

eta-os

Generated by Doxygen 1.9.6



<b>1 Namespace Index</b>	<b>1</b>
1.1 Namespace List	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Namespace Documentation</b>	<b>7</b>
4.1 ACPI Namespace Reference	7
4.1.1 Function Documentation	7
4.1.1.1 __attribute__()	7
4.1.1.2 find_table()	7
4.2 PCI Namespace Reference	8
4.2.1 Function Documentation	8
4.2.1.1 enumerate_bus()	8
4.2.1.2 enumerate_device()	8
4.2.1.3 enumerate_function()	8
4.2.1.4 enumerate_pci()	9
4.2.1.5 get_device_name()	9
4.2.1.6 get_prog_if_name()	9
4.2.1.7 get_subclass_name()	9
4.2.1.8 get_vendor_name()	9
4.2.1.9 mass_storage_controller_subclass_name()	9
4.2.2 Variable Documentation	9
4.2.2.1 device_classes	10
4.3 PIT Namespace Reference	10
4.3.1 Function Documentation	10
4.3.1.1 get_freq()	10
4.3.1.2 set_divisor()	11
4.3.1.3 set_freq()	11
4.3.1.4 sleep()	11
4.3.1.5 sleepd()	11
4.3.1.6 tick()	11
4.3.2 Variable Documentation	11
4.3.2.1 base_freq	11
4.3.2.2 divisor	11
4.3.2.3 time_since_boot	12
4.4 QWERTZKeyboard Namespace Reference	12
4.4.1 Function Documentation	12
4.4.1.1 translate()	12
4.4.2 Variable Documentation	12

4.4.2.1 <code>ascii_table</code>	12
<b>5 Class Documentation</b>	<b>13</b>
5.1 BasicRenderer Class Reference	13
5.1.1 Constructor & Destructor Documentation	13
5.1.1.1 <code>BasicRenderer()</code>	13
5.1.2 Member Function Documentation	14
5.1.2.1 <code>clear()</code>	14
5.1.2.2 <code>clear_char()</code>	14
5.1.2.3 <code>draw_char()</code> [1/2]	14
5.1.2.4 <code>draw_char()</code> [2/2]	14
5.1.2.5 <code>nextln()</code>	14
5.1.2.6 <code>print()</code>	14
5.1.3 Member Data Documentation	14
5.1.3.1 <code>clear_color</code>	15
5.1.3.2 <code>color</code>	15
5.1.3.3 <code>cursor_position</code>	15
5.1.3.4 <code>psf1_font</code>	15
5.1.3.5 <code>target_framebuffer</code>	15
5.2 Bitmap Class Reference	15
5.2.1 Member Function Documentation	16
5.2.1.1 <code>operator[]()</code>	16
5.2.1.2 <code>set()</code>	16
5.2.2 Member Data Documentation	16
5.2.2.1 <code>buffer</code>	16
5.2.2.2 <code>size</code>	16
5.3 BOOT_INFO Struct Reference	16
5.3.1 Member Data Documentation	17
5.3.1.1 <code>framebuffer</code>	17
5.3.1.2 <code>mem_map</code>	17
5.3.1.3 <code>mem_map_descriptor_size</code>	17
5.3.1.4 <code>mem_map_size</code>	17
5.3.1.5 <code>psf1_font</code>	17
5.3.1.6 <code>rsdp</code>	18
5.4 ACPI::DeviceConfig Struct Reference	18
5.4.1 Member Data Documentation	18
5.4.1.1 <code>base_addr</code>	18
5.4.1.2 <code>end_bus</code>	18
5.4.1.3 <code>pci_seg_group</code>	18
5.4.1.4 <code>reserved</code>	19
5.4.1.5 <code>start_bus</code>	19
5.5 EFI_MEMORY_DESCRIPTOR Struct Reference	19

5.5.1 Member Data Documentation . . . . .	19
5.5.1.1 attrib . . . . .	19
5.5.1.2 num_pages . . . . .	19
5.5.1.3 phys_addr . . . . .	20
5.5.1.4 type . . . . .	20
5.5.1.5 virt_addr . . . . .	20
5.6 Framebuffer Struct Reference . . . . .	20
5.6.1 Member Data Documentation . . . . .	20
5.6.1.1 base_adress . . . . .	20
5.6.1.2 buffer_size . . . . .	21
5.6.1.3 height . . . . .	21
5.6.1.4 pixels_per_scanline . . . . .	21
5.6.1.5 width . . . . .	21
5.7 GDT Struct Reference . . . . .	21
5.7.1 Member Data Documentation . . . . .	21
5.7.1.1 kernel_code . . . . .	22
5.7.1.2 kernel_data . . . . .	22
5.7.1.3 null . . . . .	22
5.7.1.4 user_code . . . . .	22
5.7.1.5 user_data . . . . .	22
5.7.1.6 user_null . . . . .	22
5.8 GDTDescriptor Struct Reference . . . . .	22
5.8.1 Member Data Documentation . . . . .	23
5.8.1.1 offset . . . . .	23
5.8.1.2 size . . . . .	23
5.9 GDTEnter Struct Reference . . . . .	23
5.9.1 Member Data Documentation . . . . .	23
5.9.1.1 access_byte . . . . .	23
5.9.1.2 base0 . . . . .	24
5.9.1.3 base1 . . . . .	24
5.9.1.4 base2 . . . . .	24
5.9.1.5 limit0 . . . . .	24
5.9.1.6 limit1_flags . . . . .	24
5.10 HeapSegHeader Struct Reference . . . . .	24
5.10.1 Member Function Documentation . . . . .	25
5.10.1.1 combine_backward() . . . . .	25
5.10.1.2 combine_forward() . . . . .	25
5.10.1.3 split() . . . . .	25
5.10.2 Member Data Documentation . . . . .	25
5.10.2.1 free . . . . .	25
5.10.2.2 last . . . . .	25
5.10.2.3 length . . . . .	26

5.10.2.4 next	26
5.11 IDTDescEntry Struct Reference	26
5.11.1 Member Function Documentation	26
5.11.1.1 get_offset()	26
5.11.1.2 set_offset()	27
5.11.2 Member Data Documentation	27
5.11.2.1 ignore	27
5.11.2.2 ist	27
5.11.2.3 offset0	27
5.11.2.4 offset1	27
5.11.2.5 offset2	27
5.11.2.6 selector	27
5.11.2.7 type_attr	28
5.12 IDTR Struct Reference	28
5.12.1 Member Data Documentation	28
5.12.1.1 limit	28
5.12.1.2 offset	28
5.13 KernelInfo Struct Reference	28
5.13.1 Member Data Documentation	29
5.13.1.1 page_table_mgr	29
5.14 ACPI::MCFGHeader Struct Reference	29
5.14.1 Member Data Documentation	29
5.14.1.1 header	29
5.14.1.2 reserved	29
5.15 PageDirEntry Struct Reference	29
5.15.1 Member Function Documentation	30
5.15.1.1 get_address()	30
5.15.1.2 get_flag()	30
5.15.1.3 set_address()	30
5.15.1.4 set_flag()	30
5.15.2 Member Data Documentation	30
5.15.2.1 value	31
5.16 PageFrameAllocator Class Reference	31
5.16.1 Member Function Documentation	31
5.16.1.1 free_page()	31
5.16.1.2 free_pages()	31
5.16.1.3 get_free_mem()	32
5.16.1.4 get_reserved_mem()	32
5.16.1.5 get_used_mem()	32
5.16.1.6 lock_page()	32
5.16.1.7 lock_pages()	32
5.16.1.8 read_efi_memory_map()	32

5.16.1.9 request_page()	32
5.16.2 Member Data Documentation	33
5.16.2.1 page_bitmap	33
5.17 PageMapIndexer Class Reference	33
5.17.1 Constructor & Destructor Documentation	33
5.17.1.1 PageMapIndexer()	33
5.17.2 Member Data Documentation	33
5.17.2.1 p_i	34
5.17.2.2 pd_i	34
5.17.2.3 pdp_i	34
5.17.2.4 pt_i	34
5.18 PageTable Struct Reference	34
5.18.1 Member Data Documentation	34
5.18.1.1 entries	34
5.19 PageTableManager Class Reference	35
5.19.1 Constructor & Destructor Documentation	35
5.19.1.1 PageTableManager()	35
5.19.2 Member Function Documentation	35
5.19.2.1 map_mem()	35
5.19.3 Member Data Documentation	35
5.19.3.1 pml4_addr	35
5.20 PCI::PCIDeviceHeader Struct Reference	36
5.20.1 Member Data Documentation	36
5.20.1.1 bist	36
5.20.1.2 cache_line_size	36
5.20.1.3 class_code	36
5.20.1.4 command	36
5.20.1.5 device_id	37
5.20.1.6 header_type	37
5.20.1.7 latency_timer	37
5.20.1.8 prog_if	37
5.20.1.9 revision_id	37
5.20.1.10 status	37
5.20.1.11 subclass	37
5.20.1.12 vendor_id	38
5.21 Point Struct Reference	38
5.21.1 Member Data Documentation	38
5.21.1.1 x	38
5.21.1.2 y	38
5.22 PSF1_FONT Struct Reference	38
5.22.1 Member Data Documentation	39
5.22.1.1 glyph_buffer	39

5.22.1.2 psf1_header . . . . .	39
5.23 PSF1_HEADER Struct Reference . . . . .	39
5.23.1 Member Data Documentation . . . . .	39
5.23.1.1 charsize . . . . .	39
5.23.1.2 magic . . . . .	39
5.23.1.3 mode . . . . .	40
5.24 ACPI::RSDP2 Struct Reference . . . . .	40
5.24.1 Member Data Documentation . . . . .	40
5.24.1.1 checksum . . . . .	40
5.24.1.2 extended_checksum . . . . .	40
5.24.1.3 length . . . . .	40
5.24.1.4 oem_id . . . . .	41
5.24.1.5 reserved . . . . .	41
5.24.1.6 revision . . . . .	41
5.24.1.7 rsdt_addr . . . . .	41
5.24.1.8 signature . . . . .	41
5.24.1.9 xsdt_adrr . . . . .	41
5.25 ACPI::SDTHeader Struct Reference . . . . .	41
5.25.1 Member Data Documentation . . . . .	42
5.25.1.1 checksum . . . . .	42
5.25.1.2 creator_id . . . . .	42
5.25.1.3 creator_revision . . . . .	42
5.25.1.4 length . . . . .	42
5.25.1.5 oem_id . . . . .	42
5.25.1.6 oem_revision . . . . .	43
5.25.1.7 oem_table_id . . . . .	43
5.25.1.8 revision . . . . .	43
5.25.1.9 signature . . . . .	43
<b>6 File Documentation . . . . .</b>	<b>45</b>
6.1 acpi.cpp File Reference . . . . .	45
6.2 acpi.h File Reference . . . . .	45
6.2.1 Variable Documentation . . . . .	46
6.2.1.1 base_addr . . . . .	46
6.2.1.2 checksum . . . . .	46
6.2.1.3 creator_id . . . . .	46
6.2.1.4 creator_revision . . . . .	47
6.2.1.5 end_bus . . . . .	47
6.2.1.6 extended_checksum . . . . .	47
6.2.1.7 header . . . . .	47
6.2.1.8 length . . . . .	47
6.2.1.9 oem_id . . . . .	47



6.2.1.10 oem_revision . . . . .	47
6.2.1.11 oem_table_id . . . . .	47
6.2.1.12 pci_seg_group . . . . .	48
6.2.1.13 reserved . . . . .	48
6.2.1.14 revision . . . . .	48
6.2.1.15 rsdt_addr . . . . .	48
6.2.1.16 signature . . . . .	48
6.2.1.17 start_bus . . . . .	48
6.2.1.18 xsdt_adrr . . . . .	48
6.3 acpi.h . . . . .	49
6.4 basic_renderer.cpp File Reference . . . . .	49
6.4.1 Variable Documentation . . . . .	49
6.4.1.1 global_renderer . . . . .	50
6.5 basic_renderer.h File Reference . . . . .	50
6.5.1 Variable Documentation . . . . .	50
6.5.1.1 global_renderer . . . . .	50
6.6 basic_renderer.h . . . . .	50
6.7 bitmap.cpp File Reference . . . . .	51
6.8 bitmap.h File Reference . . . . .	51
6.9 bitmap.h . . . . .	51
6.10 c_str.cpp File Reference . . . . .	51
6.10.1 Function Documentation . . . . .	52
6.10.1.1 to_hstring() [1/4] . . . . .	52
6.10.1.2 to_hstring() [2/4] . . . . .	52
6.10.1.3 to_hstring() [3/4] . . . . .	52
6.10.1.4 to_hstring() [4/4] . . . . .	52
6.10.1.5 to_string() [1/4] . . . . .	52
6.10.1.6 to_string() [2/4] . . . . .	53
6.10.1.7 to_string() [3/4] . . . . .	53
6.10.1.8 to_string() [4/4] . . . . .	53
6.10.2 Variable Documentation . . . . .	53
6.10.2.1 double_to_string_output . . . . .	53
6.10.2.2 hex_to_string_output_16 . . . . .	53
6.10.2.3 hex_to_string_output_32 . . . . .	53
6.10.2.4 hex_to_string_output_64 . . . . .	53
6.10.2.5 hex_to_string_output_8 . . . . .	54
6.10.2.6 int_to_string_output . . . . .	54
6.10.2.7 uint_to_string_output . . . . .	54
6.11 c_str.h File Reference . . . . .	54
6.11.1 Function Documentation . . . . .	54
6.11.1.1 to_hstring() [1/4] . . . . .	54
6.11.1.2 to_hstring() [2/4] . . . . .	55

