Elf32_Off       p_offset;
00429     Elf32_Addr      p_vaddr;
00430     Elf32_Addr      p_paddr;

```



```

00431 Elf32_Word    p_filesz;
00432 Elf32_Word    p_memsz;
00433 Elf32_Word    p_flags;
00434 Elf32_Word    p_align;
00435 } Elf32_Phdr;
00436
00437 typedef struct
00438 {
00439     Elf64_Word    p_type;
00440     Elf64_Word    p_flags;
00441     Elf64_Off     p_offset;
00442     Elf64_Addr    p_vaddr;
00443     Elf64_Addr    p_paddr;
00444     Elf64_Xword   p_filesz;
00445     Elf64_Xword   p_memsz;
00446     Elf64_Xword   p_align;
00447 } Elf64_Phdr;
00448
00449 enum Elf_PTs
00450 {
00451     PT_NULL            = 0,
00452     PT_LOAD            = 1,
00453     PT_DYNAMIC         = 2,
00454     PT_INTERP          = 3,
00455     PT_NOTE            = 4,
00456     PT_SHLIB           = 5,
00457     PT_PHDR            = 6,
00458     PT_TLS             = 7,
00459     PT_NUM             = 8,
00460     PT_LOOS            = 0x60000000,
00461     PT_HIOS            = 0x6fffffff,
00462     PT_LOPROC          = 0x70000000,
00463     PT_HIPROC          = 0x7fffffff,
00464     PT_GNU_EH_FRAME    = PT_LOOS + 0x474e550,
00465     PT_GNU_STACK       = PT_LOOS + 0x474e551,
00466     PT_GNU_RELRO       = PT_LOOS + 0x474e552,
00467     PT_L4_STACK        = PT_LOOS + 0x12,
00468     PT_L4_AUX          = PT_LOOS + 0x14,
00469 };
00470
00471 enum ELF_PFs
00472 {
00473     PF_X               = 0x1,
00474     PF_W               = 0x2,
00475     PF_R               = 0x4,
00476     PF_MASKOS          = 0x0ff00000,
00477     PF_MASKPROC        = 0x7fffffff,
00478 };
00479
00480 enum Elf_NTs_core
00481 {
00482     NT_PRSTATUS        = 1,
00483     NT_FPREGSET        = 2,
00484     NT_PRPSINFO        = 3,
00485     NT_PRXREG          = 4,
00486     NT_TASKSTRUCT      = 4,
00487     NT_PLATFORM        = 5,
00488     NT_AUXV            = 6,
00489     NT_GWINDOWS        = 7,
00490     NT_ASRS            = 8,
00491     NT_PSTATUS         = 10,
00492     NT_PSINFO          = 13,
00493     NT_PRCRED          = 14,
00494     NT_UTSNAME         = 15,
00495     NT_LWPSTATUS       = 16,
00496     NT_LWPSINFO        = 17,
00497     NT_PRFPXREG        = 20,
00498 };
00499
00500 enum Elf_NTs_obj
00501 {
00502     NT_VERSION          = 1,
00503 };
00504
00505 typedef struct
00506 {
00507     Elf32_Sword    d_tag;
00508     union
00509     {
00510         Elf32_Word    d_val;
00511         Elf32_Addr    d_ptr;
00512     } d_un;
00513 } Elf32_Dyn;
00514
00515 typedef struct

```

```

00525 {
00526     Elf64_Sxword d_tag;
00527     union
00528     {
00529         Elf64_Xword d_val;
00530         Elf64_Addr d_ptr;
00531     } d_un;
00532 } Elf64_Dyn;
00533
00535 enum Elf_DTs
00536 {
00537     DT_NULL                = 0,
00538     DT_NEEDED              = 1,
00539     DT_PLTRELSZ            = 2,
00540     DT_PLTGOT              = 3,
00541     DT_HASH                = 4,
00542     DT_STRTAB              = 5,
00543     DT_SYMTAB              = 6,
00544     DT_RELA                = 7,
00545     DT_RELASZ              = 8,
00546     DT_RELAENT             = 9,
00547     DT_STRSZ               = 10,
00548     DT_SYMENT              = 11,
00549     DT_INIT                = 12,
00550     DT_FINI                = 13,
00551     DT_SONAME              = 14,
00552     DT_RPATH               = 15,
00553     DT_SYMBOLIC            = 16,
00554     DT_REL                 = 17,
00555     DT_RELSZ               = 18,
00556     DT_RELENT              = 19,
00557     DT_PTRREL              = 20,
00558     DT_DEBUG               = 21,
00559     DT_TEXTREL             = 22,
00560     DT_JMPREL              = 23,
00561     DT_BIND_NOW            = 24,
00562     DT_INIT_ARRAY          = 25,
00563     DT_FINI_ARRAY          = 26,
00564     DT_INIT_ARRAYSZ        = 27,
00565     DT_FINI_ARRAYSZ        = 28,
00566     DT_RUNPATH              = 29,
00567     DT_FLAGS                = 30,
00568     DT_ENCODING            = 32,
00569     DT_PREINIT_ARRAY       = 32,
00570     DT_PREINIT_ARRAYSZ     = 33,
00571     DT_NUM                  = 34,
00572     DT_LOOS                 = 0x6000000d,
00573     DT_HIOS                 = 0x6ffff000,
00574     DT_LOPROC               = 0x70000000,
00575     DT_HIPROC               = 0x7fffffff,
00576 };
00577
00581 enum Elf_DFs
00582 {
00583     DF_ORIGIN               = 0x00000001,
00584     DF_SYMBOLIC             = 0x00000002,
00585     DF_TEXTREL              = 0x00000004,
00586     DF_BIND_NOW             = 0x00000008,
00587     DF_STATIC_TLS           = 0x00000010,
00588 };
00589
00594 enum Elf_DF_1s
00595 {
00596     DF_1_NOW                 = 0x00000001,
00597     DF_1_GLOBAL              = 0x00000002,
00598     DF_1_GROUP               = 0x00000004,
00599     DF_1_NODELETE           = 0x00000008,
00600     DF_1_LOADFLTR           = 0x00000010,
00601     DF_1_INITFIRST          = 0x00000020,
00602     DF_1_NOOPEN              = 0x00000040,
00603     DF_1_ORIGIN              = 0x00000080,
00604     DF_1_DIRECT              = 0x00000100,
00605     DF_1_TRANS               = 0x00000200,
00606     DF_1_INTERPOSE           = 0x00000400,
00607     DF_1_NODEFLIB           = 0x00000800,
00608     DF_1_NODUMP              = 0x00001000,
00609     DF_1_CONFALT             = 0x00002000,
00610     DF_1_ENDFILTEE           = 0x00004000,
00611     DF_1_DISPRELDNE         = 0x00008000,
00612     DF_1_DISPRELPND         = 0x00010000,
00613 };
00614
00616 enum Elf_DTF_1s
00617 {
00618     DTF_1_PARINIT            = 0x00000001,
00619     DTF_1_CONFEXP           = 0x00000002,
00620 };

```

```

00621
00623 enum Elf_DF_Pls
00624 {
00625     DF_P1_LAZYLOAD          = 0x00000001,
00626     DF_P1_GROUPPERM        = 0x00000002,
00628 };
00629
00631 typedef struct
00632 {
00633     Elf32_Addr              r_offset;
00634     Elf32_Word              r_info;
00635 } Elf32_Rel;
00636
00638 typedef struct
00639 {
00640     Elf32_Addr              r_offset;
00641     Elf32_Word              r_info;
00642     Elf32_Sword             r_addend;
00643 } Elf32_Rela;
00644
00646 typedef struct
00647 {
00648     Elf64_Addr              r_offset;
00649     Elf64_Xword             r_info;
00650 } Elf64_Rel;
00651
00653 typedef struct
00654 {
00655     Elf64_Addr              r_offset;
00656     Elf64_Xword             r_info;
00657     Elf64_Sxword            r_addend;
00658 } Elf64_Rela;
00659
00661 #define ELF32_R_SYM(i)      ((i)>>8)
00663 #define ELF32_R_TYPE(i)     ((unsigned char)(i))
00665 #define ELF32_R_INFO(s,t)   (((s)<<8)+(unsigned char)(t))
00666
00668 #define ELF64_R_SYM(i)      ((i)>>32)
00669
00671 #define ELF64_R_TYPE(i)     ((i)&0xffffffffL)
00672
00674 #define ELF64_R_INFO(s,t)   (((s)<<32)+(t)&0xffffffffL)
00675
00677 enum Elf_R_386_s
00678 {
00679     R_386_NONE              = 0,
00680     R_386_32                = 1,
00681     R_386_PC32              = 2,
00682     R_386_GOT32             = 3,
00683     R_386_PLT32             = 4,
00684     R_386_COPY              = 5,
00685     R_386_GLOB_DAT          = 6,
00686     R_386_JMP_SLOT          = 7,
00687     R_386_RELATIVE          = 8,
00688     R_386_GOTOFF            = 9,
00689     R_386_GOTPC              = 10,
00690     R_386_32PLT             = 11,
00691     R_386_TLS_TPOFF         = 14,
00692     R_386_TLS_IE             = 15,
00694     R_386_TLS_GOTIE         = 16,
00695     R_386_TLS_LE             = 17,
00696     R_386_TLS_GD             = 18,
00698     R_386_TLS_LDM           = 19,
00700     R_386_16                = 20,
00701     R_386_PC16              = 21,
00702     R_386_8                 = 22,
00703     R_386_PC8               = 23,
00704     R_386_TLS_GD_32         = 24,
00706     R_386_TLS_GD_PUSH       = 25,
00707     R_386_TLS_GD_CALL       = 26,
00709     R_386_TLS_GD_POP        = 27,
00710     R_386_TLS_LDM_32        = 28,
00712     R_386_TLS_LDM_PUSH      = 29,
00713     R_386_TLS_LDM_CALL      = 30,
00715     R_386_TLS_LDM_POP       = 31,
00716     R_386_TLS_LDO_32        = 32,
00717     R_386_TLS_IE_32         = 33,
00719     R_386_TLS_LE_32         = 34,
00721     R_386_TLS_DTPMOD32      = 35,
00722     R_386_TLS_DTPOFF32      = 36,
00723     R_386_TLS_TPOFF32       = 37,
00724     R_386_NUM                = 38,
00725 };
00726
00728
00730 enum Elf_EF_ARM_s
00731 {

```

```

00732 EF_ARM_RELEXEC          = 0x01,
00733 EF_ARM_HASENTRY          = 0x02,
00734 EF_ARM_INTERWORK        = 0x04,
00735 EF_ARM_APCS_26           = 0x08,
00736 EF_ARM_APCS_FLOAT        = 0x10,
00737 EF_ARM_PIC               = 0x20,
00738 EF_ARM_ALIGN8            = 0x40,
00739 EF_ARM_NEW_ABI           = 0x80,
00740 EF_ARM_OLD_ABI           = 0x100,
00741
00742 /* Other constants defined in the ARM ELF spec. version B-01. */
00743 /* NB. These conflict with values defined above. */
00744 EF_ARM_SYMSARESORTED     = 0x04,
00745 EF_ARM_DYNSYMSUSESEGIDX  = 0x08,
00746 EF_ARM_MAPSYMSFIRST      = 0x10,
00747 EF_ARM_EABIMASK          = 0xFF000000,
00748
00749 #define EF_ARM_EABI_VERSION(flags) ((flags) & EF_ARM_EABIMASK)
00750 EF_ARM_EABI_UNKNOWN       = 0x00000000,
00751 EF_ARM_EABI_VER1          = 0x01000000,
00752 EF_ARM_EABI_VER2          = 0x02000000,
00753 };
00754
00756 enum Elf_STT_ARM_s
00757 {
00758     STT_ARM_TFUNC          = 0xd,
00759 };
00760
00762 enum Elf_SHF_s_ARM
00763 {
00764     SHF_ARM_ENTRYSECT      = 0x10000000,
00765     SHF_ARM_COMDEF         = 0x80000000,
00766 };
00767
00770 enum Elf_ARM_SBs
00771 {
00772     PF_ARM_SB              = 0x10000000,
00773 };
00774
00775
00777 enum Elf_R_ARM_s
00778 {
00779     R_ARM_NONE              = 0,
00780     R_ARM_PC24              = 1,
00781     R_ARM_ABS32             = 2,
00782     R_ARM_REL32             = 3,
00783     R_ARM_PC13              = 4,
00784     R_ARM_ABS16             = 5,
00785     R_ARM_ABS12             = 6,
00786     R_ARM_THM_ABS5          = 7,
00787     R_ARM_ABS8              = 8,
00788     R_ARM_SBREL32           = 9,
00789     R_ARM_THM_PC22          = 10,
00790     R_ARM_THM_PC8           = 11,
00791     R_ARM AMP_VCALL9         = 12,
00792     R_ARM_SWI24             = 13,
00793     R_ARM_THM_SWI8          = 14,
00794     R_ARM_XPC25             = 15,
00795     R_ARM_THM_XPC22         = 16,
00796     R_ARM_COPY              = 20,
00797     R_ARM_GLOB_DAT          = 21,
00798     R_ARM_JUMP_SLOT         = 22,
00799     R_ARM_RELATIVE          = 23,
00800     R_ARM_GOTOFF            = 24,
00801     R_ARM_GOTPC             = 25,
00802     R_ARM_GOT32             = 26,
00803     R_ARM_PLT32             = 27,
00804     R_ARM_ALU_PCREL_7_0     = 32,
00805     R_ARM_ALU_PCREL_15_8    = 33,
00806     R_ARM_ALU_PCREL_23_15   = 34,
00807     R_ARM_LDR_SBREL_11_0    = 35,
00808     R_ARM_ALU_SBREL_19_12   = 36,
00809     R_ARM_ALU_SBREL_27_20   = 37,
00810     R_ARM_GNU_VTENTRY       = 100,
00811     R_ARM_GNU_VTINHERIT     = 101,
00812     R_ARM_THM_PC11          = 102,
00813     R_ARM_THM_PC9           = 103,
00814     R_ARM_RXPC25            = 249,
00815     R_ARM_RSBREL32          = 250,
00816     R_ARM_THM_RPC22         = 251,
00817     R_ARM_RREL32            = 252,
00818     R_ARM_RABS22            = 253,
00819     R_ARM_RPC24             = 254,
00820     R_ARM_RBASE             = 255,
00821     R_ARM_NUM               = 256,
00822 };
00823
00825 enum Elf_R_AARCH64_s

```

```

00826 {
00827     R_AARCH64_NONE          = 0,
00828     R_AARCH64_RELATIVE      = 1027,
00829 };
00830
00832 enum Elf_R_X86_64_s
00833 {
00834     R_X86_64_NONE          = 0,
00835     R_X86_64_64           = 1,
00836     R_X86_64_PC32          = 2,
00837     R_X86_64_GOT32         = 3,
00838     R_X86_64_PLT32         = 4,
00839     R_X86_64_COPY          = 5,
00840     R_X86_64_GLOB_DAT      = 6,
00841     R_X86_64_JUMP_SLOT     = 7,
00842     R_X86_64_RELATIVE      = 8,
00843     R_X86_64_GOTPCREL      = 9,
00844     R_X86_64_32           = 10,
00845     R_X86_64_32S          = 11,
00846     R_X86_64_16           = 12,
00847     R_X86_64_PC16         = 13,
00848     R_X86_64_8            = 14,
00849     R_X86_64_PC8          = 15,
00850     R_X86_64_DTPMOD64      = 16,
00851     R_X86_64_DTPOFF64      = 17,
00852     R_X86_64_TPOFF64       = 18,
00853     R_X86_64_TLSGD         = 19,
00855     R_X86_64_TLSLD         = 20,
00857     R_X86_64_DTPOFF32      = 21,
00858     R_X86_64_GOTTPOFF      = 22,
00860     R_X86_64_TPOFF32       = 23,
00861     R_X86_64_NUM           = 24,
00862 };
00863
00865 enum Elf_STNs
00866 {
00867     STN_UNDEF              = 0,
00868 };
00869
00871 typedef struct
00872 {
00873     Elf32_Word              st_name;
00874     Elf32_Addr              st_value;
00875     Elf32_Word              st_size;
00876     unsigned char           st_info;
00877     unsigned char           st_other;
00878     Elf32_Half              st_shndx;
00879 } Elf32_Sym;
00880
00882 typedef struct
00883 {
00884     Elf64_Word              st_name;
00885     unsigned char           st_info;
00886     unsigned char           st_other;
00887     Elf64_Half              st_shndx;
00888     Elf64_Addr              st_value;
00889     Elf64_Xword             st_size;
00890 } Elf64_Sym;
00891
00893 #define ELF32_ST_BIND(i)    ((i)>4)
00894
00896 #define ELF32_ST_TYPE(i)    ((i)&0xf)
00897
00899 #define ELF32_ST_INFO(b,t)  (((b)<<4)+((t)&0xf))
00900
00902 #define ELF64_ST_BIND(i)    ((i)>4)
00903
00905 #define ELF64_ST_TYPE(i)    ((i)&0xf)
00906
00908 #define ELF64_ST_INFO(b,t)  (((b)<<4)+((t)&0xf))
00909
00912 enum Elf_STBs
00913 {
00914     STB_LOCAL              = 0,
00915     STB_GLOBAL              = 1,
00916     STB_WEAK                = 2,
00917     STB_LOOS                = 10,
00918     STB_HIOS                = 12,
00919     STB_LOPROC              = 13,
00920     STB_HIPROC              = 15,
00921 };
00922
00925 enum Elf_STTs
00926 {
00927     STT_NOTYPE              = 0,
00928     STT_OBJECT              = 1,
00929     STT_FUNC                = 2,

```

```

00930     STT_SECTION    = 3,
00931     STT_FILE       = 4,
00932     STT_LOOS       = 10,
00933     STT_HIOS       = 12,
00934     STT_LOPROC     = 13,
00935     STT_HIPROC     = 15,
00936 };
00937
00939 enum Elf_ATs
00940 {
00941     AT_NULL         = 0,
00942     AT_IGNORE       = 1,
00943     AT_EXECD       = 2,
00944     AT_PHDR        = 3,
00945     AT_PHENT        = 4,
00946     AT_PHNUM        = 5,
00947     AT_PAGESZ       = 6,
00948     AT_BASE         = 7,
00949     AT_FLAGS        = 8,
00950     AT_ENTRY        = 9,
00951     AT_NOTELF       = 10,
00952     AT_UID          = 11,
00953     AT_EUID         = 12,
00954     AT_GID          = 13,
00955     AT_EGID         = 14,
00956
00957     AT_L4_AUX       = 0xf0,
00958     AT_L4_ENV       = 0xf1,
00959 };
00960
00962 typedef struct Elf32_Auxv
00963 {
00964     Elf32_Word atype;
00965     Elf32_Word avalue;
00966 } Elf32_Auxv;
00967
00969 typedef struct Elf64_Auxv
00970 {
00971     Elf64_Word atype;
00972     Elf64_Word avalue;
00973 } Elf64_Auxv;
00974
00975 typedef struct
00976 {
00977     Elf32_Word n_namesz;
00978     Elf32_Word n_descsz;
00979     Elf32_Word n_type;
00980 } Elf32_Nhdr;
00981
00982 typedef struct
00983 {
00984     Elf64_Word n_namesz;
00985     Elf64_Word n_descsz;
00986     Elf64_Word n_type;
00987 } Elf64_Nhdr;
00988
00996 static inline int l4util_elf_check_magic(ElfW(Ehdr) const *hdr);
00997
01005 static inline int l4util_elf_check_arch(ElfW(Ehdr) const *hdr);
01006
01013 static inline ElfW(Phdr) *l4util_elf_phdr(ElfW(Ehdr) const *hdr);
01014
01015
01016 /* Implementations */
01017
01018 static inline
01019 int l4util_elf_check_magic(ElfW(Ehdr) const *hdr)
01020 {
01021     return  hdr->e_ident[EI_MAG0] == ELFMAG0
01022         && hdr->e_ident[EI_MAG1] == ELFMAG1
01023         && hdr->e_ident[EI_MAG2] == ELFMAG2
01024         && hdr->e_ident[EI_MAG3] == ELFMAG3;
01025 }
01026
01027 static inline
01028 int l4util_elf_check_arch(ElfW(Ehdr) const *hdr)
01029 {
01030     return  hdr->e_ident[EI_CLASS] == L4_ARCH_EI_CLASS
01031         && hdr->e_ident[EI_DATA]   == L4_ARCH_EI_DATA
01032         && hdr->e_machine         == L4_ARCH_E_MACHINE;
01033 }
01034
01035 static inline
01036 ElfW(Phdr) *l4util_elf_phdr(ElfW(Ehdr) const *hdr)
01037 {
01038     return (ElfW(Phdr) *) ((char *)hdr + hdr->e_phoff);
01039 }

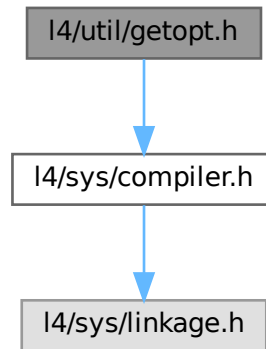
```

## 17.641 l4/util/getopt.h File Reference

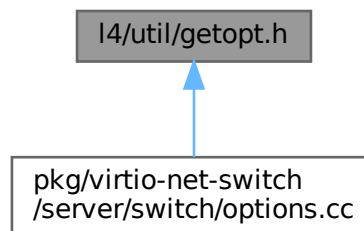
getopt

```
#include <l4/sys/compiler.h>
```

Include dependency graph for getopt.h:



This graph shows which files directly or indirectly include this file:



### 17.641.1 Detailed Description

getopt

Definition in file [getopt.h](#).

## 17.642 getopt.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #ifndef _GETOPT_H
00012 #define _GETOPT_H
00013
00014 #ifndef NULL
00015 #define NULL 0
00016 #endif
00017
00018 #include <l4/sys/compiler.h>
00019
00020 L4_BEGIN_DECLS
00021
00022 /* For communication from `getopt' to the caller.
00023  * When `getopt' finds an option that takes an argument,
00024  * the argument value is returned here.
00025  * Also, when `ordering' is RETURN_IN_ORDER,
00026  * each non-option ARGV-element is returned here.  */
00027
00028 extern char *optarg;
00029
00030 /* Index in ARGV of the next element to be scanned.
00031  * This is used for communication to and from the caller
00032  * and for communication between successive calls to `getopt'.
00033  *
00034  * On entry to `getopt', zero means this is the first call; initialize.
00035  *
00036  * When `getopt' returns -1, this is the index of the first of the
00037  * non-option elements that the caller should itself scan.
00038  *
00039  * Otherwise, `optind' communicates from one call to the next
00040  * how much of ARGV has been scanned so far.  */
00041
00042 extern int optind;
00043
00044 /* Callers store zero here to inhibit the error message `getopt' prints
00045  * for unrecognized options.  */
00046
00047 extern int opterr;
00048
00049 /* Set to an option character which was unrecognized.  */
00050
00051 extern int optopt;
00052
00053 /* Describe the long-named options requested by the application.
00054  * The LONG_OPTIONS argument to getopt_long or getopt_long_only is a vector
00055  * of `struct option' terminated by an element containing a name which is
00056  * zero.
00057  *
00058  * The field `has_arg' is:
00059  * no_argument      (or 0) if the option does not take an argument,
00060  * required_argument (or 1) if the option requires an argument,
00061  * optional_argument (or 2) if the option takes an optional argument.
00062  *
00063  * If the field `flag' is not NULL, it points to a variable that is set
00064  * to the value given in the field `val' when the option is found, but
00065  * left unchanged if the option is not found.
00066  *
00067  * To have a long-named option do something other than set an `int' to
00068  * a compiled-in constant, such as set a value from `optarg', set the
00069  * option's `flag' field to zero and its `val' field to a nonzero
00070  * value (the equivalent single-letter option character, if there is
00071  * one).  For long options that have a zero `flag' field, `getopt'
00072  * returns the contents of the `val' field.  */
00073
00074 struct option
00075 {
00076     const char *name;
00077     /* has_arg can't be an enum because some compilers complain about
00078      * type mismatches in all the code that assumes it is an int.  */
00079     int has_arg;
00080     int *flag;
00081     int val;
00082 };
00083
00084 /* Names for the values of the `has_arg' field of `struct option'.  */
00085

```



```

00086 #define no_argument    0
00087 #define required_argument 1
00088 #define optional_argument 2
00089
00090 L4_CV int getopt (int argc, char *const *argv, const char *shortopts);
00091
00092 L4_CV int getopt_long (int argc, char *const *argv, const char *shortopts,
00093                      const struct option *longopts, int *longind);
00094 L4_CV int getopt_long_only (int argc, char *const *argv,
00095                          const char *shortopts,
00096                          const struct option *longopts, int *longind);
00097
00098 L4_CV int _getopt_internal (int argc, char *const *argv,
00099                          const char *shortopts,
00100                          const struct option *longopts, int *longind,
00101                          int long_only);
00102
00103 L4_END_DECLS
00104
00105 #endif /* _GETOPT_H */

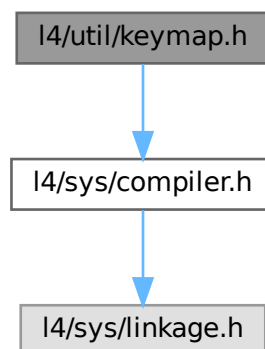
```

## 17.643 l4/util/keymap.h File Reference

Event to ASCII key mapping.

```
#include <l4/sys/compiler.h>
```

Include dependency graph for keymap.h:



### 17.643.1 Detailed Description

Event to ASCII key mapping.

Definition in file [keymap.h](#).

## 17.644 keymap.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010 #ifndef __L4UTIL__KEYMAP_H__
00011 #define __L4UTIL__KEYMAP_H__
00012
00013 #include <l4/sys/compiler.h>
00014
00015 L4_BEGIN_DECLS
00016
00017 int l4util_map_event_to_keymap(unsigned value, unsigned shift);
00018
00019 L4_END_DECLS
00020
00021
00022 #endif /* __L4UTIL__KEYMAP_H__ */

```

## 17.645 l4/sys/kip.h File Reference

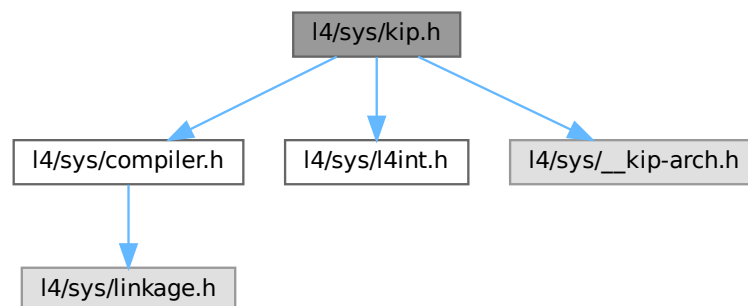
Kernel Info Page access functions.

```

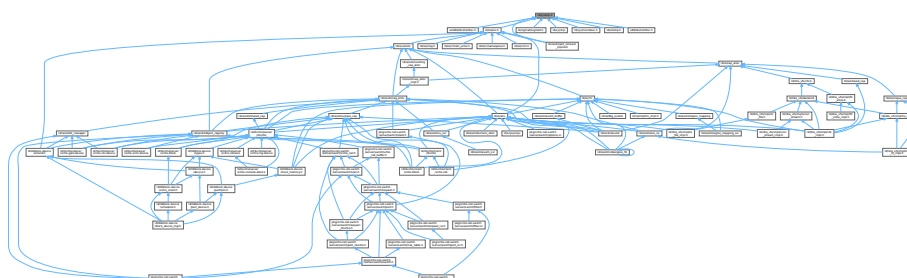
#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>
#include <l4/sys/__kip-arch.h>

```

Include dependency graph for kip.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [l4\\_kernel\\_info\\_t](#)  
*L4 Kernel Interface Page.*

## Macros

- #define **L4\_KERNEL\_INFO\_MAGIC** (0x4BE6344CL) /\* "L4μK" \*/  
*Kernel Info Page identifier ("L4μK").*
- #define **l4\_kip\_for\_each\_feature**(s)  
*Cycle through kernel features given in the KIP.*

## Typedefs

- typedef struct [l4\\_kernel\\_info\\_t](#) [l4\\_kernel\\_info\\_t](#)  
*L4 Kernel Interface Page.*

## Enumerations

- enum { [L4\\_KIP\\_OFFS\\_READ\\_US](#) = 0x900 , [L4\\_KIP\\_OFFS\\_READ\\_NS](#) = 0x980 }

## Functions

- [l4\\_kernel\\_info\\_t](#) const \* [l4\\_kip](#) (void) [L4\\_NOTHROW](#)  
*Get Kernel Info Page.*
- [l4\\_umword\\_t](#) [l4\\_kip\\_version](#) ([l4\\_kernel\\_info\\_t](#) const \*kip) [L4\\_NOTHROW](#)  
*Get the kernel version.*
- const char \* [l4\\_kip\\_version\\_string](#) ([l4\\_kernel\\_info\\_t](#) const \*kip) [L4\\_NOTHROW](#)  
*Get the kernel version string.*
- int [l4\\_kernel\\_info\\_version\\_offset](#) ([l4\\_kernel\\_info\\_t](#) const \*kip) [L4\\_NOTHROW](#)  
*Return offset in bytes of version\_strings relative to the KIP base.*
- [l4\\_cpu\\_time\\_t](#) [l4\\_kip\\_clock](#) ([l4\\_kernel\\_info\\_t](#) const \*kip) [L4\\_NOTHROW](#)  
*Return clock value from the KIP.*
- [l4\\_umword\\_t](#) [l4\\_kip\\_clock\\_lw](#) ([l4\\_kernel\\_info\\_t](#) const \*kip) [L4\\_NOTHROW](#)  
*Return least significant machine word of clock value from the KIP.*
- [l4\\_uint64\\_t](#) [l4\\_kip\\_clock\\_ns](#) ([l4\\_kernel\\_info\\_t](#) const \*kip) [L4\\_NOTHROW](#)  
*Return current clock using the KIP in nanoseconds.*
- int [l4\\_kip\\_kernel\\_has\\_feature](#) ([l4\\_kernel\\_info\\_t](#) const \*kip, char const \*str)  
*Check if kernel supports a feature.*

## 17.645.1 Detailed Description

Kernel Info Page access functions.

Definition in file [kip.h](#).

## 17.645.2 Macro Definition Documentation

### 17.645.2.1 l4\_kip\_for\_each\_feature

```
#define l4_kip_for_each_feature(  
    s)
```

**Value:**

```
for (s += __builtin_strlen(s) + 1; *s; s += __builtin_strlen(s) + 1)
```

Cycle through kernel features given in the KIP.

Cycles through all KIP kernel feature strings. *s* must be a character pointer (`char const *`) initialized with [l4\\_kip\\_version\\_string\(\)](#).

Definition at line 272 of file [kip.h](#).

Referenced by [l4\\_kip\\_kernel\\_has\\_feature\(\)](#).

## 17.645.3 Function Documentation

### 17.645.3.1 l4\_kip\_kernel\_has\_feature()

```
int l4_kip_kernel_has_feature (  
    l4_kernel_info_t const * kip,  
    char const * str) [inline]
```

Check if kernel supports a feature.

**Parameters**

<i>kip</i>	Pointer to the kernel info page (KIP).
<i>str</i>	Feature name to check.

**Returns**

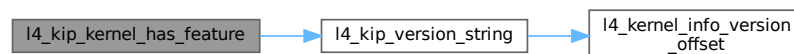
1 if the kernel supports the feature, 0 if not.

Checks the feature field in the KIP for the given string.

Definition at line 286 of file [kip.h](#).

References [l4\\_kip\\_for\\_each\\_feature](#), and [l4\\_kip\\_version\\_string\(\)](#).

Here is the call graph for this function:



## 17.646 kip.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00015 #include <l4/sys/compiler.h>
00016 #include <l4/sys/l4int.h>
00017
00018 #include <l4/sys/__kip-arch.h>
00019
00023 struct l4_kip_platform_info
00024 {
00025     char                                name[16];
00026     l4_uint32_t                        is_mp;
00027     struct l4_kip_platform_info_arch arch;
00028 };
00029
00036 typedef struct l4_kernel_info_t
00037 {
00038     /* offset 0x00 */
00039     l4_uint32_t                        magic;
00042     l4_uint32_t                        version;
00043     l4_uint8_t                        offset_version_strings;
00044     l4_uint8_t                        _fill0[3];
00045     l4_uint8_t                        kip_sys_calls;
00046     l4_uint8_t                        node;
00047     l4_uint8_t                        _fill1[2];
00048
00049     /* offset 0x10 */
00050     l4_uint64_t                        sigma0_ip;
00051     l4_uint64_t                        root_ip;
00052
00053     /* offset 0x20 */
00054     volatile l4_cpu_time_t            _clock_val;
00055     l4_uint64_t                        frequency_cpu;
00056
00057     /* offset 0x30 */
00058     l4_uint64_t                        acpi_rsdp_addr;
00059     l4_uint64_t                        dt_addr;
00060
00061     /* offset 0x40 */
00062     l4_uint64_t                        user_ptr;
00063     l4_uint64_t                        _res0[1];
00064
00065     /* offset 0x50 */
00066     l4_uint32_t                        scheduler_granularity;
00067     l4_uint32_t                        mem_descs;
00068     l4_uint32_t                        mem_descs_num;
00069     l4_uint32_t                        _res1[1];
00070
00071     /* offset 0x60 */
00072     l4_uint64_t                        _res2[2];
00073
00074     /* offset 0x70 */
00075     struct l4_kip_platform_info        platform_info;
00076 } l4_kernel_info_t;
00077
00085
00089 enum l4_kernel_info_consts_t
00090 {
00091     L4_KIP_VERSION_FIASCO              = 0x87004444,
00092     L4_KIP_VERSION_FIASCO_MASK        = 0xff00ffff,
00093 };
00094
00095 enum
00096 {
00105     L4_KIP_OFFS_READ_US                = 0x900,
00106
00116     L4_KIP_OFFS_READ_NS                = 0x980,
00117 };
00118
00122 extern l4_kernel_info_t const *l4_global_kip;
00123
00127 #define L4_KERNEL_INFO_MAGIC (0x4BE6344CL) /* "L4µK" */
00128
00129
00135 L4_INLINE l4_kernel_info_t const *l4_kip(void) L4_NOTHROW;

```

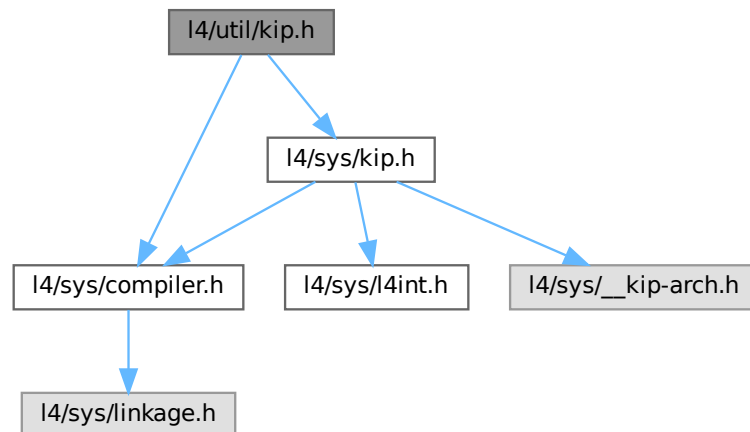
```

00136
00137
00145 L4_INLINE l4_umword_t l4_kip_version(l4_kernel_info_t const *kip) L4_NOTHROW;
00146
00154 L4_INLINE const char *l4_kip_version_string(l4_kernel_info_t const *kip) L4_NOTHROW;
00155
00164 L4_INLINE int
00165 l4_kernel_info_version_offset(l4_kernel_info_t const *kip) L4_NOTHROW;
00166
00184 L4_INLINE l4_cpu_time_t
00185 l4_kip_clock(l4_kernel_info_t const *kip) L4_NOTHROW;
00186
00199 L4_INLINE l4_umword_t
00200 l4_kip_clock_lw(l4_kernel_info_t const *kip) L4_NOTHROW
00201     L4_DEPRECATED("Use l4_kip_clock() instead");
00202
00216 L4_INLINE l4_uint64_t
00217 l4_kip_clock_ns(l4_kernel_info_t const *kip) L4_NOTHROW;
00218
00220
00221 /*****
00222  * Implementations
00223  *****/
00224
00225 L4_INLINE l4_kernel_info_t const*
00226 l4_kip(void) L4_NOTHROW
00227 { return l4_global_kip; }
00228
00229 L4_INLINE l4_umword_t
00230 l4_kip_version(l4_kernel_info_t const *kip) L4_NOTHROW
00231 { return kip->version & L4_KIP_VERSION_FIASCO_MASK; }
00232
00233 L4_INLINE const char*
00234 l4_kip_version_string(l4_kernel_info_t const *k) L4_NOTHROW
00235 { return (const char *)k + l4_kernel_info_version_offset(k); }
00236
00237 L4_INLINE int
00238 l4_kernel_info_version_offset(l4_kernel_info_t const *kip) L4_NOTHROW
00239 { return kip->offset_version_strings « 4; }
00240
00241 L4_INLINE l4_cpu_time_t
00242 l4_kip_clock(l4_kernel_info_t const *kip) L4_NOTHROW
00243 {
00244     // Use kernel-provided code to determine the current clock.
00245     typedef l4_uint64_t (*kip_time_fn_read_us)(void);
00246     kip_time_fn_read_us read_us =
00247         (kip_time_fn_read_us)((l4_uint8_t const*)kip + L4_KIP_OFFS_READ_US);
00248     return read_us();
00249 }
00250
00251 L4_INLINE l4_cpu_time_t
00252 l4_kip_clock_ns(l4_kernel_info_t const *kip) L4_NOTHROW
00253 {
00254     typedef l4_uint64_t (*kip_time_fn_read_ns)(void);
00255     kip_time_fn_read_ns read_ns =
00256         (kip_time_fn_read_ns)((l4_uint8_t const*)kip + L4_KIP_OFFS_READ_NS);
00257     return read_ns();
00258 }
00259
00260 L4_INLINE l4_umword_t
00261 l4_kip_clock_lw(l4_kernel_info_t const *kip) L4_NOTHROW
00262 {
00263     return l4_kip_clock(kip);
00264 }
00265
00272 #define l4_kip_for_each_feature(s) \
00273     for (s += __builtin_strlen(s) + 1; *s; s += __builtin_strlen(s) + 1)
00274
00285 L4_INLINE int
00286 l4_kip_kernel_has_feature(l4_kernel_info_t const *kip, char const *str)
00287 {
00288     const char *s = l4_kip_version_string(kip);
00289     if (!s)
00290         return 0;
00291     l4_kip_for_each_feature(s)
00292     {
00293         if (__builtin_strcmp(s, str) == 0)
00294             return 1;
00295     }
00296     return 0;
00297 }
00298
00299 }

```

## 17.647 l4/util/kip.h File Reference

```
#include <l4/sys/kip.h>
#include <l4/sys/compiler.h>
Include dependency graph for kip.h:
```



### Macros

- `#define l4util_kip_for_each_feature(s)`  
Cycle through kernel features given in the KIP.

### Functions

- `L4_BEGIN_DECLS int l4util_kip_kernel_has_feature (l4_kernel_info_t const *k, char const *str)`  
Check if kernel supports a feature.
- `unsigned long l4util_kip_kernel_abi_version (l4_kernel_info_t const *k)`  
Return kernel ABI version.

## 17.648 kip.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011
00012 #pragma once
00013
00014 #include <l4/sys/kip.h>
00015 #include <l4/sys/compiler.h>
00016
00022
```

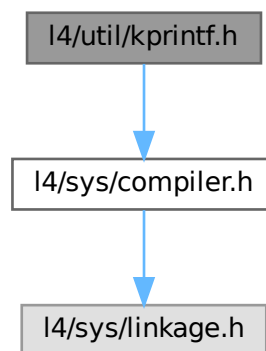
```
00023
00024 L4_BEGIN_DECLS
00025
00038 L4_CV int l4util_kip_kernel_has_feature(l4_kernel_info_t const *k, char const *str);
00039
00046 L4_CV unsigned long l4util_kip_kernel_abi_version(l4_kernel_info_t const *k);
00047
00048 L4_END_DECLS
00049
00058 #define l4util_kip_for_each_feature(s) l4_kip_for_each_feature(s)
00059
00061
```

## 17.649 l4/util/kprintf.h File Reference

printf using the kernel debugger

```
#include <l4/sys/compiler.h>
```

Include dependency graph for kprintf.h:



### 17.649.1 Detailed Description

printf using the kernel debugger

Date

04/05/2007

Author

Adam Lackorzynski [adam@os.inf.tu-dresden.de](mailto:adam@os.inf.tu-dresden.de),

Definition in file [kprintf.h](#).



## 17.650 kprintf.h

[Go to the documentation of this file.](#)

```

00001 /*****
00009 */
00010 * (c) 2007-2009 Author(s)
00011 *     economic rights: Technische Universität Dresden (Germany)
00012 * License: see LICENSE.spdx (in this directory or the directories above)
00013 */
00014
00015 #ifndef __L4UTIL__INCLUDE__KPRINTF_H__
00016 #define __L4UTIL__INCLUDE__KPRINTF_H__
00017
00018 #include <l4/sys/compiler.h>
00019
00020 L4_BEGIN_DECLS
00021
00022 L4_CV int l4_kprintf(const char *fmt, ...)
00023                 __attribute__((format (printf, 1, 2)));
00024
00025 L4_END_DECLS
00026
00027 #endif /* ! __L4UTIL__INCLUDE__KPRINTF_H__ */

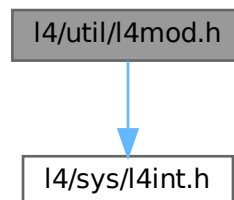
```

## 17.651 l4/util/l4mod.h File Reference

L4mod structures and constants.

```
#include <l4/sys/l4int.h>
```

Include dependency graph for l4mod.h:



### Data Structures

- struct [l4util\\_l4mod\\_mod](#)  
*A single module.*
- struct [l4util\\_l4mod\\_info](#)  
*Base module structure.*

### Enumerations

- enum [l4util\\_l4mod\\_mod\\_info\\_flag](#) {  
[L4util\\_l4mod\\_mod\\_flag\\_unspec](#) = 0 , [L4util\\_l4mod\\_mod\\_flag\\_kernel](#) = 1 , [L4util\\_l4mod\\_mod\\_flag\\_sigma0](#) =  
2 , [L4util\\_l4mod\\_mod\\_flag\\_roottask](#) = 3 ,  
[L4util\\_l4mod\\_mod\\_flag\\_mask](#) = 7 << 0 }  
*Flags for l4util\_l4mod\_mod.flags.*

## 17.651.1 Detailed Description

L4mod structures and constants.

Definition in file [l4mod.h](#).

## 17.651.2 Enumeration Type Documentation

### 17.651.2.1 l4util\_l4mod\_mod\_info\_flag

enum [l4util\\_l4mod\\_mod\\_info\\_flag](#)

Flags for [l4util\\_l4mod\\_mod.flags](#).

#### Enumerator

L4util_l4mod_mod_flag_unspec	Flag for a generic module.
L4util_l4mod_mod_flag_kernel	Flag for the kernel module.
L4util_l4mod_mod_flag_sigma0	Flag for the sigma0 module.
L4util_l4mod_mod_flag_roottask	Flag for the root task module.
L4util_l4mod_mod_flag_mask	Mask for specified flags.

Definition at line 17 of file [l4mod.h](#).

## 17.652 l4mod.h

[Go to the documentation of this file.](#)

```

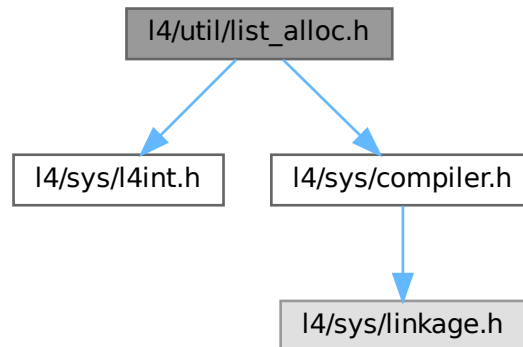
00001 /*
00002  * Copyright (C) 2021-2022, 2024 Kernkonzept GmbH.
00003  * Author(s): Adam Lackorzynski <adam@l4re.org>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00012 #pragma once
00013
00014 #include <l4/sys/l4int.h>
00015
00017 enum l4util_l4mod_mod_info_flag
00018 {
00019     L4util_l4mod_mod_flag_unspec    = 0,
00020     L4util_l4mod_mod_flag_kernel    = 1,
00021     L4util_l4mod_mod_flag_sigma0    = 2,
00022     L4util_l4mod_mod_flag_roottask  = 3,
00023     L4util_l4mod_mod_flag_mask      = 7 « 0,
00024 };
00025
00027 typedef struct
00028 {
00029     l4_uint64_t flags;
00030     l4_uint64_t mod_start;
00031     l4_uint64_t mod_end;
00032     l4_uint64_t cmdline;
00033 } l4util_l4mod_mod;
00034
00036 typedef struct
00037 {
00038     l4_uint64_t flags;
00039     l4_uint64_t cmdline;
00040     l4_uint64_t mods_addr;
00041     l4_uint32_t mods_count;
00042     l4_uint32_t _pad;
00043
00048     l4_uint64_t vbe_ctrl_info;
00049     l4_uint64_t vbe_mode_info;
00050 } l4util_l4mod_info;

```

## 17.653 l4/util/list\_alloc.h File Reference

Simple list-based allocator.

```
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>
Include dependency graph for list_alloc.h:
```



### Functions

- `L4_BEGIN_DECLS` void `l4la_free` (`l4la_free_t **first`, void `*block`, `l4_size_t` `size`)  
*Add free memory to memory pool.*
- void `* l4la_alloc` (`l4la_free_t **first`, `l4_size_t` `size`, unsigned `align`)  
*Allocate memory from pool.*
- void `l4la_dump` (`l4la_free_t **first`)  
*Show all list members.*
- void `l4la_init` (`l4la_free_t **first`)  
*Init memory pool.*
- `l4_size_t` `l4la_avail` (`l4la_free_t **first`)  
*Show available memory in pool.*

### 17.653.1 Detailed Description

Simple list-based allocator.

Taken from the Fiasco kernel.

#### Date

Alexander Warg <aw11os.inf.tu-dresden.de> Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [list\\_alloc.h](#).

## 17.653.2 Function Documentation

### 17.653.2.1 l4la\_alloc()

```
void * l4la_alloc (
    l4la_free_t ** first,
    l4_size_t size,
    unsigned align)
```

Allocate memory from pool.

#### Parameters

<i>first</i>	list identifier
<i>size</i>	length of memory block to allocate
<i>align</i>	alignment

References [L4\\_CV](#).

### 17.653.2.2 l4la\_avail()

```
l4_size_t l4la_avail (
    l4la_free_t ** first)
```

Show available memory in pool.

#### Parameters

<i>first</i>	list identifier
--------------	-----------------

References [L4\\_CV](#), and [L4\\_END\\_DECLS](#).

### 17.653.2.3 l4la\_dump()

```
void l4la_dump (
    l4la_free_t ** first)
```

Show all list members.

#### Parameters

<i>first</i>	list identifier
--------------	-----------------

References [L4\\_CV](#).

### 17.653.2.4 l4la\_free()

```
L4_BEGIN_DECLS void l4la_free (
    l4la_free_t ** first,
    void * block,
    l4_size_t size)
```

Add free memory to memory pool.

#### Parameters

---

<i>first</i>	list identifier
<i>block</i>	address of unused memory block
<i>size</i>	size of memory block

References [L4\\_CV](#).

### 17.653.2.5 l4la\_init()

```
void l4la_init (
    l4la_free_t ** first)
```

Init memory pool.

#### Parameters

<i>first</i>	list identifier
--------------	-----------------

References [L4\\_CV](#).

## 17.654 list\_alloc.h

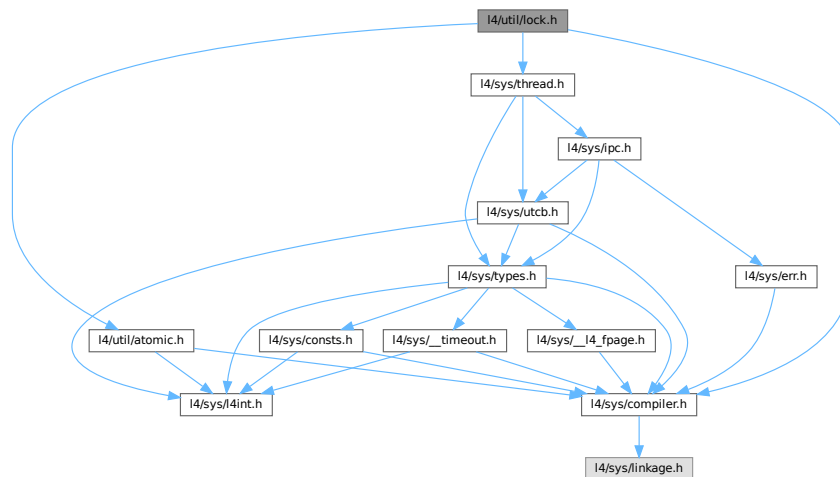
[Go to the documentation of this file.](#)

```
00001
00007
00008 /*
00009  * (c) 2003-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00010  *      Frank Mehnert <fm3@os.inf.tu-dresden.de>
00011  *      economic rights: Technische Universität Dresden (Germany)
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014
00015 #ifndef L4UTIL_L4LA_H
00016 #define L4UTIL_L4LA_H
00017
00018 #include <l4/sys/l4int.h>
00019 #include <l4/sys/compiler.h>
00020
00021 typedef struct l4la_free_t_s
00022 {
00023     struct l4la_free_t_s *next;
00024     l4_size_t             size;
00025 } l4la_free_t;
00026
00027 #define L4LA_INITIALIZER { 0 }
00028
00029 L4_BEGIN_DECLS
00030
00035 L4_CV void      l4la_free(l4la_free_t **first, void *block, l4_size_t size);
00036
00041 L4_CV void*     l4la_alloc(l4la_free_t **first, l4_size_t size, unsigned align);
00042
00045 L4_CV void      l4la_dump(l4la_free_t **first);
00046
00049 L4_CV void      l4la_init(l4la_free_t **first);
00050
00053 L4_CV l4_size_t l4la_avail(l4la_free_t **first);
00054
00055 L4_END_DECLS
00056
00057 #endif
```

## 17.655 I4/util/lock.h File Reference

Simple lock implementation.

```
#include <l4/sys/thread.h>
#include <l4/sys/compiler.h>
#include <l4/util/atomic.h>
Include dependency graph for lock.h:
```



### 17.655.1 Detailed Description

Simple lock implementation.

Does only work if all thread have the same priority!

Date

02/1997

Author

Michael Hohmuth [hohmuth@os.inf.tu-dresden.de](mailto:hohmuth@os.inf.tu-dresden.de)

Definition in file [lock.h](#).

## 17.656 lock.h

[Go to the documentation of this file.](#)

```

00001 /*****
00009 */
00010 * (c) 2000-2009 Author(s)
00011 *     economic rights: Technische Universität Dresden (Germany)
00012 * License: see LICENSE.spdx (in this directory or the directories above)
00013 */
00014
00015 /*****
00016 #ifndef __L4UTIL_LOCK_H__
00017 #define __L4UTIL_LOCK_H__
00018
00019 #include <l4/sys/thread.h>
00020 #include <l4/sys/compiler.h>
00021 #include <l4/util/atomic.h>
00022
00023 L4_BEGIN_DECLS
00024
00025 typedef l4_uint32_t l4util_simple_lock_t;
00026
00027 L4_INLINE int l4_simple_try_lock(l4util_simple_lock_t *lock);
00028 L4_INLINE void l4_simple_unlock(l4util_simple_lock_t *lock);
00029 L4_INLINE int l4_simple_lock_locked(l4util_simple_lock_t *lock);
00030 L4_INLINE void l4_simple_lock_solid(register l4util_simple_lock_t *p);
00031 L4_INLINE void l4_simple_lock(l4util_simple_lock_t * lock);
00032
00033 L4_INLINE int
00034 l4_simple_try_lock(l4util_simple_lock_t *lock)
00035 {
00036     return l4util_xchg32(lock, 1) == 0;
00037 }
00038
00039 L4_INLINE void
00040 l4_simple_unlock(l4util_simple_lock_t *lock)
00041 {
00042     *lock = 0;
00043 }
00044
00045 L4_INLINE int
00046 l4_simple_lock_locked(l4util_simple_lock_t *lock)
00047 {
00048     return (*lock == 0) ? 0 : 1;
00049 }
00050
00051 L4_INLINE void
00052 l4_simple_lock_solid(register l4util_simple_lock_t *p)
00053 {
00054     while (l4_simple_lock_locked(p) || !l4_simple_try_lock(p))
00055         l4_thread_switch(L4_INVALID_CAP);
00056 }
00057
00058 L4_INLINE void
00059 l4_simple_lock(l4util_simple_lock_t * lock)
00060 {
00061     if (!l4_simple_try_lock(lock))
00062         l4_simple_lock_solid(lock);
00063 }
00064
00065 L4_END_DECLS
00066
00067 #endif

```

## 17.657 l4/util/mb\_info.h File Reference

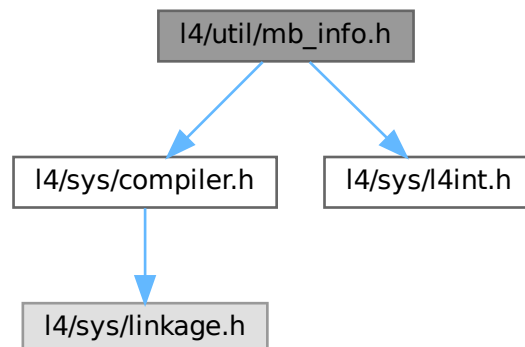
Multiboot info structure as defined by GRUB.

```

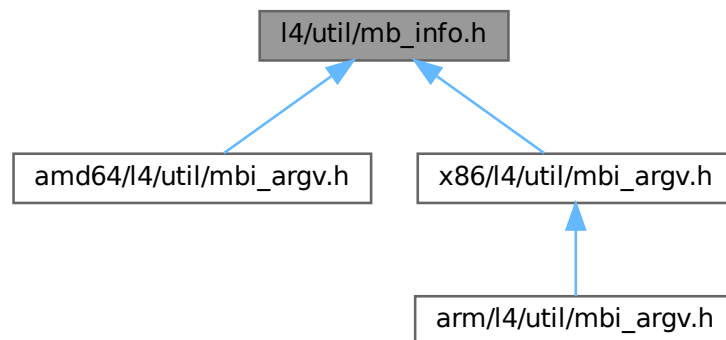
#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>

```

Include dependency graph for mb\_info.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [l4util\\_mb\\_mod\\_t](#)

The structure type "mod\_list" is used by the [multiboot\\_info](#) structure.

- struct [l4util\\_mb\\_addr\\_range\\_t](#)

INT-15, AX=E820 style "AddressRangeDescriptor" ...with a "size" parameter on the front which is the structure size - 4, pointing to the next one, up until the full buffer length of the memory map has been reached.

- struct [l4util\\_mb\\_drive\\_t](#)

Drive Info structure.

- struct [l4util\\_mb\\_apm\\_t](#)

APM BIOS info.

- struct [l4util\\_mb\\_vbe\\_ctrl\\_t](#)



- *VBE controller information.*
- struct [l4util\\_mb\\_vbe\\_mode\\_t](#)
- *VBE mode information.*
- struct [l4util\\_mb\\_info\\_t](#)
- *MultiBoot Info description.*

## Macros

- #define **MB\_ARD\_MEMORY** 1  
*usable memory "Type", all others are reserved.*
- #define **MB\_ART\_MEMORY** 1  
*Address Range Types (ART) from "Advanced Configuration and Power Interface Specification" Rev3.0a (p.*
- #define **MB\_ART\_RESERVED** 2  
*in use or reserved by system*
- #define **MB\_ART\_ACPI** 3  
*ACPI Reclaim Memory (RAM that contains ACPI tables).*
- #define **MB\_ART\_NVS** 4  
*ACPI NVS Memory (must not be used by the OS.*
- #define **MB\_ART\_UNUSABLE** 5  
*memory in which errors have been detected*
- #define [l4util\\_mb\\_for\\_each\\_mmap\\_entry](#)(i, mbi)  
*Iterate over a memory map provided in a Multiboot info.*
- #define **L4UTIL\_MB\_MEMORY** 0x00000001  
*Flags to be set in the 'flags' parameter above.*
- #define **L4UTIL\_MB\_BOOTDEV** 0x00000002  
*is there a boot device set?*
- #define **L4UTIL\_MB\_CMDLINE** 0x00000004  
*is the command-line defined?*
- #define **L4UTIL\_MB\_MODS** 0x00000008  
*are there modules to do something with?*
- #define **L4UTIL\_MB\_AOUT\_SYMS** 0x00000010  
*is there a symbol table loaded?*
- #define **L4UTIL\_MB\_ELF\_SHDR** 0x00000020  
*is there an ELF section header table?*
- #define **L4UTIL\_MB\_MEM\_MAP** 0x00000040  
*is there a full memory map?*
- #define **L4UTIL\_MB\_DRIVE\_INFO** 0x00000080  
*Is there drive info?*
- #define **L4UTIL\_MB\_CONFIG\_TABLE** 0x00000100  
*Is there a config table?*
- #define **L4UTIL\_MB\_BOOT\_LOADER\_NAME** 0x00000200  
*Is there a boot loader name?*
- #define **L4UTIL\_MB\_APM\_TABLE** 0x00000400  
*Is there a APM table?*
- #define **L4UTIL\_MB\_VIDEO\_INFO** 0x00000800  
*Is there video information?*
- #define **L4UTIL\_MB\_VALID** 0x2BADB002UL  
*If we are multiboot-compliant, this value is present in the eax register.*

### 17.657.1 Detailed Description

Multiboot info structure as defined by GRUB.

Definition in file [mb\\_info.h](#).

### 17.657.2 Macro Definition Documentation

#### 17.657.2.1 l4util\_mb\_for\_each\_mmap\_entry

```
#define l4util_mb_for_each_mmap_entry(  
    i,  
    mbi)
```

**Value:**

```
for (i = l4util_mb_first_mmap_entry(mbi);  
     (unsigned long)i < (unsigned long)mbi->mmap_addr + mbi->mmap_length;  
     i = l4util_mb_next_mmap_entry(i))
```

Iterate over a memory map provided in a Multiboot info.

#### Parameters

<i>i</i>	Name of a variable of type <a href="#">l4util_mb_addr_range_t</a> * that is consecutively assigned pointers to the entries of the memory map.
<i>mbi</i>	Pointer to the <a href="#">l4util_mb_info_t</a> where the memory map can be found.

Definition at line 332 of file [mb\\_info.h](#).

#### 17.657.2.2 L4UTIL\_MB\_MEMORY

```
#define L4UTIL_MB_MEMORY 0x00000001
```

Flags to be set in the 'flags' parameter above.

is there basic lower/upper memory information?

Definition at line 344 of file [mb\\_info.h](#).

#### 17.657.2.3 MB\_ART\_MEMORY

```
#define MB_ART_MEMORY 1
```

Address Range Types (ART) from "Advanced Configuration and Power Interface Specification" Rev3.0a (p.

390). Other values are undefined. available, usable RAM

Definition at line 64 of file [mb\\_info.h](#).

## 17.658 mb\_info.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Frank Mehnert <fm3@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011
00012 #ifndef L4UTIL_MB_INFO_H
00013 #define L4UTIL_MB_INFO_H
00014
00015 /*****
00016  * Multiboot (v1)
00017  *****/
00018
00019 #ifndef __ASSEMBLY__
00020
00021 #include <l4/sys/compiler.h>
00022 #include <l4/sys/l4int.h>
00023
00024 /*
00025  * \defgroup l4util_mb_mod Multiboot v1
00026  * \ingroup l4util_api
00027  */
00028
00033 typedef struct
00034 {
00035     l4_uint32_t mod_start;
00036     l4_uint32_t mod_end;
00037     l4_uint32_t cmdline;
00038     l4_uint32_t pad;
00039 } l4util_mb_mod_t;
00040
00041
00048 typedef struct __attribute__((packed))
00049 {
00050     l4_uint32_t struct_size;
00051     l4_uint64_t addr;
00052     l4_uint64_t size;
00053     l4_uint32_t type;
00054     /* unspecified optional padding... */
00055 } l4util_mb_addr_range_t;
00056
00058 #define MB_ARD_MEMORY 1
00059
00064 #define MB_ART_MEMORY 1
00065 #define MB_ART_RESERVED 2
00066 #define MB_ART_ACPI 3
00068 #define MB_ART_NVS 4
00069 #define MB_ART_UNUSABLE 5
00070
00071
00073 typedef struct
00074 {
00075     l4_uint32_t size;
00076     l4_uint8_t drive_number;
00077     l4_uint8_t drive_mode;
00078     l4_uint16_t drive_cylinders;
00079     l4_uint8_t drive_heads;
00080     l4_uint8_t drive_sectors;
00081     l4_uint16_t drive_ports[0];
00082 } l4util_mb_drive_t;
00083
00084 /* Drive Mode. */
00085 #define MB_DI_CHS_MODE 0
00086 #define MB_DI_LBA_MODE 1
00087
00088
00090 typedef struct
00091 {
00092     l4_uint16_t version;
00093     l4_uint16_t cseg;
00094     l4_uint32_t offset;
00095     l4_uint16_t cseg_l6;
00096     l4_uint16_t dseg_l6;
00097     l4_uint16_t flags;
00098     l4_uint16_t cseg_len;
00099     l4_uint16_t cseg_l6_len;
00100     l4_uint16_t dseg_l6_len;
00101 } __attribute__((packed)) l4util_mb_apm_t;
00102 static_assert(sizeof(l4util_mb_apm_t) == 20, "Check l4util_mb_apm_t");
00103

```

```

00104
00106 typedef struct
00107 {
00108     l4_uint8_t signature[4];
00109     l4_uint16_t version;
00110     l4_uint32_t oem_string;
00111     l4_uint32_t capabilities;
00112     l4_uint32_t video_mode;
00113     l4_uint16_t total_memory;
00114     l4_uint16_t oem_software_rev;
00115     l4_uint32_t oem_vendor_name;
00116     l4_uint32_t oem_product_name;
00117     l4_uint32_t oem_product_rev;
00118     l4_uint8_t reserved[222];
00119     l4_uint8_t oem_data[256];
00120 } __attribute__((packed)) l4util_mb_vbe_ctrl_t;
00121 static_assert(sizeof(l4util_mb_vbe_ctrl_t) == 512, "Check l4util_mb_vbe_ctrl_t");
00122
00123
00125 typedef struct
00126 {
00130     l4_uint16_t mode_attributes;
00132     l4_uint8_t win_a_attributes;
00134     l4_uint8_t win_b_attributes;
00136     l4_uint16_t win_granularity;
00138     l4_uint16_t win_size;
00140     l4_uint16_t win_a_segment;
00142     l4_uint16_t win_b_segment;
00144     l4_uint32_t win_func;
00146     l4_uint16_t bytes_per_scanline;
00148
00152     l4_uint16_t x_resolution;
00154     l4_uint16_t y_resolution;
00156     l4_uint8_t x_char_size;
00158     l4_uint8_t y_char_size;
00160     l4_uint8_t number_of_planes;
00162     l4_uint8_t bits_per_pixel;
00164     l4_uint8_t number_of_banks;
00166     l4_uint8_t memory_model;
00168     l4_uint8_t bank_size;
00170     l4_uint8_t number_of_image_pages;
00172     l4_uint8_t reserved0;
00174
00178     l4_uint8_t red_mask_size;
00180     l4_uint8_t red_field_position;
00182     l4_uint8_t green_mask_size;
00184     l4_uint8_t green_field_position;
00186     l4_uint8_t blue_mask_size;
00188     l4_uint8_t blue_field_position;
00190     l4_uint8_t reserved_mask_size;
00192     l4_uint8_t reserved_field_position;
00194     l4_uint8_t direct_color_mode_info;
00196
00200     l4_uint32_t phys_base;
00202     l4_uint32_t reserved1;
00204     l4_uint16_t reversed2;
00206
00210     l4_uint16_t linear_bytes_per_scanline;
00212     l4_uint8_t banked_number_of_image_pages;
00214     l4_uint8_t linear_number_of_image_pages;
00216     l4_uint8_t linear_red_mask_size;
00218     l4_uint8_t linear_red_field_position;
00220     l4_uint8_t linear_green_mask_size;
00222     l4_uint8_t linear_green_field_position;
00224     l4_uint8_t linear_blue_mask_size;
00226     l4_uint8_t linear_blue_field_position;
00228     l4_uint8_t linear_reserved_mask_size;
00230     l4_uint8_t linear_reserved_field_position;
00232     l4_uint32_t max_pixel_clock;
00234     l4_uint8_t reserved3[190];
00236 } __attribute__((packed)) l4util_mb_vbe_mode_t;
00237 static_assert(sizeof(l4util_mb_vbe_mode_t) == 256, "Check l4util_mb_vbe_mode_t");
00238
00239
00246
00247 typedef struct
00248 {
00249     l4_uint32_t flags;
00250     l4_uint32_t mem_lower;
00251     l4_uint32_t mem_upper;
00252     l4_uint32_t boot_device;
00253     l4_uint32_t cmdline;
00254     l4_uint32_t mods_count;
00255     l4_uint32_t mods_addr;
00256
00257     union
00258     {

```

```

00259     struct
00260     {
00262         l4_uint32_t tabsize;
00263         l4_uint32_t strsize;
00264         l4_uint32_t addr;
00265         l4_uint32_t pad;
00266     }
00267     a;
00268
00269     struct
00270     {
00272         l4_uint32_t num;
00273         l4_uint32_t size;
00274         l4_uint32_t addr;
00275         l4_uint32_t shndx;
00276     }
00277     e;
00278 }
00279 syms;
00280
00281 l4_uint32_t mmap_length;
00282 l4_uint32_t mmap_addr;
00283 l4_uint32_t drives_length;
00284 l4_uint32_t drives_addr;
00285 l4_uint32_t config_table;
00286 l4_uint32_t boot_loader_name;
00287 l4_uint32_t apm_table;
00288 l4_uint32_t vbe_ctrl_info;
00289 l4_uint32_t vbe_mode_info;
00290 l4_uint16_t vbe_mode;
00291 l4_uint16_t vbe_interface_seg;
00292 l4_uint16_t vbe_interface_off;
00293 l4_uint16_t vbe_interface_len;
00294 } l4util_mb_info_t;
00295 static_assert(sizeof(l4util_mb_info_t) == 88, "Check l4util_mb_info_t");
00296
00303 static inline l4util_mb_addr_range_t *
00304 l4util_mb_first_mmap_entry(l4util_mb_info_t *mbi)
00305 {
00306     return (l4util_mb_addr_range_t *) (l4_addr_t) mbi->mmap_addr;
00307 }
00308
00318 static inline l4util_mb_addr_range_t *
00319 l4util_mb_next_mmap_entry(l4util_mb_addr_range_t *e)
00320 {
00321     return (l4util_mb_addr_range_t *) ((l4_addr_t) e + e->struct_size
00322                                         + sizeof(e->struct_size));
00323 }
00324
00332 #define l4util_mb_for_each_mmap_entry(i, mbi) \
00333     for (i = l4util_mb_first_mmap_entry(mbi); \
00334          (unsigned long) i < (unsigned long) mbi->mmap_addr + mbi->mmap_length; \
00335          i = l4util_mb_next_mmap_entry(i)) \
00336
00337 #endif /* ! __ASSEMBLY__ */
00338
00342
00344 #define L4UTIL_MB_MEMORY      0x00000001
00345
00347 #define L4UTIL_MB_BOOTDEV     0x00000002
00348
00350 #define L4UTIL_MB_CMDLINE     0x00000004
00351
00353 #define L4UTIL_MB_MODS        0x00000008
00354
00355 /* These next two are mutually exclusive */
00357 #define L4UTIL_MB_AOUT_SYMS    0x00000010
00358
00360 #define L4UTIL_MB_ELF_SHDR     0x00000020
00361
00363 #define L4UTIL_MB_MEM_MAP      0x00000040
00364
00366 #define L4UTIL_MB_DRIVE_INFO    0x00000080
00367
00369 #define L4UTIL_MB_CONFIG_TABLE 0x00000100
00370
00372 #define L4UTIL_MB_BOOT_LOADER_NAME 0x00000200
00373
00375 #define L4UTIL_MB_APM_TABLE     0x00000400
00376
00378 #define L4UTIL_MB_VIDEO_INFO    0x00000800
00379
00380
00382 #define L4UTIL_MB_VALID         0x2BADB002UL
00383 #define L4UTIL_MB_VALID_ASM     0x2BADB002
00384
00385

```

```

00386 /*****
00387  * Multiboot2
00388  *****/
00389
00390 #ifndef __ASSEMBLY__
00391
00392 typedef struct
00393 {
00394     l4_uint32_t total_size;
00395     l4_uint32_t reserved;
00396 } __attribute__((packed)) l4util_mb2_info_t;
00397
00398 typedef struct
00399 {
00400     char string[0];
00401 } __attribute__((packed)) l4util_mb2_cmdline_tag_t;
00402
00403 typedef struct
00404 {
00405     l4_uint32_t mod_start;
00406     l4_uint32_t mod_end;
00407     char string[];
00408 } __attribute__((packed)) l4util_mb2_module_tag_t;
00409
00410 typedef struct
00411 {
00412     l4_uint64_t base_addr;
00413     l4_uint64_t length;
00414     l4_uint32_t type;
00415     l4_uint32_t reserved;
00416 } __attribute__((packed)) l4util_mb2_memmap_entry_t;
00417
00418 typedef struct
00419 {
00420     l4_uint32_t entry_size;
00421     l4_uint32_t entry_version;
00422     l4util_mb2_memmap_entry_t entries[];
00423 } __attribute__((packed)) l4util_mb2_memmap_tag_t;
00424
00425 typedef struct
00426 {
00427     char data[0];
00428 } __attribute__((packed)) l4util_mb2_rsdp_tag_t;
00429
00430
00431 struct color_info_rgb_t
00432 {
00433     l4_uint8_t framebuffer_red_field_position;
00434     l4_uint8_t framebuffer_red_mask_size;
00435     l4_uint8_t framebuffer_green_field_position;
00436     l4_uint8_t framebuffer_green_mask_size;
00437     l4_uint8_t framebuffer_blue_field_position;
00438     l4_uint8_t framebuffer_blue_mask_size;
00439 } __attribute__((packed));
00440
00441 typedef struct
00442 {
00443     l4_uint64_t framebuffer_addr;
00444     l4_uint32_t framebuffer_pitch;
00445     l4_uint32_t framebuffer_width;
00446     l4_uint32_t framebuffer_height;
00447     l4_uint8_t framebuffer_bpp;
00448     l4_uint8_t framebuffer_type;
00449     l4_uint8_t reserved;
00450
00451     // color_info;
00452     union
00453     {
00454         struct color_info_rgb_t color_info_rgb;
00455     };
00456 } __attribute__((packed)) l4util_mb2_framebuffer_tag_t;
00457
00458 typedef struct
00459 {
00460     l4_uint32_t type;
00461     l4_uint32_t size;
00462
00463     union
00464     {
00465         l4util_mb2_cmdline_tag_t cmdline;
00466         l4util_mb2_module_tag_t module;
00467         l4util_mb2_memmap_tag_t memmap;
00468         l4util_mb2_framebuffer_tag_t fb;
00469         l4util_mb2_rsdp_tag_t rsdp;
00470     };
00471 } __attribute__((packed)) l4util_mb2_tag_t;
00472

```

```

00473 #endif /* ! __ASSEMBLY__ */
00474
00475
00476 #define L4UTIL_MB2_MAGIC      0xE85250D6
00477 #define L4UTIL_MB2_ARCH_I386  0x0
00478
00479 #define L4UTIL_MB2_TERMINATOR_HEADER_TAG  0
00480 #define L4UTIL_MB2_INFO_REQUEST_HEADER_TAG 1
00481 #define L4UTIL_MB2_ENTRY_ADDRESS_HEADER_TAG 3
00482 #define L4UTIL_MB2_FRAMEBUFFER_HEADER_TAG 5
00483 #define L4UTIL_MB2_RELOCATABLE_HEADER_TAG 10
00484
00485 #define L4UTIL_MB2_TAG_FLAG_REQUIRED      0
00486
00487 #define L4UTIL_MB2_TAG_ALIGN_SHIFT      3
00488 #define L4UTIL_MB2_TAG_ALIGN           8
00489
00490 #define L4UTIL_MB2_TERMINATOR_INFO_TAG      0
00491 #define L4UTIL_MB2_BOOT_CMDLINE_INFO_TAG    1
00492 #define L4UTIL_MB2_MODULE_INFO_TAG          3
00493 #define L4UTIL_MB2_MEMORY_MAP_INFO_TAG      6
00494 #define L4UTIL_MB2_FRAMEBUFFER_INFO_TAG     8
00495 #define L4UTIL_MB2_RSDP_OLD_INFO_TAG        14
00496 #define L4UTIL_MB2_RSDP_NEW_INFO_TAG        15
00497 #define L4UTIL_MB2_IMAGE_LOAD_BASE_PHYS_INFO_TAG 21
00498
00499 #define L4UTIL_MB2_RELO_PREFERRED_NONE 0
00500 #define L4UTIL_MB2_RELO_PREFERRED_MIN  1
00501 #define L4UTIL_MB2_RELO_PREFERRED_MAX   2
00502
00503 #endif

```

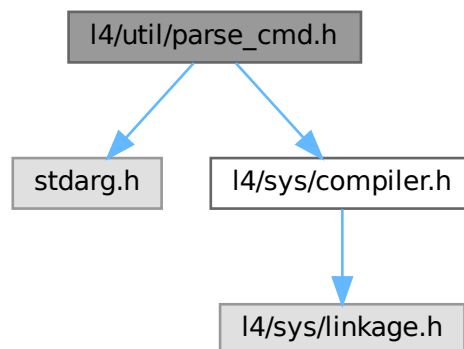
## 17.659 l4/util/parse\_cmd.h File Reference

comfortable command-line parsing

```
#include <stdarg.h>
```

```
#include <l4/sys/compiler.h>
```

Include dependency graph for parse\_cmd.h:



### Typedefs

- typedef void(\* **parse\_cmd\_fn\_t**) (int)  
Function type for `PARSE_CMD_FN`.
- typedef void(\* **parse\_cmd\_fn\_arg\_t**) (int, const char \*, int)  
Function type for `PARSE_CMD_FN_ARG`.

## Enumerations

- enum [parse\\_cmd\\_type](#)

*Types for parsing.*

## Functions

- [L4\\_BEGIN\\_DECLS](#) int [parse\\_cmdline](#) (int \*argc, const char \*\*\*argv, int arg0,...)

*Parse the command-line for specified arguments and store the values into variables.*

## 17.659.1 Detailed Description

comfortable command-line parsing

### Date

2002

### Author

Jork Loeser [jork.loeser@inf.tu-dresden.de](mailto:jork.loeser@inf.tu-dresden.de)

Definition in file [parse\\_cmd.h](#).

## 17.660 [parse\\_cmd.h](#)

[Go to the documentation of this file.](#)

```

00001
00008
00009 /*
00010  * (c) 2003-2009 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014
00015 #ifndef __PARSE_CMD_H
00016 #define __PARSE_CMD_H
00017
00018 #include <stdarg.h>
00019 #include <l4/sys/compiler.h>
00020
00026
00030 enum parse_cmd_type {
00031     PARSE_CMD_INT,
00032     PARSE_CMD_SWITCH,
00033     PARSE_CMD_STRING,
00034     PARSE_CMD_FN,
00035     PARSE_CMD_FN_ARG,
00036     PARSE_CMD_INC,
00037     PARSE_CMD_DEC,
00038 };
00039
00043 typedef L4_CV void (*parse_cmd_fn_t)(int);
00044
00048 typedef L4_CV void (*parse_cmd_fn_arg_t)(int, const char*, int);
00049
00050 L4_BEGIN_DECLS
00051
00138 L4_CV int parse_cmdline(int *argc, const char***argv, int arg0, ...);
00139 L4_CV int parse_cmdlinev(int *argc, const char***argv, int arg0, va_list va);
00140 L4_CV int parse_cmdline_extra(const char*argv0, const char*line, char delim,
00141                             int arg0,...);
00142
00143 L4_END_DECLS
00145
00146 #endif
00147

```



## 17.661 printf\_helpers.h

```

00001 /*
00002  * Copyright (C) 2025 Kernkonzept GmbH.
00003  * Author(s): Frank Mehnert <frank.mehnert@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <stddef.h>
00011 #include <stdio.h>
00012 #include <l4/sys/compiler.h>
00013
00030 L4_INLINE int l4util_human_readable_size(char *outstr, size_t outsize,
00031                                         unsigned long long bytes)
00032 {
00033     static char const *const unitstr = "BKMGT";
00034
00035     int idx = sizeof(unitstr) - 2;
00036     int order;
00037
00038     for (order = idx * 10; order > 10; order -= 10, --idx)
00039         if (bytes > (1ULL « order))
00040             break;
00041
00042     unsigned long long value = bytes » order;
00043     unsigned long long fract = (bytes - (value « order))
00044                               / ((1ULL « order) / 10 + 1);
00045
00046     return snprintf(outstr, outsize, "%llu.%1llu %ciB",
00047                    value, fract, unitstr[idx]);
00048 }

```

## 17.662 l4/util/rand.h File Reference

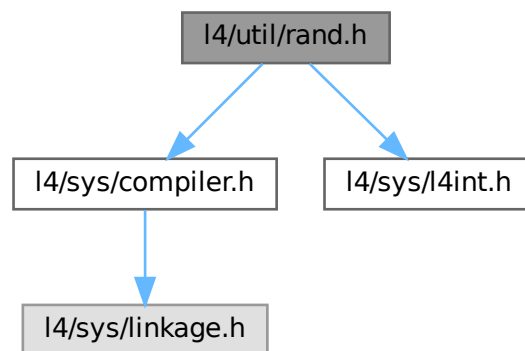
Simple Pseudo-Random Number Generator.

```

#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>

```

Include dependency graph for rand.h:



### Functions

- [l4\\_uint32\\_t l4util\\_rand](#) (void)

*Deliver next random number.*

- void `l4util_srand` (`l4_uint32_t` seed)

*Initialize random number generator.*

## 17.662.1 Detailed Description

Simple Pseudo-Random Number Generator.

### Date

1998

### Author

Lars Reuther [reuther@os.inf.tu-dresden.de](mailto:reuther@os.inf.tu-dresden.de)

Definition in file [rand.h](#).

## 17.663 rand.h

[Go to the documentation of this file.](#)

```

00001
00008 /*
00009  * (c) 2008-2009 Author(s)
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013
00014 #ifndef __L4UTIL_RAND_H
00015 #define __L4UTIL_RAND_H
00016
00017 #define L4_RAND_MAX 65535
00018
00019 #include <l4/sys/compiler.h>
00020 #include <l4/sys/l4int.h>
00021
00022 L4_BEGIN_DECLS
00023
00028
00035 L4_CV l4_uint32_t
00036 l4util_rand(void);
00037
00044 L4_CV void
00045 l4util_srand (l4_uint32_t seed);
00046
00047 L4_END_DECLS
00048
00049 #endif /* __L4UTIL_RAND_H */

```

## 17.664 l4/util/splitlog2.h File Reference

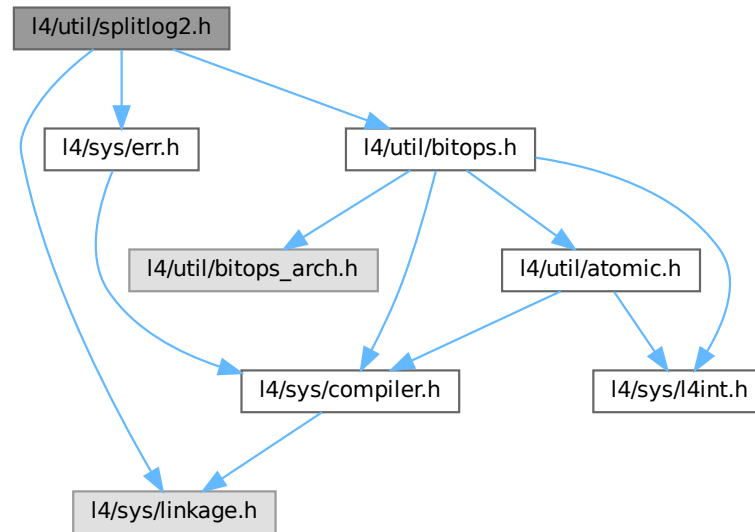
Split a range in log2 aligned and size-aligned chunks.

```

#include <l4/sys/linkage.h>
#include <l4/sys/err.h>

```

#include <l4/util/bitops.h>  
 Include dependency graph for splitlog2.h:



## Functions

- `L4_BEGIN_DECLS` long `l4util_splitlog2_hdl` (`l4_addr_t` start, `l4_addr_t` end, long(`*handler`)(`l4_addr_t` s, `l4_addr_t` e, int log2size))  
*Split a range into log2 base and size aligned chunks.*
- `l4_addr_t` `l4util_splitlog2_size` (`l4_addr_t` start, `l4_addr_t` end)  
*Return log2 base and size aligned length of a range.*

### 17.664.1 Detailed Description

Split a range in log2 aligned and size-aligned chunks.

Definition in file [splitlog2.h](#).

## 17.665 splitlog2.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010 #ifndef __L4UTIL__INCLUDE__SPLITLOG2_H__
00011 #define __L4UTIL__INCLUDE__SPLITLOG2_H__
00012
00013 #include <l4/sys/linkage.h>
00014 #include <l4/sys/err.h>
00015 #include <l4/util/bitops.h>

```

```

00016
00017 L4_BEGIN_DECLS
00018
00031 L4_INLINE long
00032 l4util_splitlog2_hdl(l4_addr_t start, l4_addr_t end,
00033                     long (*handler)(l4_addr_t s, l4_addr_t e, int log2size));
00034
00043 L4_INLINE l4_addr_t
00044 l4util_splitlog2_size(l4_addr_t start, l4_addr_t end);
00045
00046 L4_END_DECLS
00047
00048 /* Implementation */
00049
00050 L4_INLINE long
00051 l4util_splitlog2_hdl(l4_addr_t start, l4_addr_t end,
00052                     long (*handler)(l4_addr_t s, l4_addr_t e, int log2size))
00053 {
00054     if (end < start)
00055         return -L4_EINVAL;
00056     while (start <= end)
00057     {
00058         long retval;
00059         int len2 = l4util_splitlog2_size(start, end);
00060         l4_addr_t len = 1UL << len2;
00061         if ((retval = handler(start, start + len - 1, len2)))
00062             return retval;
00063         start += len;
00064     }
00065     return 0;
00066 }
00067
00068 L4_INLINE l4_addr_t
00069 l4util_splitlog2_size(l4_addr_t start, l4_addr_t end)
00070 {
00071     int start_bits = l4util_bsf(start);
00072     int len_bits = l4util_bsr(end - start + 1);
00073     if (start_bits != -1 && len_bits > start_bits)
00074         len_bits = start_bits;
00075     return len_bits;
00076 }
00077
00078 }
00079
00080 #endif /* ! __L4UTIL__INCLUDE__SPLITLOG2_H__ */

```

## 17.666 arm/l4/sys/thread.h File Reference

ARM-specific thread related definitions.

### Enumerations

- enum [L4\\_thread\\_ex\\_regs\\_flags\\_arm](#) { [L4\\_THREAD\\_EX\\_REGS\\_ARM\\_SET\\_EL\\_MASK](#) = 0x3 << 24 , [L4\\_THREAD\\_EX\\_REGS\\_ARM\\_SET\\_EL\\_KEEP](#) = 0x0 << 24 , [L4\\_THREAD\\_EX\\_REGS\\_ARM\\_SET\\_EL\\_ELO](#) = 0x1 << 24 , [L4\\_THREAD\\_EX\\_REGS\\_ARM\\_SET\\_EL\\_EL1](#) = 0x2 << 24 }

*Arm specific [L4::Thread::ex\\_regs\(\)](#) flags.*

### Functions

- [l4\\_msgtag\\_t l4\\_thread\\_arm\\_set\\_tpidruru](#) ([l4\\_cap\\_idx\\_t](#) thread, [l4\\_addr\\_t](#) tpidruru) [L4\\_NOTHROW](#)

*Set the [TPIDRURO](#) thread specific register.*

### 17.666.1 Detailed Description

ARM-specific thread related definitions.

Definition in file [thread.h](#).

## 17.667 thread.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00013 #include_next <l4/sys/thread.h>
00014
00015 // Use the full documentation from ARM64. Otherwise \parameters and \return
00016 // would occur twice in the Doxygen documentation of this function.
00020 L4_INLINE l4_msgtag_t
00021 l4_thread_arm_set_tpidruro(l4_cap_idx_t thread, l4_addr_t tpidruro) L4_NOTHROW;
00022
00027 L4_INLINE l4_msgtag_t
00028 l4_thread_arm_set_tpidruro_u(l4_cap_idx_t thread, l4_addr_t tpidruro,
00029                             l4_utcb_t *utcb) L4_NOTHROW;
00030
00039 enum l4_thread_ex_regs_flags_arm
00040 {
00042     L4_THREAD_EX_REGS_ARM_SET_EL_MASK      = 0x3 << 24,
00044     L4_THREAD_EX_REGS_ARM_SET_EL_KEEP      = 0x0 << 24,
00046     L4_THREAD_EX_REGS_ARM_SET_EL_EL0       = 0x1 << 24,
00048     L4_THREAD_EX_REGS_ARM_SET_EL_EL1       = 0x2 << 24,
00049 };
00050
00051 /* IMPLEMENTATION ----- */
00052
00053 L4_INLINE l4_msgtag_t
00054 l4_thread_arm_set_tpidruro_u(l4_cap_idx_t thread, l4_addr_t tpidruro,
00055                             l4_utcb_t *utcb) L4_NOTHROW
00056 {
00057     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00058     v->mr[0] = L4_THREAD_ARM_TPIDRURO_OP;
00059     v->mr[1] = tpidruro;
00060     return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0),
00061                       L4_IPC_NEVER);
00062 }
00063
00064 L4_INLINE l4_msgtag_t
00065 l4_thread_arm_set_tpidruro(l4_cap_idx_t thread, l4_addr_t tpidruro) L4_NOTHROW
00066 {
00067     return l4_thread_arm_set_tpidruro_u(thread, tpidruro, l4_utcb());
00068 }

```

## 17.668 arm64/l4/sys/thread.h File Reference

ARM64-specific thread related definitions.