6.11.1.3 to_hstring() [3/4]	55
6.11.1.4 to_hstring() [4/4]	55
6.11.1.5 to_string() [1/4]	55
6.11.1.6 to_string() [2/4]	55
6.11.1.7 to_string() [3/4]	55
6.11.1.8 to_string() [4/4]	55
6.12 c_str.h	56
6.13 efi_memory.cpp File Reference	56
6.13.1 Variable Documentation	56
6.13.1.1 EFI_MEMORY_TYPE_STRINGS	56
6.14 efi_memory.h File Reference	56
6.14.1 Variable Documentation	57
6.14.1.1 EFI_MEMORY_TYPE_STRINGS	57
6.15 efi_memory.h	57
6.16 framebuffer.h File Reference	57
6.17 framebuffer.h	57
6.18 gdt.cpp File Reference	58
6.18.1 Function Documentation	58
6.18.1.1 __attribute__()	58
6.19 gdt.h File Reference	58
6.19.1 Function Documentation	59
6.19.1.1 __attribute__()	59
6.19.1.2 load_gdt()	59
6.19.2 Variable Documentation	59
6.19.2.1 access_byte	59
6.19.2.2 base0	59
6.19.2.3 base1	59
6.19.2.4 base2	59
6.19.2.5 default_gdt	60
6.19.2.6 kernel_code	60
6.19.2.7 kernel_data	60
6.19.2.8 limit0	60
6.19.2.9 limit1_flags	60
6.19.2.10 null	60
6.19.2.11 offset	60
6.19.2.12 size	60
6.19.2.13 user_code	61
6.19.2.14 user_data	61
6.19.2.15 user_null	61
6.20 gdt.h	61
6.21 kb_scancode_trans.cpp File Reference	61
6.22 kb_scancode_trans.h File Reference	62

6.22.1 Macro Definition Documentation	62
6.22.1.1 BACKSPACE	62
6.22.1.2 ENTER	63
6.22.1.3 LEFT_SHIFT	63
6.22.1.4 RIGHT_SHIFT	63
6.22.1.5 SPACE	63
6.23 kb_scancode_trans.h	63
6.24 keyboard.cpp File Reference	63
6.24.1 Function Documentation	64
6.24.1.1 handle_keyboard()	64
6.24.2 Variable Documentation	64
6.24.2.1 left_shift_pressed	64
6.24.2.2 right_shift_pressed	64
6.25 keyboard.h File Reference	64
6.25.1 Function Documentation	64
6.25.1.1 handle_keyboard()	65
6.26 keyboard.h	65
6.27 idt.cpp File Reference	65
6.28 idt.h File Reference	65
6.28.1 Macro Definition Documentation	66
6.28.1.1 IDT_TA_CALLGATE	66
6.28.1.2 IDT_TA_INTERRUPTGATE	66
6.28.1.3 IDT_TA_TRAPGATE	66
6.28.2 Function Documentation	66
6.28.2.1 __attribute__()	66
6.28.3 Variable Documentation	66
6.28.3.1 limit	66
6.28.3.2 offset	66
6.29 idt.h	67
6.30 interrupts.cpp File Reference	67
6.30.1 Function Documentation	67
6.30.1.1 __attribute__()	67
6.30.1.2 pic_end_master()	67
6.30.1.3 pic_end_slave()	68
6.30.1.4 remap_pic()	68
6.31 interrupts.h File Reference	68
6.31.1 Macro Definition Documentation	68
6.31.1.1 ICW1_ICW4	68
6.31.1.2 ICW1_INIT	69
6.31.1.3 ICW4_8086	69
6.31.1.4 PIC1_CMD	69
6.31.1.5 PIC1_DATA	69

6.31.1.6 PIC2_CMD . . . . .	69
6.31.1.7 PIC2_DATA . . . . .	69
6.31.1.8 PIC_EOI . . . . .	69
6.31.2 Function Documentation . . . . .	69
6.31.2.1 __attribute__() . . . . .	70
6.31.2.2 pic_end_master() . . . . .	70
6.31.2.3 pic_end_slave() . . . . .	70
6.31.2.4 remap_pic() . . . . .	70
6.32 interrupts.h . . . . .	70
6.33 io.cpp File Reference . . . . .	70
6.33.1 Function Documentation . . . . .	71
6.33.1.1 inb() . . . . .	71
6.33.1.2 io_wait() . . . . .	71
6.33.1.3 outb() . . . . .	71
6.34 io.h File Reference . . . . .	71
6.34.1 Function Documentation . . . . .	71
6.34.1.1 inb() . . . . .	72
6.34.1.2 io_wait() . . . . .	72
6.34.1.3 outb() . . . . .	72
6.35 io.h . . . . .	72
6.36 kernel.cpp File Reference . . . . .	72
6.36.1 Function Documentation . . . . .	72
6.36.1.1 start() . . . . .	73
6.37 kernel_util.cpp File Reference . . . . .	73
6.37.1 Function Documentation . . . . .	73
6.37.1.1 init_kernel() . . . . .	73
6.37.1.2 prepare_acpi() . . . . .	73
6.37.1.3 prepare_interrupts() . . . . .	73
6.37.1.4 prepare_mem() . . . . .	74
6.37.1.5 set_idt_gate() . . . . .	74
6.37.2 Variable Documentation . . . . .	74
6.37.2.1 idtr . . . . .	74
6.37.2.2 kernel_info . . . . .	74
6.37.2.3 r . . . . .	74
6.38 kernel_util.h File Reference . . . . .	74
6.38.1 Function Documentation . . . . .	75
6.38.1.1 init_kernel() . . . . .	75
6.38.2 Variable Documentation . . . . .	75
6.38.2.1 _kernel_end . . . . .	75
6.38.2.2 _kernel_start . . . . .	75
6.39 kernel_util.h . . . . .	76
6.40 math_util.h File Reference . . . . .	76

6.41 math_util.h . . . . .	76
6.42 memory.cpp File Reference . . . . .	76
6.42.1 Function Documentation . . . . .	77
6.42.1.1 get_memory_size() . . . . .	77
6.42.1.2 memset() . . . . .	77
6.43 memory.h File Reference . . . . .	77
6.43.1 Function Documentation . . . . .	77
6.43.1.1 get_memory_size() . . . . .	78
6.43.1.2 memset() . . . . .	78
6.44 memory.h . . . . .	78
6.45 heap.cpp File Reference . . . . .	78
6.45.1 Function Documentation . . . . .	78
6.45.1.1 extend_heap() . . . . .	79
6.45.1.2 free() . . . . .	79
6.45.1.3 init_heap() . . . . .	79
6.45.1.4 malloc() . . . . .	79
6.45.2 Variable Documentation . . . . .	79
6.45.2.1 heap_end . . . . .	79
6.45.2.2 heap_start . . . . .	79
6.45.2.3 last_header . . . . .	79
6.46 heap.h File Reference . . . . .	80
6.46.1 Function Documentation . . . . .	80
6.46.1.1 extend_heap() . . . . .	80
6.46.1.2 free() . . . . .	80
6.46.1.3 init_heap() . . . . .	80
6.46.1.4 malloc() . . . . .	81
6.47 heap.h . . . . .	81
6.48 page_frame_allocator.cpp File Reference . . . . .	81
6.48.1 Variable Documentation . . . . .	81
6.48.1.1 free_memory . . . . .	81
6.48.1.2 global_allocator . . . . .	82
6.48.1.3 initialized . . . . .	82
6.48.1.4 page_bitmap_index . . . . .	82
6.48.1.5 reserved_memory . . . . .	82
6.48.1.6 used_memory . . . . .	82
6.49 page_frame_allocator.h File Reference . . . . .	82
6.49.1 Variable Documentation . . . . .	83
6.49.1.1 global_allocator . . . . .	83
6.50 page_frame_allocator.h . . . . .	83
6.51 page_map_indexer.cpp File Reference . . . . .	83
6.52 page_map_indexer.h File Reference . . . . .	83
6.53 page_map_indexer.h . . . . .	84

6.54 page_table_manager.cpp File Reference . . . . .	84
6.54.1 Variable Documentation . . . . .	84
6.54.1.1 global_ptm . . . . .	84
6.55 page_table_manager.h File Reference . . . . .	84
6.55.1 Variable Documentation . . . . .	85
6.55.1.1 global_ptm . . . . .	85
6.56 page_table_manager.h . . . . .	85
6.57 paging.cpp File Reference . . . . .	85
6.58 paging.h File Reference . . . . .	85
6.58.1 Enumeration Type Documentation . . . . .	86
6.58.1.1 PT_FLAG . . . . .	86
6.58.2 Function Documentation . . . . .	86
6.58.2.1 __attribute__ () . . . . .	86
6.58.3 Variable Documentation . . . . .	86
6.58.3.1 entries . . . . .	87
6.59 paging.h . . . . .	87
6.60 panic_screen.cpp File Reference . . . . .	87
6.60.1 Function Documentation . . . . .	87
6.60.1.1 panic() . . . . .	87
6.61 panic_screen.h File Reference . . . . .	88
6.61.1 Function Documentation . . . . .	88
6.61.1.1 panic() . . . . .	88
6.62 panic_screen.h . . . . .	88
6.63 pci.cpp File Reference . . . . .	88
6.64 pci.h File Reference . . . . .	88
6.65 pci.h . . . . .	89
6.66 pci_descriptors.cpp File Reference . . . . .	90
6.67 pit.cpp File Reference . . . . .	90
6.68 pit.h File Reference . . . . .	90
6.69 pit.h . . . . .	91
6.70 simple_font.h File Reference . . . . .	91
6.71 simple_font.h . . . . .	91

# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

ACPI	7
PCI	8
PIT	10
QWERTZKeyboard	12





## Chapter 2

# Class Index

### 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

BasicRenderer	13
Bitmap	15
BOOT_INFO	16
ACPI::DeviceConfig	18
EFI_MEMORY_DESCRIPTOR	19
Framebuffer	20
GDT	21
GDTDescriptor	22
GDTEntry	23
HeapSegHeader	24
IDTDescEntry	26
IDTR	28
KernelInfo	28
ACPI::MCFGHeader	29
PageDirEntry	29
PageFrameAllocator	31
PageMapIndexer	33
PageTable	34
PageTableManager	35
PCI::PCIDeviceHeader	36
Point	38
PSF1_FONT	38
PSF1_HEADER	39
ACPI::RSDP2	40
ACPI::SDTHeader	41



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

acpi.cpp	45
acpi.h	45
basic_renderer.cpp	49
basic_renderer.h	50
bitmap.cpp	51
bitmap.h	51
c_str.cpp	51
c_str.h	54
efi_memory.cpp	56
efi_memory.h	56
framebuffer.h	57
gdt.cpp	58
gdt.h	58
kb_scancode_trans.cpp	61
kb_scancode_trans.h	62
keyboard.cpp	63
keyboard.h	64
idt.cpp	65
idt.h	65
interrupts.cpp	67
interrupts.h	68
io.cpp	70
io.h	71
kernel.cpp	72
kernel_util.cpp	73
kernel_util.h	74
math_util.h	76
memory.cpp	76
memory.h	77
heap.cpp	78
heap.h	80
page_frame_allocator.cpp	81
page_frame_allocator.h	82
page_map_indexer.cpp	83
page_map_indexer.h	83

<a href="#">page_table_manager.cpp</a>	84
<a href="#">page_table_manager.h</a>	84
<a href="#">paging.cpp</a>	85
<a href="#">paging.h</a>	85
<a href="#">panic_screen.cpp</a>	87
<a href="#">panic_screen.h</a>	88
<a href="#">pci.cpp</a>	88
<a href="#">pci.h</a>	88
<a href="#">pci_descriptors.cpp</a>	90
<a href="#">pit.cpp</a>	90
<a href="#">pit.h</a>	90
<a href="#">simple_font.h</a>	91

## Chapter 4

# Namespace Documentation

### 4.1 ACPI Namespace Reference

#### Classes

- struct [DeviceConfig](#)
- struct [MCFGHeader](#)
- struct [RSDP2](#)
- struct [SDTHeader](#)

#### Functions

- void \* [find\\_table](#) ([SDTHeader](#) \*sdt\_header, char \*[signature](#))
- struct [ACPI::RSDP2 \\_\\_attribute\\_\\_\(\(packed\)\)](#)

#### 4.1.1 Function Documentation

##### 4.1.1.1 `__attribute__()`

```
struct ACPI::DeviceConfig ACPI::__attribute__ (  
    (packed) )
```

##### 4.1.1.2 `find_table()`

```
void * ACPI::find_table (  
    SDTHeader * sdt_header,  
    char * signature )
```

## 4.2 PCI Namespace Reference

### Classes

- struct [PCIDeviceHeader](#)

### Functions

- void [enumerate\\_function](#) (uint64\_t device\_addr, uint64\_t function)
- void [enumerate\\_device](#) (uint64\_t bus\_addr, uint64\_t device)
- void [enumerate\\_bus](#) (uint64\_t base\_addr, uint64\_t bus)
- void [enumerate\\_pci](#) ([ACPI::MCFGHeader](#) \*mcfg)
- const char \* [get\\_vendor\\_name](#) (uint16\_t vendor\_id)
- const char \* [get\\_device\\_name](#) (uint16\_t vendor\_id, uint16\_t device\_id)
- const char \* [get\\_subclass\\_name](#) (uint8\_t class\_code, uint8\_t subclass\_code)
- const char \* [get\\_prog\\_if\\_name](#) (uint8\_t class\_code, uint8\_t subclass\_code, uint8\_t prog\_if)
- const char \* [mass\\_storage\\_controller\\_subclass\\_name](#) (uint8\_t subclass\_code)

### Variables

- const char \* [device\\_classes](#) []

### 4.2.1 Function Documentation

#### 4.2.1.1 enumerate\_bus()

```
void PCI::enumerate_bus (
    uint64_t base_addr,
    uint64_t bus )
```

#### 4.2.1.2 enumerate\_device()

```
void PCI::enumerate_device (
    uint64_t bus_addr,
    uint64_t device )
```

#### 4.2.1.3 enumerate\_function()

```
void PCI::enumerate_function (
    uint64_t device_addr,
    uint64_t function )
```

#### 4.2.1.4 enumerate\_pci()

```
void PCI::enumerate_pci (
    ACPI::MCFGHeader * mcfg )
```

#### 4.2.1.5 get\_device\_name()

```
const char * PCI::get_device_name (
    uint16_t vendor_id,
    uint16_t device_id )
```

#### 4.2.1.6 get\_prog\_if\_name()

```
const char * PCI::get_prog_if_name (
    uint8_t class_code,
    uint8_t subclass_code,
    uint8_t prog_if )
```

#### 4.2.1.7 get\_subclass\_name()

```
const char * PCI::get_subclass_name (
    uint8_t class_code,
    uint8_t subclass_code )
```

#### 4.2.1.8 get\_vendor\_name()

```
const char * PCI::get_vendor_name (
    uint16_t vendor_id )
```

#### 4.2.1.9 mass\_storage\_controller\_subclass\_name()

```
const char * PCI::mass_storage_controller_subclass_name (
    uint8_t subclass_code )
```

### 4.2.2 Variable Documentation

#### 4.2.2.1 device\_classes

```
const char * PCI::device_classes
```

##### Initial value:

```
{  
    "Unclassified",  
    "Mass Storage Controller",  
    "Network Controller",  
    "Display Controller",  
    "Multimedia Controller",  
    "Memory Controller",  
    "Bridge Device",  
    "Simple Communication Controller",  
    "Base System Peripheral",  
    "Input Device Controller",  
    "Docking Station",  
    "Processor",  
    "Serial Bus Controller",  
    "Wireless Controller",  
    "Intelligent Controller",  
    "Satellite Communication Controller",  
    "Encryption Controller",  
    "Signal Processing Controller",  
    "Processing Accelerator",  
    "Non Essential Instrumentation"  
}
```

## 4.3 PIT Namespace Reference

### Functions

- void [sleepd](#) (double seconds)
- void [sleep](#) (uint64\_t ms)
- void [set\\_divisor](#) (uint16\_t divisor)
- uint64\_t [get\\_freq](#) ()
- void [set\\_freq](#) (uint64\_t freq)
- void [tick](#) ()

### Variables

- double [time\\_since\\_boot](#) = 0
- uint16\_t [divisor](#) = 65535
- const uint64\_t [base\\_freq](#) = 1193182

### 4.3.1 Function Documentation

#### 4.3.1.1 get\_freq()

```
uint64_t PIT::get_freq ( )
```



#### 4.3.1.2 set\_divisor()

```
void PIT::set_divisor (
    uint16_t divisor )
```

#### 4.3.1.3 set\_freq()

```
void PIT::set_freq (
    uint64_t freq )
```

#### 4.3.1.4 sleep()

```
void PIT::sleep (
    uint64_t ms )
```

#### 4.3.1.5 sleepd()

```
void PIT::sleepd (
    double seconds )
```

#### 4.3.1.6 tick()

```
void PIT::tick ( )
```

### 4.3.2 Variable Documentation

#### 4.3.2.1 base\_freq

```
const uint64_t PIT::base_freq = 1193182
```

#### 4.3.2.2 divisor

```
uint16_t PIT::divisor = 65535
```

#### 4.3.2.3 time\_since\_boot

```
double PIT::time_since_boot = 0
```

## 4.4 QWERTZKeyboard Namespace Reference

### Functions

- char [translate](#) (uint8\_t scancode, bool uppercase)

### Variables

- const char [ascii\\_table](#) []

### 4.4.1 Function Documentation

#### 4.4.1.1 translate()

```
char QWERTZKeyboard::translate (
    uint8_t scancode,
    bool uppercase )
```

### 4.4.2 Variable Documentation

#### 4.4.2.1 ascii\_table

```
const char QWERTZKeyboard::ascii_table
```

#### Initial value:

```
=
{
    0, 0, '1', '2', '3', '4', '5', '6', '7', '8',
    '9', '0', '-', '=', 0, 0, 'q', 'w', 'e',
    'r', 't', 'z', 'u', 'i', 'o', 'p', '[', ']',
    0, 0, 'a', 's', 'd', 'f', 'g', 'h', 'j', 'k',
    'l', ';', '\'', 'y', 0, '\'', 'x', 'c',
    'v', 'b', 'n', 'm', ',', '.', '/', 0, '*', 0, ' '
}
```

## Chapter 5

# Class Documentation

### 5.1 BasicRenderer Class Reference

```
#include <basic_renderer.h>
```

#### Public Member Functions

- [BasicRenderer](#) ([Framebuffer](#) \*[target\\_framebuffer](#), [PSF1\\_FONT](#) \*[psf1\\_font](#), unsigned int [color](#))
- void [print](#) (const char \*str)
- void [draw\\_char](#) (char chr, unsigned int xOff, unsigned yOff)
- void [draw\\_char](#) (char chr)
- void [clear](#) ()
- void [clear\\_char](#) ()
- void [nextln](#) ()

#### Public Attributes

- [Point](#) [cursor\\_position](#)
- [Framebuffer](#) \* [target\\_framebuffer](#)
- [PSF1\\_FONT](#) \* [psf1\\_font](#)
- unsigned int [color](#)
- unsigned int [clear\\_color](#)

#### 5.1.1 Constructor & Destructor Documentation

##### 5.1.1.1 BasicRenderer()

```
BasicRenderer::BasicRenderer (  
    Framebuffer * target\_framebuffer,  
    PSF1\_FONT * psf1\_font,  
    unsigned int color ) [inline]
```

## 5.1.2 Member Function Documentation

### 5.1.2.1 clear()

```
void BasicRenderer::clear ( )
```

### 5.1.2.2 clear\_char()

```
void BasicRenderer::clear_char ( )
```

### 5.1.2.3 draw\_char() [1/2]

```
void BasicRenderer::draw_char (
    char chr )
```

### 5.1.2.4 draw\_char() [2/2]

```
void BasicRenderer::draw_char (
    char chr,
    unsigned int xOff,
    unsigned yOff )
```

### 5.1.2.5 nextln()

```
void BasicRenderer::nextln ( )
```

### 5.1.2.6 print()

```
void BasicRenderer::print (
    const char * str )
```

## 5.1.3 Member Data Documentation

### 5.1.3.1 clear\_color

```
unsigned int BasicRenderer::clear_color
```

### 5.1.3.2 color

```
unsigned int BasicRenderer::color
```

### 5.1.3.3 cursor\_position

```
Point BasicRenderer::cursor_position
```

### 5.1.3.4 psf1\_font

```
PSF1_FONT* BasicRenderer::psf1_font
```

### 5.1.3.5 target\_framebuffer

```
Framebuffer* BasicRenderer::target_framebuffer
```

The documentation for this class was generated from the following files:

- [basic\\_renderer.h](#)
- [basic\\_renderer.cpp](#)

## 5.2 Bitmap Class Reference

```
#include <bitmap.h>
```

### Public Member Functions

- bool [operator\[\]](#) (uint64\_t index)
- bool [set](#) (uint64\_t index, bool value)

### Public Attributes

- size\_t [size](#)
- uint8\_t \* [buffer](#)

## 5.2.1 Member Function Documentation

### 5.2.1.1 operator[]()

```
bool Bitmap::operator[] (
    uint64_t index )
```

### 5.2.1.2 set()

```
bool Bitmap::set (
    uint64_t index,
    bool value )
```

## 5.2.2 Member Data Documentation

### 5.2.2.1 buffer

```
uint8_t* Bitmap::buffer
```

### 5.2.2.2 size

```
size_t Bitmap::size
```

The documentation for this class was generated from the following files:

- [bitmap.h](#)
- [bitmap.cpp](#)

## 5.3 BOOT\_INFO Struct Reference

```
#include <kernel_util.h>
```

## Public Attributes

- [Framebuffer](#) \* [framebuffer](#)
- [PSF1\\_FONT](#) \* [psf1\\_font](#)
- void \* [mem\\_map](#)
- [uint64\\_t](#) [mem\\_map\\_size](#)
- [uint64\\_t](#) [mem\\_map\\_descriptor\\_size](#)
- [ACPI::RSDP2](#) \* [rsdp](#)

## 5.3.1 Member Data Documentation

### 5.3.1.1 framebuffer

[Framebuffer](#)\* [BOOT\\_INFO::framebuffer](#)

### 5.3.1.2 mem\_map

void\* [BOOT\\_INFO::mem\\_map](#)

### 5.3.1.3 mem\_map\_descriptor\_size

[uint64\\_t](#) [BOOT\\_INFO::mem\\_map\\_descriptor\\_size](#)

### 5.3.1.4 mem\_map\_size

[uint64\\_t](#) [BOOT\\_INFO::mem\\_map\\_size](#)

### 5.3.1.5 psf1\_font

[PSF1\\_FONT](#)\* [BOOT\\_INFO::psf1\\_font](#)

### 5.3.1.6 rsdp

`ACPI::RSDP2* BOOT_INFO::rsdp`

The documentation for this struct was generated from the following file:

- [kernel\\_util.h](#)

## 5.4 ACPI::DeviceConfig Struct Reference

```
#include <acpi.h>
```

### Public Attributes

- `uint64_t` [base\\_addr](#)
- `uint16_t` [pci\\_seg\\_group](#)
- `uint8_t` [start\\_bus](#)
- `uint8_t` [end\\_bus](#)
- `uint32_t` [reserved](#)

### 5.4.1 Member Data Documentation

#### 5.4.1.1 base\_addr

`uint64_t` `ACPI::DeviceConfig::base_addr`

#### 5.4.1.2 end\_bus

`uint8_t` `ACPI::DeviceConfig::end_bus`

#### 5.4.1.3 pci\_seg\_group

`uint16_t` `ACPI::DeviceConfig::pci_seg_group`



#### 5.4.1.4 reserved

```
uint32_t ACPI::DeviceConfig::reserved
```

#### 5.4.1.5 start\_bus

```
uint8_t ACPI::DeviceConfig::start_bus
```

The documentation for this struct was generated from the following file:

- [acpi.h](#)

## 5.5 EFI\_MEMORY\_DESCRIPTOR Struct Reference

```
#include <efi_memory.h>
```

### Public Attributes

- [uint32\\_t type](#)
- [void \\* phys\\_addr](#)
- [void \\* virt\\_addr](#)
- [uint64\\_t num\\_pages](#)
- [uint64\\_t attrib](#)

### 5.5.1 Member Data Documentation

#### 5.5.1.1 attrib

```
uint64_t EFI_MEMORY_DESCRIPTOR::attrib
```

#### 5.5.1.2 num\_pages

```
uint64_t EFI_MEMORY_DESCRIPTOR::num_pages
```

### 5.5.1.3 phys\_addr

```
void* EFI_MEMORY_DESCRIPTOR::phys_addr
```

### 5.5.1.4 type

```
uint32_t EFI_MEMORY_DESCRIPTOR::type
```

### 5.5.1.5 virt\_addr

```
void* EFI_MEMORY_DESCRIPTOR::virt_addr
```

The documentation for this struct was generated from the following file:

- [efi\\_memory.h](#)

## 5.6 Framebuffer Struct Reference

```
#include <framebuffer.h>
```

### Public Attributes

- void \* [base\\_address](#)
- size\_t [buffer\\_size](#)
- unsigned int [width](#)
- unsigned int [height](#)
- unsigned int [pixels\\_per\\_scanline](#)

### 5.6.1 Member Data Documentation

#### 5.6.1.1 base\_address

```
void* Framebuffer::base_address
```

#### 5.6.1.2 buffer\_size

```
size_t Framebuffer::buffer_size
```

#### 5.6.1.3 height

```
unsigned int Framebuffer::height
```

#### 5.6.1.4 pixels\_per\_scanline

```
unsigned int Framebuffer::pixels_per_scanline
```

#### 5.6.1.5 width

```
unsigned int Framebuffer::width
```

The documentation for this struct was generated from the following file:

- [framebuffer.h](#)

## 5.7 GDT Struct Reference

```
#include <gdt.h>
```

### Public Attributes

- [GDTEnter null](#)
- [GDTEnter kernel\\_code](#)
- [GDTEnter kernel\\_data](#)
- [GDTEnter user\\_null](#)
- [GDTEnter user\\_code](#)
- [GDTEnter user\\_data](#)

### 5.7.1 Member Data Documentation

#### 5.7.1.1 kernel\_code

`GDTEEntry GDT::kernel_code`

#### 5.7.1.2 kernel\_data

`GDTEEntry GDT::kernel_data`

#### 5.7.1.3 null

`GDTEEntry GDT::null`

#### 5.7.1.4 user\_code

`GDTEEntry GDT::user_code`

#### 5.7.1.5 user\_data

`GDTEEntry GDT::user_data`

#### 5.7.1.6 user\_null

`GDTEEntry GDT::user_null`

The documentation for this struct was generated from the following file:

- [gdt.h](#)

## 5.8 GDTDescriptor Struct Reference

```
#include <gdt.h>
```

## Public Attributes

- `uint16_t` [size](#)
- `uint64_t` [offset](#)

### 5.8.1 Member Data Documentation

#### 5.8.1.1 offset

```
uint64_t GDTDescriptor::offset
```

#### 5.8.1.2 size

```
uint16_t GDTDescriptor::size
```

The documentation for this struct was generated from the following file:

- [gdt.h](#)

## 5.9 GDTEntry Struct Reference

```
#include <gdt.h>
```

## Public Attributes

- `uint16_t` [limit0](#)
- `uint16_t` [base0](#)
- `uint8_t` [base1](#)
- `uint8_t` [access\\_byte](#)
- `uint8_t` [limit1\\_flags](#)
- `uint8_t` [base2](#)

### 5.9.1 Member Data Documentation

#### 5.9.1.1 access\_byte

```
uint8_t GDTEntry::access_byte
```

#### 5.9.1.2 base0

```
uint16_t GDTEnter::base0
```

#### 5.9.1.3 base1

```
uint8_t GDTEnter::base1
```

#### 5.9.1.4 base2

```
uint8_t GDTEnter::base2
```

#### 5.9.1.5 limit0

```
uint16_t GDTEnter::limit0
```

#### 5.9.1.6 limit1\_flags

```
uint8_t GDTEnter::limit1_flags
```

The documentation for this struct was generated from the following file:

- [gdt.h](#)

## 5.10 HeapSegHeader Struct Reference

```
#include <heap.h>
```

### Public Member Functions

- void [combine\\_forward](#) ()
- void [combine\\_backward](#) ()
- [HeapSegHeader](#) \* [split](#) (size\_t split\_length)

## Public Attributes

- `size_t` `length`
- `HeapSegHeader *` `next`
- `HeapSegHeader *` `last`
- `bool` `free`

## 5.10.1 Member Function Documentation

### 5.10.1.1 `combine_backward()`

```
void HeapSegHeader::combine_backward ( )
```

### 5.10.1.2 `combine_forward()`

```
void HeapSegHeader::combine_forward ( )
```

### 5.10.1.3 `split()`

```
HeapSegHeader * HeapSegHeader::split (   
    size_t split_length )
```

## 5.10.2 Member Data Documentation

### 5.10.2.1 `free`

```
bool HeapSegHeader::free
```

### 5.10.2.2 `last`

```
HeapSegHeader* HeapSegHeader::last
```

### 5.10.2.3 length

```
size_t HeapSegHeader::length
```

### 5.10.2.4 next

```
HeapSegHeader* HeapSegHeader::next
```

The documentation for this struct was generated from the following files:

- [heap.h](#)
- [heap.cpp](#)

## 5.11 IDTDescEntry Struct Reference

```
#include <idt.h>
```

### Public Member Functions

- void [set\\_offset](#) (uint64\_t [offset](#))
- uint64\_t [get\\_offset](#) ()

### Public Attributes

- uint16\_t [offset0](#)
- uint16\_t [selector](#)
- uint8\_t [ist](#)
- uint8\_t [type\\_attr](#)
- uint16\_t [offset1](#)
- uint32\_t [offset2](#)
- uint32\_t [ignore](#)

### 5.11.1 Member Function Documentation

#### 5.11.1.1 get\_offset()

```
uint64_t IDTDescEntry::get_offset ( )
```



### 5.11.1.2 set\_offset()

```
void IDTDescEntry::set_offset (
    uint64_t offset )
```

## 5.11.2 Member Data Documentation

### 5.11.2.1 ignore

```
uint32_t IDTDescEntry::ignore
```

### 5.11.2.2 ist

```
uint8_t IDTDescEntry::ist
```

### 5.11.2.3 offset0

```
uint16_t IDTDescEntry::offset0
```

### 5.11.2.4 offset1

```
uint16_t IDTDescEntry::offset1
```

### 5.11.2.5 offset2

```
uint32_t IDTDescEntry::offset2
```

### 5.11.2.6 selector

```
uint16_t IDTDescEntry::selector
```

### 5.11.2.7 type\_attr

```
uint8_t IDTDescEntry::type_attr
```

The documentation for this struct was generated from the following files:

- [idt.h](#)
- [idt.cpp](#)

## 5.12 IDTR Struct Reference

```
#include <idt.h>
```

### Public Attributes

- [uint16\\_t limit](#)
- [uint64\\_t offset](#)

### 5.12.1 Member Data Documentation

#### 5.12.1.1 limit

```
uint16_t IDTR::limit
```

#### 5.12.1.2 offset

```
uint64_t IDTR::offset
```

The documentation for this struct was generated from the following file:

- [idt.h](#)

## 5.13 KernelInfo Struct Reference

```
#include <kernel_util.h>
```

### Public Attributes

- [PageTableManager \\* page\\_table\\_mgr](#)

### 5.13.1 Member Data Documentation

#### 5.13.1.1 page\_table\_mgr

[PageTableManager\\*](#) [KernelInfo::page\\_table\\_mgr](#)

The documentation for this struct was generated from the following file:

- [kernel\\_util.h](#)

## 5.14 ACPI::MCFGHeader Struct Reference

```
#include <acpi.h>
```

### Public Attributes

- [SDTHeader](#) [header](#)
- [uint64\\_t](#) [reserved](#)

### 5.14.1 Member Data Documentation

#### 5.14.1.1 header

[SDTHeader](#) [ACPI::MCFGHeader::header](#)

#### 5.14.1.2 reserved

[uint64\\_t](#) [ACPI::MCFGHeader::reserved](#)

The documentation for this struct was generated from the following file:

- [acpi.h](#)

## 5.15 PageDirEntry Struct Reference

```
#include <paging.h>
```

## Public Member Functions

- void [set\\_flag](#) ([PT\\_FLAG](#) flag, bool enabled)
- bool [get\\_flag](#) ([PT\\_FLAG](#) flag)
- void [set\\_address](#) (uint64\_t addr)
- uint64\_t [get\\_address](#) ()

## Public Attributes

- uint64\_t [value](#)

### 5.15.1 Member Function Documentation

#### 5.15.1.1 [get\\_address\(\)](#)

```
uint64_t PageDirEntry::get_address ( )
```

#### 5.15.1.2 [get\\_flag\(\)](#)

```
bool PageDirEntry::get_flag (
    PT\_FLAG flag )
```

#### 5.15.1.3 [set\\_address\(\)](#)

```
void PageDirEntry::set_address (
    uint64_t addr )
```

#### 5.15.1.4 [set\\_flag\(\)](#)

```
void PageDirEntry::set_flag (
    PT\_FLAG flag,
    bool enabled )
```

### 5.15.2 Member Data Documentation

### 5.15.2.1 value

```
uint64_t PageDirEntry::value
```

The documentation for this struct was generated from the following files:

- [paging.h](#)
- [paging.cpp](#)

## 5.16 PageFrameAllocator Class Reference

```
#include <page_frame_allocator.h>
```

### Public Member Functions

- void [read\\_efi\\_memory\\_map](#) (EFI\_MEMORY\_DESCRIPTOR \*mem\_map, size\_t mem\_map\_size, size\_t mem\_map\_desc\_size)
- void [free\\_page](#) (void \*addr)
- void [free\\_pages](#) (void \*addr, uint64\_t page\_count)
- void [lock\\_page](#) (void \*addr)
- void [lock\\_pages](#) (void \*addr, uint64\_t page\_count)
- void \* [request\\_page](#) ()
- uint64\_t [get\\_free\\_mem](#) ()
- uint64\_t [get\\_used\\_mem](#) ()
- uint64\_t [get\\_reserved\\_mem](#) ()

### Public Attributes

- [Bitmap](#) [page\\_bitmap](#)

### 5.16.1 Member Function Documentation

#### 5.16.1.1 free\_page()

```
void PageFrameAllocator::free_page (  
    void * addr )
```

#### 5.16.1.2 free\_pages()

```
void PageFrameAllocator::free_pages (  
    void * addr,  
    uint64_t page_count )
```

#### 5.16.1.3 get\_free\_mem()

```
uint64_t PageFrameAllocator::get_free_mem ( )
```

#### 5.16.1.4 get\_reserved\_mem()

```
uint64_t PageFrameAllocator::get_reserved_mem ( )
```

#### 5.16.1.5 get\_used\_mem()

```
uint64_t PageFrameAllocator::get_used_mem ( )
```

#### 5.16.1.6 lock\_page()

```
void PageFrameAllocator::lock_page (
    void * addr )
```

#### 5.16.1.7 lock\_pages()

```
void PageFrameAllocator::lock_pages (
    void * addr,
    uint64_t page_count )
```

#### 5.16.1.8 read\_efi\_memory\_map()

```
void PageFrameAllocator::read_efi_memory_map (
    EFI_MEMORY_DESCRIPTOR * mem_map,
    size_t mem_map_size,
    size_t mem_map_desc_size )
```

#### 5.16.1.9 request\_page()

```
void * PageFrameAllocator::request_page ( )
```

## 5.16.2 Member Data Documentation

### 5.16.2.1 page\_bitmap

`Bitmap PageFrameAllocator::page_bitmap`

The documentation for this class was generated from the following files:

- [page\\_frame\\_allocator.h](#)
- [page\\_frame\\_allocator.cpp](#)

## 5.17 PageMapIndexer Class Reference

```
#include <page_map_indexer.h>
```

### Public Member Functions

- [PageMapIndexer](#) (uint64\_t virt\_addr)

### Public Attributes

- uint64\_t [pdp\\_i](#)
- uint64\_t [pd\\_i](#)
- uint64\_t [pt\\_i](#)
- uint64\_t [p\\_i](#)

## 5.17.1 Constructor & Destructor Documentation

### 5.17.1.1 PageMapIndexer()

```
PageMapIndexer::PageMapIndexer (  
    uint64_t virt_addr )
```

## 5.17.2 Member Data Documentation

### 5.17.2.1 p\_i

```
uint64_t PageMapIndexer::p_i
```

### 5.17.2.2 pd\_i

```
uint64_t PageMapIndexer::pd_i
```

### 5.17.2.3 pdp\_i

```
uint64_t PageMapIndexer::pdp_i
```

### 5.17.2.4 pt\_i

```
uint64_t PageMapIndexer::pt_i
```

The documentation for this class was generated from the following files:

- [page\\_map\\_indexer.h](#)
- [page\\_map\\_indexer.cpp](#)

## 5.18 PageTable Struct Reference

```
#include <paging.h>
```

### Public Attributes

- [PageDirEntry](#) [entries](#) [512]

### 5.18.1 Member Data Documentation

#### 5.18.1.1 entries

```
PageDirEntry PageTable::entries[512]
```

The documentation for this struct was generated from the following file:

- [paging.h](#)



## 5.19 PageTableManager Class Reference

```
#include <page_table_manager.h>
```

### Public Member Functions

- [PageTableManager](#) ([PageTable](#) \*[pml4\\_addr](#))
- void [map\\_mem](#) (void \*[virt\\_mem](#), void \*[phys\\_mem](#))

### Public Attributes

- [PageTable](#) \* [pml4\\_addr](#)

### 5.19.1 Constructor & Destructor Documentation

#### 5.19.1.1 PageTableManager()

```
PageTableManager::PageTableManager (  
    PageTable * pml4\_addr )
```

### 5.19.2 Member Function Documentation

#### 5.19.2.1 map\_mem()

```
void PageTableManager::map_mem (  
    void * virt\_mem,  
    void * phys\_mem )
```

### 5.19.3 Member Data Documentation

#### 5.19.3.1 pml4\_addr

```
PageTable* PageTableManager::pml4_addr
```

The documentation for this class was generated from the following files:

- [page\\_table\\_manager.h](#)
- [page\\_table\\_manager.cpp](#)

## 5.20 PCI::PCIDeviceHeader Struct Reference

```
#include <pci.h>
```

### Public Attributes

- uint16\_t [vendor\\_id](#)
- uint16\_t [device\\_id](#)
- uint16\_t [command](#)
- uint16\_t [status](#)
- uint8\_t [revision\\_id](#)
- uint8\_t [prog\\_if](#)
- uint8\_t [subclass](#)
- uint8\_t [class\\_code](#)
- uint8\_t [cache\\_line\\_size](#)
- uint8\_t [latency\\_timer](#)
- uint8\_t [header\\_type](#)
- uint8\_t [bist](#)