### Enumerations

- enum [l4\\_thread\\_ex\\_regs\\_flags\\_arm64](#) { [L4\\_THREAD\\_EX\\_REGS\\_ARM64\\_SET\\_EL\\_MASK](#) = 0x3 << 24 , [L4\\_THREAD\\_EX\\_REGS\\_ARM64\\_SET\\_EL\\_KEEP](#) = 0x0 << 24 , [L4\\_THREAD\\_EX\\_REGS\\_ARM64\\_SET\\_EL\\_EL0](#) = 0x1 << 24 , [L4\\_THREAD\\_EX\\_REGS\\_ARM64\\_SET\\_EL\\_EL1](#) = 0x2 << 24 }

*Arm64 specific [L4::Thread::ex\\_regs\(\)](#) flags.*

### Functions

- [l4\\_msgtag\\_t l4\\_thread\\_arm\\_set\\_tpidruro](#) ([l4\\_cap\\_idx\\_t](#) thread, [l4\\_addr\\_t](#) tpidruro) [L4\\_NOTHROW](#)

*Set the [TPIDRURO](#) thread specific register.*

## 17.668.1 Detailed Description

ARM64-specific thread related definitions.

Definition in file [thread.h](#).

## 17.669 thread.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00013 #include_next <l4/sys/thread.h>
00014
00027 L4_INLINE l4_msgtag_t
00028 l4_thread_arm_set_tpidruro(l4_cap_idx_t thread, l4_addr_t tpidruro) L4_NOTHROW;
00029
00034 L4_INLINE l4_msgtag_t
00035 l4_thread_arm_set_tpidruro_u(l4_cap_idx_t thread, l4_addr_t tpidruro,
00036                             l4_utcb_t *utcb) L4_NOTHROW;
00037
00046 enum L4_thread_ex_regs_flags_arm64
00047 {
00049     L4_THREAD_EX_REGS_ARM64_SET_EL_MASK    = 0x3 « 24,
00051     L4_THREAD_EX_REGS_ARM64_SET_EL_KEEP    = 0x0 « 24,
00053     L4_THREAD_EX_REGS_ARM64_SET_EL_EL0     = 0x1 « 24,
00055     L4_THREAD_EX_REGS_ARM64_SET_EL_EL1     = 0x2 « 24,
00056 };
00057
00058 /* IMPLEMENTATION ----- */
00059
00060 L4_INLINE l4_msgtag_t
00061 l4_thread_arm_set_tpidruro_u(l4_cap_idx_t thread, l4_addr_t tpidruro,
00062                             l4_utcb_t *utcb) L4_NOTHROW
00063 {
00064     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00065     v->mr[0] = L4_THREAD_ARM_TPIDRURO_OP;
00066     v->mr[1] = tpidruro;
00067     return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0),
00068                       L4_IPC_NEVER);
00069 }
00070
00071 L4_INLINE l4_msgtag_t
00072 l4_thread_arm_set_tpidruro(l4_cap_idx_t thread, l4_addr_t tpidruro) L4_NOTHROW
00073 {
00074     return l4_thread_arm_set_tpidruro_u(thread, tpidruro, l4_utcb());
00075 }

```

## 17.670 l4/sys/thread.h File Reference

Common thread related definitions.

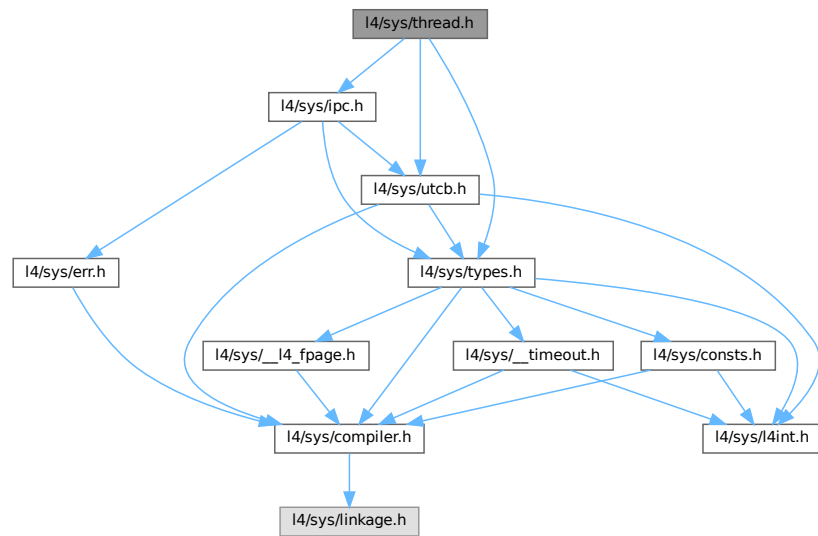
```

#include <l4/sys/types.h>
#include <l4/sys/utcb.h>

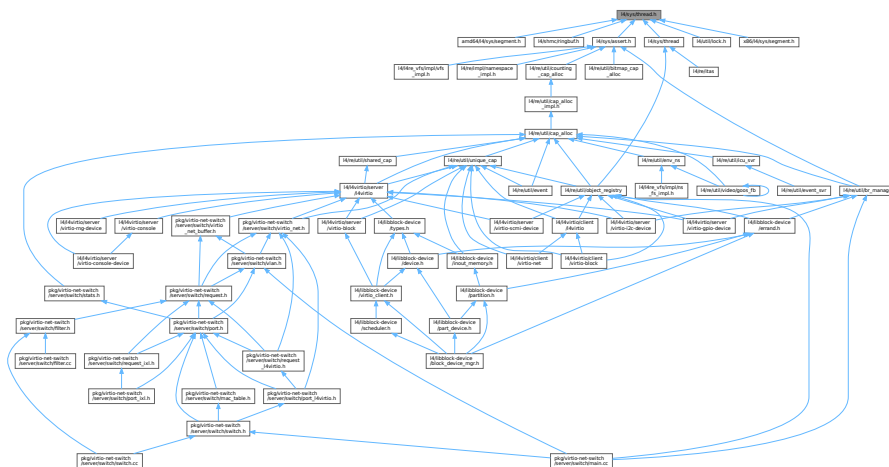
```

```
#include <l4/sys/ipc.h>
```

Include dependency graph for thread.h:



This graph shows which files directly or indirectly include this file:



## Enumerations

- enum [L4\\_thread\\_ops](#) {
  - [L4\\_THREAD\\_CONTROL\\_OP](#) = 0UL , [L4\\_THREAD\\_EX\\_REGS\\_OP](#) = 1UL , [L4\\_THREAD\\_SWITCH\\_OP](#) = 2UL , [L4\\_THREAD\\_STATS\\_OP](#) = 3UL ,
  - [L4\\_THREAD\\_VCPU\\_RESUME\\_OP](#) = 4UL , [L4\\_THREAD\\_REGISTER\\_DELETE\\_IRQ\\_OP](#) = 5UL ,
  - [L4\\_THREAD\\_MODIFY\\_SENDER\\_OP](#) = 6UL , [L4\\_THREAD\\_VCPU\\_CONTROL\\_OP](#) = 7UL ,
  - [L4\\_THREAD\\_VCPU\\_CONTROL\\_EXT\\_OP](#) = [L4\\_THREAD\\_VCPU\\_CONTROL\\_OP](#) | 0x10000 , [L4\\_THREAD\\_REGISTER\\_DO](#) = 8UL , [L4\\_THREAD\\_X86\\_GDT\\_OP](#) = 0x10UL , [L4\\_THREAD\\_ARM\\_TPIDRURO\\_OP](#) = 0x10UL ,
  - [L4\\_THREAD\\_AMD64\\_SET\\_SEGMENT\\_BASE\\_OP](#) = 0x12UL , [L4\\_THREAD\\_AMD64\\_GET\\_SEGMENT\\_INFO\\_OP](#) = 0x13UL , [L4\\_THREAD\\_OPCODE\\_MASK](#) = 0xffff }

*Operations on thread objects.*

- enum `L4_thread_control_flags` { `L4_THREAD_CONTROL_SET_PAGER` = 0x0010000 , `L4_THREAD_CONTROL_BIND_TASK` = 0x0200000 , `L4_THREAD_CONTROL_ALIEN` = 0x0400000 , `L4_THREAD_CONTROL_SET_EXC_HANDLER` = 0x1000000 }

*Flags for the thread control operation.*

- enum `L4_thread_control_mr_indices` {  
`L4_THREAD_CONTROL_MR_IDX_FLAGS` = 0 , `L4_THREAD_CONTROL_MR_IDX_PAGER` = 1 ,  
`L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER` = 2 , `L4_THREAD_CONTROL_MR_IDX_FLAG_VALS`  
= 4 ,  
`L4_THREAD_CONTROL_MR_IDX_BIND_UTCB` = 5 , `L4_THREAD_CONTROL_MR_IDX_BIND_TASK` = 6  
}

*Indices for the values in the message register for thread control.*

- enum `L4_thread_ex_regs_flags` { `L4_THREAD_EX_REGS_CANCEL` = 0x10000UL , `L4_THREAD_EX_REGS_TRIGGER_EXC` = 0x20000UL , `L4_THREAD_EX_REGS_ARCH_MASK` = 0xff000000UL }

*Flags for the thread ex-regs operation.*

## Functions

- `l4_msgtag_t l4_thread_ex_regs` (`l4_cap_idx_t` thread, `l4_addr_t` ip, `l4_addr_t` sp, `l4_umword_t` flags) `L4_NOTHROW`

*Exchange basic thread registers.*

- `l4_msgtag_t l4_thread_ex_regs_u` (`l4_cap_idx_t` thread, `l4_addr_t` ip, `l4_addr_t` sp, `l4_umword_t` flags, `l4_utcb_t` \*utcb) `L4_NOTHROW`

*Exchange basic thread registers.*

- `l4_msgtag_t l4_thread_ex_regs_ret` (`l4_cap_idx_t` thread, `l4_addr_t` \*ip, `l4_addr_t` \*sp, `l4_umword_t` \*flags) `L4_NOTHROW`

*Exchange basic thread registers and return previous values.*

- `l4_msgtag_t l4_thread_ex_regs_ret_u` (`l4_cap_idx_t` thread, `l4_addr_t` \*ip, `l4_addr_t` \*sp, `l4_umword_t` \*flags, `l4_utcb_t` \*utcb) `L4_NOTHROW`

*Exchange basic thread registers and return previous values.*

- void `l4_thread_control_start` (void) `L4_NOTHROW`

*Start a thread control API sequence.*

- void `l4_thread_control_pager` (`l4_cap_idx_t` pager) `L4_NOTHROW`

*Set the pager.*

- void `l4_thread_control_exc_handler` (`l4_cap_idx_t` exc\_handler) `L4_NOTHROW`

*Set the exception handler.*

- void `l4_thread_control_bind` (`l4_utcb_t` \*thread\_utcb, `l4_cap_idx_t` task) `L4_NOTHROW`

*Bind the thread to a task.*

- void `l4_thread_control_alien` (int on) `L4_NOTHROW`

*Enable alien mode.*

- `l4_msgtag_t l4_thread_control_commit` (`l4_cap_idx_t` thread) `L4_NOTHROW`

*Commit the thread control parameters.*

- `l4_msgtag_t l4_thread_yield` (void) `L4_NOTHROW`

*Yield current time slice.*

- `l4_msgtag_t l4_thread_switch` (`l4_cap_idx_t` to\_thread) `L4_NOTHROW`

*Switch to another thread (and donate the remaining time slice).*

- `l4_msgtag_t l4_thread_stats_time` (`l4_cap_idx_t` thread, `l4_kernel_clock_t` \*us) `L4_NOTHROW`

*Get consumed time of thread in  $\mu$ s.*

- `l4_msgtag_t l4_thread_vcpu_resume_start` (void) `L4_NOTHROW`

*vCPU return from event handler.*

- `l4_msgtag_t l4_thread_vcpu_resume_commit` (`l4_cap_idx_t` thread, `l4_msgtag_t` tag) `L4_NOTHROW`

*Commit vCPU resume.*



- [l4\\_msgtag\\_t l4\\_thread\\_vcpu\\_control \(l4\\_cap\\_idx\\_t thread, l4\\_addr\\_t vcpu\\_state\) L4\\_NOTHROW](#)  
*Enable the vCPU feature for the thread.*
- [l4\\_msgtag\\_t l4\\_thread\\_vcpu\\_control\\_u \(l4\\_cap\\_idx\\_t thread, l4\\_addr\\_t vcpu\\_state, l4\\_utcb\\_t \\*utcb\) L4\\_NOTHROW](#)  
*Enable the vCPU feature for the thread.*
- [l4\\_msgtag\\_t l4\\_thread\\_vcpu\\_control\\_ext \(l4\\_cap\\_idx\\_t thread, l4\\_addr\\_t ext\\_vcpu\\_state\) L4\\_NOTHROW](#)  
*Enable the extended vCPU feature for the thread.*
- [l4\\_msgtag\\_t l4\\_thread\\_vcpu\\_control\\_ext\\_u \(l4\\_cap\\_idx\\_t thread, l4\\_addr\\_t ext\\_vcpu\\_state, l4\\_utcb\\_t \\*utcb\) L4\\_NOTHROW](#)  
*Enable the extended vCPU feature for the thread.*
- [l4\\_msgtag\\_t l4\\_thread\\_register\\_del\\_irq \(l4\\_cap\\_idx\\_t thread, l4\\_cap\\_idx\\_t irq\) L4\\_NOTHROW](#)  
*Register an IRQ that will trigger upon deletion events.*
- [l4\\_msgtag\\_t l4\\_thread\\_modify\\_sender\\_start \(void\) L4\\_NOTHROW](#)  
*Start a thread sender modification sequence.*
- [int l4\\_thread\\_modify\\_sender\\_add \(l4\\_umword\\_t match\\_mask, l4\\_umword\\_t match, l4\\_umword\\_t del\\_bits, l4\\_umword\\_t add\\_bits, l4\\_msgtag\\_t \\*tag\) L4\\_NOTHROW](#)  
*Add a modification pattern to a sender modification sequence.*
- [l4\\_msgtag\\_t l4\\_thread\\_modify\\_sender\\_commit \(l4\\_cap\\_idx\\_t thread, l4\\_msgtag\\_t tag\) L4\\_NOTHROW](#)  
*Apply (commit) a sender modification sequence.*
- [l4\\_msgtag\\_t l4\\_thread\\_register\\_doorbell\\_irq \(l4\\_cap\\_idx\\_t thread, l4\\_cap\\_idx\\_t irq\) L4\\_NOTHROW](#)  
*Register an IRQ that will trigger when a forwarded virtual interrupt is pending.*

## 17.670.1 Detailed Description

Common thread related definitions.

Definition in file [thread.h](#).

## 17.671 thread.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *      Björn Döbel <doebel@os.inf.tu-dresden.de>,
00009  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #pragma once
00015
00016 #include <l4/sys/types.h>
00017 #include <l4/sys/utcb.h>
00018 #include <l4/sys/ipc.h>
00019
00054
00055
00083 L4_INLINE l4_msgtag_t
00084 l4_thread_ex_regs(l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp,
00085                  l4_umword_t flags) L4_NOTHROW;
00086
00093 L4_INLINE l4_msgtag_t
00094 l4_thread_ex_regs_u(l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp,
00095                    l4_umword_t flags, l4_utcb_t *utcb) L4_NOTHROW;
00096
00129 L4_INLINE l4_msgtag_t
00130 l4_thread_ex_regs_ret(l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp,
00131                      l4_umword_t *flags) L4_NOTHROW;
00132

```

```

00139 L4_INLINE l4_msgtag_t
00140 l4_thread_ex_regs_ret_u(l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp,
00141                        l4_umword_t *flags, l4_utcb_t *utcb) L4_NOTHROW;
00142
00143
00144
00170
00190 L4_INLINE void
00191 l4_thread_control_start(void) L4_NOTHROW;
00192
00197 L4_INLINE void
00198 l4_thread_control_start_u(l4_utcb_t *utcb) L4_NOTHROW;
00199
00209 L4_INLINE void
00210 l4_thread_control_pager(l4_cap_idx_t pager) L4_NOTHROW;
00211
00216 L4_INLINE void
00217 l4_thread_control_pager_u(l4_cap_idx_t pager, l4_utcb_t *utcb) L4_NOTHROW;
00218
00228 L4_INLINE void
00229 l4_thread_control_exc_handler(l4_cap_idx_t exc_handler) L4_NOTHROW;
00230
00235 L4_INLINE void
00236 l4_thread_control_exc_handler_u(l4_cap_idx_t exc_handler,
00237                                l4_utcb_t *utcb) L4_NOTHROW;
00238
00266 L4_INLINE void
00267 l4_thread_control_bind(l4_utcb_t *thread_utcb,
00268                       l4_cap_idx_t task) L4_NOTHROW;
00269
00274 L4_INLINE void
00275 l4_thread_control_bind_u(l4_utcb_t *thread_utcb,
00276                         l4_cap_idx_t task, l4_utcb_t *utcb) L4_NOTHROW;
00277
00301 L4_INLINE void
00302 l4_thread_control_alien(int on) L4_NOTHROW;
00303
00308 L4_INLINE void
00309 l4_thread_control_alien_u(l4_utcb_t *utcb, int on) L4_NOTHROW;
00310
00311
00312
00313
00330 L4_INLINE l4_msgtag_t
00331 l4_thread_control_commit(l4_cap_idx_t thread) L4_NOTHROW;
00332
00337 L4_INLINE l4_msgtag_t
00338 l4_thread_control_commit_u(l4_cap_idx_t thread, l4_utcb_t *utcb) L4_NOTHROW;
00339
00346 L4_INLINE l4_msgtag_t
00347 l4_thread_yield(void) L4_NOTHROW;
00348
00357 L4_INLINE l4_msgtag_t
00358 l4_thread_switch(l4_cap_idx_t to_thread) L4_NOTHROW;
00359
00364 L4_INLINE l4_msgtag_t
00365 l4_thread_switch_u(l4_cap_idx_t to_thread, l4_utcb_t *utcb) L4_NOTHROW;
00366
00367
00368
00378 L4_INLINE l4_msgtag_t
00379 l4_thread_stats_time(l4_cap_idx_t thread, l4_kernel_clock_t *us) L4_NOTHROW;
00380
00385 L4_INLINE l4_msgtag_t
00386 l4_thread_stats_time_u(l4_cap_idx_t thread, l4_kernel_clock_t *us,
00387                       l4_utcb_t *utcb) L4_NOTHROW;
00388
00389
00400 L4_INLINE l4_msgtag_t
00401 l4_thread_vcpu_resume_start(void) L4_NOTHROW;
00402
00407 L4_INLINE l4_msgtag_t
00408 l4_thread_vcpu_resume_start_u(l4_utcb_t *utcb) L4_NOTHROW;
00409
00457 L4_INLINE l4_msgtag_t
00458 l4_thread_vcpu_resume_commit(l4_cap_idx_t thread,
00459                             l4_msgtag_t tag) L4_NOTHROW;
00460
00465 L4_INLINE l4_msgtag_t
00466 l4_thread_vcpu_resume_commit_u(l4_cap_idx_t thread,
00467                                l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW;
00468
00469
00489 L4_INLINE l4_msgtag_t
00490 l4_thread_vcpu_control(l4_cap_idx_t thread, l4_addr_t vcpu_state) L4_NOTHROW;
00491
00499 L4_INLINE l4_msgtag_t

```

```

00500 l4_thread_vcpu_control_u(l4_cap_idx_t thread, l4_addr_t vcpu_state,
00501                          l4_utcb_t *utcb) L4_NOTHROW;
00502
00534 L4_INLINE l4_msgtag_t
00535 l4_thread_vcpu_control_ext(l4_cap_idx_t thread, l4_addr_t ext_vcpu_state) L4_NOTHROW;
00536
00544 L4_INLINE l4_msgtag_t
00545 l4_thread_vcpu_control_ext_u(l4_cap_idx_t thread, l4_addr_t ext_vcpu_state,
00546                             l4_utcb_t *utcb) L4_NOTHROW;
00547
00548
00572 L4_INLINE l4_msgtag_t
00573 l4_thread_register_del_irq(l4_cap_idx_t thread, l4_cap_idx_t irq) L4_NOTHROW;
00574
00579 L4_INLINE l4_msgtag_t
00580 l4_thread_register_del_irq_u(l4_cap_idx_t thread, l4_cap_idx_t irq,
00581                             l4_utcb_t *utcb) L4_NOTHROW;
00582
00604 L4_INLINE l4_msgtag_t
00605 l4_thread_modify_sender_start(void) L4_NOTHROW;
00606
00611 L4_INLINE l4_msgtag_t
00612 l4_thread_modify_sender_start_u(l4_utcb_t *u) L4_NOTHROW;
00613
00638 L4_INLINE int
00639 l4_thread_modify_sender_add(l4_umword_t match_mask,
00640                             l4_umword_t match,
00641                             l4_umword_t del_bits,
00642                             l4_umword_t add_bits,
00643                             l4_msgtag_t *tag) L4_NOTHROW;
00644
00649 L4_INLINE int
00650 l4_thread_modify_sender_add_u(l4_umword_t match_mask,
00651                               l4_umword_t match,
00652                               l4_umword_t del_bits,
00653                               l4_umword_t add_bits,
00654                               l4_msgtag_t *tag, l4_utcb_t *u) L4_NOTHROW;
00655
00681 L4_INLINE l4_msgtag_t
00682 l4_thread_modify_sender_commit(l4_cap_idx_t thread, l4_msgtag_t tag) L4_NOTHROW;
00683
00688 L4_INLINE l4_msgtag_t
00689 l4_thread_modify_sender_commit_u(l4_cap_idx_t thread, l4_msgtag_t tag,
00690                                  l4_utcb_t *u) L4_NOTHROW;
00691
00692
00713 L4_INLINE l4_msgtag_t
00714 l4_thread_register_doorbell_irq(l4_cap_idx_t thread,
00715                                 l4_cap_idx_t irq) L4_NOTHROW;
00716
00721 L4_INLINE l4_msgtag_t
00722 l4_thread_register_doorbell_irq_u(l4_cap_idx_t thread, l4_cap_idx_t irq,
00723                                   l4_utcb_t *u) L4_NOTHROW;
00724
00725
00732 enum L4_thread_ops
00733 {
00734     L4_THREAD_CONTROL_OP           = 0UL,
00735     L4_THREAD_EX_REGS_OP          = 1UL,
00736     L4_THREAD_SWITCH_OP           = 2UL,
00737     L4_THREAD_STATS_OP            = 3UL,
00738     L4_THREAD_VCPU_RESUME_OP       = 4UL,
00739     L4_THREAD_REGISTER_DELETE_IRQ_OP = 5UL,
00740     L4_THREAD_MODIFY_SENDER_OP     = 6UL,
00741     L4_THREAD_VCPU_CONTROL_OP      = 7UL,
00742     L4_THREAD_VCPU_CONTROL_EXT_OP  = L4_THREAD_VCPU_CONTROL_OP | 0x10000,
00743     L4_THREAD_REGISTER_DOORBELL_IRQ_OP = 8UL,
00744     L4_THREAD_X86_GDT_OP           = 0x10UL,
00745     L4_THREAD_ARM_TPIDRURO_OP      = 0x10UL,
00746     L4_THREAD_AMD64_SET_SEGMENT_BASE_OP = 0x12UL,
00747     L4_THREAD_AMD64_GET_SEGMENT_INFO_OP = 0x13UL,
00748     L4_THREAD_OPCODE_MASK         = 0xffff,
00749 };
00750
00761 enum L4_thread_control_flags
00762 {
00764     L4_THREAD_CONTROL_SET_PAGER      = 0x0010000,
00766     L4_THREAD_CONTROL_BIND_TASK      = 0x0200000,
00768     L4_THREAD_CONTROL_ALIEN          = 0x0400000,
00770     L4_THREAD_CONTROL_SET_EXC_HANDLER = 0x1000000,
00771 };
00772
00782 enum L4_thread_control_mr_indices
00783 {
00784     L4_THREAD_CONTROL_MR_IDX_FLAGS    = 0,
00785     L4_THREAD_CONTROL_MR_IDX_PAGER    = 1,
00786     L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER = 2,

```

```

00787 L4_THREAD_CONTROL_MR_IDX_FLAG_VALS = 4,
00788 L4_THREAD_CONTROL_MR_IDX_BIND_UTCB = 5,
00789 L4_THREAD_CONTROL_MR_IDX_BIND_TASK = 6,
00790 };
00791
00797 enum L4_thread_ex_regs_flags
00798 {
00799     L4_THREAD_EX_REGS_CANCEL = 0x10000UL,
00800     L4_THREAD_EX_REGS_TRIGGER_EXCEPTION = 0x20000UL,
00801
00802     L4_THREAD_EX_REGS_ARCH_MASK = 0xff000000UL,
00803 };
00804
00805
00806 /* IMPLEMENTATION ----- */
00807
00808 #include <l4/sys/ipc.h>
00809 #include <l4/sys/types.h>
00810
00811 L4_INLINE l4_msgtag_t
00812 l4_thread_ex_regs_u(l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp,
00813                    l4_umword_t flags, l4_utcb_t *utcb) L4_NOTHROW
00814 {
00815     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00816     v->mr[0] = L4_THREAD_EX_REGS_OP | flags;
00817     v->mr[1] = ip;
00818     v->mr[2] = sp;
00819     return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 3, 0, 0), L4_IPC_NEVER);
00820 }
00821
00822 L4_INLINE l4_msgtag_t
00823 l4_thread_ex_regs_ret_u(l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp,
00824                        l4_umword_t *flags, l4_utcb_t *utcb) L4_NOTHROW
00825 {
00826     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00827     l4_msgtag_t ret = l4_thread_ex_regs_u(thread, *ip, *sp, *flags, utcb);
00828     if (l4_error_u(ret, utcb))
00829         return ret;
00830
00831     *flags = v->mr[0];
00832     *ip = v->mr[1];
00833     *sp = v->mr[2];
00834     return ret;
00835 }
00836
00837 L4_INLINE void
00838 l4_thread_control_start_u(l4_utcb_t *utcb) L4_NOTHROW
00839 {
00840     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00841     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] = L4_THREAD_CONTROL_OP;
00842 }
00843
00844 L4_INLINE void
00845 l4_thread_control_pager_u(l4_cap_idx_t pager, l4_utcb_t *utcb) L4_NOTHROW
00846 {
00847     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00848     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |= L4_THREAD_CONTROL_SET_PAGER;
00849     v->mr[L4_THREAD_CONTROL_MR_IDX_PAGER] = pager;
00850 }
00851
00852 L4_INLINE void
00853 l4_thread_control_exc_handler_u(l4_cap_idx_t exc_handler,
00854                                l4_utcb_t *utcb) L4_NOTHROW
00855 {
00856     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00857     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |= L4_THREAD_CONTROL_SET_EXC_HANDLER;
00858     v->mr[L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER] = exc_handler;
00859 }
00860
00861 L4_INLINE void
00862 l4_thread_control_bind_u(l4_utcb_t *thread_utcb, l4_cap_idx_t task,
00863                         l4_utcb_t *utcb) L4_NOTHROW
00864 {
00865     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00866     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |= L4_THREAD_CONTROL_BIND_TASK;
00867     v->mr[L4_THREAD_CONTROL_MR_IDX_BIND_UTCB] = (l4_addr_t)thread_utcb;
00868     v->mr[L4_THREAD_CONTROL_MR_IDX_BIND_TASK] = L4_ITEM_MAP;
00869     v->mr[L4_THREAD_CONTROL_MR_IDX_BIND_TASK + 1] = l4_obj_fpage(task, 0, L4_CAP_FPAGE_RWS).raw;
00870 }
00871
00872 L4_INLINE void
00873 l4_thread_control_alien_u(l4_utcb_t *utcb, int on) L4_NOTHROW
00874 {
00875     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00876     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |= L4_THREAD_CONTROL_ALIEN;
00877     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAG_VALS] |= on ? L4_THREAD_CONTROL_ALIEN : 0;
00878 }

```

```

00879
00880 L4_INLINE l4_msgtag_t
00881 l4_thread_control_commit_u(l4_cap_idx_t thread, l4_utcb_t *utcb) L4_NOTHROW
00882 {
00883     int items = 0;
00884     if (l4_utcb_mr_u(utcb)->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] & L4_THREAD_CONTROL_BIND_TASK)
00885         items = 1;
00886     return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 6, items, 0), L4_IPC_NEVER);
00887 }
00888
00889
00890 L4_INLINE l4_msgtag_t
00891 l4_thread_yield(void) L4_NOTHROW
00892 {
00893     l4_ipc_receive(L4_INVALID_CAP, NULL, L4_IPC_BOTH_TIMEOUT_0);
00894     return l4_msgtag(0, 0, 0, 0);
00895 }
00896
00897 /* Preliminary, to be changed */
00898 L4_INLINE l4_msgtag_t
00899 l4_thread_switch_u(l4_cap_idx_t to_thread, l4_utcb_t *utcb) L4_NOTHROW
00900 {
00901     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00902     v->mr[0] = L4_THREAD_SWITCH_OP;
00903     return l4_ipc_call(to_thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER);
00904 }
00905
00906
00907 L4_INLINE l4_msgtag_t
00908 l4_thread_stats_time_u(l4_cap_idx_t thread, l4_kernel_clock_t *us,
00909                        l4_utcb_t *utcb) L4_NOTHROW
00910 {
00911     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00912     l4_msgtag_t res;
00913
00914     v->mr[0] = L4_THREAD_STATS_OP;
00915
00916     res = l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER);
00917
00918     if (l4_msgtag_has_error(res))
00919         return res;
00920
00921     *us = v->mr64[l4_utcb_mr64_idx(0)];
00922
00923     return res;
00924 }
00925
00926 L4_INLINE l4_msgtag_t
00927 l4_thread_vcpu_resume_start_u(l4_utcb_t *utcb) L4_NOTHROW
00928 {
00929     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00930     v->mr[0] = L4_THREAD_VCPU_RESUME_OP;
00931     return l4_msgtag(L4_PROTO_THREAD, 1, 0, 0);
00932 }
00933
00934 L4_INLINE l4_msgtag_t
00935 l4_thread_vcpu_resume_commit_u(l4_cap_idx_t thread,
00936                                l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW
00937 {
00938     return l4_ipc_call(thread, utcb, tag, L4_IPC_NEVER);
00939 }
00940
00941 L4_INLINE l4_msgtag_t
00942 l4_thread_ex_regs(l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp,
00943                   l4_umword_t flags) L4_NOTHROW
00944 {
00945     return l4_thread_ex_regs_u(thread, ip, sp, flags, l4_utcb());
00946 }
00947
00948 L4_INLINE l4_msgtag_t
00949 l4_thread_ex_regs_ret(l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp,
00950                       l4_umword_t *flags) L4_NOTHROW
00951 {
00952     return l4_thread_ex_regs_ret_u(thread, ip, sp, flags, l4_utcb());
00953 }
00954
00955 L4_INLINE void
00956 l4_thread_control_start(void) L4_NOTHROW
00957 {
00958     l4_thread_control_start_u(l4_utcb());
00959 }
00960
00961 L4_INLINE void
00962 l4_thread_control_pager(l4_cap_idx_t pager) L4_NOTHROW
00963 {
00964     l4_thread_control_pager_u(pager, l4_utcb());
00965 }

```

```

00966
00967 L4_INLINE void
00968 l4_thread_control_exc_handler(l4_cap_idx_t exc_handler) L4_NOTHROW
00969 {
00970     l4_thread_control_exc_handler_u(exc_handler, l4_utcb());
00971 }
00972
00973
00974 L4_INLINE void
00975 l4_thread_control_bind(l4_utcb_t *thread_utcb, l4_cap_idx_t task) L4_NOTHROW
00976 {
00977     l4_thread_control_bind_u(thread_utcb, task, l4_utcb());
00978 }
00979
00980 L4_INLINE void
00981 l4_thread_control_alien(int on) L4_NOTHROW
00982 {
00983     l4_thread_control_alien_u(l4_utcb(), on);
00984 }
00985
00986 L4_INLINE l4_msgtag_t
00987 l4_thread_control_commit(l4_cap_idx_t thread) L4_NOTHROW
00988 {
00989     return l4_thread_control_commit_u(thread, l4_utcb());
00990 }
00991
00992
00993
00994
00995 L4_INLINE l4_msgtag_t
00996 l4_thread_switch(l4_cap_idx_t to_thread) L4_NOTHROW
00997 {
00998     return l4_thread_switch_u(to_thread, l4_utcb());
00999 }
01000
01001
01002
01003
01004 L4_INLINE l4_msgtag_t
01005 l4_thread_stats_time(l4_cap_idx_t thread, l4_kernel_clock_t *us) L4_NOTHROW
01006 {
01007     return l4_thread_stats_time_u(thread, us, l4_utcb());
01008 }
01009
01010 L4_INLINE l4_msgtag_t
01011 l4_thread_vcpu_resume_start(void) L4_NOTHROW
01012 {
01013     return l4_thread_vcpu_resume_start_u(l4_utcb());
01014 }
01015
01016 L4_INLINE l4_msgtag_t
01017 l4_thread_vcpu_resume_commit(l4_cap_idx_t thread,
01018                               l4_msgtag_t tag) L4_NOTHROW
01019 {
01020     return l4_thread_vcpu_resume_commit_u(thread, tag, l4_utcb());
01021 }
01022
01023
01024 L4_INLINE l4_msgtag_t
01025 l4_thread_register_del_irq_u(l4_cap_idx_t thread, l4_cap_idx_t irq,
01026                               l4_utcb_t *u) L4_NOTHROW
01027 {
01028     l4_msg_regs_t *m = l4_utcb_mr_u(u);
01029     m->mr[0] = L4_THREAD_REGISTER_DELETE_IRQ_OP;
01030     m->mr[1] = l4_map_obj_control(0,0);
01031     m->mr[2] = l4_obj_fpage(irq, 0, L4_CAP_FPAGE_RWS).raw;
01032     return l4_ipc_call(thread, u, l4_msgtag(L4_PROTO_THREAD, 1, 1, 0), L4_IPC_NEVER);
01033 }
01034
01035
01036 L4_INLINE l4_msgtag_t
01037 l4_thread_register_del_irq(l4_cap_idx_t thread, l4_cap_idx_t irq) L4_NOTHROW
01038 {
01039     return l4_thread_register_del_irq_u(thread, irq, l4_utcb());
01040 }
01041
01042
01043 L4_INLINE l4_msgtag_t
01044 l4_thread_vcpu_control_u(l4_cap_idx_t thread, l4_addr_t vcpu_state,
01045                           l4_utcb_t *utcb) L4_NOTHROW
01046 {
01047     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
01048     v->mr[0] = L4_THREAD_VCPU_CONTROL_OP;
01049     v->mr[1] = vcpu_state;
01050     return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0), L4_IPC_NEVER);
01051 }
01052

```

```

01053 L4_INLINE l4_msgtag_t
01054 l4_thread_vcpu_control(l4_cap_idx_t thread, l4_addr_t vcpu_state) L4_NOTHROW
01055 { return l4_thread_vcpu_control_u(thread, vcpu_state, l4_utcb()); }
01056
01057
01058 L4_INLINE l4_msgtag_t
01059 l4_thread_vcpu_control_ext_u(l4_cap_idx_t thread, l4_addr_t ext_vcpu_state,
01060                             l4_utcb_t *utcb) L4_NOTHROW
01061 {
01062     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
01063     v->mr[0] = L4_THREAD_VCPU_CONTROL_EXT_OP;
01064     v->mr[1] = ext_vcpu_state;
01065     return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0), L4_IPC_NEVER);
01066 }
01067
01068 L4_INLINE l4_msgtag_t
01069 l4_thread_vcpu_control_ext(l4_cap_idx_t thread, l4_addr_t ext_vcpu_state) L4_NOTHROW
01070 { return l4_thread_vcpu_control_ext_u(thread, ext_vcpu_state, l4_utcb()); }
01071
01072 L4_INLINE l4_msgtag_t
01073 l4_thread_modify_sender_start_u(l4_utcb_t *u) L4_NOTHROW
01074 {
01075     l4_msg_regs_t *m = l4_utcb_mr_u(u);
01076     m->mr[0] = L4_THREAD_MODIFY_SENDER_OP;
01077     return l4_msgtag(L4_PROTO_THREAD, 1, 0, 0);
01078 }
01079
01080 L4_INLINE int
01081 l4_thread_modify_sender_add_u(l4_umword_t match_mask,
01082                               l4_umword_t match,
01083                               l4_umword_t del_bits,
01084                               l4_umword_t add_bits,
01085                               l4_msgtag_t *tag, l4_utcb_t *u) L4_NOTHROW
01086 {
01087     l4_msg_regs_t *m = l4_utcb_mr_u(u);
01088     unsigned w = l4_msgtag_words(*tag);
01089     if (w >= L4_UTCB_GENERIC_DATA_SIZE - 4)
01090         return -L4_ENOMEM;
01091
01092     m->mr[w] = match_mask;
01093     m->mr[w+1] = match;
01094     m->mr[w+2] = del_bits;
01095     m->mr[w+3] = add_bits;
01096
01097     *tag = l4_msgtag(l4_msgtag_label(*tag), w + 4, 0, 0);
01098
01099     return 0;
01100 }
01101
01102 L4_INLINE l4_msgtag_t
01103 l4_thread_modify_sender_commit_u(l4_cap_idx_t thread, l4_msgtag_t tag,
01104                                 l4_utcb_t *u) L4_NOTHROW
01105 {
01106     return l4_ipc_call(thread, u, tag, L4_IPC_NEVER);
01107 }
01108
01109 L4_INLINE l4_msgtag_t
01110 l4_thread_modify_sender_start(void) L4_NOTHROW
01111 {
01112     return l4_thread_modify_sender_start_u(l4_utcb());
01113 }
01114
01115 L4_INLINE int
01116 l4_thread_modify_sender_add(l4_umword_t match_mask,
01117                             l4_umword_t match,
01118                             l4_umword_t del_bits,
01119                             l4_umword_t add_bits,
01120                             l4_msgtag_t *tag) L4_NOTHROW
01121 {
01122     return l4_thread_modify_sender_add_u(match_mask, match,
01123                                           del_bits, add_bits, tag, l4_utcb());
01124 }
01125
01126 L4_INLINE l4_msgtag_t
01127 l4_thread_modify_sender_commit(l4_cap_idx_t thread, l4_msgtag_t tag) L4_NOTHROW
01128 {
01129     return l4_thread_modify_sender_commit_u(thread, tag, l4_utcb());
01130 }
01131
01132
01133 L4_INLINE l4_msgtag_t
01134 l4_thread_register_doorbell_irq_u(l4_cap_idx_t thread, l4_cap_idx_t irq,
01135                                   l4_utcb_t *u) L4_NOTHROW
01136 {
01137     l4_msg_regs_t *m = l4_utcb_mr_u(u);
01138     m->mr[0] = L4_THREAD_REGISTER_DOORBELL_IRQ_OP;
01139     m->mr[1] = l4_map_obj_control(0, 0);

```

```

01140  m->mr[2] = l4_obj_fpage(irq, 0, L4_CAP_FPAGE_RWS).raw;
01141  return l4_ipc_call(thread, u, l4_msgtag(L4_PROTO_THREAD, 1, 1, 0),
01142                                L4_IPC_NEVER);
01143 }
01144
01145 L4_INLINE l4_msgtag_t
01146 l4_thread_register_doorbell_irq(l4_cap_idx_t thread,
01147                                l4_cap_idx_t irq) L4_NOTHROW
01148 {
01149  return l4_thread_register_doorbell_irq_u(thread, irq, l4_utcb());
01150 }

```

## 17.672 l4/util/thread.h File Reference

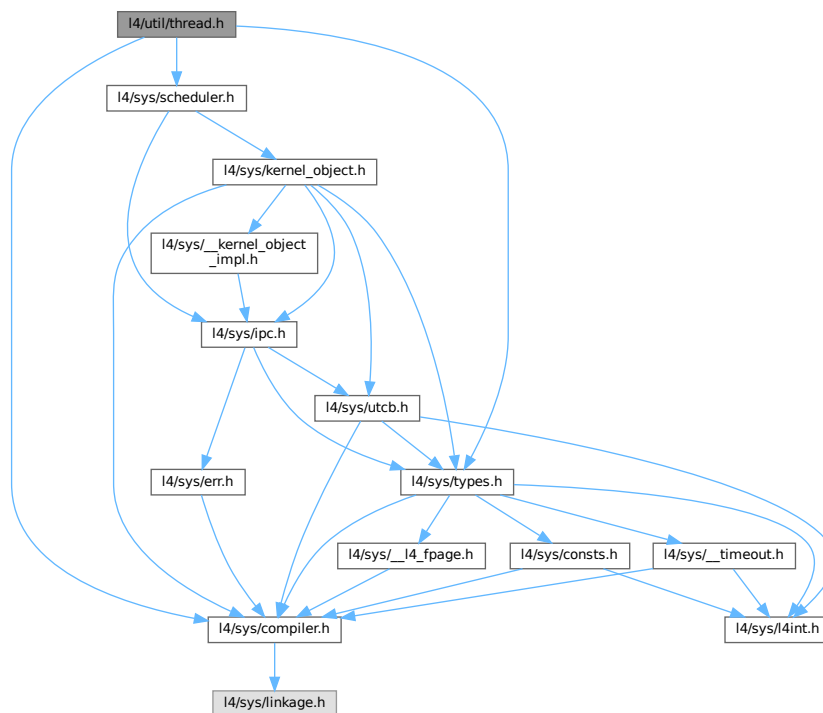
Low-level Thread Functions.

```

#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <l4/sys/scheduler.h>

```

Include dependency graph for thread.h:



### Macros

- `#define __L4UTIL_THREAD_FUNC(name)`  
*Defines a wrapper function that sets up the registers according to the calling conventions for the architecture.*



## 17.672.1 Detailed Description

Low-level Thread Functions.

Date

1997

Author

Sebastian Schönberg

Definition in file [thread.h](#).

## 17.672.2 Macro Definition Documentation

### 17.672.2.1 \_\_L4UTIL\_THREAD\_FUNC

```
#define __L4UTIL_THREAD_FUNC(  
    name)
```

**Value:**

```
void L4_NORETURN name(void)
```

Defines a wrapper function that sets up the registers according to the calling conventions for the architecture.

Use this as a function header when starting a low-level thread where only stack and instruction pointer are in a well-defined state.

Example:

```
L4UTIL_THREAD_FUNC(helper_thread) { l4_infinite_loop(); }
```

```
thread_cap->ex_regs((l4_umword_t)helper_thread, stack_addr);
```

Definition at line 71 of file [thread.h](#).

## 17.673 thread.h

[Go to the documentation of this file.](#)

```

00001
00007
00008 /*
00009  * (c) 2003-2009 Author(s)
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013
00014 #ifndef __L4_THREAD_H
00015 #define __L4_THREAD_H
00016
00017 #include <l4/sys/compiler.h>
00018 #include <l4/sys/types.h>
00019 #include <l4/sys/scheduler.h>
00020
00021 L4_BEGIN_DECLS
00022
00027
00045 L4_CV long
00046 l4util_create_thread(l4_cap_idx_t id, l4_utcb_t *thread_utcb,
00047                     l4_cap_idx_t factory,
00048                     l4_umword_t pc, l4_umword_t sp, l4_cap_idx_t pager,
00049                     l4_cap_idx_t task,
00050                     l4_cap_idx_t scheduler, l4_sched_param_t scp) L4_NOTHROW;
00051
00052 L4_END_DECLS
00053
00054 #ifndef L4UTIL_THREAD_FUNC
00071 #define __L4UTIL_THREAD_FUNC(name) void L4_NORETURN name(void)
00072 #define L4UTIL_THREAD_FUNC(name) __L4UTIL_THREAD_FUNC(name)
00073 #define __L4UTIL_THREAD_STATIC_FUNC(name) static L4_NORETURN void name(void)
00074 #define L4UTIL_THREAD_STATIC_FUNC(name) __L4UTIL_THREAD_STATIC_FUNC(name)
00075 #endif
00076
00077 #endif /* __L4_THREAD_H */

```

## 17.674 util.h

```

00001
00004 /*
00005  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00006  *     Alexander Warg <warg@os.inf.tu-dresden.de>,
00007  *     Frank Mehnert <fm3@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #ifndef __L4UTIL_UTIL_H__
00012 #define __L4UTIL_UTIL_H__
00013
00014 #include <l4/sys/types.h>
00015 #include <l4/sys/compiler.h>
00016 #include <l4/sys/ipc.h>
00017
00021
00022 L4_BEGIN_DECLS
00023
00034 L4_CV l4_timeout_s l4util_micros2l4to(l4_uint64_t us) L4_NOTHROW;
00035
00041 L4_CV void l4_sleep(l4_uint32_t ms) L4_NOTHROW;
00042
00049 L4_CV void l4_usleep(l4_uint64_t us) L4_NOTHROW;
00050
00056 L4_INLINE void l4_sleep_forever(void) L4_NOTHROW L4_NORETURN;
00057
00065 L4_INLINE void
00066 l4_touch_ro(const void *addr, unsigned size) L4_NOTHROW;
00067
00075 L4_INLINE void
00076 l4_touch_rw(const void *addr, unsigned size) L4_NOTHROW;
00077
00078
00079
00080 /*
00081  * Implementations
00082  */
00083
00084 L4_INLINE void

```

```

00085 l4_sleep_forever(void) L4_NOTHROW
00086 {
00087     for (;;)
00088         l4_ipc_sleep(L4_IPC_NEVER);
00089 }
00090
00091 L4_INLINE void
00092 l4_touch_ro(const void *addr, unsigned size) L4_NOTHROW
00093 {
00094     l4_addr_t b, e;
00095
00096     b = l4_trunc_page((l4_addr_t)addr);
00097     e = l4_trunc_page((l4_addr_t)addr + size - 1);
00098
00099     for (; b <= e; b += L4_PAGESIZE)
00100         (void) (*(volatile char *)b);
00101 }
00102
00103
00104 L4_INLINE void
00105 l4_touch_rw(const void *addr, unsigned size) L4_NOTHROW
00106 {
00107     l4_addr_t b, e;
00108
00109     b = l4_trunc_page((l4_addr_t)addr);
00110     e = l4_trunc_page((l4_addr_t)addr + size - 1);
00111
00112     for (; b <= e; b += L4_PAGESIZE)
00113         *(volatile char *)b |= 0;
00114 }
00115
00116 L4_END_DECLS
00117
00118 #endif /* __L4UTIL__UTIL_H__ */

```

## 17.675 vbus

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2011 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <l4/vbus/vbus.h>
00011 #include <l4/vbus/vbus_pm.h>
00012 #include <l4/sys/icu>
00013
00014 #include <l4/re/dataspace>
00015 #include <l4/re/dma_space>
00016 #include <l4/re/event>
00017 #include <l4/re/inhibitor>
00018
00036
00040 namespace L4vbus {
00041
00042     class Vbus;
00043
00049     template<typename DEC>
00050     class Pm
00051     {
00052     private:
00053         DEC const *self() const { return static_cast<DEC const *>(this); }
00054         DEC *self() { return static_cast<DEC *>(this); }
00055     public:
00063         int pm_suspend() const
00064         { return l4vbus_pm_suspend(self()->bus_cap().cap(), self()->dev_handle()); }
00065
00074         int pm_resume() const
00075         { return l4vbus_pm_resume(self()->bus_cap().cap(), self()->dev_handle()); }
00076     };
00077
00078
00083     class Device : public Pm<Device>
00084     {
00085     public:
00089         Device() : _dev(L4VBUS_NULL) {}
00090
00100         Device(L4::Cap<Vbus> bus, l4vbus_device_handle_t dev)
00101         : _bus(bus), _dev(dev) {}
00102

```

```

00107 L4::Cap<Vbus> bus_cap() const { return _bus; }
00108
00116 l4vbus_device_handle_t dev_handle() const { return _dev; }
00117
00118
00148 int device_by_hid(Device *child, char const *hid,
00149                 int depth = L4VBUS_MAX_DEPTH,
00150                 l4vbus_device_t *devinfo = 0) const
00151 {
00152     child->_bus = _bus;
00153     return l4vbus_get_device_by_hid(_bus.cap(), _dev, &child->_dev, hid,
00154                                     depth, devinfo);
00155 }
00156
00171 int next_device(Device *child, int depth = L4VBUS_MAX_DEPTH,
00172                l4vbus_device_t *devinfo = 0) const
00173 {
00174     child->_bus = _bus;
00175     return l4vbus_get_next_device(_bus.cap(), _dev, &child->_dev, depth,
00176                                   devinfo);
00177 }
00178
00189 int device(l4vbus_device_t *devinfo) const
00190 { return l4vbus_get_device(_bus.cap(), _dev, devinfo); }
00191
00209 int get_resource(unsigned res_idx, l4vbus_resource_t *res) const
00210 {
00211     return l4vbus_get_resource(_bus.cap(), _dev, res_idx, res);
00212 }
00213
00223 int is_compatible(char const *cid) const
00224 { return l4vbus_is_compatible(_bus.cap(), _dev, cid); }
00225
00230 bool operator == (Device const &o) const
00231 {
00232     return _bus == o._bus && _dev == o._dev;
00233 }
00234
00239 bool operator != (Device const &o) const
00240 {
00241     return _bus != o._bus || _dev != o._dev;
00242 }
00243
00244 protected:
00245     L4::Cap<Vbus> _bus;
00247     l4vbus_device_handle_t _dev;
00248 };
00249
00260 class Icu : public Device
00261 {
00262 public:
00264     enum Src_types
00265     {
00273         Src_dev_handle = L4VBUS_ICU_SRC_DEV_HANDLE
00274     };
00275
00285 int vicu(L4::Cap<L4::Icu> icu) const
00286 {
00287     return l4vbus_vicu_get_cap(_bus.cap(), _dev, icu.cap());
00288 }
00289 };
00290
00298 class Vbus : public L4::Kobject_3t<Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event>
00299 {
00300 public:
00301
00311 int request_ioport(l4vbus_resource_t *res) const
00312 {
00313     return l4vbus_request_ioport(cap(), res);
00314 }
00315
00323 int release_ioport(l4vbus_resource_t *res) const
00324 {
00325     return l4vbus_release_ioport(cap(), res);
00326 }
00327
00336 Device root() const
00337 {
00338     return Device(L4::Cap<Vbus>(cap()), L4VBUS_ROOT_BUS);
00339 }
00340
00357 int assign_dma_domain(unsigned domain_id, unsigned flags,
00358                       L4::Cap<L4Re::Dma_space> dma_space) const
00359 {
00360     flags |= L4VBUS_DMAD_L4RE_DMA_SPACE;
00361     flags &= ~L4VBUS_DMAD_KERNEL_DMA_SPACE;
00362     return l4vbus_assign_dma_domain(cap(), domain_id, flags, dma_space.cap());

```

```

00363     }
00364
00382 int assign_dma_domain(unsigned domain_id, unsigned flags,
00383                      L4::Cap<L4::Task> dma_space) const
00384 {
00385     flags |= L4VBUS_DMAD_KERNEL_DMA_SPACE;
00386     flags &= ~L4VBUS_DMAD_L4RE_DMA_SPACE;
00387     return l4vbus_assign_dma_domain(cap(), domain_id, flags, dma_space.cap());
00388 }
00389
00390 typedef L4::Typeid::Raw_ipc<Vbus> Rpcs;
00391 };
00392
00393 } // namespace L4vbus

```

## 17.676 l4/vbus/vbus.h File Reference

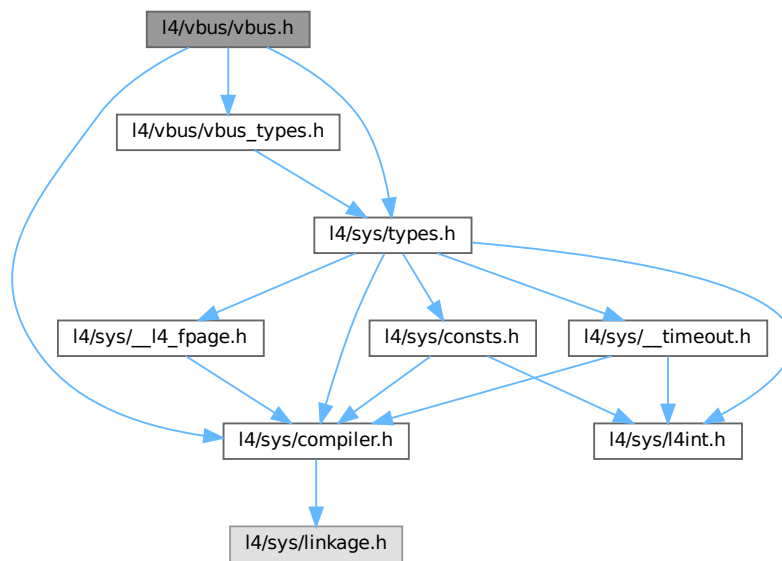
Description of the vbus C API.

```

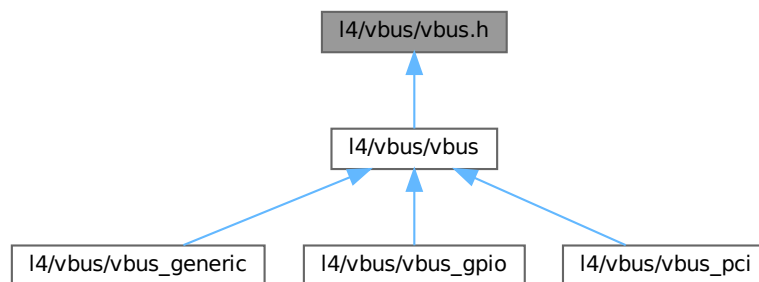
#include <l4/sys/compiler.h>
#include <l4/vbus/vbus_types.h>
#include <l4/sys/types.h>

```

Include dependency graph for vbus.h:



This graph shows which files directly or indirectly include this file:



## Enumerations

- enum { `L4VBUS_NULL` = -1L , `L4VBUS_ROOT_BUS` = 0 }
- Constants for device nodes.
- enum `l4vbus_icu_src_types` { `L4VBUS_ICU_SRC_DEV_HANDLE` = 1ULL << 63 }
- Flags that can be used with the ICU on a vbus device.
- enum `L4vbus_dma_domain_assign_flags` { `L4VBUS_DMAD_UNBIND` = 0 , `L4VBUS_DMAD_BIND` = 1 , `L4VBUS_DMAD_L4RE_DMA_SPACE` = 0 , `L4VBUS_DMAD_KERNEL_DMA_SPACE` = 2 }
- Flags for `l4vbus_assign_dma_domain()`.

## Functions

- `L4_BEGIN_DECLS` int `l4vbus_get_device_by_hid` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` parent, `l4vbus_device_handle_t` \*child, char const \*hid, int depth, `l4vbus_device_t` \*devinfo)
- Find a device by the hardware interface identifier (HID).
- int `l4vbus_get_next_device` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` parent, `l4vbus_device_handle_t` \*child, int depth, `l4vbus_device_t` \*devinfo)
- Find next child following *child*.
- int `l4vbus_get_device` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` dev, `l4vbus_device_t` \*devinfo)
- Obtain detailed information about a Vbus device.
- int `l4vbus_get_resource` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` dev, unsigned res\_idx, `l4vbus_resource_t` \*res)
- Obtain the resource description of an individual device resource.
- int `l4vbus_is_compatible` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` dev, char const \*cid)
- Check if the given device has a compatibility ID (CID) or HID that matches cid.
- int `l4vbus_get_hid` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` dev, char \*hid, unsigned long max\_len)
- Get the HID (hardware identifier) of a device.
- int `l4vbus_get_adr` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` dev, `l4_uint32_t` \*adr)
- Get the bus-specific address of a device.
- int `l4vbus_request_ioport` (`l4_cap_idx_t` vbus, `l4vbus_resource_t` const \*res)
- Request an IO port resource.
- int `l4vbus_assign_dma_domain` (`l4_cap_idx_t` vbus, unsigned domain\_id, unsigned flags, `l4_cap_idx_t` dma\_space)
- Bind or unbind a kernel DMA space or a `L4Re::Dma_space` to a DMA domain.
- int `l4vbus_release_ioport` (`l4_cap_idx_t` vbus, `l4vbus_resource_t` const \*res)
- Release a previously requested IO port resource.
- int `l4vbus_vicu_get_cap` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` icu, `l4_cap_idx_t` cap)
- Get capability of ICU.

## 17.676.1 Detailed Description

Description of the vbus C API.

Definition in file [vbus.h](#).

## 17.676.2 Enumeration Type Documentation

### 17.676.2.1 anonymous enum

anonymous enum

Constants for device nodes.

#### Enumerator

L4VBUS_NULL	NULL device.
L4VBUS_ROOT_BUS	Root device on the vbus.

Definition at line 20 of file [vbus.h](#).

### 17.676.2.2 l4vbus\_icu\_src\_types

enum [l4vbus\\_icu\\_src\\_types](#)

Flags that can be used with the ICU on a vbus device.

#### Enumerator

L4VBUS_ICU_SRC_DEV_HANDLE	Flag to denote that the value should be interpreted as a device handle. This flag may be used in the <code>source</code> parameter in <a href="#">l4_icu_msi_info()</a> to denote that the ICU should interpret the source ID as a device handle.
---------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Definition at line 26 of file [vbus.h](#).

## 17.677 vbus.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00013 #pragma once
00014
00015 #include <l4/sys/compiler.h>
00016 #include <l4/vbus/vbus_types.h>
00017 #include <l4/sys/types.h>
```

```

00018
00020 enum {
00021     L4VBUS_NULL = -1L,
00022     L4VBUS_ROOT_BUS = 0,
00023 };
00024
00026 enum l4vbus_icu_src_types {
00033     L4VBUS_ICU_SRC_DEV_HANDLE = 1ULL << 63
00034 };
00035
00053
00054 L4_BEGIN_DECLS
00055
00063 int L4_CV
00064 l4vbus_get_device_by_hid(l4_cap_idx_t vbus, l4vbus_device_handle_t parent,
00065                          l4vbus_device_handle_t *child, char const *hid,
00066                          int depth, l4vbus_device_t *devinfo);
00067
00083 int L4_CV
00084 l4vbus_get_next_device(l4_cap_idx_t vbus, l4vbus_device_handle_t parent,
00085                        l4vbus_device_handle_t *child, int depth,
00086                        l4vbus_device_t *devinfo);
00087
00101 int L4_CV
00102 l4vbus_get_device(l4_cap_idx_t vbus, l4vbus_device_handle_t dev,
00103                  l4vbus_device_t *devinfo);
00104
00113 int L4_CV
00114 l4vbus_get_resource(l4_cap_idx_t vbus, l4vbus_device_handle_t dev,
00115                     unsigned res_idx, l4vbus_resource_t *res);
00116
00117
00124 int L4_CV
00125 l4vbus_is_compatible(l4_cap_idx_t vbus, l4vbus_device_handle_t dev,
00126                      char const *cid);
00127
00138 int L4_CV
00139 l4vbus_get_hid(l4_cap_idx_t vbus, l4vbus_device_handle_t dev, char *hid,
00140                unsigned long max_len);
00141
00152 int L4_CV
00153 l4vbus_get_adr(l4_cap_idx_t vbus, l4vbus_device_handle_t dev, l4_uint32_t *adr);
00154
00168 int L4_CV
00169 l4vbus_request_ioport(l4_cap_idx_t vbus, l4vbus_resource_t const *res);
00170
00174 enum L4vbus_dma_domain_assign_flags
00175 {
00177     L4VBUS_DMAD_UNBIND = 0,
00179     L4VBUS_DMAD_BIND = 1,
00181     L4VBUS_DMAD_L4RE_DMA_SPACE = 0,
00183     L4VBUS_DMAD_KERNEL_DMA_SPACE = 2,
00184 };
00185
00207 int L4_CV
00208 l4vbus_assign_dma_domain(l4_cap_idx_t vbus, unsigned domain_id,
00209                          unsigned flags, l4_cap_idx_t dma_space);
00210
00219 int L4_CV
00220 l4vbus_release_ioport(l4_cap_idx_t vbus, l4vbus_resource_t const *res);
00221
00231 int L4_CV
00232 l4vbus_vicu_get_cap(l4_cap_idx_t vbus, l4vbus_device_handle_t icu,
00233                     l4_cap_idx_t cap);
00234
00235 L4_END_DECLS
00236

```

## 17.678 vbus\_generic

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 #include <l4/cxx/ipc_stream>
00013 #include <l4/vbus/vbus_types.h>

```



```

00014 #include <l4/vbus/vbus>
00015
00016 inline void
00017 l4vbus_device_msg(l4vbus_device_handle_t handle, l4_uint32_t op,
00018                  L4::Ipc::Iostream &s)
00019 {
00020     s « handle « op;
00021 }

```

## 17.679 vbus\_gpio

```

00001 // vi:set ft=c++: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2011 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <l4/vbus/vbus>
00011 #include <l4/vbus/vbus_gpio.h>
00012
00013 namespace L4vbus {
00014
00015     class Gpio_pin : public Device
00016     {
00017     public:
00018         Gpio_pin(Device const &dev, unsigned pin)
00019             : Device(dev), _pin(pin)
00020         {}
00021
00022         int get() const
00023         {
00024             return l4vbus_gpio_get(_bus.cap(), _dev, _pin);
00025         }
00026
00027         int set(int value) const
00028         {
00029             return l4vbus_gpio_set(_bus.cap(), _dev, _pin, value);
00030         }
00031
00032         int setup(unsigned mode, unsigned value) const
00033         {
00034             return l4vbus_gpio_setup(_bus.cap(), _dev, _pin, mode, value);
00035         }
00036
00037         int config_pull(unsigned mode) const
00038         {
00039             return l4vbus_gpio_config_pull(_bus.cap(), _dev, _pin, mode);
00040         }
00041
00042         int config_pad(unsigned func, unsigned value) const
00043         {
00044             return l4vbus_gpio_config_pad(_bus.cap(), _dev, _pin, func, value);
00045         }
00046
00047         int config_get(unsigned func, unsigned *value) const
00048         {
00049             return l4vbus_gpio_config_get(_bus.cap(), _dev, _pin, func, value);
00050         }
00051
00052         int to_irq() const
00053         {
00054             return l4vbus_gpio_to_irq(_bus.cap(), _dev, _pin);
00055         }
00056
00057         unsigned pin() const { return _pin; }
00058
00059     protected:
00060         Gpio_pin() {}
00061         unsigned _pin;
00062     };
00063
00064     class Gpio_module : public Device
00065     {
00066     public:
00067         Gpio_module(Device dev)
00068             : Device(dev)
00069         {}
00070
00071         struct Pin_slice

```

```

00147 {
00148     Pin_slice(unsigned offset, unsigned mask) : offset(offset), mask(mask) {}
00149     unsigned offset, mask;
00150 };
00151
00166 int setup(Pin_slice const &mask, unsigned mode, unsigned value) const
00167 {
00168     return l4vbus_gpio_multi_setup(_bus.cap(), _dev, mask.offset, mask.mask,
00169                                     mode, value);
00170 }
00171
00185 int config_pad(Pin_slice const &mask, unsigned func, unsigned value) const
00186 {
00187     return l4vbus_gpio_multi_config_pad(_bus.cap(), _dev, mask.offset,
00188                                         mask.mask, func, value);
00189 }
00190
00201 int get(unsigned offset, unsigned *data) const
00202 {
00203     return l4vbus_gpio_multi_get(_bus.cap(), _dev, offset, data);
00204 }
00205
00217 int set(Pin_slice const &mask, unsigned data)
00218 {
00219     return l4vbus_gpio_multi_set(_bus.cap(), _dev, mask.offset,
00220                                   mask.mask, data);
00221 }
00222
00229 Gpio_pin pin(unsigned pin) const
00230 {
00231     return Gpio_pin(*this, pin);
00232 }
00233
00234 protected:
00235     Gpio_module() {}
00236 };
00237
00238 }

```

## 17.680 vbus\_gpio-ops.h

```

00001 /*
00002  * (c) 2011 Alexander Warg <warg@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <l4/vbus/vbus_interfaces.h>
00011
00012 enum L4vbus_gpio_op
00013 {
00014     L4VBUS_GPIO_OP_SETUP = L4VBUS_INTERFACE_GPIO « L4VBUS_IFACE_SHIFT,
00015     L4VBUS_GPIO_OP_CONFIG_PAD,
00016     L4VBUS_GPIO_OP_CONFIG_GET,
00017     L4VBUS_GPIO_OP_GET,
00018     L4VBUS_GPIO_OP_SET,
00019     L4VBUS_GPIO_OP_MULTI_SETUP,
00020     L4VBUS_GPIO_OP_MULTI_CONFIG_PAD,
00021     L4VBUS_GPIO_OP_MULTI_GET,
00022     L4VBUS_GPIO_OP_MULTI_SET,
00023     L4VBUS_GPIO_OP_TO_IRQ,
00024     L4VBUS_GPIO_OP_CONFIG_PULL
00025 };

```

## 17.681 vbus\_gpio.h

```

00001 /*
00002  * (c) 2011 Alexander Warg <warg@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/sys/compiler.h>

```

```

00010 #include <l4/sys/types.h>
00011 #include <l4/vbus/vbus_types.h>
00012
00018
00019 L4_BEGIN_DECLS
00020
00024 enum L4vbus_gpio_generic_func
00025 {
00026     L4VBUS_GPIO_SETUP_INPUT    = 0x100,
00027     L4VBUS_GPIO_SETUP_OUTPUT   = 0x200,
00028     L4VBUS_GPIO_SETUP_IRQ      = 0x300,
00029 };
00030
00034 enum L4vbus_gpio_pull_modes
00035 {
00036     L4VBUS_GPIO_PIN_PULL_NONE  = 0x100,
00037     L4VBUS_GPIO_PIN_PULL_UP    = 0x200,
00038     L4VBUS_GPIO_PIN_PULL_DOWN  = 0x300,
00039 };
00040
00048 int L4_CV
00049 l4vbus_gpio_setup(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00050                  unsigned pin, unsigned mode, int value);
00051
00059 int L4_CV
00060 l4vbus_gpio_config_pull(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00061                        unsigned pin, unsigned mode);
00062
00070 int L4_CV
00071 l4vbus_gpio_config_pad(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00072                      unsigned pin, unsigned func, unsigned value);
00073
00081 int L4_CV
00082 l4vbus_gpio_config_get(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00083                      unsigned pin, unsigned func, unsigned *value);
00084
00092 int L4_CV
00093 l4vbus_gpio_get(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00094                unsigned pin);
00095
00103 int L4_CV
00104 l4vbus_gpio_set(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00105                unsigned pin, int value);
00106
00115 int L4_CV
00116 l4vbus_gpio_multi_setup(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00117                       unsigned offset, unsigned mask,
00118                       unsigned mode, unsigned value);
00119
00128 int L4_CV
00129 l4vbus_gpio_multi_config_pad(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00130                             unsigned offset, unsigned mask,
00131                             unsigned func, unsigned value);
00132
00139 int L4_CV
00140 l4vbus_gpio_multi_get(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00141                     unsigned offset, unsigned *data);
00142
00151 int L4_CV
00152 l4vbus_gpio_multi_set(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00153                     unsigned offset, unsigned mask, unsigned data);
00154
00162 int L4_CV
00163 l4vbus_gpio_to_irq(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00164                  unsigned pin);
00165
00167
00168 L4_END_DECLS

```

## 17.682 vbus\_i2c.h

```

00001 /*
00002  * (c) 2009 Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/sys/compiler.h>
00010 #include <l4/sys/types.h>
00011 #include <l4/vbus/vbus_types.h>
00012

```

```

00013 L4_BEGIN_DECLS
00014
00015 int L4_CV
00016 l4vbus_i2c_write(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00017                 l4_uint16_t addr, l4_uint8_t sub_addr,
00018                 l4_uint8_t *buffer, unsigned long size);
00019
00020 int L4_CV
00021 l4vbus_i2c_read(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00022                l4_uint16_t addr, l4_uint8_t sub_addr,
00023                l4_uint8_t *buffer, unsigned long *size);
00024
00025 L4_END_DECLS

```

## 17.683 vbus\_inhibitor.h

```

00001
00006 #pragma once
00007
00008 enum Vbus_inhibitor
00009 {
00010     L4VBUS_INHIBITOR_SUSPEND = 0,
00011     L4VBUS_INHIBITOR_SHUTDOWN = 1,
00012     L4VBUS_INHIBITOR_REBOOT = L4VBUS_INHIBITOR_SHUTDOWN,
00013     L4VBUS_INHIBITOR_WAKEUP = 2,
00014     L4VBUS_INHIBITOR_MAX
00015 };
00016

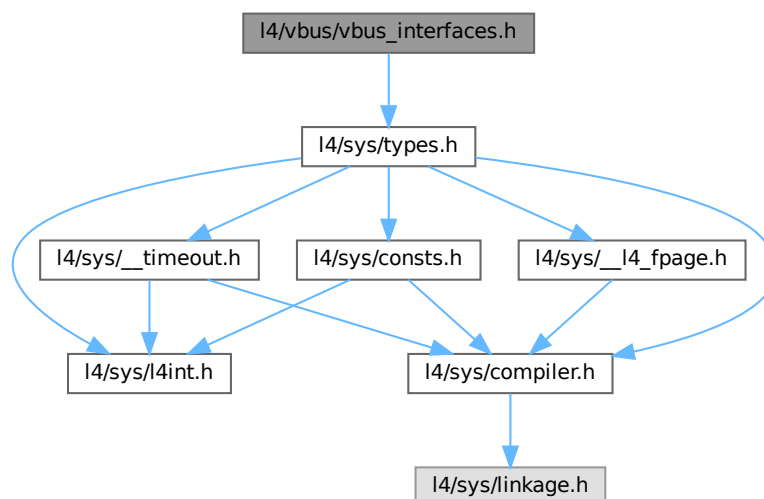
```

## 17.684 l4/vbus/vbus\_interfaces.h File Reference

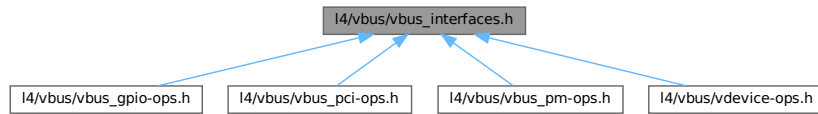
This header contains the definition of VBUS sub-interfaces and convenience functions to work with the interface IDs.

```
#include <l4/sys/types.h>
```

Include dependency graph for vbus\_interfaces.h:



This graph shows which files directly or indirectly include this file:



## Typedefs

- typedef enum [l4vbus\\_iface\\_type\\_t](#) [l4vbus\\_iface\\_type\\_t](#)  
*Different sub-interfaces a vbus device may support.*

## Enumerations

- enum [l4vbus\\_iface\\_type\\_t](#) {  
[L4VBUS\\_INTERFACE\\_ICU](#) = 0 , [L4VBUS\\_INTERFACE\\_GPIO](#) , [L4VBUS\\_INTERFACE\\_PCI](#) , [L4VBUS\\_INTERFACE\\_PCIDEV](#) ,  
[L4VBUS\\_INTERFACE\\_PM](#) , [L4VBUS\\_INTERFACE\\_BUS](#) , [L4VBUS\\_INTERFACE\\_GENERIC](#) = 0x20 }  
*Different sub-interfaces a vbus device may support.*
- enum { [L4VBUS\\_IFACE\\_SHIFT](#) = 26 }

## Functions

- unsigned [l4vbus\\_subinterface](#) (unsigned opcode)  
*Return the ID of the vbus sub-interface.*
- unsigned [l4vbus\\_interface\\_opcode](#) (unsigned opcode)  
*Return the function opcode within the sub-interface of the vbus command.*
- int [l4vbus\\_subinterface\\_supported](#) ([l4\\_uint32\\_t](#) dev\_type, [l4vbus\\_iface\\_type\\_t](#) iface\_type)  
*Check if a vbus device supports a given sub-interface.*

## 17.684.1 Detailed Description

This header contains the definition of VBUS sub-interfaces and convenience functions to work with the interface IDs.

Definition in file [vbus\\_interfaces.h](#).

## 17.684.2 Typedef Documentation

### 17.684.2.1 l4vbus\_iface\_type\_t

```
typedef enum l4vbus_iface_type_t l4vbus_iface_type_t
```

Different sub-interfaces a vbus device may support.

The IPC interface of vbus devices is divided into functional groups of sub-interfaces. Every device must implement the generic interface which provides general device information. According to the type of device, additional functionality may be supported.

The sub-interface constants are first of all used to divide the function opcode space of the interface into these functional groups (see [L4VBUS\\_IFACE\\_SHIFT](#)). They also make up a bitmask that specify the type of the device, i.e. from the point of view of the client a device is defined by the kinds of sub-interfaces it supports.

## 17.684.3 Enumeration Type Documentation

### 17.684.3.1 anonymous enum

anonymous enum

#### Enumerator

L4VBUS_IFACE_SHIFT	Sub-interface ID shift. Divides the function opcode sent via IPC into a sub-interface ID and the actual function opcode within the sub-interface.
--------------------	---------------------------------------------------------------------------------------------------------------------------------------------------

Definition at line 48 of file [vbus\\_interfaces.h](#).

### 17.684.3.2 l4vbus\_iface\_type\_t

enum [l4vbus\\_iface\\_type\\_t](#)

Different sub-interfaces a vbus device may support.

The IPC interface of vbus devices is divided into functional groups of sub-interfaces. Every device must implement the generic interface which provides general device information. According to the type of device, additional functionality may be supported.

The sub-interface constants are first of all used to divide the function opcode space of the interface into these functional groups (see L4VBUS\_IFACE\_SHIFT). They also make up a bitmask that specify the type of the device, i.e. from the point of view of the client a device is defined by the kinds of sub-interfaces it supports.

#### Enumerator

L4VBUS_INTERFACE_ICU	Interrupt Controller.
L4VBUS_INTERFACE_GPIO	GPIO.
L4VBUS_INTERFACE_PCI	PCI.
L4VBUS_INTERFACE_PCIDEV	PCI Device.
L4VBUS_INTERFACE_PM	Power Management.
L4VBUS_INTERFACE_BUS	VBus.
L4VBUS_INTERFACE_GENERIC	No specific sub interface.

Definition at line 29 of file [vbus\\_interfaces.h](#).

## 17.684.4 Function Documentation

### 17.684.4.1 l4vbus\_subinterface\_supported()

```
int l4vbus_subinterface_supported (
    l4_uint32_t dev_type,
    l4vbus_iface_type_t iface_type) [inline]
```

Check if a vbus device supports a given sub-interface.

#### Parameters

---

<i>dev_type</i>	Device type as reported in <a href="#">l4vbus_device_t</a> .
<i>iface_type</i>	Sub-interface type to check for.

## Returns

True if the device supports the sub-interface.

Definition at line 99 of file [vbus\\_interfaces.h](#).

References [L4\\_INLINE](#), and [L4VBUS\\_INTERFACE\\_GENERIC](#).

## 17.685 vbus\_interfaces.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * (c) 2014 Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00003  *
00004  * License: see LICENSE.spdx (in this directory or the directories above)
00005  */
00011 #pragma once
00012
00013 #include <l4/sys/types.h>
00014
00029 typedef enum l4vbus_iface_type_t
00030 {
00032     L4VBUS_INTERFACE_ICU = 0,
00034     L4VBUS_INTERFACE_GPIO,
00036     L4VBUS_INTERFACE_PCI,
00038     L4VBUS_INTERFACE_PCIDEV,
00040     L4VBUS_INTERFACE_PM,
00042     L4VBUS_INTERFACE_BUS,
00044     L4VBUS_INTERFACE_GENERIC = 0x20
00045 } l4vbus_iface_type_t;
00046
00047
00048 enum {
00056     L4VBUS_IFACE_SHIFT = 26
00057 };
00058
00070 L4_INLINE unsigned l4vbus_subinterface(unsigned opcode)
00071 {
00072     return opcode » L4VBUS_IFACE_SHIFT;
00073 }
00074
00086 L4_INLINE unsigned l4vbus_interface_opcode(unsigned opcode)
00087 {
00088     return opcode & ((1 « L4VBUS_IFACE_SHIFT) - 1);
00089 }
00090
00099 L4_INLINE int l4vbus_subinterface_supported(l4_uint32_t dev_type,
00100                                              l4vbus_iface_type_t iface_type)
00101 {
00102     if (iface_type == L4VBUS_INTERFACE_GENERIC)
00103         return 1;
00104
00105     return (dev_type & (1 « iface_type)) ? 1 : 0;
00106 }

```

## 17.686 vbus\_mcspi.h

```

00001 /*
00002  * (c) 2009 Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once

```

```

00008
00009 #include <l4/sys/compiler.h>
00010 #include <l4/sys/types.h>
00011 #include <l4/vbus/vbus_types.h>
00012
00013 L4_BEGIN_DECLS
00014
00015 int L4_CV
00016 l4vbus_mcspi_read(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00017                 unsigned channel, l4_umword_t *value);
00018
00019 int L4_CV
00020 l4vbus_mcspi_write(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00021                 unsigned channel, l4_umword_t value);
00022
00023 L4_END_DECLS

```

## 17.687 vbus\_pci

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <l4/vbus/vbus>
00011 #include <l4/vbus/vbus_pci.h>
00012
00013 namespace L4vbus {
00014
00015 class Pci_host_bridge : public Device
00016 {
00017 public:
00018     int cfg_read(l4_uint32_t bus, l4_uint32_t devfn, l4_uint32_t reg,
00019                 l4_uint32_t *value, l4_uint32_t width) const
00020     {
00021         return l4vbus_pci_cfg_read(bus_cap().cap(), _dev, bus,
00022                                     devfn, reg, value, width);
00023     }
00024
00025     int cfg_write(l4_uint32_t bus, l4_uint32_t devfn, l4_uint32_t reg,
00026                  l4_uint32_t value, l4_uint32_t width) const
00027     {
00028         return l4vbus_pci_cfg_write(bus_cap().cap(), _dev, bus,
00029                                     devfn, reg, value, width);
00030     }
00031
00032     int irq_enable(l4_uint32_t bus, l4_uint32_t devfn, int pin,
00033                   unsigned char *trigger, unsigned char *polarity) const
00034     {
00035         return l4vbus_pci_irq_enable(bus_cap().cap(), _dev, bus,
00036                                       devfn, pin, trigger, polarity);
00037     }
00038 };
00039
00040 class Pci_dev : public Device
00041 {
00042 public:
00043     int cfg_read(l4_uint32_t reg, l4_uint32_t *value,
00044                 l4_uint32_t width) const
00045     {
00046         return l4vbus_pcidev_cfg_read(bus_cap().cap(), _dev, reg, value, width);
00047     }
00048
00049     int cfg_write(l4_uint32_t reg, l4_uint32_t value,
00050                  l4_uint32_t width) const
00051     {
00052         return l4vbus_pcidev_cfg_write(bus_cap().cap(), _dev, reg, value, width);
00053     }
00054
00055     int irq_enable(unsigned char *trigger, unsigned char *polarity) const
00056     {
00057         return l4vbus_pcidev_irq_enable(bus_cap().cap(), _dev, trigger, polarity);
00058     }
00059 };

```



```

00140     }
00141
00142 };
00143
00144 }

```

## 17.688 vbus\_pci-ops.h

```

00001 /*
00002  * (c) 2014 Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00003  *
00004  * License: see LICENSE.spdx (in this directory or the directories above)
00005  */
00006 #pragma once
00007
00008 #include <14/vbus/vbus_interfaces.h>
00009
00010 enum
00011 {
00012     L4vbus_pciroot_cfg_read = L4VBUS_INTERFACE_PCI « L4VBUS_IFACE_SHIFT,
00013     L4vbus_pciroot_cfg_write,
00014     L4vbus_pciroot_cfg_irq_enable
00015 };
00016
00017 enum
00018 {
00019     L4vbus_pcidev_cfg_read = L4VBUS_INTERFACE_PCIDEV « L4VBUS_IFACE_SHIFT,
00020     L4vbus_pcidev_cfg_write,
00021     L4vbus_pcidev_cfg_irq_enable
00022 };

```

## 17.689 vbus\_pci.h

```

00001 /*
00002  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <14/sys/compiler.h>
00011 #include <14/vbus/vbus_types.h>
00012 #include <14/sys/types.h>
00013
00014
00015
00016
00017
00018
00019
00020
00021 L4_BEGIN_DECLS
00022
00023 int L4_CV
00030 l4vbus_pci_cfg_read(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00031                    l4_uint32_t bus, l4_uint32_t devfn,
00032                    l4_uint32_t reg, l4_uint32_t *value, l4_uint32_t width);
00033
00040 int L4_CV
00041 l4vbus_pci_cfg_write(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00042                     l4_uint32_t bus, l4_uint32_t devfn,
00043                     l4_uint32_t reg, l4_uint32_t value, l4_uint32_t width);
00044
00051 int L4_CV
00052 l4vbus_pci_irq_enable(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00053                      l4_uint32_t bus, l4_uint32_t devfn,
00054                      int pin, unsigned char *trigger,
00055                      unsigned char *polarity);
00056
00057
00064 int L4_CV
00065 l4vbus_pcidev_cfg_read(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00066                       l4_uint32_t reg, l4_uint32_t *value, l4_uint32_t width);
00067
00074 int L4_CV
00075 l4vbus_pcidev_cfg_write(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00076                        l4_uint32_t reg, l4_uint32_t value, l4_uint32_t width);
00077
00084 int L4_CV
00085 l4vbus_pcidev_irq_enable(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00086                          unsigned char *trigger,

```

```

00087             unsigned char *polarity);
00088
00089
00090
00092 L4_END_DECLS

```

## 17.690 vbus\_pm-ops.h

```

00001 /*
00002  * (c) 2013 Alexander Warg <warg@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include "vbus_interfaces.h"
00011
00012 enum L4vbus_pm_op
00013 {
00014     L4VBUS_PM_OP_SUSPEND = L4VBUS_INTERFACE_PM << L4VBUS_IFACE_SHIFT,
00015     L4VBUS_PM_OP_RESUME,
00016 };
00017

```

## 17.691 vbus\_pm.h

```

00001 /*
00002  * (c) 2013 Alexander Warg <warg@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/sys/compiler.h>
00010 #include <l4/vbus/vbus_types.h>
00011 #include <l4/sys/types.h>
00012
00018
00019 L4_BEGIN_DECLS
00020
00027 int L4_CV
00028 l4vbus_pm_suspend(l4_cap_idx_t vbus, l4vbus_device_handle_t handle);
00029
00036 int L4_CV
00037 l4vbus_pm_resume(l4_cap_idx_t vbus, l4vbus_device_handle_t handle);
00038
00040
00041 L4_END_DECLS

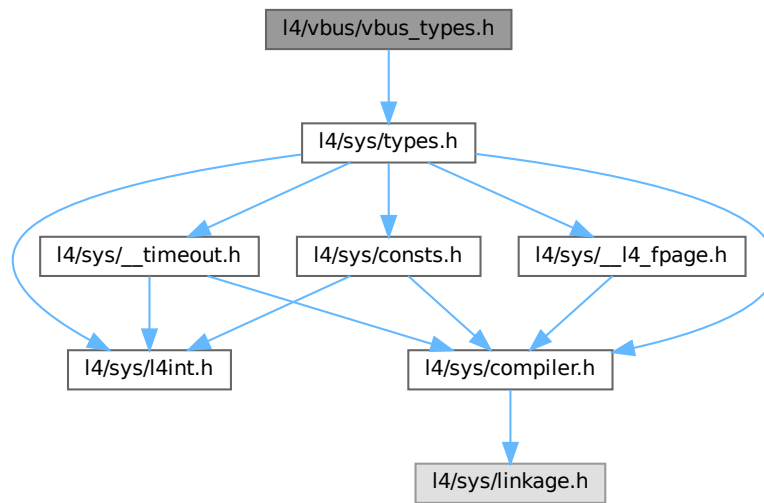
```

## 17.692 l4/vbus/vbus\_types.h File Reference

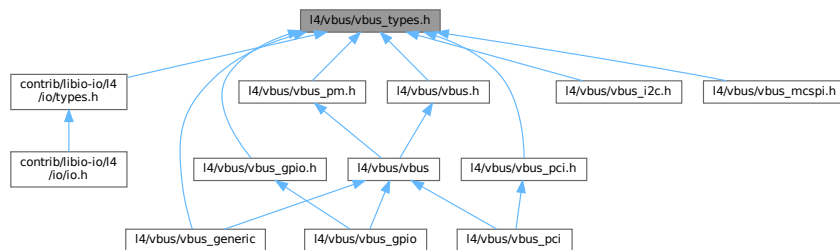
This header file contains descriptions of vbus related data types and constants.