### 5.20.1 Member Data Documentation

#### 5.20.1.1 bist

```
uint8_t PCI::PCIDeviceHeader::bist
```

#### 5.20.1.2 cache\_line\_size

```
uint8_t PCI::PCIDeviceHeader::cache_line_size
```

#### 5.20.1.3 class\_code

```
uint8_t PCI::PCIDeviceHeader::class_code
```

#### 5.20.1.4 command

```
uint16_t PCI::PCIDeviceHeader::command
```

#### 5.20.1.5 device\_id

uint16\_t PCI::PCIDeviceHeader::device\_id

#### 5.20.1.6 header\_type

uint8\_t PCI::PCIDeviceHeader::header\_type

#### 5.20.1.7 latency\_timer

uint8\_t PCI::PCIDeviceHeader::latency\_timer

#### 5.20.1.8 prog\_if

uint8\_t PCI::PCIDeviceHeader::prog\_if

#### 5.20.1.9 revision\_id

uint8\_t PCI::PCIDeviceHeader::revision\_id

#### 5.20.1.10 status

uint16\_t PCI::PCIDeviceHeader::status

#### 5.20.1.11 subclass

uint8\_t PCI::PCIDeviceHeader::subclass

### 5.20.1.12 vendor\_id

```
uint16_t PCI::PCIDeviceHeader::vendor_id
```

The documentation for this struct was generated from the following file:

- [pci.h](#)

## 5.21 Point Struct Reference

```
#include <math_util.h>
```

### Public Attributes

- long [x](#)
- long [y](#)

### 5.21.1 Member Data Documentation

#### 5.21.1.1 x

```
long Point::x
```

#### 5.21.1.2 y

```
long Point::y
```

The documentation for this struct was generated from the following file:

- [math\\_util.h](#)

## 5.22 PSF1\_FONT Struct Reference

```
#include <simple_font.h>
```

### Public Attributes

- [PSF1\\_HEADER](#) \* [psf1\\_header](#)
- void \* [glyph\\_buffer](#)

## 5.22.1 Member Data Documentation

### 5.22.1.1 glyph\_buffer

```
void* PSF1_FONT::glyph_buffer
```

### 5.22.1.2 psf1\_header

```
PSF1_HEADER* PSF1_FONT::psf1_header
```

The documentation for this struct was generated from the following file:

- [simple\\_font.h](#)

## 5.23 PSF1\_HEADER Struct Reference

```
#include <simple_font.h>
```

### Public Attributes

- unsigned char [magic](#) [2]
- unsigned char [mode](#)
- unsigned char [charsize](#)

## 5.23.1 Member Data Documentation

### 5.23.1.1 charsize

```
unsigned char PSF1_HEADER::charsize
```

### 5.23.1.2 magic

```
unsigned char PSF1_HEADER::magic[2]
```

### 5.23.1.3 mode

```
unsigned char PSF1_HEADER::mode
```

The documentation for this struct was generated from the following file:

- [simple\\_font.h](#)

## 5.24 ACPI::RSDP2 Struct Reference

```
#include <acpi.h>
```

### Public Attributes

- unsigned char [signature](#) [8]
- uint8\_t [checksum](#)
- uint8\_t [oem\\_id](#) [6]
- uint8\_t [revision](#)
- uint32\_t [rsdt\\_addr](#)
- uint32\_t [length](#)
- uint64\_t [xsdt\\_adrr](#)
- uint8\_t [extended\\_checksum](#)
- uint8\_t [reserved](#) [3]

### 5.24.1 Member Data Documentation

#### 5.24.1.1 checksum

```
uint8_t ACPI::RSDP2::checksum
```

#### 5.24.1.2 extended\_checksum

```
uint8_t ACPI::RSDP2::extended_checksum
```

#### 5.24.1.3 length

```
uint32_t ACPI::RSDP2::length
```

#### 5.24.1.4 oem\_id

```
uint8_t ACPI::RSDP2::oem_id[6]
```

#### 5.24.1.5 reserved

```
uint8_t ACPI::RSDP2::reserved[3]
```

#### 5.24.1.6 revision

```
uint8_t ACPI::RSDP2::revision
```

#### 5.24.1.7 rsdt\_addr

```
uint32_t ACPI::RSDP2::rsdt_addr
```

#### 5.24.1.8 signature

```
unsigned char ACPI::RSDP2::signature[8]
```

#### 5.24.1.9 xsdt\_addr

```
uint64_t ACPI::RSDP2::xsdt_addr
```

The documentation for this struct was generated from the following file:

- [acpi.h](#)

## 5.25 ACPI::SDTHeader Struct Reference

```
#include <acpi.h>
```

## Public Attributes

- unsigned char [signature](#) [4]
- uint32\_t [length](#)
- uint8\_t [revision](#)
- uint8\_t [checksum](#)
- uint8\_t [oem\\_id](#) [6]
- uint8\_t [oem\\_table\\_id](#) [8]
- uint32\_t [oem\\_revision](#)
- uint32\_t [creator\\_id](#)
- uint32\_t [creator\\_revision](#)

## 5.25.1 Member Data Documentation

### 5.25.1.1 checksum

uint8\_t ACPI::SDTHeader::checksum

### 5.25.1.2 creator\_id

uint32\_t ACPI::SDTHeader::creator\_id

### 5.25.1.3 creator\_revision

uint32\_t ACPI::SDTHeader::creator\_revision

### 5.25.1.4 length

uint32\_t ACPI::SDTHeader::length

### 5.25.1.5 oem\_id

uint8\_t ACPI::SDTHeader::oem\_id[6]



#### 5.25.1.6 oem\_revision

```
uint32_t ACPI::SDTHeader::oem_revision
```

#### 5.25.1.7 oem\_table\_id

```
uint8_t ACPI::SDTHeader::oem_table_id[8]
```

#### 5.25.1.8 revision

```
uint8_t ACPI::SDTHeader::revision
```

#### 5.25.1.9 signature

```
unsigned char ACPI::SDTHeader::signature[4]
```

The documentation for this struct was generated from the following file:

- [acpi.h](#)



## Chapter 6

# File Documentation

### 6.1 acpi.cpp File Reference

```
#include "acpi.h"
```

#### Namespaces

- namespace [ACPI](#)

#### Functions

- void \* [ACPI::find\\_table](#) (SDTHeader \*sdt\_header, char \*[signature](#))

### 6.2 acpi.h File Reference

```
#include <stdint.h>
```

#### Classes

- struct [ACPI::RSDP2](#)
- struct [ACPI::SDTHeader](#)
- struct [ACPI::MCFGHeader](#)
- struct [ACPI::DeviceConfig](#)

#### Namespaces

- namespace [ACPI](#)

## Functions

- struct [ACPI::RSDP2 ACPI::\\_\\_attribute\\_\\_](#) ((packed))
- void \* [ACPI::find\\_table](#) (SDTHeader \*sdt\_header, char \*signature)

## Variables

- unsigned char [signature](#) [8]
- uint8\_t [checksum](#)
- uint8\_t [oem\\_id](#) [6]
- uint8\_t [revision](#)
- uint32\_t [rsdt\\_addr](#)
- uint32\_t [length](#)
- uint64\_t [xsdt\\_adrr](#)
- uint8\_t [extended\\_checksum](#)
- uint8\_t [reserved](#) [3]
- uint8\_t [oem\\_table\\_id](#) [8]
- uint32\_t [oem\\_revision](#)
- uint32\_t [creator\\_id](#)
- uint32\_t [creator\\_revision](#)
- SDTHeader [header](#)
- uint64\_t [base\\_addr](#)
- uint16\_t [pci\\_seg\\_group](#)
- uint8\_t [start\\_bus](#)
- uint8\_t [end\\_bus](#)

## 6.2.1 Variable Documentation

### 6.2.1.1 base\_addr

uint64\_t base\_addr

### 6.2.1.2 checksum

uint8\_t checksum

### 6.2.1.3 creator\_id

uint32\_t creator\_id

#### 6.2.1.4 creator\_revision

```
uint32_t creator_revision
```

#### 6.2.1.5 end\_bus

```
uint8_t end_bus
```

#### 6.2.1.6 extended\_checksum

```
uint8_t extended_checksum
```

#### 6.2.1.7 header

```
SDTHeader header
```

#### 6.2.1.8 length

```
uint32_t length
```

#### 6.2.1.9 oem\_id

```
uint8_t oem_id
```

#### 6.2.1.10 oem\_revision

```
uint32_t oem_revision
```

#### 6.2.1.11 oem\_table\_id

```
uint8_t oem_table_id[8]
```

**6.2.1.12 pci\_seg\_group**

uint16\_t pci\_seg\_group

**6.2.1.13 reserved**

uint32\_t reserved

**6.2.1.14 revision**

uint8\_t revision

**6.2.1.15 rsdt\_addr**

uint32\_t rsdt\_addr

**6.2.1.16 signature**

unsigned char signature

**6.2.1.17 start\_bus**

uint8\_t start\_bus

**6.2.1.18 xsdt\_adrr**

uint64\_t xsdt\_adrr

## 6.3 acpi.h

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002
00003 #include <stdint.h>
00004
00005 namespace ACPI
00006 {
00007     struct RSDP2
00008     {
00009         unsigned char signature[8];
00010         uint8_t checksum;
00011         uint8_t oem_id[6];
00012         uint8_t revision;
00013         uint32_t rsdt_addr;
00014         uint32_t length;
00015         uint64_t xsdt_addr;
00016         uint8_t extended_checksum;
00017         uint8_t reserved[3];
00018     }__attribute__((packed));
00019
00020     struct SDTHeader
00021     {
00022         unsigned char signature[4];
00023         uint32_t length;
00024         uint8_t revision;
00025         uint8_t checksum;
00026         uint8_t oem_id[6];
00027         uint8_t oem_table_id[8];
00028         uint32_t oem_revision;
00029         uint32_t creator_id;
00030         uint32_t creator_revision;
00031     }__attribute__((packed));
00032
00033     struct MCFGHeader // contains information about the pci bus
00034     {
00035         SDTHeader header;
00036         uint64_t reserved;
00037     }__attribute__((packed));
00038
00039     struct DeviceConfig
00040     {
00041         uint64_t base_addr;
00042         uint16_t pci_seg_group;
00043         uint8_t start_bus;
00044         uint8_t end_bus;
00045         uint32_t reserved;
00046     }__attribute__((packed));
00047
00048     void* find_table(SDTHeader* sdt_header, char* signature);
00049 }

```

## 6.4 basic\_renderer.cpp File Reference

```
#include "basic_renderer.h"
```

### Variables

- [BasicRenderer](#) \* [global\\_renderer](#)

### 6.4.1 Variable Documentation

### 6.4.1.1 global\_renderer

```
BasicRenderer* global_renderer
```

## 6.5 basic\_renderer.h File Reference

```
#include <stdint.h>
#include "math_util.h"
#include "framebuffer.h"
#include "simple_font.h"
```

### Classes

- class [BasicRenderer](#)

### Variables

- [BasicRenderer](#) \* [global\\_renderer](#)

### 6.5.1 Variable Documentation

#### 6.5.1.1 global\_renderer

```
BasicRenderer* global_renderer [extern]
```

## 6.6 basic\_renderer.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <stdint.h>
00004
00005 #include "math_util.h"
00006 #include "framebuffer.h"
00007 #include "simple_font.h"
00008
00009 class BasicRenderer
00010 {
00011 public:
00012     BasicRenderer(Framebuffer* target_framebuffer, PSF1_FONT* psfl_font, unsigned int color)
00013     {
00014         this->target_framebuffer = target_framebuffer;
00015         this->psfl_font = psfl_font;
00016         this->color = color;
00017         this->cursor_position = { 0,0 };
00018     };
00019
00020     Point cursor_position;
00021     Framebuffer* target_framebuffer;
00022     PSF1_FONT* psfl_font;
00023     unsigned int color;
00024     unsigned int clear_color;
00025     void print(const char* str);
00026     void draw_char(char chr, unsigned int xOff, unsigned int yOff);
00027     void draw_char(char chr);
00028     void clear();
00029     void clear_char();
00030     void nextln();
00031 };
00032
00033 extern BasicRenderer* global_renderer;
```



## 6.7 bitmap.cpp File Reference

```
#include "bitmap.h"
```

## 6.8 bitmap.h File Reference

```
#include <stddef.h>
#include <stdint.h>
```

### Classes

- class [Bitmap](#)

## 6.9 bitmap.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <stddef.h>
00004 #include <stdint.h>
00005
00006 class Bitmap
00007 {
00008 public:
00009     size_t size;
00010     uint8_t* buffer;
00011     bool operator[] (uint64_t index);
00012     bool set(uint64_t index, bool value);
00013 };
```

## 6.10 c\_str.cpp File Reference

```
#include "c_str.h"
```

### Functions

- const char \* [to\\_string](#) (int64\_t value)
- const char \* [to\\_string](#) (uint64\_t value)
- const char \* [to\\_string](#) (double value, uint8\_t precision)
- const char \* [to\\_string](#) (double value)
- const char \* [to\\_hstring](#) (uint64\_t value)
- const char \* [to\\_hstring](#) (uint32\_t value)
- const char \* [to\\_hstring](#) (uint16\_t value)
- const char \* [to\\_hstring](#) (uint8\_t value)

## Variables

- char [int\\_to\\_string\\_output](#) [128]
- char [uint\\_to\\_string\\_output](#) [128]
- char [double\\_to\\_string\\_output](#) [128]
- char [hex\\_to\\_string\\_output\\_64](#) [128]
- char [hex\\_to\\_string\\_output\\_32](#) [128]
- char [hex\\_to\\_string\\_output\\_16](#) [128]
- char [hex\\_to\\_string\\_output\\_8](#) [128]

## 6.10.1 Function Documentation

### 6.10.1.1 to\_hstring() [1/4]

```
const char * to_hstring (  
    uint16_t value )
```

### 6.10.1.2 to\_hstring() [2/4]

```
const char * to_hstring (  
    uint32_t value )
```

### 6.10.1.3 to\_hstring() [3/4]

```
const char * to_hstring (  
    uint64_t value )
```

### 6.10.1.4 to\_hstring() [4/4]

```
const char * to_hstring (  
    uint8_t value )
```

### 6.10.1.5 to\_string() [1/4]

```
const char * to_string (  
    double value )
```

#### 6.10.1.6 to\_string() [2/4]

```
const char * to_string (
    double value,
    uint8_t precision )
```

#### 6.10.1.7 to\_string() [3/4]

```
const char * to_string (
    int64_t value )
```

#### 6.10.1.8 to\_string() [4/4]

```
const char * to_string (
    uint64_t value )
```

### 6.10.2 Variable Documentation

#### 6.10.2.1 double\_to\_string\_output

```
char double_to_string_output[128]
```

#### 6.10.2.2 hex\_to\_string\_output\_16

```
char hex_to_string_output_16[128]
```

#### 6.10.2.3 hex\_to\_string\_output\_32

```
char hex_to_string_output_32[128]
```

#### 6.10.2.4 hex\_to\_string\_output\_64

```
char hex_to_string_output_64[128]
```

#### 6.10.2.5 hex\_to\_string\_output\_8

```
char hex_to_string_output_8[128]
```

#### 6.10.2.6 int\_to\_string\_output

```
char int_to_string_output[128]
```

#### 6.10.2.7 uint\_to\_string\_output

```
char uint_to_string_output[128]
```

## 6.11 c\_str.h File Reference

```
#include <stdint.h>
```

### Functions

- const char \* [to\\_string](#) (int64\_t value)
- const char \* [to\\_string](#) (uint64\_t value)
- const char \* [to\\_string](#) (double value, uint8\_t precision)
- const char \* [to\\_string](#) (double value)
- const char \* [to\\_hstring](#) (uint64\_t value)
- const char \* [to\\_hstring](#) (uint32\_t value)
- const char \* [to\\_hstring](#) (uint16\_t value)
- const char \* [to\\_hstring](#) (uint8\_t value)

### 6.11.1 Function Documentation

#### 6.11.1.1 to\_hstring() [1/4]

```
const char * to_hstring (  
    uint16_t value )
```

**6.11.1.2 to\_hstring() [2/4]**

```
const char * to_hstring (
    uint32_t value )
```

**6.11.1.3 to\_hstring() [3/4]**

```
const char * to_hstring (
    uint64_t value )
```

**6.11.1.4 to\_hstring() [4/4]**

```
const char * to_hstring (
    uint8_t value )
```

**6.11.1.5 to\_string() [1/4]**

```
const char * to_string (
    double value )
```

**6.11.1.6 to\_string() [2/4]**

```
const char * to_string (
    double value,
    uint8_t precision )
```

**6.11.1.7 to\_string() [3/4]**

```
const char * to_string (
    int64_t value )
```

**6.11.1.8 to\_string() [4/4]**

```
const char * to_string (
    uint64_t value )
```

## 6.12 c\_str.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <stdint.h>
00004
00005 const char* to_string(int64_t value);
00006 const char* to_string(uint64_t value);
00007 const char* to_string(double value, uint8_t precision);
00008 const char* to_string(double value);
00009 const char* to_hstring(uint64_t value);
00010 const char* to_hstring(uint32_t value);
00011 const char* to_hstring(uint16_t value);
00012 const char* to_hstring(uint8_t value);
```

## 6.13 efi\_memory.cpp File Reference

```
#include "efi_memory.h"
```

### Variables

- const char \* [EFI\\_MEMORY\\_TYPE\\_STRINGS](#) [ ]

### 6.13.1 Variable Documentation

#### 6.13.1.1 EFI\_MEMORY\_TYPE\_STRINGS

```
const char* EFI_MEMORY_TYPE_STRINGS[ ]
```

#### Initial value:

```
{
    "EfiReservedMemoryType",
    "EfiLoaderCode",
    "EfiLoaderData",
    "EfiBootServicesCode",
    "EfiBootServicesData",
    "EfiRuntimeServicesCode",
    "EfiRuntimeServicesData",
    "EfiConventionalMemory",
    "EfiUnusableMemory",
    "EfiACPIReclaimMemory",
    "EfiACPIMemoryNVS",
    "EfiMemoryMappedIO",
    "EfiMemoryMappedIOPortSpace",
    "EfiPalCode",
}
```

## 6.14 efi\_memory.h File Reference

```
#include <stdint.h>
```

## Classes

- struct [EFI\\_MEMORY\\_DESCRIPTOR](#)

## Variables

- const char \* [EFI\\_MEMORY\\_TYPE\\_STRINGS](#) []

### 6.14.1 Variable Documentation

#### 6.14.1.1 EFI\_MEMORY\_TYPE\_STRINGS

```
const char* EFI_MEMORY_TYPE_STRINGS[] [extern]
```

## 6.15 efi\_memory.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <stdint.h>
00004
00005 struct EFI_MEMORY_DESCRIPTOR
00006 {
00007     uint32_t type;
00008     void* phys_addr;
00009     void* virt_addr;
00010     uint64_t num_pages;
00011     uint64_t attrib;
00012 };
00013
00014 extern const char* EFI_MEMORY_TYPE_STRINGS[];
```

## 6.16 framebuffer.h File Reference

```
#include <stddef.h>
```

## Classes

- struct [Framebuffer](#)

## 6.17 framebuffer.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <stddef.h>
00004
00005 struct Framebuffer
00006 {
00007     void* base_address;
00008     size_t buffer_size;
00009     unsigned int width;
00010     unsigned int height;
00011     unsigned int pixels_per_scanline;
00012 };
```

## 6.18 gdt.cpp File Reference

```
#include "gdt.h"
```

### Functions

- [\\_\\_attribute\\_\\_](#) ((aligned(4096))) [GDT default\\_gdt](#)

### 6.18.1 Function Documentation

#### 6.18.1.1 [\\_\\_attribute\\_\\_](#)()

```
\_\_attribute\_\_ (
    (aligned(4096)) )
```

## 6.19 gdt.h File Reference

```
#include <stdint.h>
```

### Classes

- struct [GDTDescriptor](#)
- struct [GDTEEntry](#)
- struct [GDT](#)

### Functions

- struct [GDTDescriptor](#) [\\_\\_attribute\\_\\_](#) ((packed))
- void [load\\_gdt](#) ([GDTDescriptor](#) \*gtd\_desc)

### Variables

- [uint16\\_t](#) [size](#)
- [uint64\\_t](#) [offset](#)
- [uint16\\_t](#) [limit0](#)
- [uint16\\_t](#) [base0](#)
- [uint8\\_t](#) [base1](#)
- [uint8\\_t](#) [access\\_byte](#)
- [uint8\\_t](#) [limit1\\_flags](#)
- [uint8\\_t](#) [base2](#)
- [GDTEEntry](#) [null](#)
- [GDTEEntry](#) [kernel\\_code](#)
- [GDTEEntry](#) [kernel\\_data](#)
- [GDTEEntry](#) [user\\_null](#)
- [GDTEEntry](#) [user\\_code](#)
- [GDTEEntry](#) [user\\_data](#)
- [GDT](#) [default\\_gdt](#)



## 6.19.1 Function Documentation

### 6.19.1.1 `__attribute__()`

```
struct GDT __attribute__ (  
    (packed) )
```

### 6.19.1.2 `load_gdt()`

```
void load_gdt (  
    GDTDescriptor * gtd_desc )
```

## 6.19.2 Variable Documentation

### 6.19.2.1 `access_byte`

```
uint8_t access_byte
```

### 6.19.2.2 `base0`

```
uint16_t base0
```

### 6.19.2.3 `base1`

```
uint8_t base1
```

### 6.19.2.4 `base2`

```
uint8_t base2
```

#### 6.19.2.5 default\_gdt

`GDT default_gdt [extern]`

#### 6.19.2.6 kernel\_code

`GDTEntry kernel_code`

#### 6.19.2.7 kernel\_data

`GDTEntry kernel_data`

#### 6.19.2.8 limit0

`uint16_t limit0`

#### 6.19.2.9 limit1\_flags

`uint8_t limit1_flags`

#### 6.19.2.10 null

`GDTEntry null`

#### 6.19.2.11 offset

`uint64_t offset`

#### 6.19.2.12 size

`uint16_t size`

**6.19.2.13 user\_code**

```
GDTEEntry user_code
```

**6.19.2.14 user\_data**

```
GDTEEntry user_data
```

**6.19.2.15 user\_null**

```
GDTEEntry user_null
```

**6.20 gdt.h**

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <stdint.h>
00004
00005 struct GDTDescriptor
00006 {
00007     uint16_t size;
00008     uint64_t offset;
00009 }__attribute__((packed));
00010
00011 struct GDTEEntry
00012 {
00013     uint16_t limit0;
00014     uint16_t base0;
00015     uint8_t base1;
00016     uint8_t access_byte;
00017     uint8_t limit1_flags;
00018     uint8_t base2;
00019 }__attribute__((packed));
00020
00021 struct GDT
00022 {
00023     GDTEEntry null;
00024     GDTEEntry kernel_code;
00025     GDTEEntry kernel_data;
00026     GDTEEntry user_null;
00027     GDTEEntry user_code;
00028     GDTEEntry user_data;
00029 }__attribute__((packed))
00030 __attribute__((aligned(4096)));
00031
00032 extern GDT default_gdt;
00033
00034 extern "C" void load_gdt(GDTDescriptor* gtd_desc);
```

**6.21 kb\_scancode\_trans.cpp File Reference**

```
#include "kb_scancode_trans.h"
```

## Namespaces

- namespace [QWERTZKeyboard](#)

## Functions

- char [QWERTZKeyboard::translate](#) (uint8\_t scancode, bool uppercase)

## Variables

- const char [QWERTZKeyboard::ascii\\_table](#) []

## 6.22 kb\_scancode\_trans.h File Reference

```
#include <stdint.h>
```

## Namespaces

- namespace [QWERTZKeyboard](#)

## Macros

- #define [LEFT\\_SHIFT](#) 0x2a
- #define [RIGHT\\_SHIFT](#) 0x36
- #define [ENTER](#) 0x1c
- #define [BACKSPACE](#) 0x0e
- #define [SPACE](#) 0x39

## Functions

- char [QWERTZKeyboard::translate](#) (uint8\_t scancode, bool uppercase)

### 6.22.1 Macro Definition Documentation

#### 6.22.1.1 BACKSPACE

```
#define BACKSPACE 0x0e
```

### 6.22.1.2 ENTER

```
#define ENTER 0x1c
```

### 6.22.1.3 LEFT\_SHIFT

```
#define LEFT_SHIFT 0x2a
```

### 6.22.1.4 RIGHT\_SHIFT

```
#define RIGHT_SHIFT 0x36
```

### 6.22.1.5 SPACE

```
#define SPACE 0x39
```

## 6.23 kb\_scancode\_trans.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <stdint.h>
00004
00005 namespace QWERTZKeyboard
00006 {
00007     #define LEFT_SHIFT 0x2a
00008     #define RIGHT_SHIFT 0x36
00009     #define ENTER 0x1c
00010     #define BACKSPACE 0x0e
00011     #define SPACE 0x39
00012
00013     extern const char ascii_table[];
00014     char translate(uint8_t scancode, bool uppercase);
00015 }
```

## 6.24 keyboard.cpp File Reference

```
#include "keyboard.h"
```

### Functions

- void [handle\\_keyboard](#) (uint8\_t scancode)

## Variables

- bool [left\\_shift\\_pressed](#)
- bool [right\\_shift\\_pressed](#)

### 6.24.1 Function Documentation

#### 6.24.1.1 `handle_keyboard()`

```
void handle_keyboard (
    uint8_t scancode )
```

### 6.24.2 Variable Documentation

#### 6.24.2.1 `left_shift_pressed`

```
bool left_shift_pressed
```

#### 6.24.2.2 `right_shift_pressed`

```
bool right_shift_pressed
```

## 6.25 keyboard.h File Reference

```
#include <stdint.h>
#include "kb_scancode_trans.h"
#include "../basic_renderer.h"
```

## Functions

- void [handle\\_keyboard](#) (uint8\_t scancode)