```
#include <l4/sys/types.h>
```

Include dependency graph for vbus\_types.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [l4vbus\\_resource\\_t](#)  
*Description of a single vbus resource.*
- struct [l4vbus\\_device\\_t](#)  
*Detailed information about a vbus device.*

## Typedefs

- typedef [l4\\_mword\\_t](#) [l4vbus\\_device\\_handle\\_t](#)  
*Device handle for a device on the vbus.*
- typedef [l4\\_addr\\_t](#) [l4vbus\\_paddr\\_t](#)  
*Address of resources on the vbus.*

## Enumerations

- enum [l4vbus\\_resource\\_type\\_t](#) {  
[L4VBUS\\_RESOURCE\\_INVALID](#) = 0 , [L4VBUS\\_RESOURCE\\_IRQ](#) , [L4VBUS\\_RESOURCE\\_MEM](#) ,  
[L4VBUS\\_RESOURCE\\_PORT](#) ,  
[L4VBUS\\_RESOURCE\\_BUS](#) , [L4VBUS\\_RESOURCE\\_GPIO](#) , [L4VBUS\\_RESOURCE\\_DMA\\_DOMAIN](#) ,  
[L4VBUS\\_RESOURCE\\_MAX](#) }

*Description of vbus resource types.*

- enum [l4vbus\\_resource\\_flags\\_t](#) {  
[L4VBUS\\_RESOURCE\\_F\\_MEM\\_R](#) = 0x1 , [L4VBUS\\_RESOURCE\\_F\\_MEM\\_W](#) = 0x2 , [L4VBUS\\_RESOURCE\\_F\\_MEM\\_PREFE](#)  
= 0x10 , [L4VBUS\\_RESOURCE\\_F\\_MEM\\_CACHEABLE](#) = 0x20 ,  
[L4VBUS\\_RESOURCE\\_F\\_MEM\\_MMIO\\_READ](#) = 0x2000 , [L4VBUS\\_RESOURCE\\_F\\_MEM\\_MMIO\\_WRITE](#)  
= 0x4000 }

*Description of vbus resource flags.*

- enum [l4vbus\\_device\\_flags\\_t](#) { [L4VBUS\\_DEVICE\\_F\\_CHILDREN](#) = 0x10 }

*Flags describing device properties, see [l4vbus\\_device\\_t](#).*

## 17.692.1 Detailed Description

This header file contains descriptions of vbus related data types and constants.

Definition in file [vbus\\_types.h](#).

## 17.692.2 Enumeration Type Documentation

### 17.692.2.1 l4vbus\_device\_flags\_t

enum [l4vbus\\_device\\_flags\\_t](#)

Flags describing device properties, see [l4vbus\\_device\\_t](#).

#### Enumerator

<a href="#">L4VBUS_DEVICE_F_CHILDREN</a>	Device has child devices.
------------------------------------------	---------------------------

Definition at line 92 of file [vbus\\_types.h](#).

### 17.692.2.2 l4vbus\_resource\_flags\_t

enum [l4vbus\\_resource\\_flags\\_t](#)

Description of vbus resource flags.

#### Enumerator

<a href="#">L4VBUS_RESOURCE_F_MEM_R</a>	Memory resource is readable.
-----------------------------------------	------------------------------

L4VBUS_RESOURCE_F_MEM_W	Memory resource is writeable.
L4VBUS_RESOURCE_F_MEM_PREFETCHABLE	Memory resource is prefetchable. Clients may map it buffered or non-cached.
L4VBUS_RESOURCE_F_MEM_CACHEABLE	Memory resource is cacheable. This implies that the memory resource is prefetchable. If not set, clients must not map it cached. If the resource is neither cacheable nor prefetchable, clients must map it non-cached!
L4VBUS_RESOURCE_F_MEM_MMIO_READ	Reading needs to be performed using the MMIO space protocol.
L4VBUS_RESOURCE_F_MEM_MMIO_WRITE	Writing needs to be performed using the MMIO space protocol.

Definition at line 51 of file [vbus\\_types.h](#).

### 17.692.2.3 l4vbus\_resource\_type\_t

enum [l4vbus\\_resource\\_type\\_t](#)

Description of vbus resource types.

#### Enumerator

L4VBUS_RESOURCE_INVALID	Invalid type.
L4VBUS_RESOURCE_IRQ	Interrupt resource.
L4VBUS_RESOURCE_MEM	I/O memory resource.
L4VBUS_RESOURCE_PORT	I/O port resource (x86 only).
L4VBUS_RESOURCE_BUS	Bus resource.
L4VBUS_RESOURCE_GPIO	Gpio resource.
L4VBUS_RESOURCE_DMA_DOMAIN	DMA domain.
L4VBUS_RESOURCE_MAX	Maximum resource id.

Definition at line 39 of file [vbus\\_types.h](#).

## 17.693 vbus\_types.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00013 #pragma once
00014
00015 #include <l4/sys/types.h>
00016
00018 typedef l4_mword_t l4vbus_device_handle_t;
00020 typedef l4_addr_t l4vbus_paddr_t;
00021
00023 typedef struct {

```

```

00025  l4_uint16_t    type;
00027  l4_uint16_t    flags;
00029  l4vbus_paddr_t start;
00031  l4vbus_paddr_t end;
00033  l4vbus_device_handle_t provider;
00035  l4_uint32_t id;
00036 } l4vbus_resource_t;
00037
00039 enum l4vbus_resource_type_t {
00040     L4VBUS_RESOURCE_INVALID = 0,
00041     L4VBUS_RESOURCE_IRQ,
00042     L4VBUS_RESOURCE_MEM,
00043     L4VBUS_RESOURCE_PORT,
00044     L4VBUS_RESOURCE_BUS,
00045     L4VBUS_RESOURCE_GPIO,
00046     L4VBUS_RESOURCE_DMA_DOMAIN,
00047     L4VBUS_RESOURCE_MAX,
00048 };
00049
00051 enum l4vbus_resource_flags_t {
00053     L4VBUS_RESOURCE_F_MEM_R = 0x1,
00055     L4VBUS_RESOURCE_F_MEM_W = 0x2,
00060     L4VBUS_RESOURCE_F_MEM_PREFETCHABLE = 0x10,
00067     L4VBUS_RESOURCE_F_MEM_CACHEABLE = 0x20,
00069     L4VBUS_RESOURCE_F_MEM_MMIO_READ = 0x2000,
00071     L4VBUS_RESOURCE_F_MEM_MMIO_WRITE = 0x4000,
00072 };
00073
00074 enum l4vbus_consts_t {
00075     L4VBUS_DEV_NAME_LEN = 64,
00076     L4VBUS_MAX_DEPTH = 100,
00077 };
00078
00080 typedef struct {
00082     l4_uint32_t    type;
00084     char           name[L4VBUS_DEV_NAME_LEN];
00086     unsigned       num_resources;
00088     unsigned       flags;
00089 } l4vbus_device_t;
00090
00092 enum l4vbus_device_flags_t {
00093     L4VBUS_DEVICE_F_CHILDREN = 0x10,
00094 };

```

## 17.694 vdevice-ops.h

```

00001 /*
00002  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include "vbus_interfaces.h"
00012
00013 enum L4vbus_vdevice_op
00014 {
00015     L4vbus_vdevice_hid = L4VBUS_INTERFACE_GENERIC « L4VBUS_IFACE_SHIFT,
00016     L4vbus_vdevice_adr,
00017     L4vbus_vdevice_get_by_hid,
00018     L4vbus_vdevice_get_next,
00019     L4vbus_vdevice_get_resource,
00020     L4vbus_vdevice_get_hid,
00021     L4vbus_vdevice_is_compatible,
00022     L4vbus_vdevice_get,
00023 };
00024
00025 enum {
00026     L4vbus_vbus_request_resource = L4VBUS_INTERFACE_BUS « L4VBUS_IFACE_SHIFT,
00027     L4vbus_vbus_release_resource,
00028     L4vbus_vbus_assign_dma_domain,
00029 };
00030
00031 enum
00032 {
00033     L4vbus_vicu_get_cap = L4VBUS_INTERFACE_ICU « L4VBUS_IFACE_SHIFT
00034 };
00035

```



```

00024 class State
00025 {
00026 public:
00027     State() {}
00028
00034     explicit State(unsigned v) : _s(v) {}
00035
00041     void add(unsigned bits) throw() { _s |= bits; }
00042
00048     void clear(unsigned bits) throw() { _s &= ~bits; }
00049
00055     void set(unsigned v) throw() { _s = v; }
00056
00057 private:
00058     __typeof__(((l4_vcpu_state_t *)0)->state) _s;
00059 };
00060
00065 class Vcpu : private l4_vcpu_state_t
00066 {
00067 public:
00071     void irq_disable() throw()
00072     { l4vcpu_irq_disable(this); }
00073
00078     unsigned irq_disable_save() throw()
00079     { return l4vcpu_irq_disable_save(this); }
00080
00081     l4_vcpu_state_t *s() { return this; }
00082     l4_vcpu_state_t const *s() const { return this; }
00083
00088     State *state() throw()
00089     {
00090         static_assert(sizeof(State) == sizeof(l4_vcpu_state_t::state),
00091             "size mismatch");
00092         return reinterpret_cast<State*>(&(l4_vcpu_state_t::state));
00093     }
00094
00099     State state() const throw()
00100     { return static_cast<State>(l4_vcpu_state_t::state); }
00101
00106     State *saved_state() throw()
00107     {
00108         static_assert(sizeof(State) == sizeof(l4_vcpu_state_t::saved_state),
00109             "size mismatch");
00110         return reinterpret_cast<State*>(&(l4_vcpu_state_t::saved_state));
00111     }
00116     State saved_state() const throw()
00117     { return static_cast<State>(l4_vcpu_state_t::saved_state); }
00118
00122     l4_uint16_t sticky_flags() const throw()
00123     { return l4_vcpu_state_t::sticky_flags; }
00124
00135     void irq_enable(l4_utcb_t *utcb, l4vcpu_event_hndl_t do_event_work_cb,
00136         l4vcpu_setup_ipc_t setup_ipc) throw()
00137     { l4vcpu_irq_enable(this, utcb, do_event_work_cb, setup_ipc); }
00138
00150     void irq_restore(unsigned s, l4_utcb_t *utcb,
00151         l4vcpu_event_hndl_t do_event_work_cb,
00152         l4vcpu_setup_ipc_t setup_ipc) throw()
00153     { l4vcpu_irq_restore(this, s, utcb, do_event_work_cb, setup_ipc); }
00154
00166     void wait_for_event(l4_utcb_t *utcb, l4vcpu_event_hndl_t do_event_work_cb,
00167         l4vcpu_setup_ipc_t setup_ipc) throw()
00168     { l4vcpu_wait_for_event(this, utcb, do_event_work_cb, setup_ipc); }
00169
00174     void task(L4::Cap<L4::Task> const task = L4::Cap<L4::Task>::Invalid) throw()
00175     { user_task = task.cap(); }
00176
00181     int is_page_fault_entry() const
00182     { return l4vcpu_is_page_fault_entry(this); }
00183
00188     int is_irq_entry() const
00189     { return l4vcpu_is_irq_entry(this); }
00190
00195     l4_vcpu_regs_t *r() throw()
00196     { return &(l4_vcpu_state_t::r); }
00197
00202     l4_vcpu_regs_t const *r() const throw()
00203     { return &(l4_vcpu_state_t::r); }
00204
00209     l4_vcpu_ipc_regs_t *i() throw()
00210     { return &(l4_vcpu_state_t::i); }
00211
00216     l4_vcpu_ipc_regs_t const *i() const throw()
00217     { return &(l4_vcpu_state_t::i); }
00218
00225     void entry_sp(l4_umword_t sp)
00226     { l4_vcpu_state_t::entry_sp = sp; }

```



```

00227
00232 void entry_ip(l4_umword_t ip)
00233 { l4_vcpu_state_t::entry_ip = ip; }
00234
00246 L4_CV static int
00247 ext_alloc(Vcpu **vcpu,
00248           l4_addr_t *ext_state,
00249           L4::Cap<L4::Task> task = L4Re::Env::env()->task(),
00250           L4::Cap<L4Re::Rm> rm = L4Re::Env::env()->rm()) throw();
00251
00259 static inline Vcpu *cast(void *x) throw()
00260 { return reinterpret_cast<Vcpu *>(x); }
00261
00269 static inline Vcpu *cast(l4_addr_t x) throw()
00270 { return reinterpret_cast<Vcpu *>(x); }
00271
00275 void print_state(const char *prefix = "") const throw()
00276 { l4vcpu_print_state(this, prefix); }
00277 };
00278
00279
00280 }

```

## 17.697 l4/sys/vcpu.h File Reference

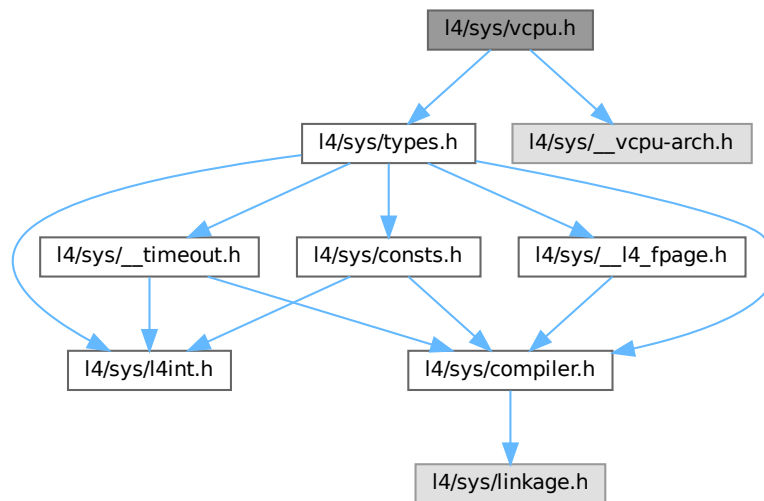
### vCPU API

```

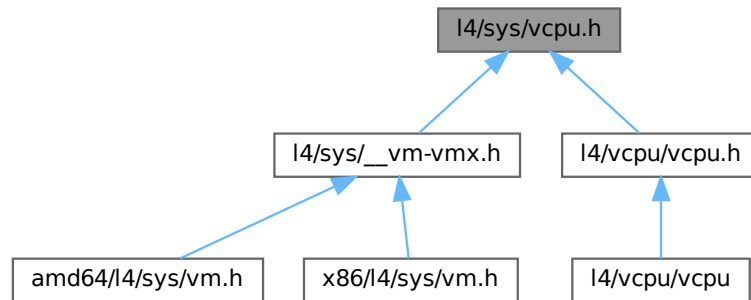
#include <l4/sys/types.h>
#include <l4/sys/__vcpu-arch.h>

```

Include dependency graph for vcpu.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [l4\\_vcpu\\_state\\_t](#)  
*State of a vCPU.*

## Typedefs

- typedef struct l4\_vcpu\_state\_t [l4\\_vcpu\\_state\\_t](#)  
*State of a vCPU.*

## Enumerations

- enum [L4\\_vcpu\\_state\\_flags](#) {  
[L4\\_VCPU\\_F\\_IRQ](#) = 0x01 , [L4\\_VCPU\\_F\\_PAGE\\_FAULTS](#) = 0x02 , [L4\\_VCPU\\_F\\_EXCEPTIONS](#) = 0x04 ,  
[L4\\_VCPU\\_F\\_USER\\_MODE](#) = 0x20 ,  
[L4\\_VCPU\\_F\\_FPU\\_ENABLED](#) = 0x80 }  
*State flags of a vCPU.*
- enum [L4\\_vcpu\\_sticky\\_flags](#) { [L4\\_VCPU\\_SF\\_IRQ\\_PENDING](#) = 0x01 }  
*Sticky flags of a vCPU.*

## Functions

- int [l4\\_vcpu\\_check\\_version](#) ([l4\\_vcpu\\_state\\_t](#) const \*vcpu) [L4\\_NOTHROW](#)  
*Check if a vCPU state has the right version.*

## 17.697.1 Detailed Description

vCPU API

Definition in file [vcpu.h](#).

## 17.697.2 Function Documentation

### 17.697.2.1 l4\_vcpu\_check\_version()

```
int l4_vcpu_check_version (  
    l4_vcpu_state_t const * vcpu) [inline]
```

Check if a vCPU state has the right version.

#### Parameters

---

<code>vcpu</code>	A pointer to an initialized vCPU state.
-------------------	-----------------------------------------

### Return values

1	If the vCPU state has a matching version ID for the current vCPU user-level structures.
0	If the vCPU state has a different (incompatible) version ID than the current vCPU user-level structures.

Definition at line 191 of file [vcpu.h](#).

References [L4\\_NOTHROW](#), and [L4\\_VCPU\\_STATE\\_VERSION](#).

## 17.698 vcpu.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00012 #pragma once
00013
00014 #include <l4/sys/types.h>
00015 #include <l4/sys/__vcpu-arch.h>
00016
00070
00075 typedef struct l4_vcpu_state_t
00076 {
00077     l4_umword_t    version;
00080     l4_umword_t    user_data[7];
00081     l4_vcpu_regs_t r;
00082     l4_vcpu_ipc_regs_t i;
00083
00084     l4_uint16_t    state;
00085     l4_uint16_t    saved_state;
00086     l4_uint16_t    sticky_flags;
00087     l4_uint16_t    _reserved;
00088
00089     l4_cap_idx_t   user_task;
00090
00091     l4_umword_t    entry_sp;
00092     l4_umword_t    entry_ip;
00093     l4_umword_t    reserved_sp;
00094     l4_vcpu_arch_state_t arch_state;
00095 } l4_vcpu_state_t;
00096
00101 enum L4_vcpu_state_flags
00102 {
00114     L4_VCPU_F_IRQ           = 0x01,
00115
00129     L4_VCPU_F_PAGE_FAULTS = 0x02,
00130
00142     L4_VCPU_F_EXCEPTIONS = 0x04,
00143
00152     L4_VCPU_F_USER_MODE   = 0x20,
00153
00160     L4_VCPU_F_FPU_ENABLED = 0x80,
00161 };
00162
00167 enum L4_vcpu_sticky_flags
00168 {
00171     L4_VCPU_SF_IRQ_PENDING = 0x01,
00172 };
00173
00185 L4_INLINE int
00186 l4_vcpu_check_version(l4_vcpu_state_t const *vcpu) L4_NOTHROW;
00187
00188 /* IMPLEMENTATION: -----*/
00189

```

```

00190 L4_INLINE int
00191 l4_vcpu_check_version(l4_vcpu_state_t const *vcpu) L4_NOTHROW
00192 {
00193     return vcpu->version == L4_VCPU_STATE_VERSION;
00194 }

```

## 17.699 l4/vcpu/vcpu.h File Reference

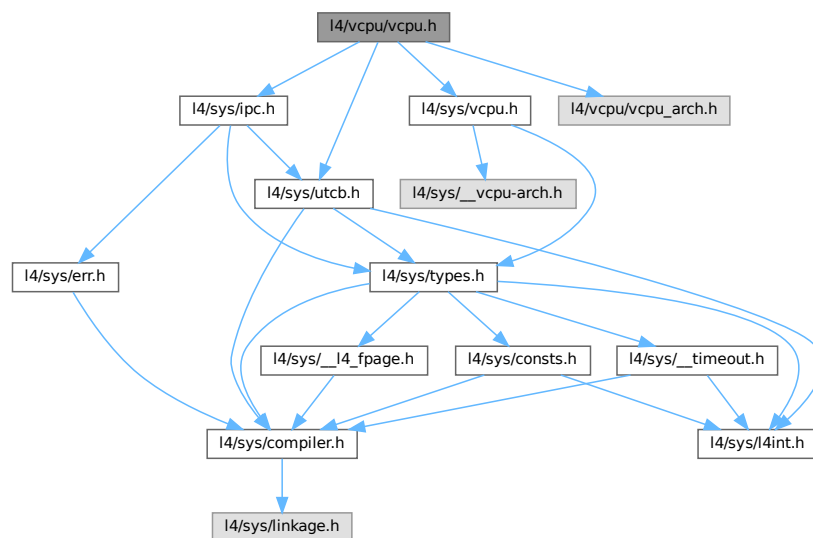
vCPU support library (C interface).

```

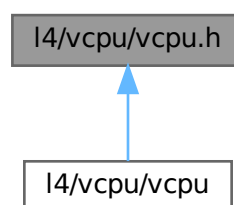
#include <l4/sys/vcpu.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>
#include <l4/vcpu/vcpu_arch.h>

```

Include dependency graph for vcpu.h:



This graph shows which files directly or indirectly include this file:



## Functions

- void [l4vcpu\\_irq\\_disable](#) ([l4\\_vcpu\\_state\\_t](#) \*vcpu) [L4\\_NOTHROW](#)  
*Disable a vCPU for event delivery.*
- unsigned [l4vcpu\\_irq\\_disable\\_save](#) ([l4\\_vcpu\\_state\\_t](#) \*vcpu) [L4\\_NOTHROW](#)  
*Disable a vCPU for event delivery and return previous state.*
- void [l4vcpu\\_irq\\_enable](#) ([l4\\_vcpu\\_state\\_t](#) \*vcpu, [l4\\_utcb\\_t](#) \*utcb, [l4vcpu\\_event\\_hndl\\_t](#) do\_event\_work\_cb, [l4vcpu\\_setup\\_ipc\\_t](#) setup\_ipc) [L4\\_NOTHROW](#)  
*Enable a vCPU for event delivery.*
- void [l4vcpu\\_irq\\_restore](#) ([l4\\_vcpu\\_state\\_t](#) \*vcpu, unsigned s, [l4\\_utcb\\_t](#) \*utcb, [l4vcpu\\_event\\_hndl\\_t](#) do\_event\_work\_cb, [l4vcpu\\_setup\\_ipc\\_t](#) setup\_ipc) [L4\\_NOTHROW](#)  
*Restore a previously saved IRQ/event state.*
- void [l4vcpu\\_wait\\_for\\_event](#) ([l4\\_vcpu\\_state\\_t](#) \*vcpu, [l4\\_utcb\\_t](#) \*utcb, [l4vcpu\\_event\\_hndl\\_t](#) do\_event\_work\_cb, [l4vcpu\\_setup\\_ipc\\_t](#) setup\_ipc) [L4\\_NOTHROW](#)  
*Wait for event.*
- void [l4vcpu\\_print\\_state](#) (const [l4\\_vcpu\\_state\\_t](#) \*vcpu, const char \*prefix) [L4\\_NOTHROW](#)  
*Print the state of a vCPU.*
- int [l4vcpu\\_is\\_irq\\_entry](#) ([l4\\_vcpu\\_state\\_t](#) const \*vcpu) [L4\\_NOTHROW](#)  
*Return whether the entry reason was an IRQ/IPC message.*
- int [l4vcpu\\_is\\_page\\_fault\\_entry](#) ([l4\\_vcpu\\_state\\_t](#) const \*vcpu) [L4\\_NOTHROW](#)  
*Return whether the entry reason was a page fault.*
- int [l4vcpu\\_ext\\_alloc](#) ([l4\\_vcpu\\_state\\_t](#) \*\*vcpu, [l4\\_addr\\_t](#) \*ext\_state, [l4\\_cap\\_idx\\_t](#) task, [l4\\_cap\\_idx\\_t](#) regmgr) [L4\\_NOTHROW](#)  
*Allocate state area for an extended vCPU.*

## 17.699.1 Detailed Description

vCPU support library (C interface).

Definition in file [vcpu.h](#).

## 17.700 vcpu.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00003  *      economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00011 #pragma once
00012
00013 #include <l4/sys/vcpu.h>
00014 #include <l4/sys/utcb.h>
00015
00016 __BEGIN_DECLS
00017
00026
00032
00033 typedef void (*l4vcpu_event_hndl_t) (l4_vcpu_state_t *vcpu);
00034 typedef void (*l4vcpu_setup_ipc_t) (l4_utcb_t *utcb);
00035
00042 L4_CV L4_INLINE
00043 void
00044 l4vcpu_irq_disable(l4_vcpu_state_t *vcpu) L4_NOTHROW;
00045
00054 L4_CV L4_INLINE
00055 unsigned
00056 l4vcpu_irq_disable_save(l4_vcpu_state_t *vcpu) L4_NOTHROW;

```

```

00057
00070 L4_CV L4_INLINE
00071 void
00072 l4vcpu_irq_enable(l4_vcpu_state_t *vcpu, l4_utcb_t *utcb,
00073                  l4vcpu_event_hndl_t do_event_work_cb,
00074                  l4vcpu_setup_ipc_t setup_ipc) L4_NOTHROW;
00075
00090 L4_CV L4_INLINE
00091 void
00092 l4vcpu_irq_restore(l4_vcpu_state_t *vcpu, unsigned s,
00093                   l4_utcb_t *utcb,
00094                   l4vcpu_event_hndl_t do_event_work_cb,
00095                   l4vcpu_setup_ipc_t setup_ipc) L4_NOTHROW;
00096
00110 L4_CV L4_INLINE
00111 void
00112 l4vcpu_wait(l4_vcpu_state_t *vcpu, l4_utcb_t *utcb,
00113             l4_timeout_t to,
00114             l4vcpu_event_hndl_t do_event_work_cb,
00115             l4vcpu_setup_ipc_t setup_ipc) L4_NOTHROW;
00116
00130 L4_CV L4_INLINE
00131 void
00132 l4vcpu_wait_for_event(l4_vcpu_state_t *vcpu, l4_utcb_t *utcb,
00133                      l4vcpu_event_hndl_t do_event_work_cb,
00134                      l4vcpu_setup_ipc_t setup_ipc) L4_NOTHROW;
00135
00136
00144 L4_CV void
00145 l4vcpu_print_state(const l4_vcpu_state_t *vcpu, const char *prefix) L4_NOTHROW;
00146
00150 L4_CV void
00151 l4vcpu_print_state_arch(const l4_vcpu_state_t *vcpu, const char *prefix) L4_NOTHROW;
00152
00162 L4_CV L4_INLINE
00163 int
00164 l4vcpu_is_irq_entry(l4_vcpu_state_t const *vcpu) L4_NOTHROW;
00165
00174 L4_CV L4_INLINE
00175 int
00176 l4vcpu_is_page_fault_entry(l4_vcpu_state_t const *vcpu) L4_NOTHROW;
00177
00189 L4_CV int
00190 l4vcpu_ext_alloc(l4_vcpu_state_t **vcpu, l4_addr_t *ext_state,
00191                 l4_cap_idx_t task, l4_cap_idx_t regmgr) L4_NOTHROW;
00192
00193 /* ===== */
00194 /* Implementations */
00195
00196 #include <l4/sys/ipc.h>
00197 #include <l4/vcpu/vcpu_arch.h>
00198
00199 L4_CV L4_INLINE
00200 void
00201 l4vcpu_irq_disable(l4_vcpu_state_t *vcpu) L4_NOTHROW
00202 {
00203     vcpu->state &= ~L4_VCPU_F_IRQ;
00204     l4_barrier();
00205 }
00206
00207 L4_CV L4_INLINE
00208 unsigned
00209 l4vcpu_irq_disable_save(l4_vcpu_state_t *vcpu) L4_NOTHROW
00210 {
00211     unsigned s = vcpu->state;
00212     l4vcpu_irq_disable(vcpu);
00213     return s;
00214 }
00215
00216 L4_CV L4_INLINE
00217 void
00218 l4vcpu_wait(l4_vcpu_state_t *vcpu, l4_utcb_t *utcb,
00219             l4_timeout_t to,
00220             l4vcpu_event_hndl_t do_event_work_cb,
00221             l4vcpu_setup_ipc_t setup_ipc) L4_NOTHROW
00222 {
00223     l4vcpu_irq_disable(vcpu);
00224     setup_ipc(utcb);
00225     vcpu->i.tag = l4_ipc_wait(utcb, &vcpu->i.label, to);
00226     if (L4_LIKELY(!l4_msgtag_has_error(vcpu->i.tag)))
00227         do_event_work_cb(vcpu);
00228 }
00229
00230 L4_CV L4_INLINE
00231 void
00232 l4vcpu_irq_enable(l4_vcpu_state_t *vcpu, l4_utcb_t *utcb,

```

```

00233         l4vcpu_event_hndl_t do_event_work_cb,
00234         l4vcpu_setup_ipc_t setup_ipc) L4_NOTHROW
00235 {
00236     if (!(vcpu->state & L4_VCPU_F_IRQ))
00237     {
00238         setup_ipc(utcb);
00239         l4_barrier();
00240     }
00241
00242     while (1)
00243     {
00244         vcpu->state |= L4_VCPU_F_IRQ;
00245         l4_barrier();
00246
00247         if (L4_LIKELY(!(vcpu->sticky_flags & L4_VCPU_SF_IRQ_PENDING)))
00248             break;
00249
00250         l4vcpu_wait(vcpu, utcb, L4_IPC_BOTH_TIMEOUT_0,
00251                     do_event_work_cb, setup_ipc);
00252     }
00253 }
00254
00255 L4_CV L4_INLINE
00256 void
00257 l4vcpu_irq_restore(l4_vcpu_state_t *vcpu, unsigned s,
00258                   l4_utcb_t *utcb,
00259                   l4vcpu_event_hndl_t do_event_work_cb,
00260                   l4vcpu_setup_ipc_t setup_ipc) L4_NOTHROW
00261 {
00262     if (s & L4_VCPU_F_IRQ)
00263         l4vcpu_irq_enable(vcpu, utcb, do_event_work_cb, setup_ipc);
00264     else if (vcpu->state & L4_VCPU_F_IRQ)
00265         l4vcpu_irq_disable(vcpu);
00266 }
00267
00268 L4_CV L4_INLINE
00269 void
00270 l4vcpu_wait_for_event(l4_vcpu_state_t *vcpu, l4_utcb_t *utcb,
00271                      l4vcpu_event_hndl_t do_event_work_cb,
00272                      l4vcpu_setup_ipc_t setup_ipc) L4_NOTHROW
00273 {
00274     l4vcpu_wait(vcpu, utcb, L4_IPC_NEVER, do_event_work_cb, setup_ipc);
00275 }
00276
00277 __END_DECLS

```

## 17.701 ipc-invoke.h

```

00001
00009 /*
00010  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00011  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00012  *      Björn Döbel <doebel@os.inf.tu-dresden.de>
00013  *      economic rights: Technische Universität Dresden (Germany)
00014  *
00015  * License: see LICENSE.spdx (in this directory or the directories above)
00016  */
00017
00018 #pragma once
00019
00020 /*
00021  * Some words about the sysenter entry frame: Since the sysenter instruction
00022  * automatically reloads the instruction pointer (eip) and the stack pointer
00023  * (esp) after kernel entry, we have to save both registers preliminary to
00024  * that instruction. We use ecx to store the user-level esp and save eip onto
00025  * the stack. The ecx register contains the IPC timeout and has to be saved
00026  * onto the stack, too. The ebp register is saved for compatibility reasons
00027  * with the Hazelnut kernel. Both the esp and the ss register are also pushed
00028  * onto the stack to be able to return using the "lret" instruction from the
00029  * sysexit trampoline page if Small Address Spaces are enabled.
00030  */
00031
00032 #ifdef __PIC__
00033 # define L4S_PIC_SAVE "push %%ebx; "
00034 # define L4S_PIC_RESTORE "pop %%ebx; "
00035 # define L4S_PIC_CLOBBER
00036 # define L4S_PIC_SYSCALL , [func] "m" (__l4sys_invoke_indirect)
00037 # if 1
00038 extern void (*__l4sys_invoke_indirect)(void);
00039 # define IPC_SYSENTER      "# indirect sys invoke \n\t" \
00040                          "call *%[func] \n\t"
00041 # else
00042 # define L4S_PIC_SYSCALL

```



```

00043 # define IPC_SYSENTER      "call __l4sys_invoke_direct@plt  \n\t"
00044 # endif
00045 # define IPC_SYSENTER_ASM    call __l4sys_invoke_direct@plt
00046 #else
00051 #define IPC_SYSENTER      "call __l4sys_invoke_direct  \n\t"
00056 #define IPC_SYSENTER_ASM  call __l4sys_invoke_direct
00061 # define L4S_PIC_SAVE
00066 # define L4S_PIC_RESTORE
00071 # define L4S_PIC_CLOBBER , "ebx"
00072 # define L4S_PIC_SYSCALL
00073
00074 #endif
00079 #define L4_ENTER_KERNEL L4S_PIC_SAVE "push %%ebp; " \
00080 IPC_SYSENTER
00081 " pop %%ebp; " L4S_PIC_RESTORE
00082

```

## 17.702 ipc-l42-gcc3-nopic.h

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Frank Mehnert <fm3@os.inf.tu-dresden.de>,
00010  *      Jork Löser <jork@os.inf.tu-dresden.de>
00011  *      economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * License: see LICENSE.spdx (in this directory or the directories above)
00014  */
00015 #pragma once
00016
00017 #include <l4/sys/consts.h>
00018
00019 L4_INLINE l4_msgtag_t
00020 l4_ipc(l4_cap_idx_t dest, l4_utcb_t *u,
00021        l4_umword_t flags,
00022        l4_umword_t slabel,
00023        l4_msgtag_t tag,
00024        l4_umword_t *rlabel,
00025        l4_timeout_t timeout) L4_NOTHROW
00026 {
00027     l4_umword_t dummy, dummy1, dummy2;
00028
00029     (void)u;
00030
00031     __asm__ __volatile__
00032     (L4_ENTER_KERNEL
00033      :
00034      "=d" (dummy2),
00035      "=S" (slabel),
00036      "=c" (dummy1),
00037      "=D" (dummy),
00038      "=a" (tag.raw)
00039      :
00040      "S" (slabel),
00041      "c" (timeout),
00042      "a" (tag.raw),
00043      "d" (dest | flags)
00044      : L4S_PIC_SYSCALL
00045      :
00046      "memory", "cc" L4S_PIC_CLOBBER
00047     );
00048
00049     if (rlabel)
00050         *rlabel = slabel;
00051
00052     return tag;
00053 }

```



# Chapter 18

## Examples

### 18.1 hello/server/src/main.c

This is the famous "Hello World!" program.

This is the famous "Hello World!" program.

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *             Frank Mehnert <fm3@os.inf.tu-dresden.de>,
 *             Lukas Grützmacher <lg2@os.inf.tu-dresden.de>
 *             economic rights: Technische Universität Dresden (Germany)
 *
 * License: see LICENSE.spdx (in this directory or the directories above)
 */
#include <stdio.h>
#include <unistd.h>

int
main(void)
{
    for (;;)
    {
        puts("Hello World!");
        sleep(1);
    }
}
```

### 18.2 examples/sys/ipc/ipc\_example.c

This example shows how two threads can exchange data using the [L4](#) IPC mechanism.

This example shows how two threads can exchange data using the [L4](#) IPC mechanism. One thread is sending an integer to the other thread which is returning the square of the integer. Both values are printed.

```
/*
 * (c) 2008-2009 Author(s)
 *             economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/sys/ipc.h>

#include <pthread-l4.h>
#include <unistd.h>
```

```

#include <stdio.h>

static pthread_t t2;

/* Thread1 is the initiator thread, i.e. it initiates the IPC calls. In
 * other words, it takes the client role. It uses L4 IPC mechanisms to send
 * an integer value to thread2 and received a calculation result back. */
static void *thread1_fn(void *arg)
{
    l4_msgtag_t tag;
    int ipc_error;
    unsigned long value = 1;
    (void)arg;

    while (1)
    {
        printf("Sending: %ld\n", value);

        /* Store the value which we want to have squared in the first message
         * register of our UTCB. */
        l4_utcb_mr()->mr[0] = value;

        /* To an L4 IPC call, i.e. send a message to thread2 and wait for a
         * reply from thread2. The '1' in the msgtag denotes that we want to
         * transfer one word of our message registers (i.e. MR0). No timeout. */
        tag = l4_ipc_call(pthread_l4_cap(t2), l4_utcb(),
                          l4_msgtag(0, 1, 0, 0), L4_IPC_NEVER);
        /* Check for IPC error, if yes, print out the IPC error code, if not,
         * print the received result. */
        ipc_error = l4_ipc_error(tag, l4_utcb());
        if (ipc_error)
            fprintf(stderr, "thread1: IPC error: %x\n", ipc_error);
        else
            printf("Received: %ld\n", l4_utcb_mr()->mr[0]);

        /* Wait some time and increment our value. */
        sleep(1);
        value++;
    }
    return NULL;
}

/* Thread2 is in the server role, i.e. it waits for requests from others and
 * sends back the calculation results. */
static void *thread2_fn(void *arg)
{
    l4_msgtag_t tag;
    l4_umword_t label;
    int ipc_error;
    (void)arg;

    /* Wait for requests from any thread. No timeout, i.e. wait forever. */
    tag = l4_ipc_wait(l4_utcb(), &label, L4_IPC_NEVER);
    while (1)
    {
        /* Check if we had any IPC failure, if yes, print the error code
         * and just wait again. */
        ipc_error = l4_ipc_error(tag, l4_utcb());
        if (ipc_error)
        {
            fprintf(stderr, "thread2: IPC error: %x\n", ipc_error);
            tag = l4_ipc_wait(l4_utcb(), &label, L4_IPC_NEVER);
            continue;
        }

        /* So, the IPC was ok, now take the value out of message register 0
         * of the UTCB and store the square of it back to it. */
        l4_utcb_mr()->mr[0] = l4_utcb_mr()->mr[0] * l4_utcb_mr()->mr[0];

        /* Send the reply and wait again for new messages.
         * The '1' in the msgtag indicated that we want to transfer 1 word in
         * the message registers (i.e. MR0) */
        tag = l4_ipc_reply_and_wait(l4_utcb(), l4_msgtag(0, 1, 0, 0),
                                    &label, L4_IPC_NEVER);
    }
    return NULL;
}

int main(void)
{
    // We will have two threads, one is already running the main function, the
    // other (thread2) will be created using pthread_create.

    if (pthread_create(&t2, NULL, thread2_fn, NULL))
    {
        fprintf(stderr, "Thread creation failed\n");
        return 1;
    }
}

```

```

    }

    // Just run thread1 in the main thread
    thread1_fn(NULL);
    return 0;
}

```

## 18.3 examples/sys/ipc/ipc.cfg

Sample configuration file for the IPC example.

Sample configuration file for the IPC example.

```

# vim:se ft=lua:

local L4 = require("L4");

L4.default_loader:start({}, "rom/ex_ipc1");

```

## 18.4 examples/sys/start-with-exc/main.c

This example shows how to start a newly created thread with a defined set of CPU registers.

This example shows how to start a newly created thread with a defined set of CPU registers.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *             Alexander Warg <warg@os.inf.tu-dresden.de>,
 *             Björn Döbel <doebel@os.inf.tu-dresden.de>,
 *             Frank Mehnert <fm3@os.inf.tu-dresden.de>
 *             economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
/*
 * Start a thread with an exception reply. This example does only work on
 * the x86-32 and ARM architectures.
 */

#include <l4/sys/thread.h>
#include <l4/sys/factory.h>
#include <l4/sys/ipc.h>
#include <l4/sys/utcb.h>
#include <l4/util/util.h>
#include <l4/re/env.h>
#include <l4/re/c/util/cap_alloc.h>

#include <stdlib.h>
#include <stdio.h>

/* Stack for the thread to be created. 8kB are enough. */
static char thread_stack[8 « 10];

/* The thread to be created. For illustration it will print out its
 * register set.
 */
static void L4_STICKY(thread_func(l4_umword_t *d))
{
    while (1)
    {
        printf("hey, I'm a thread\n");
        printf("got register values: %ld %ld %ld %ld %ld %ld %ld %ld\n",
               d[7], d[6], d[5], d[4], d[2], d[1], d[0]);
        l4_sleep(800);
    }
}

/* Startup trick for this example. Put all the CPU registers on the stack so
 * that the C function above can get it on the stack. */
asm(
    ".global thread    \n"
    "thread:          \n"

```

```

#ifdef ARCH_x86
" pusha \n"
" push %esp \n"
" call thread_func \n"
#endif
#ifdef ARCH_arm
"      push {r0-r7} \n"
"      mov r0, sp \n"
"      bl thread_func \n"
#endif
#ifdef ARCH_arm64
"      stp x0, x1, [sp, #0]! \n"
"      stp x2, x3, [sp, #0]! \n"
"      stp x4, x5, [sp, #0]! \n"
"      stp x6, x7, [sp, #0]! \n"
"      mov x0, sp \n"
"      bl thread_func \n"
#endif
);
extern void thread(void);

/* Our main function */
int main(void)
{
    /* Get a capability slot for our new thread. */
    l4_cap_idx_t t1 = l4re_util_cap_alloc();
    l4_utcb_t *u = l4_utcb();
    l4_exc_regs_t *e = l4_utcb_exc_u(u);
    l4_msgtag_t tag;
    int err;

    printf("Example showing how to start a thread with an exception.\n");
    /* We do not want to implement a pager here, take the shortcut. */
    printf("Make sure to start this program with ldr-flags=eager_map\n");

    if (l4_is_invalid_cap(t1))
        return 1;

    /* Create the thread using our default factory */
    tag = l4_factory_create_thread(l4re_env()->factory, t1);
    if (l4_error(tag))
        return 1;

    /* Setup the thread by setting the pager and task. */
    l4_thread_control_start();
    l4_thread_control_pager(l4re_env()->main_thread);
    l4_thread_control_exc_handler(l4re_env()->main_thread);
    l4_thread_control_bind((l4_utcb_t *)l4re_env()->first_free_utcb,
                          L4RE_THIS_TASK_CAP);
    tag = l4_thread_control_commit(t1);
    if (l4_error(tag))
        return 2;

    /* Start the thread by finally setting instruction and stack pointer */
    tag = l4_thread_ex_regs(t1,
                           (l4_umword_t)thread,
                           (l4_umword_t)thread_stack + sizeof(thread_stack),
                           L4_THREAD_EX_REGS_TRIGGER_EXCEPTION);

    if (l4_error(tag))
        return 3;

    l4_sched_param_t sp = l4_sched_param(1, 0);
    tag = l4_scheduler_run_thread(l4re_env()->scheduler, t1, &sp);
    if (l4_error(tag))
        return 4;

    /* Receive initial exception from just started thread */
    tag = l4_ipc_receive(t1, u, L4_IPC_NEVER);
    if ((err = l4_ipc_error(tag, u)))
    {
        printf("Umm, ipc error: %x\n", err);
        return 1;
    }

    /* We expect an exception IPC */
    if (!l4_msgtag_is_exception(tag))
    {
        printf("PF?: %lx %lx (not prepared to handle this) %ld\n",
              l4_utcb_mr_u(u)->mr[0], l4_utcb_mr_u(u)->mr[1], l4_msgtag_label(tag));
        return 1;
    }

    /* Fill out the complete register set of the new thread */
    e->sp = (l4_umword_t)(thread_stack + sizeof(thread_stack));
#ifdef ARCH_x86
    e->ip = (l4_umword_t)thread;
    e->edi = 0;

```

```

    e->esi = 1;
    e->ebp = 2;
    e->ebx = 4;
    e->edx = 5;
    e->ecx = 6;
    e->eax = 7;
#endif
#ifdef ARCH_arm
    e->pc = (l4_umword_t)thread;
    e->r[0] = 0;
    e->r[1] = 1;
    e->r[2] = 2;
    e->r[3] = 3;
    e->r[4] = 4;
    e->r[5] = 5;
    e->r[6] = 6;
    e->r[7] = 7;
#endif
#ifdef ARCH_arm64
    e->pc = (l4_umword_t)thread;
    e->r[0] = 0;
    e->r[1] = 1;
    e->r[2] = 2;
    e->r[3] = 3;
    e->r[4] = 4;
    e->r[5] = 5;
    e->r[6] = 6;
    e->r[7] = 7;
#endif
/* Send a complete exception */
tag = l4_msgtag(0, L4_UTCB_EXCEPTION_REGS_SIZE, 0, 0);

/* Send reply and start the thread with the defined CPU register set */
tag = l4_ipc_send(tl, u, tag, L4_IPC_NEVER);
if ((err = l4_ipc_error(tag, u)))
    printf("Error sending IPC: %x\n", err);

/* Idle around */
while (1)
    l4_sleep(10000);

return 0;
}

```

## 18.5 examples/sys/singlestep/main.c

This example shows how a thread can be single stepped on the x86 architecture.

This example shows how a thread can be single stepped on the x86 architecture.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *           Alexander Warg <warg@os.inf.tu-dresden.de>,
 *           Björn Döbel <doebel@os.inf.tu-dresden.de>
 *           economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
/*
 * Single stepping example for the x86-32 architecture.
 */
#include <l4/sys/ipc.h>
#include <l4/sys/factory.h>
#include <l4/sys/thread.h>
#include <l4/sys/utcb.h>
#include <l4/sys/kdebug.h>

#include <l4/util/util.h>
#include <l4/re/env.h>
#include <l4/re/c/util/cap_alloc.h>

#include <stdlib.h>
#include <stdio.h>
#include <string.h>

static char thread_stack[8 « 10];

static void thread_func(void)
{

```

```

while (1)
{
    unsigned long d = 0;

    /* Enable single stepping */
    asm volatile("pushf; pop %0; or $256,%0; push %0; popf\n"
        : "=r" (d) : "r" (d));

    /* Some instructions */
    asm volatile("nop");
    asm volatile("nop");
    asm volatile("nop");
    asm volatile("mov $0x12345000, %%edx" : : : "edx"); // a non-existent cap
    asm volatile("int $0x30\n");
    asm volatile("nop");
    asm volatile("nop");
    asm volatile("nop");

    /* Disabled single stepping */
    asm volatile("pushf; pop %0; and $~256,%0; push %0; popf\n"
        : "=r" (d) : "r" (d));

    /* You won't see those */
    asm volatile("nop");
    asm volatile("nop");
    asm volatile("nop");
}

int main(void)
{
    l4_msgtag_t tag;
    int ipc_stat = 0;
    l4_cap_idx_t th = l4re_util_cap_alloc();
    l4_exc_regs_t exc;
    l4_umword_t mr0, mr1;
    l4_utcb_t *u = l4_utcb();

    printf("Singlestep testing\n");

    if (l4_is_invalid_cap(th))
        return 1;

    l4_touch_rw(thread_stack, sizeof(thread_stack));
    l4_touch_ro(thread_func, 1);

    tag = l4_factory_create_thread(l4re_env()->factory, th);
    if (l4_error(tag))
        return 1;

    l4_thread_control_start();
    l4_thread_control_pager(l4re_env()->main_thread);
    l4_thread_control_exc_handler(l4re_env()->main_thread);
    l4_thread_control_bind((l4_utcb_t *)l4re_env()->first_free_utcb,
        L4RE_THIS_TASK_CAP);
    l4_thread_control_alien(1);
    tag = l4_thread_control_commit(th);
    if (l4_error(tag))
        return 2;

    tag = l4_thread_ex_regs(th, (l4_umword_t)thread_func,
        (l4_umword_t)thread_stack + sizeof(thread_stack),
        0);
    if (l4_error(tag))
        return 3;

    l4_sched_param_t sp = l4_sched_param(1, 0);
    tag = l4_scheduler_run_thread(l4re_env()->scheduler, th, &sp);
    if (l4_error(tag))
        return 4;

    /* Pager/Exception loop */
    if (l4_msgtag_has_error(tag = l4_ipc_receive(th, u, L4_IPC_NEVER)))
    {
        printf("l4_ipc_receive failed");
        return 5;
    }

    memcpy(&exc, l4_utcb_exc(), sizeof(exc));
    mr0 = l4_utcb_mr()->mr[0];
    mr1 = l4_utcb_mr()->mr[1];

    for (;;)
    {
        if (l4_msgtag_is_exception(tag))
        {
            printf("PC = %08lx Trap = %08lx Err = %08lx, SP = %08lx SC-Nr: %lx\n",
                l4_utcb_exc_pc(&exc), exc.trapno, exc.err,

```



```

        exc.sp, exc.err » 3);
    if (exc.err » 3)
    {
        if (!(exc.err & 4))
        {
            tag = l4_msgtag(L4_PROTO_ALLOW_SYSCALL,
                           L4_UTCB_EXCEPTION_REGS_SIZE, 0, 0);
            if (ipc_stat)
                enter_kdebug("Should not be 1");
        }
        else
        {
            tag = l4_msgtag(L4_PROTO_NONE,
                           L4_UTCB_EXCEPTION_REGS_SIZE, 0, 0);
            if (!ipc_stat)
                enter_kdebug("Should not be 0");
        }
        ipc_stat = !ipc_stat;
    }
    l4_sleep(100);
}
else
    printf("Umm, non-handled request: %ld, %08lx %08lx\n",
          l4_msgtag_label(tag), mr0, mr1);

memcpy(l4_utcb_exc(), &exc, sizeof(exc));

/* Reply and wait */
if (l4_msgtag_has_error(tag = l4_ipc_call(th, u, tag, L4_IPC_NEVER)))
{
    printf("l4_ipc_call failed\n");
    return 5;
}
memcpy(&exc, l4_utcb_exc(), sizeof(exc));
mr0 = l4_utcb_mr()->mr[0];
mr1 = l4_utcb_mr()->mr[1];
}

return 0;
}

```

## 18.6 examples/sys/aliens/main.c

This example shows how system call tracing can be done.

This example shows how system call tracing can be done.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>,
 *      Björn Döbel <doebel@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
/*
 * Example to show syscall tracing.
 */
#if defined(ARCH_x86) || defined(ARCH_amd64)
// MEASURE only works on x86/amd64
// #define MEASURE
#endif

#include <l4/sys/ipc.h>
#include <l4/sys/thread.h>
#include <l4/sys/factory.h>
#include <l4/sys/utcb.h>
#include <l4/util/util.h>
#include <l4/re/env.h>
#include <l4/re/c/util/cap_alloc.h>
#include <l4/re/c/util/kumem_alloc.h>
#include <l4/sys/debugger.h>

#include <stdlib.h>
#include <stdio.h>
#include <string.h>

/* Architecture specifics */

```

```

#if defined(ARCH_x86) || defined(ARCH_amd64)

static int
is_alien_after_call(l4_exc_regs_t const *exc)
{
    #if defined(ARCH_x86)
        return exc->err & 4;
    #else
        return exc->err == 1;
    #endif
}

static inline void
_print_exc_state(l4_exc_regs_t const *exc)
{
    printf("PC=%08lx SP=%08lx Err=%08lx Trap=%lx, %s syscall, SC-Nr: %lx\n",
        l4_utcb_exc_pc(exc), exc->sp, exc->err,
        exc->trapno, is_alien_after_call(exc) ? " after" : "before",
        exc->err >> 3);
}

#elif defined(ARCH_arm)

static int
is_alien_after_call(l4_exc_regs_t const *exc)
{ return exc->err & 0x40; } // TODO: Should change this to (1 << 16)

static inline void
_print_exc_state(l4_exc_regs_t const *exc)
{
    printf("PC=%08lx SP=%08lx ULR=%08lx CPSR=%08lx Err=%lx/%lx, %s syscall\n",
        l4_utcb_exc_pc(exc), exc->sp, exc->ulr, exc->cpsr,
        exc->err, exc->err >> 26,
        is_alien_after_call(exc) ? " after" : "before");
}

#elif defined(ARCH_arm64)

static int
is_alien_after_call(l4_exc_regs_t const *exc)
{ return exc->err & (1ul << 16); }

static inline void
_print_exc_state(l4_exc_regs_t const *exc)
{
    printf("PC=%08lx SP=%08lx PSTATE=%08lx Err=%lx/%lx, %s syscall\n",
        l4_utcb_exc_pc(exc), exc->sp, exc->pstate,
        exc->err, exc->err >> 26,
        is_alien_after_call(exc) ? " after" : "before");
}

#elif defined(ARCH_mips)

static int
is_alien_after_call(l4_exc_regs_t const *exc)
{ return 0; }

static inline void
_print_exc_state(l4_exc_regs_t const *exc)
{
    printf("PC=%08lx SP=%08lx Cause=%lx, %s syscall\n",
        l4_utcb_exc_pc(exc), exc->sp, exc->cause,
        is_alien_after_call(exc) ? " after" : "before");
}

#elif defined(ARCH_riscv)

static int
is_alien_after_call(l4_exc_regs_t const *exc)
{ return exc->cause == L4_riscv_ec_l4_alien_after_syscall; }

static inline void
_print_exc_state(l4_exc_regs_t const *exc)
{
    printf("PC=%08lx SP=%08lx Cause=%lx, %s syscall\n",
        l4_utcb_exc_pc(exc), exc->sp, exc->cause,
        is_alien_after_call(exc) ? " after" : "before");
}

#else

static int
is_alien_after_call(l4_exc_regs_t const *exc)
{ return exc->err & 1; }

static inline void
_print_exc_state(l4_exc_regs_t const *exc)

```

```

{
    printf("PC=%08lx SP=%08lx, %s syscall\n",
        l4_utcb_exc_pc(exc), exc->sp,
        is_alien_after_call(exc) ? " after" : "before");
}

#endif

/* Measurement mode specifics.
 *
 * In measurement mode the code is less verbose and uses RDTSC for alien exception
 * performance measurement.
 */
#ifdef MEASURE

#include <l4/util/rdtsc.h>

static inline void
calibrate_timer(void)
{
    l4_calibrate_tsc(l4re_kip());
}

static inline void
print_timediff(l4_cpu_time_t start)
{
    e = l4_rdtsc();
    printf("time %lld\n", l4_tsc_to_ns(e - start));
}

static inline void
alien_sleep(void)
{
    l4_sleep(0);
}

static inline void
print_exc_state(l4_exc_regs_t const *exc)
{
    if (0)
        _print_exc_state(exc);
}

#else

static inline void
calibrate_timer(void)
{
}

static inline void
print_timediff(l4_cpu_time_t start)
{
    (void)start;
}

static inline l4_cpu_time_t
l4_rdtsc(void)
{
    return 0;
}

static inline void
alien_sleep(void)
{
    l4_sleep(1000);
}

static inline void
print_exc_state(l4_exc_regs_t const *exc)
{
    _print_exc_state(exc);
}

#endif

static char alien_thread_stack[8 « 10];
static l4_cap_idx_t alien;

static void alien_thread(void)
{
    while (1)
    {
        l4_ipc_call(0x1234 « L4_CAP_SHIFT, l4_utcb(),
            l4_msgtag(0, 0, 0, 0), L4_IPC_NEVER);
        alien_sleep();
    }
}

```

```

    }
}

int main(void)
{
    l4_msgtag_t tag;
    l4_cpu_time_t s;
    l4_utcb_t *u = l4_utcb();
    l4_exc_regs_t exc;
    l4_umword_t mr0, mr1;

    printf("Alien feature testing\n");

    l4_debugger_set_object_name(l4re_env()->main_thread, "alientest");

    /* Start alien thread */
    if (l4_is_invalid_cap(alien = l4re_util_cap_alloc()))
        return 1;

    l4_touch_rw(alien_thread_stack, sizeof(alien_thread_stack));

    tag = l4_factory_create_thread(l4re_env()->factory, alien);
    if (l4_error(tag))
        return 2;

    l4_debugger_set_object_name(alien, "alienth");

    l4_addr_t kumem;
    if (l4re_util_kumem_alloc(&kumem, 0, L4RE_THIS_TASK_CAP, l4re_env()->rm))
        return 3;

    l4_thread_control_start();
    l4_thread_control_pager(l4re_env()->main_thread);
    l4_thread_control_exc_handler(l4re_env()->main_thread);
    l4_thread_control_bind((l4_utcb_t *)kumem, L4RE_THIS_TASK_CAP);
    l4_thread_control_alien(1);
    tag = l4_thread_control_commit(alien);
    if (l4_error(tag))
        return 4;

    tag = l4_thread_ex_regs(alien,
                           (l4_umword_t)alien_thread,
                           (l4_umword_t)alien_thread_stack + sizeof(alien_thread_stack),
                           0);

    if (l4_error(tag))
        return 5;

    l4_sched_param_t sp = l4_sched_param(1, 0);
    tag = l4_scheduler_run_thread(l4re_env()->scheduler, alien, &sp);
    if (l4_error(tag))
        return 6;

    calibrate_timer();

    /* Pager/Exception loop */
    if (l4_msgtag_has_error(tag = l4_ipc_receive(alien, u, L4_IPC_NEVER)))
    {
        printf("l4_ipc_receive failed");
        return 7;
    }

    memcpy(&exc, l4_utcb_exc(), sizeof(exc));
    mr0 = l4_utcb_mr()->mr[0];
    mr1 = l4_utcb_mr()->mr[1];

    for (;;)
    {
        s = l4_rdtsc();

        if (l4_msgtag_is_exception(tag))
        {
            print_exc_state(&exc);
            tag = l4_msgtag(is_alien_after_call(&exc)
                           ? 0 : L4_PROTO_ALLOW_SYSCALL,
                           L4_UTCB_EXCEPTION_REGS_SIZE, 0, 0);
        }
        else
            printf("Umm, non-handled request (like PF): %lx %lx\n", mr0, mr1);

        memcpy(l4_utcb_exc(), &exc, sizeof(exc));

        /* Reply and wait */
        if (l4_msgtag_has_error(tag = l4_ipc_call(alien, u, tag, L4_IPC_NEVER)))
        {
            printf("l4_ipc_call failed\n");
            return 8;
        }
    }
}

```

```

        memcpy(&exc, l4_utcb_exc(), sizeof(exc));
        mr0 = l4_utcb_mr()->mr[0];
        mr1 = l4_utcb_mr()->mr[1];
        print_timediff(s);
    }

    return 0;
}

```

## 18.7 examples/sys/utcb-ipc/main.c

This example shows how to send IPC using the UTCB to store payload.

This example shows how to send IPC using the UTCB to store payload.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>,
 *      Björn Döbel <doebel@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/sys/ipc.h>
#include <l4/sys/thread.h>
#include <l4/sys/factory.h>
#include <l4/sys/utcb.h>
#include <l4/sys/task.h>
#include <l4/sys/vcon.h>
#include <l4/re/env.h>
#include <l4/re/c/util/cap_alloc.h>
#include <l4/re/c/util/kumem_alloc.h>
#include <l4/util/thread.h>

#include <stdio.h>
#include <string.h>

static unsigned char stack2[8 « 10] __attribute__((aligned(8)));
static l4_cap_idx_t thread1_cap, thread2_cap;

static void vlogprintn(const char *s, int l)
{
    if (l > L4_VCON_WRITE_SIZE)
        l = L4_VCON_WRITE_SIZE;

    l4_vcon_send(L4_BASE_LOG_CAP, s, l);
}

static void vlogprint(const char *s)
{
    vlogprintn(s, strlen(s));
}

static void vlogputc(const char c)
{
    vlogprintn(&c, 1);
}

static void thread1(void)
{
    l4_msg_regs_t *mr = l4_utcb_mr();
    l4_msgtag_t tag;
    int i, j;

    printf("Thread1 up (%p)\n", l4_utcb());

    for (i = 0; i < 10; i++)
    {
        for (j = 0; j < L4_UTCB_GENERIC_DATA_SIZE; j++)
            mr->mr[j] = 'A' + (i + j) % ('~' - 'A' + 1);
        tag = l4_msgtag(0, L4_UTCB_GENERIC_DATA_SIZE, 0, 0);
        if (l4_msgtag_has_error(l4_ipc_send(thread2_cap, l4_utcb(), tag, L4_IPC_NEVER)))
            printf("IPC-send error\n");
    }

    mr->mr[0] = 1;
    if (l4_msgtag_has_error(l4_ipc_send(thread2_cap, l4_utcb(), tag, L4_IPC_NEVER)))

```

```

    printf("IPC-send error\n");

    printf("Thread1 done\n");
}

L4UTIL_THREAD_STATIC_FUNC(thread2)
{
    l4_msgtag_t tag;
    l4_msg_regs_t mr;
    unsigned i;

    // No printf() here because this would require a working pthread environment!
    vlogprint("Thread2 up\n");

    while (1)
    {
        if (l4_msgtag_has_error(tag = l4_ipc_receive(thread1_cap, l4_utcb(), L4_IPC_NEVER)))
            vlogprint("IPC receive error\n");
        memcpy(&mr, l4_utcb_mr(), sizeof(mr));
        if (mr.mr[0] == 1) // exit notification
            break;
        vlogprint("Thread2 receive: ");
        for (i = 0; i < l4_msgtag_words(tag); i++)
            vlogprintc((char)mr.mr[i]);
        vlogprint("\n");
    }

    vlogprint("Thread2 done, switching to thread1\n");
    if (l4_msgtag_has_error(l4_ipc_send(thread1_cap, l4_utcb(),
                                      tag, L4_IPC_NEVER)))
        vlogprint("IPC-send error\n");

    // In theory this could hit if the above IPC send operation doesn't switch
    // to the other thread.
    __builtin_trap();
}

int main(void)
{
    l4_msgtag_t tag;

    thread1_cap = l4re_env()->main_thread;
    thread2_cap = l4re_util_cap_alloc();

    if (l4_is_invalid_cap(thread2_cap))
    {
        printf("Cannot allocate thread2 capability\n");
        return 1;
    }

    tag = l4_factory_create_thread(l4re_env()->factory, thread2_cap);
    if (l4_error(tag))
    {
        printf("Cannot create thread2\n");
        return 2;
    }

    l4_addr_t kumem;
    if (l4re_util_kumem_alloc(&kumem, 0, L4RE_THIS_TASK_CAP, l4re_env()->rm))
    {
        printf("Cannot allocate UTCB for thread2\n");
        return 3;
    }

    l4_thread_control_start();
    l4_thread_control_pager(l4re_env()->rm);
    l4_thread_control_exc_handler(l4re_env()->rm);
    l4_thread_control_bind((l4_utcb_t *)kumem, L4RE_THIS_TASK_CAP);
    tag = l4_thread_control_commit(thread2_cap);
    if (l4_error(tag))
    {
        printf("Cannot set thread2 thread parameters\n");
        return 4;
    }

    tag = l4_thread_ex_regs(thread2_cap,
                           (l4_umword_t)thread2,
                           (l4_umword_t)(stack2 + sizeof(stack2)), 0);
    if (l4_error(tag))
    {
        printf("Cannot set thread2 IP/SP\n");
        return 5;
    }

    l4_sched_param_t sp = l4_sched_param(1, 0);
    tag = l4_scheduler_run_thread(l4re_env()->scheduler, thread2_cap, &sp);
    if (l4_error(tag))

```

```

    {
        printf("Cannot start thread2\n");
        return 6;
    }

thread1();

if (l4_msgtag_has_error(l4_ipc_receive(thread2_cap, l4_utcb(),
                                      L4_IPC_NEVER)))
    printf("IPC-receive error\n");

l4_task_unmap(L4RE_THIS_TASK_CAP,
              l4_obj_fpage(thread2_cap, 0, L4_FPAGE_RWX),
              L4_FP_ALL_SPACES);

printf("Terminated thread2. Terminating.\n");
return 0;
}

```

## 18.8 examples/sys/isr/main.c

Example of an interrupt service routine.

Example of an interrupt service routine.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *           Alexander Warg <warg@os.inf.tu-dresden.de>,
 *           Björn Döbel <doebel@os.inf.tu-dresden.de>
 *           economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
/*
 * This example shall show how to connect to an interrupt, receive interrupt
 * events and detach again. As the interrupt source we'll use the virtual
 * key interrupt -- pin 0 of the log capability.
 */

#include <l4re/c/util/cap_alloc.h>
#include <l4re/c/namespace.h>
#include <l4/sys/factory.h>
#include <l4/sys/icu.h>
#include <l4/sys/irq.h>
#include <l4/sys/vcon.h>
#include <l4/sys/utcb.h>

#include <stdio.h>

int main(void)
{
    int const irqno = 0;
    l4_cap_idx_t irqcap, icucap;
    l4_vcon_attr_t attr;
    long err;

    icucap = l4re_env()->log;

    /* Get a free capability slot for the ICU capability. */
    if (l4_is_invalid_cap(icucap))
    {
        printf("Did not find the Vlog ICU.\n");
        return 1;
    }

    /* Get another free capability slot for the corresponding IRQ object. */
    if (l4_is_invalid_cap(irqcap = l4re_util_cap_alloc()))
    {
        printf("Cannot allocate capability slot.\n");
        return 1;
    }

    /* Create IRQ object. */
    if ((err = l4_error(l4_factory_create_irq(l4re_env()->factory, irqcap))))
    {
        printf("Could not create IRQ object: %ld (%s).\n", err, l4sys_errtostr(err));
        return 1;
    }

    /*

```

```

    * Bind the recently allocated IRQ object to the IRQ number irqno as provided
    * by the ICU.
    */
if ((err = l4_error(l4_icu_bind(icucap, irqno, irqcap)))
{
    printf("Could not bind IRQ%d to ICU: %ld (%s).\n", irqno, err, l4sys_errtostr(err));
    return 1;
}

if ((err = l4_error(l4_vcon_get_attr(icucap, &attr)))
{
    printf("Could not get Vcon attributes: %ld (%s).\n", err, l4sys_errtostr(err));
    return 1;
}

/* Disable echo at Vcon console. */
attr.l_flags &= ~L4_VCON_ECHO;

if ((err = l4_error(l4_vcon_set_attr(icucap, &attr)))
{
    printf("Could not set Vcon attributes: %ld (%s).\n", err, l4sys_errtostr(err));
    return 1;
}

printf("Vcon echo disabled.\n");

/* Bind ourselves to the IRQ. Define the IPC label which is sent if an IRQ
 * IPC arrives. */
if ((err = l4_error(l4_rcv_ep_bind_thread(irqcap, l4re_env()->main_thread, 0x1234)))
{
    printf("Could not bind to IRQ%d: %ld (%s).\n", irqno, err, l4sys_errtostr(err));
    return 1;
}

printf("Attached to key IRQ %d.\nPress keys now, Shift-Q to exit.\n", irqno);

/* IRQ receive loop. */
while (1)
{
    /* Wait for the interrupt to happen. If we received an IRQ, the label
     * return code is set to 0. If we didn't receive an IRQ, the error flag
     * in the message tag is set and l4_error() reads the IPC error code from
     * the UTCB. */
    l4_umword_t label;
    if ((err = l4_error(l4_irq_wait(irqcap, &label, L4_IPC_NEVER)))
        printf("Could not receive IRQ: %ld (%s).\n", err, l4sys_errtostr(err));
    else
    {
        char buf[128];
        int n;

        if (label != 0x1234)
        {
            printf("Unexpected label %0lx -- ignoring interrupt.\n", label);
            continue;
        }

        /* Process the interrupt -- may do a 'break' */
        printf("Got IRQ with expected label 0x%lx.\n", label);
        n = l4_vcon_read(icucap, buf, sizeof(buf));
        if (n < 0)
            printf("Could not read from Vcon interface: %d (%s).\n", n, l4sys_errtostr(n));
        else
        {
            unsigned i;
            int terminate = 0;
            for (i = 0; i < (unsigned)n && i < sizeof(buf); ++i)
            {
                int c = (unsigned char)buf[i];
                if (c >= 32 && c < 128) // Filter UTF-8 encodings.
                    printf("Got key '%c'.\n", c);
                else
                    printf("Got keycode %d.\n", c);
                if (buf[i] == 'Q')
                    terminate = 1;
            }

            if (terminate)
                break;
        }
    }
}

/* We're done, detach from the interrupt. */
if ((err = l4_error(l4_irq_detach(irqcap)))

```



```

    printf("Could not detach from IRQ: %ld (%s).\n", err, l4sys_errtostr(err));

    printf("Application terminated.\n");
    return 0;
}

```

## 18.9 examples/clntsrv/src/server.cc

Client/Server example using C++ infrastructure – Server implementation.

Client/Server example using C++ infrastructure – Server implementation.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <stdio.h>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>
#include <l4/re/util/object_registry>
#include <l4/re/util/br_manager>
#include <l4/sys/cxx/ipc_epiface>

#include "shared.h"

static L4Re::Util::Registry_server<> server;

class Calculation_server : public L4::Epiface_t<Calculation_server, Calc>
{
public:
    int op_sub(Calc::Rights, l4_uint32_t a, l4_uint32_t b, l4_uint32_t &res)
    {
        res = a - b;
        return 0;
    }

    int op_neg(Calc::Rights, l4_uint32_t a, l4_uint32_t &res)
    {
        res = -a;
        return 0;
    }
};

int
main()
{
    static Calculation_server calc;

    // Register calculation server
    if (!server.registry()->register_obj(&calc, "calc_server").is_valid())
    {
        printf("Could not register my service, is there a 'calc_server' in the caps table?\n");
        return 1;
    }

    printf("Welcome to the calculation server!\n"
           "I can do subtractions and negations.\n");

    // Wait for client requests
    server.loop();

    return 0;
}

```

## 18.10 examples/clntsrv/src/client.cc

Client/Server example using C++ infrastructure – Client implementation.

Client/Server example using C++ infrastructure – Client implementation.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/sys/err.h>
#include <l4/sys/types.h>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>

#include <stdio.h>
#include "shared.h"

int
main()
{
    L4::Cap<Calc> server = L4Re::Env::env()->get_cap<Calc>("calc_server");
    if (!server.is_valid())
    {
        printf("Could not get server capability!\n");
        return 1;
    }

    l4_uint32_t val1 = 8;
    l4_uint32_t val2 = 5;

    printf("Asking for %d - %d\n", val1, val2);
    if (server->sub(val1, val2, &val1))
    {
        printf("Error talking to server\n");
        return 1;
    }
    printf("Result of subtract call: %d\n", val1);

    printf("Asking for -%d\n", val1);
    if (server->neg(val1, &val1))
    {
        printf("Error talking to server\n");
        return 1;
    }
    printf("Result of negate call: %d\n", val1);

    return 0;
}

```

## 18.11 examples/clntsrv/src/shared.h

Client/Server example using C++ infrastructure – Shared header file.

Client/Server example using C++ infrastructure – Shared header file.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */

#pragma once

#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_iface>

struct Calc : L4::Kobject_t<Calc, L4::Kobject, 0x44>
{
    L4_INLINE_RPC(int, sub, (l4_uint32_t a, l4_uint32_t b, l4_uint32_t *res));
    L4_INLINE_RPC(int, neg, (l4_uint32_t a, l4_uint32_t *res));
    typedef L4::Typeid::Rpc<sub_t, neg_t> Rpc;
};

```

## 18.12 examples/clntsrv/configs/clntsrv.cfg

Sample configuration file for the client/server example.

Sample configuration file for the client/server example.

```
-- vim:set ft=lua:

-- Include L4 functionality
local L4 = require("L4");

-- Some shortcut for less typing
local ld = L4.default_loader;

-- Channel for the two programs to talk to each other.
local calc_server = ld:new_channel();

-- The server program, getting the channel in server mode.
ld:start({ caps = { calc_server = calc_server:svr() },
         log = { "server", "blue" } },
        "rom/ex_clntsrv-server");

-- The client program, getting the 'calc_server' channel to be able to talk
-- to the server. The client will be started with a green log output.
ld:start({ caps = { calc_server = calc_server },
         log = { "client", "green" } },
        "rom/ex_clntsrv-client");
```

## 18.13 examples/libs/l4re/c/ma+rm.c

Coarse grained memory allocation, in C.

Coarse grained memory allocation, in C.

```
/*
 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */

#include <l4re/c/mem_alloc.h>
#include <l4re/c/rm.h>
#include <l4re/c/util/cap_alloc.h>
#include <l4re/sys/err.h>
#include <stdio.h>
#include <string.h>

static int allocate_mem(unsigned long size_in_bytes, unsigned long flags,
                        void **virt_addr)
{
    int r;
    l4re_ds_t ds;

    /* Allocate a free capability index for our data space */
    ds = l4re_util_cap_alloc();
    if (l4_is_invalid_cap(ds))
        return -L4_ENOMEM;

    size_in_bytes = l4_trunc_page(size_in_bytes);

    /* Allocate memory via a dataspace */
    if ((r = l4re_ma_alloc(size_in_bytes, ds, flags))
        return r;

    /* Make the dataspace visible in our address space */
    *virt_addr = 0;
    if ((r = l4re_rm_attach(virt_addr, size_in_bytes,
                           L4RE_RM_F_SEARCH_ADDR | L4RE_RM_F_RWX, ds, 0,
                           flags & L4RE_MA_SUPER_PAGES
                           ? L4_SUPERPAGESHIFT : L4_PAGESHIFT)))
    {
        /* Free dataspace again */
        l4re_util_cap_free_um(ds);
        return r;
    }
}
```

```

    }

    /* Done, virtual address is in virt_addr */
    return 0;
}

static int free_mem(void *virt_addr)
{
    int r;
    l4re_ds_t ds;

    /* Detach memory from our address space */
    if ((r = l4re_rm_detach_ds(virt_addr, &ds)))
        return r;

    /* Free memory at our memory allocator */
    l4re_util_cap_free_um(ds);

    /* All went ok */
    return 0;
}

int main(void)
{
    void *virt;

    /* Allocate memory: 16k Bytes (usually) */
    if (allocate_mem(4 * L4_PAGESIZE, 0, &virt))
        return 1;

    printf("Allocated memory.\n");

    /* Do something with the memory */
    memset(virt, 0x12, 4 * L4_PAGESIZE);

    printf("Touched memory.\n");

    /* Free memory */
    if (free_mem(virt))
        return 2;

    printf("Freed and done. Bye.\n");

    return 0;
}

```

## 18.14 examples/libs/l4re/c++/mem\_alloc/ma+rm.cc

Coarse grained memory allocation, in C++.

Coarse grained memory allocation, in C++.

```

/*
 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */

#include <l4/re/mem_alloc>
#include <l4/re/rm>
#include <l4/re/env>
#include <l4/re/dataspace>
#include <l4/re/util/cap_alloc>
#include <l4/sys/err.h>
#include <cstdio>
#include <cstring>

static int allocate_mem(unsigned long size_in_bytes, unsigned long flags,
                        void **virt_addr)
{
    int r;
    L4::Cap<L4Re::Dataspace> d;

    /* Allocate a free capability index for our data space */
    d = L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>();
    if (!d.is_valid())

```

```

    return -L4_ENOMEM;

size_in_bytes = l4_trunc_page(size_in_bytes);

/* Allocate memory via a dataspace */
if ((r = L4Re::Env::env()->mem_alloc()->alloc(size_in_bytes, d, flags)))
    return r;

/* Make the dataspace visible in our address space */
*virt_addr = 0;
if ((r = L4Re::Env::env()->rm()->attach(virt_addr, size_in_bytes,
                                       L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
                                       L4::Ipc::make_cap_rw(d), 0,
                                       flags & L4Re::Mem_alloc::Super_pages
                                       ? L4_SUPERPAGESHIFT : L4_PAGESHIFT)))
    return r;

/* Done, virtual address is in virt_addr */
return 0;
}

static int free_mem(void *virt_addr)
{
    int r;
    L4::Cap<L4Re::Dataspace> ds;

    /* Detach memory from our address space */
    if ((r = L4Re::Env::env()->rm()->detach(virt_addr, &ds)))
        return r;

    /* Release and return capability slot to allocator */
    L4Re::Util::cap_alloc.free(ds, L4Re::Env::env()->task().cap());

    /* All went ok */
    return 0;
}

int main(void)
{
    void *virt;

    /* Allocate memory: 16k Bytes (usually) */
    if (allocate_mem(4 * L4_PAGESIZE, 0, &virt))
        return 1;

    printf("Allocated memory.\n");

    /* Do something with the memory */
    memset(virt, 0x12, 4 * L4_PAGESIZE);

    printf("Touched memory.\n");

    /* Free memory */
    if (free_mem(virt))
        return 2;

    printf("Freed and done. Bye.\n");

    return 0;
}

```

## 18.15 examples/libs/l4re/c++/shared\_ds/ds\_clnt.cc

Sharing memory between applications, client side.

Sharing memory between applications, client side.

```

/*
 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */

#include <l4/re/util/cap_alloc> // L4::Cap
#include <l4/re/dataspace>     // L4Re::Dataspace
#include <l4/re/rm>             // L4::Rm
#include <l4/re/env>            // L4::Env

```

```

#include <l4/sys/cache.h>

#include <cstring>
#include <cstdio>
#include <unistd.h>

#include "interface.h"

int main()
{
    /*
     * Try to get server interface cap.
     */

    L4::Cap<My_interface> svr = L4Re::Env::env()->get_cap<My_interface>("shm");
    if (!svr.is_valid())
    {
        printf("Could not get the server capability\n");
        return 1;
    }

    /*
     * Alloc data space cap slot
     */
    L4::Cap<L4Re::Dataspace> ds = L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>();
    if (!ds.is_valid())
    {
        printf("Could not get capability slot!\n");
        return 1;
    }

    /*
     * Alloc server notifier IRQ cap slot
     */
    L4::Cap<L4::Irq> irq = L4Re::Util::cap_alloc.alloc<L4::Irq>();
    if (!irq.is_valid())
    {
        printf("Could not get capability slot!\n");
        return 1;
    }

    /*
     * Request shared data-space cap.
     */
    if (svr->get_shared_buffer(ds, irq))
    {
        printf("Could not get shared memory dataspace!\n");
        return 1;
    }

    /*
     * Attach to arbitrary region
     */
    char *addr = 0;
    int err = L4Re::Env::env()->rm()->attach(&addr, ds->size(),
                                           L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
                                           L4::Ipc::make_cap_rw(ds));

    if (err < 0)
    {
        printf("Error attaching data space: %s\n", l4sys_errtostr(err));
        return 1;
    }

    printf("Content: %s\n", addr);

    // wait a bit for the demo effect
    printf("Sleeping a bit...\n");
    sleep(1);

    /*
     * Fill in new stuff
     */
    memset(addr, 0, ds->size());
    char const * msg = "Hello from client, too!";
    printf("Setting new content in shared memory\n");
    snprintf(addr, strlen(msg)+1, msg);
    l4_cache_clean_data((unsigned long)addr,
                        (unsigned long)addr + strlen(msg) + 1);

    // notify the server
    irq->trigger();

    /*
     * Detach region containing addr, result should be Detached_ds (other results
     * only apply if we split regions etc.).
     */
    err = L4Re::Env::env()->rm()->detach(addr, 0);

```

```

if (err)
    printf("Failed to detach region\n");

/* Free objects and capabilities, just for completeness. */
L4Re::Util::cap_alloc.free(ds, L4Re::This_task);
L4Re::Util::cap_alloc.free(irq, L4Re::This_task);

return 0;
}

```

## 18.16 examples/libs/l4re/c++/shared\_ds/ds\_srv.cc

Sharing memory between applications, server/creator side.

Sharing memory between applications, server/creator side.

```

/*
 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */

#include <l4/re/env>
#include <l4/re/error_helper>
#include <l4/re/namespace>
#include <l4/re/util/cap_alloc>
#include <l4/re/util/object_registry>
#include <l4/re/dataspace>
#include <l4/cxx/ipc_server>
#include <l4/util/util.h>

#include <l4/sys/typeinfo_svr>

#include <cstring>
#include <cstdio>
#include <unistd.h>
#include <pthread.h>
#include <pthread-l4.h>
#include <thread>

#include "interface.h"

class My_server_obj : public L4::Server_object_t<L4::Kobject>
{
private:
    L4::Cap<L4Re::Dataspace> _shm;
    L4::Cap<L4::Irq> _irq;

public:
    explicit My_server_obj(L4::Cap<L4Re::Dataspace> shm, L4::Cap<L4::Irq> irq)
        : _shm(shm), _irq(irq)
    {}

    int dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios);
};

int My_server_obj::dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios)
{
    // we don't care about the original object reference, however
    // we could read out the access rights from the lowest 2 bits
    (void) obj;

    l4_msgtag_t t;
    ios » t; // extract the tag

    switch (t.label())
    {
    case L4::Meta::Protocol:
        // handle the meta protocol requests, implementing the
        // runtime dynamic type system for L4 objects.
        return L4::Util::handle_meta_request<My_interface>(ios);
    case 0:
        // since we have just one operation we have no opcode dispatch,
        // and just return the data-space and the notifier IRQ capabilities
        ios « _shm « _irq;
        return 0;
    }
}

```

```

        default:
            // every other protocol is not supported.
            return -L4_EBADPROTO;
        }
    }

class Shm_observer : public L4::Irq_handler_object
{
private:
    char *_shm;

public:
    explicit Shm_observer(char *shm)
        : _shm(shm)
    {}

    int dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios);
};

int Shm_observer::dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios)
{
    // We don't care about the original object reference, however
    // we could read out the access rights from the lowest 2 bits
    (void)obj;

    // Since we end up here in this function, we got a 'message' from the IRQ
    // that is bound to us. The 'ios' stream won't contain any valuable info.
    (void)ios;

    printf("Client sent us: %s\n", _shm);

    return 0;
}

enum
{
    DS_SIZE = 4 « 12,
};

static char *get_ds(L4::Cap<L4Re::Dataspace> *_ds)
{
    *_ds = L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>();
    if (!(*_ds).is_valid())
    {
        printf("Dataspace allocation failed.\n");
        return 0;
    }

    int err = L4Re::Env::env()->mem_alloc()->alloc(DS_SIZE, *_ds, 0);
    if (err < 0)
    {
        printf("mem_alloc->alloc() failed.\n");
        L4Re::Util::cap_alloc.free(*_ds);
        return 0;
    }

    /*
     * Attach DS to local address space
     */
    char *_addr = 0;
    err = L4Re::Env::env()->rm()->attach(&_addr, (*_ds)->size(),
                                         L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
                                         L4::Ipc::make_cap_rw(*_ds));

    if (err < 0)
    {
        printf("Error attaching data space: %s\n", l4sys_errtostr(err));
        L4Re::Util::cap_alloc.free(*_ds);
        return 0;
    }

    /*
     * Success! Write something to DS.
     */
    printf("Attached DS\n");
    static char const * const msg = "[DS] Hello from server!";
    snprintf(_addr, strlen(msg) + 1, msg);

    return _addr;
}

static void *server_thread(void *)
{
    L4::Cap<L4::Thread> l4_thread = Pthread::L4::cap(pthread_self());
    L4Re::Util::Registry_server<> server(l4_thread, L4Re::Env::env()->factory());

    L4::Cap<L4Re::Dataspace> ds;

```



```

char *addr;

if (!(addr = get_ds(&ds)))
    return nullptr;

// First the IRQ handler, because we need it in the My_server_obj object
Shm_observer observer(addr);

// Registering the observer as an IRQ handler, this allocates an
// IRQ object using the factory of our server.
L4::Cap<L4::Irq> irq = server.registry()->register_irq_obj(&observer);

// Now the initial server object shared with the client via our parent.
// it provides the data-space and the IRQ capabilities to a client.
My_server_obj server_obj(ds, irq);

// Registering the server object to the capability 'shm' in our the L4Re::Env.
// This capability must be provided by the parent. (see the shared_ds.lua)
server.registry()->register_obj(&server_obj, "shm");

// Run our server loop.
server.loop();
}

int main()
{
    pthread_attr_t pattr;

    if (pthread_attr_init(&pattr))
        L4Re::throw_error(-L4_ENOMEM, "Initialize pthread attributes");

    pthread_t thr;
    L4Re::chksys(pthread_create(&thr, &pattr, server_thread, nullptr),
        "Create server thread");
    L4Re::chksys(pthread_attr_destroy(&pattr), "Destroy pthread attributes");

    l4_sleep_forever();

    return 0;
}

```

## 18.17 examples/libs/l4re/c++/shared\_ds/shared\_ds.cfg

Sharing memory between applications, configuration file.

Sharing memory between applications, configuration file.

```

-- Include L4 functionality
local L4 = require("L4");

-- Create a channel from the client to the server
local channel = L4.default_loader:new_channel();

-- Start the server, giving the channel with full server rights.
-- The server will have a yellow log output.
L4.default_loader:start(
{
    caps = { shm = channel:svr() },
    log = { "server", "yellow" }
},
"rom/ex_l4re_ds_srv"
);

-- Start the client, giving it the channel with read only rights. The
-- log output will be green.
L4.default_loader:start(
{
    caps = { shm = channel },
    log = { "client", "green" },
    l4re_dbg = L4.Dbg.Warn
},
"rom/ex_l4re_ds_clnt"
);

```

## 18.18 examples/libs/l4re/streammap/server.cc

Client/Server example showing how to map a page to another task – Server implementation.

Client/Server example showing how to map a page to another task – Server implementation. Note that there's also a shared memory library that supplies this functionality in more convenient way.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <stdio.h>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>
#include <l4/re/util/object_registry>
#include <l4/cxx/ipc_server>

#include "shared.h"

static char page_to_map[L4_PAGESIZE] __attribute__((aligned(L4_PAGESIZE)));

static L4Re::Util::Registry_server<> server;

class Smap_server : public L4::Server_object_t<Mapper>
{
public:
    int dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios);
};

int
Smap_server::dispatch(l4_umword_t, L4::Ipc::Iostream &ios)
{
    l4_msgtag_t t;
    ios » t;

    // We're only talking the Map_example protocol
    if (t.label() != Mapper::Protocol)
        return -L4_EBADPROTO;

    L4::Opcode opcode;
    ios » opcode;

    switch (opcode)
    {
    case Mapper::Do_map:
        l4_addr_t snd_base;
        ios » snd_base;
        // put something into the page to read it out at the other side
        snprintf(page_to_map, sizeof(page_to_map), "Hello from the server!");
        printf("Sending to client\n");
        // send page
        ios « L4::Ipc::Snd_fpage::mem((l4_addr_t)page_to_map, L4_PAGESHIFT,
                                     L4_FPAGE_RO, snd_base);

        return L4_EOK;
    default:
        return -L4_ENOSYS;
    }
}

int
main()
{
    static Smap_server smap;

    // Register server
    if (!server.registry()->register_obj(&smap, "smap").is_valid())
    {
        printf("Could not register my service, read-only namespace?\n");
        return 1;
    }

    printf("Welcome to the memory map example server!\n");

    // Wait for client requests
    server.loop();

    return 0;
}

```

## 18.19 examples/libs/l4re/streammap/client.cc

Client/Server example showing how to map a page to another task – Client implementation.

Client/Server example showing how to map a page to another task – Client implementation. Note that there's also a shared memory library that supplies this functionality in more convenient way.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/sys/err.h>
#include <l4/sys/types.h>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>
#include <l4/cxx/ipc_stream>

#include <stdio.h>

#include "shared.h"

static int
func_smap_call(L4::Cap<void> const &server)
{
    L4::Ipc::Iostream s(l4_utcb());
    l4_addr_t addr = 0;
    int err;

    if ((err = L4Re::Env::env()->rm()->reserve_area(&addr, L4_PAGESIZE,
                                                    L4Re::Rm::F::Search_addr)))
    {
        printf("The reservation of one page within our virtual memory failed with %d\n", err);
        return 1;
    }

    s << L4::Opcode(Mapper::Do_map)
      << (l4_addr_t)addr;
    s << L4::Ipc::Rcv_fpage::mem((l4_addr_t)addr, L4_PAGESHIFT, 0);
    int r = l4_error(s.call(server.cap(), Mapper::Protocol));
    if (r)
        return r; // failure

    printf("String sent by server: %s\n", (char *)addr);

    return 0; // ok
}

int
main()
{
    L4::Cap<void> server = L4Re::Env::env()->get_cap<void>("smap");
    if (!server.is_valid())
    {
        printf("Could not get capability slot!\n");
        return 1;
    }

    printf("Asking for page from server\n");

    if (func_smap_call(server))
    {
        printf("Error talking to server\n");
        return 1;
    }
    printf("It worked!\n");

    L4Re::Util::cap_alloc.free(server, L4Re::This_task);

    return 0;
}

```

## 18.20 examples/libs/l4re/streammap/streammap.cfg

Sample configuration file for the client/server map example.

Sample configuration file for the client/server map example.

```
-- vim:set ft=lua:

-- Include L4 functionality
local L4 = require("L4");

-- Channel for the communication between the server and the client.
local smap_channel = L4.default_loader:new_channel();

-- The server program, using the 'smap' channel in server
-- mode. The log prefix will be 'server', colored yellow.
L4.default_loader:start({ caps = { smap = smap_channel:svr() },
                        log = { "server", "yellow" }},
                      "rom/ex_smap-server");

-- The client program.
-- It is given the 'smap' channel to be able to talk to the server.
-- The log prefix will be 'client', colored green.
L4.default_loader:start({ caps = { smap = smap_channel },
                        log = { "client", "green" }},
                      "rom/ex_smap-client");
```

## 18.21 examples/libs/libirq/loop.c

libirq usage example using a self-created thread.

libirq usage example using a self-created thread.