### 6.25.1 Function Documentation

### 6.25.1.1 handle\_keyboard()

```
void handle_keyboard (
    uint8_t scancode )
```

## 6.26 keyboard.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <stdint.h>
00004
00005 #include "kb_scancode_trans.h"
00006 #include "../basic_renderer.h"
00007
00008 void handle_keyboard(uint8_t scancode);
```

## 6.27 idt.cpp File Reference

```
#include "idt.h"
```

## 6.28 idt.h File Reference

```
#include <stdint.h>
```

### Classes

- struct [IDTDescEntry](#)
- struct [IDTR](#)

### Macros

- #define [IDT\\_TA\\_INTERRUPTGATE](#) 0b10001110
- #define [IDT\\_TA\\_CALLGATE](#) 0b10001100
- #define [IDT\\_TA\\_TRAPGATE](#) 0b10001111

### Functions

- struct [IDTR](#) [\\_\\_attribute\\_\\_\(\(packed\)\)](#)

### Variables

- uint16\_t [limit](#)
- uint64\_t [offset](#)

## 6.28.1 Macro Definition Documentation

### 6.28.1.1 IDT\_TA\_CALLGATE

```
#define IDT_TA_CALLGATE 0b10001100
```

### 6.28.1.2 IDT\_TA\_INTERRUPTGATE

```
#define IDT_TA_INTERRUPTGATE 0b10001110
```

### 6.28.1.3 IDT\_TA\_TRAPGATE

```
#define IDT_TA_TRAPGATE 0b10001111
```

## 6.28.2 Function Documentation

### 6.28.2.1 \_\_attribute\_\_()

```
struct IDTR __attribute__ (  
    (packed) )
```

## 6.28.3 Variable Documentation

### 6.28.3.1 limit

```
uint16_t limit
```

### 6.28.3.2 offset

```
uint64_t offset
```



## 6.29 idt.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <stdint.h>
00004
00005 #define IDT_TA_INTERRUPTGATE 0b10001110
00006 #define IDT_TA_CALLGATE     0b10001100
00007 #define IDT_TA_TRAPGATE     0b10001111
00008
00009 struct IDTDescEntry
00010 {
00011     uint16_t offset0;
00012     uint16_t selector;
00013     uint8_t ist;          // interrupt stack table
00014     uint8_t type_attr;
00015     uint16_t offset1;
00016     uint32_t offset2;
00017     uint32_t ignore;
00018
00019     void set_offset(uint64_t offset);
00020     uint64_t get_offset();
00021 };
00022
00023 struct IDTR
00024 {
00025     uint16_t limit;
00026     uint64_t offset;
00027 } __attribute__((packed));
```

## 6.30 interrupts.cpp File Reference

```
#include "interrupts.h"
```

### Functions

- `__attribute__((interrupt)) void page_fault_handler(InterruptFrame *frame)`
- `void pic_end_master ()`
- `void pic_end_slave ()`
- `void remap_pic ()`

### 6.30.1 Function Documentation

#### 6.30.1.1 \_\_attribute\_\_((interrupt))

```
__attribute__((interrupt)) (
    (interrupt) )
```

#### 6.30.1.2 pic\_end\_master()

```
void pic_end_master ( )
```

### 6.30.1.3 pic\_end\_slave()

```
void pic_end_slave ( )
```

### 6.30.1.4 remap\_pic()

```
void remap_pic ( )
```

## 6.31 interrupts.h File Reference

```
#include "../basic_renderer.h"  
#include "../panic_screen.h"  
#include "../io.h"  
#include "../input/keyboard.h"  
#include "../scheduling/pit/pit.h"
```

### Macros

- #define [PIC1\\_CMD](#) 0x20
- #define [PIC1\\_DATA](#) 0x21
- #define [PIC2\\_CMD](#) 0xA0
- #define [PIC2\\_DATA](#) 0xA1
- #define [PIC\\_EOI](#) 0x20
- #define [ICW1\\_INIT](#) 0x10
- #define [ICW1\\_ICW4](#) 0x01
- #define [ICW4\\_8086](#) 0x01

### Functions

- [\\_\\_attribute\\_\\_](#)((interrupt)) void [page\\_fault\\_handler](#)(InterruptFrame \*frame)
- void [remap\\_pic](#) ()
- void [pic\\_end\\_master](#) ()
- void [pic\\_end\\_slave](#) ()

### 6.31.1 Macro Definition Documentation

#### 6.31.1.1 ICW1\_ICW4

```
#define ICW1_ICW4 0x01
```

### 6.31.1.2 ICW1\_INIT

```
#define ICW1_INIT 0x10
```

### 6.31.1.3 ICW4\_8086

```
#define ICW4_8086 0x01
```

### 6.31.1.4 PIC1\_CMD

```
#define PIC1_CMD 0x20
```

### 6.31.1.5 PIC1\_DATA

```
#define PIC1_DATA 0x21
```

### 6.31.1.6 PIC2\_CMD

```
#define PIC2_CMD 0xA0
```

### 6.31.1.7 PIC2\_DATA

```
#define PIC2_DATA 0xA1
```

### 6.31.1.8 PIC\_EOI

```
#define PIC_EOI 0x20
```

## 6.31.2 Function Documentation

### 6.31.2.1 \_\_attribute\_\_((interrupt))

```
__attribute__((interrupt))
```

### 6.31.2.2 pic\_end\_master()

```
void pic_end_master ( )
```

### 6.31.2.3 pic\_end\_slave()

```
void pic_end_slave ( )
```

### 6.31.2.4 remap\_pic()

```
void remap_pic ( )
```

## 6.32 interrupts.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "../basic_renderer.h"
00004 #include "../panic_screen.h"
00005 #include "../io.h"
00006 #include "../input/keyboard.h"
00007 #include "../scheduling/pit/pit.h"
00008
00009 #define PIC1_CMD 0x20
00010 #define PIC1_DATA 0x21
00011 #define PIC2_CMD 0xA0
00012 #define PIC2_DATA 0xA1
00013 #define PIC_EOI 0x20
00014
00015 #define ICW1_INIT 0x10
00016 #define ICW1_ICW4 0x01
00017 #define ICW4_8086 0x01
00018
00019 struct InterruptFrame;
00020 __attribute__((interrupt)) void page_fault_handler(InterruptFrame* frame);
00021 __attribute__((interrupt)) void double_fault_handler(InterruptFrame* frame);
00022 __attribute__((interrupt)) void gp_fault_handler(InterruptFrame* frame);
00023 __attribute__((interrupt)) void keyboard_int_handler(InterruptFrame* frame);
00024 __attribute__((interrupt)) void pit_int_handler(InterruptFrame* frame);
00025
00026 void remap_pic();
00027 void pic_end_master();
00028 void pic_end_slave();
```

## 6.33 io.cpp File Reference

```
#include "io.h"
```

## Functions

- void [outb](#) (uint16\_t port, uint8\_t value)
- uint8\_t [inb](#) (uint16\_t port)
- void [io\\_wait](#) ()

### 6.33.1 Function Documentation

#### 6.33.1.1 inb()

```
uint8_t inb (
    uint16_t port )
```

#### 6.33.1.2 io\_wait()

```
void io_wait ( )
```

#### 6.33.1.3 outb()

```
void outb (
    uint16_t port,
    uint8_t value )
```

## 6.34 io.h File Reference

```
#include <stdint.h>
```

## Functions

- void [outb](#) (uint16\_t port, uint8\_t value)
- uint8\_t [inb](#) (uint16\_t port)
- void [io\\_wait](#) ()

### 6.34.1 Function Documentation

#### 6.34.1.1 inb()

```
uint8_t inb (
    uint16_t port )
```

#### 6.34.1.2 io\_wait()

```
void io_wait ( )
```

#### 6.34.1.3 outb()

```
void outb (
    uint16_t port,
    uint8_t value )
```

### 6.35 io.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <stdint.h>
00004
00005 void outb(uint16_t port, uint8_t value);
00006 uint8_t inb(uint16_t port);
00007 void io_wait();
```

### 6.36 kernel.cpp File Reference

```
#include "kernel_util.h"
#include "memory/heap.h"
#include "scheduling/pit/pit.h"
```

#### Functions

- void [start](#) ([BOOT\\_INFO](#) \*boot\_info)

#### 6.36.1 Function Documentation

### 6.36.1.1 start()

```
void start (
    BOOT\_INFO * boot_info )
```

## 6.37 kernel\_util.cpp File Reference

```
#include "kernel_util.h"
```

### Functions

- void [prepare\\_mem](#) ([BOOT\\_INFO](#) \*boot\_info)
- void [set\\_idt\\_gate](#) (void \*handler, uint8\_t entry\_offset, uint8\_t type\_attr, uint8\_t selector)
- void [prepare\\_interrupts](#) ()
- void [prepare\\_acpi](#) ([BOOT\\_INFO](#) \*boot\_info)
- [KernelInfo](#) [init\\_kernel](#) ([BOOT\\_INFO](#) \*boot\_info)

### Variables

- [KernelInfo](#) [kernel\\_info](#)
- [IDTR](#) [idtr](#)
- [BasicRenderer](#) [r](#) = [BasicRenderer](#)(NULL, NULL, 0xffffffff)

## 6.37.1 Function Documentation

### 6.37.1.1 init\_kernel()

```
KernelInfo init_kernel (
    BOOT\_INFO * boot_info )
```

### 6.37.1.2 prepare\_acpi()

```
void prepare_acpi (
    BOOT\_INFO * boot_info )
```

### 6.37.1.3 prepare\_interrupts()

```
void prepare_interrupts ( )
```

#### 6.37.1.4 prepare\_mem()

```
void prepare_mem (
    BOOT_INFO * boot_info )
```

#### 6.37.1.5 set\_idt\_gate()

```
void set_idt_gate (
    void * handler,
    uint8_t entry_offset,
    uint8_t type_attr,
    uint8_t selector )
```

### 6.37.2 Variable Documentation

#### 6.37.2.1 idtr

```
IDTR idtr
```

#### 6.37.2.2 kernel\_info

```
KernelInfo kernel_info
```

#### 6.37.2.3 r

```
BasicRenderer r = BasicRenderer(NULL, NULL, 0xffffffff)
```

## 6.38 kernel\_util.h File Reference

```
#include <stddef.h>
#include "io.h"
#include "pci.h"
#include "acpi.h"
#include "gdt/gdt.h"
#include "c_str.h"
#include "memory.h"
#include "bitmap.h"
#include "efi_memory.h"
#include "interrupts/idt.h"
#include "interrupts/interrupts.h"
#include "basic_renderer.h"
#include "paging/paging.h"
#include "memory/heap.h"
#include "paging/page_map_indexer.h"
#include "paging/page_table_manager.h"
#include "paging/page_frame_allocator.h"
```



## Classes

- struct [BOOT\\_INFO](#)
- struct [KernelInfo](#)

## Functions

- [KernelInfo init\\_kernel](#) ([BOOT\\_INFO](#) \*boot\_info)

## Variables

- uint64\_t [\\_kernel\\_start](#)
- uint64\_t [\\_kernel\\_end](#)

### 6.38.1 Function Documentation

#### 6.38.1.1 init\_kernel()

```
KernelInfo init_kernel (  
    BOOT_INFO * boot_info )
```

### 6.38.2 Variable Documentation

#### 6.38.2.1 \_kernel\_end

```
uint64_t _kernel_end [extern]
```

#### 6.38.2.2 \_kernel\_start

```
uint64_t _kernel_start [extern]
```

## 6.39 kernel\_util.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <stddef.h>
00004
00005 #include "io.h"
00006 #include "pci.h"
00007 #include "acpi.h"
00008 #include "gdt/gdt.h"
00009 #include "c_str.h"
00010 #include "memory.h"
00011 #include "bitmap.h"
00012 #include "efi_memory.h"
00013 #include "interrupts/idt.h"
00014 #include "interrupts/interrupts.h"
00015 #include "basic_renderer.h"
00016 #include "paging/paging.h"
00017 #include "memory/heap.h"
00018 #include "paging/page_map_indexer.h"
00019 #include "paging/page_table_manager.h"
00020 #include "paging/page_frame_allocator.h"
00021
00022 struct BOOT_INFO
00023 {
00024     Framebuffer* framebuffer;
00025     PSF1_FONT* psfl_font;
00026     void* mem_map;
00027     uint64_t mem_map_size;
00028     uint64_t mem_map_descriptor_size;
00029     ACPI::RSDP2* rsdp;
00030 };
00031
00032 struct KernelInfo
00033 {
00034     PageTableManager* page_table_mgr;
00035 };
00036
00037 KernelInfo init_kernel(BOOT_INFO* boot_info);
00038
00039 extern uint64_t _kernel_start;
00040 extern uint64_t _kernel_end;
```

## 6.40 math\_util.h File Reference

### Classes

- struct [Point](#)

## 6.41 math\_util.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 struct Point
00004 {
00005     long x;
00006     long y;
00007 };
```

## 6.42 memory.cpp File Reference

```
#include "memory.h"
```

## Functions

- uint64\_t [get\\_memory\\_size](#) (EFI\_MEMORY\_DESCRIPTOR \*mem\_map, uint64\_t mem\_map\_entries, uint64\_t mem\_map\_desc\_size)
- void [memset](#) (void \*start, uint8\_t value, uint64\_t num)