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */

#include <l4/irq/irq.h>
#include <l4/util/util.h>
#include <stdio.h>
#include <pthread.h>

enum { IRQ_NO = 17 };

static void isr_handler(void)
{
    printf("Got IRQ %d\n", IRQ_NO);
}

static void *isr_thread(void *data)
{
    l4irq_t *irq;
    (void)data;

    if (!(irq = l4irq_attach(IRQ_NO)))
        return NULL;

    while (1)
    {
        if (l4irq_wait(irq))
            continue;
        isr_handler();
    }

    return NULL;
}

int main(void)
{
    pthread_t thread;

    if (pthread_create(&thread, NULL, isr_thread, NULL))
        return 1;

    l4_sleep_forever();
    return 0;
}
```

## 18.22 examples/libs/libirq/async\_isr.c

libirq usage example using asynchronous ISR handler functionality.

libirq usage example using asynchronous ISR handler functionality.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
/*
 * This example shall show how to use the libirq.
 */

#include <l4/irq/irq.h>
#include <l4/util/util.h>

#include <stdio.h>

enum { IRQ_NO = 17 };

static void isr_handler(void *data)
{
    (void)data;
    printf("Got IRQ %d\n", IRQ_NO);
}

int main(void)
{
    const int seconds = 5;
    l4irq_t *irqdesc;

    if (!(irqdesc = l4irq_request(IRQ_NO, isr_handler, 0, 0xff, 0)))
    {
        printf("Requesting IRQ %d failed\n", IRQ_NO);
        return 1;
    }

    printf("Attached to key IRQ %d\nPress keys now, will terminate in %d seconds\n",
           IRQ_NO, seconds);

    l4_sleep(seconds * 1000);

    if (l4irq_release(irqdesc))
    {
        printf("Failed to release IRQ\n");
        return 1;
    }

    printf("Bye\n");
    return 0;
}

```

## 18.23 examples/sys/migrate/thread\_migrate.cc

Thread migration example.

Thread migration example.

```

/*
 * (c) 2008-2009 Author(s)
 *     economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/sys/scheduler>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>

#include <pthread-l4.h>
#include <unistd.h>

```

```

#include <stdio.h>
#include <string.h>

enum { NR_THREADS = 12 };
static L4::Cap<L4::Thread> threads[NR_THREADS];
static l4_umword_t          cpu_map, cpu_nrs;

/* Function for the threads. The content is not really relevant, so lets
 * just sleep around a bit. */
static void *thread_fn(void *)
{
    while (1)
        sleep(1);

    return 0;
}

/* Check how many CPUs we have available.
 */
static int check_cpus(void)
{
    l4_sched_cpu_set_t cs = l4_sched_cpu_set(0, 0);

    if (l4_error(L4Re::Env::env()->scheduler()->info(&cpu_nrs, &cs)) < 0)
        return 1;

    cpu_map = cs.map;

    printf("%ld maximal supported CPUs.\n", cpu_nrs);
    if (cpu_nrs >= L4_MWORD_BITS)
    {
        printf("Will only handle %ld CPUs.\n", cpu_nrs);
        cpu_nrs = L4_MWORD_BITS;
    }
    else if (cpu_nrs == 1)
        printf("Only found 1 CPU.\n");

    return cpu_nrs < 2;
}

/* Create a couple of threads and store their capabilities in an array */
static int create_threads(void)
{
    unsigned i;

    for (i = 0; i < NR_THREADS; ++i)
    {
        pthread_t t;

        if (pthread_create(&t, NULL, thread_fn, NULL))
            return 1;

        threads[i] = L4::Cap<L4::Thread>(pthread_l4_cap(t));
    }
    printf("Created %d threads.\n", NR_THREADS);
    return 0;
}

/* Helper function to get the next CPU */
static unsigned get_next_cpu(unsigned c)
{
    unsigned x = c;
    for (;;)
    {
        x = (x + 1) % cpu_nrs;
        if (L4Re::Env::env()->scheduler()->is_online(x))
            return x;
        if (x == c)
            return c;
    }
}

/* Function that shuffles the threads on the available CPUs */
static void shuffle(void)
{
    unsigned start = 0;
    while (1)
    {
        unsigned t;
        unsigned c = start;
        for (t = 0; t < NR_THREADS; ++t)
        {
            l4_sched_param_t sp = l4_sched_param(20);
            c = get_next_cpu(c);
            sp.affinity = l4_sched_cpu_set(c, 0);
            if (l4_error(L4Re::Env::env()->scheduler()->run_thread(threads[t], sp)))
                printf("Error migrating thread%02d to CPU%02d\n", t, c);
        }
        start = c;
    }
}

```

```

        printf("Migrated Thread%02d -> CPU%02d\n", t, c);
    }

    start++;
    if (start == cpu_nrs)
        start = 0;
    sleep(1);
}

int main(void)
{
    if (check_cpus())
        return 1;

    if (create_threads())
        return 1;

    shuffle();

    return 0;
}

```

## 18.24 examples/sys/migrate/thread\_migrate.cfg

Sample configuration file for the thread migration example.

Sample configuration file for the thread migration example.

```

-- vim:set ft=lua:

local L4 = require("L4");

-- The log prefix will be 'migrate', colored green.
L4.default_loader:start({ log = { "migrate", "green" } },
                        "rom/ex_thread_migrate");

```

## 18.25 tmpfs/lib/src/fs.cc

Example file system for [L4Re::Vfs](#).

Example file system for [L4Re::Vfs](#).

```

/*
 * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU Lesser General Public License 2.1.
 * Please see the COPYING-LGPL-2.1 file for details.
 */

#include <l4/l4re_vfs/backend>
#include <l4/cxx/string>
#include <l4/cxx/avl_tree>

#include <sys/stat.h>
#include <sys/ioctl.h>
#include <dirent.h>

#include <cstdio>
#include <cstdlib>
#include <cstring>

namespace {

using namespace L4Re::Vfs;
using cxx::Ref_ptr;

class File_data
{
public:

```

```

File_data() : _buf(0), _size(0) {}

unsigned long put(unsigned long offset,
                 unsigned long bufsize, void *srcbuf);
unsigned long get(unsigned long offset,
                 unsigned long bufsize, void *dstbuf);

unsigned long size(unsigned long offset);
unsigned long size() const { return _size; }

~File_data() noexcept { free(_buf); }

private:
    void *_buf;
    unsigned long _size;
};

unsigned long
File_data::put(unsigned long offset, unsigned long bufsize, void *srcbuf)
{
    if (offset + bufsize > _size)
        size(offset + bufsize);

    if (!_buf)
        return 0;

    memcpy((char *)_buf + offset, srcbuf, bufsize);
    return bufsize;
}

unsigned long
File_data::get(unsigned long offset, unsigned long bufsize, void *dstbuf)
{
    unsigned long s = bufsize;

    if (offset > _size)
        return 0;

    if (offset + bufsize > _size)
        s = _size - offset;

    memcpy(dstbuf, (char *)_buf + offset, s);
    return s;
}

unsigned long
File_data::size(unsigned long offset)
{
    if (offset != _size)
    {
        _size = offset;
        _buf = realloc(_buf, _size);
    }

    if (!_buf)
        return 0;
    return -ENOSPC;
}

class Node : public cxx::Avl_tree_node
{
public:
    Node(const char *path, mode_t mode)
        : _ref_cnt(0), _path(strdup(path))
    {
        memset(&_amp;_info, 0, sizeof(_info));
        _info.st_mode = mode;
    }

    const char *path() const { return _path; }
    struct stat64 *info() { return &_amp;_info; }

    void add_ref() noexcept { ++_ref_cnt; }
    int remove_ref() noexcept { return --_ref_cnt; }

    bool is_dir() const { return S_ISDIR(_info.st_mode); }

    virtual ~Node() { free(_path); }

private:
    int _ref_cnt;
    char *_path;
    struct stat64 _info;
};

struct Node_get_key

```



```

{
    typedef cxx::String Key_type;
    static Key_type key_of(Node const *n)
    { return n->path(); }
};

struct Path_avl_tree_compare
{
    bool operator () (const char *l, const char *r) const
    { return strcmp(l, r) < 0; }
    bool operator () (const cxx::String l, const cxx::String r) const
    {
        int v = strncmp(l.start(), r.start(), cxx::min(l.len(), r.len()));
        return v < 0 || (v == 0 && l.len() < r.len());
    }
};

class Pers_file : public Node
{
public:
    Pers_file(const char *name, mode_t mode)
        : Node(name, (mode & 0777) | __S_IFREG) {}
    File_data const &data() const { return _data; }
    File_data &data() { return _data; }
private:
    File_data _data;
};

class Pers_dir : public Node
{
private:
    typedef cxx::Avl_tree<Node, Node_get_key, Path_avl_tree_compare> Tree;
    Tree _tree;

public:
    Pers_dir(const char *name, mode_t mode)
        : Node(name, (mode & 0777) | __S_IFDIR) {}
    Ref_ptr<Node> find_path(cxx::String);
    bool add_node(Ref_ptr<Node> const &);

    typedef Tree::Const_iterator Const_iterator;
    Const_iterator begin() const { return _tree.begin(); }
    Const_iterator end() const { return _tree.end(); }
};

Ref_ptr<Node> Pers_dir::find_path(cxx::String path)
{
    return cxx::ref_ptr(_tree.find_node(path));
}

bool Pers_dir::add_node(Ref_ptr<Node> const &n)
{
    bool e = _tree.insert(n.ptr()).second;
    if (e)
        n->add_ref();
    return e;
}

class Tmpfs_dir : public Be_file
{
public:
    explicit Tmpfs_dir(Ref_ptr<Pers_dir> const &d) noexcept
        : _dir(d), _getdents_state(false) {}
    int get_entry(const char *, int, mode_t, Ref_ptr<File> *) noexcept override;
    ssize_t getdents(char *, size_t) noexcept override;
    int fstat(struct stat64 *buf) const noexcept override;
    int utime(const struct utimbuf *) noexcept override;
    int fchmod(mode_t) noexcept override;
    int mkdir(const char *, mode_t) noexcept override;
    int unlink(const char *) noexcept override;
    int rename(const char *, const char *) noexcept override;
    int faccessat(const char *, int, int) noexcept override;

private:
    int walk_path(cxx::String const &s,
                  Ref_ptr<Node> *ret, cxx::String *remaining = 0);

    Ref_ptr<Pers_dir> _dir;
    bool _getdents_state;
    Pers_dir::Const_iterator _getdents_iter;
};

class Tmpfs_file : public Be_file_pos
{
public:
    explicit Tmpfs_file(Ref_ptr<Pers_file> const &f) noexcept

```

```

    : Be_file_pos(), _file(f) {}

    off64_t size() const noexcept;
    int fstat(struct stat64 *buf) const noexcept override;
    int ftruncate(off64_t p) noexcept override;
    int ioctl(unsigned long, va_list) noexcept override;
    int utime(const struct utimbuf *) noexcept override;
    int fchmod(mode_t) noexcept override;

private:
    ssize_t preadv(const struct iovec *v, int iovcnt, off64_t p) noexcept
        override;
    ssize_t pwritev(const struct iovec *v, int iovcnt, off64_t p) noexcept
        override;
    Ref_ptr<Pers_file> _file;
};

ssize_t Tmpfs_file::preadv(const struct iovec *v, int iovcnt, off64_t p)
    noexcept
{
    if (iovcnt < 0)
        return -EINVAL;

    ssize_t sum = 0;
    for (int i = 0; i < iovcnt; ++i)
    {
        sum += _file->data().get(p, v[i].iov_len, v[i].iov_base);
        p += v[i].iov_len;
    }
    return sum;
}

ssize_t Tmpfs_file::pwritev(const struct iovec *v, int iovcnt, off64_t p)
    noexcept
{
    if (iovcnt < 0)
        return -EINVAL;

    ssize_t sum = 0;
    for (int i = 0; i < iovcnt; ++i)
    {
        sum += _file->data().put(p, v[i].iov_len, v[i].iov_base);
        p += v[i].iov_len;
    }
    return sum;
}

int Tmpfs_file::fstat(struct stat64 *buf) const noexcept
{
    _file->info()->st_size = _file->data().size();
    memcpy(buf, _file->info(), sizeof(*buf));
    return 0;
}

int Tmpfs_file::ftruncate(off64_t p) noexcept
{
    if (p < 0)
        return -EINVAL;

    if (_file->data().size(p) == 0)
        return 0;

    return -EIO; // most likely ENOSPC, but can't report that
}

off64_t Tmpfs_file::size() const noexcept
{
    return _file->data().size();
}

int
Tmpfs_file::ioctl(unsigned long v, va_list args) noexcept
{
    switch (v)
    {
        case FIONREAD: // return amount of data still available
        {
            int *available = va_arg(args, int *);
            *available = _file->data().size() - pos();
            return 0;
        }
        default:
        {
            return -EINVAL;
        }
    }
}

int
Tmpfs_file::utime(const struct utimbuf *times) noexcept
{
    _file->info()->st_atime = times->actime;
    _file->info()->st_mtime = times->modtime;
    return 0;
}

```

```

}

int
Tmpfs_file::fchmod(mode_t m) noexcept
{
    _file->info()->st_mode = m;
    return 0;
}

int
Tmpfs_dir::faccessat(const char *path, int mode, int) noexcept
{
    Ref_ptr<Node> node;
    cxx::String name = path;

    int err = walk_path(name, &node, &name);
    if (err < 0)
        return err;

    if (mode == F_OK) // existence check
        return 0;

    struct stat64 *stats = node->info();

    if ((mode & R_OK) && !(stats->st_mode & S_IRUSR))
        return -EACCES;

    if ((mode & W_OK) && !(stats->st_mode & S_IWUSR))
        return -EACCES;

    if ((mode & X_OK) && !(stats->st_mode & S_IXUSR))
        return -EACCES;

    return 0;
}

int
Tmpfs_dir::get_entry(const char *name, int flags, mode_t mode,
                    Ref_ptr<File> *file) noexcept
{
    Ref_ptr<Node> path;
    if (!*name)
    {
        *file = cxx::ref_ptr(this);
        return 0;
    }

    cxx::String n = name;

    int e = walk_path(n, &path, &n);

    if (e == -ENOTDIR)
        return e;

    if (!(flags & O_CREAT) && e < 0)
        return e;

    if ((flags & O_CREAT) && e == -ENOENT)
    {
        Ref_ptr<Node> node(new Pers_file(n.start(), mode));
        // when ENOENT is return, path is always a directory
        bool e = cxx::ref_ptr_static_cast<Pers_dir>(path)->add_node(node);
        if (!e)
            return -ENOMEM;
        path = node;
    }

    if (path->is_dir())
        *file = cxx::ref_ptr(new Tmpfs_dir(cxx::ref_ptr_static_cast<Pers_dir>(path)));
    else
        *file = cxx::ref_ptr(new Tmpfs_file(cxx::ref_ptr_static_cast<Pers_file>(path)));

    if (!*file)
        return -ENOMEM;

    return 0;
}

ssize_t
Tmpfs_dir::getdents(char *buf, size_t sz) noexcept
{
    struct dirent64 *d = (struct dirent64 *)buf;
    ssize_t ret = 0;

    if (!_getdents_state)
    {

```

```

        _getdents_iter = _dir->begin();
        _getdents_state = true;
    }
    else if (_getdents_iter == _dir->end())
    {
        _getdents_state = false;
        return 0;
    }

    for (; _getdents_iter != _dir->end(); ++_getdents_iter)
    {
        unsigned l = strlen(_getdents_iter->path()) + 1;
        if (l > sizeof(d->d_name))
            l = sizeof(d->d_name);

        unsigned n = offsetof (struct dirent64, d_name) + 1;
        n = (n + sizeof(long) - 1) & ~(sizeof(long) - 1);

        if (n > sz)
            break;

        d->d_ino = 1;
        d->d_off = 0;
        memcpy(d->d_name, _getdents_iter->path(), l);
        d->d_reclen = n;
        d->d_type = DT_REG;
        ret += n;
        sz -= n;
        d = (struct dirent64 *) ((unsigned long)d + n);
    }

    return ret;
}

int
Tmpfs_dir::fstat(struct stat64 *buf) const noexcept
{
    memcpy(buf, _dir->info(), sizeof(*buf));
    return 0;
}

int
Tmpfs_dir::utime(const struct utimbuf *times) noexcept
{
    _dir->info()->st_atime = times->actime;
    _dir->info()->st_mtime = times->modtime;
    return 0;
}

int
Tmpfs_dir::fchmod(mode_t m) noexcept
{
    _dir->info()->st_mode = m;
    return 0;
}

int
Tmpfs_dir::walk_path(cxx::String const &s,
                    Ref_ptr<Node> *ret, cxx::String *remaining)
{
    Ref_ptr<Pers_dir> p = _dir;
    cxx::String s = s;
    Ref_ptr<Node> n;

    while (1)
    {
        if (s.len() == 0)
        {
            *ret = p;
            return 0;
        }

        cxx::String::Index sep = s.find("/");

        if (sep - s.start() == 1 && *s.start() == '.')
        {
            s = s.substr(s.start() + 2);
            continue;
        }

        n = p->find_path(s.head(sep - s.start()));

        if (!n)
        {
            *ret = p;
            if (remaining)
                *remaining = s.head(sep - s.start());
        }
    }
}

```

```

        return -ENOENT;
    }

    if (sep == s.end())
    {
        *ret = n;
        return 0;
    }

    if (!n->is_dir())
        return -ENOTDIR;

    s = s.substr(sep + 1);

    p = cxx::ref_ptr_static_cast<Pers_dir>(n);
}

*ret = n;

return 0;
}

int
Tmpfs_dir::mkdir(const char *name, mode_t mode) noexcept
{
    Ref_ptr<Node> node = _dir;
    cxx::String p = cxx::String(name);
    cxx::String path, last = p;
    cxx::String::Index s = p.rfind("/");

    // trim '/'s at the end
    while (p.len() && s == p.end() - 1)
    {
        p.len(p.len() - 1);
        s = p.rfind("/");
    }

    //printf("MKDIR '%s' p=%p %p\n", name, p.start(), s);

    if (s != p.end())
    {
        path = p.head(s);
        last = p.substr(s + 1, p.end() - s);

        int e = walk_path(path, &node);
        if (e < 0)
            return e;
    }

    if (!node->is_dir())
        return -ENOTDIR;

    // due to path walking we can end up with an empty name
    if (p.len() == 0 || p == cxx::String("."))
        return 0;

    Ref_ptr<Pers_dir> dnode = cxx::ref_ptr_static_cast<Pers_dir>(node);

    Ref_ptr<Pers_dir> dir(new Pers_dir(last.start(), mode));
    return dnode->add_node(dir) ? 0 : -EEXIST;
}

int
Tmpfs_dir::unlink(const char *name) noexcept
{
    cxx::Ref_ptr<Node> n;

    int e = walk_path(name, &n);
    if (e < 0)
        return -ENOENT;

    printf("Unimplemented (if file exists): %s(%s)\n", __func__, name);
    return -ENOMEM;
}

int
Tmpfs_dir::rename(const char *old, const char *newn) noexcept
{
    printf("Unimplemented: %s(%s, %s)\n", __func__, old, newn);
    return -ENOMEM;
}

class Tmpfs_fs : public Be_file_system
{

```

```

public:
    Tmpfs_fs() : Be_file_system("tmpfs") {}
    int mount(char const *source, unsigned long mountflags,
              void const *data, cxx::Ref_ptr<File> *dir) noexcept override
    {
        (void)mountflags;
        (void)source;
        (void)data;
        *dir = cxx::ref_ptr(new Tmpfs_dir(cxx::ref_ptr(new Pers_dir("root", 0777))));
        if (!*dir)
            return -ENOMEM;
        return 0;
    }
};

static Tmpfs_fs _tmpfs L4RE_VFS_FILE_SYSTEM_ATTRIBUTE;
}

```

## 18.26 examples/libs/shmc/prodcons.c

Simple shared memory example.

Simple shared memory example.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */

/*
 * This example uses shared memory between two threads, one producer, one
 * consumer.
 */

#include <l4/shmc/shmc.h>

#include <l4/util/util.h>

#include <stdio.h>
#include <string.h>
#include <pthread-l4.h>

#include <l4/sys/thread.h>
#include <l4/sys/debugger.h>
#include <l4/sys/kip.h>
#include <l4/re/env.h>

#define LOG(args...)      printf(NAME ": " args)
#define CHK(func)         do                                \
    {                     \
        long r = (func); \
        if (r)           \
        {                \
            printf(NAME ": Failure %ld (%s) at line %d.\n", \
                  r, l4sys_errtostr(r), __LINE__); \
            return (void *)-1; \
        }                \
    } while (0)

static const char some_data[] = "Hi consumer!";

static inline l4_cap_idx_t self(void) { return pthread_l4_cap(pthread_self()); }

#define NAME "PRODUCER"
static void *thread_producer(void *d)
{
    (void)d;
    l4shmc_chunk_t p_one;
    l4shmc_signal_t s_one, s_done;
    l4shmc_area_t shmarea;
    l4_kernel_clock_t try_until;

    l4_debugger_set_object_name(self(), "producer");

    // attach this thread to the shm object

```

```

CHK(l4shmc_attach("testshm", &shmarea));

// add a chunk
CHK(l4shmc_add_chunk(&shmarea, "one", 1024, &p_one));

// add a signal
CHK(l4shmc_add_signal(&shmarea, "testshm_prod", &s_one));

CHK(l4shmc_attach_signal(&shmarea, "testshm_done", self(), &s_done));

// connect chunk and signal
CHK(l4shmc_connect_chunk_signal(&p_one, &s_one));

CHK(l4shmc_mark_client_initialized(&shmarea));

try_until = l4_kip_clock(l4re_kip()) + 10 * 1000000;

for (;;)
{
    l4_umword_t clients;
    l4shmc_get_initialized_clients(&shmarea, &clients);
    if (clients == 3UL)
        break;
    if (l4_kip_clock(l4re_kip()) >= try_until)
    {
        LOG("consumer not initialized within time\n");
        return (void *)-1;
    }
}

LOG("Ready.\n");

while (1)
{
    while (l4shmc_chunk_try_to_take(&p_one))
        printf("Uh, should not happen!\n"); //l4_thread_yield();

    memcpy(l4shmc_chunk_ptr(&p_one), some_data, sizeof(some_data));

    CHK(l4shmc_chunk_ready_sig(&p_one, sizeof(some_data)));

    LOG("Sent data.\n");

    CHK(l4shmc_wait_signal(&s_done));
}

l4_sleep_forever();
return NULL;
}

#undef NAME
#define NAME "CONSUMER"
static void *thread_consumer(void *d)
{
    (void)d;
    l4shmc_area_t shmarea;
    l4shmc_chunk_t p_one;
    l4shmc_signal_t s_one, s_done;
    l4_kernel_clock_t try_until;

    l4_debugger_set_object_name(self(), "consumer");

    // attach to shared memory area
    CHK(l4shmc_attach("testshm", &shmarea));

    // get chunk 'one'
    CHK(l4shmc_get_chunk(&shmarea, "one", &p_one));

    // add a signal
    CHK(l4shmc_add_signal(&shmarea, "testshm_done", &s_done));

    // attach signal to this thread
    CHK(l4shmc_attach_signal(&shmarea, "testshm_prod", self(), &s_one));

    // connect chunk and signal
    CHK(l4shmc_connect_chunk_signal(&p_one, &s_one));

    CHK(l4shmc_mark_client_initialized(&shmarea));

    try_until = l4_kip_clock(l4re_kip()) + 10 * 1000000;

    for (;;)
    {
        l4_umword_t clients;
        l4shmc_get_initialized_clients(&shmarea, &clients);
        if (clients == 3UL)

```

```
        break;
    if (l4_kip_clock(l4re_kip()) >= try_until)
    {
        LOG("producer not initialized within time\n");
        return (void *)-1;
    }
}

LOG("Ready.\n");

while (1)
{
    CHK(l4shmc_wait_chunk(&p_one));

    LOG("Received from chunk one: '%s'.\n",
        (char *)l4shmc_chunk_ptr(&p_one));
    memset(l4shmc_chunk_ptr(&p_one), 0, l4shmc_chunk_size(&p_one));

    CHK(l4shmc_chunk_consumed(&p_one));

    CHK(l4shmc_trigger(&s_done));
}

return NULL;
}

int main(void)
{
    pthread_t one, two;
    long r;

    // create shared memory area
    if ((r = l4shmc_create("testshm")) < 0)
    {
        printf("Error %ld (%s) creating shared memory area\n",
            r, l4sys_errtostr(r));
        return 1;
    }

    // create two threads, one for producer, one for consumer
    pthread_create(&one, 0, thread_producer, 0);
    pthread_create(&two, 0, thread_consumer, 0);

    // now sleep, the two threads are doing the work
    l4_sleep_forever();

    return 0;
}
```



# Index

%L4 Inter-Process Communication (IPC), [9](#)

\_\_lface

L4::Kobject\_2t< Derived, Base1, Base2, PROTO, S\_DEMAND >, [1324](#)

L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO, S\_DEMAND >, [1328](#)

\_\_lface\_list

L4::Kobject\_2t< Derived, Base1, Base2, PROTO, S\_DEMAND >, [1324](#)

L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO, S\_DEMAND >, [1328](#)

\_\_L4UTIL\_THREAD\_FUNC

thread.h, [3401](#)

\_\_check\_protocols\_\_

L4::Kobject\_2t< Derived, Base1, Base2, PROTO, S\_DEMAND >, [1325](#)

L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO, S\_DEMAND >, [1329](#)

\_\_kdebug\_3\_text

kdebug.h, [3210](#)

\_\_kdebug\_op

kdebug.h, [3211](#)

\_\_kdebug\_op\_1

kdebug.h, [3213](#)

\_\_kdebug\_text

kdebug.h, [3214](#)

\_\_vcpu-arch.h

L4\_vcpu\_e\_field\_ids, [2489](#), [2493](#)

L4\_VCPU\_E\_VTMR\_CFG, [2490](#), [2493](#)

L4\_VCPU\_STATE\_VERSION, [2487](#), [2489](#), [2493](#), [2496](#)

~Be\_file\_system

L4Re::Vfs::Be\_file\_system, [1864](#)

~H\_list\_item\_t

cxx::H\_list\_item\_t< ELEM\_TYPE >, [945](#)

~Lock\_guard

L4::Lock\_guard, [1340](#)

~S

L4::Factory::S, [1105](#)

~Unique\_region

L4Re::Rm::Unique\_region< T >, [1762](#)

Rm::Unique\_region< T >, [2343](#)

a

L4Re::Video::Pixel\_info, [1928](#)

access\_once

cxx, [743](#)

Access\_width

L4Re::Mmio\_space, [1707](#)

acquire

L4Re::Inhibitor, [1676](#)

add

L4::lpc\_svr::Timeout\_queue, [1278](#)

L4::Thread::Modify\_senders, [1456](#)

L4::Thread\_group, [1460](#)

L4vcpu::State, [2031](#)

L4virtio::Svr::Driver\_mem\_list\_t< DATA >, [2190](#)

Virtio\_vlan\_mangle, [2363](#)

add\_block

L4virtio::Driver::Block\_device, [2058](#)

add\_image\_info

L4::Debugger, [1064](#)

add\_irq\_status

L4virtio::Svr::Dev\_config, [2169](#)

add\_ku\_mem

L4::Task, [1427](#)

add\_monitor\_port

Virtio\_switch, [2359](#)

add\_port

Virtio\_switch, [2359](#)

add\_timeout

L4::lpc\_svr::Server\_iface, [1268](#)

L4::lpc\_svr::Timeout\_queue\_hooks< BR\_MAN >, [1284](#)      HOOKS,

add\_trusted\_dataspaces

L4virtio::Svr::Device\_t< DATA >, [2184](#)

AHCI driver, [94](#)

alien

L4::Thread::Attr, [1452](#)

align\_to

L4::lpc::Msg, [768](#)

all

L4::Kip::Mem\_desc, [1311](#), [1312](#)

alloc

cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc >, [858](#)

cxx::Base\_slab\_static< Obj\_size, Slab\_size, Max\_free, Alloc >, [863](#)

cxx::List\_alloc, [954](#)

cxx::Slab< Type, Slab\_size, Max\_free, Alloc >, [982](#)

cxx::Slab\_static< Type, Slab\_size, Max\_free, Alloc >, [987](#)

L4Re::Cap\_alloc, [1609](#), [1610](#)

L4Re::Mem\_alloc, [1699](#)

L4Re::Util::Cap\_alloc, [1768](#)

L4Re::Util::Counting\_cap\_alloc< COUNTER-TYPE, Dbg >, [1802](#), [1803](#)

alloc\_buffer\_demand

- L4::lpc\_svr::Br\_manager\_no\_buffers, [1248](#)
  - L4::lpc\_svr::Server\_iface, [1268](#)
  - L4Re::Util::Br\_manager, [1787](#)
- alloc\_descriptor
  - L4virtio::Driver::Virtqueue, [2091](#)
- alloc\_dynamic\_entry
  - L4Re::Util::Names::Name\_space, [1832](#)
- alloc\_fd
  - L4Re::Vfs::Fs, [1878](#)
- alloc\_max
  - cxx::List\_alloc, [954](#)
- allocate
  - L4Re::Dataspace, [1621](#)
  - L4Re::Util::Dataspace\_svr, [1807](#)
- AMD64 Virtual Registers (UTCB), [282](#)
- amd64/l4/sys/\_\_kip-arch.h, [2484](#)
- amd64/l4/sys/\_\_vcpu-arch.h, [2485](#), [2487](#)
- amd64/l4/sys/cache.h, [3072](#), [3073](#)
- amd64/l4/sys/consts.h, [3090](#), [3091](#)
- amd64/l4/sys/ktrace\_events.h, [2497](#)
- amd64/l4/sys/l4int.h, [3235](#), [3236](#)
- amd64/l4/sys/linkage.h, [2509](#), [2510](#)
- amd64/l4/sys/segment.h, [2463](#), [2468](#)
- amd64/l4/sys/utcb.h, [3293](#), [3294](#)
- amd64/l4/sys/vm.h, [2514](#)
- amd64/l4/util/bitops\_arch.h, [2516](#), [2517](#)
- amd64/l4/util/cpu.h, [2524](#), [2525](#)
- amd64/l4/util/irq.h, [2666](#), [2667](#)
- amd64/l4/util/l4\_macros.h, [2529](#)
- amd64/l4/util/mbi\_argv.h, [2532](#), [2533](#)
- amd64/l4/util/perform.h, [2441](#), [2442](#)
- amd64/l4/util/port\_io.h, [2477](#)
- amd64/l4/util/rdtsc.h, [2453](#), [2454](#)
- amd64/l4/util/spin.h, [2461](#), [2462](#)
- amd64/l4f/l4/sys/ipc.h, [3190](#)
- amd64/l4f/l4/sys/segment.h, [2469](#), [2472](#)
- amd64/l4f/l4/util/port\_io.h, [2478](#)
- Application and Server Building Blocks, [30](#)
- Arch
  - L4::Kip::Mem\_desc, [1309](#)
- Arch\_acpi\_nvs
  - L4::Kip::Mem\_desc, [1309](#)
- Arch\_acpi\_tables
  - L4::Kip::Mem\_desc, [1308](#)
- Arch\_sub\_type\_common
  - L4::Kip::Mem\_desc, [1308](#)
- ARM Virtual Registers (UTCB), [280](#)
- arm/l4/sys/\_\_kip-arch.h, [2485](#)
- arm/l4/sys/\_\_vcpu-arch.h, [2488](#), [2490](#)
- arm/l4/sys/atomic.h, [3324](#)
- arm/l4/sys/cache.h, [3074](#), [3075](#)
- arm/l4/sys/consts.h, [3091](#), [3092](#)
- arm/l4/sys/ktrace\_events.h, [2500](#)
- arm/l4/sys/l4int.h, [3236](#)
- arm/l4/sys/linkage.h, [2510](#), [2511](#)
- arm/l4/sys/mem\_op.h, [2512](#), [2513](#)
- arm/l4/sys/platform\_control.h, [3253](#)
- arm/l4/sys/task.h, [3278](#)
- arm/l4/sys/thread.h, [3388](#), [3389](#)
- arm/l4/sys/utcb.h, [3295](#), [3297](#)
- arm/l4/sys/vm, [3316](#)
- arm/l4/sys/vm.h, [2514](#), [2515](#)
- arm/l4/util/bitops\_arch.h, [2520](#), [2521](#)
- arm/l4/util/cpu.h, [2526](#)
- arm/l4/util/irq.h, [2668](#)
- arm/l4/util/l4\_macros.h, [2529](#), [2530](#)
- arm/l4/util/mbi\_argv.h, [2533](#), [2534](#)
- arm/l4f/l4/sys/ipc.h, [3191](#)
- arm/l4f/l4/sys/syscall\_defs.h, [2536](#)
- ARM64 Virtual Registers (UTCB), [281](#)
- arm64/l4/sys/\_\_kip-arch.h, [2485](#)
- arm64/l4/sys/\_\_vcpu-arch.h, [2491](#), [2494](#)
- arm64/l4/sys/cache.h, [3076](#), [3078](#)
- arm64/l4/sys/consts.h, [3092](#), [3093](#)
- arm64/l4/sys/ktrace\_events.h, [2503](#)
- arm64/l4/sys/l4int.h, [3237](#)
- arm64/l4/sys/linkage.h, [2511](#)
- arm64/l4/sys/platform\_control.h, [3253](#)
- arm64/l4/sys/task.h, [3278](#)
- arm64/l4/sys/thread.h, [3389](#), [3390](#)
- arm64/l4/sys/utcb.h, [3298](#), [3299](#)
- arm64/l4/sys/vm, [3316](#)
- arm64/l4/sys/vm.h, [2516](#)
- arm64/l4f/l4/sys/ipc.h, [3192](#)
- arm\_smccc.h
  - l4\_arm\_smccc\_call, [3070](#)
- assert.h
  - l4\_assert, [3320](#)
- assign\_dma\_domain
  - L4vbus::Vbus, [2024](#), [2025](#)
- associate
  - L4Re::Dma\_space, [1641](#)
- AT\_BASE
  - ELF binary format, [701](#)
- AT\_EGID
  - ELF binary format, [701](#)
- AT\_ENTRY
  - ELF binary format, [701](#)
- AT\_EUID
  - ELF binary format, [701](#)
- AT\_EXECFD
  - ELF binary format, [701](#)
- AT\_FLAGS
  - ELF binary format, [701](#)
- AT\_GID
  - ELF binary format, [701](#)
- AT\_IGNORE
  - ELF binary format, [701](#)
- AT\_L4\_AUX
  - ELF binary format, [701](#)
- AT\_L4\_ENV
  - ELF binary format, [701](#)
- AT\_NOTELF
  - ELF binary format, [701](#)
- AT\_NULL
  - ELF binary format, [701](#)

- AT\_PAGESZ
  - ELF binary format, [701](#)
- AT\_PHDR
  - ELF binary format, [701](#)
- AT\_PHENT
  - ELF binary format, [701](#)
- AT\_PHNUM
  - ELF binary format, [701](#)
- AT\_UID
  - ELF binary format, [701](#)
- Atomic Instructions, [667](#)
  - l4util\_add16, [669](#)
  - l4util\_add16\_res, [669](#)
  - l4util\_add32, [669](#)
  - l4util\_add32\_res, [670](#)
  - l4util\_add8, [670](#)
  - l4util\_add8\_res, [670](#)
  - l4util\_and16, [671](#)
  - l4util\_and16\_res, [671](#)
  - l4util\_and32, [671](#)
  - l4util\_and32\_res, [672](#)
  - l4util\_and8, [672](#)
  - l4util\_and8\_res, [672](#)
  - l4util\_atomic\_add, [673](#)
  - l4util\_atomic\_inc, [673](#)
  - l4util\_cmpxchg, [673](#)
  - l4util\_cmpxchg16, [674](#)
  - l4util\_cmpxchg32, [674](#)
  - l4util\_cmpxchg8, [675](#)
  - l4util\_dec16, [675](#)
  - l4util\_dec16\_res, [676](#)
  - l4util\_dec32, [676](#)
  - l4util\_dec32\_res, [676](#)
  - l4util\_dec8, [676](#)
  - l4util\_dec8\_res, [677](#)
  - l4util\_inc16, [677](#)
  - l4util\_inc16\_res, [677](#)
  - l4util\_inc32, [677](#)
  - l4util\_inc32\_res, [678](#)
  - l4util\_inc8, [678](#)
  - l4util\_inc8\_res, [678](#)
  - l4util\_or16, [678](#)
  - l4util\_or16\_res, [679](#)
  - l4util\_or32, [679](#)
  - l4util\_or32\_res, [679](#)
  - l4util\_or8, [680](#)
  - l4util\_or8\_res, [680](#)
  - l4util\_sub16, [680](#)
  - l4util\_sub16\_res, [681](#)
  - l4util\_sub32, [681](#)
  - l4util\_sub32\_res, [681](#)
  - l4util\_sub8, [682](#)
  - l4util\_sub8\_res, [682](#)
  - l4util\_xchg, [682](#)
  - l4util\_xchg16, [683](#)
  - l4util\_xchg32, [683](#)
  - l4util\_xchg8, [683](#)
- atomic\_clear\_bit
  - cxx::Bitmap\_base, [884](#)
  - L4Re::Util::Bitmap\_base, [1777](#)
- atomic\_get\_and\_clear
  - cxx::Bitmap\_base, [884](#)
  - L4Re::Util::Bitmap\_base, [1777](#)
- atomic\_get\_and\_set
  - cxx::Bitmap\_base, [885](#)
  - L4Re::Util::Bitmap\_base, [1777](#)
- atomic\_set\_bit
  - cxx::Bitmap\_base, [885](#)
  - L4Re::Util::Bitmap\_base, [1778](#)
- attach
  - L4Re::Rm, [1737](#), [1739](#)
  - L4Re::Util::Event\_buffer\_t< PAYLOAD >, [1819](#)
  - Rm, [2327](#), [2328](#)
- Attach\_flags
  - L4Re::Rm::F, [1757](#)
  - Rm::F, [2338](#)
- Attach\_mask
  - L4Re::Rm::F, [1757](#)
  - Rm::F, [2339](#)
- Attr
  - L4::Thread::Attr, [1452](#)
- Attribute
  - L4Re::Dma\_space, [1640](#)
- Attributes
  - L4Re::Dma\_space, [1640](#)
- atype
  - Elf32\_Auxv, [1008](#)
  - Elf64\_Auxv, [1018](#)
- Auxiliary data, [600](#)
- avail
  - cxx::List\_alloc, [955](#)
- avail\_align
  - L4virtio::Virtqueue, [2278](#)
- avail\_size
  - L4virtio::Virtqueue, [2278](#)
- Avl\_map
  - cxx::Avl\_map< KEY\_TYPE, DATA\_TYPE, COMPARE, ALLOC >, [838](#)
- ax
  - l4\_vcpu\_regs\_t, [1571](#)
- b
  - L4Re::Video::Pixel\_info, [1928](#), [1929](#)
- backtrace.h
  - l4util\_backtrace, [3334](#)
- Bad\_address
  - L4virtio::Svr::Bad\_descriptor, [2102](#)
- Bad\_descriptor
  - L4virtio::Svr::Bad\_descriptor, [2102](#)
- Bad\_flags
  - L4virtio::Svr::Bad\_descriptor, [2102](#)
- Bad\_next
  - L4virtio::Svr::Bad\_descriptor, [2102](#)
- Bad\_rights
  - L4virtio::Svr::Bad\_descriptor, [2102](#)
- Bad\_size
  - L4virtio::Svr::Bad\_descriptor, [2102](#)

- Base API, [157](#)
- Base\_avl\_set
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >, [900](#), [901](#)
- Basic Macros, [160](#)
  - I4\_align\_stack\_for\_direct\_fncall, [164](#)
  - L4\_EXPORT, [162](#)
  - L4\_HIDDEN, [162](#)
  - I4\_infinite\_loop, [165](#)
  - L4\_NOTHROW, [163](#)
- Be\_file\_system
  - L4Re::Vfs::Be\_file\_system, [1864](#)
- begin
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >, [902](#)
  - cxx::Bits::Bst< Node, Get\_key, Compare >, [917](#), [918](#)
- Bidirectional
  - L4Re::Dma\_space, [1641](#)
- bind
  - L4::lcu, [1113](#)
  - L4::lommu, [1130](#)
  - L4::Thread::Attr, [1452](#)
- bind\_notification\_irq
  - L4virtio::Driver::Device, [2066](#)
- bind\_rx\_notification\_irq
  - L4virtio::Driver::Virtio\_net\_device, [2080](#)
- bind\_snd\_destination
  - L4::Rcv\_endpoint, [1371](#)
- bind\_thread
  - L4::Rcv\_endpoint, [1372](#)
- bind\_vcpu
  - L4::Irq, [1291](#)
- bit
  - cxx::Bitmap\_base, [886](#)
  - L4Re::Util::Bitmap\_base, [1778](#)
- Bit Manipulation, [685](#)
  - I4util\_bsf, [686](#)
  - I4util\_bsr, [687](#)
  - I4util\_btc, [687](#)
  - I4util\_btr, [687](#)
  - I4util\_bts, [689](#)
  - I4util\_clear\_bit, [690](#)
  - I4util\_complement\_bit, [690](#)
  - I4util\_find\_first\_set\_bit, [691](#)
  - I4util\_find\_first\_zero\_bit, [691](#)
  - I4util\_next\_power2, [691](#)
  - I4util\_set\_bit, [692](#)
  - I4util\_test\_bit, [692](#)
- bit\_index
  - cxx::Bitmap\_base, [887](#)
  - L4Re::Util::Bitmap\_base, [1779](#)
- Bitmap graphics and fonts, [657](#)
- bitmap.h
  - gfxbitmap\_bmap, [2830](#)
  - gfxbitmap\_color\_pix\_t, [2829](#)
  - gfxbitmap\_color\_t, [2829](#)
  - gfxbitmap\_convert\_color, [2830](#)
  - gfxbitmap\_copy, [2830](#)
  - gfxbitmap\_fill, [2831](#)
  - gfxbitmap\_set, [2832](#)
- Bits
  - cxx::Bitfield< T, LSB, MSB >, [867](#)
- bits\_per\_pixel
  - L4Re::Video::Pixel\_info, [1929](#)
- Bits\_type
  - cxx::Bitfield< T, LSB, MSB >, [867](#)
- Block\_dev\_base
  - L4virtio::Svr::Block\_dev\_base< Ds\_data >, [2107](#)
- Block\_device::Device\_discard\_feature, [807](#)
- Block\_device::Device\_mgr< DEV, FACTORY, SCHEDULER >, [808](#)
  - parse\_device\_name, [809](#)
- Block\_device::Device\_with\_notification\_domain< DEV >, [810](#)
- Block\_device::Dma\_region\_info, [811](#)
- Block\_device::Errand::Errand, [811](#)
  - expired, [814](#)
- Block\_device::Errand::Poll\_errand, [815](#)
  - expired, [817](#)
- Block\_device::Impl::Partitioned\_device\_discard\_mixin< PART\_DEV, BASE\_DEV, bool >, [818](#)
- Block\_device::Impl::Partitioned\_device\_discard\_mixin< PART\_DEV, BASE\_DEV, true >, [819](#)
- Block\_device::Inout\_block, [821](#)
- Block\_device::Inout\_memory< DEV >, [822](#)
- Block\_device::Mem\_region\_info, [823](#)
- Block\_device::Notification\_domain, [823](#)
- Block\_device::Partition\_info, [824](#)
- Block\_device::Partition\_reader< DEV >, [825](#)
- Block\_device::Partitioned\_device< BASE\_DEV >, [826](#)
- Block\_device::Pending\_request, [828](#)
  - fail\_request, [828](#)
  - handle\_request, [828](#)
- Block\_device::Rr\_scheduler< DEV >, [829](#)
- Block\_device::Scheduler\_base< DEV >, [831](#)
- Bootloader
  - L4::Kip::Mem\_desc, [1309](#)
- Bootstrap, the %L4 kernel bootstrapper, [99](#)
- bp
  - I4\_vcpu\_regs\_t, [1571](#)
- Br\_bytes
  - L4::lpc::Msg, [768](#)
- buf
  - L4Re::Util::Event\_buffer\_t< PAYLOAD >, [1819](#)
- buf\_cp\_in
  - L4::lpc, [760](#)
- buf\_cp\_out
  - L4::lpc, [761](#)
- buf\_in
  - L4::lpc, [761](#)
- Buffer, [832](#)
- buffer
  - L4Re::Util::Event\_t< PAYLOAD >, [1825](#)
- Buffer Registers (BRs), [279](#)
  - L4\_BDR\_IO\_SHIFT, [280](#)

- L4\_BDR\_MEM\_SHIFT, 280
  - L4\_BDR\_OBJ\_SHIFT, 280
  - l4\_buffer\_desc\_consts\_t, 279
- Bufferable
  - L4Re::Dataspace::F, 1633
- Buffered
  - L4::lpc::Snd\_fpage, 1219
- bus\_cap
  - L4vbus::Device, 1974
- bx
  - l4\_vcpu\_regs\_t, 1572
- bytes\_per\_pixel
  - L4Re::Video::Pixel\_info, 1930
- c
  - L4::Kobject\_2t< Derived, Base1, Base2, PROTO, S\_DEMAND >, 1325
  - L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO, S\_DEMAND >, 1329
- C++ Exceptions, 607
- C++ IPC Interface Definition., 199
- C\_bits
  - cx::Bitmap\_base, 884
  - L4Re::Util::Bitmap\_base, 1777
- Cache Consistency, 200
  - l4\_cache\_clean\_data, 201
  - l4\_cache\_coherent, 201
  - l4\_cache\_dma\_coherent, 202
  - l4\_cache\_flush\_data, 202
  - l4\_cache\_inv\_data, 203
- Cache\_buffered
  - L4Re::Rm::F, 1758
  - Rm::F, 2339
- Cache\_normal
  - L4Re::Rm::F, 1758
  - Rm::F, 2339
- Cache\_uncached
  - L4Re::Rm::F, 1758
  - Rm::F, 2339
- Cacheable
  - L4Re::Dataspace::F, 1633
- Cached
  - L4::lpc::Snd\_fpage, 1219
- Cacheopt
  - L4::lpc::Snd\_fpage, 1219
- Caching\_mask
  - L4Re::Dataspace::F, 1633
  - L4Re::Rm::F, 1758
  - Rm::F, 2339
- Caching\_shift
  - L4Re::Dataspace::F, 1632
  - L4Re::Rm, 1737
  - Rm, 2327
- call
  - L4::Arm\_smccc, 1029
  - L4::lpc::lostream, 1155
- Cap
  - L4::Cap< T >, 1042
  - L4::lpc::Cap< T >, 1145
- cap
  - L4::Cap\_base, 1049
  - L4::Invalid\_capability, 1123
  - L4::Kobject, 1319
- cap\_alloc
  - L4Re Capability API, 594
- Cap\_base
  - L4::Cap\_base, 1047, 1048
- cap\_cast
  - L4, 752, 753
- cap\_dynamic\_cast
  - L4, 754
- cap\_equal
  - L4::Task, 1428
- Cap\_mask
  - L4::lpc::Cap< T >, 1145
- cap\_received
  - L4::lpc::Snd\_fpage, 1222
- cap\_reinterpret\_cast
  - L4, 755, 756
- Cap\_type
  - L4::Cap\_base, 1046
- cap\_valid
  - L4::Task, 1429
- Capabilities, 450
  - L4\_BASE\_ARM\_SMCCC\_CAP, 452
  - L4\_BASE\_CAPS\_LAST, 452
  - L4\_BASE\_DEBUGGER\_CAP, 452
  - L4\_BASE\_FACTORY\_CAP, 452
  - L4\_BASE\_ICU\_CAP, 452
  - L4\_BASE\_IOMMU\_CAP, 452
  - L4\_BASE\_LOG\_CAP, 452
  - L4\_BASE\_PAGER\_CAP, 452
  - L4\_BASE\_SCHEDULER\_CAP, 452
  - L4\_BASE\_TASK\_CAP, 452
  - L4\_BASE\_THREAD\_CAP, 452
  - l4\_cap\_consts\_t, 451
  - l4\_cap\_idx\_t, 451
  - L4\_CAP\_MASK, 452
  - L4\_CAP\_OFFSET, 452
  - L4\_CAP\_SHIFT, 452
  - L4\_CAP\_SIZE, 452
  - l4\_capability\_equal, 453
  - l4\_default\_caps\_t, 452
  - L4\_INVALID\_CAP, 452
  - l4\_is\_invalid\_cap, 453
  - l4\_is\_valid\_cap, 453
- Capabilities and Naming, 25
- capability
  - L4\_DISABLE\_COPY, 3084
- Capability allocator, 575
  - l4re\_util\_cap\_last, 576
- Caps
  - L4::Type\_info::Demand\_t< CAPS, FLAGS, MEM, PORTS >, 1474
- caps
  - l4re\_env\_t, 1950
- cast

- L4vcpu::Vcpu, [2035](#)
- cfg\_read
  - L4vbus::Pci\_dev, [2006](#)
  - L4vbus::Pci\_host\_bridge, [2012](#)
- cfg\_write
  - L4vbus::Pci\_dev, [2006](#)
  - L4vbus::Pci\_host\_bridge, [2012](#)
- change\_mode
  - L4::Uart, [1518](#)
  - L4::Uart\_apb, [1524](#)
- change\_queue\_config
  - L4virtio::Svr::Dev\_config, [2169](#)
- char\_avail
  - L4::Uart, [1518](#)
  - L4::Uart\_apb, [1525](#)
- check\_castable\_from
  - L4::Cap< T >, [1043](#)
- check\_convertible\_from
  - L4::Cap< T >, [1043](#)
- check\_ports
  - Virtio\_switch, [2360](#)
- check\_ready
  - L4Re::Vfs::Be\_file, [1861](#)
  - L4Re::Vfs::Generic\_file, [1884](#)
- check\_size
  - L4::lpc::Msg, [769](#), [770](#)
- chkcap
  - L4Re, [782](#)
- chkipc
  - L4Re, [784](#)
- chksys
  - L4Re, [785](#), [786](#)
- Chunks, [617](#)
  - l4shmc\_add\_chunk, [618](#)
  - l4shmc\_chunk\_capacity, [619](#)
  - l4shmc\_chunk\_ptr, [619](#)
  - l4shmc\_chunk\_signal, [619](#)
  - l4shmc\_get\_chunk, [620](#)
  - l4shmc\_get\_chunk\_to, [620](#)
  - l4shmc\_iterate\_chunk, [621](#)
- clamp
  - Small C++ Template Library, [646](#)
- Class
  - L4::Kobject\_2t< Derived, Base1, Base2, PROTO, S\_DEMAND >, [1325](#)
  - L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO, S\_DEMAND >, [1328](#)
- clear
  - cxx::Bits::Basic\_list< POLICY >, [911](#)
  - L4::Types::Flags< BITS\_ENUM, UNDERLYING >, [1503](#)
  - L4drivers::Register\_tmpl< BITS, BLOCK >, [1601](#)
  - L4Re::Dataspace, [1622](#)
  - L4Re::Util::Dataspace\_svr, [1808](#)
  - L4vcpu::State, [2031](#)
- clear\_bit
  - cxx::Bitmap\_base, [888](#)
  - L4Re::Util::Bitmap\_base, [1779](#)
- Coherent
  - L4Re::Dma\_space, [1641](#)
- Color\_component
  - L4Re::Video::Color\_component, [1908](#)
- Com\_error
  - L4::Com\_error, [1060](#)
- Comfortable Command Line Parsing, [721](#)
  - parse\_cmdline, [722](#)
- Compound
  - L4::lpc::Snd\_fpage, [1220](#)
- config\_get
  - L4vbus::Gpio\_pin, [1992](#)
- config\_pad
  - L4vbus::Gpio\_module, [1984](#)
  - L4vbus::Gpio\_pin, [1992](#)
- config\_pull
  - L4vbus::Gpio\_pin, [1994](#)
- config\_queue
  - L4virtio::Device, [2049](#)
  - L4virtio::Driver::Device, [2067](#)
- Cons, the Console Multiplexer, [91](#)
- Console API, [596](#)
- console\_multiport\_bfm\_t
  - L4virtio::Svr::Console::Features, [2138](#)
- Console\_port
  - L4virtio::Svr::Console::Control\_message, [2115](#)
- console\_size\_bfm\_t
  - L4virtio::Svr::Console::Features, [2138](#)
- consumed
  - L4virtio::Svr::Virtqueue, [2264](#)
- Consumer, [625](#), [634](#)
  - l4shmc\_chunk\_consumed, [625](#)
  - l4shmc\_chunk\_size, [626](#)
  - l4shmc\_chunk\_try\_to\_take\_for\_reading, [626](#)
  - l4shmc\_enable\_chunk, [627](#)
  - l4shmc\_enable\_signal, [635](#)
  - l4shmc\_is\_chunk\_ready, [627](#)
  - l4shmc\_wait\_any, [636](#)
  - l4shmc\_wait\_any\_to, [636](#)
  - l4shmc\_wait\_any\_try, [637](#)
  - l4shmc\_wait\_chunk, [627](#)
  - l4shmc\_wait\_chunk\_to, [629](#)
  - l4shmc\_wait\_chunk\_try, [629](#)
  - l4shmc\_wait\_signal, [637](#)
  - l4shmc\_wait\_signal\_to, [637](#)
  - l4shmc\_wait\_signal\_try, [638](#)
- contains
  - L4virtio::Svr::Driver\_mem\_region\_t< DATA >, [2198](#)
- Continue
  - L4::lpc::Snd\_fpage, [1219](#)
- Continuous
  - L4Re::Mem\_alloc, [1699](#)
- contrib/libio-io/l4/io/io.h, [2536](#), [2539](#)
- contrib/libio-io/l4/io/types.h, [2810](#)
- control
  - L4::Thread, [1439](#)
- Conventional

- L4::Kip::Mem\_desc, 1309
- copy
  - L4::Cap< T >, 1043
  - L4::Cap\_base, 1050
  - L4Re::Util::Dataspace\_svr, 1808
- copy\_in
  - L4Re::Dataspace, 1623
- copy\_pkt
  - Virtio\_vlan\_mangle, 2363
- copy\_receive\_cap
  - L4Re::Util::Names::Name\_space, 1832
- copy\_to
  - L4virtio::Svr::Data\_buffer, 2163
- count
  - L4::Kip::Mem\_desc, 1312, 1313
- Counting\_cap\_alloc
  - L4Re::Util::Counting\_cap\_alloc< COUNTER-  
TYPE, Dbg >, 1802
- CPU related functions, 658
  - l4util\_cpu\_capabilities, 658
  - l4util\_cpu\_capabilities\_nocheck, 659
  - l4util\_cpu\_has\_cpuid, 659
- cpu\_allow\_shutdown
  - L4::Platform\_control, 1357
- cpu\_disable
  - L4::Platform\_control, 1358
- cpu\_enable
  - L4::Platform\_control, 1359
- create
  - L4::Factory, 1094, 1095
- create\_buffer
  - L4Re::Video::Goos, 1915
- create\_factory
  - L4::Factory, 1096
- create\_gate
  - L4::Factory, 1097
- create\_task
  - L4::Factory, 1098
- create\_thread\_group
  - L4::Factory, 1100
- create\_view
  - L4Re::Video::Goos, 1916
- cur\_buf
  - Net\_transfer, 2321
- current\_flags
  - L4virtio::Svr::Request\_processor, 2203
- cx
  - l4\_vcpu\_regs\_t, 1572
- cxx, 741
  - access\_once, 743
  - gcd, 744
  - lcm, 745
  - write\_now, 746
- cxx::arith::Ld< V >, 834
- cxx::Avl\_map< KEY\_TYPE, DATA\_TYPE, COMPARE,  
ALLOC >, 835
  - Avl\_map, 838
  - insert, 839
  - operator[], 839, 840
- cxx::Avl\_set< ITEM\_TYPE, COMPARE, ALLOC >, 840
- cxx::Avl\_tree< Node, Get\_key, Compare >, 844
  - insert, 850
  - Iterator, 850
  - remove, 850
- cxx::Avl\_tree\_node, 853
- cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc  
>, 855
  - alloc, 858
  - free, 858
  - free\_objects, 858
  - max\_free\_slabs, 858
  - object\_size, 858
  - objects\_per\_slab, 858
  - slab\_size, 858
  - total\_objects, 859
- cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc  
>::Slab\_i, 859
- cxx::Base\_slab\_static< Obj\_size, Slab\_size, Max\_free,  
Alloc >, 860
  - alloc, 863
  - free, 863
  - free\_objects, 864
  - max\_free\_slabs, 863
  - object\_size, 863
  - objects\_per\_slab, 863
  - slab\_size, 863
  - total\_objects, 864
- cxx::Bitfield< T, LSB, MSB >, 865
  - Bits, 867
  - Bits\_type, 867
  - get, 868
  - get\_unshifted, 868
  - Low\_mask, 868
  - Lsb, 867
  - Mask, 868
  - Masks, 867
  - Msb, 867
  - set, 868
  - set\_dirty, 869
  - set\_unshifted, 869
  - set\_unshifted\_dirty, 870
  - Shift\_type, 867
  - val, 871
  - val\_dirty, 871
  - val\_unshifted, 872
- cxx::Bitfield< T, LSB, MSB >::Value< TT >, 873
- cxx::Bitfield< T, LSB, MSB >::Value\_base< TT >, 874
- cxx::Bitfield< T, LSB, MSB >::Value\_unshifted< TT >,  
875
- cxx::Bitmap< BITS >, 876
  - scan\_zero, 880
- cxx::Bitmap\_base, 881
  - atomic\_clear\_bit, 884
  - atomic\_get\_and\_clear, 884
  - atomic\_get\_and\_set, 885
  - atomic\_set\_bit, 885



- bit, 886
- bit\_index, 887
- C\_bits, 884
- clear\_bit, 888
- operator[], 889, 890
- scan\_zero, 890
- set\_bit, 891
- W\_bits, 884
- word\_index, 892
- cxx::Bitmap\_base::Bit, 893
- cxx::Bitmap\_base::Char< BITS >, 894
- cxx::Bitmap\_base::Word< BITS >, 895
- cxx::Bits, 747
- cxx::Bits::Avl\_map\_get\_key< KEY\_TYPE >, 896
- cxx::Bits::Avl\_set\_get\_key< KEY\_TYPE >, 896
- cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >, 897
  - Base\_avl\_set, 900, 901
  - begin, 902
  - E\_exist, 900
  - E\_inval, 900
  - E\_noent, 900
  - E\_nomem, 900
  - end, 902, 903
  - erase, 903
  - find\_node, 904
  - insert, 904
  - lower\_bound\_node, 905
  - rbegin, 905, 906
  - remove, 906
  - rend, 907
- cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >::Node, 908
  - operator->, 909
  - operator\*, 909
  - valid, 909
- cxx::Bits::Basic\_list< POLICY >, 910
  - clear, 911
  - iter, 911
- cxx::Bits::Bst< Node, Get\_key, Compare >, 913
  - begin, 917, 918
  - dir, 918, 919
  - end, 920
  - find, 920
  - find\_node, 921
  - lower\_bound\_node, 922
  - rbegin, 922, 923
  - remove\_all, 923
  - remove\_tree, 925
  - rend, 926
- cxx::Bits::Bst\_node, 927
- cxx::Bits::Direction, 929
  - Direction\_e, 930
  - L, 930
  - N, 930
  - operator!, 931
  - R, 930
- cxx::Bits::Smart\_ptr\_list< ITEM >, 931
  - pop\_front, 934
- cxx::Bits::Smart\_ptr\_list\_item< T, STORE\_T >, 934
- cxx::H\_list< T, POLICY >, 936
  - erase, 940
  - insert, 940
  - insert\_after, 941
  - insert\_before, 941
  - iter, 942
  - pop\_front, 942
  - remove, 943
  - replace, 943
- cxx::H\_list\_item\_t< ELEM\_TYPE >, 944
  - ~H\_list\_item\_t, 945
  - H\_list\_item\_t, 945
- cxx::H\_list\_t< T >, 946
- cxx::List< D, Alloc >, 950
  - operator[], 951
- cxx::List< D, Alloc >::Iter, 952
- cxx::List\_alloc, 953
  - alloc, 954
  - alloc\_max, 954
  - avail, 955
  - free, 955
  - List\_alloc, 953
- cxx::List\_item, 956
  - push\_back, 957
  - push\_front, 958
  - remove, 959
- cxx::List\_item::Iter, 960
- cxx::List\_item::T\_iter< T, Poly >, 961
- cxx::Lt\_functor< Obj >, 963
- cxx::New\_allocator< \_Type >, 963
- cxx::Nothrow, 964
- cxx::Pair< First, Second >, 965
  - Pair, 967
- cxx::Pair\_first\_compare< Cmp, Typ >, 968
  - operator(), 969
  - Pair\_first\_compare, 968
- cxx::Ref\_obj\_list\_item< T >, 969
- cxx::Ref\_ptr< T, CNT >, 971
  - get, 975
  - ptr, 975
  - Ref\_ptr, 974
  - release, 975
- cxx::S\_list< T, POLICY >, 976
  - pop\_front, 979
- cxx::Slab< Type, Slab\_size, Max\_free, Alloc >, 979
  - alloc, 982
  - free, 983
- cxx::Slab\_static< Type, Slab\_size, Max\_free, Alloc >, 984
  - alloc, 987
- cxx::static\_vector< T, IDX >, 988
- cxx::String, 989
  - find, 993
  - from\_dec, 994
  - from\_hex, 994
  - starts\_with, 995



- String, [993](#)
- cxx::Weak\_ref< T >, [996](#)
- cxx::Weak\_ref\_base, [1000](#)
- cxx::Weak\_ref\_base::List, [1003](#)
- d\_tag
  - Elf32\_Dyn, [1009](#)
  - Elf64\_Dyn, [1019](#)
- data
  - L4::lpc::Varg, [1228](#)
- Data\_buffer
  - L4virtio::Svr::Data\_buffer, [2162](#)
- data\_size
  - L4virtio::Svr::Block\_request< Ds\_data >, [2112](#)
- data\_space
  - L4Re::Vfs::Be\_file, [1861](#)
  - L4Re::Vfs::Regular\_file, [1897](#)
- Dataspace interface, [534](#)
  - l4re\_ds\_allocate, [536](#)
  - l4re\_ds\_clear, [536](#)
  - l4re\_ds\_copy\_in, [537](#)
  - L4RE\_DS\_F\_BUFFERABLE, [535](#)
  - L4RE\_DS\_F\_CACHEABLE, [535](#)
  - L4RE\_DS\_F\_CACHING\_MASK, [535](#)
  - L4RE\_DS\_F\_CACHING\_SHIFT, [535](#)
  - L4RE\_DS\_F\_NORMAL, [535](#)
  - L4RE\_DS\_F\_UNCACHEABLE, [535](#)
  - l4re\_ds\_flags, [537](#)
  - l4re\_ds\_info, [538](#)
  - l4re\_ds\_map\_flags, [535](#)
  - l4re\_ds\_map\_info, [538](#)
  - l4re\_ds\_size, [539](#)
- dbg\_events
  - L4Re::Env, [1648](#), [1649](#)
- debug
  - L4Re::Debug\_obj, [1636](#)
- Debug interface, [539](#)
  - l4re\_debug\_obj\_debug, [540](#)
- Debugging API, [596](#)
- dec
  - L4Re::Util::Counter< COUNTER >, [1797](#)
  - L4Re::Util::Counter\_atomic< COUNTER >, [1800](#)
- dec\_refcnt
  - L4::Kobject, [1320](#)
- Dedicated
  - L4::Kip::Mem\_desc, [1309](#)
- delete\_buffer
  - L4Re::Video::Goos, [1917](#)
- delete\_obj
  - L4::Task, [1430](#)
- delete\_view
  - L4Re::Video::Goos, [1918](#)
- Demand
  - L4::Kobject\_typeid< T >, [1333](#)
  - L4::Type\_info::Demand, [1470](#)
- demand
  - L4::Kobject\_typeid< T >, [1333](#)
  - L4::Kobject\_typeid< void >, [1336](#)
- Deprecated List, [103](#)
- desc
  - L4virtio::Driver::Virtqueue, [2091](#)
  - L4virtio::Svr::Virtqueue, [2265](#)
  - L4virtio::Svr::Virtqueue::Head\_desc, [2272](#)
- desc\_align
  - L4virtio::Virtqueue, [2279](#)
- desc\_avail
  - L4virtio::Svr::Virtqueue, [2266](#)
- desc\_size
  - L4virtio::Virtqueue, [2280](#)
- detach
  - L4::Irq, [1292](#)
  - L4Re::Rm, [1740](#)–[1742](#)
  - L4Re::Util::Event\_buffer\_t< PAYLOAD >, [1819](#)
  - Rm, [2330](#)–[2332](#)
- Detach\_again
  - L4Re::Rm, [1737](#)
  - Rm, [2327](#)
- Detach\_exact
  - L4Re::Rm, [1736](#)
  - Rm, [2326](#)
- Detach\_flags
  - L4Re::Rm, [1736](#)
  - Rm, [2326](#)
- Detach\_free
  - L4Re::Rm::F, [1758](#)
  - Rm::F, [2339](#)
- Detach\_keep
  - L4Re::Rm, [1736](#)
  - Rm, [2326](#)
- Detach\_overlap
  - L4Re::Rm, [1736](#)
  - Rm, [2326](#)
- Detach\_result
  - L4Re::Rm, [1736](#)
  - Rm, [2326](#)
- Detached\_ds
  - L4Re::Rm, [1737](#)
  - Rm, [2327](#)
- Dev\_config
  - L4virtio::Svr::Dev\_config, [2167](#), [2168](#)
- dev\_handle
  - L4vbus::Device, [1974](#)
- Device
  - L4vbus::Device, [1973](#)
  - L4virtio::Svr::Console::Device, [2122](#), [2123](#)
- device
  - L4vbus::Device, [1975](#)
- Device\_add
  - L4virtio::Svr::Console::Control\_message, [2115](#)
- device\_by\_hid
  - L4vbus::Device, [1975](#)
- device\_config
  - L4virtio::Device, [2050](#)
- device\_error
  - L4virtio::Svr::Device\_t< DATA >, [2185](#)
- device\_notification\_irq
  - L4virtio::Device, [2051](#)

- device\_notify\_irq
  - L4virtio::Svr::Device\_t< DATA >, 2185
- Device\_ready
  - L4virtio::Svr::Console::Control\_message, 2115
- Device\_remove
  - L4virtio::Svr::Console::Control\_message, 2115
- DF\_1\_CONFALT
  - ELF binary format, 704
- DF\_1\_DIRECT
  - ELF binary format, 703
- DF\_1\_DISPRELDNE
  - ELF binary format, 704
- DF\_1\_DISPRELPND
  - ELF binary format, 704
- DF\_1\_ENDFILTEE
  - ELF binary format, 704
- DF\_1\_GLOBAL
  - ELF binary format, 703
- DF\_1\_GROUP
  - ELF binary format, 703
- DF\_1\_INITFIRST
  - ELF binary format, 703
- DF\_1\_INTERPOSE
  - ELF binary format, 703
- DF\_1\_LOADFLTR
  - ELF binary format, 703
- DF\_1\_NODEFLIB
  - ELF binary format, 703
- DF\_1\_NODELETE
  - ELF binary format, 703
- DF\_1\_NODUMP
  - ELF binary format, 703
- DF\_1\_NOOPEN
  - ELF binary format, 703
- DF\_1\_NOW
  - ELF binary format, 703
- DF\_1\_ORIGIN
  - ELF binary format, 703
- DF\_BIND\_NOW
  - ELF binary format, 704
- DF\_ORIGIN
  - ELF binary format, 704
- DF\_P1\_GROUPEPERM
  - ELF binary format, 704
- DF\_P1\_LAZYLOAD
  - ELF binary format, 704
- DF\_STATIC\_TLS
  - ELF binary format, 704
- DF\_SYMBOLIC
  - ELF binary format, 704
- DF\_TEXTREL
  - ELF binary format, 704
- df1
  - l4\_vm\_vmx\_vcpu\_infos\_t, 1582
- di
  - l4\_vcpu\_regs\_t, 1572
- dir
  - cxx::Bits::Bst< Node, Get\_key, Compare >, 918, 919
- Direction
  - L4Re::Dma\_space, 1641
- Direction\_e
  - cxx::Bits::Direction, 930
- disable
  - L4virtio::Virtqueue, 2281
- disable\_notify
  - L4virtio::Svr::Virtqueue, 2266
- disassociate
  - L4Re::Dma\_space, 1642
- dispatch
  - L4::Basic\_registry, 1034
  - L4::Epiface, 1077
  - L4::Epiface\_t< Derived, IFACE, BASE, bool >, 1084
  - L4::Irqp\_t< Derived, BASE, bool >, 1305
  - L4::Server\_object, 1407, 1408
- dispatch\_meta\_request
  - L4::Server\_object\_t< IFACE, BASE >, 1413
- DMA space, 291
- DMA Space Interface, 540
  - l4re\_dma\_space\_associate, 541
  - l4re\_dma\_space\_disassociate, 542
  - l4re\_dma\_space\_map, 542
  - l4re\_dma\_space\_t, 541
  - l4re\_dma\_space\_unmap, 543
- dma\_space.h
  - L4RE\_DMA\_SPACE\_BIDIRECTIONAL, 2844
  - L4RE\_DMA\_SPACE\_COHERENT, 2845
  - l4re\_dma\_space\_direction, 2844
  - L4RE\_DMA\_SPACE\_FROM\_DEVICE, 2844
  - L4RE\_DMA\_SPACE\_NONE, 2844
  - L4RE\_DMA\_SPACE\_PHYS\_SPACE, 2845
  - l4re\_dma\_space\_space\_attribs, 2844
  - L4RE\_DMA\_SPACE\_TO\_DEVICE, 2844
- done
  - L4virtio::Svr::Data\_buffer, 2163
  - Net\_transfer, 2321
- down
  - L4::Semaphore, 1396
- driver\_acknowledge
  - L4virtio::Driver::Device, 2067
- driver\_connect
  - L4virtio::Driver::Device, 2068
- Driver\_mem\_region\_t
  - L4virtio::Svr::Driver\_mem\_region\_t< DATA >, 2198
- drop\_requests
  - L4virtio\_port, 2315
  - Virtio\_net\_request, 2355
- drv\_base
  - L4virtio::Svr::Driver\_mem\_region\_t< DATA >, 2198
- ds
  - L4virtio::Svr::Dev\_config, 2170

- L4virtio::Svr::Driver\_mem\_region\_t< DATA >, DT\_RELA
    - ELF binary format, [705](#)
- [2199](#)
- Ds\_map\_mask
  - L4Re::Rm::F, [1758](#)
  - Rm::F, [2339](#)
- ds\_offset
  - L4virtio::Svr::Driver\_mem\_region\_t< DATA >, DT\_RELENT
    - ELF binary format, [705](#)
- [2199](#)
- DT\_BIND\_NOW
  - ELF binary format, [705](#)
- DT\_DEBUG
  - ELF binary format, [705](#)
- DT\_ENCODING
  - ELF binary format, [705](#)
- DT\_FINI
  - ELF binary format, [705](#)
- DT\_FINI\_ARRAY
  - ELF binary format, [705](#)
- DT\_FINI\_ARRAYSZ
  - ELF binary format, [705](#)
- DT\_FLAGS
  - ELF binary format, [705](#)
- DT\_HASH
  - ELF binary format, [705](#)
- DT\_HIOS
  - ELF binary format, [705](#)
- DT\_HIPROC
  - ELF binary format, [705](#)
- DT\_INIT
  - ELF binary format, [705](#)
- DT\_INIT\_ARRAY
  - ELF binary format, [705](#)
- DT\_INIT\_ARRAYSZ
  - ELF binary format, [705](#)
- DT\_JMPREL
  - ELF binary format, [705](#)
- DT\_LOOS
  - ELF binary format, [705](#)
- DT\_LOPROC
  - ELF binary format, [705](#)
- DT\_NEEDED
  - ELF binary format, [705](#)
- DT\_NULL
  - ELF binary format, [705](#)
- DT\_NUM
  - ELF binary format, [705](#)
- DT\_PLTGOT
  - ELF binary format, [705](#)
- DT\_PLTRELSZ
  - ELF binary format, [705](#)
- DT\_PREINIT\_ARRAY
  - ELF binary format, [705](#)
- DT\_PREINIT\_ARRAYSZ
  - ELF binary format, [705](#)
- DT\_PTRREL
  - ELF binary format, [705](#)
- DT\_REL
  - ELF binary format, [705](#)
- DT\_RELA
    - ELF binary format, [705](#)
  - DT\_RELAENT
    - ELF binary format, [705](#)
  - DT\_RELASZ
    - ELF binary format, [705](#)
  - DT\_RELENT
    - ELF binary format, [705](#)
  - DT\_RELSZ
    - ELF binary format, [705](#)
  - DT\_RPATH
    - ELF binary format, [705](#)
  - DT\_RUNPATH
    - ELF binary format, [705](#)
  - DT\_SONAME
    - ELF binary format, [705](#)
  - DT\_STRSZ
    - ELF binary format, [705](#)
  - DT\_STRTAB
    - ELF binary format, [705](#)
  - DT\_SYMBOLIC
    - ELF binary format, [705](#)
  - DT\_SYMENT
    - ELF binary format, [705](#)
  - DT\_SYMTAB
    - ELF binary format, [705](#)
  - DT\_TEXTREL
    - ELF binary format, [705](#)
- dump
  - L4Re::Video::Color\_component, [1908](#)
  - L4Re::Video::Pixel\_info, [1931](#)
  - L4virtio::Virtqueue, [2281](#)
- dx
  - l4\_vcpu\_regs\_t, [1572](#)
- E\_exist
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COM-PARE, ALLOC, GET\_KEY >, [900](#)
- e\_flags
  - Elf32\_Ehdr, [1010](#)
  - Elf64\_Ehdr, [1020](#)
- E\_inval
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COM-PARE, ALLOC, GET\_KEY >, [900](#)
- e\_machine
  - Elf32\_Ehdr, [1010](#)
  - Elf64\_Ehdr, [1020](#)
- E\_noent
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COM-PARE, ALLOC, GET\_KEY >, [900](#)
- E\_nomem
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COM-PARE, ALLOC, GET\_KEY >, [900](#)
- e\_type
  - Elf32\_Ehdr, [1011](#)
  - Elf64\_Ehdr, [1021](#)
- e\_version
  - Elf32\_Ehdr, [1011](#)
  - Elf64\_Ehdr, [1021](#)

Eager\_map  
     L4Re::Rm::F, 1757  
     Rm::F, 2339  
 EDID parsing functionality, 459  
     libedid\_block\_size, 459  
     libedid\_check\_header, 459  
     libedid\_checksum, 460  
     Libedid\_consts, 459  
     libedid\_dump, 460  
     libedid\_dump\_standard\_timings, 460  
     libedid\_num\_ext\_blocks, 460  
     libedid\_pnp\_id, 462  
     libedid\_prefered\_resolution, 462  
     libedid\_revision, 462  
     libedid\_version, 462  
 EF\_ARM\_ALIGN8  
     ELF binary format, 706  
 EI\_ABIVERSION  
     ELF binary format, 706  
 EI\_CLASS  
     ELF binary format, 706  
 EI\_DATA  
     ELF binary format, 706  
 EI\_MAG0  
     ELF binary format, 706  
 EI\_MAG1  
     ELF binary format, 706  
 EI\_MAG2  
     ELF binary format, 706  
 EI\_MAG3  
     ELF binary format, 706  
 EI\_NIDENT  
     ELF binary format, 701  
 EI\_OSABI  
     ELF binary format, 706  
 EI\_PAD  
     ELF binary format, 706  
 EI\_VERSION  
     ELF binary format, 706  
 ELF binary format, 693  
     AT\_BASE, 701  
     AT\_EGID, 701  
     AT\_ENTRY, 701  
     AT\_EUID, 701  
     AT\_EXECFD, 701  
     AT\_FLAGS, 701  
     AT\_GID, 701  
     AT\_IGNORE, 701  
     AT\_L4\_AUX, 701  
     AT\_L4\_ENV, 701  
     AT\_NOTELF, 701  
     AT\_NULL, 701  
     AT\_PAGESZ, 701  
     AT\_PHDR, 701  
     AT\_PHENT, 701  
     AT\_PHNUM, 701  
     AT\_UID, 701  
     DF\_1\_CONFALT, 704  
     DF\_1\_DIRECT, 703  
     DF\_1\_DISPRELDNE, 704  
     DF\_1\_DISPRELPND, 704  
     DF\_1\_ENDFILTEE, 704  
     DF\_1\_GLOBAL, 703  
     DF\_1\_GROUP, 703  
     DF\_1\_INITFIRST, 703  
     DF\_1\_INTERPOSE, 703  
     DF\_1\_LOADFLTR, 703  
     DF\_1\_NODEFLIB, 703  
     DF\_1\_NODELETE, 703  
     DF\_1\_NODUMP, 703  
     DF\_1\_NOOPEN, 703  
     DF\_1\_NOW, 703  
     DF\_1\_ORIGIN, 703  
     DF\_BIND\_NOW, 704  
     DF\_ORIGIN, 704  
     DF\_P1\_GROUPPERM, 704  
     DF\_P1\_LAZYLOAD, 704  
     DF\_STATIC\_TLS, 704  
     DF\_SYMBOLIC, 704  
     DF\_TEXTREL, 704  
     DT\_BIND\_NOW, 705  
     DT\_DEBUG, 705  
     DT\_ENCODING, 705  
     DT\_FINI, 705  
     DT\_FINI\_ARRAY, 705  
     DT\_FINI\_ARRAYSZ, 705  
     DT\_FLAGS, 705  
     DT\_HASH, 705  
     DT\_HIOS, 705  
     DT\_HIPROC, 705  
     DT\_INIT, 705  
     DT\_INIT\_ARRAY, 705  
     DT\_INIT\_ARRAYSZ, 705  
     DT\_JMPREL, 705  
     DT\_LOOS, 705  
     DT\_LOPROC, 705  
     DT\_NEEDED, 705  
     DT\_NULL, 705  
     DT\_NUM, 705  
     DT\_PLTGOT, 705  
     DT\_PLTRELSZ, 705  
     DT\_PREINIT\_ARRAY, 705  
     DT\_PREINIT\_ARRAYSZ, 705  
     DT\_PTRREL, 705  
     DT\_REL, 705  
     DT\_RELA, 705  
     DT\_RELAENT, 705  
     DT\_RELASZ, 705  
     DT\_RELENT, 705  
     DT\_RELSZ, 705  
     DT\_RPATH, 705  
     DT\_RUNPATH, 705  
     DT\_SONAME, 705  
     DT\_STRSZ, 705  
     DT\_STRTAB, 705  
     DT\_SYMBOLIC, 705

DT\_SYMENT, [705](#)  
DT\_SYMTAB, [705](#)  
DT\_TEXTREL, [705](#)  
EF\_ARM\_ALIGN8, [706](#)  
EI\_ABIVERSION, [706](#)  
EI\_CLASS, [706](#)  
EI\_DATA, [706](#)  
EI\_MAG0, [706](#)  
EI\_MAG1, [706](#)  
EI\_MAG2, [706](#)  
EI\_MAG3, [706](#)  
EI\_NIDENT, [701](#)  
EI\_OSABI, [706](#)  
EI\_PAD, [706](#)  
EI\_VERSION, [706](#)  
ELF32\_R\_TYPE, [699](#)  
ELF32\_ST\_BIND, [699](#)  
ELF32\_ST\_TYPE, [699](#)  
ELF64\_R\_TYPE, [699](#)  
ELF64\_ST\_BIND, [700](#)  
ELF64\_ST\_TYPE, [700](#)  
Elf\_ARM\_SBs, [701](#)  
Elf\_ATs, [701](#)  
Elf\_CLASSs, [701](#)  
Elf\_DATAs, [703](#)  
Elf\_DF\_1s, [703](#)  
Elf\_DF\_P1s, [704](#)  
Elf\_DFs, [704](#)  
Elf\_DTs, [704](#)  
Elf\_EF\_ARM\_s, [705](#)  
Elf\_EIs, [706](#)  
Elf\_EMs, [706](#)  
Elf\_ETs, [708](#)  
Elf\_EVs, [709](#)  
Elf\_MAGs, [709](#)  
Elf\_NTs\_core, [709](#)  
Elf\_NTs\_obj, [710](#)  
Elf\_OSABIs, [710](#)  
ELF\_PFs, [711](#)  
Elf\_PTs, [711](#)  
Elf\_R\_386\_s, [712](#)  
Elf\_R\_AARCH64\_s, [713](#)  
Elf\_R\_ARM\_s, [713](#)  
Elf\_R\_X86\_64\_s, [714](#)  
Elf\_SHF\_s\_ARM, [715](#)  
Elf\_SHFs, [715](#)  
Elf\_SHNs, [715](#)  
Elf\_SHTs, [717](#)  
Elf\_STBs, [717](#)  
Elf\_STTs, [718](#)  
ELFCLASS32, [703](#)  
ELFCLASS64, [703](#)  
ELFCLASSNONE, [703](#)  
ELFCLASSNUM, [703](#)  
ELFDATA2LSB, [703](#)  
ELFDATA2MSB, [703](#)  
ELFDATANONE, [703](#)  
ELFDATANUM, [703](#)  
ELFMAG0, [709](#)  
ELFMAG1, [709](#)  
ELFMAG2, [709](#)  
ELFMAG3, [709](#)  
ELFOSABI\_AIX, [711](#)  
ELFOSABI\_ARM, [711](#)  
ELFOSABI\_FREEBSD, [711](#)  
ELFOSABI\_HPUX, [710](#)  
ELFOSABI\_IRIX, [711](#)  
ELFOSABI\_LINUX, [711](#)  
ELFOSABI\_MODESTO, [711](#)  
ELFOSABI\_NETBSD, [710](#)  
ELFOSABI\_NONE, [710](#)  
ELFOSABI\_OPENBSD, [711](#)  
ELFOSABI\_SOLARIS, [711](#)  
ELFOSABI\_STANDALONE, [711](#)  
ELFOSABI\_SYSV, [710](#)  
ELFOSABI\_TRU64, [711](#)  
EM\_386, [707](#)  
EM\_68HC05, [707](#)  
EM\_68HC08, [707](#)  
EM\_68HC11, [707](#)  
EM\_68HC12, [707](#)  
EM\_68HC16, [707](#)  
EM\_68K, [707](#)  
EM\_860, [707](#)  
EM\_88K, [707](#)  
EM\_960, [707](#)  
EM\_AARCH64, [708](#)  
EM\_ALPHA, [707](#)  
EM\_ALTERA\_NIOS2, [708](#)  
EM\_ARC, [707](#)  
EM\_ARC\_A5, [708](#)  
EM\_ARM, [707](#)  
EM\_AVR, [708](#)  
EM\_COLDFIRE, [707](#)  
EM\_CRIS, [708](#)  
EM\_D10V, [708](#)  
EM\_D30V, [708](#)  
EM\_FIREPATH, [708](#)  
EM\_FR20, [707](#)  
EM\_FR30, [708](#)  
EM\_FX66, [707](#)  
EM\_H8\_300, [707](#)  
EM\_H8\_300H, [707](#)  
EM\_H8\_500, [707](#)  
EM\_H8S, [707](#)  
EM\_HUANY, [708](#)  
EM\_IA\_64, [707](#)  
EM\_JAVELIN, [708](#)  
EM\_M32, [707](#)  
EM\_M32R, [708](#)  
EM\_MICROBLAZE, [708](#)  
EM\_MIPS, [707](#)  
EM\_MIPS\_RS4\_BE, [707](#)  
EM\_MIPS\_X, [707](#)  
EM\_MMIX, [708](#)  
EM\_MN10200, [708](#)

EM\_MN10300, [708](#)  
EM\_NONE, [707](#)  
EM\_OPENRISC, [708](#)  
EM\_PARISC, [707](#)  
EM\_PDSP, [707](#)  
EM\_PJ, [708](#)  
EM\_PPC, [707](#)  
EM\_PRISM, [708](#)  
EM\_RCE, [707](#)  
EM\_RH32, [707](#)  
EM\_RISCV, [708](#)  
EM\_SH, [707](#)  
EM\_SPARC, [707](#)  
EM\_SPARC32PLUS, [707](#)  
EM\_SPARC64, [707](#)  
EM\_SPARCV9, [707](#)  
EM\_ST19, [708](#)  
EM\_ST7, [707](#)  
EM\_ST9PLUS, [707](#)  
EM\_SVX, [708](#)  
EM\_TILEGX, [708](#)  
EM\_TILEPRO, [708](#)  
EM\_TRICORE, [707](#)  
EM\_V800, [707](#)  
EM\_V850, [708](#)  
EM\_VAX, [708](#)  
EM\_VPP500, [707](#)  
EM\_X86\_64, [707](#)  
EM\_XTENSA, [708](#)  
EM\_ZSP, [708](#)  
ET\_CORE, [709](#)  
ET\_DYN, [709](#)  
ET\_EXEC, [709](#)  
ET\_HIPROC, [709](#)  
ET\_LOPROC, [709](#)  
ET\_NONE, [709](#)  
ET\_REL, [709](#)  
EV\_CURRENT, [709](#)  
EV\_NONE, [709](#)  
NT\_ASRS, [710](#)  
NT\_AUXV, [710](#)  
NT\_FPREGSET, [710](#)  
NT\_GWINDOWS, [710](#)  
NT\_LWPSINFO, [710](#)  
NT\_LWPSTATUS, [710](#)  
NT\_PLATFORM, [710](#)  
NT\_PRCRED, [710](#)  
NT\_PRFPXREG, [710](#)  
NT\_PRPSINFO, [710](#)  
NT\_PRSTATUS, [710](#)  
NT\_PRXREG, [710](#)  
NT\_PSINFO, [710](#)  
NT\_PSTATUS, [710](#)  
NT\_TASKSTRUCT, [710](#)  
NT\_UTSNAME, [710](#)  
NT\_VERSION, [710](#)  
PF\_ARM\_SB, [701](#)  
PF\_MASKOS, [711](#)  
PF\_MASKPROC, [711](#)  
PF\_R, [711](#)  
PF\_W, [711](#)  
PF\_X, [711](#)  
PT\_DYNAMIC, [711](#)  
PT\_GNU\_EH\_FRAME, [712](#)  
PT\_GNU\_RELRO, [712](#)  
PT\_GNU\_STACK, [712](#)  
PT\_HIOS, [712](#)  
PT\_HIPROC, [712](#)  
PT\_INTERP, [711](#)  
PT\_L4\_AUX, [712](#)  
PT\_L4\_STACK, [712](#)  
PT\_LOAD, [711](#)  
PT\_LOOS, [712](#)  
PT\_LOPROC, [712](#)  
PT\_NOTE, [712](#)  
PT\_NULL, [711](#)  
PT\_NUM, [712](#)  
PT\_PHDR, [712](#)  
PT\_SHLIB, [712](#)  
PT\_TLS, [712](#)  
R\_386\_32, [712](#)  
R\_386\_COPY, [712](#)  
R\_386\_GLOB\_DAT, [712](#)  
R\_386\_GOT32, [712](#)  
R\_386\_GOTOFF, [712](#)  
R\_386\_GOTPC, [712](#)  
R\_386\_JMP\_SLOT, [712](#)  
R\_386\_NONE, [712](#)  
R\_386\_NUM, [713](#)  
R\_386\_PC32, [712](#)  
R\_386\_PLT32, [712](#)  
R\_386\_RELATIVE, [712](#)  
R\_386\_TLS\_DTPMOD32, [713](#)  
R\_386\_TLS\_DTPOFF32, [713](#)  
R\_386\_TLS\_GD, [712](#)  
R\_386\_TLS\_GD\_32, [712](#)  
R\_386\_TLS\_GD\_CALL, [713](#)  
R\_386\_TLS\_GD\_POP, [713](#)  
R\_386\_TLS\_GD\_PUSH, [713](#)  
R\_386\_TLS\_GOTIE, [712](#)  
R\_386\_TLS\_IE, [712](#)  
R\_386\_TLS\_IE\_32, [713](#)  
R\_386\_TLS\_LDM, [712](#)  
R\_386\_TLS\_LDM\_32, [713](#)  
R\_386\_TLS\_LDM\_CALL, [713](#)  
R\_386\_TLS\_LDM\_POP, [713](#)  
R\_386\_TLS\_LDM\_PUSH, [713](#)  
R\_386\_TLS\_LDO\_32, [713](#)  
R\_386\_TLS\_LE, [712](#)  
R\_386\_TLS\_LE\_32, [713](#)  
R\_386\_TLS\_TPOFF, [712](#)  
R\_386\_TLS\_TPOFF32, [713](#)  
R\_AARCH64\_NONE, [713](#)  
R\_ARM\_ABS12, [713](#)  
R\_ARM\_ABS16, [713](#)  
R\_ARM\_ABS32, [713](#)

R\_ARM\_ABS8, 714  
R\_ARM\_COPY, 714  
R\_ARM\_GLOB\_DAT, 714  
R\_ARM\_GOT32, 714  
R\_ARM\_GOTOFF, 714  
R\_ARM\_GOTPC, 714  
R\_ARM\_JUMP\_SLOT, 714  
R\_ARM\_NONE, 713  
R\_ARM\_NUM, 714  
R\_ARM\_PC24, 713  
R\_ARM\_PLT32, 714  
R\_ARM\_REL32, 713  
R\_ARM\_RELATIVE, 714  
R\_ARM\_THM\_PC11, 714  
R\_ARM\_THM\_PC9, 714  
R\_X86\_64\_16, 714  
R\_X86\_64\_32, 714  
R\_X86\_64\_32S, 714  
R\_X86\_64\_64, 714  
R\_X86\_64\_8, 714  
R\_X86\_64\_COPY, 714  
R\_X86\_64\_DTPMOD64, 714  
R\_X86\_64\_DTPOFF32, 715  
R\_X86\_64\_DTPOFF64, 714  
R\_X86\_64\_GLOB\_DAT, 714  
R\_X86\_64\_GOT32, 714  
R\_X86\_64\_GOTPCREL, 714  
R\_X86\_64\_GOTTPOFF, 715  
R\_X86\_64\_JUMP\_SLOT, 714  
R\_X86\_64\_NONE, 714  
R\_X86\_64\_PC16, 714  
R\_X86\_64\_PC32, 714  
R\_X86\_64\_PC8, 714  
R\_X86\_64\_PLT32, 714  
R\_X86\_64\_RELATIVE, 714  
R\_X86\_64\_TLSD, 715  
R\_X86\_64\_TLSD, 715  
R\_X86\_64\_TPOFF32, 715  
R\_X86\_64\_TPOFF64, 714  
SHF\_ALLOC, 715  
SHF\_ARM\_COMDEF, 715  
SHF\_ARM\_ENTRYSECT, 715  
SHF\_EXECINSTR, 715  
SHF\_GROUP, 715  
SHF\_INFO\_LINK, 715  
SHF\_MASKOS, 715  
SHF\_MASKPROC, 715  
SHF\_MERGE, 715  
SHF\_OS\_NONCONFORMING, 715  
SHF\_STRINGS, 715  
SHF\_TLS, 715  
SHF\_WRITE, 715  
SHN\_ABS, 717  
SHN\_COMMON, 717  
SHN\_HIPROC, 717  
SHN\_HIRESERVE, 717  
SHN\_LOPROC, 717  
SHN\_LORESERVE, 717  
SHN\_UNDEF, 717  
SHT\_DYNAMIC, 717  
SHT\_DYNSYM, 717  
SHT\_FINI\_ARRAY, 717  
SHT\_GROUP, 717  
SHT\_HASH, 717  
SHT\_HIOS, 717  
SHT\_HIPROC, 717  
SHT\_HIUSER, 717  
SHT\_INIT\_ARRAY, 717  
SHT\_LOOS, 717  
SHT\_LOPROC, 717  
SHT\_LOUSER, 717  
SHT\_NOBITS, 717  
SHT\_NOTE, 717  
SHT\_NULL, 717  
SHT\_NUM, 717  
SHT\_PREINIT\_ARRAY, 717  
SHT\_PROGBITS, 717  
SHT\_REL, 717  
SHT\_RELA, 717  
SHT\_SHLIB, 717  
SHT\_STRTAB, 717  
SHT\_SYMTAB, 717  
SHT\_SYMTAB\_SHNDX, 717  
STB\_GLOBAL, 718  
STB\_HIOS, 718  
STB\_HIPROC, 718  
STB\_LOCAL, 718  
STB\_LOOS, 718  
STB\_LOPROC, 718  
STB\_WEAK, 718  
STT\_FILE, 718  
STT\_FUNC, 718  
STT\_HIOS, 718  
STT\_HIPROC, 718  
STT\_LOOS, 718  
STT\_LOPROC, 718  
STT\_NOTYPE, 718  
STT\_OBJECT, 718  
STT\_SECTION, 718  
Elf32\_Auxv, 1007  
    atype, 1008  
Elf32\_Dyn, 1008  
    d\_tag, 1009  
Elf32\_Ehdr, 1009  
    e\_flags, 1010  
    e\_machine, 1010  
    e\_type, 1011  
    e\_version, 1011  
Elf32\_Phdr, 1012  
    p\_flags, 1013  
    p\_type, 1013  
ELF32\_R\_TYPE  
    ELF binary format, 699  
Elf32\_Rel, 1013  
Elf32\_Rela, 1014  
Elf32\_Shdr, 1015

- sh\_flags, [1016](#)
- sh\_type, [1016](#)
- ELF32\_ST\_BIND
  - ELF binary format, [699](#)
- ELF32\_ST\_TYPE
  - ELF binary format, [699](#)
- Elf32\_Sym, [1016](#)
- Elf64\_Auxv, [1017](#)
  - atype, [1018](#)
- Elf64\_Dyn, [1018](#)
  - d\_tag, [1019](#)
- Elf64\_Ehdr, [1019](#)
  - e\_flags, [1020](#)
  - e\_machine, [1020](#)
  - e\_type, [1021](#)
  - e\_version, [1021](#)
- Elf64\_Phdr, [1022](#)
  - p\_flags, [1023](#)
  - p\_type, [1023](#)
- ELF64\_R\_TYPE
  - ELF binary format, [699](#)
- Elf64\_Rel, [1023](#)
- Elf64\_Rela, [1024](#)
- Elf64\_Shdr, [1025](#)
  - sh\_flags, [1026](#)
  - sh\_type, [1026](#)
- ELF64\_ST\_BIND
  - ELF binary format, [700](#)
- ELF64\_ST\_TYPE
  - ELF binary format, [700](#)
- Elf64\_Sym, [1026](#)
- Elf\_ARM\_SBs
  - ELF binary format, [701](#)
- Elf\_ATs
  - ELF binary format, [701](#)
- Elf\_CIASSs
  - ELF binary format, [701](#)
- Elf\_DATAs
  - ELF binary format, [703](#)
- Elf\_DF\_1s
  - ELF binary format, [703](#)
- Elf\_DF\_P1s
  - ELF binary format, [704](#)
- Elf\_DFs
  - ELF binary format, [704](#)
- Elf\_DTs
  - ELF binary format, [704](#)
- Elf\_EF\_ARM\_s
  - ELF binary format, [705](#)
- Elf\_EIs
  - ELF binary format, [706](#)
- Elf\_EMs
  - ELF binary format, [706](#)
- Elf\_ETs
  - ELF binary format, [708](#)
- Elf\_EVs
  - ELF binary format, [709](#)
- Elf\_MAGs

- ELF binary format, [709](#)
- Elf\_NTscore
  - ELF binary format, [709](#)
- Elf\_NTsobj
  - ELF binary format, [710](#)
- Elf\_OSABIs
  - ELF binary format, [710](#)
- ELF\_PFs
  - ELF binary format, [711](#)
- Elf\_PTsc
  - ELF binary format, [711](#)
- Elf\_R\_386\_s
  - ELF binary format, [712](#)
- Elf\_R\_AARCH64\_s
  - ELF binary format, [713](#)
- Elf\_R\_ARM\_s
  - ELF binary format, [713](#)
- Elf\_R\_X86\_64\_s
  - ELF binary format, [714](#)
- Elf\_SHF\_s\_ARM
  - ELF binary format, [715](#)
- Elf\_SHFs
  - ELF binary format, [715](#)
- Elf\_SHNs
  - ELF binary format, [715](#)
- Elf\_SHTs
  - ELF binary format, [717](#)
- Elf\_STBs
  - ELF binary format, [717](#)
- Elf\_STTs
  - ELF binary format, [718](#)
- ELFCLASS32
  - ELF binary format, [703](#)
- ELFCLASS64
  - ELF binary format, [703](#)
- ELFCLASSNONE
  - ELF binary format, [703](#)
- ELFCLASSNUM
  - ELF binary format, [703](#)
- ELFDATA2LSB
  - ELF binary format, [703](#)
- ELFDATA2MSB
  - ELF binary format, [703](#)
- ELFDATANONE
  - ELF binary format, [703](#)
- ELFDATANUM
  - ELF binary format, [703](#)
- ELFMAG0
  - ELF binary format, [709](#)
- ELFMAG1
  - ELF binary format, [709](#)
- ELFMAG2
  - ELF binary format, [709](#)
- ELFMAG3
  - ELF binary format, [709](#)
- ELFOSABI\_AIX
  - ELF binary format, [711](#)
- ELFOSABI\_ARM



- ELF binary format, [711](#)
- ELFOSABI\_FREEBSD
  - ELF binary format, [711](#)
- ELFOSABI\_HPUX
  - ELF binary format, [710](#)
- ELFOSABI\_IRIX
  - ELF binary format, [711](#)
- ELFOSABI\_LINUX
  - ELF binary format, [711](#)
- ELFOSABI\_MODESTO
  - ELF binary format, [711](#)
- ELFOSABI\_NETBSD
  - ELF binary format, [710](#)
- ELFOSABI\_NONE
  - ELF binary format, [710](#)
- ELFOSABI\_OPENBSD
  - ELF binary format, [711](#)
- ELFOSABI\_SOLARIS
  - ELF binary format, [711](#)
- ELFOSABI\_STANDALONE
  - ELF binary format, [711](#)
- ELFOSABI\_SYSV
  - ELF binary format, [710](#)
- ELFOSABI\_TRU64
  - ELF binary format, [711](#)
- EM\_386
  - ELF binary format, [707](#)
- EM\_68HC05
  - ELF binary format, [707](#)
- EM\_68HC08
  - ELF binary format, [707](#)
- EM\_68HC11
  - ELF binary format, [707](#)
- EM\_68HC12
  - ELF binary format, [707](#)
- EM\_68HC16
  - ELF binary format, [707](#)
- EM\_68K
  - ELF binary format, [707](#)
- EM\_860
  - ELF binary format, [707](#)
- EM\_88K
  - ELF binary format, [707](#)
- EM\_960
  - ELF binary format, [707](#)
- EM\_AARCH64
  - ELF binary format, [708](#)
- EM\_ALPHA
  - ELF binary format, [707](#)
- EM\_ALTERA\_NIOS2
  - ELF binary format, [708](#)
- EM\_ARC
  - ELF binary format, [707](#)
- EM\_ARC\_A5
  - ELF binary format, [708](#)
- EM\_ARM
  - ELF binary format, [707](#)
- EM\_AVR

- ELF binary format, [708](#)
- EM\_COLDFIRE
  - ELF binary format, [707](#)
- EM\_CRIS
  - ELF binary format, [708](#)
- EM\_D10V
  - ELF binary format, [708](#)
- EM\_D30V
  - ELF binary format, [708](#)
- EM\_FIREPATH
  - ELF binary format, [708](#)
- EM\_FR20
  - ELF binary format, [707](#)
- EM\_FR30
  - ELF binary format, [708](#)
- EM\_FX66
  - ELF binary format, [707](#)
- EM\_H8\_300
  - ELF binary format, [707](#)
- EM\_H8\_300H
  - ELF binary format, [707](#)
- EM\_H8\_500
  - ELF binary format, [707](#)
- EM\_H8S
  - ELF binary format, [707](#)
- EM\_HUANY
  - ELF binary format, [708](#)
- EM\_IA\_64
  - ELF binary format, [707](#)
- EM\_JAVELIN
  - ELF binary format, [708](#)
- EM\_M32
  - ELF binary format, [707](#)
- EM\_M32R
  - ELF binary format, [708](#)
- EM\_MICROBLAZE
  - ELF binary format, [708](#)
- EM\_MIPS
  - ELF binary format, [707](#)
- EM\_MIPS\_RS4\_BE
  - ELF binary format, [707](#)
- EM\_MIPS\_X
  - ELF binary format, [707](#)
- EM\_MMIX
  - ELF binary format, [708](#)
- EM\_MN10200
  - ELF binary format, [708](#)
- EM\_MN10300
  - ELF binary format, [708](#)
- EM\_NONE
  - ELF binary format, [707](#)
- EM\_OPENRISC
  - ELF binary format, [708](#)
- EM\_PARISC
  - ELF binary format, [707](#)
- EM\_PDSP
  - ELF binary format, [707](#)
- EM\_PJ

- ELF binary format, [708](#)
- EM\_PPC
  - ELF binary format, [707](#)
- EM\_PRISM
  - ELF binary format, [708](#)
- EM\_RCE
  - ELF binary format, [707](#)
- EM\_RH32
  - ELF binary format, [707](#)
- EM\_RISCV
  - ELF binary format, [708](#)
- EM\_SH
  - ELF binary format, [707](#)
- EM\_SPARC
  - ELF binary format, [707](#)
- EM\_SPARC32PLUS
  - ELF binary format, [707](#)
- EM\_SPARC64
  - ELF binary format, [707](#)
- EM\_SPARCV9
  - ELF binary format, [707](#)
- EM\_ST19
  - ELF binary format, [708](#)
- EM\_ST7
  - ELF binary format, [707](#)
- EM\_ST9PLUS
  - ELF binary format, [707](#)
- EM\_SVX
  - ELF binary format, [708](#)
- EM\_TILEGX
  - ELF binary format, [708](#)
- EM\_TILEPRO
  - ELF binary format, [708](#)
- EM\_TRICORE
  - ELF binary format, [707](#)
- EM\_V800
  - ELF binary format, [707](#)
- EM\_V850
  - ELF binary format, [708](#)
- EM\_VAX
  - ELF binary format, [708](#)
- EM\_VPP500
  - ELF binary format, [707](#)
- EM\_X86\_64
  - ELF binary format, [707](#)
- EM\_XTENSA
  - ELF binary format, [708](#)
- EM\_ZSP
  - ELF binary format, [708](#)
- emerg\_write\_bfm\_t
  - L4virtio::Svr::Console::Features, [2138](#)
- eMMC driver, [88](#)
- empty
  - L4virtio::Svr::Driver\_mem\_region\_t< DATA >, [2199](#)
- enable\_notify
  - L4virtio::Svr::Virtqueue, [2267](#)
- enable\_rx\_irq
  - L4::Uart, [1518](#)
  - L4::Uart\_apb, [1525](#)
- end
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >, [902](#), [903](#)
  - cxx::Bits::Bst< Node, Get\_key, Compare >, [920](#)
  - L4::Kip::Mem\_desc, [1313](#)
- enqueue\_descriptor
  - L4virtio::Driver::Virtqueue, [2091](#)
- enter\_kdebug
  - kdebug.h, [3216](#)
- entry\_ip
  - L4vcpu::Vcpu, [2036](#)
- entry\_sp
  - L4vcpu::Vcpu, [2036](#)
- env
  - L4Re::Env, [1649](#)
- env.h
  - l4re\_env\_t, [2891](#)
- erase
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >, [903](#)
  - cxx::H\_list< T, POLICY >, [940](#)
- err\_no
  - L4::Runtime\_error, [1382](#)
- Error
  - L4virtio::Svr::Bad\_descriptor, [2102](#)
- error
  - l4\_thread\_regs\_t, [1564](#)
- Error codes, [213](#)
  - L4\_E2BIG, [214](#)
  - L4\_EACCESS, [214](#)
  - L4\_EADDRNOTAVAIL, [214](#)
  - L4\_EAGAIN, [214](#)
  - L4\_EBADPROTO, [214](#)
  - L4\_EBUSY, [214](#)
  - L4\_EEXIST, [214](#)
  - L4\_EFAULT, [214](#)
  - L4\_EINVAL, [214](#)
  - L4\_EIO, [214](#)
  - L4\_EIPC\_HI, [214](#)
  - L4\_EIPC\_LO, [214](#)
  - L4\_EMMSGMISSARG, [214](#)
  - L4\_EMMSGTOOLONG, [214](#)
  - L4\_EMMSGTOOSHORT, [214](#)
  - L4\_ENAMETOOLONG, [214](#)
  - L4\_ENODEV, [214](#)
  - L4\_ENOENT, [214](#)
  - L4\_ENOMEM, [214](#)
  - L4\_ENOREPLY, [214](#)
  - L4\_ENOSPC, [214](#)
  - L4\_ENOSYS, [214](#)
  - L4\_ENOTDIR, [214](#)
  - L4\_ENXIO, [214](#)
  - L4\_EOK, [214](#)
  - L4\_EPERM, [214](#)
  - L4\_ERANGE, [214](#)
  - L4\_ERRNOMAX, [214](#)

- l4\_error\_code\_t, [214](#)
- Error Handling, [251](#)
  - l4\_error, [253](#)
  - L4\_IPC\_ENOT\_EXISTENT, [253](#)
  - l4\_ipc\_error, [254](#)
  - l4\_ipc\_error\_code, [255](#)
  - L4\_IPC\_ERROR\_MASK, [252](#)
  - l4\_ipc\_is\_rcv\_error, [256](#)
  - l4\_ipc\_is\_snd\_error, [256](#)
  - L4\_IPC\_REABORTED, [253](#)
  - L4\_IPC\_RECANCELED, [253](#)
  - L4\_IPC\_REMAPFAILED, [253](#)
  - L4\_IPC\_REMSGCUT, [253](#)
  - L4\_IPC\_RERCVPFTO, [253](#)
  - L4\_IPC\_RESNDPFTO, [253](#)
  - L4\_IPC\_RETIMEOUT, [253](#)
  - L4\_IPC\_SEABORTED, [253](#)
  - L4\_IPC\_SECANCELED, [253](#)
  - L4\_IPC\_SEMAPFAILED, [253](#)
  - L4\_IPC\_SEMSGCUT, [253](#)
  - L4\_IPC\_SERCVPFTO, [253](#)
  - L4\_IPC\_SESNDPFTO, [253](#)
  - L4\_IPC\_SETIMEOUT, [253](#)
  - L4\_IPC\_SND\_ERR\_MASK, [253](#)
  - l4\_ipc\_tcr\_error\_t, [252](#)
- ET\_CORE
  - ELF binary format, [709](#)
- ET\_DYN
  - ELF binary format, [709](#)
- ET\_EXEC
  - ELF binary format, [709](#)
- ET\_HIPROC
  - ELF binary format, [709](#)
- ET\_LOPROC
  - ELF binary format, [709](#)
- ET\_NONE
  - ELF binary format, [709](#)
- ET\_REL
  - ELF binary format, [709](#)
- EV\_CURRENT
  - ELF binary format, [709](#)
- EV\_NONE
  - ELF binary format, [709](#)
- Event API, [599](#)
- Event interface, [544](#)
  - l4re\_event\_get\_axis\_info, [545](#)
  - l4re\_event\_get\_buffer, [545](#)
  - l4re\_event\_get\_num\_streams, [546](#)
  - l4re\_event\_get\_stream\_info, [546](#)
  - l4re\_event\_get\_stream\_info\_for\_id, [547](#)
- Event\_buffer\_t
  - L4Re::Event\_buffer\_t< PAYLOAD >, [1670](#)
- Events
  - L4virtio::Svr::Console::Control\_message, [2114](#)
- ex\_regs
  - L4::Thread, [1440](#), [1441](#)
- exc\_handler
  - L4::Thread::Attr, [1453](#)
- exception
  - L4::Exception, [1088](#)
- Exception registers, [276](#)
  - l4\_utcb\_exc, [276](#)
  - l4\_utcb\_exc\_is\_ex\_regs\_exception, [277](#)
  - l4\_utcb\_exc\_is\_pf, [277](#)
  - l4\_utcb\_exc\_pc, [278](#)
  - l4\_utcb\_exc\_pc\_set, [278](#)
- execute
  - L4Re::Ned::Cmd\_control, [1720](#), [1721](#)
- expired
  - Block\_device::Errand::Errand, [814](#)
  - Block\_device::Errand::Poll\_errand, [817](#)
  - L4::lpc\_svr::Timeout, [1276](#)
- ext\_alloc
  - L4vcpu::Vcpu, [2038](#)
- Extended vCPU support, [739](#)
  - l4vcpu\_ext\_alloc, [739](#)
- extra\_str
  - L4::Runtime\_error, [1382](#)
- F\_above
  - L4Re::Video::View, [1937](#)
- F\_auto\_refresh
  - L4Re::Video::Goos, [1915](#)
- F\_dyn\_allocated
  - L4Re::Video::View, [1936](#)
- F\_dynamic\_buffers
  - L4Re::Video::Goos, [1915](#)
- F\_dynamic\_views
  - L4Re::Video::Goos, [1915](#)
- F\_flags\_mask
  - L4Re::Video::View, [1937](#)
- F\_fully\_dynamic
  - L4Re::Video::View, [1937](#)
- F\_l4re\_video\_goos\_auto\_refresh
  - Video API, [579](#)
- F\_l4re\_video\_goos\_dynamic\_buffers
  - Video API, [579](#)
- F\_l4re\_video\_goos\_dynamic\_views
  - Video API, [579](#)
- F\_l4re\_video\_goos\_pointer
  - Video API, [579](#)
- F\_l4re\_video\_view\_above
  - Video API, [579](#)
- F\_l4re\_video\_view\_dyn\_allocated
  - Video API, [579](#)
- F\_l4re\_video\_view\_flags\_mask
  - Video API, [579](#)
- F\_l4re\_video\_view\_none
  - Video API, [579](#)
- F\_l4re\_video\_view\_set\_background
  - Video API, [579](#)
- F\_l4re\_video\_view\_set\_buffer
  - Video API, [579](#)
- F\_l4re\_video\_view\_set\_buffer\_offset
  - Video API, [579](#)
- F\_l4re\_video\_view\_set\_bytes\_per\_line
  - Video API, [579](#)

- F\_l4re\_video\_view\_set\_flags
  - Video API, [579](#)
- F\_l4re\_video\_view\_set\_pixel
  - Video API, [579](#)
- F\_l4re\_video\_view\_set\_position
  - Video API, [579](#)
- F\_none
  - L4Re::Video::View, [1936](#)
- F\_pointer
  - L4Re::Video::Goos, [1915](#)
- F\_set\_background
  - L4Re::Video::View, [1936](#)
- F\_set\_buffer
  - L4Re::Video::View, [1936](#)
- F\_set\_buffer\_offset
  - L4Re::Video::View, [1936](#)
- F\_set\_bytes\_per\_line
  - L4Re::Video::View, [1936](#)
- F\_set\_flags
  - L4Re::Video::View, [1937](#)
- F\_set\_pixel
  - L4Re::Video::View, [1936](#)
- F\_set\_position
  - L4Re::Video::View, [1936](#)
- faccessat
  - L4Re::Vfs::Directory, [1868](#)
- Factory, [293](#)
  - l4\_factory\_create, [295](#)
  - l4\_factory\_create\_factory, [295](#)
  - l4\_factory\_create\_gate, [296](#)
  - l4\_factory\_create\_irq, [297](#)
  - l4\_factory\_create\_task, [298](#)
  - l4\_factory\_create\_thread, [299](#)
  - l4\_factory\_create\_thread\_group, [300](#)
  - l4\_factory\_create\_vcpu\_context, [301](#)
  - l4\_factory\_create\_vm, [302](#)
- factory
  - L4Re::Env, [1650](#), [1651](#)
- fail\_request
  - Block\_device::Pending\_request, [828](#)
- fchmod
  - L4Re::Vfs::Generic\_file, [1885](#)
- fdatasync
  - L4Re::Vfs::Regular\_file, [1897](#)
- feature\_negotiated
  - L4virtio::Driver::Device, [2070](#)
- features
  - l4\_icu\_info\_t, [1550](#)
- Fiasco extensions, [165](#)
  - fiasco\_gdt\_get\_entry\_offset, [166](#)
  - fiasco\_gdt\_set, [167](#)
  - fiasco\_ldt\_set, [168](#)
- fiasco\_amd64\_segment\_info
  - segment.h, [2465](#)
- fiasco\_amd64\_set\_fs
  - segment.h, [2466](#), [2470](#)
- fiasco\_amd64\_set\_segment\_base
  - segment.h, [2467](#), [2471](#)
- fiasco\_dump\_kmem\_stats
  - kdump.h, [3227](#)
- fiasco\_gdt\_get\_entry\_offset
  - Fiasco extensions, [166](#)
- fiasco\_gdt\_set
  - Fiasco extensions, [167](#)
- fiasco\_ldt\_set
  - Fiasco extensions, [168](#)
- fiasco\_tbuf\_log
  - Kernel Tracing, [179](#)
- fiasco\_tbuf\_log\_3val
  - Kernel Tracing, [180](#)
- fiasco\_tbuf\_log\_binary
  - Kernel Tracing, [181](#)
- finalize\_request
  - L4virtio::Svr::Block\_dev\_base< Ds\_data >, [2108](#)
- find
  - cxx::Bits::Bst< Node, Get\_key, Compare >, [920](#)
  - cxx::String, [993](#)
  - L4::Basic\_registry, [1034](#)
  - L4Re::Rm, [1744](#)
  - L4virtio::Svr::Driver\_mem\_list\_t< DATA >, [2190](#)
  - Rm, [2332](#)
- find\_next\_used
  - L4virtio::Driver::Virtqueue, [2092](#)
- find\_node
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >, [904](#)
  - cxx::Bits::Bst< Node, Get\_key, Compare >, [921](#)
- finish
  - L4virtio::Svr::Virtqueue, [2268](#), [2269](#)
- finish\_rx
  - L4virtio::Driver::Virtio\_net\_device, [2080](#)
- first
  - L4::Kip::Mem\_desc, [1314](#)
- first\_free\_cap
  - L4Re::Env, [1651](#)
- first\_free\_utcb
  - L4Re::Env, [1652](#)
- Fixed\_paddr
  - L4Re::Mem\_alloc, [1699](#)
- Flags
  - L4::Type\_info::Demand\_t< CAPS, FLAGS, MEM, PORTS >, [1474](#)
  - L4::Types::Flags< BITS\_ENUM, UNDERLYING >, [1502](#)
  - L4Re::Dataspace::F, [1632](#)
  - L4Re::Video::Goos, [1915](#)
  - L4Re::Video::View, [1936](#)
- flags
  - l4\_exc\_regs\_t, [1547](#)
  - l4\_msgtag\_t, [1556](#)
  - L4Re::Dataspace, [1624](#)
  - l4re\_env\_cap\_entry\_t, [1948](#)
  - L4virtio::Svr::Driver\_mem\_region\_t< DATA >, [2199](#)
- Flexpages, [182](#)
  - L4\_CAP\_FPAGE\_D, [185](#)

- L4\_CAP\_FPAGE\_R, [185](#)
- L4\_cap\_fpage\_rights, [185](#)
- L4\_CAP\_FPAGE\_RO, [185](#)
- L4\_CAP\_FPAGE\_RS, [186](#)
- L4\_CAP\_FPAGE\_RSD, [186](#)
- L4\_CAP\_FPAGE\_RW, [185](#)
- L4\_CAP\_FPAGE\_RWD, [186](#)
- L4\_CAP\_FPAGE\_RWS, [186](#)
- L4\_CAP\_FPAGE\_RWSD, [186](#)
- L4\_CAP\_FPAGE\_S, [185](#)
- L4\_CAP\_FPAGE\_W, [185](#)
- l4\_fpage, [188](#)
- L4\_FPAGE\_ADDR\_BITS, [186](#)
- L4\_FPAGE\_ADDR\_SHIFT, [186](#)
- l4\_fpage\_all, [188](#)
- L4\_fpage\_consts, [186](#)
- l4\_fpage\_contains, [189](#)
- L4\_fpage\_control, [187](#)
- L4\_FPAGE\_CONTROL\_MASK, [187](#)
- L4\_FPAGE\_CONTROL\_OFFSET\_SHIFT, [187](#)
- l4\_fpage\_invalid, [190](#)
- L4\_FPAGE\_IO, [188](#)
- l4\_fpage\_ioport, [190](#)
- l4\_fpage\_max\_order, [190](#)
- l4\_fpage\_memaddr, [191](#)
- L4\_FPAGE\_MEMORY, [188](#)
- L4\_FPAGE\_OBJ, [188](#)
- l4\_fpage\_obj, [192](#)
- l4\_fpage\_page, [192](#)
- L4\_fpage\_rights, [187](#)
- l4\_fpage\_rights, [193](#)
- L4\_FPAGE\_RIGHTS\_ALL, [187](#)
- L4\_FPAGE\_RIGHTS\_BITS, [186](#)
- L4\_FPAGE\_RIGHTS\_MASK, [186](#)
- L4\_FPAGE\_RIGHTS\_SHIFT, [186](#)
- L4\_FPAGE\_RO, [187](#)
- L4\_FPAGE\_RW, [187](#)
- L4\_FPAGE\_RWX, [187](#)
- L4\_FPAGE\_RX, [187](#)
- l4\_fpage\_set\_rights, [193](#)
- l4\_fpage\_size, [194](#)
- L4\_FPAGE\_SIZE\_BITS, [186](#)
- L4\_FPAGE\_SIZE\_SHIFT, [186](#)
- L4\_FPAGE\_SPECIAL, [188](#)
- L4\_fpage\_type, [187](#)
- l4\_fpage\_type, [195](#)
- L4\_FPAGE\_TYPE\_BITS, [186](#)
- L4\_FPAGE\_TYPE\_SHIFT, [186](#)
- L4\_FPAGE\_W, [187](#)
- L4\_FPAGE\_X, [187](#)
- l4\_iofpage, [195](#)
- L4\_IOPORT\_MAX, [185](#)
- l4\_is\_fpage\_valid, [196](#)
- l4\_is\_fpage\_writable, [197](#)
- l4\_obj\_fpage, [198](#)
- L4\_WHOLE\_ADDRESS\_SPACE, [184](#)
- L4\_WHOLE\_IOADDRESS\_SPACE, [184](#)
- flush
- Mac\_table< Size >, [2317](#)
- font.h
  - gfxbitmap\_font\_data, [2835](#)
  - gfxbitmap\_font\_get, [2835](#)
  - gfxbitmap\_font\_height, [2836](#)
  - gfxbitmap\_font\_init, [2836](#)
  - gfxbitmap\_font\_text, [2836](#)
  - gfxbitmap\_font\_text\_scale, [2837](#)
  - gfxbitmap\_font\_width, [2837](#)
- foreach\_available\_event
  - L4Re::Util::Event\_buffer\_consumer\_t< PAYLOAD >, [1815](#)
- fpage
  - L4::Cap\_base, [1052](#)
- free
  - cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc >, [858](#)
  - cxx::Base\_slab\_static< Obj\_size, Slab\_size, Max\_free, Alloc >, [863](#)
  - cxx::List\_alloc, [955](#)
  - cxx::Slab< Type, Slab\_size, Max\_free, Alloc >, [983](#)
  - L4Re::Cap\_alloc, [1610](#)
  - L4Re::Util::\_Cap\_alloc, [1768](#)
  - L4Re::Util::Counting\_cap\_alloc< COUNTER-TYPE, Dbg >, [1803](#)
- free\_area
  - L4Re::Rm, [1745](#)
  - Rm, [2333](#)
- free\_capability
  - L4Re::Util::Names::Name\_space, [1833](#)
- free\_descriptor
  - L4virtio::Driver::Virtqueue, [2092](#)
- free\_dynamic\_entry
  - L4Re::Util::Names::Name\_space, [1833](#)
- free\_epiface
  - L4Re::Util::Names::Name\_space, [1834](#)
- free\_fd
  - L4Re::Vfs::Fs, [1879](#)
- free\_marker
  - l4\_thread\_regs\_t, [1564](#)
- free\_objects
  - cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc >, [858](#)
  - cxx::Base\_slab\_static< Obj\_size, Slab\_size, Max\_free, Alloc >, [864](#)
- from\_ci
  - L4::lpc::Cap< T >, [1146](#)
- from\_dec
  - cxx::String, [994](#)
- From\_device
  - L4Re::Dma\_space, [1641](#)
- from\_hex
  - cxx::String, [994](#)
- from\_raw
  - L4::Types::Flags< BITS\_ENUM, UNDERLYING >, [1503](#)
- fstat

- L4Re::Vfs::Be\_file, [1861](#)
  - L4Re::Vfs::Generic\_file, [1885](#)
- fsync
  - L4Re::Vfs::Regular\_file, [1897](#)
- ftruncate
  - L4Re::Vfs::Regular\_file, [1898](#)
- full
  - L4virtio::Svr::Driver\_mem\_list\_t< DATA >, [2191](#)
- Functions for rendering bitmap data in frame buffers, [657](#)
- Functions for rendering bitmap fonts to frame buffers, [658](#)
- g
  - L4Re::Video::Pixel\_info, [1931](#), [1932](#)
- gcd
  - cxx, [744](#)
- generic\_write
  - L4::Uart, [1519](#)
- get
  - cxx::Bitfield< T, LSB, MSB >, [868](#)
  - cxx::Ref\_ptr< T, CNT >, [975](#)
  - L4::lpc::Iostream, [1156](#), [1157](#)
  - L4::lpc::Istream, [1164](#), [1165](#)
  - L4Re::Core::Ref\_ptr< T, CNT >, [1617](#)
  - L4Re::Env, [1652](#)
  - L4Re::Rm::Unique\_region< T >, [1762](#)
  - L4Re::Video::Color\_component, [1909](#)
  - L4vbus::Gpio\_module, [1984](#)
  - L4vbus::Gpio\_pin, [1994](#)
  - L4virtio::Ptr< T >, [2099](#)
  - Rm::Unique\_region< T >, [2343](#)
- get\_areas
  - L4Re::Rm, [1747](#)
  - Rm, [2334](#)
- get\_attr
  - L4::Vcon, [1534](#)
- get\_avail\_idx
  - L4virtio::Virtqueue, [2282](#)
- get\_axis\_info
  - L4Re::Event, [1663](#)
- get\_buffer
  - L4Re::Event, [1664](#)
- get\_buffer\_demand
  - L4::Epiface, [1078](#)
  - L4::Server\_object\_t< IFACE, BASE >, [1413](#)
- get\_cap
  - L4Re::Env, [1653](#)
- get\_char
  - L4::Uart, [1519](#)
  - L4::Uart\_apb, [1525](#)
- get\_cmd
  - L4virtio::Svr::Dev\_config, [2170](#)
- get\_epiface
  - L4Re::Util::Names::Name\_space, [1835](#)
- get\_fb
  - L4Re::Util::Video::Goos\_svr, [1855](#)
- get\_file
  - L4Re::Vfs::Fs, [1879](#)
- get\_info
  - L4Re::Rm, [1749](#)
  - Rm, [2334](#)
- get\_infos
  - L4::lpc\_gate, [1244](#)
- get\_lock
  - L4Re::Vfs::Regular\_file, [1899](#)
- get\_num\_streams
  - L4Re::Event, [1665](#)
- get\_object\_name
  - L4::Debugger, [1064](#)
- get\_random
  - L4Re::Random, [1730](#)
- get\_rcv\_cap
  - L4::lpc\_svr::Server\_iface, [1268](#)
  - L4Re::Util::Br\_manager, [1788](#)
- get\_regions
  - L4Re::Rm, [1750](#)
  - Rm, [2335](#)
- get\_request
  - L4virtio::Svr::Virtio\_gpio< Request\_handler, Epiface >::Request\_processor, [2237](#)
  - L4virtio::Svr::Virtio\_i2c< Request\_handler, Epiface >::Request\_processor, [2248](#)
  - Virtio\_net\_request, [2356](#)
- get\_resource
  - L4vbus::Device, [1976](#)
- get\_static\_buffer
  - L4Re::Video::Goos, [1919](#)
- get\_status\_flags
  - L4Re::Vfs::Generic\_file, [1886](#)
- get\_stream\_info
  - L4Re::Event, [1666](#)
- get\_stream\_info\_for\_id
  - L4Re::Event, [1667](#)
- get\_stream\_state\_for\_id
  - L4Re::Event, [1668](#)
- get\_tail\_avail\_idx
  - L4virtio::Virtqueue, [2282](#)
- get\_unshifted
  - cxx::Bitfield< T, LSB, MSB >, [868](#)
- get\_value
  - L4::lpc::Varg, [1228](#)
- get\_writeback
  - L4virtio::Svr::Block\_dev\_base< Ds\_data >, [2109](#)
- getitimer
  - L4Re::Itas, [1682](#)
- gfxbitmap\_bmap
  - bitmap.h, [2830](#)
- gfxbitmap\_color\_pix\_t
  - bitmap.h, [2829](#)
- gfxbitmap\_color\_t
  - bitmap.h, [2829](#)
- gfxbitmap\_convert\_color
  - bitmap.h, [2830](#)
- gfxbitmap\_copy
  - bitmap.h, [2830](#)
- gfxbitmap\_fill
  - bitmap.h, [2831](#)

- gfxbitmap\_font\_data
  - font.h, [2835](#)
- gfxbitmap\_font\_get
  - font.h, [2835](#)
- gfxbitmap\_font\_height
  - font.h, [2836](#)
- gfxbitmap\_font\_init
  - font.h, [2836](#)
- gfxbitmap\_font\_text
  - font.h, [2836](#)
- gfxbitmap\_font\_text\_scale
  - font.h, [2837](#)
- gfxbitmap\_font\_width
  - font.h, [2837](#)
- gfxbitmap\_offset, [1027](#)
- gfxbitmap\_set
  - bitmap.h, [2832](#)
- global\_id
  - L4::Debugger, [1065](#)
- gran\_offset
  - l4\_sched\_cpu\_set\_t, [1560](#)
- Grant
  - L4::lpc::Snd\_fpage, [1220](#)
- granularity
  - l4\_sched\_cpu\_set\_t, [1558](#)
- guest\_features
  - L4virtio::Svr::Dev\_config, [2170](#)
- H\_list\_item\_t
  - cxx::H\_list\_item\_t< ELEM\_TYPE >, [945](#)
- handle\_control\_message
  - L4virtio::Svr::Console::Virtio\_con, [2149](#)
- handle\_expired\_timeouts
  - L4::lpc\_svr::Timeout\_queue, [1278](#)
- handle\_l4virtio\_port\_tx
  - Virtio\_switch, [2360](#)
- handle\_mem\_cmd\_write
  - L4virtio::Svr::Device\_t< DATA >, [2185](#)
- handle\_request
  - Block\_device::Pending\_request, [828](#)
- has\_alpha
  - L4Re::Video::Pixel\_info, [1932](#)
- has\_error
  - l4\_msgtag\_t, [1556](#)
- has\_more
  - L4virtio::Svr::Request\_processor, [2203](#)
- hdr
  - L4virtio::Svr::Dev\_config, [2171](#)
- i
  - L4vcpu::Vcpu, [2038](#), [2039](#)
- IA32 Port I/O API, [725](#)
  - l4util\_in16, [726](#)
  - l4util\_in32, [726](#)
  - l4util\_in8, [726](#)
  - l4util\_ins16, [727](#)
  - l4util\_ins32, [727](#)
  - l4util\_ins8, [727](#)
  - l4util\_ioport\_map, [728](#)
- l4util\_out16, [729](#)
- l4util\_out32, [729](#)
- l4util\_out8, [730](#)
- l4util\_outs16, [730](#)
- l4util\_outs32, [730](#)
- l4util\_outs8, [731](#)
- id
  - L4::Kobject\_typeid< T >, [1333](#)
  - L4::Kobject\_typeid< void >, [1336](#)
- id\_received
  - L4::lpc::Snd\_fpage, [1222](#)
- idle\_time
  - L4::Scheduler, [1388](#)
- Ignore\_sigaction
  - L4Re::Itas, [1682](#)
- In\_area
  - L4Re::Rm::F, [1757](#)
  - Rm::F, [2339](#)
- inc
  - L4Re::Util::Counter< COUNTER >, [1797](#)
  - L4Re::Util::Counter\_atomic< COUNTER >, [1800](#)
- include.mk - Header File Role, [40](#)
- indirect\_bfm\_t
  - L4virtio::Virtqueue::Desc::Flags, [2297](#)
- Info
  - L4::Kip::Mem\_desc, [1309](#)
- info
  - L4::lcu, [1114](#)
  - L4::Scheduler, [1389](#)
  - L4Re::Dataspace, [1625](#)
  - L4Re::Mem\_alloc, [1700](#)
  - L4Re::Video::Goos, [1920](#)
  - L4Re::Video::View, [1937](#)
- Info\_acpi\_rsdp
  - L4::Kip::Mem\_desc, [1309](#)
- Info\_sub\_type
  - L4::Kip::Mem\_desc, [1309](#)
- inhibitor.h
  - l4re\_inhibitor\_acquire, [2851](#)
  - l4re\_inhibitor\_next\_lock\_info, [2851](#)
  - l4re\_inhibitor\_release, [2852](#)
- init
  - L4Re::Util::Event\_t< PAYLOAD >, [1825](#)
  - L4virtio::Svr::Driver\_mem\_list\_t< DATA >, [2192](#)
- init\_infos
  - L4Re::Util::Video::Goos\_svr, [1855](#)
- init\_mem\_info
  - L4virtio::Svr::Device\_t< DATA >, [2185](#)
- init\_poll
  - L4Re::Util::Event\_t< PAYLOAD >, [1826](#)
- init\_queue
  - L4virtio::Driver::Virtqueue, [2094](#)
- Initial Environment, [584](#)
  - l4re\_env, [585](#)
  - l4re\_env\_get\_cap, [586](#)
  - l4re\_env\_get\_cap\_e, [586](#)
  - l4re\_env\_get\_cap\_l, [587](#)
  - l4re\_kip, [588](#)



- Initial Environment and Application Bootstrapping, [27](#)
- Initial Memory Allocator and Factory, [30](#)
- initial\_caps
  - L4Re::Env, [1654](#), [1655](#)
- initialize\_rings
  - L4virtio::Driver::Virtqueue, [2096](#)
- insert
  - cxx::Avl\_map< KEY\_TYPE, DATA\_TYPE, COMPARE, ALLOC >, [839](#)
  - cxx::Avl\_tree< Node, Get\_key, Compare >, [850](#)
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >, [904](#)
  - cxx::H\_list< T, POLICY >, [940](#)
- insert\_after
  - cxx::H\_list< T, POLICY >, [941](#)
- insert\_before
  - cxx::H\_list< T, POLICY >, [941](#)
- Integer Types, [457](#)
- interface
  - L4::Meta, [1344](#)
- Interface Definition Language, [32](#)
- Interface for asynchronous ISR handlers with a given IRQ capability., [483](#)
  - l4irq\_request\_cap, [484](#)
- Interface for asynchronous ISR handlers., [481](#)
  - l4irq\_release, [482](#)
  - l4irq\_request, [482](#)
- Interface using direct functionality., [474](#), [479](#)
  - l4irq\_attach, [475](#)
  - l4irq\_attach\_cap, [479](#)
  - l4irq\_attach\_cap\_ft, [480](#)
  - l4irq\_attach\_ft, [475](#)
  - l4irq\_attach\_thread, [476](#)
  - l4irq\_attach\_thread\_cap, [480](#)
  - l4irq\_attach\_thread\_cap\_ft, [481](#)
  - l4irq\_attach\_thread\_ft, [476](#)
  - l4irq\_detach, [477](#)
  - l4irq\_unmask, [477](#)
  - l4irq\_unmask\_and\_wait\_any, [477](#)
  - l4irq\_wait, [478](#)
  - l4irq\_wait\_any, [478](#)
- Internal, [610](#)
  - L4SHMC\_RINGBUF\_DATA, [611](#)
  - L4SHMC\_RINGBUF\_DATA\_SIZE, [611](#)
  - L4SHMC\_RINGBUF\_HEAD, [611](#)
- Internal constants, [643](#)
- Internal functions, [685](#)
- Internal Helpers, [200](#)
- internal\_loop
  - L4::Server< LOOP\_HOOKS >, [1402](#)
- Interrupt controller, [325](#)
  - l4\_icu\_bind, [328](#)
  - l4\_icu\_bind\_u, [329](#)
  - L4\_ICU\_FLAG\_MSI, [328](#)
  - L4\_icu\_flags, [327](#)
  - l4\_icu\_info, [330](#)
  - l4\_icu\_info\_t, [327](#)
  - l4\_icu\_info\_u, [331](#)
  - l4\_icu\_mask, [332](#)
  - l4\_icu\_mask\_u, [333](#)
  - l4\_icu\_msi\_info, [334](#)
  - l4\_icu\_msi\_info\_u, [335](#)
  - l4\_icu\_set\_mode, [336](#)
  - l4\_icu\_set\_mode\_u, [337](#)
  - l4\_icu\_unbind, [338](#)
  - l4\_icu\_unbind\_u, [339](#)
  - l4\_icu\_unmask, [340](#)
  - l4\_icu\_unmask\_u, [341](#)
- Introduction, [3](#)
- Invalid
  - L4::Cap\_base, [1047](#)
  - L4virtio::Ptr< T >, [2099](#)
- Invalid\_capability
  - L4::Invalid\_capability, [1123](#)
- Invalid\_type
  - L4virtio::Ptr< T >, [2099](#)
- io
  - L4::lpc::Gen\_fpage, [1149](#)
- io
  - L4::lpc::Rcv\_fpage, [1209](#)
  - L4::lpc::Snd\_fpage, [1222](#)
- IO interface, [463](#)
  - L4IO\_DEVICE\_ANY, [465](#)
  - L4IO\_DEVICE\_INVALID, [465](#)
  - L4IO\_DEVICE\_OTHER, [465](#)
  - L4IO\_DEVICE\_PCI, [465](#)
  - l4io\_device\_types\_t, [464](#)
  - L4IO\_DEVICE\_USB, [465](#)
  - l4io\_has\_resource, [466](#)
  - l4io\_iomem\_flags\_t, [465](#)
  - l4io\_lookup\_device, [467](#)
  - l4io\_lookup\_resource, [467](#)
  - L4IO\_MEM\_CACHED, [465](#)
  - L4IO\_MEM\_EAGER\_MAP, [465](#)
  - L4IO\_MEM\_NONCACHED, [465](#)
  - L4IO\_MEM\_USE\_MTRR, [465](#)
  - L4IO\_MEM\_USE\_RESERVED\_AREA, [465](#)
  - l4io\_release\_iomem, [468](#)
  - l4io\_release\_ioport, [468](#)
  - l4io\_request\_iomem, [468](#)
  - l4io\_request\_iomem\_region, [469](#)
  - l4io\_request\_ioport, [470](#)
  - l4io\_request\_resource\_iomem, [470](#)
  - L4IO\_RESOURCE\_ANY, [465](#)
  - L4IO\_RESOURCE\_INVALID, [465](#)
  - L4IO\_RESOURCE\_IRQ, [465](#)
  - L4IO\_RESOURCE\_MEM, [465](#)
  - L4IO\_RESOURCE\_PORT, [465](#)
  - l4io\_resource\_t, [464](#)
  - l4io\_resource\_types\_t, [465](#)
- Io, the Io Server, [62](#)
- io.h
  - l4io\_get\_root\_device, [2538](#)
  - l4io\_iterate\_devices, [2538](#)
  - l4io\_request\_all\_ioports, [2538](#)
  - l4io\_request\_icu, [2539](#)



- io\_page\_fault
  - L4::Io\_pager, 1126
- ioctl
  - L4Re::Vfs::Special\_file, 1906
- loststream
  - L4::lpc::loststream, 1154
- IPC Helpers, 472
  - throw\_ipc\_exception, 472, 473
- IPC-Gate API, 286
  - l4\_ipc\_gate\_get\_infos, 287
  - l4\_rcv\_ep\_bind\_snd\_destination, 288
  - l4\_rcv\_ep\_bind\_thread, 289
- ipc.h
  - l4\_ipc\_to\_errno, 3194
- ipc\_client
  - L4\_RPC\_DEF, 3113
- ipc\_iface
  - L4\_INLINE\_RPC, 3120
  - L4\_INLINE\_RPC\_NF, 3120
  - L4\_INLINE\_RPC\_NF\_OP, 3120
  - L4\_INLINE\_RPC\_OP, 3121
  - L4\_RPC, 3121
  - L4\_RPC\_NF, 3122
  - L4\_RPC\_NF\_OP, 3122
  - L4\_RPC\_OP, 3123
- ipc\_stream
  - operator<<, 2601–2603
  - operator>>, 2603–2607
- irq
  - L4Re::Util::Event\_t< PAYLOAD >, 1826
- IRQ handling library, 474
- irq\_disable\_save
  - L4vcpu::Vcpu, 2039
- irq\_enable
  - L4vbus::Pci\_dev, 2007
  - L4vbus::Pci\_host\_bridge, 2013
  - L4vcpu::Vcpu, 2039
- irq\_restore
  - L4vcpu::Vcpu, 2040
- IRQs, 342
  - l4\_irq\_bind\_vcpu, 344
  - l4\_irq\_bind\_vcpu\_u, 345
  - l4\_irq\_detach, 347
  - l4\_irq\_detach\_u, 348
  - L4\_IRQ\_F\_BOTH, 344
  - L4\_IRQ\_F\_BOTH\_EDGE, 344
  - L4\_IRQ\_F\_CLEAR\_WAKEUP, 344
  - L4\_IRQ\_F\_EDGE, 344
  - L4\_IRQ\_F\_LEVEL, 344
  - L4\_IRQ\_F\_LEVEL\_HIGH, 344
  - L4\_IRQ\_F\_LEVEL\_LOW, 344
  - L4\_IRQ\_F\_MASK, 344
  - L4\_IRQ\_F\_NEG, 344
  - L4\_IRQ\_F\_NEG\_EDGE, 344
  - L4\_IRQ\_F\_NONE, 344
  - L4\_IRQ\_F\_POS, 344
  - L4\_IRQ\_F\_POS\_EDGE, 344
  - L4\_IRQ\_F\_SET\_MODE, 344
  - L4\_IRQ\_F\_SET\_WAKEUP, 344
  - l4\_irq\_mode, 343
  - l4\_irq\_receive, 349
  - l4\_irq\_receive\_u, 350
  - l4\_irq\_trigger, 351
  - l4\_irq\_trigger\_u, 352
  - l4\_irq\_unmask, 353
  - l4\_irq\_unmask\_u, 354
  - l4\_irq\_wait, 355
  - l4\_irq\_wait\_u, 356
- is\_compatible
  - L4vbus::Device, 1977
- is\_compound
  - L4::lpc::Snd\_fpage, 1223
- is\_irq\_entry
  - L4vcpu::Vcpu, 2040
- is\_nil
  - L4::lpc::Varg, 1229
- is\_of
  - L4::lpc::Varg, 1230
- is\_of\_int
  - L4::lpc::Varg, 1231
- is\_online
  - L4::Scheduler, 1390
- is\_page\_fault\_entry
  - L4vcpu::Vcpu, 2041
- is\_static
  - L4Re::Util::Dataspace\_svr, 1809
- is\_valid
  - L4::Cap\_base, 1053
  - L4Re::Rm::Unique\_region< T >, 1763
  - L4virtio::Ptr< T >, 2100
  - Rm::Unique\_region< T >, 2343
- is\_virtual
  - L4::Kip::Mem\_desc, 1315
- is\_writable
  - L4virtio::Svr::Driver\_mem\_region\_t< DATA >, 2200
- Istream
  - L4::lpc::Istream, 1163
- itas
  - L4Re::Env, 1655
- Item\_bytes
  - L4::lpc::Msg, 768
- Item\_words
  - L4::lpc::Msg, 768
- iter
  - cxx::Bits::Basic\_list< POLICY >, 911
  - cxx::H\_list< T, POLICY >, 942
- Iterator
  - cxx::Avl\_tree< Node, Get\_key, Compare >, 850
- kdebug.h
  - \_\_kdebug\_3\_text, 3210
  - \_\_kdebug\_op, 3211
  - \_\_kdebug\_op\_1, 3213
  - \_\_kdebug\_text, 3214
  - enter\_kdebug, 3216
  - l4\_kdebug\_ops\_t, 3210

- outchar, [3216](#)
- outdec, [3217](#)
- outhex12, [3218](#)
- outhex16, [3218](#)
- outhex20, [3219](#)
- outhex32, [3219](#)
- outhex64, [3221](#)
- outhex8, [3221](#)
- outnstring, [3222](#)
- outstring, [3223](#)
- outumword, [3224](#)
- kdump.h
  - fiasco\_dump\_kmem\_stats, [3227](#)
- Kept\_ds
  - L4Re::Rm, [1737](#)
  - Rm, [2327](#)
- Kernel
  - L4Re::Rm::F, [1758](#)
  - Rm::F, [2339](#)
- Kernel ABI, [18](#)
- Kernel Debugger, [169](#)
  - l4\_debugger\_add\_image\_info, [170](#)
  - l4\_debugger\_get\_object\_name, [171](#)
  - l4\_debugger\_global\_id, [172](#)
  - l4\_debugger\_kobj\_to\_id, [172](#)
  - l4\_debugger\_query\_log\_name, [174](#)
  - l4\_debugger\_query\_log\_typeid, [175](#)
  - l4\_debugger\_set\_object\_name, [176](#)
  - l4\_debugger\_switch\_log, [176](#)
- Kernel Factory, [48](#)
- Kernel Information Dump, [178](#)
- Kernel Interface Page, [436](#)
  - l4\_kernel\_info\_t, [438](#)
  - l4\_kernel\_info\_version\_offset, [439](#)
  - l4\_kip, [440](#)
  - l4\_kip\_clock, [440](#)
  - l4\_kip\_clock\_lw, [441](#)
  - l4\_kip\_clock\_ns, [442](#)
  - L4\_KIP\_OFFS\_READ\_NS, [438](#)
  - L4\_KIP\_OFFS\_READ\_US, [438](#)
  - l4\_kip\_version, [442](#)
  - l4\_kip\_version\_string, [444](#)
- Kernel Interface Page API, [719](#)
  - l4util\_kip\_for\_each\_feature, [719](#)
  - l4util\_kip\_kernel\_abi\_version, [720](#)
  - l4util\_kip\_kernel\_has\_feature, [721](#)
- Kernel Objects, [284](#)
- Kernel Tracing, [179](#)
  - fiasco\_tbuf\_log, [179](#)
  - fiasco\_tbuf\_log\_3val, [180](#)
  - fiasco\_tbuf\_log\_binary, [181](#)
- Kernel-provided semaphore, [370](#)
  - l4\_semaphore\_down, [371](#)
  - l4\_semaphore\_up, [372](#)
- kip.h
  - l4\_kip\_for\_each\_feature, [3364](#)
  - l4\_kip\_kernel\_has\_feature, [3364](#)
- kobj\_to\_id
  - L4::Debugger, [1066](#)
- kobject\_typeid
  - L4 kernel object type information, [292](#)
- Kumem allocator utility, [576](#)
- Kumem utilities, [594](#)
  - kumem\_alloc, [595](#)
- kumem\_alloc
  - Kumem utilities, [595](#)
- kumem\_alloc.h
  - l4re\_util\_kumem\_alloc, [2867](#)
- L
  - cxx::Bits::Direction, [930](#)
- L4, [748](#)
  - cap\_cast, [752](#), [753](#)
  - cap\_dynamic\_cast, [754](#)
  - cap\_reinterpret\_cast, [755](#), [756](#)
  - PROTO\_ANY, [752](#)
  - PROTO\_EMPTY, [752](#)
  - round\_order, [756](#)
  - trunc\_order, [757](#)
- L4 IPC Opcodes, [484](#)
  - L4\_ICU\_OP\_BIND, [485](#)
  - L4\_ICU\_OP\_INFO, [485](#)
  - L4\_ICU\_OP\_MASK, [486](#)
  - L4\_ICU\_OP\_MSI\_INFO, [486](#)
  - L4\_ICU\_OP\_SET\_MODE, [486](#)
  - L4\_ICU\_OP\_UNBIND, [485](#)
  - L4\_ICU\_OP\_UNMASK, [486](#)
  - L4\_icu\_opcode, [485](#)
  - L4\_IPC\_GATE\_BIND\_OP, [486](#)
  - L4\_IPC\_GATE\_GET\_INFO\_OP, [486](#)
  - L4\_ipc\_gate\_ops, [486](#)
  - L4\_PLATFORM\_CTL\_CPU\_ALLOW\_SHUTDOWN\_OP, [487](#)
  - L4\_PLATFORM\_CTL\_CPU\_DISABLE\_OP, [487](#)
  - L4\_PLATFORM\_CTL\_CPU\_ENABLE\_OP, [487](#)
  - L4\_platform\_ctl\_ops, [486](#)
  - L4\_PLATFORM\_CTL\_SET\_TASK\_ASID\_OP, [487](#)
  - L4\_PLATFORM\_CTL\_SYS\_SHUTDOWN\_OP, [487](#)
  - L4\_PLATFORM\_CTL\_SYS\_SUSPEND\_OP, [486](#)
  - L4\_TASK\_ADD\_KU\_MEM\_OP, [487](#)
  - L4\_TASK\_CAP\_INFO\_OP, [487](#)
  - L4\_TASK\_LDT\_SET\_X86\_OP, [487](#)
  - L4\_TASK\_MAP\_OP, [487](#)
  - L4\_TASK\_MAP\_VGICC\_ARM\_OP, [487](#)
  - L4\_task\_ops, [487](#)
  - L4\_TASK\_UNMAP\_OP, [487](#)
  - L4\_THREAD\_AMD64\_GET\_SEGMENT\_INFO\_OP, [488](#)
  - L4\_THREAD\_AMD64\_SET\_SEGMENT\_BASE\_OP, [488](#)
  - L4\_THREAD\_ARM\_TPIDRURO\_OP, [488](#)
  - L4\_THREAD\_CONTROL\_OP, [487](#)
  - L4\_THREAD\_EX\_REGS\_OP, [487](#)
  - L4\_THREAD\_MODIFY\_SENDER\_OP, [487](#)
  - L4\_THREAD\_OPCODE\_MASK, [488](#)
  - L4\_thread\_ops, [487](#)

- L4\_THREAD\_REGISTER\_DELETE\_IRQ\_OP, [487](#)
- L4\_THREAD\_REGISTER\_DOORBELL\_IRQ\_OP, [487](#)
- L4\_THREAD\_STATS\_OP, [487](#)
- L4\_THREAD\_SWITCH\_OP, [487](#)
- L4\_THREAD\_VCPU\_CONTROL\_OP, [487](#)
- L4\_THREAD\_VCPU\_RESUME\_OP, [487](#)
- L4\_THREAD\_X86\_GDT\_OP, [487](#)
- L4\_VCON\_GET\_ATTR\_OP, [488](#)
- L4\_vcon\_ops, [488](#)
- L4\_VCON\_READ\_OP, [488](#)
- L4\_VCON\_SET\_ATTR\_OP, [488](#)
- L4\_VCON\_WRITE\_OP, [488](#)
- L4 kernel object type information, [291](#)
  - kobject\_typeid, [292](#)
- L4 Vbus functions, [502](#)
  - l4vbus\_assign\_dma\_domain, [504](#)
  - L4vbus\_dma\_domain\_assign\_flags, [503](#)
  - L4VBUS\_DMAD\_BIND, [503](#)
  - L4VBUS\_DMAD\_KERNEL\_DMA\_SPACE, [503](#)
  - L4VBUS\_DMAD\_L4RE\_DMA\_SPACE, [503](#)
  - L4VBUS\_DMAD\_UNBIND, [503](#)
  - l4vbus\_get\_adr, [504](#)
  - l4vbus\_get\_device, [505](#)
  - l4vbus\_get\_device\_by\_hid, [506](#)
  - l4vbus\_get\_hid, [507](#)
  - l4vbus\_get\_next\_device, [507](#)
  - l4vbus\_get\_resource, [509](#)
  - l4vbus\_is\_compatible, [510](#)
  - l4vbus\_release\_ioport, [510](#)
  - l4vbus\_request\_ioport, [511](#)
  - l4vbus\_vicu\_get\_cap, [512](#)
- L4 VIRTIO Block Device, [499](#)
  - L4virtio\_block\_operations, [499](#)
  - L4VIRTIO\_BLOCK\_S\_IOERR, [500](#)
  - L4VIRTIO\_BLOCK\_S\_OK, [500](#)
  - L4VIRTIO\_BLOCK\_S\_UNSUPP, [500](#)
  - L4virtio\_block\_status, [500](#)
  - L4VIRTIO\_BLOCK\_T\_DISCARD, [500](#)
  - L4VIRTIO\_BLOCK\_T\_FLUSH, [500](#)
  - L4VIRTIO\_BLOCK\_T\_GET\_ID, [500](#)
  - L4VIRTIO\_BLOCK\_T\_IN, [500](#)
  - L4VIRTIO\_BLOCK\_T\_OUT, [500](#)
  - L4VIRTIO\_BLOCK\_T\_WRITE\_ZEROES, [500](#)
- L4 VIRTIO Input Device, [500](#)
- L4 VIRTIO Interface, [488](#)
- L4 VIRTIO Network Device, [501](#)
- L4 VIRTIO Transport Layer, [489](#)
  - L4\_virtio\_cmd, [491](#)
  - L4\_virtio\_irq\_status, [491](#)
  - L4\_virtio\_opcodes, [492](#)
  - L4VIRTIO\_CMD\_CFG\_CHANGED, [491](#)
  - L4VIRTIO\_CMD\_CFG\_QUEUE, [491](#)
  - L4VIRTIO\_CMD\_MASK, [491](#)
  - L4VIRTIO\_CMD\_NONE, [491](#)
  - L4VIRTIO\_CMD\_NOTIFY\_QUEUE, [491](#)
  - L4VIRTIO\_CMD\_SET\_STATUS, [491](#)
  - l4virtio\_config\_queue, [494](#)
  - l4virtio\_config\_queue\_t, [491](#)
  - l4virtio\_config\_queues, [494](#)
  - l4virtio\_device\_config, [495](#)
  - l4virtio\_device\_config\_ds, [495](#)
  - L4virtio\_device\_ids, [492](#)
  - l4virtio\_device\_notification\_irq, [495](#)
  - L4virtio\_device\_status, [493](#)
  - L4virtio\_feature\_bits, [493](#)
  - L4VIRTIO\_FEATURE\_CMD\_CONFIG, [494](#)
  - L4VIRTIO\_FEATURE\_VERSION\_1, [494](#)
  - L4VIRTIO\_ID\_9P, [493](#)
  - L4VIRTIO\_ID\_BALLOON, [493](#)
  - L4VIRTIO\_ID\_BLOCK, [492](#)
  - L4VIRTIO\_ID\_CAIF, [493](#)
  - L4VIRTIO\_ID\_CONSOLE, [492](#)
  - L4VIRTIO\_ID\_CRYPTOP, [493](#)
  - L4VIRTIO\_ID\_FS, [493](#)
  - L4VIRTIO\_ID\_GPIO, [493](#)
  - L4VIRTIO\_ID\_GPU, [493](#)
  - L4VIRTIO\_ID\_I2C, [493](#)
  - L4VIRTIO\_ID\_INPUT, [493](#)
  - L4VIRTIO\_ID\_NET, [492](#)
  - L4VIRTIO\_ID\_RNG, [492](#)
  - L4VIRTIO\_ID\_RPMSG, [493](#)
  - L4VIRTIO\_ID\_RPROC\_SERIAL, [493](#)
  - L4VIRTIO\_ID\_SCMI, [493](#)
  - L4VIRTIO\_ID\_SCSI, [493](#)
  - L4VIRTIO\_ID\_SOCKET, [493](#)
  - L4VIRTIO\_ID\_VSOCKET, [493](#)
  - L4VIRTIO\_IRQ\_STATUS\_CONFIG, [492](#)
  - L4VIRTIO\_IRQ\_STATUS\_VRING, [492](#)
  - L4VIRTIO\_OP\_CONFIG\_QUEUE, [492](#)
  - L4VIRTIO\_OP\_DEVICE\_CONFIG, [492](#)
  - L4VIRTIO\_OP\_GET\_DEVICE\_IRQ, [492](#)
  - L4VIRTIO\_OP\_REGISTER\_DS, [492](#)
  - L4VIRTIO\_OP\_SET\_STATUS, [492](#)
  - l4virtio\_register\_ds, [497](#)
  - l4virtio\_set\_status, [498](#)
  - L4VIRTIO\_STATUS\_ACKNOWLEDGE, [493](#)
  - L4VIRTIO\_STATUS\_DEVICE\_NEEDS\_RESET, [493](#)
  - L4VIRTIO\_STATUS\_DRIVER, [493](#)
  - L4VIRTIO\_STATUS\_DRIVER\_OK, [493](#)
  - L4VIRTIO\_STATUS\_FAILED, [493](#)
  - L4VIRTIO\_STATUS\_FEATURES\_OK, [493](#)
- l4/cxx/alloc.h, [2540](#), [2541](#)
- l4/cxx/arith, [2541](#)
- l4/cxx/atomic.h, [3324](#), [3325](#)
- l4/cxx/avl\_map, [2542](#), [2543](#)
- l4/cxx/avl\_set, [2544](#), [2546](#)
- l4/cxx/avl\_tree, [2549](#), [2552](#)
- l4/cxx/basic\_ostream, [2556](#), [2557](#)
- l4/cxx/basic\_vector.h, [2560](#)
- l4/cxx/bitfield, [2561](#)
- l4/cxx/bitmap, [2563](#)
- l4/cxx/bits/bst.h, [2566](#), [2568](#)
- l4/cxx/bits/bst\_base.h, [2570](#), [2572](#)
- l4/cxx/bits/bst\_iter.h, [2573](#), [2575](#)

l4/cxx/bits/list\_basics.h, 2576  
l4/cxx/bits/smart\_ptr\_list.h, 2578, 2580  
l4/cxx/bits/type\_traits.h, 2581  
l4/cxx/dlist, 2584  
l4/cxx/exceptions, 2589, 2591  
l4/cxx/hlist, 2593  
l4/cxx/iostream, 2595, 2596  
l4/cxx/ipc\_helper, 2596, 2598  
l4/cxx/ipc\_server, 3130, 3132  
l4/cxx/ipc\_stream, 2598, 2608  
l4/cxx/ipc\_timeout\_queue, 2616  
l4/cxx/l4iostream, 2618, 2619  
l4/cxx/l4types.h, 2619, 2620  
l4/cxx/list, 2620  
l4/cxx/list\_alloc, 2624  
l4/cxx/lock\_guard.h, 2630, 2631  
l4/cxx/main\_thread, 2632, 2633  
l4/cxx/minmax, 2633  
l4/cxx/numeric, 2634  
l4/cxx/observer, 2634  
l4/cxx/pair, 2635, 2636  
l4/cxx/ref\_ptr, 2637  
l4/cxx/ref\_ptr\_list, 2640, 2642  
l4/cxx/slab\_alloc, 2642  
l4/cxx/slist, 2646  
l4/cxx/static\_container, 2649  
l4/cxx/static\_vector, 2649  
l4/cxx/std\_alloc, 2650  
l4/cxx/std\_ops, 2650  
l4/cxx/string, 2651  
l4/cxx/string.h, 2654, 2655  
l4/cxx/thread, 3282, 3284  
l4/cxx/type\_list, 2655  
l4/cxx/type\_traits, 2656  
l4/cxx/unique\_ptr, 2660  
l4/cxx/unique\_ptr\_list, 2662, 2664  
l4/cxx/utils, 2664  
l4/cxx/weak\_ref, 2665  
l4/irq/irq.h, 2669, 2671  
l4/l4re\_vfs/backend, 2677  
l4/l4re\_vfs/impl/default\_ops\_impl.h, 2680  
l4/l4re\_vfs/impl/fd\_store.h, 2681  
l4/l4re\_vfs/impl/fd\_store\_impl.h, 2682  
l4/l4re\_vfs/impl/ns\_fs.h, 2682  
l4/l4re\_vfs/impl/ns\_fs\_impl.h, 2683  
l4/l4re\_vfs/impl/ro\_file.h, 2688  
l4/l4re\_vfs/impl/ro\_file\_impl.h, 2688  
l4/l4re\_vfs/impl/vcon\_stream.h, 2690  
l4/l4re\_vfs/impl/vcon\_stream\_impl.h, 2690  
l4/l4re\_vfs/impl/vfs\_impl.h, 2692  
l4/l4re\_vfs/vfs.h, 2705  
l4/l4virtio/client/l4virtio, 2716  
l4/l4virtio/client/virtio-block, 2736  
l4/l4virtio/client/virtio-net, 2714  
l4/l4virtio/l4virtio, 2719  
l4/l4virtio/server/l4virtio, 2721  
l4/l4virtio/server/virtio, 2732  
l4/l4virtio/server/virtio-block, 2739  
l4/l4virtio/server/virtio-console, 2746  
l4/l4virtio/server/virtio-console-device, 2752  
l4/l4virtio/server/virtio-gpio-device, 2755  
l4/l4virtio/server/virtio-i2c-device, 2761  
l4/l4virtio/server/virtio-rng-device, 2765  
l4/l4virtio/server/virtio-scmi-device, 2768  
l4/l4virtio/virtio.h, 2777  
l4/l4virtio/virtio\_block.h, 2780  
l4/l4virtio/virtio\_input.h, 2781  
l4/l4virtio/virtio\_net.h, 2438  
l4/l4virtio/virtqueue, 2782  
l4/libblock-device/block\_device\_mgr.h, 2786  
l4/libblock-device/debug.h, 2398  
l4/libblock-device/device.h, 2792  
l4/libblock-device/errand.h, 2793  
l4/libblock-device/gpt.h, 2795  
l4/libblock-device/inout\_memory.h, 2795  
l4/libblock-device/part\_device.h, 2797  
l4/libblock-device/partition.h, 2799  
l4/libblock-device/request.h, 2425  
l4/libblock-device/scheduler.h, 2802  
l4/libblock-device/types.h, 2811  
l4/libblock-device/virtio\_client.h, 2817  
l4/libedid/edid.h, 2826, 2827  
l4/libgfxbitmap/bitmap.h, 2827, 2832  
l4/libgfxbitmap/font.h, 2833, 2838  
l4/libgfxbitmap/support, 2839, 2840  
l4/re/c/dataspace.h, 2840, 2842  
l4/re/c/debug.h, 2399, 2400  
l4/re/c/dma\_space.h, 2843, 2845  
l4/re/c/event.h, 2845, 2847  
l4/re/c/event\_buffer.h, 2849  
l4/re/c/inhibitor.h, 2850, 2853  
l4/re/c/log.h, 2853, 2854  
l4/re/c/mem\_alloc.h, 2855, 2856  
l4/re/c/namespace.h, 2856, 2858  
l4/re/c/parent.h, 2858, 2860  
l4/re/c/rm.h, 2860, 2862  
l4/re/c/util/cap\_alloc.h, 2864, 2865  
l4/re/c/util/kumem\_alloc.h, 2866, 2867  
l4/re/c/util/video/goos\_fb.h, 2868  
l4/re/c/video/colors.h, 2869, 2871  
l4/re/c/video/goos.h, 2871, 2873  
l4/re/c/video/view.h, 2874, 2876  
l4/re/cap\_alloc, 2943, 2944  
l4/re/console, 2877  
l4/re/consts, 3102, 3103  
l4/re/consts.h, 3093, 3094  
l4/re/dataspace, 2878, 2879  
l4/re/dataspace-sys.h, 2881, 2882  
l4/re/dbg\_events, 2882  
l4/re/debug, 2959, 2961  
l4/re/dma\_space, 2882, 2884  
l4/re/elf\_aux.h, 2885, 2886  
l4/re/env, 2887, 2888  
l4/re/env.h, 2889, 2891  
l4/re/error\_helper, 2893, 2894  
l4/re/event, 2964

l4/re/event-sys.h, 2896  
l4/re/event.h, 2848  
l4/re/event\_enums.h, 2896  
l4/re/impl/dataspace\_impl.h, 2902, 2903  
l4/re/impl/mem\_alloc\_impl.h, 2905, 2906  
l4/re/impl/namespace\_impl.h, 2906, 2907  
l4/re/impl/rm\_impl.h, 2909, 2910  
l4/re/inhibitor, 2911  
l4/re/inhibitor-sys.h, 2911  
l4/re/itas, 2912  
l4/re/l4aux.h, 2913, 2914  
l4/re/log, 2914, 2915  
l4/re/log-sys.h, 2916  
l4/re/mem\_alloc, 2916, 2918  
l4/re/mem\_alloc-sys.h, 2919  
l4/re/mmio\_space, 2920, 2921  
l4/re/namespace, 2921, 2923  
l4/re/namespace-sys.h, 2924  
l4/re/parent, 2925, 2926  
l4/re/parent-sys.h, 2926, 2927  
l4/re/protocols.h, 2927, 2928  
l4/re/random, 2928, 2930  
l4/re/remote\_access, 2930  
l4/re/rm, 2931, 2932  
l4/re/rm-sys.h, 2936, 2937  
l4/re/shared\_cap, 2996, 2998  
l4/re/unique\_cap, 3001, 3003  
l4/re/util/bitmap\_cap\_alloc, 2937, 2938  
l4/re/util/br\_manager, 2939  
l4/re/util/cap, 2941, 2943  
l4/re/util/cap\_alloc, 2946, 2948  
l4/re/util/cap\_alloc\_impl.h, 2949, 2950  
l4/re/util/counting\_cap\_alloc, 2951, 2953  
l4/re/util/dataspace\_svr, 2956  
l4/re/util/debug, 2961  
l4/re/util/env\_ns, 2963  
l4/re/util/event, 2966, 2967  
l4/re/util/event\_buffer, 2969  
l4/re/util/event\_svr, 2970  
l4/re/util/icu\_svr, 2971  
l4/re/util/item\_alloc, 2973, 2975  
l4/re/util/kumem\_alloc, 2976, 2977  
l4/re/util/meta, 3243  
l4/re/util/name\_space\_svr, 2977  
l4/re/util/object\_registry, 2983  
l4/re/util/poll\_timeout\_kipclock, 2985  
l4/re/util/region\_mapping, 2986, 2987  
l4/re/util/region\_mapping\_svr, 2993  
l4/re/util/shared\_cap, 2999, 3000  
l4/re/util/unique\_cap, 3003, 3005  
l4/re/util/vcon\_svr, 3006  
l4/re/util/video/goos\_fb, 3007  
l4/re/util/video/goos\_svr, 3009  
l4/re/video/colors, 3010  
l4/re/video/goos, 3012  
l4/re/video/goos-sys.h, 3015, 3016  
l4/re/video/view, 3016  
l4/shmc/ringbuf.h, 3016, 3024  
l4/shmc/shmc.h, 3026, 3029  
l4/sigma0/sigma0.h, 3031, 3033  
l4/sys/\_\_kernel\_object\_impl.h, 3034  
l4/sys/\_\_ktrace-impl.h, 3034, 3036  
l4/sys/\_\_l4\_fpage.h, 3036  
l4/sys/\_\_platform\_control-arm.h, 3041  
l4/sys/\_\_task-arm.h, 3041  
l4/sys/\_\_timeout.h, 3042  
l4/sys/\_\_typeinfo.h, 3044, 3046  
l4/sys/\_\_vcpu-arm.h, 3056  
l4/sys/\_\_vm-arm.h, 3057, 3059  
l4/sys/\_\_vm-svm.h, 3059  
l4/sys/\_\_vm-vmx.h, 3061  
l4/sys/arm\_smccc, 3067, 3069  
l4/sys/arm\_smccc.h, 3069, 3071  
l4/sys/assert.h, 3318, 3320  
l4/sys/cache.h, 3079, 3081  
l4/sys/capability, 3082, 3085  
l4/sys/compiler.h, 3086, 3088  
l4/sys/consts.h, 3095, 3097  
l4/sys/cxx/capability.h, 3099  
l4/sys/cxx/consts, 3103  
l4/sys/cxx/ipc\_array, 3104  
l4/sys/cxx/ipc\_basics, 3107  
l4/sys/cxx/ipc\_client, 3111, 3113  
l4/sys/cxx/ipc\_epiface, 3114  
l4/sys/cxx/ipc\_iface, 3118, 3124  
l4/sys/cxx/ipc\_legacy, 3129  
l4/sys/cxx/ipc\_ret\_array, 3129  
l4/sys/cxx/ipc\_server, 3132  
l4/sys/cxx/ipc\_server\_loop, 3136  
l4/sys/cxx/ipc\_string, 3139  
l4/sys/cxx/ipc\_types, 3141, 3142  
l4/sys/cxx/ipc\_varg, 3149  
l4/sys/cxx/smart\_capability\_1x, 3155, 3156  
l4/sys/cxx/types, 3158, 3159  
l4/sys/debugger, 3162, 3163  
l4/sys/debugger.h, 3163, 3165  
l4/sys/err.h, 3168, 3170  
l4/sys/exception, 3170, 3172  
l4/sys/factory, 3172, 3174  
l4/sys/factory.h, 3176, 3178  
l4/sys/icu, 3182, 3184  
l4/sys/icu.h, 3184, 3187  
l4/sys/iommu, 3190  
l4/sys/ipc.h, 3192, 3195  
l4/sys/ipc\_gate, 3200, 3201  
l4/sys/ipc\_gate.h, 3201, 3204  
l4/sys/irq, 3205, 3206  
l4/sys/irq.h, 2672, 2673  
l4/sys/kdebug.h, 3208, 3224  
l4/sys/kdump.h, 3227, 3228  
l4/sys/kernel\_object.h, 3228, 3229  
l4/sys/kip, 3230, 3231  
l4/sys/kip.h, 3362, 3365  
l4/sys/kobject, 3233  
l4/sys/ktrace.h, 3233, 3235  
l4/sys/l4int.h, 3237, 3238

l4/sys/memdesc.h, 3240, 3241  
 l4/sys/meta, 3243, 3245  
 l4/sys/obj\_info.h, 3245, 3247  
 l4/sys/pager, 3249, 3251  
 l4/sys/platform\_control, 3251, 3253  
 l4/sys/platform\_control.h, 3254, 3255  
 l4/sys/rcv\_endpoint, 3257, 3259  
 l4/sys/rcv\_endpoint.h, 3259, 3261  
 l4/sys/scheduler, 3262, 3264  
 l4/sys/scheduler.h, 2805, 2808  
 l4/sys/semaphore, 3264, 3266  
 l4/sys/semaphore.h, 3266, 3268  
 l4/sys/smart\_capability, 3268, 3270  
 l4/sys/snd\_destination, 3272, 3274  
 l4/sys/snd\_destination.h, 3274, 3275  
 l4/sys/task, 3275, 3277  
 l4/sys/task.h, 3278, 3280  
 l4/sys/thread, 3284, 3286  
 l4/sys/thread.h, 3390, 3393  
 l4/sys/thread\_group, 3288  
 l4/sys/thread\_group.h, 3289, 3290  
 l4/sys/typeinfo\_svr, 3291, 3292  
 l4/sys/types.h, 2812, 2815  
 l4/sys/utcb.h, 3300, 3302  
 l4/sys/vcon, 3307, 3309  
 l4/sys/vcon.h, 3309, 3312  
 l4/sys/vcpu, 3425, 3428  
 l4/sys/vcpu\_context, 3315, 3316  
 l4/sys/vcpu\_context.h, 3316  
 l4/sys/vm, 3317, 3318  
 l4/util/assert.h, 3321, 3322  
 l4/util/atomic.h, 3326, 3329  
 l4/util/backtrace.h, 3333, 3335  
 l4/util/base64.h, 3335, 3337  
 l4/util/bitops.h, 3337, 3339  
 l4/util/elf.h, 3342, 3348  
 l4/util/getopt.h, 3359, 3360  
 l4/util/keymap.h, 3361, 3362  
 l4/util/kip.h, 3367  
 l4/util/kprintf.h, 3368, 3369  
 l4/util/l4\_macros.h, 2530, 2531  
 l4/util/l4mod.h, 3369, 3370  
 l4/util/list\_alloc.h, 3371, 3373  
 l4/util/lock.h, 3374, 3375  
 l4/util/mb\_info.h, 3375, 3379  
 l4/util/parse\_cmd.h, 3383, 3384  
 l4/util/printf\_helpers.h, 3385  
 l4/util/rand.h, 3385, 3386  
 l4/util/splitlog2.h, 3386, 3387  
 l4/util/thread.h, 3400, 3402  
 l4/util/util.h, 3402  
 l4/vbus/vbus, 3403  
 l4/vbus/vbus.h, 3405, 3407  
 l4/vbus/vbus\_generic, 3408  
 l4/vbus/vbus\_gpio, 3409  
 l4/vbus/vbus\_gpio-ops.h, 3410  
 l4/vbus/vbus\_gpio.h, 3410  
 l4/vbus/vbus\_i2c.h, 3411  
 l4/vbus/vbus\_inhibitor.h, 3412  
 l4/vbus/vbus\_interfaces.h, 3412, 3415  
 l4/vbus/vbus\_mcspi.h, 3415  
 l4/vbus/vbus\_pci, 3416  
 l4/vbus/vbus\_pci-ops.h, 3417  
 l4/vbus/vbus\_pci.h, 3417  
 l4/vbus/vbus\_pm-ops.h, 3418  
 l4/vbus/vbus\_pm.h, 3418  
 l4/vbus/vbus\_types.h, 3418, 3421  
 l4/vbus/vdevice-ops.h, 3422  
 l4/vcpu/vcpu, 3423  
 l4/vcpu/vcpu.h, 3429, 3430  
 L4::Alloc\_list, 1028  
 L4::Arm\_smccc, 1029  
     call, 1029  
 L4::Base\_exception, 1030  
 L4::Basic\_registry, 1033  
     dispatch, 1034  
     find, 1034  
 L4::Bounds\_error, 1035  
 L4::Cap< T >, 1038  
     Cap, 1042  
     check\_castable\_from, 1043  
     check\_convertible\_from, 1043  
     copy, 1043  
     move, 1044  
 L4::Cap\_base, 1044  
     cap, 1049  
     Cap\_base, 1047, 1048  
     Cap\_type, 1046  
     copy, 1050  
     fpage, 1052  
     Invalid, 1047  
     is\_valid, 1053  
     move, 1054  
     No\_init, 1047  
     No\_init\_type, 1047  
     snd\_base, 1055  
     validate, 1056, 1057  
 L4::Com\_error, 1058  
     Com\_error, 1060  
 L4::Debugger, 1061  
     add\_image\_info, 1064  
     get\_object\_name, 1064  
     global\_id, 1065  
     kobj\_to\_id, 1066  
     query\_log\_name, 1066  
     query\_log\_typeid, 1067  
     set\_object\_name, 1068  
     switch\_log, 1069  
 L4::Element\_already\_exists, 1070  
 L4::Element\_not\_found, 1072  
 L4::Epiface, 1075  
     dispatch, 1077  
     get\_buffer\_demand, 1078  
     obj\_cap, 1078  
     server\_iface, 1078  
     set\_server, 1079



- L4::Epiface\_t< Derived, IFACE, BASE, bool >, 1080
  - dispatch, 1084
- L4::Epiface\_t0< RPC\_IFACE, BASE >, 1084
  - obj\_cap, 1087
- L4::Exception, 1087
  - exception, 1088
- L4::Exception\_tracer, 1089
- L4::Factory, 1090
  - create, 1094, 1095
  - create\_factory, 1096
  - create\_gate, 1097
  - create\_task, 1098
  - create\_thread\_group, 1100
- L4::Factory::Lstr, 1101
  - Lstr, 1102
- L4::Factory::Nil, 1103
- L4::Factory::S, 1103
  - ~S, 1105
  - operator l4\_msgtag\_t, 1106
  - operator<<, 1106, 1107
  - put, 1107–1109
  - S, 1105
- L4::lcu, 1109
  - bind, 1113
  - info, 1114
  - mask, 1115
  - msi\_info, 1116
  - set\_mode, 1117
  - unbind, 1118
- L4::lcu::Info, 1119
- L4::Invalid\_capability, 1120
  - cap, 1123
  - Invalid\_capability, 1123
- L4::io\_pager, 1124
  - io\_page\_fault, 1126
- L4::lommu, 1127
  - bind, 1130
  - unbind, 1130
- L4::IOModifier, 1132
- L4::lpc, 758
  - buf\_cp\_in, 760
  - buf\_cp\_out, 761
  - buf\_in, 761
  - make\_cap, 762
  - make\_cap\_full, 762
  - make\_cap\_rw, 763
  - make\_cap\_rws, 764
  - msg\_ptr, 765
  - read, 765
  - str\_cp\_in, 765
- L4::lpc::Array< ELEM\_TYPE, LEN\_TYPE >, 1132
- L4::lpc::Array\_in\_buf< ELEM\_TYPE, LEN\_TYPE, MAX >, 1135
- L4::lpc::Array\_ref< ELEM\_TYPE, LEN\_TYPE >, 1137
- L4::lpc::As\_value< T >, 1138
- L4::lpc::Call, 1139
- L4::lpc::Call\_t< RIGHTS >, 1140
- L4::lpc::Call\_zero\_send\_timeout, 1141
- L4::lpc::Cap< T >, 1143
  - Cap, 1145
  - Cap\_mask, 1145
  - from\_ci, 1146
  - Rights\_mask, 1145
- L4::lpc::Gen\_fpage, 1146
  - lo, 1149
  - Memory, 1149
  - Obj, 1149
  - Special, 1149
  - Type, 1148
- L4::lpc::In\_out< T >, 1149
- L4::lpc::lostream, 1150
  - call, 1155
  - get, 1156, 1157
  - lostream, 1154
  - put, 1157, 1158
  - reply\_and\_wait, 1158, 1159
  - reset, 1160
- L4::lpc::lstream, 1161
  - get, 1164, 1165
  - lstream, 1163
  - receive, 1167
  - reset, 1168
  - skip, 1168
  - tag, 1170
  - wait, 1171, 1172
- L4::lpc::Msg, 766
  - align\_to, 768
  - Br\_bytes, 768
  - check\_size, 769, 770
  - Item\_bytes, 768
  - Item\_words, 768
  - Mr\_bytes, 768
  - Mr\_words, 768
  - msg\_add, 771
  - msg\_get, 772
  - Word\_bytes, 768
- L4::lpc::Msg::Cls\_buffer, 1173
- L4::lpc::Msg::Cls\_data, 1174
- L4::lpc::Msg::Cls\_item, 1175
- L4::lpc::Msg::Dir\_in, 1176
- L4::lpc::Msg::Dir\_out, 1177
- L4::lpc::Msg::Do\_in\_data, 1178
- L4::lpc::Msg::Do\_in\_items, 1179
- L4::lpc::Msg::Do\_out\_data, 1180
- L4::lpc::Msg::Do\_out\_items, 1181
- L4::lpc::Msg::Do\_rcv\_buffers, 1182
- L4::lpc::Msg::Elem< Array< A, LEN > >, 1185
- L4::lpc::Msg::Elem< Array< A, LEN > & >, 1184
- L4::lpc::Msg::Elem< Array\_ref< A, LEN > & >, 1186
- L4::lpc::Msg::False, 1187
- L4::lpc::Msg::Is\_valid\_rpc\_type< T >, 1189
- L4::lpc::Msg::Svr\_arg\_pack< IPC\_TYPE >, 1191
- L4::lpc::Msg::Svr\_val\_ops< MTYPE, DIR, CLASS >, 1191
- L4::lpc::Msg::True, 1193
- L4::lpc::Msg\_ptr< T >, 1196

- Msg\_ptr, 1196
- L4::lpc::Opt< T >, 1197
- L4::lpc::Ostream, 1199
  - put, 1202, 1203
  - send, 1203
  - tag, 1204
- L4::lpc::Out< T >, 1205
- L4::lpc::Rcv\_fpage, 1206
  - io, 1209
  - mem, 1209
  - obj, 1210
  - Rcv\_fpage, 1208
  - rcv\_task, 1211
- L4::lpc::Ret\_array< T >, 1212
- L4::lpc::Send\_only, 1213
- L4::lpc::Small\_buf, 1214
  - Small\_buf, 1214, 1215
- L4::lpc::Snd\_fpage, 1215
  - Buffered, 1219
  - Cached, 1219
  - Cacheopt, 1219
  - cap\_received, 1222
  - Compound, 1220
  - Continue, 1219
  - Grant, 1220
  - id\_received, 1222
  - io, 1222
  - is\_compound, 1223
  - Last, 1220
  - local\_id\_received, 1223
  - Map, 1220
  - Map\_type, 1220
  - mem, 1223
  - More, 1220
  - None, 1219
  - obj, 1224
  - Single, 1220
  - Snd\_fpage, 1220, 1221
  - Uncached, 1219
- L4::lpc::Str\_cp\_in< T >, 1226
  - Str\_cp\_in, 1226
- L4::lpc::Varg, 1227
  - data, 1228
  - get\_value, 1228
  - is\_nil, 1229
  - is\_of, 1230
  - is\_of\_int, 1231
  - length, 1231
  - tag, 1232
  - type, 1232
  - value, 1232
- L4::lpc::Varg\_list< MAX >, 1233
- L4::lpc::Varg\_list\_ref, 1235
  - Varg\_list\_ref, 1237
- L4::lpc::Varg\_list\_ref::Iterator, 1238
- L4::lpc\_gate, 1239
  - get\_infos, 1244
- L4::lpc\_svr, 773
  - L4::lpc\_svr::Br\_manager\_no\_buffers, 1245
    - alloc\_buffer\_demand, 1248
  - L4::lpc\_svr::Compound\_reply, 1249
  - L4::lpc\_svr::Dbg\_dispatch< R, Exc, Printer >, 1251
  - L4::lpc\_svr::Default\_loop\_hooks, 1253
  - L4::lpc\_svr::Default\_setup\_wait, 1255
  - L4::lpc\_svr::Default\_timeout, 1256
  - L4::lpc\_svr::Direct\_dispatch< R >, 1258
  - L4::lpc\_svr::Direct\_dispatch< R \* >, 1260
  - L4::lpc\_svr::Exc\_dispatch< R, Exc >, 1262
  - L4::lpc\_svr::Ignore\_errors, 1263
  - L4::lpc\_svr::Server\_iface, 1265
    - add\_timeout, 1268
    - alloc\_buffer\_demand, 1268
    - get\_rcv\_cap, 1268
    - rcv\_cap, 1270, 1271
    - realloc\_rcv\_cap, 1272
    - remove\_timeout, 1272
  - L4::lpc\_svr::Timeout, 1273
    - expired, 1276
    - timeout, 1276
  - L4::lpc\_svr::Timeout\_queue, 1277
    - add, 1278
    - handle\_expired\_timeouts, 1278
    - next\_timeout, 1278
    - remove, 1279
    - timeout\_expired, 1279
  - L4::lpc\_svr::Timeout\_queue\_hooks< HOOKS, BR\_MAN >, 1280
    - add\_timeout, 1284
    - remove\_timeout, 1284
  - L4::lrq, 1286
    - bind\_vcpu, 1291
    - detach, 1292
    - receive, 1293
    - unmask, 1294
    - wait, 1295
  - L4::lrq\_eoi, 1296
    - unmask, 1297
  - L4::lrq\_handler\_object, 1298
  - L4::lrqep\_t< Derived, BASE, bool >, 1302
    - dispatch, 1305
    - obj\_cap, 1305
  - L4::Kip::Mem\_desc, 1306
    - all, 1311, 1312
    - Arch, 1309
    - Arch\_acpi\_nvs, 1309
    - Arch\_acpi\_tables, 1308
    - Arch\_sub\_type\_common, 1308
    - Bootloader, 1309
    - Conventional, 1309
    - count, 1312, 1313
    - Dedicated, 1309
    - end, 1313
    - first, 1314
    - Info, 1309
    - Info\_acpi\_rsd, 1309
    - Info\_sub\_type, 1309



- is\_virtual, [1315](#)
- Mem\_desc, [1310](#)
- Mem\_type, [1309](#)
- Reserved, [1309](#)
- Reserved\_heap, [1309](#)
- Reserved\_kernel, [1309](#)
- Reserved\_mmio, [1309](#)
- set, [1315](#)
- Shared, [1309](#)
- size, [1316](#)
- start, [1316](#)
- sub\_type, [1317](#)
- type, [1317](#)
- Undefined, [1309](#)
- L4::Kobject, [1318](#)
  - cap, [1319](#)
  - dec\_refcnt, [1320](#)
- L4::Kobject\_2t< Derived, Base1, Base2, PROTO, S\_DEMAND >, [1322](#)
  - \_\_lface, [1324](#)
  - \_\_lface\_list, [1324](#)
  - \_\_check\_protocols\_\_, [1325](#)
  - c, [1325](#)
  - Class, [1325](#)
- L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO, S\_DEMAND >, [1326](#)
  - \_\_lface, [1328](#)
  - \_\_lface\_list, [1328](#)
  - \_\_check\_protocols\_\_, [1329](#)
  - c, [1329](#)
  - Class, [1328](#)
- L4::Kobject\_demand< T >, [1329](#)
- L4::Kobject\_t< Derived, Base, PROTO, S\_DEMAND >, [1330](#)
- L4::Kobject\_typeid< T >, [1332](#)
  - Demand, [1333](#)
  - demand, [1333](#)
  - id, [1333](#)
  - proto\_dispatch, [1334](#)
- L4::Kobject\_typeid< void >, [1335](#)
  - demand, [1336](#)
  - id, [1336](#)
  - proto\_dispatch, [1337](#)
- L4::Kobject\_x< Derived, ARGS >, [1338](#)
- L4::Lock\_guard, [1339](#)
  - ~Lock\_guard, [1340](#)
  - Lock\_guard, [1340](#)
  - operator=, [1341](#)
  - status, [1341](#)
- L4::Meta, [1342](#)
  - interface, [1344](#)
  - num\_interfaces, [1345](#)
  - supports, [1346](#)
- L4::Out\_of\_memory, [1347](#)
- L4::Pager, [1351](#)
  - page\_fault, [1353](#)
- L4::Platform\_control, [1354](#)
  - cpu\_allow\_shutdown, [1357](#)
  - cpu\_disable, [1358](#)
  - cpu\_enable, [1359](#)
  - system\_shutdown, [1360](#)
  - system\_suspend, [1361](#)
- L4::Poll\_timeout\_counter, [1362](#)
  - Poll\_timeout\_counter, [1363](#)
  - set, [1363](#)
  - timed\_out, [1364](#)
- L4::Poll\_timeout\_kipclock, [1364](#)
  - Poll\_timeout\_kipclock, [1365](#)
  - set, [1366](#)
  - test, [1366](#)
  - timed\_out, [1367](#)
- L4::Proto\_t< P >, [1368](#)
- L4::Rcv\_endpoint, [1368](#)
  - bind\_snd\_destination, [1371](#)
  - bind\_thread, [1372](#)
- L4::Registry\_iface, [1374](#)
  - register\_irq\_obj, [1376](#)
  - register\_obj, [1376](#), [1377](#)
  - unregister\_obj, [1378](#)
- L4::Runtime\_error, [1378](#)
  - err\_no, [1382](#)
  - extra\_str, [1382](#)
  - Runtime\_error, [1381](#)
- L4::Scheduler, [1383](#)
  - idle\_time, [1388](#)
  - info, [1389](#)
  - is\_online, [1390](#)
  - run\_thread, [1391](#)
- L4::Semaphore, [1392](#)
  - down, [1396](#)
  - up, [1397](#)
- L4::Server< LOOP\_HOOKS >, [1398](#)
  - internal\_loop, [1402](#)
  - loop, [1403](#)
  - loop\_dbg, [1403](#)
  - Server, [1402](#)
- L4::Server\_object, [1404](#)
  - dispatch, [1407](#), [1408](#)
- L4::Server\_object\_t< IFACE, BASE >, [1409](#)
  - dispatch\_meta\_request, [1413](#)
  - get\_buffer\_demand, [1413](#)
  - proto\_dispatch, [1413](#)
- L4::Server\_object\_x< Derived, IFACE, BASE >, [1415](#)
- L4::Smart\_cap< T, SMART >, [1418](#)
  - Smart\_cap, [1422](#)
- L4::String, [1422](#)
- L4::Task, [1423](#)
  - add\_ku\_mem, [1427](#)
  - cap\_equal, [1428](#)
  - cap\_valid, [1429](#)
  - delete\_obj, [1430](#)
  - map, [1431](#)
  - release\_cap, [1432](#)
  - unmap, [1433](#)
  - unmap\_batch, [1434](#)
- L4::Thread, [1435](#)

- control, [1439](#)
- ex\_regs, [1440](#), [1441](#)
- modify\_senders, [1442](#)
- register\_del\_irq, [1443](#)
- register\_doorbell\_irq, [1444](#)
- stats\_time, [1445](#)
- switch\_to, [1446](#)
- vcpu\_control, [1446](#)
- vcpu\_control\_ext, [1447](#)
- vcpu\_resume\_commit, [1448](#)
- vcpu\_resume\_start, [1450](#)
- L4::Thread::Attr, [1451](#)
  - alien, [1452](#)
  - Attr, [1452](#)
  - bind, [1452](#)
  - exc\_handler, [1453](#)
  - pager, [1454](#)
- L4::Thread::Modify\_senders, [1455](#)
  - add, [1456](#)
- L4::Thread\_group, [1458](#)
  - add, [1460](#)
  - remove, [1461](#)
- L4::Triggerable, [1462](#)
  - trigger, [1465](#)
- L4::Type\_info, [1466](#)
- L4::Type\_info::Demand, [1467](#)
  - Demand, [1470](#)
  - no\_demand, [1470](#)
- L4::Type\_info::Demand\_t< CAPS, FLAGS, MEM, PORTS >, [1471](#)
  - Caps, [1474](#)
  - Flags, [1474](#)
  - Mem, [1474](#)
  - Ports, [1474](#)
- L4::Type\_info::Demand\_union\_t< D1, D2 >, [1474](#)
- L4::Typeid, [774](#)
- L4::Typeid::Detail::\_Rpc< OPCODE, O, Default\_op< R > >::Rpc< Y >, [1479](#)
- L4::Typeid::Detail::\_Rpc< OPCODE, O, R, X... >, [1479](#)
- L4::Typeid::Detail::\_Rpc< OPCODE, O, R, X... >::Rpc< Y >, [1482](#)
- L4::Typeid::Detail::\_Rpc< OPCODE, O, X >, [1478](#)
- L4::Typeid::Detail::Rpc\_end, [1483](#)
- L4::Typeid::P\_dispatch< LIST >, [1483](#)
- L4::Typeid::Raw\_ipc< CLASS >, [1484](#)
- L4::Typeid::Rpc\_nocode< OPERATION >, [1485](#)
- L4::Typeid::Rpc< RPCS >, [1488](#)
- L4::Typeid::Rpc\_code< OPCODE\_TYPE >, [1489](#)
- L4::Typeid::Rpc\_code< OPCODE\_TYPE >::F< RPCS >, [1490](#)
- L4::Typeid::Rpc\_sys< ARG >, [1493](#)
- L4::Types, [774](#)
- L4::Types::Bool< V >, [1495](#)
- L4::Types::False, [1497](#)
- L4::Types::Flags< BITS\_ENUM, UNDERLYING >, [1499](#)
  - clear, [1503](#)
  - control, [1439](#)
  - ex\_regs, [1440](#), [1441](#)
  - modify\_senders, [1442](#)
  - register\_del\_irq, [1443](#)
  - register\_doorbell\_irq, [1444](#)
  - stats\_time, [1445](#)
  - switch\_to, [1446](#)
  - vcpu\_control, [1446](#)
  - vcpu\_control\_ext, [1447](#)
  - vcpu\_resume\_commit, [1448](#)
  - vcpu\_resume\_start, [1450](#)
- L4::Types::Flags\_ops\_t< DT >, [1503](#)
- L4::Types::Flags\_t< DT, T >, [1506](#)
- L4::Types::Int\_for\_size< SIZE, bool >, [1508](#)
- L4::Types::Int\_for\_type< T >, [1509](#)
- L4::Types::Same< A, B >, [1510](#)
- L4::Types::True, [1513](#)
- L4::Uart, [1516](#)
  - change\_mode, [1518](#)
  - char\_avail, [1518](#)
  - enable\_rx\_irq, [1518](#)
  - generic\_write, [1519](#)
  - get\_char, [1519](#)
  - mode, [1520](#)
  - rate, [1520](#)
  - shutdown, [1520](#)
  - startup, [1520](#)
  - write, [1521](#)
- L4::Uart\_apb, [1521](#)
  - change\_mode, [1524](#)
  - char\_avail, [1525](#)
  - enable\_rx\_irq, [1525](#)
  - get\_char, [1525](#)
  - shutdown, [1526](#)
  - startup, [1526](#)
  - write, [1526](#)
- L4::Unknown\_error, [1527](#)
- L4::Vcon, [1529](#)
  - get\_attr, [1534](#)
  - read, [1535](#)
  - read\_with\_flags, [1536](#)
  - send, [1537](#)
  - set\_attr, [1538](#)
  - write, [1538](#)
- L4::Vm, [1539](#)
  - vgicc\_map, [1543](#)
- l4\_addr\_consts\_t
  - Memory related, [208](#)
- l4\_align\_stack\_for\_direct\_fncall
  - Basic Macros, [164](#)
- L4\_AMD64\_SEGMENT\_FS
  - segment.h, [2465](#)
- L4\_AMD64\_SEGMENT\_GS
  - segment.h, [2465](#)
- l4\_arm\_smccc\_call
  - arm\_smccc.h, [3070](#)
- l4\_assert
  - assert.h, [3320](#)
- L4\_BASE\_ARM\_SMCCC\_CAP
  - Capabilities, [452](#)
- L4\_BASE\_CAPS\_LAST
  - Capabilities, [452](#)
- L4\_BASE\_DEBUGGER\_CAP
  - Capabilities, [452](#)
- L4\_BASE\_FACTORY\_CAP

- Capabilities, [452](#)
- L4\_BASE\_ICU\_CAP
  - Capabilities, [452](#)
- L4\_BASE\_IOMMU\_CAP
  - Capabilities, [452](#)
- L4\_BASE\_LOG\_CAP
  - Capabilities, [452](#)
- L4\_BASE\_PAGER\_CAP
  - Capabilities, [452](#)
- L4\_BASE\_SCHEDULER\_CAP
  - Capabilities, [452](#)
- L4\_BASE\_TASK\_CAP
  - Capabilities, [452](#)
- L4\_BASE\_THREAD\_CAP
  - Capabilities, [452](#)
- L4\_BDR\_IO\_SHIFT
  - Buffer Registers (BRs), [280](#)
- L4\_BDR\_MEM\_SHIFT
  - Buffer Registers (BRs), [280](#)
- L4\_BDR\_OBJ\_SHIFT
  - Buffer Registers (BRs), [280](#)
- l4\_buf\_regs\_t, [1544](#)
- l4\_buffer\_desc\_consts\_t
  - Buffer Registers (BRs), [279](#)
- l4\_busy\_wait\_ns
  - Timestamp Counter, [661](#)
- l4\_busy\_wait\_us
  - Timestamp Counter, [662](#)
- l4\_bytes\_to\_mwords
  - Memory related, [209](#)
- l4\_cache\_clean\_data
  - Cache Consistency, [201](#)
- l4\_cache\_coherent
  - Cache Consistency, [201](#)
- l4\_cache\_dma\_coherent
  - Cache Consistency, [202](#)
- l4\_cache\_flush\_data
  - Cache Consistency, [202](#)
- l4\_cache\_inv\_data
  - Cache Consistency, [203](#)
- l4\_calibrate\_tsc
  - Timestamp Counter, [662](#)
- l4\_cap\_consts\_t
  - Capabilities, [451](#)
- L4\_CAP\_FPAGE\_D
  - Flexpages, [185](#)
- L4\_CAP\_FPAGE\_R
  - Flexpages, [185](#)
- L4\_cap\_fpage\_rights
  - Flexpages, [185](#)
- L4\_CAP\_FPAGE\_RO
  - Flexpages, [185](#)
- L4\_CAP\_FPAGE\_RS
  - Flexpages, [186](#)
- L4\_CAP\_FPAGE\_RSD
  - Flexpages, [186](#)
- L4\_CAP\_FPAGE\_RW
  - Flexpages, [185](#)
- L4\_CAP\_FPAGE\_RWD
  - Flexpages, [186](#)
- L4\_CAP\_FPAGE\_RWS
  - Flexpages, [186](#)
- L4\_CAP\_FPAGE\_RWSD
  - Flexpages, [186](#)
- L4\_CAP\_FPAGE\_S
  - Flexpages, [185](#)
- L4\_CAP\_FPAGE\_W
  - Flexpages, [185](#)
- l4\_cap\_idx\_t
  - Capabilities, [451](#)
- L4\_CAP\_MASK
  - Capabilities, [452](#)
- L4\_CAP\_OFFSET
  - Capabilities, [452](#)
- L4\_CAP\_SHIFT
  - Capabilities, [452](#)
- L4\_CAP\_SIZE
  - Capabilities, [452](#)
- l4\_capability\_equal
  - Capabilities, [453](#)
- l4\_capability\_next
  - types.h, [2814](#)
- l4\_debugger\_add\_image\_info
  - Kernel Debugger, [170](#)
- l4\_debugger\_get\_object\_name
  - Kernel Debugger, [171](#)
- l4\_debugger\_global\_id
  - Kernel Debugger, [172](#)
- l4\_debugger\_kobj\_to\_id
  - Kernel Debugger, [172](#)
- l4\_debugger\_query\_log\_name
  - Kernel Debugger, [174](#)
- l4\_debugger\_query\_log\_typeid
  - Kernel Debugger, [175](#)
- l4\_debugger\_query\_obj\_infos
  - obj\_info.h, [3246](#)
- l4\_debugger\_set\_object\_name
  - Kernel Debugger, [176](#)
- l4\_debugger\_switch\_log
  - Kernel Debugger, [176](#)
- l4\_default\_caps\_t
  - Capabilities, [452](#)
- L4\_DISABLE\_COPY
  - capability, [3084](#)
- L4\_E2BIG
  - Error codes, [214](#)
- L4\_EACCESS
  - Error codes, [214](#)
- L4\_EADDRNOTAVAIL
  - Error codes, [214](#)
- L4\_EAGAIN
  - Error codes, [214](#)
- L4\_EBADPROTO
  - Error codes, [214](#)
- L4\_EBUSY
  - Error codes, [214](#)

- L4\_EEXIST
  - Error codes, [214](#)
- L4\_EFAULT
  - Error codes, [214](#)
- L4\_EINVAL
  - Error codes, [214](#)
- L4\_EIO
  - Error codes, [214](#)
- L4\_EIPC\_HI
  - Error codes, [214](#)
- L4\_EIPC\_LO
  - Error codes, [214](#)
- L4\_MSGMISSARG
  - Error codes, [214](#)
- L4\_MSGTOO LONG
  - Error codes, [214](#)
- L4\_MSGTOO SHORT
  - Error codes, [214](#)
- L4\_ENAMETOO LONG
  - Error codes, [214](#)
- L4\_ENODEV
  - Error codes, [214](#)
- L4\_ENOENT
  - Error codes, [214](#)
- L4\_ENOMEM
  - Error codes, [214](#)
- L4\_ENOREPLY
  - Error codes, [214](#)
- L4\_ENOSPC
  - Error codes, [214](#)
- L4\_ENOSYS
  - Error codes, [214](#)
- L4\_ENOTDIR
  - Error codes, [214](#)
- L4\_ENXIO
  - Error codes, [214](#)
- L4\_EOK
  - Error codes, [214](#)
- L4\_EPERM
  - Error codes, [214](#)
- L4\_ERANGE
  - Error codes, [214](#)
- L4\_ERRNOMAX
  - Error codes, [214](#)
- l4\_error
  - Error Handling, [253](#)
- l4\_error\_code\_t
  - Error codes, [214](#)
- l4\_exc\_regs\_t, [1545](#)
  - flags, [1547](#)
  - ss, [1547](#)
- L4\_EXPORT
  - Basic Macros, [162](#)
- l4\_factory\_create
  - Factory, [295](#)
- l4\_factory\_create\_factory
  - Factory, [295](#)
- l4\_factory\_create\_gate
  - Factory, [296](#)
- l4\_factory\_create\_irq
  - Factory, [297](#)
- l4\_factory\_create\_task
  - Factory, [298](#)
- l4\_factory\_create\_thread
  - Factory, [299](#)
- l4\_factory\_create\_thread\_group
  - Factory, [300](#)
- l4\_factory\_create\_vcpu\_context
  - Factory, [301](#)
- l4\_factory\_create\_vm
  - Factory, [302](#)
- L4\_FP\_ALL\_SPACES
  - Task, [374](#)
- L4\_FP\_DELETE\_OBJ
  - Task, [374](#)
- L4\_FP\_OTHER\_SPACES
  - Task, [374](#)
- l4\_fpage
  - Flexpages, [188](#)
- L4\_FPAGE\_ADDR\_BITS
  - Flexpages, [186](#)
- L4\_FPAGE\_ADDR\_SHIFT
  - Flexpages, [186](#)
- l4\_fpage\_all
  - Flexpages, [188](#)
- L4\_FPAGE\_BUFFERABLE
  - Message Items, [239](#)
- L4\_FPAGE\_C\_IPCGATE\_SVR
  - Message Items, [241](#)
- L4\_FPAGE\_C\_NO\_REF\_CNT
  - Message Items, [240](#)
- L4\_FPAGE\_C\_OBJ\_RIGHT1
  - Message Items, [240](#)
- L4\_FPAGE\_C\_OBJ\_RIGHT2
  - Message Items, [241](#)
- L4\_FPAGE\_C\_OBJ\_RIGHT3
  - Message Items, [241](#)
- L4\_FPAGE\_C\_OBJ\_RIGHTS
  - Message Items, [241](#)
- L4\_FPAGE\_C\_REF\_CNT
  - Message Items, [240](#)
- L4\_FPAGE\_CACHE\_OPT
  - Message Items, [239](#)
- l4\_fpage\_cacheability\_opt\_t
  - Message Items, [238](#)
- L4\_FPAGE\_CACHEABLE
  - Message Items, [239](#)
- L4\_fpage\_consts
  - Flexpages, [186](#)
- l4\_fpage\_contains
  - Flexpages, [189](#)
- L4\_fpage\_control
  - Flexpages, [187](#)
- L4\_FPAGE\_CONTROL\_MASK
  - Flexpages, [187](#)
- L4\_FPAGE\_CONTROL\_OFFSET\_SHIFT

- Flexpages, [187](#)
- l4\_fpage\_invalid
  - Flexpages, [190](#)
- L4\_FPAGE\_IO
  - Flexpages, [188](#)
- l4\_fpage\_ioport
  - Flexpages, [190](#)
- l4\_fpage\_max\_order
  - Flexpages, [190](#)
- l4\_fpage\_memaddr
  - Flexpages, [191](#)
- L4\_FPAGE\_MEMORY
  - Flexpages, [188](#)
- L4\_FPAGE\_OBJ
  - Flexpages, [188](#)
- l4\_fpage\_obj
  - Flexpages, [192](#)
- l4\_fpage\_page
  - Flexpages, [192](#)
- L4\_fpage\_rights
  - Flexpages, [187](#)
- l4\_fpage\_rights
  - Flexpages, [193](#)
- L4\_FPAGE\_RIGHTS\_ALL
  - Flexpages, [187](#)
- L4\_FPAGE\_RIGHTS\_BITS
  - Flexpages, [186](#)
- L4\_FPAGE\_RIGHTS\_MASK
  - Flexpages, [186](#)
- L4\_FPAGE\_RIGHTS\_SHIFT
  - Flexpages, [186](#)
- L4\_FPAGE\_RO
  - Flexpages, [187](#)
- L4\_FPAGE\_RW
  - Flexpages, [187](#)
- L4\_FPAGE\_RWX
  - Flexpages, [187](#)
- L4\_FPAGE\_RX
  - Flexpages, [187](#)
- l4\_fpage\_set\_rights
  - Flexpages, [193](#)
- l4\_fpage\_size
  - Flexpages, [194](#)
- L4\_FPAGE\_SIZE\_BITS
  - Flexpages, [186](#)
- L4\_FPAGE\_SIZE\_SHIFT
  - Flexpages, [186](#)
- L4\_FPAGE\_SPECIAL
  - Flexpages, [188](#)
- l4\_fpage\_t, [1548](#)
- L4\_fpage\_type
  - Flexpages, [187](#)
- l4\_fpage\_type
  - Flexpages, [195](#)
- L4\_FPAGE\_TYPE\_BITS
  - Flexpages, [186](#)
- L4\_FPAGE\_TYPE\_SHIFT
  - Flexpages, [186](#)
- L4\_FPAGE\_UNCACHEABLE
  - Message Items, [239](#)
- L4\_FPAGE\_W
  - Flexpages, [187](#)
- L4\_FPAGE\_X
  - Flexpages, [187](#)
- l4\_get\_hz
  - Timestamp Counter, [663](#)
- L4\_HIDDEN
  - Basic Macros, [162](#)
- l4\_icu\_bind
  - Interrupt controller, [328](#)
- l4\_icu\_bind\_u
  - Interrupt controller, [329](#)
- L4\_ICU\_FLAG\_MSI
  - Interrupt controller, [328](#)
- L4\_icu\_flags
  - Interrupt controller, [327](#)
- l4\_icu\_info
  - Interrupt controller, [330](#)
- l4\_icu\_info\_t, [1549](#)
  - features, [1550](#)
  - Interrupt controller, [327](#)
- l4\_icu\_info\_u
  - Interrupt controller, [331](#)
- l4\_icu\_mask
  - Interrupt controller, [332](#)
- l4\_icu\_mask\_u
  - Interrupt controller, [333](#)
- l4\_icu\_msi\_info
  - Interrupt controller, [334](#)
- l4\_icu\_msi\_info\_t, [1550](#)
- l4\_icu\_msi\_info\_u
  - Interrupt controller, [335](#)
- L4\_ICU\_OP\_BIND
  - L4 IPC Opcodes, [485](#)
- L4\_ICU\_OP\_INFO
  - L4 IPC Opcodes, [485](#)
- L4\_ICU\_OP\_MASK
  - L4 IPC Opcodes, [486](#)
- L4\_ICU\_OP\_MSI\_INFO
  - L4 IPC Opcodes, [486](#)
- L4\_ICU\_OP\_SET\_MODE
  - L4 IPC Opcodes, [486](#)
- L4\_ICU\_OP\_UNBIND
  - L4 IPC Opcodes, [485](#)
- L4\_ICU\_OP\_UNMASK
  - L4 IPC Opcodes, [486](#)
- L4\_icu\_opcode
  - L4 IPC Opcodes, [485](#)
- l4\_icu\_set\_mode
  - Interrupt controller, [336](#)
- l4\_icu\_set\_mode\_u
  - Interrupt controller, [337](#)
- l4\_icu\_unbind
  - Interrupt controller, [338](#)
- l4\_icu\_unbind\_u
  - Interrupt controller, [339](#)

- I4\_icu\_unmask
  - Interrupt controller, [340](#)
- I4\_icu\_unmask\_u
  - Interrupt controller, [341](#)
- I4\_infinite\_loop
  - Basic Macros, [165](#)
- L4\_INLINE\_RPC
  - ipc\_iface, [3120](#)
- L4\_INLINE\_RPC\_NF
  - ipc\_iface, [3120](#)
- L4\_INLINE\_RPC\_NF\_OP
  - ipc\_iface, [3120](#)
- L4\_INLINE\_RPC\_OP
  - ipc\_iface, [3121](#)
- L4\_INVALID\_ADDR
  - Memory related, [209](#)
- L4\_INVALID\_CAP
  - Capabilities, [452](#)
- I4\_iofpage
  - Flexpages, [195](#)
- L4\_IOPORT\_MAX
  - Flexpages, [185](#)
- I4\_ipc
  - Object Invocation, [218](#)
- I4\_ipc\_call
  - Object Invocation, [220](#)
- L4\_IPC\_ENOT\_EXISTENT
  - Error Handling, [253](#)
- I4\_ipc\_error
  - Error Handling, [254](#)
- I4\_ipc\_error\_code
  - Error Handling, [255](#)
- L4\_IPC\_ERROR\_MASK
  - Error Handling, [252](#)
- L4\_IPC\_GATE\_BIND\_OP
  - L4 IPC Opcodes, [486](#)
- L4\_IPC\_GATE\_GET\_INFO\_OP
  - L4 IPC Opcodes, [486](#)
- I4\_ipc\_gate\_get\_infos
  - IPC-Gate API, [287](#)
- L4\_ipc\_gate\_ops
  - L4 IPC Opcodes, [486](#)
- I4\_ipc\_is\_rcv\_error
  - Error Handling, [256](#)
- I4\_ipc\_is\_snd\_error
  - Error Handling, [256](#)
- L4\_IPC\_REABORTED
  - Error Handling, [253](#)
- L4\_IPC\_RECANCELED
  - Error Handling, [253](#)
- I4\_ipc\_receive
  - Object Invocation, [223](#)
- L4\_IPC\_REMAPFAILED
  - Error Handling, [253](#)
- L4\_IPC\_REMSGCUT
  - Error Handling, [253](#)
- I4\_ipc\_reply\_and\_wait
  - Object Invocation, [225](#)
- L4\_IPC\_RERCVPFTO
  - Error Handling, [253](#)
- L4\_IPC\_RESNDPFTO
  - Error Handling, [253](#)
- L4\_IPC\_RETIMEOUT
  - Error Handling, [253](#)
- L4\_IPC\_SEABORTED
  - Error Handling, [253](#)
- L4\_IPC\_SECANCELED
  - Error Handling, [253](#)
- L4\_IPC\_SEMAPFAILED
  - Error Handling, [253](#)
- L4\_IPC\_SEMSGCUT
  - Error Handling, [253](#)
- I4\_ipc\_send
  - Object Invocation, [227](#)
- I4\_ipc\_send\_and\_wait
  - Object Invocation, [228](#)
- L4\_IPC\_SERCVPFTO
  - Error Handling, [253](#)
- L4\_IPC\_SESNDFPFTO
  - Error Handling, [253](#)
- L4\_IPC\_SETIMEOUT
  - Error Handling, [253](#)
- I4\_ipc\_sleep
  - Object Invocation, [229](#)
- I4\_ipc\_sleep\_ms
  - Object Invocation, [231](#)
- I4\_ipc\_sleep\_us
  - Object Invocation, [233](#)
- L4\_IPC\_SND\_ERR\_MASK
  - Error Handling, [253](#)
- I4\_ipc\_tcr\_error\_t
  - Error Handling, [252](#)
- I4\_ipc\_timeout
  - Timeouts, [245](#)
- L4\_IPC\_TIMEOUT\_0
  - Timeouts, [244](#)
- I4\_ipc\_to\_errno
  - ipc.h, [3194](#)
- I4\_ipc\_wait
  - Object Invocation, [234](#)
- I4\_irq\_bind\_vcpu
  - IRQs, [344](#)
- I4\_irq\_bind\_vcpu\_u
  - IRQs, [345](#)
- I4\_irq\_detach
  - IRQs, [347](#)
- I4\_irq\_detach\_u
  - IRQs, [348](#)
- L4\_IRQ\_F\_BOTH
  - IRQs, [344](#)
- L4\_IRQ\_F\_BOTH\_EDGE
  - IRQs, [344](#)
- L4\_IRQ\_F\_CLEAR\_WAKEUP
  - IRQs, [344](#)
- L4\_IRQ\_F\_EDGE
  - IRQs, [344](#)

- L4\_IRQ\_F\_LEVEL
  - IRQs, [344](#)
- L4\_IRQ\_F\_LEVEL\_HIGH
  - IRQs, [344](#)
- L4\_IRQ\_F\_LEVEL\_LOW
  - IRQs, [344](#)
- L4\_IRQ\_F\_MASK
  - IRQs, [344](#)
- L4\_IRQ\_F\_NEG
  - IRQs, [344](#)
- L4\_IRQ\_F\_NEG\_EDGE
  - IRQs, [344](#)
- L4\_IRQ\_F\_NONE
  - IRQs, [344](#)
- L4\_IRQ\_F\_POS
  - IRQs, [344](#)
- L4\_IRQ\_F\_POS\_EDGE
  - IRQs, [344](#)
- L4\_IRQ\_F\_SET\_MODE
  - IRQs, [344](#)
- L4\_IRQ\_F\_SET\_WAKEUP
  - IRQs, [344](#)
- L4\_irq\_mode
  - IRQs, [343](#)
- l4\_irq\_receive
  - IRQs, [349](#)
- l4\_irq\_receive\_u
  - IRQs, [350](#)
- l4\_irq\_trigger
  - IRQs, [351](#)
- l4\_irq\_trigger\_u
  - IRQs, [352](#)
- l4\_irq\_unmask
  - IRQs, [353](#)
- l4\_irq\_unmask\_u
  - IRQs, [354](#)
- l4\_irq\_wait
  - IRQs, [355](#)
- l4\_irq\_wait\_u
  - IRQs, [356](#)
- l4\_is\_fpage\_valid
  - Flexpages, [196](#)
- l4\_is\_fpage\_writable
  - Flexpages, [197](#)
- l4\_is\_invalid\_cap
  - Capabilities, [453](#)
- l4\_is\_valid\_cap
  - Capabilities, [453](#)
- L4\_ITEM\_CONT
  - Message Items, [239](#)
- L4\_ITEM\_MAP
  - Message Items, [239](#)
- l4\_kdebug\_ops\_t
  - kdebug.h, [3210](#)
- l4\_kernel\_info\_get\_mem\_desc\_end
  - Memory descriptors (C version), [448](#)
- l4\_kernel\_info\_get\_mem\_desc\_is\_virtual
  - Memory descriptors (C version), [448](#)
- l4\_kernel\_info\_get\_mem\_desc\_start
  - Memory descriptors (C version), [448](#)
- l4\_kernel\_info\_get\_mem\_desc\_subtype
  - Memory descriptors (C version), [448](#)
- l4\_kernel\_info\_get\_mem\_desc\_type
  - Memory descriptors (C version), [449](#)
- l4\_kernel\_info\_get\_num\_mem\_descs
  - Memory descriptors (C version), [449](#)
- l4\_kernel\_info\_mem\_desc\_t, [1551](#)
  - Memory descriptors (C version), [446](#)
- l4\_kernel\_info\_set\_mem\_desc
  - Memory descriptors (C version), [449](#)
- l4\_kernel\_info\_t, [1552](#)
  - Kernel Interface Page, [438](#)
- l4\_kernel\_info\_version\_offset
  - Kernel Interface Page, [439](#)
- l4\_kip
  - Kernel Interface Page, [440](#)
- l4\_kip\_clock
  - Kernel Interface Page, [440](#)
- l4\_kip\_clock\_lw
  - Kernel Interface Page, [441](#)
- l4\_kip\_clock\_ns
  - Kernel Interface Page, [442](#)
- l4\_kip\_for\_each\_feature
  - kip.h, [3364](#)
- l4\_kip\_kernel\_has\_feature
  - kip.h, [3364](#)
- L4\_KIP\_OFFS\_READ\_NS
  - Kernel Interface Page, [438](#)
- L4\_KIP\_OFFS\_READ\_US
  - Kernel Interface Page, [438](#)
- l4\_kip\_version
  - Kernel Interface Page, [442](#)
- l4\_kip\_version\_string
  - Kernel Interface Page, [444](#)
- L4\_LOG2\_PAGESIZE
  - Memory related, [205](#)
- L4\_LOG2\_SUPERPAGESIZE
  - Memory related, [205](#)
- l4\_map\_control
  - Message Items, [241](#)
- L4\_MAP\_ITEM\_GRANT
  - Message Items, [239](#)
- L4\_MAP\_ITEM\_MAP
  - Message Items, [239](#)
- l4\_map\_obj\_control
  - Message Items, [241](#)
- l4\_mem\_archspecific\_acpi\_nvs
  - Memory descriptors (C version), [447](#)
- l4\_mem\_archspecific\_acpi\_tables
  - Memory descriptors (C version), [447](#)
- l4\_mem\_archspecific\_sub\_type\_common\_t
  - Memory descriptors (C version), [446](#)
- l4\_mem\_info\_acpi\_rsdp
  - Memory descriptors (C version), [447](#)
- l4\_mem\_info\_sub\_type\_t
  - Memory descriptors (C version), [447](#)

- L4\_mem\_op\_widths
  - Memory operations., [455](#)
- l4\_mem\_read
  - Memory operations., [455](#)
- l4\_mem\_reserved\_heap
  - Memory descriptors (C version), [447](#)
- l4\_mem\_reserved\_kernel
  - Memory descriptors (C version), [447](#)
- l4\_mem\_reserved\_mmio
  - Memory descriptors (C version), [447](#)
- l4\_mem\_type\_archspecific
  - Memory descriptors (C version), [447](#)
- l4\_mem\_type\_bootloader
  - Memory descriptors (C version), [447](#)
- l4\_mem\_type\_conventional
  - Memory descriptors (C version), [447](#)
- l4\_mem\_type\_dedicated
  - Memory descriptors (C version), [447](#)
- l4\_mem\_type\_info
  - Memory descriptors (C version), [447](#)
- l4\_mem\_type\_reserved
  - Memory descriptors (C version), [447](#)
- l4\_mem\_type\_shared
  - Memory descriptors (C version), [447](#)
- l4\_mem\_type\_t
  - Memory descriptors (C version), [447](#)
- l4\_mem\_type\_undefined
  - Memory descriptors (C version), [447](#)
- L4\_MEM\_WIDTH\_1BYTE
  - Memory operations., [455](#)
- L4\_MEM\_WIDTH\_2BYTE
  - Memory operations., [455](#)
- L4\_MEM\_WIDTH\_4BYTE
  - Memory operations., [455](#)
- l4\_mem\_write
  - Memory operations., [456](#)
- l4\_msg\_item\_consts\_t
  - Message Items, [239](#)
- l4\_msg\_regs\_t, [1553](#)
- l4\_msgtag
  - Message Tag, [261](#)
- L4\_MSGTAG\_ERROR
  - Message Tag, [259](#)
- L4\_MSGTAG\_FLAGS
  - Message Tag, [259](#)
- L4\_msgtag\_flags
  - Message Tag, [259](#)
- l4\_msgtag\_flags
  - Message Tag, [262](#)
- l4\_msgtag\_has\_error
  - Message Tag, [264](#)
- l4\_msgtag\_is\_exception
  - Message Tag, [264](#)
- l4\_msgtag\_is\_io\_page\_fault
  - Message Tag, [265](#)
- l4\_msgtag\_is\_page\_fault
  - Message Tag, [265](#)
- l4\_msgtag\_is\_sigma0
  - Message Tag, [266](#)
- l4\_msgtag\_items
  - Message Tag, [267](#)
- l4\_msgtag\_label
  - Message Tag, [267](#)
- L4\_msgtag\_protocol
  - Message Tag, [259](#)
- L4\_MSGTAG\_SCHEDULE
  - Message Tag, [259](#)
- l4\_msgtag\_t, [1554](#)
  - flags, [1556](#)
  - has\_error, [1556](#)
  - Message Tag, [259](#)
- L4\_MSGTAG\_TRANSFER\_FPU
  - Message Tag, [259](#)
- l4\_msgtag\_words
  - Message Tag, [269](#)
- L4\_NOTHROW
  - Basic Macros, [163](#)
- l4\_ns\_to\_tsc
  - Timestamp Counter, [663](#)
- l4\_obj\_fpage
  - Flexpages, [198](#)
- L4\_obj\_fpage\_ctl
  - Message Items, [240](#)
- L4\_PAGEMASK
  - Memory related, [205](#)
- L4\_PAGESHIFT
  - Memory related, [206](#)
- l4\_platform\_ctl\_cpu\_allow\_shutdown
  - Platform Control C API, [358](#)
- L4\_PLATFORM\_CTL\_CPU\_ALLOW\_SHUTDOWN\_OP
  - L4 IPC Opcodes, [487](#)
- l4\_platform\_ctl\_cpu\_disable
  - Platform Control C API, [358](#)
- L4\_PLATFORM\_CTL\_CPU\_DISABLE\_OP
  - L4 IPC Opcodes, [487](#)
- l4\_platform\_ctl\_cpu\_enable
  - Platform Control C API, [359](#)
- L4\_PLATFORM\_CTL\_CPU\_ENABLE\_OP
  - L4 IPC Opcodes, [487](#)
- l4\_platform\_ctl\_ops
  - L4 IPC Opcodes, [486](#)
- l4\_platform\_ctl\_proto
  - Message Tag, [260](#)
- l4\_platform\_ctl\_set\_task\_asid
  - Platform Control C API, [360](#)
- L4\_PLATFORM\_CTL\_SET\_TASK\_ASID\_OP
  - L4 IPC Opcodes, [487](#)
- L4\_PLATFORM\_CTL\_SYS\_SHUTDOWN\_OP
  - L4 IPC Opcodes, [487](#)
- L4\_PLATFORM\_CTL\_SYS\_SUSPEND\_OP
  - L4 IPC Opcodes, [486](#)
- l4\_platform\_ctl\_system\_shutdown
  - Platform Control C API, [361](#)
- l4\_platform\_ctl\_system\_suspend
  - Platform Control C API, [361](#)
- L4\_PROTO\_ALLOW\_SYSCALL



- Message Tag, [260](#)
- L4\_PROTO\_DEBUGGER
  - Message Tag, [260](#)
- L4\_PROTO\_DMA\_SPACE
  - Message Tag, [260](#)
- L4\_PROTO\_EXCEPTION
  - Message Tag, [260](#)
- L4\_PROTO\_FACTORY
  - Message Tag, [260](#)
- L4\_PROTO\_IO\_PAGE\_FAULT
  - Message Tag, [260](#)
- L4\_PROTO\_IOMMU
  - Message Tag, [260](#)
- L4\_PROTO\_IRQ
  - Message Tag, [260](#)
- L4\_PROTO\_IRQ\_SENDER
  - Message Tag, [260](#)
- L4\_PROTO\_KOBJECT
  - Message Tag, [260](#)
- L4\_PROTO\_LOG
  - Message Tag, [260](#)
- L4\_PROTO\_META
  - Message Tag, [260](#)
- L4\_PROTO\_NONE
  - Message Tag, [260](#)
- L4\_PROTO\_PAGE\_FAULT
  - Message Tag, [260](#)
- L4\_PROTO\_PF\_EXCEPTION
  - Message Tag, [260](#)
- L4\_PROTO\_PLATFORM\_CTL
  - Message Tag, [260](#)
- L4\_PROTO\_SCHEDULER
  - Message Tag, [260](#)
- L4\_PROTO\_SEMAPHORE
  - Message Tag, [260](#)
- L4\_PROTO\_SIGMA0
  - Message Tag, [260](#)
- L4\_PROTO\_SMCCC
  - Message Tag, [260](#)
- L4\_PROTO\_TASK
  - Message Tag, [260](#)
- L4\_PROTO\_THREAD
  - Message Tag, [260](#)
- L4\_PROTO\_THREAD\_GROUP
  - Message Tag, [260](#)
- L4\_PROTO\_VCPU\_CONTEXT
  - Message Tag, [260](#)
- L4\_PROTO\_VM
  - Message Tag, [260](#)
- L4\_RCV\_EP\_BIND\_OP
  - rcv\_endpoint.h, [3261](#)
- l4\_rcv\_ep\_bind\_snd\_destination
  - IPC-Gate API, [288](#)
- l4\_rcv\_ep\_bind\_thread
  - IPC-Gate API, [289](#)
- L4\_rcv\_ep\_ops
  - rcv\_endpoint.h, [3261](#)
- L4\_RCV\_ITEM\_FORWARD\_MAPPINGS
  - Message Items, [240](#)
- L4\_RCV\_ITEM\_LOCAL\_ID
  - Message Items, [240](#)
- L4\_RCV\_ITEM\_SINGLE\_CAP
  - Message Items, [240](#)
- l4\_rcv\_timeout
  - Timeouts, [245](#)
- l4\_rdpmc
  - Timestamp Counter, [664](#)
- l4\_rdpmc\_32
  - Timestamp Counter, [664](#)
- l4\_rdtsc
  - Timestamp Counter, [664](#)
- l4\_rdtsc\_32
  - Timestamp Counter, [665](#)
- l4\_round\_page
  - Memory related, [209](#)
- l4\_round\_size
  - Memory related, [210](#)
- L4\_RPC
  - ipc\_iface, [3121](#)
- L4\_RPC\_DEF
  - ipc\_client, [3113](#)
- L4\_RPC\_NF
  - ipc\_iface, [3122](#)
- L4\_RPC\_NF\_OP
  - ipc\_iface, [3122](#)
- L4\_RPC\_OP
  - ipc\_iface, [3123](#)
- l4\_sched\_cpu\_set
  - Scheduler, [365](#)
- l4\_sched\_cpu\_set\_t, [1557](#)
  - gran\_offset, [1560](#)
  - granularity, [1558](#)
  - offset, [1558](#)
  - set, [1558](#)
- l4\_sched\_param
  - Scheduler, [365](#)
- l4\_sched\_param\_t, [1561](#)
  - prio, [1562](#)
- L4\_SCHEDULER\_CLASS\_FIXED\_PRIO
  - Scheduler, [364](#)
- L4\_SCHEDULER\_CLASS\_WFQ
  - Scheduler, [364](#)
- L4\_scheduler\_classes
  - Scheduler, [364](#)
- l4\_scheduler\_idle\_time
  - Scheduler, [366](#)
- L4\_SCHEDULER\_IDLE\_TIME\_OP
  - Scheduler, [365](#)
- l4\_scheduler\_info
  - Scheduler, [367](#)
- L4\_SCHEDULER\_INFO\_OP
  - Scheduler, [364](#)
- l4\_scheduler\_info\_with\_classes
  - Scheduler, [368](#)
- l4\_scheduler\_is\_online
  - Scheduler, [369](#)

- L4\_scheduler\_ops
  - Scheduler, [364](#)
- l4\_scheduler\_run\_thread
  - Scheduler, [369](#)
- L4\_SCHEDULER\_RUN\_THREAD\_OP
  - Scheduler, [365](#)
- l4\_semaphore\_down
  - Kernel-provided semaphore, [371](#)
- l4\_semaphore\_up
  - Kernel-provided semaphore, [372](#)
- l4\_sleep
  - Utility Functions, [653](#)
- l4\_snd\_fpage\_t, [1562](#)
- l4\_snd\_timeout
  - Timeouts, [246](#)
- l4\_sndfpage\_add
  - Object Invocation, [235](#)
- L4\_SUPERPAGEMASK
  - Memory related, [206](#)
- L4\_SUPERPAGESHIFT
  - Memory related, [207](#)
- L4\_SUPERPAGESIZE
  - Memory related, [207](#)
- L4\_sys\_segment
  - segment.h, [2465](#)
- l4\_syscall\_flags\_t
  - Object Invocation, [217](#)
- L4\_SYSF\_CALL
  - Object Invocation, [218](#)
- L4\_SYSF\_NONE
  - Object Invocation, [218](#)
- L4\_SYSF\_OPEN\_WAIT
  - Object Invocation, [218](#)
- L4\_SYSF\_RECV
  - Object Invocation, [218](#)
- L4\_SYSF\_REPLY
  - Object Invocation, [218](#)
- L4\_SYSF\_REPLY\_AND\_WAIT
  - Object Invocation, [218](#)
- L4\_SYSF\_SEND
  - Object Invocation, [218](#)
- L4\_SYSF\_SEND\_AND\_WAIT
  - Object Invocation, [218](#)
- L4\_SYSF\_WAIT
  - Object Invocation, [218](#)
- l4\_task\_add\_ku\_mem
  - Task, [375](#)
- L4\_TASK\_ADD\_KU\_MEM\_OP
  - L4 IPC Opcodes, [487](#)
- l4\_task\_cap\_equal
  - Task, [375](#)
- L4\_TASK\_CAP\_INFO\_OP
  - L4 IPC Opcodes, [487](#)
- l4\_task\_cap\_valid
  - Task, [377](#)
- l4\_task\_delete\_obj
  - Task, [378](#)
- L4\_TASK\_LDT\_SET\_X86\_OP
  - L4 IPC Opcodes, [487](#)
- L4\_task\_ldt\_x86\_consts
  - segment.h, [2465](#), [2474](#)
- L4\_TASK\_LDT\_X86\_ENTRY\_SIZE
  - segment.h, [2465](#), [2474](#)
- L4\_TASK\_LDT\_X86\_MAX\_ENTRIES
  - segment.h, [2465](#), [2474](#)
- l4\_task\_map
  - Task, [379](#)
- L4\_TASK\_MAP\_OP
  - L4 IPC Opcodes, [487](#)
- L4\_TASK\_MAP\_VGICC\_ARM\_OP
  - L4 IPC Opcodes, [487](#)
- L4\_task\_ops
  - L4 IPC Opcodes, [487](#)
- l4\_task\_release\_cap
  - Task, [380](#)
- l4\_task\_unmap
  - Task, [382](#)
- l4\_task\_unmap\_batch
  - Task, [384](#)
- L4\_TASK\_UNMAP\_OP
  - L4 IPC Opcodes, [487](#)
- l4\_task\_vgicc\_map
  - Task, [385](#)
- L4\_THREAD\_AMD64\_GET\_SEGMENT\_INFO\_OP
  - L4 IPC Opcodes, [488](#)
- L4\_THREAD\_AMD64\_SET\_SEGMENT\_BASE\_OP
  - L4 IPC Opcodes, [488](#)
- l4\_thread\_arm\_set\_tpidruro
  - Thread, [390](#)
- L4\_THREAD\_ARM\_TPIDRURO\_OP
  - L4 IPC Opcodes, [488](#)
- L4\_THREAD\_CONTROL\_ALIEN
  - Thread, [389](#)
- l4\_thread\_control\_alien
  - Thread control, [410](#)
- l4\_thread\_control\_bind
  - Thread control, [411](#)
- L4\_THREAD\_CONTROL\_BIND\_TASK
  - Thread, [389](#)
- l4\_thread\_control\_commit
  - Thread control, [412](#)
- l4\_thread\_control\_exc\_handler
  - Thread control, [413](#)
- L4\_thread\_control\_flags
  - Thread, [388](#)
- L4\_THREAD\_CONTROL\_MR\_IDX\_BIND\_TASK
  - Thread, [389](#)
- L4\_THREAD\_CONTROL\_MR\_IDX\_BIND\_UTCB
  - Thread, [389](#)
- L4\_THREAD\_CONTROL\_MR\_IDX\_EXC\_HANDLER
  - Thread, [389](#)
- L4\_THREAD\_CONTROL\_MR\_IDX\_FLAG\_VALS
  - Thread, [389](#)
- L4\_THREAD\_CONTROL\_MR\_IDX\_FLAGS
  - Thread, [389](#)
- L4\_THREAD\_CONTROL\_MR\_IDX\_PAGER

- Thread, [389](#)
- L4\_thread\_control\_mr\_indices
  - Thread, [389](#)
- L4\_THREAD\_CONTROL\_OP
  - L4 IPC Opcodes, [487](#)
- l4\_thread\_control\_pager
  - Thread control, [414](#)
- L4\_THREAD\_CONTROL\_SET\_EXC\_HANDLER
  - Thread, [389](#)
- L4\_THREAD\_CONTROL\_SET\_PAGER
  - Thread, [388](#)
- l4\_thread\_control\_start
  - Thread control, [414](#)
- l4\_thread\_ex\_regs
  - Thread, [391](#)
- L4\_THREAD\_EX\_REGS\_ARCH\_MASK
  - Thread, [389](#)
- L4\_THREAD\_EX\_REGS\_ARM64\_SET\_EL\_EL0
  - Thread, [390](#)
- L4\_THREAD\_EX\_REGS\_ARM64\_SET\_EL\_EL1
  - Thread, [390](#)
- L4\_THREAD\_EX\_REGS\_ARM64\_SET\_EL\_KEEP
  - Thread, [390](#)
- L4\_THREAD\_EX\_REGS\_ARM64\_SET\_EL\_MASK
  - Thread, [390](#)
- L4\_THREAD\_EX\_REGS\_ARM\_SET\_EL\_EL0
  - Thread, [390](#)
- L4\_THREAD\_EX\_REGS\_ARM\_SET\_EL\_EL1
  - Thread, [390](#)
- L4\_THREAD\_EX\_REGS\_ARM\_SET\_EL\_KEEP
  - Thread, [390](#)
- L4\_THREAD\_EX\_REGS\_ARM\_SET\_EL\_MASK
  - Thread, [390](#)
- L4\_THREAD\_EX\_REGS\_CANCEL
  - Thread, [389](#)
- L4\_thread\_ex\_regs\_flags
  - Thread, [389](#)
- L4\_thread\_ex\_regs\_flags\_arm
  - Thread, [389](#)
- L4\_thread\_ex\_regs\_flags\_arm64
  - Thread, [390](#)
- L4\_THREAD\_EX\_REGS\_OP
  - L4 IPC Opcodes, [487](#)
- l4\_thread\_ex\_regs\_ret
  - Thread, [392](#)
- l4\_thread\_ex\_regs\_ret\_u
  - Thread, [393](#)
- L4\_THREAD\_EX\_REGS\_TRIGGER\_EXCEPTION
  - Thread, [389](#)
- l4\_thread\_ex\_regs\_u
  - Thread, [394](#)
- l4\_thread\_group\_add
  - Thread groups, [421](#)
- l4\_thread\_group\_remove
  - Thread groups, [421](#)
- l4\_thread\_modify\_sender\_add
  - Thread, [395](#)
- l4\_thread\_modify\_sender\_commit
  - Thread, [396](#)
- L4\_THREAD\_MODIFY\_SENDER\_OP
  - L4 IPC Opcodes, [487](#)
- l4\_thread\_modify\_sender\_start
  - Thread, [397](#)
- L4\_THREAD\_OPCODE\_MASK
  - L4 IPC Opcodes, [488](#)
- L4\_thread\_ops
  - L4 IPC Opcodes, [487](#)
- l4\_thread\_register\_del\_irq
  - Thread, [398](#)
- L4\_THREAD\_REGISTER\_DELETE\_IRQ\_OP
  - L4 IPC Opcodes, [487](#)
- l4\_thread\_register\_doorbell\_irq
  - Thread, [399](#)
- L4\_THREAD\_REGISTER\_DOORBELL\_IRQ\_OP
  - L4 IPC Opcodes, [487](#)
- l4\_thread\_regs\_t, [1563](#)
  - error, [1564](#)
  - free\_marker, [1564](#)
- L4\_THREAD\_STATS\_OP
  - L4 IPC Opcodes, [487](#)
- l4\_thread\_stats\_time
  - Thread, [400](#)
- l4\_thread\_switch
  - Thread, [401](#)
- L4\_THREAD\_SWITCH\_OP
  - L4 IPC Opcodes, [487](#)
- l4\_thread\_vcpu\_control
  - Thread, [401](#)
- l4\_thread\_vcpu\_control\_ext
  - Thread, [402](#)
- l4\_thread\_vcpu\_control\_ext\_u
  - Thread, [403](#)
- L4\_THREAD\_VCPU\_CONTROL\_OP
  - L4 IPC Opcodes, [487](#)
- l4\_thread\_vcpu\_control\_u
  - Thread, [405](#)
- l4\_thread\_vcpu\_resume\_commit
  - Thread, [406](#)
- L4\_THREAD\_VCPU\_RESUME\_OP
  - L4 IPC Opcodes, [487](#)
- l4\_thread\_vcpu\_resume\_start
  - Thread, [407](#)
- L4\_THREAD\_X86\_GDT\_OP
  - L4 IPC Opcodes, [487](#)
- l4\_thread\_yield
  - Thread, [408](#)
- l4\_timeout
  - Timeouts, [246](#)
- l4\_timeout\_abs
  - Timeouts, [247](#)
- l4\_timeout\_get
  - Timeouts, [248](#)
- l4\_timeout\_is\_absolute
  - Timeouts, [249](#)
- l4\_timeout\_rel
  - Timeouts, [249](#)

- `l4_timeout_rel_get`
  - Timeouts, [250](#)
- `l4_timeout_s`, [1564](#)
  - Timeouts, [245](#)
- `l4_timeout_t`, [1565](#)
  - Timeouts, [245](#)
- `L4_TIMEOUT_US_MAX`
  - Timeouts, [244](#)
- `l4_touch_ro`
  - Utility Functions, [653](#)
- `l4_touch_rw`
  - Utility Functions, [654](#)
- `l4_trunc_page`
  - Memory related, [211](#)
- `l4_trunc_size`
  - Memory related, [212](#)
- `l4_tsc_init`
  - Timestamp Counter, [665](#)
- `l4_tsc_to_ns`
  - Timestamp Counter, [666](#)
- `l4_tsc_to_s_and_ns`
  - Timestamp Counter, [666](#)
- `l4_tsc_to_us`
  - Timestamp Counter, [667](#)
- `L4_TYPES_FLAGS_OPS_DEF`
  - types, [3159](#)
- `l4_unmap_flags_t`
  - Task, [374](#)
- `l4_usleep`
  - Utility Functions, [654](#)
- `l4_utcb_br`
  - Virtual Registers (UTCBS), [273](#)
- `L4_UTCB_BUF_REGS_OFFSET`
  - x86 Virtual Registers (UTCB), [283](#)
- `L4_utcb_consts_x86`
  - x86 Virtual Registers (UTCB), [283](#)
- `l4_utcb_exc`
  - Exception registers, [276](#)
- `l4_utcb_exc_is_ex_regs_exception`
  - Exception registers, [277](#)
- `l4_utcb_exc_is_pf`
  - Exception registers, [277](#)
- `l4_utcb_exc_pc`
  - Exception registers, [278](#)
- `l4_utcb_exc_pc_set`
  - Exception registers, [278](#)
- `L4_UTCB_EXCEPTION_REGS_SIZE`
  - x86 Virtual Registers (UTCB), [283](#)
- `L4_UTCB_GENERIC_BUFFERS_SIZE`
  - x86 Virtual Registers (UTCB), [283](#)
- `L4_UTCB_GENERIC_DATA_SIZE`
  - x86 Virtual Registers (UTCB), [283](#)
- `L4_UTCB_INHERIT_FPU`
  - x86 Virtual Registers (UTCB), [283](#)
- `l4_utcb_mr`
  - Virtual Registers (UTCBS), [273](#)
- `l4_utcb_mr64_idx`
  - Timeouts, [250](#)
- `L4_UTCB_MSG_REGS_OFFSET`
  - x86 Virtual Registers (UTCB), [283](#)
- `L4_UTCB_OFFSET`
  - x86 Virtual Registers (UTCB), [283](#)
- `l4_utcb_t`
  - Virtual Registers (UTCBS), [272](#)
- `l4_utcb_tcr`
  - Virtual Registers (UTCBS), [274](#)
- `L4_UTCB_THREAD_REGS_OFFSET`
  - x86 Virtual Registers (UTCB), [283](#)
- `l4_vcon_attr_t`, [1566](#)
  - set\_raw, [1567](#)
  - Virtual Console, [424](#)
- `L4_VCON_ECHO`
  - Virtual Console, [424](#)
- `l4_vcon_get_attr`
  - Virtual Console, [425](#)
- `L4_VCON_GET_ATTR_OP`
  - L4 IPC Opcodes, [488](#)
- `l4_vcon_get_attr_u`
  - Virtual Console, [426](#)
- `L4_vcon_i_flags`
  - Virtual Console, [424](#)
- `L4_VCON_ICANON`
  - Virtual Console, [424](#)
- `L4_VCON_ICRNL`
  - Virtual Console, [424](#)
- `L4_VCON_IGNCR`
  - Virtual Console, [424](#)
- `L4_VCON_INLCR`
  - Virtual Console, [424](#)
- `L4_vcon_l_flags`
  - Virtual Console, [424](#)
- `L4_vcon_o_flags`
  - Virtual Console, [424](#)
- `L4_VCON_OCRNL`
  - Virtual Console, [425](#)
- `L4_VCON_ONLCR`
  - Virtual Console, [425](#)
- `L4_VCON_ONLRET`
  - Virtual Console, [425](#)
- `L4_vcon_ops`
  - L4 IPC Opcodes, [488](#)
- `l4_vcon_read`
  - Virtual Console, [427](#)
- `L4_vcon_read_flags`
  - vcon.h, [3312](#)
- `L4_VCON_READ_OP`
  - L4 IPC Opcodes, [488](#)
- `L4_VCON_READ_SIZE`
  - Virtual Console, [425](#)
- `L4_VCON_READ_SIZE_MASK`
  - vcon.h, [3312](#)
- `L4_VCON_READ_STAT_BREAK`
  - vcon.h, [3312](#)
- `L4_VCON_READ_STAT_DONE`
  - vcon.h, [3312](#)
- `l4_vcon_read_u`

- Virtual Console, [428](#)
- `l4_vcon_read_with_flags`
  - Virtual Console, [429](#)
- `l4_vcon_send`
  - Virtual Console, [430](#)
- `l4_vcon_send_u`
  - Virtual Console, [431](#)
- `l4_vcon_set_attr`
  - Virtual Console, [432](#)
- `L4_VCON_SET_ATTR_OP`
  - L4 IPC Opcodes, [488](#)
- `l4_vcon_set_attr_raw`
  - Virtual Console, [433](#)
- `l4_vcon_set_attr_u`
  - Virtual Console, [433](#)
- `L4_vcon_size_consts`
  - Virtual Console, [425](#)
- `l4_vcon_write`
  - Virtual Console, [434](#)
- `L4_VCON_WRITE_OP`
  - L4 IPC Opcodes, [488](#)
- `L4_VCON_WRITE_SIZE`
  - Virtual Console, [425](#)
- `l4_vcon_write_u`
  - Virtual Console, [435](#)
- `l4_vcpu_arch_state_t`, [1568](#)
- `l4_vcpu_check_version`
  - `vcpu.h`, [3427](#)
- `L4_vcpu_e_field_ids`
  - `__vcpu-arch.h`, [2489](#), [2493](#)
- `L4_VCPU_E_VTMR_CFG`
  - `__vcpu-arch.h`, [2490](#), [2493](#)
- `L4_VCPU_F_EXCEPTIONS`
  - vCPU API, [418](#)
- `L4_VCPU_F_FPU_ENABLED`
  - vCPU API, [418](#)
- `L4_VCPU_F_IRQ`
  - vCPU API, [418](#)
- `L4_VCPU_F_PAGE_FAULTS`
  - vCPU API, [418](#)
- `L4_VCPU_F_USER_MODE`
  - vCPU API, [418](#)
- `l4_vcpu_ipc_regs_t`, [1568](#)
- `L4_VCPU_OFFSET_EXT_INFOS`
  - vCPU API, [419](#), [420](#)
- `L4_VCPU_OFFSET_EXT_STATE`
  - vCPU API, [419](#)
- `l4_vcpu_regs_t`, [1569](#)
  - `ax`, [1571](#)
  - `bp`, [1571](#)
  - `bx`, [1572](#)
  - `cx`, [1572](#)
  - `di`, [1572](#)
  - `dx`, [1572](#)
  - `si`, [1572](#)
- `L4_VCPU_SF_IRQ_PENDING`
  - vCPU API, [420](#)
- `L4_vcpu_state_flags`
  - vCPU API, [417](#)
- `L4_vcpu_state_offset`
  - vCPU API, [418](#), [419](#)
- `l4_vcpu_state_t`, [1573](#)
  - version, [1576](#)
- `L4_VCPU_STATE_VERSION`
  - `__vcpu-arch.h`, [2487](#), [2489](#), [2493](#), [2496](#)
- `L4_vcpu_sticky_flags`
  - vCPU API, [420](#)
- `L4_virtio_cmd`
  - L4 VIRTIO Transport Layer, [491](#)
- `L4_virtio_irq_status`
  - L4 VIRTIO Transport Layer, [491](#)
- `L4_virtio_opcodes`
  - L4 VIRTIO Transport Layer, [492](#)
- `l4_vm_svm_vmc_b_control_area`, [1577](#)
- `l4_vm_svm_vmc_b_state_save_area`, [1577](#)
- `l4_vm_svm_vmc_b_state_save_area_seg`, [1578](#)
- `l4_vm_svm_vmc_b_t`, [1579](#)
- `l4_vm_tz_state`, [1581](#)
- `L4_VM_VMX_BASIC_REG`
  - VM API for VMX, [310](#)
- `L4_vm_vmx_caps_regs`
  - VM API for VMX, [309](#)
- `l4_vm_vmx_clear`
  - VM API for VMX, [311](#)
- `L4_VM_VMX_CR0_FIXED0_REG`
  - VM API for VMX, [310](#)
- `L4_VM_VMX_CR0_FIXED1_REG`
  - VM API for VMX, [310](#)
- `L4_VM_VMX_CR4_FIXED0_REG`
  - VM API for VMX, [310](#)
- `L4_VM_VMX_CR4_FIXED1_REG`
  - VM API for VMX, [310](#)
- `L4_vm_vmx_dfl1_regs`
  - VM API for VMX, [310](#)
- `L4_VM_VMX_ENTRY_CTL5_DFL1_REG`
  - VM API for VMX, [310](#)
- `L4_VM_VMX_EPT_VPID_CAP_REG`
  - VM API for VMX, [310](#)
- `L4_VM_VMX_EXIT_CTL5_DFL1_REG`
  - VM API for VMX, [310](#)
- `l4_vm_vmx_field_len`
  - VM API for VMX, [312](#)
- `l4_vm_vmx_field_order`
  - VM API for VMX, [313](#)
- `l4_vm_vmx_get_caps`
  - VM API for VMX, [313](#)
- `l4_vm_vmx_get_caps_default1`
  - VM API for VMX, [315](#)
- `l4_vm_vmx_get_cr2_index`
  - VM API for VMX, [315](#)
- `l4_vm_vmx_get_hw_vmcs`
  - VM API for VMX, [315](#)
- `L4_VM_VMX_MISC_REG`
  - VM API for VMX, [310](#)
- `L4_VM_VMX_NESTED_REVISION`
  - VM API for VMX, [310](#)

- L4\_VM\_VMX\_NUM\_CAPS\_REGS
  - VM API for VMX, [310](#)
- L4\_VM\_VMX\_NUM\_DFL1\_REGS
  - VM API for VMX, [310](#)
- L4\_VM\_VMX\_PINBASED\_CTL5\_DFL1\_REG
  - VM API for VMX, [310](#)
- L4\_VM\_VMX\_PROCBASED\_CTL5\_2\_REG
  - VM API for VMX, [310](#)
- L4\_VM\_VMX\_PROCBASED\_CTL5\_DFL1\_REG
  - VM API for VMX, [310](#)
- l4\_vm\_vmx\_ptr\_load
  - VM API for VMX, [316](#)
- l4\_vm\_vmx\_read
  - VM API for VMX, [317](#)
- l4\_vm\_vmx\_read\_16
  - VM API for VMX, [317](#)
- l4\_vm\_vmx\_read\_32
  - VM API for VMX, [318](#)
- l4\_vm\_vmx\_read\_64
  - VM API for VMX, [319](#)
- l4\_vm\_vmx\_read\_nat
  - VM API for VMX, [319](#)
- l4\_vm\_vmx\_set\_hw\_vmcs
  - VM API for VMX, [320](#)
- L4\_vm\_vmx\_sw\_fields
  - VM API for VMX, [310](#)
- L4\_VM\_VMX\_TRUE\_ENTRY\_CTL5\_REG
  - VM API for VMX, [310](#)
- L4\_VM\_VMX\_TRUE\_EXIT\_CTL5\_REG
  - VM API for VMX, [310](#)
- L4\_VM\_VMX\_TRUE\_PINBASED\_CTL5\_REG
  - VM API for VMX, [310](#)
- L4\_VM\_VMX\_TRUE\_PROCBASED\_CTL5\_REG
  - VM API for VMX, [310](#)
- l4\_vm\_vmx\_vcpu\_infos\_t, [1581](#)
  - df11, [1582](#)
- l4\_vm\_vmx\_vcpu\_state\_t, [1582](#)
  - VM API for VMX, [307](#)
- l4\_vm\_vmx\_vcpu\_vmcs\_t, [1584](#)
  - VM API for VMX, [307](#)
- L4\_VM\_VMX\_VMCS\_CR2
  - VM API for VMX, [311](#)
- L4\_VM\_VMX\_VMCS\_ENUM\_REG
  - VM API for VMX, [310](#)
- L4\_VM\_VMX\_VMCS\_MSR\_CSTAR
  - VM API for VMX, [311](#)
- L4\_VM\_VMX\_VMCS\_MSR\_KERNEL\_GS\_BASE
  - VM API for VMX, [311](#)
- L4\_VM\_VMX\_VMCS\_MSR\_LSTAR
  - VM API for VMX, [311](#)
- L4\_VM\_VMX\_VMCS\_MSR\_STAR
  - VM API for VMX, [311](#)
- L4\_VM\_VMX\_VMCS\_MSR\_SYSCALL\_MASK
  - VM API for VMX, [311](#)
- L4\_VM\_VMX\_VMCS\_MSR\_TSC\_AUX
  - VM API for VMX, [311](#)
- L4\_VM\_VMX\_VMCS\_NAT\_ARG0
  - VM API for VMX, [311](#)
- L4\_VM\_VMX\_VMCS\_NAT\_ARG1
  - VM API for VMX, [311](#)
- L4\_VM\_VMX\_VMCS\_NAT\_ARG2
  - VM API for VMX, [311](#)
- L4\_VM\_VMX\_VMCS\_NAT\_ARG3
  - VM API for VMX, [311](#)
- L4\_VM\_VMX\_VMCS\_SIZE\_DIRTY\_BITMAP
  - VM API for VMX, [311](#)
- L4\_VM\_VMX\_VMCS\_SIZE\_VALUES
  - VM API for VMX, [311](#)
- L4\_vm\_vmx\_vmcs\_sizes
  - VM API for VMX, [311](#)
- L4\_VM\_VMX\_VMCS\_XCR0
  - VM API for VMX, [311](#)
- l4\_vm\_vmx\_write
  - VM API for VMX, [321](#)
- l4\_vm\_vmx\_write\_16
  - VM API for VMX, [322](#)
- l4\_vm\_vmx\_write\_32
  - VM API for VMX, [323](#)
- l4\_vm\_vmx\_write\_64
  - VM API for VMX, [323](#)
- l4\_vm\_vmx\_write\_nat
  - VM API for VMX, [324](#)
- l4\_vmx\_offset\_table\_t, [1585](#)
  - VM API for VMX, [308](#)
- L4\_WHOLE\_ADDRESS\_SPACE
  - Flexpages, [184](#)
- L4\_WHOLE\_IOADDRESS\_SPACE
  - Flexpages, [184](#)
- L4drivers::Mmio\_register\_block< MAX\_BITS >, [1587](#)
- L4drivers::Register\_block< MAX\_BITS, BLOCK >, [1589](#)
  - operator[], [1590](#), [1591](#)
  - r, [1591](#), [1592](#)
- L4drivers::Register\_block\_base< MAX\_BITS >, [1593](#)
- L4drivers::Register\_block\_impl< BASE, MAX\_BITS >, [1594](#)
- L4drivers::Register\_block\_tmpl< BLOCK >, [1596](#)
- L4drivers::Register\_tmpl< BITS, BLOCK >, [1597](#)
  - clear, [1601](#)
  - modify, [1601](#)
  - operator=, [1602](#)
  - set, [1602](#)
  - write, [1603](#)
- L4drivers::Ro\_register\_block< MAX\_BITS, BLOCK >, [1604](#)
  - operator[], [1605](#)
  - r, [1605](#)
- L4drivers::Ro\_register\_tmpl< BITS, BLOCK >, [1606](#)
  - operator value\_type, [1607](#)
  - read, [1607](#)
- L4IO\_DEVICE\_ANY
  - IO interface, [465](#)
- L4IO\_DEVICE\_INVALID
  - IO interface, [465](#)
- L4IO\_DEVICE\_OTHER
  - IO interface, [465](#)

- L4IO\_DEVICE\_PCI
  - IO interface, [465](#)
- l4io\_device\_types\_t
  - IO interface, [464](#)
- L4IO\_DEVICE\_USB
  - IO interface, [465](#)
- l4io\_get\_root\_device
  - io.h, [2538](#)
- l4io\_has\_resource
  - IO interface, [466](#)
- l4io\_iomem\_flags\_t
  - IO interface, [465](#)
- l4io\_iterate\_devices
  - io.h, [2538](#)
- l4io\_lookup\_device
  - IO interface, [467](#)
- l4io\_lookup\_resource
  - IO interface, [467](#)
- L4IO\_MEM\_CACHED
  - IO interface, [465](#)
- L4IO\_MEM\_EAGER\_MAP
  - IO interface, [465](#)
- L4IO\_MEM\_NONCACHED
  - IO interface, [465](#)
- L4IO\_MEM\_USE\_MTRR
  - IO interface, [465](#)
- L4IO\_MEM\_USE\_RESERVED\_AREA
  - IO interface, [465](#)
- l4io\_release\_iomem
  - IO interface, [468](#)
- l4io\_release\_ioport
  - IO interface, [468](#)
- l4io\_request\_all\_ioports
  - io.h, [2538](#)
- l4io\_request\_icu
  - io.h, [2539](#)
- l4io\_request\_iomem
  - IO interface, [468](#)
- l4io\_request\_iomem\_region
  - IO interface, [469](#)
- l4io\_request\_ioport
  - IO interface, [470](#)
- l4io\_request\_resource\_iomem
  - IO interface, [470](#)
- L4IO\_RESOURCE\_ANY
  - IO interface, [465](#)
- L4IO\_RESOURCE\_INVALID
  - IO interface, [465](#)
- L4IO\_RESOURCE\_IRQ
  - IO interface, [465](#)
- L4IO\_RESOURCE\_MEM
  - IO interface, [465](#)
- L4IO\_RESOURCE\_PORT
  - IO interface, [465](#)
- l4io\_resource\_t
  - IO interface, [464](#)
- l4io\_resource\_types\_t
  - IO interface, [465](#)
- l4irq\_attach
  - Interface using direct functionality., [475](#)
- l4irq\_attach\_cap
  - Interface using direct functionality., [479](#)
- l4irq\_attach\_cap\_ft
  - Interface using direct functionality., [480](#)
- l4irq\_attach\_ft
  - Interface using direct functionality., [475](#)
- l4irq\_attach\_thread
  - Interface using direct functionality., [476](#)
- l4irq\_attach\_thread\_cap
  - Interface using direct functionality., [480](#)
- l4irq\_attach\_thread\_cap\_ft
  - Interface using direct functionality., [481](#)
- l4irq\_attach\_thread\_ft
  - Interface using direct functionality., [476](#)
- l4irq\_detach
  - Interface using direct functionality., [477](#)
- l4irq\_release
  - Interface for asynchronous ISR handlers., [482](#)
- l4irq\_request
  - Interface for asynchronous ISR handlers., [482](#)
- l4irq\_request\_cap
  - Interface for asynchronous ISR handlers with a given IRQ capability., [484](#)
- l4irq\_unmask
  - Interface using direct functionality., [477](#)
- l4irq\_unmask\_and\_wait\_any
  - Interface using direct functionality., [477](#)
- l4irq\_wait
  - Interface using direct functionality., [478](#)
- l4irq\_wait\_any
  - Interface using direct functionality., [478](#)
- l4la\_alloc
  - list\_alloc.h, [3372](#)
- l4la\_avail
  - list\_alloc.h, [3372](#)
- l4la\_dump
  - list\_alloc.h, [3372](#)
- l4la\_free
  - list\_alloc.h, [3372](#)
- l4la\_init
  - list\_alloc.h, [3373](#)
- l4mod.h
  - L4util\_l4mod\_mod\_flag\_kernel, [3370](#)
  - L4util\_l4mod\_mod\_flag\_mask, [3370](#)
  - L4util\_l4mod\_mod\_flag\_roottask, [3370](#)
  - L4util\_l4mod\_mod\_flag\_sigma0, [3370](#)
  - L4util\_l4mod\_mod\_flag\_unspec, [3370](#)
  - L4util\_l4mod\_mod\_info\_flag, [3370](#)
- L4Re, [775](#)
  - chkcapp, [782](#)
  - chkipc, [784](#)
  - chksys, [785](#), [786](#)
  - make\_shared\_cap, [788](#)
  - make\_shared\_del\_cap, [790](#)
  - make\_unique\_cap, [791](#)
  - make\_unique\_del\_cap, [791](#)



- Shared\_cap, [777](#)
- shared\_cap, [778](#)
- Shared\_del\_cap, [778](#)
- shared\_del\_cap, [778](#)
- throw\_error, [792](#)
- Unique\_cap, [780](#)
- unique\_cap, [780](#)
- Unique\_del\_cap, [780](#)
- unique\_del\_cap, [782](#)
- L4Re Build System, [35](#)
- L4Re C Interface, [531](#)
- L4Re C++ Interface, [589](#)
- L4Re Capability API, [592](#)
  - cap\_alloc, [594](#)
  - make\_ref\_cap, [593](#)
  - make\_ref\_del\_cap, [593](#)
- L4Re ELF Auxiliary Information, [597](#)
  - L4RE\_ELF\_AUX\_ELEM, [598](#)
  - L4RE\_ELF\_AUX\_ELEM\_T, [598](#)
  - L4RE\_ELF\_AUX\_T\_EX\_REGS\_FLAGS, [599](#)
  - L4RE\_ELF\_AUX\_T\_KIP\_ADDR, [599](#)
  - L4RE\_ELF\_AUX\_T\_NONE, [599](#)
  - L4RE\_ELF\_AUX\_T\_STACK\_ADDR, [599](#)
  - L4RE\_ELF\_AUX\_T\_STACK\_SIZE, [599](#)
  - L4RE\_ELF\_AUX\_T\_VMA, [599](#)
- L4Re Protocol identifiers, [603](#)
  - L4RE\_PROTO\_DATASPACE, [605](#)
  - L4RE\_PROTO\_DEBUG, [605](#)
  - L4RE\_PROTO\_DMA\_SPACE, [605](#)
  - L4RE\_PROTO\_EVENT, [605](#)
  - L4RE\_PROTO\_GOOS, [605](#)
  - L4RE\_PROTO\_INHIBITOR, [605](#)
  - L4RE\_PROTO\_ITAS, [605](#)
  - L4RE\_PROTO\_MEM\_ALLOC, [605](#)
  - L4RE\_PROTO\_MMIO\_SPACE, [605](#)
  - L4RE\_PROTO\_NAMESPACE, [605](#)
  - L4RE\_PROTO\_PARENT, [605](#)
  - L4RE\_PROTO\_REMOTE\_ACCESS, [605](#)
  - L4RE\_PROTO\_RM, [605](#)
  - L4RE\_PROTO\_RSVD\_1, [605](#)
- L4re\_protocols, [604](#)
- L4Re Servers, [51](#)
- L4Re Util C Interface, [534](#)
- L4Re Util C++ Interface, [591](#)
- L4Re::Cap\_alloc, [1608](#)
  - alloc, [1609](#), [1610](#)
  - free, [1610](#)
- L4Re::Console, [1611](#)
- L4Re::Core::Ref\_ptr< T, CNT >, [1614](#)
  - get, [1617](#)
  - ptr, [1617](#)
  - Ref\_ptr, [1616](#)
  - release, [1617](#)
- L4Re::Dataspace, [1618](#)
  - allocate, [1621](#)
  - clear, [1622](#)
  - copy\_in, [1623](#)
  - flags, [1624](#)
  - info, [1625](#)
  - map, [1626](#)
  - map\_info, [1627](#)
  - map\_region, [1628](#)
  - size, [1630](#)
- L4Re::Dataspace::F, [1631](#)
  - Bufferable, [1633](#)
  - Cacheable, [1633](#)
  - Caching\_mask, [1633](#)
  - Caching\_shift, [1632](#)
  - Flags, [1632](#)
  - Normal, [1633](#)
  - R, [1632](#)
  - Rights\_mask, [1633](#)
  - Ro, [1632](#)
  - RW, [1632](#)
  - RWX, [1632](#)
  - RX, [1632](#)
  - Uncacheable, [1633](#)
  - W, [1632](#)
  - X, [1632](#)
- L4Re::Dataspace::Stats, [1633](#)
- L4Re::Debug\_obj, [1634](#)
  - debug, [1636](#)
- L4Re::Default\_event\_payload, [1637](#)
- L4Re::Dma\_space, [1638](#)
  - associate, [1641](#)
  - Attribute, [1640](#)
  - Attributes, [1640](#)
  - Bidirectional, [1641](#)
  - Coherent, [1641](#)
  - Direction, [1641](#)
  - disassociate, [1642](#)
  - From\_device, [1641](#)
  - map, [1643](#)
  - No\_sync, [1641](#)
  - None, [1641](#)
  - Phys\_space, [1641](#)
  - Space\_attrib, [1641](#)
  - To\_device, [1641](#)
  - unmap, [1644](#)
- L4Re::Env, [1645](#)
  - dbg\_events, [1648](#), [1649](#)
  - env, [1649](#)
  - factory, [1650](#), [1651](#)
  - first\_free\_cap, [1651](#)
  - first\_free\_utcb, [1652](#)
  - get, [1652](#)
  - get\_cap, [1653](#)
  - initial\_caps, [1654](#), [1655](#)
  - itas, [1655](#)
  - log, [1656](#)
  - main\_thread, [1656](#)
  - mem\_alloc, [1657](#)
  - parent, [1657](#)
  - rm, [1658](#)
  - scheduler, [1658](#)
  - task, [1659](#)



- utcb\_area, 1659
- L4Re::Event, 1660
  - get\_axis\_info, 1663
  - get\_buffer, 1664
  - get\_num\_streams, 1665
  - get\_stream\_info, 1666
  - get\_stream\_info\_for\_id, 1667
  - get\_stream\_state\_for\_id, 1668
- L4Re::Event\_buffer\_t< PAYLOAD >, 1669
  - Event\_buffer\_t, 1670
  - next, 1671
  - put, 1671
- L4Re::Event\_buffer\_t< PAYLOAD >::Event, 1672
- L4Re::Inhibitor, 1673
  - acquire, 1676
  - Name\_max, 1676
  - next\_lock\_info, 1677
  - release, 1677
- L4Re::Itas, 1678
  - getitimer, 1682
  - Ignore\_sigaction, 1682
  - raise, 1683
  - register\_thread, 1684
  - setitimer, 1684
  - sigaction, 1685
  - sigaltstack, 1685
  - sigpending, 1686
  - sigprocmask, 1687
  - unregister\_thread, 1688
- L4Re::Log, 1689
  - print, 1694
  - printn, 1694
- L4Re::Mem\_alloc, 1695
  - alloc, 1699
  - Continuous, 1699
  - Fixed\_paddr, 1699
  - info, 1700
  - Mem\_alloc\_flags, 1699
  - Pinned, 1699
  - Super\_pages, 1699
- L4Re::Mem\_alloc::Stats, 1701
  - mem\_free, 1702
  - mem\_limit, 1702
  - mem\_used, 1702
  - quota, 1702
  - quota\_used, 1703
- L4Re::Mmio\_space, 1703
  - Access\_width, 1707
  - mmio\_read, 1708
  - mmio\_write, 1709
  - Wd\_16bit, 1708
  - Wd\_32bit, 1708
  - Wd\_64bit, 1708
  - Wd\_8bit, 1708
- L4Re::Namespace, 1710
  - Link, 1715
  - Overwrite, 1715
  - Partly\_resolved, 1714
  - query, 1715, 1716
  - Query\_result\_flags, 1714
  - Query\_timeout, 1714
  - Register\_flags, 1714
  - register\_obj, 1717
  - Ro, 1715
  - Rs, 1715
  - Rw, 1715
  - Rws, 1715
  - Strong, 1715
  - To\_default, 1714
  - To\_non\_blocking, 1714
  - Trusted, 1715
  - unlink, 1718
- L4Re::Ned::Cmd\_control, 1720
  - execute, 1720, 1721
- L4Re::Parent, 1722
  - signal, 1725
- L4Re::Random, 1727
  - get\_random, 1730
- L4Re::Rm, 1731
  - attach, 1737, 1739
  - Caching\_shift, 1737
  - detach, 1740–1742
  - Detach\_again, 1737
  - Detach\_exact, 1736
  - Detach\_flags, 1736
  - Detach\_keep, 1736
  - Detach\_overlap, 1736
  - Detach\_result, 1736
  - Detached\_ds, 1737
  - find, 1744
  - free\_area, 1745
  - get\_areas, 1747
  - get\_info, 1749
  - get\_regions, 1750
  - Kept\_ds, 1737
  - Region\_flag\_shifts, 1737
  - reserve\_area, 1751, 1753
  - Split\_ds, 1737
- L4Re::Rm::Area, 1755
- L4Re::Rm::F, 1756
  - Attach\_flags, 1757
  - Attach\_mask, 1757
  - Cache\_buffered, 1758
  - Cache\_normal, 1758
  - Cache\_uncached, 1758
  - Caching\_mask, 1758
  - Detach\_free, 1758
  - Ds\_map\_mask, 1758
  - Eager\_map, 1757
  - In\_area, 1757
  - Kernel, 1758
  - No\_eager\_map, 1757
  - Pager, 1758
  - R, 1758
  - Region\_flags, 1757
  - Region\_flags\_mask, 1758

- Reserved, 1758
- Rights\_mask, 1758
- RW, 1758
- RWX, 1758
- RX, 1758
- Search\_addr, 1757
- W, 1758
- X, 1758
- L4Re::Rm::Region, 1758
- L4Re::Rm::Unique\_region< T >, 1759
  - ~Unique\_region, 1762
  - get, 1762
  - is\_valid, 1763
  - operator=, 1763
  - release, 1763
  - reset, 1764
  - Unique\_region, 1761, 1762
- L4Re::Smart\_cap\_auto< Unmap\_flags >, 1764
- L4Re::Smart\_count\_cap< Unmap\_flags >, 1765
- L4Re::Util, 794
  - make\_shared\_cap, 802
  - make\_shared\_del\_cap, 802
  - make\_unique\_cap, 802
  - make\_unique\_del\_cap, 803
  - Shared\_cap, 797
  - shared\_cap, 797
  - Shared\_del\_cap, 798
  - shared\_del\_cap, 798
  - Unique\_cap, 799
  - unique\_cap, 800
  - Unique\_del\_cap, 800
  - unique\_del\_cap, 801
- L4Re::Util::Cap\_alloc, 1766
  - alloc, 1768
  - free, 1768
- L4Re::Util::Bitmap< BITS >, 1769
  - scan\_zero, 1773
- L4Re::Util::Bitmap\_base, 1773
  - atomic\_clear\_bit, 1777
  - atomic\_get\_and\_clear, 1777
  - atomic\_get\_and\_set, 1777
  - atomic\_set\_bit, 1778
  - bit, 1778
  - bit\_index, 1779
  - C\_bits, 1777
  - clear\_bit, 1779
  - operator[], 1779, 1780
  - scan\_zero, 1780
  - set\_bit, 1781
  - W\_bits, 1777
  - word\_index, 1781
- L4Re::Util::Bitmap\_base::Bit, 1782
- L4Re::Util::Bitmap\_base::Char< BITS >, 1782
- L4Re::Util::Bitmap\_base::Word< BITS >, 1783
- L4Re::Util::Br\_manager, 1784
  - alloc\_buffer\_demand, 1787
  - get\_rcv\_cap, 1788
  - realloc\_rcv\_cap, 1788
  - set\_rcv\_cap\_flags, 1789
- L4Re::Util::Br\_manager\_hooks, 1790
- L4Re::Util::Br\_manager\_timeout\_hooks, 1792
- L4Re::Util::Cap\_alloc\_base, 1796
- L4Re::Util::Counter< COUNTER >, 1797
  - dec, 1797
  - inc, 1797
- L4Re::Util::Counter\_atomic< COUNTER >, 1799
  - dec, 1800
  - inc, 1800
- L4Re::Util::Counting\_cap\_alloc< COUNTERTYPE, Dbg >, 1800
  - alloc, 1802, 1803
  - Counting\_cap\_alloc, 1802
  - free, 1803
  - release, 1804
  - setup, 1805
  - take, 1805
- L4Re::Util::Dataspace\_svr, 1806
  - allocate, 1807
  - clear, 1808
  - copy, 1808
  - is\_static, 1809
  - map, 1809
  - map\_hook, 1810
  - map\_info, 1811
  - page\_shift, 1811
  - release, 1812
  - take, 1812
- L4Re::Util::Event\_buffer\_consumer\_t< PAYLOAD >, 1813
  - foreach\_available\_event, 1815
  - process, 1815
- L4Re::Util::Event\_buffer\_t< PAYLOAD >, 1816
  - attach, 1819
  - buf, 1819
  - detach, 1819
- L4Re::Util::Event\_svr< SVR >, 1820
- L4Re::Util::Event\_t< PAYLOAD >, 1822
  - buffer, 1825
  - init, 1825
  - init\_poll, 1826
  - irq, 1826
  - Mode, 1824
  - Mode\_irq, 1825
  - Mode\_polling, 1825
- L4Re::Util::Item\_alloc\_base, 1827
- L4Re::Util::Names::Name, 1827
- L4Re::Util::Names::Name\_space, 1831
  - alloc\_dynamic\_entry, 1832
  - copy\_receive\_cap, 1832
  - free\_capability, 1833
  - free\_dynamic\_entry, 1833
  - free\_epiface, 1834
  - get\_epiface, 1835
- L4Re::Util::Object\_registry, 1836
  - Object\_registry, 1838
  - register\_irq\_obj, 1839

- register\_obj, [1839](#), [1840](#)
- unregister\_obj, [1841](#)
- L4Re::Util::Ref\_cap< T >, [1842](#)
- L4Re::Util::Ref\_del\_cap< T >, [1843](#)
- L4Re::Util::Registry\_server< LOOP\_HOOKS >, [1844](#)
  - loop, [1849](#)
  - loop\_dbg, [1849](#)
  - Registry\_server, [1848](#), [1849](#)
- L4Re::Util::Smart\_cap\_auto< Unmap\_flags >, [1850](#)
- L4Re::Util::Smart\_count\_cap< Unmap\_flags >, [1851](#)
- L4Re::Util::Vcon\_svr< SVR >, [1852](#)
- L4Re::Util::Video::Goos\_svr, [1853](#)
  - get\_fb, [1855](#)
  - init\_infos, [1855](#)
  - refresh, [1855](#)
  - screen\_info, [1857](#)
  - view\_info, [1857](#)
- L4Re::Vfs, [803](#)
- L4Re::Vfs::Be\_file, [1858](#)
  - check\_ready, [1861](#)
  - data\_space, [1861](#)
  - fstat, [1861](#)
  - unlock\_all\_locks, [1862](#)
- L4Re::Vfs::Be\_file\_system, [1863](#)
  - ~Be\_file\_system, [1864](#)
  - Be\_file\_system, [1864](#)
  - type, [1865](#)
- L4Re::Vfs::Directory, [1865](#)
  - faccessat, [1868](#)
  - link, [1868](#)
  - mkdir, [1868](#)
  - rename, [1869](#)
  - rmdir, [1869](#)
  - symlink, [1869](#)
  - unlink, [1870](#)
- L4Re::Vfs::File, [1871](#)
- L4Re::Vfs::File\_system, [1874](#)
  - mount, [1875](#)
  - type, [1876](#)
- L4Re::Vfs::Fs, [1876](#)
  - alloc\_fd, [1878](#)
  - free\_fd, [1879](#)
  - get\_file, [1879](#)
  - mount, [1879](#)
  - set\_fd, [1880](#)
- L4Re::Vfs::Generic\_file, [1881](#)
  - check\_ready, [1884](#)
  - fchmod, [1885](#)
  - fstat, [1885](#)
  - get\_status\_flags, [1886](#)
  - Ready\_type, [1884](#)
  - set\_status\_flags, [1887](#)
  - unlock\_all\_locks, [1888](#)
- L4Re::Vfs::Mman, [1890](#)
- L4Re::Vfs::Ops, [1891](#)
- L4Re::Vfs::Regular\_file, [1894](#)
  - data\_space, [1897](#)
  - fdatasync, [1897](#)
  - fsync, [1897](#)
  - ftruncate, [1898](#)
  - get\_lock, [1899](#)
  - lseek, [1900](#)
  - readv, [1901](#)
  - set\_lock, [1902](#)
  - writew, [1903](#)
- L4Re::Vfs::Special\_file, [1904](#)
  - ioctl, [1906](#)
- L4Re::Video::Color\_component, [1907](#)
  - Color\_component, [1908](#)
  - dump, [1908](#)
  - get, [1909](#)
  - operator==, [1909](#)
  - set, [1910](#)
  - shift, [1910](#)
  - size, [1911](#)
- L4Re::Video::Goos, [1912](#)
  - create\_buffer, [1915](#)
  - create\_view, [1916](#)
  - delete\_buffer, [1917](#)
  - delete\_view, [1918](#)
  - F\_auto\_refresh, [1915](#)
  - F\_dynamic\_buffers, [1915](#)
  - F\_dynamic\_views, [1915](#)
  - F\_pointer, [1915](#)
  - Flags, [1915](#)
  - get\_static\_buffer, [1919](#)
  - info, [1920](#)
  - view, [1921](#)
- L4Re::Video::Goos::Info, [1922](#)
- L4Re::Video::Pixel\_info, [1924](#)
  - a, [1928](#)
  - b, [1928](#), [1929](#)
  - bits\_per\_pixel, [1929](#)
  - bytes\_per\_pixel, [1930](#)
  - dump, [1931](#)
  - g, [1931](#), [1932](#)
  - has\_alpha, [1932](#)
  - operator==, [1932](#)
  - padding, [1933](#)
  - Pixel\_info, [1926](#), [1927](#)
  - r, [1933](#), [1934](#)
- L4Re::Video::View, [1935](#)
  - F\_above, [1937](#)
  - F\_dyn\_allocated, [1936](#)
  - F\_flags\_mask, [1937](#)
  - F\_fully\_dynamic, [1937](#)
  - F\_none, [1936](#)
  - F\_set\_background, [1936](#)
  - F\_set\_buffer, [1936](#)
  - F\_set\_buffer\_offset, [1936](#)
  - F\_set\_bytes\_per\_line, [1936](#)
  - F\_set\_flags, [1937](#)
  - F\_set\_pixel, [1936](#)
  - F\_set\_position, [1936](#)
  - Flags, [1936](#)
  - info, [1937](#)

- refresh, [1938](#)
- set\_info, [1938](#)
- set\_viewport, [1939](#)
- stack, [1940](#)
- V\_flags, [1937](#)
- L4Re::Video::View::Info, [1941](#)
- l4re\_aux\_t, [1944](#)
- l4re\_debug\_obj\_debug
  - Debug interface, [540](#)
- l4re\_dma\_space\_associate
  - DMA Space Interface, [541](#)
- L4RE\_DMA\_SPACE\_BIDIRECTIONAL
  - dma\_space.h, [2844](#)
- L4RE\_DMA\_SPACE\_COHERENT
  - dma\_space.h, [2845](#)
- l4re\_dma\_space\_direction
  - dma\_space.h, [2844](#)
- l4re\_dma\_space\_disassociate
  - DMA Space Interface, [542](#)
- L4RE\_DMA\_SPACE\_FROM\_DEVICE
  - dma\_space.h, [2844](#)
- l4re\_dma\_space\_map
  - DMA Space Interface, [542](#)
- L4RE\_DMA\_SPACE\_NONE
  - dma\_space.h, [2844](#)
- L4RE\_DMA\_SPACE\_PHYS\_SPACE
  - dma\_space.h, [2845](#)
- l4re\_dma\_space\_space\_attribs
  - dma\_space.h, [2844](#)
- l4re\_dma\_space\_t
  - DMA Space Interface, [541](#)
- L4RE\_DMA\_SPACE\_TO\_DEVICE
  - dma\_space.h, [2844](#)
- l4re\_dma\_space\_unmap
  - DMA Space Interface, [543](#)
- l4re\_ds\_allocate
  - Dataspace interface, [536](#)
- l4re\_ds\_clear
  - Dataspace interface, [536](#)
- l4re\_ds\_copy\_in
  - Dataspace interface, [537](#)
- L4RE\_DS\_F\_BUFFERABLE
  - Dataspace interface, [535](#)
- L4RE\_DS\_F\_CACHEABLE
  - Dataspace interface, [535](#)
- L4RE\_DS\_F\_CACHING\_MASK
  - Dataspace interface, [535](#)
- L4RE\_DS\_F\_CACHING\_SHIFT
  - Dataspace interface, [535](#)
- L4RE\_DS\_F\_NORMAL
  - Dataspace interface, [535](#)
- L4RE\_DS\_F\_UNCACHEABLE
  - Dataspace interface, [535](#)
- l4re\_ds\_flags
  - Dataspace interface, [537](#)
- l4re\_ds\_info
  - Dataspace interface, [538](#)
- l4re\_ds\_map\_flags
  - Dataspace interface, [535](#)
- l4re\_ds\_map\_info
  - Dataspace interface, [538](#)
- l4re\_ds\_size
  - Dataspace interface, [539](#)
- l4re\_ds\_stats\_t, [1945](#)
- L4RE\_ELF\_AUX\_ELEM
  - L4Re ELF Auxiliary Information, [598](#)
- L4RE\_ELF\_AUX\_ELEM\_T
  - L4Re ELF Auxiliary Information, [598](#)
- l4re\_elf\_aux\_mword\_t, [1945](#)
- l4re\_elf\_aux\_t, [1946](#)
- L4RE\_ELF\_AUX\_T\_EX\_REGS\_FLAGS
  - L4Re ELF Auxiliary Information, [599](#)
- L4RE\_ELF\_AUX\_T\_KIP\_ADDR
  - L4Re ELF Auxiliary Information, [599](#)
- L4RE\_ELF\_AUX\_T\_NONE
  - L4Re ELF Auxiliary Information, [599](#)
- L4RE\_ELF\_AUX\_T\_STACK\_ADDR
  - L4Re ELF Auxiliary Information, [599](#)
- L4RE\_ELF\_AUX\_T\_STACK\_SIZE
  - L4Re ELF Auxiliary Information, [599](#)
- L4RE\_ELF\_AUX\_T\_VMA
  - L4Re ELF Auxiliary Information, [599](#)
- l4re\_elf\_aux\_vma\_t, [1946](#)
- l4re\_env
  - Initial Environment, [585](#)
- l4re\_env\_cap\_entry\_t, [1947](#)
  - flags, [1948](#)
  - l4re\_env\_cap\_entry\_t, [1948](#)
- l4re\_env\_get\_cap
  - Initial Environment, [586](#)
- l4re\_env\_get\_cap\_e
  - Initial Environment, [586](#)
- l4re\_env\_get\_cap\_l
  - Initial Environment, [587](#)
- l4re\_env\_t, [1949](#)
  - caps, [1950](#)
  - env.h, [2891](#)
- l4re\_event\_get\_axis\_info
  - Event interface, [545](#)
- l4re\_event\_get\_buffer
  - Event interface, [545](#)
- l4re\_event\_get\_num\_streams
  - Event interface, [546](#)
- l4re\_event\_get\_stream\_info
  - Event interface, [546](#)
- l4re\_event\_get\_stream\_info\_for\_id
  - Event interface, [547](#)
- l4re\_event\_t, [1951](#)
- l4re\_inhibitor\_acquire
  - inhibitor.h, [2851](#)
- l4re\_inhibitor\_next\_lock\_info
  - inhibitor.h, [2851](#)
- l4re\_inhibitor\_release
  - inhibitor.h, [2852](#)
- l4re\_kip
  - Initial Environment, [588](#)

- l4re\_log\_print
  - Log interface, [548](#)
- l4re\_log\_print\_srv
  - Log interface, [548](#)
- l4re\_log\_printn
  - Log interface, [549](#)
- l4re\_log\_printn\_srv
  - Log interface, [549](#)
- l4re\_ma\_alloc
  - Memory allocator, [551](#)
- l4re\_ma\_alloc\_align
  - Memory allocator, [552](#)
- l4re\_ma\_alloc\_align\_srv
  - Memory allocator, [554](#)
- l4re\_ma\_flags
  - Memory allocator, [551](#)
- l4re\_ns\_query\_srv
  - Namespace interface, [557](#)
- l4re\_ns\_query\_to\_srv
  - Namespace interface, [558](#)
- l4re\_ns\_register\_flags
  - Namespace interface, [557](#)
- l4re\_ns\_register\_obj\_srv
  - Namespace interface, [559](#)
- l4re\_parent\_signal
  - parent.h, [2859](#)
- L4RE\_PROTO\_DATASPACE
  - L4Re Protocol identifiers, [605](#)
- L4RE\_PROTO\_DEBUG
  - L4Re Protocol identifiers, [605](#)
- L4RE\_PROTO\_DMA\_SPACE
  - L4Re Protocol identifiers, [605](#)
- L4RE\_PROTO\_EVENT
  - L4Re Protocol identifiers, [605](#)
- L4RE\_PROTO\_GOOS
  - L4Re Protocol identifiers, [605](#)
- L4RE\_PROTO\_INHIBITOR
  - L4Re Protocol identifiers, [605](#)
- L4RE\_PROTO\_ITAS
  - L4Re Protocol identifiers, [605](#)
- L4RE\_PROTO\_MEM\_ALLOC
  - L4Re Protocol identifiers, [605](#)
- L4RE\_PROTO\_MMIO\_SPACE
  - L4Re Protocol identifiers, [605](#)
- L4RE\_PROTO\_NAMESPACE
  - L4Re Protocol identifiers, [605](#)
- L4RE\_PROTO\_PARENT
  - L4Re Protocol identifiers, [605](#)
- L4RE\_PROTO\_REMOTE\_ACCESS
  - L4Re Protocol identifiers, [605](#)
- L4RE\_PROTO\_RM
  - L4Re Protocol identifiers, [605](#)
- L4RE\_PROTO\_RSVD\_1
  - L4Re Protocol identifiers, [605](#)
- l4re\_protocols
  - L4Re Protocol identifiers, [604](#)
- l4re\_rm\_attach
  - Region map interface, [562](#)
- l4re\_rm\_attach\_srv
  - Region map interface, [563](#)
- L4RE\_RM\_CACHING\_SHIFT
  - Region map interface, [562](#)
- l4re\_rm\_detach
  - Region map interface, [564](#)
- l4re\_rm\_detach\_ds
  - Region map interface, [565](#)
- l4re\_rm\_detach\_ds\_unmap
  - Region map interface, [566](#)
- l4re\_rm\_detach\_srv
  - Region map interface, [567](#)
- l4re\_rm\_detach\_unmap
  - Region map interface, [567](#)
- L4RE\_RM\_F\_ATTACH\_FLAGS
  - Region map interface, [562](#)
- L4RE\_RM\_F\_CACHE\_BUFFERED
  - Region map interface, [562](#)
- L4RE\_RM\_F\_CACHE\_NORMAL
  - Region map interface, [562](#)
- L4RE\_RM\_F\_CACHE\_UNCACHED
  - Region map interface, [562](#)
- L4RE\_RM\_F\_CACHING
  - Region map interface, [562](#)
- L4RE\_RM\_F\_DETACH\_FREE
  - Region map interface, [562](#)
- L4RE\_RM\_F\_EAGER\_MAP
  - Region map interface, [562](#)
- L4RE\_RM\_F\_IN\_AREA
  - Region map interface, [562](#)
- L4RE\_RM\_F\_KERNEL
  - Region map interface, [562](#)
- L4RE\_RM\_F\_NO\_EAGER\_MAP
  - Region map interface, [562](#)
- L4RE\_RM\_F\_PAGER
  - Region map interface, [562](#)
- L4RE\_RM\_F\_R
  - Region map interface, [562](#)
- L4RE\_RM\_F\_RESERVED
  - Region map interface, [562](#)
- L4RE\_RM\_F\_SEARCH\_ADDR
  - Region map interface, [562](#)
- l4re\_rm\_find
  - Region map interface, [568](#)
- l4re\_rm\_find\_srv
  - Region map interface, [569](#)
- l4re\_rm\_flags\_values
  - Region map interface, [561](#)
- l4re\_rm\_free\_area
  - Region map interface, [570](#)
- l4re\_rm\_free\_area\_srv
  - Region map interface, [571](#)
- l4re\_rm\_get\_info
  - Region map interface, [571](#)
- l4re\_rm\_get\_info\_srv
  - Region map interface, [572](#)
- L4RE\_RM\_REGION\_FLAGS
  - Region map interface, [562](#)

- l4re\_rm\_reserve\_area
  - Region map interface, [573](#)
- l4re\_rm\_reserve\_area\_srv
  - Region map interface, [574](#)
- l4re\_rm\_show\_lists
  - Region map interface, [575](#)
- l4re\_util\_cap\_last
  - Capability allocator, [576](#)
- l4re\_util\_kumem\_alloc
  - kumem\_alloc.h, [2867](#)
- l4re\_video\_color\_component\_t, [1952](#)
- l4re\_video\_goos\_create\_buffer
  - Video API, [579](#)
- l4re\_video\_goos\_create\_view
  - Video API, [580](#)
- l4re\_video\_goos\_delete\_buffer
  - Video API, [580](#)
- l4re\_video\_goos\_delete\_view
  - Video API, [580](#)
- l4re\_video\_goos\_get\_static\_buffer
  - Video API, [581](#)
- l4re\_video\_goos\_get\_view
  - Video API, [581](#)
- l4re\_video\_goos\_info
  - Video API, [581](#)
- l4re\_video\_goos\_info\_flags\_t
  - Video API, [578](#)
- l4re\_video\_goos\_info\_t, [1952](#)
- l4re\_video\_goos\_refresh
  - Video API, [582](#)
- l4re\_video\_pixel\_info\_t, [1954](#)
- l4re\_video\_view\_get\_info
  - Video API, [582](#)
- l4re\_video\_view\_info\_flags\_t
  - Video API, [579](#)
- l4re\_video\_view\_info\_t, [1955](#)
- l4re\_video\_view\_refresh
  - Video API, [582](#)
- l4re\_video\_view\_set\_info
  - Video API, [583](#)
- l4re\_video\_view\_set\_viewport
  - Video API, [583](#)
- l4re\_video\_view\_stack
  - Video API, [584](#)
- l4re\_video\_view\_t, [1956](#)
  - Video API, [578](#)
- L4SHM-based ring buffer implementation, [609](#)
- l4shmc\_add\_chunk
  - Chunks, [618](#)
- l4shmc\_add\_signal
  - Signals, [631](#)
- l4shmc\_area\_overhead
  - Shared Memory Library, [613](#)
- l4shmc\_area\_size
  - Shared Memory Library, [613](#)
- l4shmc\_area\_size\_free
  - Shared Memory Library, [614](#)
- l4shmc\_attach
  - Shared Memory Library, [614](#)
- l4shmc\_attach\_signal
  - Signals, [631](#)
- l4shmc\_check\_magic
  - Signals, [632](#)
- l4shmc\_chunk\_capacity
  - Chunks, [619](#)
- l4shmc\_chunk\_consumed
  - Consumer, [625](#)
- l4shmc\_chunk\_overhead
  - Shared Memory Library, [615](#)
- l4shmc\_chunk\_ptr
  - Chunks, [619](#)
- l4shmc\_chunk\_ready
  - Producer, [622](#)
- l4shmc\_chunk\_ready\_sig
  - Producer, [622](#)
- l4shmc\_chunk\_signal
  - Chunks, [619](#)
- l4shmc\_chunk\_size
  - Consumer, [626](#)
- l4shmc\_chunk\_try\_to\_take
  - Producer, [623](#)
- l4shmc\_chunk\_try\_to\_take\_for\_overwriting
  - Producer, [623](#)
- l4shmc\_chunk\_try\_to\_take\_for\_reading
  - Consumer, [626](#)
- l4shmc\_chunk\_try\_to\_take\_for\_writing
  - Producer, [624](#)
- l4shmc\_connect\_chunk\_signal
  - Shared Memory Library, [615](#)
- l4shmc\_create
  - Shared Memory Library, [615](#)
- l4shmc\_enable\_chunk
  - Consumer, [627](#)
- l4shmc\_enable\_signal
  - Consumer, [635](#)
- l4shmc\_get\_chunk
  - Chunks, [620](#)
- l4shmc\_get\_chunk\_to
  - Chunks, [620](#)
- l4shmc\_get\_client\_nr
  - Shared Memory Library, [616](#)
- l4shmc\_get\_initialized\_clients
  - Shared Memory Library, [616](#)
- l4shmc\_get\_signal
  - Signals, [632](#)
- l4shmc\_is\_chunk\_clear
  - Producer, [624](#)
- l4shmc\_is\_chunk\_ready
  - Consumer, [627](#)
- l4shmc\_iterate\_chunk
  - Chunks, [621](#)
- l4shmc\_mark\_client\_initialized
  - Shared Memory Library, [617](#)
- l4shmc\_rb\_attach\_receiver
  - ringbuf.h, [3018](#)
- l4shmc\_rb\_attach\_sender

- ringbuf.h, [3018](#)
- l4shmc\_rb\_deinit\_buffer
  - ringbuf.h, [3019](#)
- l4shmc\_rb\_init\_buffer
  - ringbuf.h, [3019](#)
- l4shmc\_rb\_init\_receiver
  - ringbuf.h, [3020](#)
- l4shmc\_rb\_receiver\_copy\_out
  - ringbuf.h, [3020](#)
- l4shmc\_rb\_receiver\_notify\_done
  - ringbuf.h, [3021](#)
- l4shmc\_rb\_receiver\_read\_next\_size
  - ringbuf.h, [3021](#)
- l4shmc\_rb\_receiver\_wait\_for\_data
  - ringbuf.h, [3021](#)
- l4shmc\_rb\_sender\_alloc\_packet
  - ringbuf.h, [3022](#)
- l4shmc\_rb\_sender\_commit\_packet
  - ringbuf.h, [3022](#)
- l4shmc\_rb\_sender\_next\_copy\_in
  - ringbuf.h, [3023](#)
- l4shmc\_rb\_sender\_put\_data
  - ringbuf.h, [3023](#)
- L4SHMC\_RINGBUF\_DATA
  - Internal, [611](#)
- L4SHMC\_RINGBUF\_DATA\_SIZE
  - Internal, [611](#)
- L4SHMC\_RINGBUF\_HEAD
  - Internal, [611](#)
- l4shmc\_ringbuf\_head\_t, [1957](#)
- l4shmc\_ringbuf\_t, [1958](#)
- l4shmc\_signal\_cap
  - Signals, [633](#)
- l4shmc\_trigger
  - Producer, [633](#)
- l4shmc\_wait\_any
  - Consumer, [636](#)
- l4shmc\_wait\_any\_to
  - Consumer, [636](#)
- l4shmc\_wait\_any\_try
  - Consumer, [637](#)
- l4shmc\_wait\_chunk
  - Consumer, [627](#)
- l4shmc\_wait\_chunk\_to
  - Consumer, [629](#)
- l4shmc\_wait\_chunk\_try
  - Consumer, [629](#)
- l4shmc\_wait\_signal
  - Consumer, [637](#)
- l4shmc\_wait\_signal\_to
  - Consumer, [637](#)
- l4shmc\_wait\_signal\_try
  - Consumer, [638](#)
- l4sigma0\_debug\_dump
  - Sigma0 API, [640](#)
- L4SIGMA0\_IPCERROR
  - Sigma0 API, [640](#)
- l4sigma0\_map\_anypage
  - Sigma0 API, [640](#)
- l4sigma0\_map\_errstr
  - Sigma0 API, [641](#)
- l4sigma0\_map\_iomem
  - Sigma0 API, [641](#)
- l4sigma0\_map\_kip
  - Sigma0 API, [642](#)
- l4sigma0\_map\_mem
  - Sigma0 API, [642](#)
- L4SIGMA0\_NOFPAGE
  - Sigma0 API, [640](#)
- L4SIGMA0\_NOTALIGNED
  - Sigma0 API, [640](#)
- L4SIGMA0\_OK
  - Sigma0 API, [640](#)
- l4sigma0\_return\_flags\_t
  - Sigma0 API, [640](#)
- L4SIGMA0\_SMALLERFPAGE
  - Sigma0 API, [640](#)
- l4util\_add16
  - Atomic Instructions, [669](#)
- l4util\_add16\_res
  - Atomic Instructions, [669](#)
- l4util\_add32
  - Atomic Instructions, [669](#)
- l4util\_add32\_res
  - Atomic Instructions, [670](#)
- l4util\_add8
  - Atomic Instructions, [670](#)
- l4util\_add8\_res
  - Atomic Instructions, [670](#)
- l4util\_and16
  - Atomic Instructions, [671](#)
- l4util\_and16\_res
  - Atomic Instructions, [671](#)
- l4util\_and32
  - Atomic Instructions, [671](#)
- l4util\_and32\_res
  - Atomic Instructions, [672](#)
- l4util\_and8
  - Atomic Instructions, [672](#)
- l4util\_and8\_res
  - Atomic Instructions, [672](#)
- l4util\_atomic\_add
  - Atomic Instructions, [673](#)
- l4util\_atomic\_inc
  - Atomic Instructions, [673](#)
- l4util\_backtrace
  - backtrace.h, [3334](#)
- l4util\_bsf
  - Bit Manipulation, [686](#)
- l4util\_bsr
  - Bit Manipulation, [687](#)
- l4util\_btc
  - Bit Manipulation, [687](#)
- l4util\_btr
  - Bit Manipulation, [687](#)
- l4util\_bts



- Bit Manipulation, [689](#)
- `l4util_clear_bit`
  - Bit Manipulation, [690](#)
- `l4util_cmpxchg`
  - Atomic Instructions, [673](#)
- `l4util_cmpxchg16`
  - Atomic Instructions, [674](#)
- `l4util_cmpxchg32`
  - Atomic Instructions, [674](#)
- `l4util_cmpxchg8`
  - Atomic Instructions, [675](#)
- `l4util_complement_bit`
  - Bit Manipulation, [690](#)
- `l4util_cpu_capabilities`
  - CPU related functions, [658](#)
- `l4util_cpu_capabilities_nocheck`
  - CPU related functions, [659](#)
- `l4util_cpu_has_cpuid`
  - CPU related functions, [659](#)
- `l4util_dec16`
  - Atomic Instructions, [675](#)
- `l4util_dec16_res`
  - Atomic Instructions, [676](#)
- `l4util_dec32`
  - Atomic Instructions, [676](#)
- `l4util_dec32_res`
  - Atomic Instructions, [676](#)
- `l4util_dec8`
  - Atomic Instructions, [676](#)
- `l4util_dec8_res`
  - Atomic Instructions, [677](#)
- `l4util_find_first_set_bit`
  - Bit Manipulation, [691](#)
- `l4util_find_first_zero_bit`
  - Bit Manipulation, [691](#)
- `l4util_in16`
  - IA32 Port I/O API, [726](#)
- `l4util_in32`
  - IA32 Port I/O API, [726](#)
- `l4util_in8`
  - IA32 Port I/O API, [726](#)
- `l4util_inc16`
  - Atomic Instructions, [677](#)
- `l4util_inc16_res`
  - Atomic Instructions, [677](#)
- `l4util_inc32`
  - Atomic Instructions, [677](#)
- `l4util_inc32_res`
  - Atomic Instructions, [678](#)
- `l4util_inc8`
  - Atomic Instructions, [678](#)
- `l4util_inc8_res`
  - Atomic Instructions, [678](#)
- `l4util_ins16`
  - IA32 Port I/O API, [727](#)
- `l4util_ins32`
  - IA32 Port I/O API, [727](#)
- `l4util_ins8`
  - IA32 Port I/O API, [727](#)
- `l4util_ioport_map`
  - IA32 Port I/O API, [728](#)
- `l4util_kip_for_each_feature`
  - Kernel Interface Page API, [719](#)
- `l4util_kip_kernel_abi_version`
  - Kernel Interface Page API, [720](#)
- `l4util_kip_kernel_has_feature`
  - Kernel Interface Page API, [721](#)
- `l4util_l4mod_info`, [1959](#)
  - `vbe_ctrl_info`, [1960](#)
- `l4util_l4mod_mod`, [1960](#)
- `L4util_l4mod_mod_flag_kernel`
  - `l4mod.h`, [3370](#)
- `L4util_l4mod_mod_flag_mask`
  - `l4mod.h`, [3370](#)
- `L4util_l4mod_mod_flag_roottask`
  - `l4mod.h`, [3370](#)
- `L4util_l4mod_mod_flag_sigma0`
  - `l4mod.h`, [3370](#)
- `L4util_l4mod_mod_flag_unspec`
  - `l4mod.h`, [3370](#)
- `l4util_l4mod_mod_info_flag`
  - `l4mod.h`, [3370](#)
- `l4util_mb_addr_range_t`, [1961](#)
- `l4util_mb_apm_t`, [1962](#)
- `l4util_mb_drive_t`, [1963](#)
- `l4util_mb_for_each_mmap_entry`
  - `mb_info.h`, [3378](#)
- `l4util_mb_info_t`, [1964](#)
- `L4UTIL_MB_MEMORY`
  - `mb_info.h`, [3378](#)
- `l4util_mb_mod_t`, [1965](#)
- `l4util_mb_vbe_ctrl_t`, [1966](#)
- `l4util_mb_vbe_mode_t`, [1967](#)
- `l4util_micros2l4to`
  - Utility Functions, [655](#)
- `l4util_next_power2`
  - Bit Manipulation, [691](#)
- `l4util_or16`
  - Atomic Instructions, [678](#)
- `l4util_or16_res`
  - Atomic Instructions, [679](#)
- `l4util_or32`
  - Atomic Instructions, [679](#)
- `l4util_or32_res`
  - Atomic Instructions, [679](#)
- `l4util_or8`
  - Atomic Instructions, [680](#)
- `l4util_or8_res`
  - Atomic Instructions, [680](#)
- `l4util_out16`
  - IA32 Port I/O API, [729](#)
- `l4util_out32`
  - IA32 Port I/O API, [729](#)
- `l4util_out8`
  - IA32 Port I/O API, [730](#)
- `l4util_outs16`



- IA32 Port I/O API, [730](#)
- l4util\_outs32
  - IA32 Port I/O API, [730](#)
- l4util\_outs8
  - IA32 Port I/O API, [731](#)
- l4util\_rand
  - Random number support, [724](#)
- l4util\_set\_bit
  - Bit Manipulation, [692](#)
- l4util\_splitlog2\_hdl
  - Utility Functions, [655](#)
- l4util\_splitlog2\_size
  - Utility Functions, [656](#)
- l4util\_srand
  - Random number support, [724](#)
- l4util\_sub16
  - Atomic Instructions, [680](#)
- l4util\_sub16\_res
  - Atomic Instructions, [681](#)
- l4util\_sub32
  - Atomic Instructions, [681](#)
- l4util\_sub32\_res
  - Atomic Instructions, [681](#)
- l4util\_sub8
  - Atomic Instructions, [682](#)
- l4util\_sub8\_res
  - Atomic Instructions, [682](#)
- l4util\_test\_bit
  - Bit Manipulation, [692](#)
- l4util\_xchg
  - Atomic Instructions, [682](#)
- l4util\_xchg16
  - Atomic Instructions, [683](#)
- l4util\_xchg32
  - Atomic Instructions, [683](#)
- l4util\_xchg8
  - Atomic Instructions, [683](#)
- L4vbus, [804](#)
- L4vbus GPIO functions, [513](#)
  - l4vbus\_gpio\_config\_get, [514](#)
  - l4vbus\_gpio\_config\_pad, [515](#)
  - l4vbus\_gpio\_config\_pull, [516](#)
  - L4vbus\_gpio\_generic\_func, [514](#)
  - l4vbus\_gpio\_get, [516](#)
  - l4vbus\_gpio\_multi\_config\_pad, [517](#)
  - l4vbus\_gpio\_multi\_get, [518](#)
  - l4vbus\_gpio\_multi\_set, [519](#)
  - l4vbus\_gpio\_multi\_setup, [520](#)
  - L4VBUS\_GPIO\_PIN\_PULL\_DOWN, [514](#)
  - L4VBUS\_GPIO\_PIN\_PULL\_NONE, [514](#)
  - L4VBUS\_GPIO\_PIN\_PULL\_UP, [514](#)
  - L4vbus\_gpio\_pull\_modes, [514](#)
  - l4vbus\_gpio\_set, [520](#)
  - l4vbus\_gpio\_setup, [521](#)
  - L4VBUS\_GPIO\_SETUP\_INPUT, [514](#)
  - L4VBUS\_GPIO\_SETUP\_IRQ, [514](#)
  - L4VBUS\_GPIO\_SETUP\_OUTPUT, [514](#)
  - l4vbus\_gpio\_to\_irq, [522](#)
- L4vbus PCI functions, [523](#)
  - l4vbus\_pci\_cfg\_read, [524](#)
  - l4vbus\_pci\_cfg\_write, [524](#)
  - l4vbus\_pci\_irq\_enable, [525](#)
  - l4vbus\_pciddev\_cfg\_read, [527](#)
  - l4vbus\_pciddev\_cfg\_write, [528](#)
  - l4vbus\_pciddev\_irq\_enable, [528](#)
- L4vbus power management functions, [529](#)
  - l4vbus\_pm\_resume, [530](#)
  - l4vbus\_pm\_suspend, [530](#)
- L4vbus::Device, [1971](#)
  - bus\_cap, [1974](#)
  - dev\_handle, [1974](#)
  - Device, [1973](#)
  - device, [1975](#)
  - device\_by\_hid, [1975](#)
  - get\_resource, [1976](#)
  - is\_compatible, [1977](#)
  - next\_device, [1978](#)
  - operator!=, [1979](#)
  - operator==, [1979](#)
- L4vbus::Gpio\_module, [1980](#)
  - config\_pad, [1984](#)
  - get, [1984](#)
  - pin, [1985](#)
  - set, [1986](#)
  - setup, [1987](#)
- L4vbus::Gpio\_module::Pin\_slice, [1988](#)
- L4vbus::Gpio\_pin, [1988](#)
  - config\_get, [1992](#)
  - config\_pad, [1992](#)
  - config\_pull, [1994](#)
  - get, [1994](#)
  - pin, [1995](#)
  - set, [1995](#)
  - setup, [1996](#)
  - to\_irq, [1996](#)
- L4vbus::Icu, [1997](#)
  - Src\_dev\_handle, [2001](#)
  - Src\_types, [2001](#)
  - vicu, [2001](#)
- L4vbus::Pci\_dev, [2002](#)
  - cfg\_read, [2006](#)
  - cfg\_write, [2006](#)
  - irq\_enable, [2007](#)
- L4vbus::Pci\_host\_bridge, [2008](#)
  - cfg\_read, [2012](#)
  - cfg\_write, [2012](#)
  - irq\_enable, [2013](#)
- L4vbus::Pm< DEC >, [2014](#)
  - pm\_resume, [2016](#)
  - pm\_suspend, [2016](#)
- L4vbus::Vbus, [2017](#)
  - assign\_dma\_domain, [2024](#), [2025](#)
  - release\_ioport, [2026](#)
  - request\_ioport, [2026](#)
  - root, [2027](#)
- l4vbus\_assign\_dma\_domain

- L4 Vbus functions, [504](#)
- L4VBUS\_DEVICE\_F\_CHILDREN
  - vbus\_types.h, [3420](#)
- l4vbus\_device\_flags\_t
  - vbus\_types.h, [3420](#)
- l4vbus\_device\_t, [2028](#)
- L4vbus\_dma\_domain\_assign\_flags
  - L4 Vbus functions, [503](#)
- L4VBUS\_DMAD\_BIND
  - L4 Vbus functions, [503](#)
- L4VBUS\_DMAD\_KERNEL\_DMA\_SPACE
  - L4 Vbus functions, [503](#)
- L4VBUS\_DMAD\_L4RE\_DMA\_SPACE
  - L4 Vbus functions, [503](#)
- L4VBUS\_DMAD\_UNBIND
  - L4 Vbus functions, [503](#)
- l4vbus\_get\_adr
  - L4 Vbus functions, [504](#)
- l4vbus\_get\_device
  - L4 Vbus functions, [505](#)
- l4vbus\_get\_device\_by\_hid
  - L4 Vbus functions, [506](#)
- l4vbus\_get\_hid
  - L4 Vbus functions, [507](#)
- l4vbus\_get\_next\_device
  - L4 Vbus functions, [507](#)
- l4vbus\_get\_resource
  - L4 Vbus functions, [509](#)
- l4vbus\_gpio\_config\_get
  - L4vbus GPIO functions, [514](#)
- l4vbus\_gpio\_config\_pad
  - L4vbus GPIO functions, [515](#)
- l4vbus\_gpio\_config\_pull
  - L4vbus GPIO functions, [516](#)
- L4vbus\_gpio\_generic\_func
  - L4vbus GPIO functions, [514](#)
- l4vbus\_gpio\_get
  - L4vbus GPIO functions, [516](#)
- l4vbus\_gpio\_multi\_config\_pad
  - L4vbus GPIO functions, [517](#)
- l4vbus\_gpio\_multi\_get
  - L4vbus GPIO functions, [518](#)
- l4vbus\_gpio\_multi\_set
  - L4vbus GPIO functions, [519](#)
- l4vbus\_gpio\_multi\_setup
  - L4vbus GPIO functions, [520](#)
- L4VBUS\_GPIO\_PIN\_PULL\_DOWN
  - L4vbus GPIO functions, [514](#)
- L4VBUS\_GPIO\_PIN\_PULL\_NONE
  - L4vbus GPIO functions, [514](#)
- L4VBUS\_GPIO\_PIN\_PULL\_UP
  - L4vbus GPIO functions, [514](#)
- L4vbus\_gpio\_pull\_modes
  - L4vbus GPIO functions, [514](#)
- l4vbus\_gpio\_set
  - L4vbus GPIO functions, [520](#)
- l4vbus\_gpio\_setup
  - L4vbus GPIO functions, [521](#)
- L4VBUS\_GPIO\_SETUP\_INPUT
  - L4vbus GPIO functions, [514](#)
- L4VBUS\_GPIO\_SETUP\_IRQ
  - L4vbus GPIO functions, [514](#)
- L4VBUS\_GPIO\_SETUP\_OUTPUT
  - L4vbus GPIO functions, [514](#)
- l4vbus\_gpio\_to\_irq
  - L4vbus GPIO functions, [522](#)
- L4VBUS\_ICU\_SRC\_DEV\_HANDLE
  - vbus.h, [3407](#)
- l4vbus\_icu\_src\_types
  - vbus.h, [3407](#)
- L4VBUS\_IFACE\_SHIFT
  - vbus\_interfaces.h, [3414](#)
- l4vbus\_iface\_type\_t
  - vbus\_interfaces.h, [3413](#), [3414](#)
- L4VBUS\_INTERFACE\_BUS
  - vbus\_interfaces.h, [3414](#)
- L4VBUS\_INTERFACE\_GENERIC
  - vbus\_interfaces.h, [3414](#)
- L4VBUS\_INTERFACE\_GPIO
  - vbus\_interfaces.h, [3414](#)
- L4VBUS\_INTERFACE\_ICU
  - vbus\_interfaces.h, [3414](#)
- L4VBUS\_INTERFACE\_PCI
  - vbus\_interfaces.h, [3414](#)
- L4VBUS\_INTERFACE\_PCIDEV
  - vbus\_interfaces.h, [3414](#)
- L4VBUS\_INTERFACE\_PM
  - vbus\_interfaces.h, [3414](#)
- l4vbus\_is\_compatible
  - L4 Vbus functions, [510](#)
- L4VBUS\_NULL
  - vbus.h, [3407](#)
- l4vbus\_pci\_cfg\_read
  - L4vbus PCI functions, [524](#)
- l4vbus\_pci\_cfg\_write
  - L4vbus PCI functions, [524](#)
- l4vbus\_pci\_irq\_enable
  - L4vbus PCI functions, [525](#)
- l4vbus\_pciddev\_cfg\_read
  - L4vbus PCI functions, [527](#)
- l4vbus\_pciddev\_cfg\_write
  - L4vbus PCI functions, [528](#)
- l4vbus\_pciddev\_irq\_enable
  - L4vbus PCI functions, [528](#)
- l4vbus\_pm\_resume
  - L4vbus power management functions, [530](#)
- l4vbus\_pm\_suspend
  - L4vbus power management functions, [530](#)
- l4vbus\_release\_ioport
  - L4 Vbus functions, [510](#)
- l4vbus\_request\_ioport
  - L4 Vbus functions, [511](#)
- L4VBUS\_RESOURCE\_BUS
  - vbus\_types.h, [3421](#)
- L4VBUS\_RESOURCE\_DMA\_DOMAIN
  - vbus\_types.h, [3421](#)

- L4VBUS\_RESOURCE\_F\_MEM\_CACHEABLE
  - vbus\_types.h, [3421](#)
- L4VBUS\_RESOURCE\_F\_MEM\_MMIO\_READ
  - vbus\_types.h, [3421](#)
- L4VBUS\_RESOURCE\_F\_MEM\_MMIO\_WRITE
  - vbus\_types.h, [3421](#)
- L4VBUS\_RESOURCE\_F\_MEM\_PREFETCHABLE
  - vbus\_types.h, [3421](#)
- L4VBUS\_RESOURCE\_F\_MEM\_R
  - vbus\_types.h, [3420](#)
- L4VBUS\_RESOURCE\_F\_MEM\_W
  - vbus\_types.h, [3421](#)
- l4vbus\_resource\_flags\_t
  - vbus\_types.h, [3420](#)
- L4VBUS\_RESOURCE\_GPIO
  - vbus\_types.h, [3421](#)
- L4VBUS\_RESOURCE\_INVALID
  - vbus\_types.h, [3421](#)
- L4VBUS\_RESOURCE\_IRQ
  - vbus\_types.h, [3421](#)
- L4VBUS\_RESOURCE\_MAX
  - vbus\_types.h, [3421](#)
- L4VBUS\_RESOURCE\_MEM
  - vbus\_types.h, [3421](#)
- L4VBUS\_RESOURCE\_PORT
  - vbus\_types.h, [3421](#)
- l4vbus\_resource\_t, [2029](#)
- l4vbus\_resource\_type\_t
  - vbus\_types.h, [3421](#)
- L4VBUS\_ROOT\_BUS
  - vbus.h, [3407](#)
- l4vbus\_subinterface\_supported
  - vbus\_interfaces.h, [3414](#)
- l4vbus\_vicu\_get\_cap
  - L4 Vbus functions, [512](#)
- L4vcpu::State, [2030](#)
  - add, [2031](#)
  - clear, [2031](#)
  - set, [2031](#)
  - State, [2030](#)
- L4vcpu::Vcpu, [2032](#)
  - cast, [2035](#)
  - entry\_ip, [2036](#)
  - entry\_sp, [2036](#)
  - ext\_alloc, [2038](#)
  - i, [2038](#), [2039](#)
  - irq\_disable\_save, [2039](#)
  - irq\_enable, [2039](#)
  - irq\_restore, [2040](#)
  - is\_irq\_entry, [2040](#)
  - is\_page\_fault\_entry, [2041](#)
  - r, [2041](#), [2042](#)
  - saved\_state, [2042](#)
  - state, [2042](#), [2043](#)
  - task, [2043](#)
  - wait\_for\_event, [2044](#)
- l4vcpu\_ext\_alloc
  - Extended vCPU support, [739](#)
- l4vcpu\_irq\_disable
  - vCPU Support Library, [733](#)
- l4vcpu\_irq\_disable\_save
  - vCPU Support Library, [734](#)
- l4vcpu\_irq\_enable
  - vCPU Support Library, [734](#)
- l4vcpu\_irq\_restore
  - vCPU Support Library, [735](#)
- l4vcpu\_is\_irq\_entry
  - vCPU Support Library, [736](#)
- l4vcpu\_is\_page\_fault\_entry
  - vCPU Support Library, [737](#)
- l4vcpu\_print\_state
  - vCPU Support Library, [737](#)
- l4vcpu\_wait\_for\_event
  - vCPU Support Library, [738](#)
- l4vio\_net\_p2p, a virtual network point-to-point link, [68](#)
- L4virtio, [805](#)
- L4virtio::Device, [2045](#)
  - config\_queue, [2049](#)
  - device\_config, [2050](#)
  - device\_notification\_irq, [2051](#)
  - register\_ds, [2052](#)
  - set\_status, [2053](#)
- L4virtio::Driver::Block\_device, [2054](#)
  - add\_block, [2058](#)
  - process\_request, [2058](#)
  - process\_used\_queue, [2059](#)
  - send\_request, [2060](#)
  - setup\_device, [2060](#)
  - start\_request, [2062](#)
- L4virtio::Driver::Block\_device::Handle, [2063](#)
- L4virtio::Driver::Device, [2063](#)
  - bind\_notification\_irq, [2066](#)
  - config\_queue, [2067](#)
  - driver\_acknowledge, [2067](#)
  - driver\_connect, [2068](#)
  - feature\_negotiated, [2070](#)
  - max\_queue\_size, [2070](#)
  - register\_ds, [2071](#)
  - send, [2072](#)
  - send\_and\_wait, [2073](#)
  - wait, [2074](#)
  - wait\_for\_next\_used, [2075](#)
- L4virtio::Driver::Virtio\_net\_device, [2076](#)
  - bind\_rx\_notification\_irq, [2080](#)
  - finish\_rx, [2080](#)
  - rx\_pkt, [2080](#)
  - rx\_queue\_size, [2081](#)
  - setup\_device, [2081](#)
  - tx, [2083](#)
  - tx\_queue\_size, [2084](#)
  - wait\_rx, [2085](#)
- L4virtio::Driver::Virtio\_net\_device::Packet, [2086](#)
- L4virtio::Driver::Virtqueue, [2086](#)
  - alloc\_descriptor, [2091](#)
  - desc, [2091](#)
  - enqueue\_descriptor, [2091](#)

- find\_next\_used, [2092](#)
- free\_descriptor, [2092](#)
- init\_queue, [2094](#)
- initialize\_rings, [2096](#)
- L4virtio::Ptr< T >, [2097](#)
- get, [2099](#)
- Invalid, [2099](#)
- Invalid\_type, [2099](#)
- is\_valid, [2100](#)
- L4virtio::Svr::Bad\_descriptor, [2101](#)
- Bad\_address, [2102](#)
- Bad\_descriptor, [2102](#)
- Bad\_flags, [2102](#)
- Bad\_next, [2102](#)
- Bad\_rights, [2102](#)
- Bad\_size, [2102](#)
- Error, [2102](#)
- message, [2103](#)
- L4virtio::Svr::Block\_dev\_base< Ds\_data >, [2103](#)
- Block\_dev\_base, [2107](#)
- finalize\_request, [2108](#)
- get\_writeback, [2109](#)
- set\_blk\_size, [2109](#)
- set\_config\_wce, [2109](#)
- set\_discard, [2110](#)
- set\_size\_max, [2110](#)
- set\_topology, [2110](#)
- set\_write\_zeroes, [2111](#)
- L4virtio::Svr::Block\_request< Ds\_data >, [2111](#)
- data\_size, [2112](#)
- next\_block, [2113](#)
- L4virtio::Svr::Console::Control\_message, [2114](#)
- Console\_port, [2115](#)
- Device\_add, [2115](#)
- Device\_ready, [2115](#)
- Device\_remove, [2115](#)
- Events, [2114](#)
- Port\_name, [2115](#)
- Port\_open, [2115](#)
- Port\_ready, [2115](#)
- Resize, [2115](#)
- L4virtio::Svr::Console::Control\_request, [2115](#)
- L4virtio::Svr::Console::Device, [2117](#)
- Device, [2122](#), [2123](#)
- notify\_queue, [2124](#)
- port, [2124](#)
- port\_read, [2125](#)
- port\_write, [2127](#)
- process\_device\_ready, [2129](#)
- process\_port\_open, [2130](#)
- process\_port\_ready, [2130](#)
- L4virtio::Svr::Console::Device\_port, [2131](#)
- L4virtio::Svr::Console::Features, [2135](#)
- console\_multiport\_bfm\_t, [2138](#)
- console\_size\_bfm\_t, [2138](#)
- emerg\_write\_bfm\_t, [2138](#)
- L4virtio::Svr::Console::Port, [2139](#)
- Port\_added, [2142](#)
- Port\_disabled, [2142](#)
- Port\_failed, [2142](#)
- Port\_num\_states, [2142](#)
- Port\_open, [2142](#)
- Port\_ready, [2142](#)
- Port\_status, [2142](#)
- state\_transitions, [2143](#)
- L4virtio::Svr::Console::Port::Transition, [2143](#)
- L4virtio::Svr::Console::Virtio\_con, [2144](#)
- handle\_control\_message, [2149](#)
- notify\_queue, [2150](#)
- port, [2151](#)
- port\_add, [2152](#)
- port\_name, [2153](#)
- port\_open, [2154](#)
- port\_remove, [2155](#)
- process\_device\_ready, [2156](#)
- process\_port\_open, [2157](#)
- process\_port\_ready, [2157](#)
- reset\_device, [2158](#)
- send\_control\_message, [2159](#)
- Virtio\_con, [2148](#)
- L4virtio::Svr::Data\_buffer, [2161](#)
- copy\_to, [2163](#)
- Data\_buffer, [2162](#)
- done, [2163](#)
- set, [2164](#)
- skip, [2164](#)
- L4virtio::Svr::Dev\_config, [2165](#)
- add\_irq\_status, [2169](#)
- change\_queue\_config, [2169](#)
- Dev\_config, [2167](#), [2168](#)
- ds, [2170](#)
- get\_cmd, [2170](#)
- guest\_features, [2170](#)
- hdr, [2171](#)
- negotiated\_features, [2172](#)
- qconfig, [2172](#)
- reset\_cmd, [2173](#)
- reset\_queue, [2173](#)
- set\_device\_needs\_reset, [2174](#)
- set\_status, [2175](#)
- status, [2175](#)
- L4virtio::Svr::Dev\_features, [2176](#)
- L4virtio::Svr::Dev\_status, [2179](#)
- running, [2181](#)
- L4virtio::Svr::Device\_t< DATA >, [2182](#)
- add\_trusted\_dataspaces, [2184](#)
- device\_error, [2185](#)
- device\_notify\_irq, [2185](#)
- handle\_mem\_cmd\_write, [2185](#)
- init\_mem\_info, [2185](#)
- register\_driver\_irq, [2186](#)
- reset\_queue\_config, [2186](#)
- setup\_queue, [2186](#)
- L4virtio::Svr::Driver\_mem\_list\_t< DATA >, [2188](#)
- add, [2190](#)
- find, [2190](#)

- full, [2191](#)
- init, [2192](#)
- load\_desc, [2192](#), [2193](#)
- remove, [2194](#)
- L4virtio::Svr::Driver\_mem\_region\_t< DATA >, [2194](#)
  - contains, [2198](#)
  - Driver\_mem\_region\_t, [2198](#)
  - drv\_base, [2198](#)
  - ds, [2199](#)
  - ds\_offset, [2199](#)
  - empty, [2199](#)
  - flags, [2199](#)
  - is\_writable, [2200](#)
  - local, [2200](#)
  - local\_base, [2200](#)
  - size, [2201](#)
- L4virtio::Svr::Request\_processor, [2201](#)
  - current\_flags, [2203](#)
  - has\_more, [2203](#)
  - next, [2203](#)
  - start, [2205](#), [2207](#)
- L4virtio::Svr::Scmi::Base\_attr\_t, [2208](#)
- L4virtio::Svr::Scmi::Base\_proto, [2209](#)
- L4virtio::Svr::Scmi::Perf\_proto, [2211](#)
- L4virtio::Svr::Scmi::Performance\_attr\_t, [2214](#)
- L4virtio::Svr::Scmi::Performance\_describe\_level\_t, [2215](#)
- L4virtio::Svr::Scmi::Performance\_describe\_levels\_n\_t, [2215](#)
- L4virtio::Svr::Scmi::Performance\_domain\_attr\_t, [2217](#)
- L4virtio::Svr::Scmi::Proto< OBSERV >, [2219](#)
- L4virtio::Svr::Scmi::Scmi\_dev, [2220](#)
- L4virtio::Svr::Scmi::Scmi\_hdr\_t, [2224](#)
- L4virtio::Svr::Virtio\_gpio< Request\_handler, Epiface >, [2225](#)
- L4virtio::Svr::Virtio\_gpio< Request\_handler, Epiface >::Host\_irq, [2230](#)
- L4virtio::Svr::Virtio\_gpio< Request\_handler, Epiface >::Irq\_handler, [2234](#)
- L4virtio::Svr::Virtio\_gpio< Request\_handler, Epiface >::Request\_processor, [2235](#)
  - get\_request, [2237](#)
- L4virtio::Svr::Virtio\_i2c< Request\_handler, Epiface >, [2238](#)
- L4virtio::Svr::Virtio\_i2c< Request\_handler, Epiface >::Host\_irq, [2242](#)
- L4virtio::Svr::Virtio\_i2c< Request\_handler, Epiface >::Request\_processor, [2246](#)
  - get\_request, [2248](#)
- L4virtio::Svr::Virtio\_rng< Rnd\_state, Epiface >, [2249](#)
- L4virtio::Svr::Virtio\_rng< Rnd\_state, Epiface >::Host\_irq, [2253](#)
- L4virtio::Svr::Virtio\_rng< Rnd\_state, Epiface >::Request\_processor, [2257](#)
- L4virtio::Svr::Virtqueue, [2259](#)
  - consumed, [2264](#)
  - desc, [2265](#)
  - desc\_avail, [2266](#)
  - disable\_notify, [2266](#)
  - enable\_notify, [2267](#)
  - finish, [2268](#), [2269](#)
  - next\_avail, [2270](#)
  - rewind\_avail, [2271](#)
- L4virtio::Svr::Virtqueue::Head\_desc, [2272](#)
  - desc, [2272](#)
  - operator bool, [2273](#)
  - valid, [2273](#)
- L4virtio::Virtqueue, [2274](#)
  - avail\_align, [2278](#)
  - avail\_size, [2278](#)
  - desc\_align, [2279](#)
  - desc\_size, [2280](#)
  - disable, [2281](#)
  - dump, [2281](#)
  - get\_avail\_idx, [2282](#)
  - get\_tail\_avail\_idx, [2282](#)
  - no\_notify\_guest, [2283](#)
  - no\_notify\_host, [2283](#), [2284](#)
  - num, [2284](#)
  - ready, [2285](#)
  - setup, [2286](#)
  - setup\_simple, [2287](#)
  - total\_size, [2288](#), [2289](#)
  - used\_align, [2289](#)
  - used\_size, [2290](#)
- L4virtio::Virtqueue::Avail, [2291](#)
- L4virtio::Virtqueue::Avail::Flags, [2293](#)
  - no\_irq\_bfm\_t, [2294](#)
- L4virtio::Virtqueue::Desc, [2294](#)
- L4virtio::Virtqueue::Desc::Flags, [2296](#)
  - indirect\_bfm\_t, [2297](#)
  - next\_bfm\_t, [2297](#)
  - write\_bfm\_t, [2298](#)
- L4virtio::Virtqueue::Used, [2298](#)
- L4virtio::Virtqueue::Used::Flags, [2299](#)
  - no\_notify\_bfm\_t, [2300](#)
- L4virtio::Virtqueue::Used\_elem, [2301](#)
  - Used\_elem, [2301](#)
- I4virtio\_block\_config\_t, [2302](#)
- I4virtio\_block\_discard\_t, [2303](#)
- I4virtio\_block\_header\_t, [2304](#)
- L4virtio\_block\_operations
  - L4 VIRTIO Block Device, [499](#)
- L4VIRTIO\_BLOCK\_S\_IOERR
  - L4 VIRTIO Block Device, [500](#)
- L4VIRTIO\_BLOCK\_S\_OK
  - L4 VIRTIO Block Device, [500](#)
- L4VIRTIO\_BLOCK\_S\_UNSUPP
  - L4 VIRTIO Block Device, [500](#)
- L4virtio\_block\_status
  - L4 VIRTIO Block Device, [500](#)
- L4VIRTIO\_BLOCK\_T\_DISCARD
  - L4 VIRTIO Block Device, [500](#)
- L4VIRTIO\_BLOCK\_T\_FLUSH
  - L4 VIRTIO Block Device, [500](#)
- L4VIRTIO\_BLOCK\_T\_GET\_ID

- L4 VIRTIO Block Device, [500](#)
- L4VIRTIO\_BLOCK\_T\_IN
  - L4 VIRTIO Block Device, [500](#)
- L4VIRTIO\_BLOCK\_T\_OUT
  - L4 VIRTIO Block Device, [500](#)
- L4VIRTIO\_BLOCK\_T\_WRITE\_ZEROES
  - L4 VIRTIO Block Device, [500](#)
- L4VIRTIO\_CMD\_CFG\_CHANGED
  - L4 VIRTIO Transport Layer, [491](#)
- L4VIRTIO\_CMD\_CFG\_QUEUE
  - L4 VIRTIO Transport Layer, [491](#)
- L4VIRTIO\_CMD\_MASK
  - L4 VIRTIO Transport Layer, [491](#)
- L4VIRTIO\_CMD\_NONE
  - L4 VIRTIO Transport Layer, [491](#)
- L4VIRTIO\_CMD\_NOTIFY\_QUEUE
  - L4 VIRTIO Transport Layer, [491](#)
- L4VIRTIO\_CMD\_SET\_STATUS
  - L4 VIRTIO Transport Layer, [491](#)
- l4virtio\_config\_hdr\_t, [2305](#)
- l4virtio\_config\_queue
  - L4 VIRTIO Transport Layer, [494](#)
- l4virtio\_config\_queue\_t, [2306](#)
  - L4 VIRTIO Transport Layer, [491](#)
- l4virtio\_config\_queues
  - L4 VIRTIO Transport Layer, [494](#)
- l4virtio\_device\_config
  - L4 VIRTIO Transport Layer, [495](#)
- l4virtio\_device\_config\_ds
  - L4 VIRTIO Transport Layer, [495](#)
- L4virtio\_device\_ids
  - L4 VIRTIO Transport Layer, [492](#)
- l4virtio\_device\_notification\_irq
  - L4 VIRTIO Transport Layer, [495](#)
- L4virtio\_device\_status
  - L4 VIRTIO Transport Layer, [493](#)
- L4virtio\_feature\_bits
  - L4 VIRTIO Transport Layer, [493](#)
- L4VIRTIO\_FEATURE\_CMD\_CONFIG
  - L4 VIRTIO Transport Layer, [494](#)
- L4VIRTIO\_FEATURE\_VERSION\_1
  - L4 VIRTIO Transport Layer, [494](#)
- L4VIRTIO\_ID\_9P
  - L4 VIRTIO Transport Layer, [493](#)
- L4VIRTIO\_ID\_BALLOON
  - L4 VIRTIO Transport Layer, [493](#)
- L4VIRTIO\_ID\_BLOCK
  - L4 VIRTIO Transport Layer, [492](#)
- L4VIRTIO\_ID\_CAIF
  - L4 VIRTIO Transport Layer, [493](#)
- L4VIRTIO\_ID\_CONSOLE
  - L4 VIRTIO Transport Layer, [492](#)
- L4VIRTIO\_ID\_CRYPTIO
  - L4 VIRTIO Transport Layer, [493](#)
- L4VIRTIO\_ID\_FS
  - L4 VIRTIO Transport Layer, [493](#)
- L4VIRTIO\_ID\_GPIO
  - L4 VIRTIO Transport Layer, [493](#)
- L4VIRTIO\_ID\_GPU
  - L4 VIRTIO Transport Layer, [493](#)
- L4VIRTIO\_ID\_I2C
  - L4 VIRTIO Transport Layer, [493](#)
- L4VIRTIO\_ID\_INPUT
  - L4 VIRTIO Transport Layer, [493](#)
- L4VIRTIO\_ID\_NET
  - L4 VIRTIO Transport Layer, [492](#)
- L4VIRTIO\_ID\_RNG
  - L4 VIRTIO Transport Layer, [492](#)
- L4VIRTIO\_ID\_RPMSG
  - L4 VIRTIO Transport Layer, [493](#)
- L4VIRTIO\_ID\_RPROC\_SERIAL
  - L4 VIRTIO Transport Layer, [493](#)
- L4VIRTIO\_ID\_SCMI
  - L4 VIRTIO Transport Layer, [493](#)
- L4VIRTIO\_ID\_SCSI
  - L4 VIRTIO Transport Layer, [493](#)
- L4VIRTIO\_ID\_SOCKET
  - L4 VIRTIO Transport Layer, [493](#)
- L4VIRTIO\_ID\_VSOCK
  - L4 VIRTIO Transport Layer, [493](#)
- l4virtio\_input\_absinfo\_t, [2307](#)
- l4virtio\_input\_config\_t, [2308](#)
- l4virtio\_input\_devids\_t, [2308](#)
- l4virtio\_input\_event\_t, [2309](#)
- L4VIRTIO\_IRQ\_STATUS\_CONFIG
  - L4 VIRTIO Transport Layer, [492](#)
- L4VIRTIO\_IRQ\_STATUS\_VRING
  - L4 VIRTIO Transport Layer, [492](#)
- l4virtio\_net\_config\_t, [2309](#)
- l4virtio\_net\_header\_t, [2310](#)
- L4VIRTIO\_OP\_CONFIG\_QUEUE
  - L4 VIRTIO Transport Layer, [492](#)
- L4VIRTIO\_OP\_DEVICE\_CONFIG
  - L4 VIRTIO Transport Layer, [492](#)
- L4VIRTIO\_OP\_GET\_DEVICE\_IRQ
  - L4 VIRTIO Transport Layer, [492](#)
- L4VIRTIO\_OP\_REGISTER\_DS
  - L4 VIRTIO Transport Layer, [492](#)
- L4VIRTIO\_OP\_SET\_STATUS
  - L4 VIRTIO Transport Layer, [492](#)
- L4virtio\_port, [2311](#)
  - drop\_requests, [2315](#)
- l4virtio\_register\_ds
  - L4 VIRTIO Transport Layer, [497](#)
- l4virtio\_set\_status
  - L4 VIRTIO Transport Layer, [498](#)
- L4VIRTIO\_STATUS\_ACKNOWLEDGE
  - L4 VIRTIO Transport Layer, [493](#)
- L4VIRTIO\_STATUS\_DEVICE\_NEEDS\_RESET
  - L4 VIRTIO Transport Layer, [493](#)
- L4VIRTIO\_STATUS\_DRIVER
  - L4 VIRTIO Transport Layer, [493](#)
- L4VIRTIO\_STATUS\_DRIVER\_OK
  - L4 VIRTIO Transport Layer, [493](#)
- L4VIRTIO\_STATUS\_FAILED
  - L4 VIRTIO Transport Layer, [493](#)

- L4VIRTIO\_STATUS\_FEATURES\_OK
  - L4 VIRTIO Transport Layer, [493](#)
- Last
  - L4::lpc::Snd\_fpage, [1220](#)
- lcm
  - cxx, [745](#)
- learn
  - Mac\_table< Size >, [2318](#)
- length
  - L4::lpc::Varg, [1231](#)
- Libedid\_block\_size
  - EDID parsing functionality, [459](#)
- libedid\_check\_header
  - EDID parsing functionality, [459](#)
- libedid\_checksum
  - EDID parsing functionality, [460](#)
- Libedid\_consts
  - EDID parsing functionality, [459](#)
- libedid\_dump
  - EDID parsing functionality, [460](#)
- libedid\_dump\_standard\_timings
  - EDID parsing functionality, [460](#)
- libedid\_num\_ext\_blocks
  - EDID parsing functionality, [460](#)
- libedid\_pnp\_id
  - EDID parsing functionality, [462](#)
- libedid\_preferred\_resolution
  - EDID parsing functionality, [462](#)
- libedid\_revision
  - EDID parsing functionality, [462](#)
- libedid\_version
  - EDID parsing functionality, [462](#)
- Link
  - L4Re::Namespace, [1715](#)
- link
  - L4Re::Vfs::Directory, [1868](#)
- List\_alloc
  - cxx::List\_alloc, [953](#)
- list\_alloc.h
  - l4la\_alloc, [3372](#)
  - l4la\_avail, [3372](#)
  - l4la\_dump, [3372](#)
  - l4la\_free, [3372](#)
  - l4la\_init, [3373](#)
- load\_desc
  - L4virtio::Svr::Driver\_mem\_list\_t< DATA >, [2192](#), [2193](#)
- local
  - L4virtio::Svr::Driver\_mem\_region\_t< DATA >, [2200](#)
- local\_base
  - L4virtio::Svr::Driver\_mem\_region\_t< DATA >, [2200](#)
- local\_id\_received
  - L4::lpc::Snd\_fpage, [1223](#)
- Lock\_guard
  - L4::Lock\_guard, [1340](#)
- log
  - L4Re::Env, [1656](#)
- Log interface, [547](#)
  - l4re\_log\_print, [548](#)
  - l4re\_log\_print\_srv, [548](#)
  - l4re\_log\_printn, [549](#)
  - l4re\_log\_printn\_srv, [549](#)
- Logging interface, [601](#)
- lookup
  - Mac\_table< Size >, [2318](#)
- loop
  - L4::Server< LOOP\_HOOKS >, [1403](#)
  - L4Re::Util::Registry\_server< LOOP\_HOOKS >, [1849](#)
- loop\_dbg
  - L4::Server< LOOP\_HOOKS >, [1403](#)
  - L4Re::Util::Registry\_server< LOOP\_HOOKS >, [1849](#)
- Low-Level Thread Functions, [725](#)
- Low\_mask
  - cxx::Bitfield< T, LSB, MSB >, [868](#)
- lower\_bound\_node
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >, [905](#)
  - cxx::Bits::Bst< Node, Get\_key, Compare >, [922](#)
- Lsb
  - cxx::Bitfield< T, LSB, MSB >, [867](#)
- lseek
  - L4Re::Vfs::Regular\_file, [1900](#)
- Lstr
  - L4::Factory::Lstr, [1102](#)
- Mac\_addr, [2316](#)
- Mac\_table< Size >, [2316](#)
  - flush, [2317](#)
  - learn, [2318](#)
  - lookup, [2318](#)
- Mag, the GUI Multiplexer, [85](#)
- main\_thread
  - L4Re::Env, [1656](#)
- make\_cap
  - L4::lpc, [762](#)
- make\_cap\_full
  - L4::lpc, [762](#)
- make\_cap\_rw
  - L4::lpc, [763](#)
- make\_cap\_rws
  - L4::lpc, [764](#)
- make\_ref\_cap
  - L4Re Capability API, [593](#)
- make\_ref\_del\_cap
  - L4Re Capability API, [593](#)
- make\_shared\_cap
  - L4Re, [788](#)
  - L4Re::Util, [802](#)
- make\_shared\_del\_cap
  - L4Re, [790](#)
  - L4Re::Util, [802](#)
- make\_unique\_cap
  - L4Re, [791](#)



- L4Re::Util, [802](#)
- make\_unique\_del\_cap
  - L4Re, [791](#)
  - L4Re::Util, [803](#)
- Map
  - L4::lpc::Snd\_fpage, [1220](#)
- map
  - L4::Task, [1431](#)
  - L4Re::Dataspace, [1626](#)
  - L4Re::Dma\_space, [1643](#)
  - L4Re::Util::Dataspace\_svr, [1809](#)
- map\_hook
  - L4Re::Util::Dataspace\_svr, [1810](#)
- map\_info
  - L4Re::Dataspace, [1627](#)
  - L4Re::Util::Dataspace\_svr, [1811](#)
- map\_region
  - L4Re::Dataspace, [1628](#)
- Map\_type
  - L4::lpc::Snd\_fpage, [1220](#)
- Mask
  - cxx::Bitfield< T, LSB, MSB >, [868](#)
- mask
  - L4::lcu, [1115](#)
- Masks
  - cxx::Bitfield< T, LSB, MSB >, [867](#)
- max
  - Small C++ Template Library, [646](#)
- max\_free\_slabs
  - cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc >, [858](#)
  - cxx::Base\_slab\_static< Obj\_size, Slab\_size, Max\_free, Alloc >, [863](#)
- max\_queue\_size
  - L4virtio::Driver::Device, [2070](#)
- MB\_ART\_MEMORY
  - mb\_info.h, [3378](#)
- mb\_info.h
  - l4util\_mb\_for\_each\_mmap\_entry, [3378](#)
  - L4UTIL\_MB\_MEMORY, [3378](#)
  - MB\_ART\_MEMORY, [3378](#)
- Mem
  - L4::Type\_info::Demand\_t< CAPS, FLAGS, MEM, PORTS >, [1474](#)
- mem
  - L4::lpc::Rcv\_fpage, [1209](#)
  - L4::lpc::Snd\_fpage, [1223](#)
- mem\_alloc
  - L4Re::Env, [1657](#)
- Mem\_alloc\_flags
  - L4Re::Mem\_alloc, [1699](#)
- Mem\_desc
  - L4::Kip::Mem\_desc, [1310](#)
- mem\_free
  - L4Re::Mem\_alloc::Stats, [1702](#)
- mem\_limit
  - L4Re::Mem\_alloc::Stats, [1702](#)
- Mem\_type
  - L4::Kip::Mem\_desc, [1309](#)
- mem\_used
  - L4Re::Mem\_alloc::Stats, [1702](#)
- Memory
  - L4::lpc::Gen\_fpage, [1149](#)
- Memory allocator, [550](#)
  - l4re\_ma\_alloc, [551](#)
  - l4re\_ma\_alloc\_align, [552](#)
  - l4re\_ma\_alloc\_align\_srv, [554](#)
  - l4re\_ma\_flags, [551](#)
- Memory descriptors (C version), [445](#)
  - l4\_kernel\_info\_get\_mem\_desc\_end, [448](#)
  - l4\_kernel\_info\_get\_mem\_desc\_is\_virtual, [448](#)
  - l4\_kernel\_info\_get\_mem\_desc\_start, [448](#)
  - l4\_kernel\_info\_get\_mem\_desc\_subtype, [448](#)
  - l4\_kernel\_info\_get\_mem\_desc\_type, [449](#)
  - l4\_kernel\_info\_get\_num\_mem\_descs, [449](#)
  - l4\_kernel\_info\_mem\_desc\_t, [446](#)
  - l4\_kernel\_info\_set\_mem\_desc, [449](#)
  - l4\_mem\_archspecific\_acpi\_nvs, [447](#)
  - l4\_mem\_archspecific\_acpi\_tables, [447](#)
  - l4\_mem\_archspecific\_sub\_type\_common\_t, [446](#)
  - l4\_mem\_info\_acpi\_rsd, [447](#)
  - l4\_mem\_info\_sub\_type\_t, [447](#)
  - l4\_mem\_reserved\_heap, [447](#)
  - l4\_mem\_reserved\_kernel, [447](#)
  - l4\_mem\_reserved\_mmio, [447](#)
  - l4\_mem\_type\_archspecific, [447](#)
  - l4\_mem\_type\_bootloader, [447](#)
  - l4\_mem\_type\_conventional, [447](#)
  - l4\_mem\_type\_dedicated, [447](#)
  - l4\_mem\_type\_info, [447](#)
  - l4\_mem\_type\_reserved, [447](#)
  - l4\_mem\_type\_shared, [447](#)
  - l4\_mem\_type\_t, [447](#)
  - l4\_mem\_type\_undefined, [447](#)
- Memory management - Data Spaces and the Region Map, [29](#)
- Memory operations., [454](#)
  - L4\_mem\_op\_widths, [455](#)
  - l4\_mem\_read, [455](#)
  - L4\_MEM\_WIDTH\_1BYTE, [455](#)
  - L4\_MEM\_WIDTH\_2BYTE, [455](#)
  - L4\_MEM\_WIDTH\_4BYTE, [455](#)
  - l4\_mem\_write, [456](#)
- Memory related, [204](#)
  - l4\_addr\_consts\_t, [208](#)
  - l4\_bytes\_to\_mwords, [209](#)
  - L4\_INVALID\_ADDR, [209](#)
  - L4\_LOG2\_PAGESIZE, [205](#)
  - L4\_LOG2\_SUPERPAGESIZE, [205](#)
  - L4\_PAGEMASK, [205](#)
  - L4\_PAGESHIFT, [206](#)
  - l4\_round\_page, [209](#)
  - l4\_round\_size, [210](#)
  - L4\_SUPERPAGEMASK, [206](#)
  - L4\_SUPERPAGESHIFT, [207](#)
  - L4\_SUPERPAGESIZE, [207](#)



- [l4\\_trunc\\_page](#), [211](#)
- [l4\\_trunc\\_size](#), [212](#)
- message
  - [L4virtio::Svr::Bad\\_descriptor](#), [2103](#)
- Message Items, [236](#)
  - [L4\\_FPAGE\\_BUFFERABLE](#), [239](#)
  - [L4\\_FPAGE\\_C\\_IPCGATE\\_SVR](#), [241](#)
  - [L4\\_FPAGE\\_C\\_NO\\_REF\\_CNT](#), [240](#)
  - [L4\\_FPAGE\\_C\\_OBJ\\_RIGHT1](#), [240](#)
  - [L4\\_FPAGE\\_C\\_OBJ\\_RIGHT2](#), [241](#)
  - [L4\\_FPAGE\\_C\\_OBJ\\_RIGHT3](#), [241](#)
  - [L4\\_FPAGE\\_C\\_OBJ\\_RIGHTS](#), [241](#)
  - [L4\\_FPAGE\\_C\\_REF\\_CNT](#), [240](#)
  - [L4\\_FPAGE\\_CACHE\\_OPT](#), [239](#)
  - [l4\\_fpage\\_cacheability\\_opt\\_t](#), [238](#)
  - [L4\\_FPAGE\\_CACHEABLE](#), [239](#)
  - [L4\\_FPAGE\\_UNCACHEABLE](#), [239](#)
  - [L4\\_ITEM\\_CONT](#), [239](#)
  - [L4\\_ITEM\\_MAP](#), [239](#)
  - [l4\\_map\\_control](#), [241](#)
  - [L4\\_MAP\\_ITEM\\_GRANT](#), [239](#)
  - [L4\\_MAP\\_ITEM\\_MAP](#), [239](#)
  - [l4\\_map\\_obj\\_control](#), [241](#)
  - [l4\\_msg\\_item\\_consts\\_t](#), [239](#)
  - [L4\\_obj\\_fpage\\_ctl](#), [240](#)
  - [L4\\_RCV\\_ITEM\\_FORWARD\\_MAPPINGS](#), [240](#)
  - [L4\\_RCV\\_ITEM\\_LOCAL\\_ID](#), [240](#)
  - [L4\\_RCV\\_ITEM\\_SINGLE\\_CAP](#), [240](#)
- Message Registers (MRs), [275](#)
- Message Tag, [257](#)
  - [l4\\_msgtag](#), [261](#)
  - [L4\\_MSGTAG\\_ERROR](#), [259](#)
  - [L4\\_MSGTAG\\_FLAGS](#), [259](#)
  - [L4\\_msgtag\\_flags](#), [259](#)
  - [l4\\_msgtag\\_flags](#), [262](#)
  - [l4\\_msgtag\\_has\\_error](#), [264](#)
  - [l4\\_msgtag\\_is\\_exception](#), [264](#)
  - [l4\\_msgtag\\_is\\_io\\_page\\_fault](#), [265](#)
  - [l4\\_msgtag\\_is\\_page\\_fault](#), [265](#)
  - [l4\\_msgtag\\_is\\_sigma0](#), [266](#)
  - [l4\\_msgtag\\_items](#), [267](#)
  - [l4\\_msgtag\\_label](#), [267](#)
  - [L4\\_msgtag\\_protocol](#), [259](#)
  - [L4\\_MSGTAG\\_SCHEDULE](#), [259](#)
  - [l4\\_msgtag\\_t](#), [259](#)
  - [L4\\_MSGTAG\\_TRANSFER\\_FPU](#), [259](#)
  - [l4\\_msgtag\\_words](#), [269](#)
  - [L4\\_platform\\_ctl\\_proto](#), [260](#)
  - [L4\\_PROTO\\_ALLOW\\_SYSCALL](#), [260](#)
  - [L4\\_PROTO\\_DEBUGGER](#), [260](#)
  - [L4\\_PROTO\\_DMA\\_SPACE](#), [260](#)
  - [L4\\_PROTO\\_EXCEPTION](#), [260](#)
  - [L4\\_PROTO\\_FACTORY](#), [260](#)
  - [L4\\_PROTO\\_IO\\_PAGE\\_FAULT](#), [260](#)
  - [L4\\_PROTO\\_IOMMU](#), [260](#)
  - [L4\\_PROTO\\_IRQ](#), [260](#)
  - [L4\\_PROTO\\_IRQ\\_SENDER](#), [260](#)
  - [L4\\_PROTO\\_KOBJECT](#), [260](#)
  - [L4\\_PROTO\\_LOG](#), [260](#)
  - [L4\\_PROTO\\_META](#), [260](#)
  - [L4\\_PROTO\\_NONE](#), [260](#)
  - [L4\\_PROTO\\_PAGE\\_FAULT](#), [260](#)
  - [L4\\_PROTO\\_PF\\_EXCEPTION](#), [260](#)
  - [L4\\_PROTO\\_PLATFORM\\_CTL](#), [260](#)
  - [L4\\_PROTO\\_SCHEDULER](#), [260](#)
  - [L4\\_PROTO\\_SEMAPHORE](#), [260](#)
  - [L4\\_PROTO\\_SIGMA0](#), [260](#)
  - [L4\\_PROTO\\_SMCCC](#), [260](#)
  - [L4\\_PROTO\\_TASK](#), [260](#)
  - [L4\\_PROTO\\_THREAD](#), [260](#)
  - [L4\\_PROTO\\_THREAD\\_GROUP](#), [260](#)
  - [L4\\_PROTO\\_VCPU\\_CONTEXT](#), [260](#)
  - [L4\\_PROTO\\_VM](#), [260](#)
- min
  - [Small C++ Template Library](#), [647](#)
- mkdir
  - [L4Re::Vfs::Directory](#), [1868](#)
- mmio\_read
  - [L4Re::Mmio\\_space](#), [1708](#)
- mmio\_write
  - [L4Re::Mmio\\_space](#), [1709](#)
- Mode
  - [L4Re::Util::Event\\_t< PAYLOAD >](#), [1824](#)
- mode
  - [L4::Uart](#), [1520](#)
- Mode\_irq
  - [L4Re::Util::Event\\_t< PAYLOAD >](#), [1825](#)
- Mode\_polling
  - [L4Re::Util::Event\\_t< PAYLOAD >](#), [1825](#)
- modify
  - [L4drivers::Register\\_tmpl< BITS, BLOCK >](#), [1601](#)
- modify\_senders
  - [L4::Thread](#), [1442](#)
- Moe, the Root-Task, [53](#)
- More
  - [L4::lpc::Snd\\_fpage](#), [1220](#)
- mount
  - [L4Re::Vfs::File\\_system](#), [1875](#)
  - [L4Re::Vfs::Fs](#), [1879](#)
- move
  - [L4::Cap< T >](#), [1044](#)
  - [L4::Cap\\_base](#), [1054](#)
- Mr\_bytes
  - [L4::lpc::Msg](#), [768](#)
- Mr\_words
  - [L4::lpc::Msg](#), [768](#)
- Msb
  - [cxx::Bitfield< T, LSB, MSB >](#), [867](#)
- msg\_add
  - [L4::lpc::Msg](#), [771](#)
- msg\_get
  - [L4::lpc::Msg](#), [772](#)
- Msg\_ptr
  - [L4::lpc::Msg\\_ptr< T >](#), [1196](#)
- msg\_ptr
  - [L4::lpc](#), [765](#)

- msi\_info
  - L4::Icu, [1116](#)
- N
  - cxx::Bits::Direction, [930](#)
- Name-space API, [601](#)
- Name\_max
  - L4Re::Inhibitor, [1676](#)
- Namespace interface, [556](#)
  - l4re\_ns\_query\_srv, [557](#)
  - l4re\_ns\_query\_to\_srv, [558](#)
  - l4re\_ns\_register\_flags, [557](#)
  - l4re\_ns\_register\_obj\_srv, [559](#)
- Ned, the Init Process, [57](#)
- negotiated\_features
  - L4virtio::Svr::Dev\_config, [2172](#)
- Net\_transfer, [2319](#)
  - cur\_buf, [2321](#)
  - done, [2321](#)
- next
  - L4Re::Event\_buffer\_t< PAYLOAD >, [1671](#)
  - L4virtio::Svr::Request\_processor, [2203](#)
- next\_avail
  - L4virtio::Svr::Virtqueue, [2270](#)
- next\_bfm\_t
  - L4virtio::Virtqueue::Desc::Flags, [2297](#)
- next\_block
  - L4virtio::Svr::Block\_request< Ds\_data >, [2113](#)
- next\_device
  - L4vbus::Device, [1978](#)
- next\_lock\_info
  - L4Re::Inhibitor, [1677](#)
- next\_timeout
  - L4::lpc\_svr::Timeout\_queue, [1278](#)
- no\_demand
  - L4::Type\_info::Demand, [1470](#)
- No\_eager\_map
  - L4Re::Rm::F, [1757](#)
  - Rm::F, [2339](#)
- No\_init
  - L4::Cap\_base, [1047](#)
- No\_init\_type
  - L4::Cap\_base, [1047](#)
- no\_irq\_bfm\_t
  - L4virtio::Virtqueue::Avail::Flags, [2294](#)
- no\_notify\_bfm\_t
  - L4virtio::Virtqueue::Used::Flags, [2300](#)
- no\_notify\_guest
  - L4virtio::Virtqueue, [2283](#)
- no\_notify\_host
  - L4virtio::Virtqueue, [2283](#), [2284](#)
- No\_sync
  - L4Re::Dma\_space, [1641](#)
- None
  - L4::lpc::Snd\_fpage, [1219](#)
  - L4::Types::Flags< BITS\_ENUM, UNDERLYING >, [1502](#)
  - L4Re::Dma\_space, [1641](#)
- None\_type
  - L4::Types::Flags< BITS\_ENUM, UNDERLYING >, [1502](#)
- Normal
  - L4Re::Dataspace::F, [1633](#)
- notify\_queue
  - L4virtio::Svr::Console::Device, [2124](#)
  - L4virtio::Svr::Console::Virtio\_con, [2150](#)
  - Virtio\_net, [2354](#)
- NT\_ASRS
  - ELF binary format, [710](#)
- NT\_AUXV
  - ELF binary format, [710](#)
- NT\_FPREGSET
  - ELF binary format, [710](#)
- NT\_GWINDOWS
  - ELF binary format, [710](#)
- NT\_LWPSINFO
  - ELF binary format, [710](#)
- NT\_LWPSTATUS
  - ELF binary format, [710](#)
- NT\_PLATFORM
  - ELF binary format, [710](#)
- NT\_PRCRED
  - ELF binary format, [710](#)
- NT\_PRFPXREG
  - ELF binary format, [710](#)
- NT\_PRPSINFO
  - ELF binary format, [710](#)
- NT\_PRSTATUS
  - ELF binary format, [710](#)
- NT\_PRXREG
  - ELF binary format, [710](#)
- NT\_PSINFO
  - ELF binary format, [710](#)
- NT\_PSTATUS
  - ELF binary format, [710](#)
- NT\_TASKSTRUCT
  - ELF binary format, [710](#)
- NT\_UTSNAME
  - ELF binary format, [710](#)
- NT\_VERSION
  - ELF binary format, [710](#)
- num
  - L4virtio::Virtqueue, [2284](#)
- num\_interfaces
  - L4::Meta, [1345](#)
- NVMe server, [82](#)
- Obj
  - L4::lpc::Gen\_fpage, [1149](#)
- obj
  - L4::lpc::Rcv\_fpage, [1210](#)
  - L4::lpc::Snd\_fpage, [1224](#)
- obj\_cap
  - L4::Epiface, [1078](#)
  - L4::Epiface\_t0< RPC\_IFACE, BASE >, [1087](#)
  - L4::lreqp\_t< Derived, BASE, bool >, [1305](#)
- obj\_info.h
  - l4\_debugger\_query\_obj\_infos, [3246](#)

- Object Invocation, [215](#)
  - [l4\\_ipc](#), [218](#)
  - [l4\\_ipc\\_call](#), [220](#)
  - [l4\\_ipc\\_receive](#), [223](#)
  - [l4\\_ipc\\_reply\\_and\\_wait](#), [225](#)
  - [l4\\_ipc\\_send](#), [227](#)
  - [l4\\_ipc\\_send\\_and\\_wait](#), [228](#)
  - [l4\\_ipc\\_sleep](#), [229](#)
  - [l4\\_ipc\\_sleep\\_ms](#), [231](#)
  - [l4\\_ipc\\_sleep\\_us](#), [233](#)
  - [l4\\_ipc\\_wait](#), [234](#)
  - [l4\\_sndfpage\\_add](#), [235](#)
  - [l4\\_syscall\\_flags\\_t](#), [217](#)
  - [L4\\_SYSF\\_CALL](#), [218](#)
  - [L4\\_SYSF\\_NONE](#), [218](#)
  - [L4\\_SYSF\\_OPEN\\_WAIT](#), [218](#)
  - [L4\\_SYSF\\_RECV](#), [218](#)
  - [L4\\_SYSF\\_REPLY](#), [218](#)
  - [L4\\_SYSF\\_REPLY\\_AND\\_WAIT](#), [218](#)
  - [L4\\_SYSF\\_SEND](#), [218](#)
  - [L4\\_SYSF\\_SEND\\_AND\\_WAIT](#), [218](#)
  - [L4\\_SYSF\\_WAIT](#), [218](#)
- Object\_registry
  - [L4Re::Util::Object\\_registry](#), [1838](#)
- object\_size
  - [cxx::Base\\_slab< Obj\\_size, Slab\\_size, Max\\_free, Alloc >](#), [858](#)
  - [cxx::Base\\_slab\\_static< Obj\\_size, Slab\\_size, Max\\_free, Alloc >](#), [863](#)
- objects\_per\_slab
  - [cxx::Base\\_slab< Obj\\_size, Slab\\_size, Max\\_free, Alloc >](#), [858](#)
  - [cxx::Base\\_slab\\_static< Obj\\_size, Slab\\_size, Max\\_free, Alloc >](#), [863](#)
- offset
  - [l4\\_sched\\_cpu\\_set\\_t](#), [1558](#)
- op\_create
  - [Switch\\_factory](#), [2349](#)
- operator bool
  - [L4virtio::Svr::Virtqueue::Head\\_desc](#), [2273](#)
- operator l4\_msgtag\_t
  - [L4::Factory::S](#), [1106](#)
- operator new
  - [Small C++ Template Library](#), [647](#)
- operator value\_type
  - [L4drivers::Ro\\_register\\_tmpl< BITS, BLOCK >](#), [1607](#)
- operator!
  - [cxx::Bits::Direction](#), [931](#)
- operator!=
  - [L4vbus::Device](#), [1979](#)
- operator<<
  - [ipc\\_stream](#), [2601–2603](#)
  - [L4::Factory::S](#), [1106](#), [1107](#)
- operator>>
  - [ipc\\_stream](#), [2603–2607](#)
- operator()
  - [cxx::Pair\\_first\\_compare< Cmp, Typ >](#), [969](#)
- operator->
  - [cxx::Bits::Base\\_avl\\_set< ITEM\\_TYPE, COMPARE, ALLOC, GET\\_KEY >::Node](#), [909](#)
- operator=
  - [L4::Lock\\_guard](#), [1341](#)
  - [L4drivers::Register\\_tmpl< BITS, BLOCK >](#), [1602](#)
  - [L4Re::Rm::Unique\\_region< T >](#), [1763](#)
  - [Rm::Unique\\_region< T >](#), [2344](#)
- operator==
  - [L4Re::Video::Color\\_component](#), [1909](#)
  - [L4Re::Video::Pixel\\_info](#), [1932](#)
  - [L4vbus::Device](#), [1979](#)
- operator[]
  - [cxx::Avl\\_map< KEY\\_TYPE, DATA\\_TYPE, COMPARE, ALLOC >](#), [839](#), [840](#)
  - [cxx::Bitmap\\_base](#), [889](#), [890](#)
  - [cxx::List< D, Alloc >](#), [951](#)
  - [L4drivers::Register\\_block< MAX\\_BITS, BLOCK >](#), [1590](#), [1591](#)
  - [L4drivers::Ro\\_register\\_block< MAX\\_BITS, BLOCK >](#), [1605](#)
  - [L4Re::Util::Bitmap\\_base](#), [1779](#), [1780](#)
- operator\*
  - [cxx::Bits::Base\\_avl\\_set< ITEM\\_TYPE, COMPARE, ALLOC, GET\\_KEY >::Node](#), [909](#)
- outchar
  - [kdebug.h](#), [3216](#)
- outdec
  - [kdebug.h](#), [3217](#)
- outhex12
  - [kdebug.h](#), [3218](#)
- outhex16
  - [kdebug.h](#), [3218](#)
- outhex20
  - [kdebug.h](#), [3219](#)
- outhex32
  - [kdebug.h](#), [3219](#)
- outhex64
  - [kdebug.h](#), [3221](#)
- outhex8
  - [kdebug.h](#), [3221](#)
- outnstring
  - [kdebug.h](#), [3222](#)
- outstring
  - [kdebug.h](#), [3223](#)
- outumword
  - [kdebug.h](#), [3224](#)
- Overview, [1](#)
- Overwrite
  - [L4Re::Namespace](#), [1715](#)
- p\_flags
  - [Elf32\\_Phdr](#), [1013](#)
  - [Elf64\\_Phdr](#), [1023](#)
- p\_type
  - [Elf32\\_Phdr](#), [1013](#)
  - [Elf64\\_Phdr](#), [1023](#)
- padding
  - [L4Re::Video::Pixel\\_info](#), [1933](#)

- page\_fault
  - L4Re::Pager, [1353](#)
- page\_shift
  - L4Re::Util::Dataspace\_svr, [1811](#)
- Pager
  - L4Re::Rm::F, [1758](#)
  - Rm::F, [2339](#)
- pager
  - L4Re::Thread::Attr, [1454](#)
- Pair
  - cxx::Pair< First, Second >, [967](#)
- Pair\_first\_compare
  - cxx::Pair\_first\_compare< Cmp, Typ >, [968](#)
- parent
  - L4Re::Env, [1657](#)
- Parent API, [602](#)
- Parent interface, [560](#)
- parent.h
  - l4re\_parent\_signal, [2859](#)
- parse\_cmdline
  - Comfortable Command Line Parsing, [722](#)
- parse\_device\_name
  - Block\_device::Device\_mgr< DEV, FACTORY, SCHEDULER >, [809](#)
- Partly\_resolved
  - L4Re::Namespace, [1714](#)
- PF\_ARM\_SB
  - ELF binary format, [701](#)
- PF\_MASKOS
  - ELF binary format, [711](#)
- PF\_MASKPROC
  - ELF binary format, [711](#)
- PF\_R
  - ELF binary format, [711](#)
- PF\_W
  - ELF binary format, [711](#)
- PF\_X
  - ELF binary format, [711](#)
- Phys\_space
  - L4Re::Dma\_space, [1641](#)
- pin
  - L4vbus::Gpio\_module, [1985](#)
  - L4vbus::Gpio\_pin, [1995](#)
- Pinned
  - L4Re::Mem\_alloc, [1699](#)
- Pixel\_info
  - L4Re::Video::Pixel\_info, [1926](#), [1927](#)
- pkg/drivers-frst/include/ARCH-amd64/asm\_access.h, [2367](#)
- pkg/drivers-frst/include/ARCH-arm/asm\_access.h, [2367](#)
- pkg/drivers-frst/include/ARCH-arm64/asm\_access.h, [2368](#)
- pkg/drivers-frst/include/ARCH-mips/asm\_access.h, [2369](#)
- pkg/drivers-frst/include/ARCH-ppc32/asm\_access.h, [2369](#)
- pkg/drivers-frst/include/ARCH-riscv/asm\_access.h, [2370](#)
- pkg/drivers-frst/include/ARCH-sparc/asm\_access.h, [2371](#)
- pkg/drivers-frst/include/ARCH-x86/asm\_access.h, [2371](#)
- pkg/drivers-frst/include/asm\_access\_gen.h, [2372](#)
- pkg/drivers-frst/include/hw\_mmio\_register\_block, [2372](#)
- pkg/drivers-frst/include/hw\_register\_block, [2373](#)
- pkg/drivers-frst/include/io\_regblock.h, [2376](#)
- pkg/drivers-frst/include/io\_regblock\_port.h, [2379](#)
- pkg/drivers-frst/include/Makefile, [2412](#)
- pkg/drivers-frst/include/poll\_timeout\_counter.h, [2379](#)
- pkg/drivers-frst/uart/include/Makefile, [2412](#)
- pkg/drivers-frst/uart/include/uart\_16550.h, [2380](#)
- pkg/drivers-frst/uart/include/uart\_16550\_dw.h, [2381](#)
- pkg/drivers-frst/uart/include/uart\_apb.h, [2382](#)
- pkg/drivers-frst/uart/include/uart\_base.h, [2382](#)
- pkg/drivers-frst/uart/include/uart\_cadence.h, [2383](#)
- pkg/drivers-frst/uart/include/uart\_dcc-v6.h, [2384](#)
- pkg/drivers-frst/uart/include/uart\_dm.h, [2384](#)
- pkg/drivers-frst/uart/include/uart\_dummy.h, [2385](#)
- pkg/drivers-frst/uart/include/uart\_geni.h, [2385](#)
- pkg/drivers-frst/uart/include/uart\_imx.h, [2386](#)
- pkg/drivers-frst/uart/include/uart\_leon3.h, [2387](#)
- pkg/drivers-frst/uart/include/uart\_linflex.h, [2387](#)
- pkg/drivers-frst/uart/include/uart\_lpuart.h, [2388](#)
- pkg/drivers-frst/uart/include/uart\_mvbu.h, [2388](#)
- pkg/drivers-frst/uart/include/uart\_of.h, [2389](#)
- pkg/drivers-frst/uart/include/uart\_omap35x.h, [2389](#)
- pkg/drivers-frst/uart/include/uart\_pl011.h, [2390](#)
- pkg/drivers-frst/uart/include/uart\_s3c2410.h, [2390](#)
- pkg/drivers-frst/uart/include/uart\_sa1000.h, [2391](#)
- pkg/drivers-frst/uart/include/uart\_sbi.h, [2392](#)
- pkg/drivers-frst/uart/include/uart\_sh.h, [2392](#)
- pkg/drivers-frst/uart/include/uart\_sifive.h, [2393](#)
- pkg/drivers-frst/uart/include/uart\_tegra-tcu.h, [2393](#)
- pkg/l4re-core/ned/doc/tutorial.lua, [2394](#)
- pkg/l4re-core/ned/lib/include/cmd\_control, [2397](#)
- pkg/l4re-core/ned/lib/include/Makefile, [2412](#)
- pkg/virtio-net-switch/server/switch/debug.h, [2397](#)
- pkg/virtio-net-switch/server/switch/filter.cc, [2400](#)
- pkg/virtio-net-switch/server/switch/filter.h, [2400](#)
- pkg/virtio-net-switch/server/switch/mac\_addr.h, [2401](#)
- pkg/virtio-net-switch/server/switch/mac\_table.h, [2402](#)
- pkg/virtio-net-switch/server/switch/main.cc, [2404](#)
- pkg/virtio-net-switch/server/switch/Makefile, [2412](#)
- pkg/virtio-net-switch/server/switch/options.cc, [2413](#)
- pkg/virtio-net-switch/server/switch/options.h, [2415](#)
- pkg/virtio-net-switch/server/switch/port.h, [2416](#)
- pkg/virtio-net-switch/server/switch/port\_ixl.h, [2418](#)
- pkg/virtio-net-switch/server/switch/port\_l4virtio.h, [2420](#)
- pkg/virtio-net-switch/server/switch/request.h, [2424](#)
- pkg/virtio-net-switch/server/switch/request\_ixl.h, [2425](#)
- pkg/virtio-net-switch/server/switch/request\_l4virtio.h, [2427](#)
- pkg/virtio-net-switch/server/switch/stats.h, [2430](#)
- pkg/virtio-net-switch/server/switch/switch.cc, [2431](#)
- pkg/virtio-net-switch/server/switch/switch.h, [2434](#)
- pkg/virtio-net-switch/server/switch/virtio\_net.h, [2435](#)

- pkg/virtio-net-switch/server/switch/virtio\_net\_buffer.h, 2439
- pkg/virtio-net-switch/server/switch/vlan.h, 2440
- Platform Control C API, 357
  - l4\_platform\_ctl\_cpu\_allow\_shutdown, 358
  - l4\_platform\_ctl\_cpu\_disable, 358
  - l4\_platform\_ctl\_cpu\_enable, 359
  - l4\_platform\_ctl\_set\_task\_asid, 360
  - l4\_platform\_ctl\_system\_shutdown, 361
  - l4\_platform\_ctl\_system\_suspend, 361
- pm\_resume
  - L4vbus::Pm< DEC >, 2016
- pm\_suspend
  - L4vbus::Pm< DEC >, 2016
- Poll\_timeout\_counter
  - L4::Poll\_timeout\_counter, 1363
- Poll\_timeout\_kipclock
  - L4::Poll\_timeout\_kipclock, 1365
- pop\_front
  - cxx::Bits::Smart\_ptr\_list< ITEM >, 934
  - cxx::H\_list< T, POLICY >, 942
  - cxx::S\_list< T, POLICY >, 979
- port
  - L4virtio::Svr::Console::Device, 2124
  - L4virtio::Svr::Console::Virtio\_con, 2151
- port\_add
  - L4virtio::Svr::Console::Virtio\_con, 2152
- Port\_added
  - L4virtio::Svr::Console::Port, 2142
- port\_available
  - Virtio\_switch, 2361
- Port\_disabled
  - L4virtio::Svr::Console::Port, 2142
- Port\_failed
  - L4virtio::Svr::Console::Port, 2142
- Port\_name
  - L4virtio::Svr::Console::Control\_message, 2115
- port\_name
  - L4virtio::Svr::Console::Virtio\_con, 2153
- Port\_num\_states
  - L4virtio::Svr::Console::Port, 2142
- Port\_open
  - L4virtio::Svr::Console::Control\_message, 2115
  - L4virtio::Svr::Console::Port, 2142
- port\_open
  - L4virtio::Svr::Console::Virtio\_con, 2154
- port\_read
  - L4virtio::Svr::Console::Device, 2125
- Port\_ready
  - L4virtio::Svr::Console::Control\_message, 2115
  - L4virtio::Svr::Console::Port, 2142
- port\_remove
  - L4virtio::Svr::Console::Virtio\_con, 2155
- Port\_status
  - L4virtio::Svr::Console::Port, 2142
- port\_write
  - L4virtio::Svr::Console::Device, 2127
- Ports
  - L4::Type\_info::Demand\_t< CAPS, FLAGS, MEM, PORTS >, 1474
- print
  - L4Re::Log, 1694
- println
  - L4Re::Log, 1694
- prio
  - l4\_sched\_param\_t, 1562
- process
  - L4Re::Util::Event\_buffer\_consumer\_t< PAYLOAD >, 1815
- process\_device\_ready
  - L4virtio::Svr::Console::Device, 2129
  - L4virtio::Svr::Console::Virtio\_con, 2156
- process\_port\_open
  - L4virtio::Svr::Console::Device, 2130
  - L4virtio::Svr::Console::Virtio\_con, 2157
- process\_port\_ready
  - L4virtio::Svr::Console::Device, 2130
  - L4virtio::Svr::Console::Virtio\_con, 2157
- process\_request
  - L4virtio::Driver::Block\_device, 2058
- process\_used\_queue
  - L4virtio::Driver::Block\_device, 2059
- Producer, 621, 633
  - l4shmc\_chunk\_ready, 622
  - l4shmc\_chunk\_ready\_sig, 622
  - l4shmc\_chunk\_try\_to\_take, 623
  - l4shmc\_chunk\_try\_to\_take\_for\_overwriting, 623
  - l4shmc\_chunk\_try\_to\_take\_for\_writing, 624
  - l4shmc\_is\_chunk\_clear, 624
  - l4shmc\_trigger, 633
- prog.mk - Application Role, 37
- Program Input and Output, 30
- Programming for L4Re, 9
- PROTO\_ANY
  - L4, 752
- proto\_dispatch
  - L4::Kobject\_typeid< T >, 1334
  - L4::Kobject\_typeid< void >, 1337
  - L4::Server\_object\_t< IFACE, BASE >, 1413
- PROTO\_EMPTY
  - L4, 752
- PT\_DYNAMIC
  - ELF binary format, 711
- PT\_GNU\_EH\_FRAME
  - ELF binary format, 712
- PT\_GNU\_RELRO
  - ELF binary format, 712
- PT\_GNU\_STACK
  - ELF binary format, 712
- PT\_HIOS
  - ELF binary format, 712
- PT\_HIPROC
  - ELF binary format, 712
- PT\_INTERP
  - ELF binary format, 711
- PT\_L4\_AUX

- ELF binary format, [712](#)
- PT\_L4\_STACK
  - ELF binary format, [712](#)
- PT\_LOAD
  - ELF binary format, [711](#)
- PT\_LOOS
  - ELF binary format, [712](#)
- PT\_LOPROC
  - ELF binary format, [712](#)
- PT\_NOTE
  - ELF binary format, [712](#)
- PT\_NULL
  - ELF binary format, [711](#)
- PT\_NUM
  - ELF binary format, [712](#)
- PT\_PHDR
  - ELF binary format, [712](#)
- PT\_SHLIB
  - ELF binary format, [712](#)
- PT\_TLS
  - ELF binary format, [712](#)
- Pthread Support, [31](#)
- ptr
  - cxx::Ref\_ptr< T, CNT >, [975](#)
  - L4Re::Core::Ref\_ptr< T, CNT >, [1617](#)
- push\_back
  - cxx::List\_item, [957](#)
- push\_front
  - cxx::List\_item, [958](#)
- put
  - L4::Factory::S, [1107–1109](#)
  - L4::lpc::lostream, [1157](#), [1158](#)
  - L4::lpc::Ostream, [1202](#), [1203](#)
  - L4Re::Event\_buffer\_t< PAYLOAD >, [1671](#)
- qconfig
  - L4virtio::Svr::Dev\_config, [2172](#)
- query
  - L4Re::Namespace, [1715](#), [1716](#)
- query\_log\_name
  - L4::Debugger, [1066](#)
- query\_log\_typeid
  - L4::Debugger, [1067](#)
- Query\_result\_flags
  - L4Re::Namespace, [1714](#)
- Query\_timeout
  - L4Re::Namespace, [1714](#)
- quota
  - L4Re::Mem\_alloc::Stats, [1702](#)
- quota\_used
  - L4Re::Mem\_alloc::Stats, [1703](#)
- R
  - cxx::Bits::Direction, [930](#)
  - L4Re::Dataspace::F, [1632](#)
  - L4Re::Rm::F, [1758](#)
  - Rm::F, [2339](#)
- r
  - L4drivers::Register\_block< MAX\_BITS, BLOCK >, [1591](#), [1592](#)
  - L4drivers::Ro\_register\_block< MAX\_BITS, BLOCK >, [1605](#)
  - L4Re::Video::Pixel\_info, [1933](#), [1934](#)
  - L4vcpu::Vcpu, [2041](#), [2042](#)
  - R\_386\_32
    - ELF binary format, [712](#)
  - R\_386\_COPY
    - ELF binary format, [712](#)
  - R\_386\_GLOB\_DAT
    - ELF binary format, [712](#)
  - R\_386\_GOT32
    - ELF binary format, [712](#)
  - R\_386\_GOTOFF
    - ELF binary format, [712](#)
  - R\_386\_GOTPC
    - ELF binary format, [712](#)
  - R\_386\_JMP\_SLOT
    - ELF binary format, [712](#)
  - R\_386\_NONE
    - ELF binary format, [712](#)
  - R\_386\_NUM
    - ELF binary format, [713](#)
  - R\_386\_PC32
    - ELF binary format, [712](#)
  - R\_386\_PLT32
    - ELF binary format, [712](#)
  - R\_386\_RELATIVE
    - ELF binary format, [712](#)
  - R\_386\_TLS\_DTPMOD32
    - ELF binary format, [713](#)
  - R\_386\_TLS\_DTPOFF32
    - ELF binary format, [713](#)
  - R\_386\_TLS\_GD
    - ELF binary format, [712](#)
  - R\_386\_TLS\_GD\_32
    - ELF binary format, [712](#)
  - R\_386\_TLS\_GD\_CALL
    - ELF binary format, [713](#)
  - R\_386\_TLS\_GD\_POP
    - ELF binary format, [713](#)
  - R\_386\_TLS\_GD\_PUSH
    - ELF binary format, [713](#)
  - R\_386\_TLS\_GOTIE
    - ELF binary format, [712](#)
  - R\_386\_TLS\_IE
    - ELF binary format, [712](#)
  - R\_386\_TLS\_IE\_32
    - ELF binary format, [713](#)
  - R\_386\_TLS\_LDM
    - ELF binary format, [712](#)
  - R\_386\_TLS\_LDM\_32
    - ELF binary format, [713](#)
  - R\_386\_TLS\_LDM\_CALL
    - ELF binary format, [713](#)
  - R\_386\_TLS\_LDM\_POP
    - ELF binary format, [713](#)

- R\_386\_TLS\_LDM\_PUSH
  - ELF binary format, [713](#)
- R\_386\_TLS\_LDO\_32
  - ELF binary format, [713](#)
- R\_386\_TLS\_LE
  - ELF binary format, [712](#)
- R\_386\_TLS\_LE\_32
  - ELF binary format, [713](#)
- R\_386\_TLS\_TPOFF
  - ELF binary format, [712](#)
- R\_386\_TLS\_TPOFF32
  - ELF binary format, [713](#)
- R\_AARCH64\_NONE
  - ELF binary format, [713](#)
- R\_ARM\_ABS12
  - ELF binary format, [713](#)
- R\_ARM\_ABS16
  - ELF binary format, [713](#)
- R\_ARM\_ABS32
  - ELF binary format, [713](#)
- R\_ARM\_ABS8
  - ELF binary format, [714](#)
- R\_ARM\_COPY
  - ELF binary format, [714](#)
- R\_ARM\_GLOB\_DAT
  - ELF binary format, [714](#)
- R\_ARM\_GOT32
  - ELF binary format, [714](#)
- R\_ARM\_GOTOFF
  - ELF binary format, [714](#)
- R\_ARM\_GOTPC
  - ELF binary format, [714](#)
- R\_ARM\_JUMP\_SLOT
  - ELF binary format, [714](#)
- R\_ARM\_NONE
  - ELF binary format, [713](#)
- R\_ARM\_NUM
  - ELF binary format, [714](#)
- R\_ARM\_PC24
  - ELF binary format, [713](#)
- R\_ARM\_PLT32
  - ELF binary format, [714](#)
- R\_ARM\_REL32
  - ELF binary format, [713](#)
- R\_ARM\_RELATIVE
  - ELF binary format, [714](#)
- R\_ARM\_THM\_PC11
  - ELF binary format, [714](#)
- R\_ARM\_THM\_PC9
  - ELF binary format, [714](#)
- R\_X86\_64\_16
  - ELF binary format, [714](#)
- R\_X86\_64\_32
  - ELF binary format, [714](#)
- R\_X86\_64\_32S
  - ELF binary format, [714](#)
- R\_X86\_64\_64
  - ELF binary format, [714](#)
- R\_X86\_64\_8
  - ELF binary format, [714](#)
- R\_X86\_64\_COPY
  - ELF binary format, [714](#)
- R\_X86\_64\_DTPMOD64
  - ELF binary format, [714](#)
- R\_X86\_64\_DTPOFF32
  - ELF binary format, [715](#)
- R\_X86\_64\_DTPOFF64
  - ELF binary format, [714](#)
- R\_X86\_64\_GLOB\_DAT
  - ELF binary format, [714](#)
- R\_X86\_64\_GOT32
  - ELF binary format, [714](#)
- R\_X86\_64\_GOTPCREL
  - ELF binary format, [714](#)
- R\_X86\_64\_GOTTPOFF
  - ELF binary format, [715](#)
- R\_X86\_64\_JUMP\_SLOT
  - ELF binary format, [714](#)
- R\_X86\_64\_NONE
  - ELF binary format, [714](#)
- R\_X86\_64\_PC16
  - ELF binary format, [714](#)
- R\_X86\_64\_PC32
  - ELF binary format, [714](#)
- R\_X86\_64\_PC8
  - ELF binary format, [714](#)
- R\_X86\_64\_PLT32
  - ELF binary format, [714](#)
- R\_X86\_64\_RELATIVE
  - ELF binary format, [714](#)
- R\_X86\_64\_TLSD
  - ELF binary format, [715](#)
- R\_X86\_64\_TLSD
  - ELF binary format, [715](#)
- R\_X86\_64\_TPOFF32
  - ELF binary format, [715](#)
- R\_X86\_64\_TPOFF64
  - ELF binary format, [714](#)
- raise
  - L4Re::Itas, [1683](#)
- RAM configuration, [81](#)
- Random number support, [724](#)
  - l4util\_rand, [724](#)
  - l4util\_srand, [724](#)
- rate
  - L4::Uart, [1520](#)
- rbegin
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >, [905](#), [906](#)
  - cxx::Bits::Bst< Node, Get\_key, Compare >, [922](#), [923](#)
- rcv\_cap
  - L4::lpc\_svr::Server\_iface, [1270](#), [1271](#)
- rcv\_endpoint.h
  - L4\_RCV\_EP\_BIND\_OP, [3261](#)
  - L4\_rcv\_ep\_ops, [3261](#)



- Rcv\_fpage
  - L4::lpc::Rcv\_fpage, [1208](#)
- rcv\_task
  - L4::lpc::Rcv\_fpage, [1211](#)
- read
  - L4::lpc, [765](#)
  - L4::Vcon, [1535](#)
  - L4drivers::Ro\_register\_tmpl< BITS, BLOCK >, [1607](#)
- read\_with\_flags
  - L4::Vcon, [1536](#)
- readv
  - L4Re::Vfs::Regular\_file, [1901](#)
- ready
  - L4virtio::Virtqueue, [2285](#)
- Ready\_type
  - L4Re::Vfs::Generic\_file, [1884](#)
- realloc\_rcv\_cap
  - L4::lpc\_svr::Server\_iface, [1272](#)
  - L4Re::Util::Br\_manager, [1788](#)
- Realtime API, [257](#)
- receive
  - L4::lpc::Istream, [1167](#)
  - L4::Irq, [1293](#)
- Receiver, [610](#)
- Ref\_ptr
  - cxx::Ref\_ptr< T, CNT >, [974](#)
  - L4Re::Core::Ref\_ptr< T, CNT >, [1616](#)
- refresh
  - L4Re::Util::Video::Goos\_svr, [1855](#)
  - L4Re::Video::View, [1938](#)
- Region map API, [605](#)
- Region map interface, [560](#)
  - l4re\_rm\_attach, [562](#)
  - l4re\_rm\_attach\_srv, [563](#)
  - L4RE\_RM\_CACHING\_SHIFT, [562](#)
  - l4re\_rm\_detach, [564](#)
  - l4re\_rm\_detach\_ds, [565](#)
  - l4re\_rm\_detach\_ds\_unmap, [566](#)
  - l4re\_rm\_detach\_srv, [567](#)
  - l4re\_rm\_detach\_unmap, [567](#)
  - L4RE\_RM\_F\_ATTACH\_FLAGS, [562](#)
  - L4RE\_RM\_F\_CACHE\_BUFFERED, [562](#)
  - L4RE\_RM\_F\_CACHE\_NORMAL, [562](#)
  - L4RE\_RM\_F\_CACHE\_UNCACHED, [562](#)
  - L4RE\_RM\_F\_CACHING, [562](#)
  - L4RE\_RM\_F\_DETACH\_FREE, [562](#)
  - L4RE\_RM\_F\_EAGER\_MAP, [562](#)
  - L4RE\_RM\_F\_IN\_AREA, [562](#)
  - L4RE\_RM\_F\_KERNEL, [562](#)
  - L4RE\_RM\_F\_NO\_EAGER\_MAP, [562](#)
  - L4RE\_RM\_F\_PAGER, [562](#)
  - L4RE\_RM\_F\_R, [562](#)
  - L4RE\_RM\_F\_RESERVED, [562](#)
  - L4RE\_RM\_F\_SEARCH\_ADDR, [562](#)
  - l4re\_rm\_find, [568](#)
  - l4re\_rm\_find\_srv, [569](#)
  - l4re\_rm\_flags\_values, [561](#)
  - l4re\_rm\_free\_area, [570](#)
  - l4re\_rm\_free\_area\_srv, [571](#)
  - l4re\_rm\_get\_info, [571](#)
  - l4re\_rm\_get\_info\_srv, [572](#)
  - L4RE\_RM\_REGION\_FLAGS, [562](#)
  - l4re\_rm\_reserve\_area, [573](#)
  - l4re\_rm\_reserve\_area\_srv, [574](#)
  - l4re\_rm\_show\_lists, [575](#)
- Region\_flag\_shifts
  - L4Re::Rm, [1737](#)
  - Rm, [2327](#)
- Region\_flags
  - L4Re::Rm::F, [1757](#)
  - Rm::F, [2339](#)
- Region\_flags\_mask
  - L4Re::Rm::F, [1758](#)
  - Rm::F, [2339](#)
- register\_del\_irq
  - L4::Thread, [1443](#)
- register\_doorbell\_irq
  - L4::Thread, [1444](#)
- register\_driver\_irq
  - L4virtio::Svr::Device\_t< DATA >, [2186](#)
- register\_ds
  - L4virtio::Device, [2052](#)
  - L4virtio::Driver::Device, [2071](#)
- Register\_flags
  - L4Re::Namespace, [1714](#)
- register\_irq\_obj
  - L4::Registry\_iface, [1376](#)
  - L4Re::Util::Object\_registry, [1839](#)
- register\_obj
  - L4::Registry\_iface, [1376](#), [1377](#)
  - L4Re::Namespace, [1717](#)
  - L4Re::Util::Object\_registry, [1839](#), [1840](#)
- register\_thread
  - L4Re::Itas, [1684](#)
- Registry\_server
  - L4Re::Util::Registry\_server< LOOP\_HOOKS >, [1848](#), [1849](#)
- release
  - cxx::Ref\_ptr< T, CNT >, [975](#)
  - L4Re::Core::Ref\_ptr< T, CNT >, [1617](#)
  - L4Re::Inhibitor, [1677](#)
  - L4Re::Rm::Unique\_region< T >, [1763](#)
  - L4Re::Util::Counting\_cap\_alloc< COUNTER-TYPE, Dbg >, [1804](#)
  - L4Re::Util::Dataspace\_svr, [1812](#)
  - Rm::Unique\_region< T >, [2344](#)
- release\_cap
  - L4::Task, [1432](#)
- release\_ioport
  - L4vbus::Vbus, [2026](#)
- remove
  - cxx::Avl\_tree< Node, Get\_key, Compare >, [850](#)
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >, [906](#)
  - cxx::H\_list< T, POLICY >, [943](#)



- cxx::List\_item, [959](#)
  - L4::lpc\_svr::Timeout\_queue, [1279](#)
  - L4::Thread\_group, [1461](#)
  - L4virtio::Svr::Driver\_mem\_list\_t< DATA >, [2194](#)
  - Virtio\_vlan\_mangle, [2364](#)
- remove\_all
  - cxx::Bits::Bst< Node, Get\_key, Compare >, [923](#)
- remove\_timeout
  - L4::lpc\_svr::Server\_iface, [1272](#)
  - L4::lpc\_svr::Timeout\_queue\_hooks< HOOKS, BR\_MAN >, [1284](#)
- remove\_tree
  - cxx::Bits::Bst< Node, Get\_key, Compare >, [925](#)
- rename
  - L4Re::Vfs::Directory, [1869](#)
- rend
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, ALLOC, GET\_KEY >, [907](#)
  - cxx::Bits::Bst< Node, Get\_key, Compare >, [926](#)
- replace
  - cxx::H\_list< T, POLICY >, [943](#)
- reply\_and\_wait
  - L4::lpc::lostream, [1158](#), [1159](#)
- Reply\_compound
  - Server-Side IPC framework, [650](#)
- Reply\_mode
  - Server-Side IPC framework, [650](#)
- Reply\_separate
  - Server-Side IPC framework, [650](#)
- request\_ioport
  - L4vbus::Vbus, [2026](#)
- reserve\_area
  - L4Re::Rm, [1751](#), [1753](#)
  - Rm, [2335](#), [2336](#)
- Reserved
  - L4::Kip::Mem\_desc, [1309](#)
  - L4Re::Rm::F, [1758](#)
  - Rm::F, [2339](#)
- Reserved\_heap
  - L4::Kip::Mem\_desc, [1309](#)
- Reserved\_kernel
  - L4::Kip::Mem\_desc, [1309](#)
- Reserved\_mmio
  - L4::Kip::Mem\_desc, [1309](#)
- reset
  - L4::lpc::lostream, [1160](#)
  - L4::lpc::lstream, [1168](#)
  - L4Re::Rm::Unique\_region< T >, [1764](#)
  - Rm::Unique\_region< T >, [2344](#)
- reset\_cmd
  - L4virtio::Svr::Dev\_config, [2173](#)
- reset\_device
  - L4virtio::Svr::Console::Virtio\_con, [2158](#)
- reset\_queue
  - L4virtio::Svr::Dev\_config, [2173](#)
- reset\_queue\_config
  - L4virtio::Svr::Device\_t< DATA >, [2186](#)
- Resize
  - L4virtio::Svr::Console::Control\_message, [2115](#)
- rewind\_avail
  - L4virtio::Svr::Virtqueue, [2271](#)
- rewrite\_hdr
  - Virtio\_vlan\_mangle, [2365](#)
- Rights\_mask
  - L4::lpc::Cap< T >, [1145](#)
  - L4Re::Dataspace::F, [1633](#)
  - L4Re::Rm::F, [1758](#)
  - Rm::F, [2339](#)
- ringbuf.h
  - l4shmc\_rb\_attach\_receiver, [3018](#)
  - l4shmc\_rb\_attach\_sender, [3018](#)
  - l4shmc\_rb\_deinit\_buffer, [3019](#)
  - l4shmc\_rb\_init\_buffer, [3019](#)
  - l4shmc\_rb\_init\_receiver, [3020](#)
  - l4shmc\_rb\_receiver\_copy\_out, [3020](#)
  - l4shmc\_rb\_receiver\_notify\_done, [3021](#)
  - l4shmc\_rb\_receiver\_read\_next\_size, [3021](#)
  - l4shmc\_rb\_receiver\_wait\_for\_data, [3021](#)
  - l4shmc\_rb\_sender\_alloc\_packet, [3022](#)
  - l4shmc\_rb\_sender\_commit\_packet, [3022](#)
  - l4shmc\_rb\_sender\_next\_copy\_in, [3023](#)
  - l4shmc\_rb\_sender\_put\_data, [3023](#)
- Rm, [2322](#)
  - attach, [2327](#), [2328](#)
  - Caching\_shift, [2327](#)
  - detach, [2330](#)–[2332](#)
  - Detach\_again, [2327](#)
  - Detach\_exact, [2326](#)
  - Detach\_flags, [2326](#)
  - Detach\_keep, [2326](#)
  - Detach\_overlap, [2326](#)
  - Detach\_result, [2326](#)
  - Detached\_ds, [2327](#)
  - find, [2332](#)
  - free\_area, [2333](#)
  - get\_areas, [2334](#)
  - get\_info, [2334](#)
  - get\_regions, [2335](#)
  - Kept\_ds, [2327](#)
  - Region\_flag\_shifts, [2327](#)
  - reserve\_area, [2335](#), [2336](#)
  - Split\_ds, [2327](#)
- rm
  - L4Re::Env, [1658](#)
- Rm::Area, [2337](#)
- Rm::F, [2338](#)
  - Attach\_flags, [2338](#)
  - Attach\_mask, [2339](#)
  - Cache\_buffered, [2339](#)
  - Cache\_normal, [2339](#)
  - Cache\_uncached, [2339](#)
  - Caching\_mask, [2339](#)
  - Detach\_free, [2339](#)
  - Ds\_map\_mask, [2339](#)
  - Eager\_map, [2339](#)
  - In\_area, [2339](#)

- Kernel, [2339](#)
- No\_eager\_map, [2339](#)
- Pager, [2339](#)
- R, [2339](#)
- Region\_flags, [2339](#)
- Region\_flags\_mask, [2339](#)
- Reserved, [2339](#)
- Rights\_mask, [2339](#)
- RW, [2339](#)
- RWX, [2339](#)
- RX, [2339](#)
- Search\_addr, [2339](#)
- W, [2339](#)
- X, [2339](#)
- Rm::Region, [2340](#)
- Rm::Unique\_region< T >, [2340](#)
  - ~Unique\_region, [2343](#)
  - get, [2343](#)
  - is\_valid, [2343](#)
  - operator=, [2344](#)
  - release, [2344](#)
  - reset, [2344](#)
  - Unique\_region, [2342](#), [2343](#)
- rmdir
  - L4Re::Vfs::Directory, [1869](#)
- Ro
  - L4Re::Dataspace::F, [1632](#)
  - L4Re::Namespace, [1715](#)
- root
  - L4vbus::Vbus, [2027](#)
- round\_order
  - L4, [756](#)
- Rs
  - L4Re::Namespace, [1715](#)
- RTC driver, [82](#)
- run\_thread
  - L4::Scheduler, [1391](#)
- running
  - L4virtio::Svr::Dev\_status, [2181](#)
- Runtime\_error
  - L4::Runtime\_error, [1381](#)
- RW
  - L4Re::Dataspace::F, [1632](#)
  - L4Re::Rm::F, [1758](#)
  - Rm::F, [2339](#)
- Rw
  - L4Re::Namespace, [1715](#)
- Rws
  - L4Re::Namespace, [1715](#)
- RWX
  - L4Re::Dataspace::F, [1632](#)
  - L4Re::Rm::F, [1758](#)
  - Rm::F, [2339](#)
- RX
  - L4Re::Dataspace::F, [1632](#)
  - L4Re::Rm::F, [1758](#)
  - Rm::F, [2339](#)
- rx\_pkt
  - L4virtio::Driver::Virtio\_net\_device, [2080](#)
- rx\_queue\_size
  - L4virtio::Driver::Virtio\_net\_device, [2081](#)
- S
  - L4::Factory::S, [1105](#)
- saved\_state
  - L4vcpu::Vcpu, [2042](#)
- scan\_zero
  - cxx::Bitmap< BITS >, [880](#)
  - cxx::Bitmap\_base, [890](#)
  - L4Re::Util::Bitmap< BITS >, [1773](#)
  - L4Re::Util::Bitmap\_base, [1780](#)
- Scheduler, [362](#)
  - l4\_sched\_cpu\_set, [365](#)
  - l4\_sched\_param, [365](#)
  - L4\_SCHEDULER\_CLASS\_FIXED\_PRIO, [364](#)
  - L4\_SCHEDULER\_CLASS\_WFQ, [364](#)
  - L4\_scheduler\_classes, [364](#)
  - l4\_scheduler\_idle\_time, [366](#)
  - L4\_SCHEDULER\_IDLE\_TIME\_OP, [365](#)
  - l4\_scheduler\_info, [367](#)
  - L4\_SCHEDULER\_INFO\_OP, [364](#)
  - l4\_scheduler\_info\_with\_classes, [368](#)
  - l4\_scheduler\_is\_online, [369](#)
  - L4\_scheduler\_ops, [364](#)
  - l4\_scheduler\_run\_thread, [369](#)
  - L4\_SCHEDULER\_RUN\_THREAD\_OP, [365](#)
- scheduler
  - L4Re::Env, [1658](#)
- screen\_info
  - L4Re::Util::Video::Goos\_svr, [1857](#)
- Search\_addr
  - L4Re::Rm::F, [1757](#)
  - Rm::F, [2339](#)
- segment.h
  - fiasco\_amd64\_segment\_info, [2465](#)
  - fiasco\_amd64\_set\_fs, [2466](#), [2470](#)
  - fiasco\_amd64\_set\_segment\_base, [2467](#), [2471](#)
  - L4\_AMD64\_SEGMENT\_FS, [2465](#)
  - L4\_AMD64\_SEGMENT\_GS, [2465](#)
  - L4\_sys\_segment, [2465](#)
  - L4\_task\_ldt\_x86\_consts, [2465](#), [2474](#)
  - L4\_TASK\_LDT\_X86\_ENTRY\_SIZE, [2465](#), [2474](#)
  - L4\_TASK\_LDT\_X86\_MAX\_ENTRIES, [2465](#), [2474](#)
- send
  - L4::lpc::Ostream, [1203](#)
  - L4::Vcon, [1537](#)
  - L4virtio::Driver::Device, [2072](#)
- send\_and\_wait
  - L4virtio::Driver::Device, [2073](#)
- send\_control\_message
  - L4virtio::Svr::Console::Virtio\_con, [2159](#)
- send\_request
  - L4virtio::Driver::Block\_device, [2060](#)
- Sender, [609](#)
- Server
  - L4::Server< LOOP\_HOOKS >, [1402](#)
- Server-Side IPC framework, [649](#)

- Reply\_compound, [650](#)
- Reply\_mode, [650](#)
- Reply\_separate, [650](#)
- server\_iface
  - L4::Epiface, [1078](#)
- set
  - cxx::Bitfield< T, LSB, MSB >, [868](#)
  - L4::Kip::Mem\_desc, [1315](#)
  - L4::Poll\_timeout\_counter, [1363](#)
  - L4::Poll\_timeout\_kipclock, [1366](#)
  - I4\_sched\_cpu\_set\_t, [1558](#)
  - L4drivers::Register\_tmpl< BITS, BLOCK >, [1602](#)
  - L4Re::Video::Color\_component, [1910](#)
  - L4vbus::Gpio\_module, [1986](#)
  - L4vbus::Gpio\_pin, [1995](#)
  - L4vcpu::State, [2031](#)
  - L4virtio::Svr::Data\_buffer, [2164](#)
- set\_attr
  - L4::Vcon, [1538](#)
- set\_bit
  - cxx::Bitmap\_base, [891](#)
  - L4Re::Util::Bitmap\_base, [1781](#)
- set\_blk\_size
  - L4virtio::Svr::Block\_dev\_base< Ds\_data >, [2109](#)
- set\_config\_wce
  - L4virtio::Svr::Block\_dev\_base< Ds\_data >, [2109](#)
- set\_device\_needs\_reset
  - L4virtio::Svr::Dev\_config, [2174](#)
- set\_dirty
  - cxx::Bitfield< T, LSB, MSB >, [869](#)
- set\_discard
  - L4virtio::Svr::Block\_dev\_base< Ds\_data >, [2110](#)
- set\_fd
  - L4Re::Vfs::Fs, [1880](#)
- set\_info
  - L4Re::Video::View, [1938](#)
- set\_lock
  - L4Re::Vfs::Regular\_file, [1902](#)
- set\_mode
  - L4::lcu, [1117](#)
- set\_object\_name
  - L4::Debugger, [1068](#)
- set\_raw
  - I4\_vcon\_attr\_t, [1567](#)
- set\_rcv\_cap\_flags
  - L4Re::Util::Br\_manager, [1789](#)
- set\_server
  - L4::Epiface, [1079](#)
- set\_size\_max
  - L4virtio::Svr::Block\_dev\_base< Ds\_data >, [2110](#)
- set\_status
  - L4virtio::Device, [2053](#)
  - L4virtio::Svr::Dev\_config, [2175](#)
- set\_status\_flags
  - L4Re::Vfs::Generic\_file, [1887](#)
- set\_topology
  - L4virtio::Svr::Block\_dev\_base< Ds\_data >, [2110](#)
- set\_unshifted
  - cxx::Bitfield< T, LSB, MSB >, [869](#)
- set\_unshifted\_dirty
  - cxx::Bitfield< T, LSB, MSB >, [870](#)
- set\_viewport
  - L4Re::Video::View, [1939](#)
- set\_write\_zeroes
  - L4virtio::Svr::Block\_dev\_base< Ds\_data >, [2111](#)
- settimer
  - L4Re::Itas, [1684](#)
- setup
  - L4Re::Util::Counting\_cap\_alloc< COUNTER-  
TYPE, Dbg >, [1805](#)
  - L4vbus::Gpio\_module, [1987](#)
  - L4vbus::Gpio\_pin, [1996](#)
  - L4virtio::Virtqueue, [2286](#)
- setup\_device
  - L4virtio::Driver::Block\_device, [2060](#)
  - L4virtio::Driver::Virtio\_net\_device, [2081](#)
- setup\_queue
  - L4virtio::Svr::Device\_t< DATA >, [2186](#)
- setup\_simple
  - L4virtio::Virtqueue, [2287](#)
- sh\_flags
  - Elf32\_Shdr, [1016](#)
  - Elf64\_Shdr, [1026](#)
- sh\_type
  - Elf32\_Shdr, [1016](#)
  - Elf64\_Shdr, [1026](#)
- Shared
  - L4::Kip::Mem\_desc, [1309](#)
- Shared Memory Library, [612](#)
- I4shmc\_area\_overhead, [613](#)
- I4shmc\_area\_size, [613](#)
- I4shmc\_area\_size\_free, [614](#)
- I4shmc\_attach, [614](#)
- I4shmc\_chunk\_overhead, [615](#)
- I4shmc\_connect\_chunk\_signal, [615](#)
- I4shmc\_create, [615](#)
- I4shmc\_get\_client\_nr, [616](#)
- I4shmc\_get\_initialized\_clients, [616](#)
- I4shmc\_mark\_client\_initialized, [617](#)
- Shared\_cap
  - L4Re, [777](#)
  - L4Re::Util, [797](#)
- shared\_cap
  - L4Re, [778](#)
  - L4Re::Util, [797](#)
- Shared\_del\_cap
  - L4Re, [778](#)
  - L4Re::Util, [798](#)
- shared\_del\_cap
  - L4Re, [778](#)
  - L4Re::Util, [798](#)
- SHF\_ALLOC
  - ELF binary format, [715](#)
- SHF\_ARM\_COMDEF
  - ELF binary format, [715](#)
- SHF\_ARM\_ENTRYSECT

- ELF binary format, [715](#)
- SHF\_EXECINSTR
  - ELF binary format, [715](#)
- SHF\_GROUP
  - ELF binary format, [715](#)
- SHF\_INFO\_LINK
  - ELF binary format, [715](#)
- SHF\_MASKOS
  - ELF binary format, [715](#)
- SHF\_MASKPROC
  - ELF binary format, [715](#)
- SHF\_MERGE
  - ELF binary format, [715](#)
- SHF\_OS\_NONCONFORMING
  - ELF binary format, [715](#)
- SHF\_STRINGS
  - ELF binary format, [715](#)
- SHF\_TLS
  - ELF binary format, [715](#)
- SHF\_WRITE
  - ELF binary format, [715](#)
- shift
  - L4Re::Video::Color\_component, [1910](#)
- Shift\_type
  - cxx::Bitfield< T, LSB, MSB >, [867](#)
- SHN\_ABS
  - ELF binary format, [717](#)
- SHN\_COMMON
  - ELF binary format, [717](#)
- SHN\_HIPROC
  - ELF binary format, [717](#)
- SHN\_HIRESERVE
  - ELF binary format, [717](#)
- SHN\_LOPROC
  - ELF binary format, [717](#)
- SHN\_LORESERVE
  - ELF binary format, [717](#)
- SHN\_UNDEF
  - ELF binary format, [717](#)
- SHT\_DYNAMIC
  - ELF binary format, [717](#)
- SHT\_DYNSYM
  - ELF binary format, [717](#)
- SHT\_FINI\_ARRAY
  - ELF binary format, [717](#)
- SHT\_GROUP
  - ELF binary format, [717](#)
- SHT\_HASH
  - ELF binary format, [717](#)
- SHT\_HIOS
  - ELF binary format, [717](#)
- SHT\_HIPROC
  - ELF binary format, [717](#)
- SHT\_HIUSER
  - ELF binary format, [717](#)
- SHT\_INIT\_ARRAY
  - ELF binary format, [717](#)
- SHT\_LOOS
  - ELF binary format, [717](#)
- SHT\_LOPROC
  - ELF binary format, [717](#)
- SHT\_LOUSER
  - ELF binary format, [717](#)
- SHT\_NOBITS
  - ELF binary format, [717](#)
- SHT\_NOTE
  - ELF binary format, [717](#)
- SHT\_NULL
  - ELF binary format, [717](#)
- SHT\_NUM
  - ELF binary format, [717](#)
- SHT\_PREINIT\_ARRAY
  - ELF binary format, [717](#)
- SHT\_PROGBITS
  - ELF binary format, [717](#)
- SHT\_REL
  - ELF binary format, [717](#)
- SHT\_RELA
  - ELF binary format, [717](#)
- SHT\_SHLIB
  - ELF binary format, [717](#)
- SHT\_STRTAB
  - ELF binary format, [717](#)
- SHT\_SYMTAB
  - ELF binary format, [717](#)
- SHT\_SYMTAB\_SHNDX
  - ELF binary format, [717](#)
- shutdown
  - L4::Uart, [1520](#)
  - L4::Uart\_apb, [1526](#)
- si
  - l4\_vcpu\_regs\_t, [1572](#)
- sigaction
  - L4Re::Itas, [1685](#)
- sigaltstack
  - L4Re::Itas, [1685](#)
- Sigma0 API, [639](#)
  - l4sigma0\_debug\_dump, [640](#)
  - L4SIGMA0\_IPCERROR, [640](#)
  - l4sigma0\_map\_anypage, [640](#)
  - l4sigma0\_map\_errstr, [641](#)
  - l4sigma0\_map\_iomem, [641](#)
  - l4sigma0\_map\_kip, [642](#)
  - l4sigma0\_map\_mem, [642](#)
  - L4SIGMA0\_NOFPAGE, [640](#)
  - L4SIGMA0\_NOTALIGNED, [640](#)
  - L4SIGMA0\_OK, [640](#)
  - l4sigma0\_return\_flags\_t, [640](#)
  - L4SIGMA0\_SMALLERFPAGE, [640](#)
- Sigma0, the Root-Pager, [53](#)
- signal
  - L4Re::Parent, [1725](#)
- Signals, [630](#)
  - l4shmc\_add\_signal, [631](#)
  - l4shmc\_attach\_signal, [631](#)
  - l4shmc\_check\_magic, [632](#)

- l4shmc\_get\_signal, [632](#)
- l4shmc\_signal\_cap, [633](#)
- sigpending
  - L4Re::Itas, [1686](#)
- sigprocmask
  - L4Re::Itas, [1687](#)
- Single
  - L4::lpc::Snd\_fpage, [1220](#)
- size
  - L4::Kip::Mem\_desc, [1316](#)
  - L4Re::Dataspace, [1630](#)
  - L4Re::Video::Color\_component, [1911](#)
  - L4virtio::Svr::Driver\_mem\_region\_t< DATA >, [2201](#)
- skip
  - L4::lpc::Istream, [1168](#)
  - L4virtio::Svr::Data\_buffer, [2164](#)
- slab\_size
  - cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc >, [858](#)
  - cxx::Base\_slab\_static< Obj\_size, Slab\_size, Max\_free, Alloc >, [863](#)
- Small C++ Template Library, [644](#)
  - clamp, [646](#)
  - max, [646](#)
  - min, [647](#)
  - operator new, [647](#)
- Small\_buf
  - L4::lpc::Small\_buf, [1214](#), [1215](#)
- Smart\_cap
  - L4::Smart\_cap< T, SMART >, [1422](#)
- snd\_base
  - L4::Cap\_base, [1055](#)
- Snd\_fpage
  - L4::lpc::Snd\_fpage, [1220](#), [1221](#)
- Space\_attrb
  - L4Re::Dma\_space, [1641](#)
- Spaces and Mappings, [26](#)
- Special
  - L4::lpc::Gen\_fpage, [1149](#)
- Split\_ds
  - L4Re::Rm, [1737](#)
  - Rm, [2327](#)
- Src\_dev\_handle
  - L4vbus::lcu, [2001](#)
- Src\_types
  - L4vbus::lcu, [2001](#)
- ss
  - l4\_exc\_regs\_t, [1547](#)
- stack
  - L4Re::Video::View, [1940](#)
- start
  - L4::Kip::Mem\_desc, [1316](#)
  - L4virtio::Svr::Request\_processor, [2205](#), [2207](#)
- start\_request
  - L4virtio::Driver::Block\_device, [2062](#)
- starts\_with
  - cxx::String, [995](#)
- startup
  - L4::Uart, [1520](#)
  - L4::Uart\_apb, [1526](#)
- State
  - L4vcpu::State, [2030](#)
- state
  - L4vcpu::Vcpu, [2042](#), [2043](#)
- state\_transitions
  - L4virtio::Svr::Console::Port, [2143](#)
- stats\_time
  - L4::Thread, [1445](#)
- status
  - L4::Lock\_guard, [1341](#)
  - L4virtio::Svr::Dev\_config, [2175](#)
- STB\_GLOBAL
  - ELF binary format, [718](#)
- STB\_HIOS
  - ELF binary format, [718](#)
- STB\_HIPROC
  - ELF binary format, [718](#)
- STB\_LOCAL
  - ELF binary format, [718](#)
- STB\_LOOS
  - ELF binary format, [718](#)
- STB\_LOPROC
  - ELF binary format, [718](#)
- STB\_WEAK
  - ELF binary format, [718](#)
- Str\_cp\_in
  - L4::lpc::Str\_cp\_in< T >, [1226](#)
- str\_cp\_in
  - L4::lpc, [765](#)
- String
  - cxx::String, [993](#)
- Strong
  - L4Re::Namespace, [1715](#)
- STT\_FILE
  - ELF binary format, [718](#)
- STT\_FUNC
  - ELF binary format, [718](#)
- STT\_HIOS
  - ELF binary format, [718](#)
- STT\_HIPROC
  - ELF binary format, [718](#)
- STT\_LOOS
  - ELF binary format, [718](#)
- STT\_LOPROC
  - ELF binary format, [718](#)
- STT\_NOTYPE
  - ELF binary format, [718](#)
- STT\_OBJECT
  - ELF binary format, [718](#)
- STT\_SECTION
  - ELF binary format, [718](#)
- sub\_type
  - L4::Kip::Mem\_desc, [1317](#)
- Super\_pages
  - L4Re::Mem\_alloc, [1699](#)

- supports
  - L4::Meta, [1346](#)
- Switch\_factory, [2345](#)
  - op\_create, [2349](#)
- switch\_log
  - L4::Debugger, [1069](#)
- switch\_to
  - L4::Thread, [1446](#)
- symlink
  - L4Re::Vfs::Directory, [1869](#)
- system\_shutdown
  - L4::Platform\_control, [1360](#)
- system\_suspend
  - L4::Platform\_control, [1361](#)
- tag
  - L4::lpc::Istream, [1170](#)
  - L4::lpc::Ostream, [1204](#)
  - L4::lpc::Varg, [1232](#)
- take
  - L4Re::Util::Counting\_cap\_alloc< COUNTER-  
TYPE, Dbg >, [1805](#)
  - L4Re::Util::Dataspace\_svr, [1812](#)
- Task, [373](#)
  - L4\_FP\_ALL\_SPACES, [374](#)
  - L4\_FP\_DELETE\_OBJ, [374](#)
  - L4\_FP\_OTHER\_SPACES, [374](#)
  - I4\_task\_add\_ku\_mem, [375](#)
  - I4\_task\_cap\_equal, [375](#)
  - I4\_task\_cap\_valid, [377](#)
  - I4\_task\_delete\_obj, [378](#)
  - I4\_task\_map, [379](#)
  - I4\_task\_release\_cap, [380](#)
  - I4\_task\_unmap, [382](#)
  - I4\_task\_unmap\_batch, [384](#)
  - I4\_task\_vgicc\_map, [385](#)
  - I4\_unmap\_flags\_t, [374](#)
- task
  - L4Re::Env, [1659](#)
  - L4vcpu::Vcpu, [2043](#)
- test
  - L4::Poll\_timeout\_kipclock, [1366](#)
- test.mk - Test Application Role, [41](#)
- The L4Re IPC Framework, [648](#)
- Thread, [386](#)
  - I4\_thread\_arm\_set\_tpidruro, [390](#)
  - L4\_THREAD\_CONTROL\_ALIEN, [389](#)
  - L4\_THREAD\_CONTROL\_BIND\_TASK, [389](#)
  - L4\_thread\_control\_flags, [388](#)
  - L4\_THREAD\_CONTROL\_MR\_IDX\_BIND\_TASK, [389](#)
  - L4\_THREAD\_CONTROL\_MR\_IDX\_BIND\_UTCB, [389](#)
  - L4\_THREAD\_CONTROL\_MR\_IDX\_EXC\_HANDLER, [389](#)
  - L4\_THREAD\_CONTROL\_MR\_IDX\_FLAG\_VALS, [389](#)
  - L4\_THREAD\_CONTROL\_MR\_IDX\_FLAGS, [389](#)
  - L4\_THREAD\_CONTROL\_MR\_IDX\_PAGER, [389](#)
  - L4\_thread\_control\_mr\_indices, [389](#)
  - L4\_THREAD\_CONTROL\_SET\_EXC\_HANDLER, [389](#)
  - L4\_THREAD\_CONTROL\_SET\_PAGER, [388](#)
  - I4\_thread\_ex\_regs, [391](#)
  - L4\_THREAD\_EX\_REGS\_ARCH\_MASK, [389](#)
  - L4\_THREAD\_EX\_REGS\_ARM64\_SET\_EL\_EL0, [390](#)
  - L4\_THREAD\_EX\_REGS\_ARM64\_SET\_EL\_EL1, [390](#)
  - L4\_THREAD\_EX\_REGS\_ARM64\_SET\_EL\_KEEP, [390](#)
  - L4\_THREAD\_EX\_REGS\_ARM64\_SET\_EL\_MASK, [390](#)
  - L4\_THREAD\_EX\_REGS\_ARM\_SET\_EL\_EL0, [390](#)
  - L4\_THREAD\_EX\_REGS\_ARM\_SET\_EL\_EL1, [390](#)
  - L4\_THREAD\_EX\_REGS\_ARM\_SET\_EL\_KEEP, [390](#)
  - L4\_THREAD\_EX\_REGS\_ARM\_SET\_EL\_MASK, [390](#)
  - L4\_THREAD\_EX\_REGS\_CANCEL, [389](#)
  - L4\_thread\_ex\_regs\_flags, [389](#)
  - L4\_thread\_ex\_regs\_flags\_arm, [389](#)
  - L4\_thread\_ex\_regs\_flags\_arm64, [390](#)
  - I4\_thread\_ex\_regs\_ret, [392](#)
  - I4\_thread\_ex\_regs\_ret\_u, [393](#)
  - L4\_THREAD\_EX\_REGS\_TRIGGER\_EXCEPTION, [389](#)
  - I4\_thread\_ex\_regs\_u, [394](#)
  - I4\_thread\_modify\_sender\_add, [395](#)
  - I4\_thread\_modify\_sender\_commit, [396](#)
  - I4\_thread\_modify\_sender\_start, [397](#)
  - I4\_thread\_register\_del\_irq, [398](#)
  - I4\_thread\_register\_doorbell\_irq, [399](#)
  - I4\_thread\_stats\_time, [400](#)
  - I4\_thread\_switch, [401](#)
  - I4\_thread\_vcpu\_control, [401](#)
  - I4\_thread\_vcpu\_control\_ext, [402](#)
  - I4\_thread\_vcpu\_control\_ext\_u, [403](#)
  - I4\_thread\_vcpu\_control\_u, [405](#)
  - I4\_thread\_vcpu\_resume\_commit, [406](#)
  - I4\_thread\_vcpu\_resume\_start, [407](#)
  - I4\_thread\_yield, [408](#)
- Thread control, [409](#)
  - I4\_thread\_control\_alien, [410](#)
  - I4\_thread\_control\_bind, [411](#)
  - I4\_thread\_control\_commit, [412](#)
  - I4\_thread\_control\_exc\_handler, [413](#)
  - I4\_thread\_control\_pager, [414](#)
  - I4\_thread\_control\_start, [414](#)
- Thread Control Registers (TCRs), [280](#)
- Thread groups, [420](#)
  - I4\_thread\_group\_add, [421](#)
  - I4\_thread\_group\_remove, [421](#)
- thread.h
  - \_\_L4UTIL\_THREAD\_FUNC, [3401](#)

- throw\_error
  - L4Re, [792](#)
- throw\_ipc\_exception
  - IPC Helpers, [472, 473](#)
- timed\_out
  - L4::Poll\_timeout\_counter, [1364](#)
  - L4::Poll\_timeout\_kipclock, [1367](#)
- timeout
  - L4::lpc\_svr::Timeout, [1276](#)
- timeout\_expired
  - L4::lpc\_svr::Timeout\_queue, [1279](#)
- Timeouts, [243](#)
  - l4\_ipc\_timeout, [245](#)
  - L4\_IPC\_TIMEOUT\_0, [244](#)
  - l4\_rcv\_timeout, [245](#)
  - l4\_snd\_timeout, [246](#)
  - l4\_timeout, [246](#)
  - l4\_timeout\_abs, [247](#)
  - l4\_timeout\_get, [248](#)
  - l4\_timeout\_is\_absolute, [249](#)
  - l4\_timeout\_rel, [249](#)
  - l4\_timeout\_rel\_get, [250](#)
  - l4\_timeout\_s, [245](#)
  - l4\_timeout\_t, [245](#)
  - L4\_TIMEOUT\_US\_MAX, [244](#)
  - l4\_utcb\_mr64\_idx, [250](#)
- Timestamp Counter, [660](#)
  - l4\_busy\_wait\_ns, [661](#)
  - l4\_busy\_wait\_us, [662](#)
  - l4\_calibrate\_tsc, [662](#)
  - l4\_get\_hz, [663](#)
  - l4\_ns\_to\_tsc, [663](#)
  - l4\_rdpmc, [664](#)
  - l4\_rdpmc\_32, [664](#)
  - l4\_rdtsc, [664](#)
  - l4\_rdtsc\_32, [665](#)
  - l4\_tsc\_init, [665](#)
  - l4\_tsc\_to\_ns, [666](#)
  - l4\_tsc\_to\_s\_and\_ns, [666](#)
  - l4\_tsc\_to\_us, [667](#)
- To\_default
  - L4Re::Namespace, [1714](#)
- To\_device
  - L4Re::Dma\_space, [1641](#)
- to\_irq
  - L4vbus::Gpio\_pin, [1996](#)
- To\_non\_blocking
  - L4Re::Namespace, [1714](#)
- total\_objects
  - cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc >, [859](#)
  - cxx::Base\_slab\_static< Obj\_size, Slab\_size, Max\_free, Alloc >, [864](#)
- total\_size
  - L4virtio::Virtqueue, [2288, 2289](#)
- trigger
  - L4::Triggerable, [1465](#)
- trunc\_order
  - L4, [757](#)
- Trusted
  - L4Re::Namespace, [1715](#)
- Tutorial, [7](#)
- tx
  - L4virtio::Driver::Virtio\_net\_device, [2083](#)
- tx\_queue\_size
  - L4virtio::Driver::Virtio\_net\_device, [2084](#)
- Type
  - L4::lpc::Gen\_fpage, [1148](#)
- type
  - L4::lpc::Varg, [1232](#)
  - L4::Kip::Mem\_desc, [1317](#)
  - L4Re::Vfs::Be\_file\_system, [1865](#)
  - L4Re::Vfs::File\_system, [1876](#)
- types
  - L4\_TYPES\_FLAGS\_OPS\_DEF, [3159](#)
- types.h
  - l4\_capability\_next, [2814](#)
- unbind
  - L4::lcu, [1118](#)
  - L4::lommu, [1130](#)
- Uncacheable
  - L4Re::Dataspace::F, [1633](#)
- Uncached
  - L4::lpc::Snd\_fpage, [1219](#)
- Undefined
  - L4::Kip::Mem\_desc, [1309](#)
- Unique\_cap
  - L4Re, [780](#)
  - L4Re::Util, [799](#)
- unique\_cap
  - L4Re, [780](#)
  - L4Re::Util, [800](#)
- Unique\_del\_cap
  - L4Re, [780](#)
  - L4Re::Util, [800](#)
- unique\_del\_cap
  - L4Re, [782](#)
  - L4Re::Util, [801](#)
- Unique\_region
  - L4Re::Rm::Unique\_region< T >, [1761, 1762](#)
  - Rm::Unique\_region< T >, [2342, 2343](#)
- unlink
  - L4Re::Namespace, [1718](#)
  - L4Re::Vfs::Directory, [1870](#)
- unlock\_all\_locks
  - L4Re::Vfs::Be\_file, [1862](#)
  - L4Re::Vfs::Generic\_file, [1888](#)
- unmap
  - L4::Task, [1433](#)
  - L4Re::Dma\_space, [1644](#)
- unmap\_batch
  - L4::Task, [1434](#)
- unmask
  - L4::Irq, [1294](#)
  - L4::Irq\_eoi, [1297](#)
- unregister\_obj



- L4::Registry\_iface, [1378](#)
  - L4Re::Util::Object\_registry, [1841](#)
- unregister\_thread
  - L4Re::Itas, [1688](#)
- up
  - L4::Semaphore, [1397](#)
- used\_align
  - L4virtio::Virtqueue, [2289](#)
- Used\_elem
  - L4virtio::Virtqueue::Used\_elem, [2301](#)
- used\_size
  - L4virtio::Virtqueue, [2290](#)
- utcb\_area
  - L4Re::Env, [1659](#)
- Utility Functions, [650](#)
  - l4\_sleep, [653](#)
  - l4\_touch\_ro, [653](#)
  - l4\_touch\_rw, [654](#)
  - l4\_usleep, [654](#)
  - l4util\_micros2l4to, [655](#)
  - l4util\_splitlog2\_hdl, [655](#)
  - l4util\_splitlog2\_size, [656](#)
- Uvmm, the virtual machine monitor, [73](#)
- uvmm\_dtg The device tree generator for Uvmm, [97](#)
- V\_flags
  - L4Re::Video::View, [1937](#)
- val
  - cxx::Bitfield< T, LSB, MSB >, [871](#)
- val\_dirty
  - cxx::Bitfield< T, LSB, MSB >, [871](#)
- val\_unshifted
  - cxx::Bitfield< T, LSB, MSB >, [872](#)
- valid
  - cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COM-PARE, ALLOC, GET\_KEY >::Node, [909](#)
  - L4virtio::Svr::Virtqueue::Head\_desc, [2273](#)
- validate
  - L4::Cap\_base, [1056](#), [1057](#)
- value
  - L4::lpc::Varg, [1232](#)
- Varg\_list\_ref
  - L4::lpc::Varg\_list\_ref, [1237](#)
- vbe\_ctrl\_info
  - l4util\_l4mod\_info, [1960](#)
- Vbus API, [608](#)
- vbus.h
  - L4VBUS\_ICU\_SRC\_DEV\_HANDLE, [3407](#)
  - l4vbus\_icu\_src\_types, [3407](#)
  - L4VBUS\_NULL, [3407](#)
  - L4VBUS\_ROOT\_BUS, [3407](#)
- vbus\_interfaces.h
  - L4VBUS\_IFACE\_SHIFT, [3414](#)
  - l4vbus\_iface\_type\_t, [3413](#), [3414](#)
  - L4VBUS\_INTERFACE\_BUS, [3414](#)
  - L4VBUS\_INTERFACE\_GENERIC, [3414](#)
  - L4VBUS\_INTERFACE\_GPIO, [3414](#)
  - L4VBUS\_INTERFACE\_ICU, [3414](#)
  - L4VBUS\_INTERFACE\_PCI, [3414](#)
  - L4VBUS\_INTERFACE\_PCIDEV, [3414](#)
  - L4VBUS\_INTERFACE\_PM, [3414](#)
  - l4vbus\_subinterface\_supported, [3414](#)
- vbus\_types.h
  - L4VBUS\_DEVICE\_F\_CHILDREN, [3420](#)
  - l4vbus\_device\_flags\_t, [3420](#)
  - L4VBUS\_RESOURCE\_BUS, [3421](#)
  - L4VBUS\_RESOURCE\_DMA\_DOMAIN, [3421](#)
  - L4VBUS\_RESOURCE\_F\_MEM\_CACHEABLE, [3421](#)
  - L4VBUS\_RESOURCE\_F\_MEM\_MMIO\_READ, [3421](#)
  - L4VBUS\_RESOURCE\_F\_MEM\_MMIO\_WRITE, [3421](#)
  - L4VBUS\_RESOURCE\_F\_MEM\_PREFETCHABLE, [3421](#)
  - L4VBUS\_RESOURCE\_F\_MEM\_R, [3420](#)
  - L4VBUS\_RESOURCE\_F\_MEM\_W, [3421](#)
  - l4vbus\_resource\_flags\_t, [3420](#)
  - L4VBUS\_RESOURCE\_GPIO, [3421](#)
  - L4VBUS\_RESOURCE\_INVALID, [3421](#)
  - L4VBUS\_RESOURCE\_IRQ, [3421](#)
  - L4VBUS\_RESOURCE\_MAX, [3421](#)
  - L4VBUS\_RESOURCE\_MEM, [3421](#)
  - L4VBUS\_RESOURCE\_PORT, [3421](#)
  - l4vbus\_resource\_type\_t, [3421](#)
- vcon.h
  - L4\_vcon\_read\_flags, [3312](#)
  - L4\_VCON\_READ\_SIZE\_MASK, [3312](#)
  - L4\_VCON\_READ\_STAT\_BREAK, [3312](#)
  - L4\_VCON\_READ\_STAT\_DONE, [3312](#)
- vCPU API, [415](#)
  - L4\_VCPU\_F\_EXCEPTIONS, [418](#)
  - L4\_VCPU\_F\_FPU\_ENABLED, [418](#)
  - L4\_VCPU\_F\_IRQ, [418](#)
  - L4\_VCPU\_F\_PAGE\_FAULTS, [418](#)
  - L4\_VCPU\_F\_USER\_MODE, [418](#)
  - L4\_VCPU\_OFFSET\_EXT\_INFOS, [419](#), [420](#)
  - L4\_VCPU\_OFFSET\_EXT\_STATE, [419](#)
  - L4\_VCPU\_SF\_IRQ\_PENDING, [420](#)
  - L4\_vcpu\_state\_flags, [417](#)
  - L4\_vcpu\_state\_offset, [418](#), [419](#)
  - L4\_vcpu\_sticky\_flags, [420](#)
- vCPU Support Library, [732](#)
  - l4vcpu\_irq\_disable, [733](#)
  - l4vcpu\_irq\_disable\_save, [734](#)
  - l4vcpu\_irq\_enable, [734](#)
  - l4vcpu\_irq\_restore, [735](#)
  - l4vcpu\_is\_irq\_entry, [736](#)
  - l4vcpu\_is\_page\_fault\_entry, [737](#)
  - l4vcpu\_print\_state, [737](#)
  - l4vcpu\_wait\_for\_event, [738](#)
- vcpu.h
  - l4\_vcpu\_check\_version, [3427](#)
- vcpu\_control
  - L4::Thread, [1446](#)
- vcpu\_control\_ext
  - L4::Thread, [1447](#)



- vcpu\_resume\_commit
  - L4::Thread, [1448](#)
- vcpu\_resume\_start
  - L4::Thread, [1450](#)
- version
  - l4\_vcpu\_state\_t, [1576](#)
- vgicc\_map
  - L4::Vm, [1543](#)
- vicu
  - L4vbus::lcu, [2001](#)
- Video API, [577](#), [606](#)
  - F\_l4re\_video\_goos\_auto\_refresh, [579](#)
  - F\_l4re\_video\_goos\_dynamic\_buffers, [579](#)
  - F\_l4re\_video\_goos\_dynamic\_views, [579](#)
  - F\_l4re\_video\_goos\_pointer, [579](#)
  - F\_l4re\_video\_view\_above, [579](#)
  - F\_l4re\_video\_view\_dyn\_allocated, [579](#)
  - F\_l4re\_video\_view\_flags\_mask, [579](#)
  - F\_l4re\_video\_view\_none, [579](#)
  - F\_l4re\_video\_view\_set\_background, [579](#)
  - F\_l4re\_video\_view\_set\_buffer, [579](#)
  - F\_l4re\_video\_view\_set\_buffer\_offset, [579](#)
  - F\_l4re\_video\_view\_set\_bytes\_per\_line, [579](#)
  - F\_l4re\_video\_view\_set\_flags, [579](#)
  - F\_l4re\_video\_view\_set\_pixel, [579](#)
  - F\_l4re\_video\_view\_set\_position, [579](#)
  - l4re\_video\_goos\_create\_buffer, [579](#)
  - l4re\_video\_goos\_create\_view, [580](#)
  - l4re\_video\_goos\_delete\_buffer, [580](#)
  - l4re\_video\_goos\_delete\_view, [580](#)
  - l4re\_video\_goos\_get\_static\_buffer, [581](#)
  - l4re\_video\_goos\_get\_view, [581](#)
  - l4re\_video\_goos\_info, [581](#)
  - l4re\_video\_goos\_info\_flags\_t, [578](#)
  - l4re\_video\_goos\_refresh, [582](#)
  - l4re\_video\_view\_get\_info, [582](#)
  - l4re\_video\_view\_info\_flags\_t, [579](#)
  - l4re\_video\_view\_refresh, [582](#)
  - l4re\_video\_view\_set\_info, [583](#)
  - l4re\_video\_view\_set\_viewport, [583](#)
  - l4re\_video\_view\_stack, [584](#)
  - l4re\_video\_view\_t, [578](#)
- view
  - L4Re::Video::Goos, [1921](#)
- view\_info
  - L4Re::Util::Video::Goos\_svr, [1857](#)
- Virtio Net Switch, [731](#)
- Virtio Net Switch, a virtual network switch, [69](#)
- Virtio\_con
  - L4virtio::Svr::Console::Virtio\_con, [2148](#)
- Virtio\_net, [2349](#)
  - notify\_queue, [2354](#)
- Virtio\_net\_request, [2354](#)
  - drop\_requests, [2355](#)
  - get\_request, [2356](#)
- Virtio\_switch, [2357](#)
  - add\_monitor\_port, [2359](#)
  - add\_port, [2359](#)
  - check\_ports, [2360](#)
  - handle\_l4virtio\_port\_tx, [2360](#)
  - port\_available, [2361](#)
  - Virtio\_switch, [2358](#)
- Virtio\_vlan\_mangle, [2362](#)
  - add, [2363](#)
  - copy\_pkt, [2363](#)
  - remove, [2364](#)
  - rewrite\_hdr, [2365](#)
  - Virtio\_vlan\_mangle, [2363](#)
- Virtual Console, [422](#)
  - l4\_vcon\_attr\_t, [424](#)
  - L4\_VCON\_ECHO, [424](#)
  - l4\_vcon\_get\_attr, [425](#)
  - l4\_vcon\_get\_attr\_u, [426](#)
  - L4\_vcon\_i\_flags, [424](#)
  - L4\_VCON\_ICANON, [424](#)
  - L4\_VCON\_ICRNL, [424](#)
  - L4\_VCON\_IGNCR, [424](#)
  - L4\_VCON\_INLCR, [424](#)
  - L4\_vcon\_l\_flags, [424](#)
  - L4\_vcon\_o\_flags, [424](#)
  - L4\_VCON\_OCRNL, [425](#)
  - L4\_VCON\_ONLCR, [425](#)
  - L4\_VCON\_ONLRET, [425](#)
  - l4\_vcon\_read, [427](#)
  - L4\_VCON\_READ\_SIZE, [425](#)
  - l4\_vcon\_read\_u, [428](#)
  - l4\_vcon\_read\_with\_flags, [429](#)
  - l4\_vcon\_send, [430](#)
  - l4\_vcon\_send\_u, [431](#)
  - l4\_vcon\_set\_attr, [432](#)
  - l4\_vcon\_set\_attr\_raw, [433](#)
  - l4\_vcon\_set\_attr\_u, [433](#)
  - L4\_vcon\_size\_consts, [425](#)
  - l4\_vcon\_write, [434](#)
  - L4\_VCON\_WRITE\_SIZE, [425](#)
  - l4\_vcon\_write\_u, [435](#)
- Virtual Machines, [303](#)
- Virtual Registers (UTCBs), [270](#)
  - l4\_utcb\_br, [273](#)
  - l4\_utcb\_mr, [273](#)
  - l4\_utcb\_t, [272](#)
  - l4\_utcb\_tcr, [274](#)
- VM API for SVM, [304](#)
- VM API for TZ, [325](#)
- VM API for VMX, [305](#)
  - L4\_VM\_VMX\_BASIC\_REG, [310](#)
  - L4\_vm\_vmx\_caps\_regs, [309](#)
  - l4\_vm\_vmx\_clear, [311](#)
  - L4\_VM\_VMX\_CR0\_FIXED0\_REG, [310](#)
  - L4\_VM\_VMX\_CR0\_FIXED1\_REG, [310](#)
  - L4\_VM\_VMX\_CR4\_FIXED0\_REG, [310](#)
  - L4\_VM\_VMX\_CR4\_FIXED1\_REG, [310](#)
  - L4\_vm\_vmx\_dfl1\_regs, [310](#)
  - L4\_VM\_VMX\_ENTRY\_CTL5\_DFL1\_REG, [310](#)
  - L4\_VM\_VMX\_EPT\_VPID\_CAP\_REG, [310](#)
  - L4\_VM\_VMX\_EXIT\_CTL5\_DFL1\_REG, [310](#)

[l4\\_vm\\_vmx\\_field\\_len](#), 312  
[l4\\_vm\\_vmx\\_field\\_order](#), 313  
[l4\\_vm\\_vmx\\_get\\_caps](#), 313  
[l4\\_vm\\_vmx\\_get\\_caps\\_default1](#), 315  
[l4\\_vm\\_vmx\\_get\\_cr2\\_index](#), 315  
[l4\\_vm\\_vmx\\_get\\_hw\\_vmcs](#), 315  
[L4\\_VM\\_VMX\\_MISC\\_REG](#), 310  
[L4\\_VM\\_VMX\\_NESTED\\_REVISION](#), 310  
[L4\\_VM\\_VMX\\_NUM\\_CAPS\\_REGS](#), 310  
[L4\\_VM\\_VMX\\_NUM\\_DFL1\\_REGS](#), 310  
[L4\\_VM\\_VMX\\_PINBASED\\_CTL5\\_DFL1\\_REG](#), 310  
[L4\\_VM\\_VMX\\_PROCBASED\\_CTL52\\_REG](#), 310  
[L4\\_VM\\_VMX\\_PROCBASED\\_CTL5\\_DFL1\\_REG](#), 310  
[l4\\_vm\\_vmx\\_ptr\\_load](#), 316  
[l4\\_vm\\_vmx\\_read](#), 317  
[l4\\_vm\\_vmx\\_read\\_16](#), 317  
[l4\\_vm\\_vmx\\_read\\_32](#), 318  
[l4\\_vm\\_vmx\\_read\\_64](#), 319  
[l4\\_vm\\_vmx\\_read\\_nat](#), 319  
[l4\\_vm\\_vmx\\_set\\_hw\\_vmcs](#), 320  
[L4\\_vm\\_vmx\\_sw\\_fields](#), 310  
[L4\\_VM\\_VMX\\_TRUE\\_ENTRY\\_CTL5\\_REG](#), 310  
[L4\\_VM\\_VMX\\_TRUE\\_EXIT\\_CTL5\\_REG](#), 310  
[L4\\_VM\\_VMX\\_TRUE\\_PINBASED\\_CTL5\\_REG](#), 310  
[L4\\_VM\\_VMX\\_TRUE\\_PROCBASED\\_CTL5\\_REG](#), 310  
[l4\\_vm\\_vmx\\_vcpu\\_state\\_t](#), 307  
[l4\\_vm\\_vmx\\_vcpu\\_vmcs\\_t](#), 307  
[L4\\_VM\\_VMX\\_VMCS\\_CR2](#), 311  
[L4\\_VM\\_VMX\\_VMCS\\_ENUM\\_REG](#), 310  
[L4\\_VM\\_VMX\\_VMCS\\_MSR\\_CSTAR](#), 311  
[L4\\_VM\\_VMX\\_VMCS\\_MSR\\_KERNEL\\_GS\\_BASE](#), 311  
[L4\\_VM\\_VMX\\_VMCS\\_MSR\\_LSTAR](#), 311  
[L4\\_VM\\_VMX\\_VMCS\\_MSR\\_STAR](#), 311  
[L4\\_VM\\_VMX\\_VMCS\\_MSR\\_SYSCALL\\_MASK](#), 311  
[L4\\_VM\\_VMX\\_VMCS\\_MSR\\_TSC\\_AUX](#), 311  
[L4\\_VM\\_VMX\\_VMCS\\_NAT\\_ARG0](#), 311  
[L4\\_VM\\_VMX\\_VMCS\\_NAT\\_ARG1](#), 311  
[L4\\_VM\\_VMX\\_VMCS\\_NAT\\_ARG2](#), 311  
[L4\\_VM\\_VMX\\_VMCS\\_NAT\\_ARG3](#), 311  
[L4\\_VM\\_VMX\\_VMCS\\_SIZE\\_DIRTY\\_BITMAP](#), 311  
[L4\\_VM\\_VMX\\_VMCS\\_SIZE\\_VALUES](#), 311  
[l4\\_vm\\_vmx\\_vmcs\\_sizes](#), 311  
[L4\\_VM\\_VMX\\_VMCS\\_XCR0](#), 311  
[l4\\_vm\\_vmx\\_write](#), 321  
[l4\\_vm\\_vmx\\_write\\_16](#), 322  
[l4\\_vm\\_vmx\\_write\\_32](#), 323  
[l4\\_vm\\_vmx\\_write\\_64](#), 323  
[l4\\_vm\\_vmx\\_write\\_nat](#), 324  
[l4\\_vmx\\_offset\\_table\\_t](#), 308

## W

[L4Re::Dataspace::F](#), 1632  
[L4Re::Rm::F](#), 1758  
[Rm::F](#), 2339

## W\_bits

[cxx::Bitmap\\_base](#), 884  
[L4Re::Util::Bitmap\\_base](#), 1777

## wait

[L4::lpc::lstream](#), 1171, 1172  
[L4::lrc](#), 1295  
[L4virtio::Driver::Device](#), 2074

## wait\_for\_event

[L4vcpu::Vcpu](#), 2044

## wait\_for\_next\_used

[L4virtio::Driver::Device](#), 2075

## wait\_rx

[L4virtio::Driver::Virtio\\_net\\_device](#), 2085

## Wd\_16bit

[L4Re::Mmio\\_space](#), 1708

## Wd\_32bit

[L4Re::Mmio\\_space](#), 1708

## Wd\_64bit

[L4Re::Mmio\\_space](#), 1708

## Wd\_8bit

[L4Re::Mmio\\_space](#), 1708

## Word\_bytes

[L4::lpc::Msg](#), 768

## word\_index

[cxx::Bitmap\\_base](#), 892  
[L4Re::Util::Bitmap\\_base](#), 1781

## write

[L4::Uart](#), 1521  
[L4::Uart\\_apb](#), 1526  
[L4::Vcon](#), 1538  
[L4drivers::Register\\_tmpl< BITS, BLOCK >](#), 1603

## write\_bfm\_t

[L4virtio::Virtqueue::Desc::Flags](#), 2298

## write\_now

[cxx](#), 746

## writev

[L4Re::Vfs::Regular\\_file](#), 1903

## X

[L4Re::Dataspace::F](#), 1632  
[L4Re::Rm::F](#), 1758  
[Rm::F](#), 2339

## x86 Virtual Registers (UTCB), 282

[L4\\_UTCB\\_BUF\\_REGS\\_OFFSET](#), 283  
[L4\\_utcb\\_consts\\_x86](#), 283  
[L4\\_UTCB\\_EXCEPTION\\_REGS\\_SIZE](#), 283  
[L4\\_UTCB\\_GENERIC\\_BUFFERS\\_SIZE](#), 283  
[L4\\_UTCB\\_GENERIC\\_DATA\\_SIZE](#), 283  
[L4\\_UTCB\\_INHERIT\\_FPU](#), 283  
[L4\\_UTCB\\_MSG\\_REGS\\_OFFSET](#), 283  
[L4\\_UTCB\\_OFFSET](#), 283  
[L4\\_UTCB\\_THREAD\\_REGS\\_OFFSET](#), 283

[x86/l4/sys/\\_\\_kip-arch.h](#), 2485

[x86/l4/sys/\\_\\_vcpu-arch.h](#), 2495, 2496

[x86/l4/sys/cache.h](#), 3081, 3082

[x86/l4/sys/consts.h](#), 3098, 3099

[x86/l4/sys/ipc-invoke.h](#), 3432

[x86/l4/sys/ktrace\\_events.h](#), 2506

[x86/l4/sys/l4int.h](#), 3239

x86/l4/sys/linkage.h, [2511](#), [2512](#)  
x86/l4/sys/segment.h, [2473](#), [2474](#)  
x86/l4/sys/utcb.h, [3304](#), [3306](#)  
x86/l4/sys/vm.h, [2516](#)  
x86/l4/util/bitops\_arch.h, [2521](#)  
x86/l4/util/cpu.h, [2527](#), [2528](#)  
x86/l4/util/irq.h, [2675](#), [2676](#)  
x86/l4/util/l4\_macros.h, [2531](#)  
x86/l4/util/mbi\_argv.h, [2534](#), [2535](#)  
x86/l4/util/perform.h, [2447](#), [2448](#)  
x86/l4/util/port\_io.h, [2478](#), [2480](#)  
x86/l4/util/rdtsc.h, [2457](#), [2458](#)  
x86/l4/util/spin.h, [2462](#), [2463](#)  
x86/l4f/l4/sys/ipc-l42-gcc3-nopic.h, [3433](#)  
x86/l4f/l4/sys/ipc.h, [3198](#), [3199](#)  
x86/l4f/l4/sys/segment.h, [2475](#), [2476](#)  
x86/l4f/l4/util/port\_io.h, [2482](#), [2484](#)