### 6.42.1 Function Documentation

#### 6.42.1.1 get\_memory\_size()

```
uint64_t get_memory_size (
    EFI_MEMORY_DESCRIPTOR * mem_map,
    uint64_t mem_map_entries,
    uint64_t mem_map_desc_size )
```

#### 6.42.1.2 memset()

```
void memset (
    void * start,
    uint8_t value,
    uint64_t num )
```

## 6.43 memory.h File Reference

```
#include <stdint.h>
#include "efi_memory.h"
```

## Functions

- uint64\_t [get\\_memory\\_size](#) (EFI\_MEMORY\_DESCRIPTOR \*mem\_map, uint64\_t mem\_map\_entries, uint64\_t mem\_map\_desc\_size)
- void [memset](#) (void \*start, uint8\_t value, uint64\_t num)

### 6.43.1 Function Documentation

### 6.43.1.1 `get_memory_size()`

```
uint64_t get_memory_size (
    EFI_MEMORY_DESCRIPTOR * mem_map,
    uint64_t mem_map_entries,
    uint64_t mem_map_desc_size )
```

### 6.43.1.2 `memset()`

```
void memset (
    void * start,
    uint8_t value,
    uint64_t num )
```

## 6.44 `memory.h`

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <stdint.h>
00004 #include "efi_memory.h"
00005
00006 uint64_t get_memory_size(EFI_MEMORY_DESCRIPTOR* mem_map, uint64_t mem_map_entries, uint64_t
    mem_map_desc_size);
00007 void memset(void* start, uint8_t value, uint64_t num);
```

## 6.45 `heap.cpp` File Reference

```
#include "heap.h"
```

### Functions

- void `init_heap` (void \*heap\_addr, size\_t page\_count)
- void `free` (void \*addr)
- void \* `malloc` (size\_t size)
- void `extend_heap` (size\_t length)

### Variables

- void \* `heap_start`
- void \* `heap_end`
- `HeapSegHeader` \* `last_header`

### 6.45.1 Function Documentation

### 6.45.1.1 `extend_heap()`

```
void extend_heap (
    size_t length )
```

### 6.45.1.2 `free()`

```
void free (
    void * addr )
```

### 6.45.1.3 `init_heap()`

```
void init_heap (
    void * heap_addr,
    size_t page_count )
```

### 6.45.1.4 `malloc()`

```
void * malloc (
    size_t size )
```

## 6.45.2 Variable Documentation

### 6.45.2.1 `heap_end`

```
void* heap_end
```

### 6.45.2.2 `heap_start`

```
void* heap_start
```

### 6.45.2.3 `last_header`

```
HeapSegHeader* last_header
```

## 6.46 heap.h File Reference

```
#include <stdint.h>
#include <stddef.h>
#include "../paging/page_table_manager.h"
#include "../paging/page_frame_allocator.h"
```

### Classes

- struct [HeapSegHeader](#)

### Functions

- void [init\\_heap](#) (void \*heap\_addr, size\_t page\_count)
- void \* [malloc](#) (size\_t size)
- void [free](#) (void \*addr)
- void [extend\\_heap](#) (size\_t length)

### 6.46.1 Function Documentation

#### 6.46.1.1 [extend\\_heap\(\)](#)

```
void extend_heap (
    size_t length )
```

#### 6.46.1.2 [free\(\)](#)

```
void free (
    void * addr )
```

#### 6.46.1.3 [init\\_heap\(\)](#)

```
void init_heap (
    void * heap_addr,
    size_t page_count )
```

### 6.46.1.4 malloc()

```
void * malloc (
    size_t size )
```

## 6.47 heap.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <stdint.h>
00004 #include <stddef.h>
00005
00006 #include "../paging/page_table_manager.h"
00007 #include "../paging/page_frame_allocator.h"
00008
00009 struct HeapSegHeader
00010 {
00011     size_t length;
00012     HeapSegHeader* next;
00013     HeapSegHeader* last;
00014     bool free;
00015
00016     void combine_forward();
00017     void combine_backward();
00018     HeapSegHeader* split(size_t split_length);
00019 };
00020
00021 void init_heap(void* heap_addr, size_t page_count);
00022
00023 void* malloc(size_t size);
00024 void free(void* addr);
00025
00026 void extend_heap(size_t length);
```

## 6.48 page\_frame\_allocator.cpp File Reference

```
#include "page_frame_allocator.h"
```

### Variables

- uint64\_t [free\\_memory](#)
- uint64\_t [reserved\\_memory](#)
- uint64\_t [used\\_memory](#)
- bool [initialized](#) = false
- [PageFrameAllocator](#) [global\\_allocator](#)
- uint64\_t [page\\_bitmap\\_index](#) = 0

### 6.48.1 Variable Documentation

#### 6.48.1.1 free\_memory

```
uint64_t free_memory
```

#### 6.48.1.2 global\_allocator

```
PageFrameAllocator global_allocator
```

#### 6.48.1.3 initialized

```
bool initialized = false
```

#### 6.48.1.4 page\_bitmap\_index

```
uint64_t page_bitmap_index = 0
```

#### 6.48.1.5 reserved\_memory

```
uint64_t reserved_memory
```

#### 6.48.1.6 used\_memory

```
uint64_t used_memory
```

## 6.49 page\_frame\_allocator.h File Reference

```
#include <stdint.h>
#include "../bitmap.h"
#include "../memory.h"
#include "../efi_memory.h"
```

### Classes

- class [PageFrameAllocator](#)

### Variables

- [PageFrameAllocator](#) global\_allocator



## 6.49.1 Variable Documentation

### 6.49.1.1 global\_allocator

```
PageFrameAllocator global_allocator [extern]
```

## 6.50 page\_frame\_allocator.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <stdint.h>
00004 #include "../bitmap.h"
00005 #include "../memory.h"
00006 #include "../efi_memory.h"
00007
00008 class PageFrameAllocator
00009 {
00010 public:
00011     void read_efi_memory_map(EFI_MEMORY_DESCRIPTOR* mem_map, size_t mem_map_size, size_t
mem_map_desc_size);
00012     void free_page(void* addr);
00013     void free_pages(void* addr, uint64_t page_count);
00014     void lock_page(void* addr);
00015     void lock_pages(void* addr, uint64_t page_count);
00016
00017     void* request_page();
00018
00019     uint64_t get_free_mem();
00020     uint64_t get_used_mem();
00021     uint64_t get_reserved_mem();
00022     Bitmap page_bitmap;
00023
00024 private:
00025     void init_bitmap(size_t bitmap_size, void* buffer_addr);
00026     void reserve_page(void* addr);
00027     void reserve_pages(void* addr, uint64_t page_count);
00028     void unreserve_page(void* addr);
00029     void unreserve_pages(void* addr, uint64_t page_count);
00030 };
00031
00032 extern PageFrameAllocator global_allocator;
```

## 6.51 page\_map\_indexer.cpp File Reference

```
#include "page_map_indexer.h"
```

## 6.52 page\_map\_indexer.h File Reference

```
#include <stdint.h>
```

### Classes

- class [PageMapIndexer](#)

## 6.53 page\_map\_indexer.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <stdint.h>
00004
00005 class PageMapIndexer
00006 {
00007 public:
00008     PageMapIndexer(uint64_t virt_addr);
00009     uint64_t pdp_i; // Page Directory Pointer Index
00010     uint64_t pd_i; // Page Directory Index
00011     uint64_t pt_i; // Page Table Index
00012     uint64_t p_i; // Page Index
00013 };
```

## 6.54 page\_table\_manager.cpp File Reference

```
#include <stdint.h>
#include "../memory.h"
#include "page_map_indexer.h"
#include "page_table_manager.h"
#include "page_frame_allocator.h"
```

### Variables

- [PageTableManager global\\_ptm](#) = NULL

### 6.54.1 Variable Documentation

#### 6.54.1.1 global\_ptm

```
PageTableManager global_ptm = NULL
```

## 6.55 page\_table\_manager.h File Reference

```
#include "paging.h"
```

### Classes

- class [PageTableManager](#)

### Variables

- [PageTableManager global\\_ptm](#)

## 6.55.1 Variable Documentation

### 6.55.1.1 global\_ptm

```
PageTableManager global_ptm [extern]
```

## 6.56 page\_table\_manager.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "paging.h"
00004
00005 class PageTableManager
00006 {
00007 public:
00008     PageTableManager(PageTable* pml4_addr);
00009     PageTable* pml4_addr;
00010     void map_mem(void* virt_mem, void* phys_mem);
00011 };
00012
00013 extern PageTableManager global_ptm;
```

## 6.57 paging.cpp File Reference

```
#include "paging.h"
```

## 6.58 paging.h File Reference

```
#include <stdint.h>
```

### Classes

- struct [PageDirEntry](#)
- struct [PageTable](#)

### Enumerations

- enum [PT\\_FLAG](#) {  
    [Present](#) = 0 , [ReadWrite](#) = 1 , [UserSuper](#) = 2 , [WriteThrough](#) = 3 ,  
    [CacheDisabled](#) = 4 , [Accessed](#) = 5 , [LargerPages](#) = 7 , [Custom0](#) = 9 ,  
    [Custom1](#) = 10 , [Custom2](#) = 11 , [NX](#) = 63 }

## Functions

- struct [PageTable](#) [\\_\\_attribute\\_\\_](#) ((aligned(4096)))

## Variables

- [PageDirEntry](#) [entries](#) [512]

## 6.58.1 Enumeration Type Documentation

### 6.58.1.1 PT\_FLAG

enum [PT\\_FLAG](#)

#### Enumerator

Present	
ReadWrite	
UserSuper	
WriteThrough	
CacheDisabled	
Accessed	
LargerPages	
Custom0	
Custom1	
Custom2	
NX	

## 6.58.2 Function Documentation

### 6.58.2.1 \_\_attribute\_\_()

```
struct PageTable \_\_attribute\_\_ (  
    (aligned(4096)) )
```

## 6.58.3 Variable Documentation

### 6.58.3.1 entries

```
PageDirEntry entries[512]
```

## 6.59 paging.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <stdint.h>
00004
00005 enum PT_FLAG
00006 {
00007     Present = 0,
00008     ReadWrite = 1,
00009     UserSuper = 2,
00010     WriteThrough = 3,
00011     CacheDisabled = 4,
00012     Accessed = 5,
00013     LargerPages = 7,
00014     Custom0 = 9,
00015     Custom1 = 10,
00016     Custom2 = 11,
00017     NX = 63 // EXPERIMENTAL !! ONLY ON MODERN SYSTEMS
00018 };
00019
00020 struct PageDirEntry
00021 {
00022     uint64_t value;
00023     void set_flag(PT_FLAG flag, bool enabled);
00024     bool get_flag(PT_FLAG flag);
00025     void set_address(uint64_t addr);
00026     uint64_t get_address();
00027 };
00028
00029 struct PageTable
00030 {
00031     PageDirEntry entries[512];
00032
00033 }__attribute__((aligned(4096)));
```

## 6.60 panic\_screen.cpp File Reference

```
#include "panic_screen.h"
#include "basic_renderer.h"
#include "c_str.h"
```

### Functions

- void `panic` (const char \*msg)

### 6.60.1 Function Documentation

#### 6.60.1.1 panic()

```
void panic (
    const char * msg )
```

## 6.61 panic\_screen.h File Reference

### Functions

- void [panic](#) (const char \*msg)

### 6.61.1 Function Documentation

#### 6.61.1.1 panic()

```
void panic (  
    const char * msg )
```

## 6.62 panic\_screen.h

[Go to the documentation of this file.](#)

```
00001 #pragma once  
00002  
00003 void panic(const char* msg);
```

## 6.63 pci.cpp File Reference

```
#include "pci.h"
```

### Namespaces

- namespace [PCI](#)

### Functions

- void [PCI::enumerate\\_function](#) (uint64\_t device\_addr, uint64\_t function)
- void [PCI::enumerate\\_device](#) (uint64\_t bus\_addr, uint64\_t device)
- void [PCI::enumerate\\_bus](#) (uint64\_t [base\\_addr](#), uint64\_t bus)
- void [PCI::enumerate\\_pci](#) ([ACPI::MCFGHeader](#) \*mcfg)

## 6.64 pci.h File Reference

```
#include <stdint.h>  
#include "acpi.h"  
#include "c_str.h"  
#include "basic_renderer.h"  
#include "paging/page_table_manager.h"
```

## Classes

- struct [PCI::PCIDeviceHeader](#)

## Namespaces

- namespace [PCI](#)

## Functions

- void [PCI::enumerate\\_pci](#) ([ACPI::MCFGHeader](#) \*mcfg)
- const char \* [PCI::get\\_vendor\\_name](#) (uint16\_t vendor\_id)
- const char \* [PCI::get\\_device\\_name](#) (uint16\_t vendor\_id, uint16\_t device\_id)
- const char \* [PCI::get\\_subclass\\_name](#) (uint8\_t class\_code, uint8\_t subclass\_code)
- const char \* [PCI::get\\_prog\\_if\\_name](#) (uint8\_t class\_code, uint8\_t subclass\_code, uint8\_t prog\_if)

## Variables

- const char \* [PCI::device\\_classes](#) []

## 6.65 pci.h

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002
00003 #include <stdint.h>
00004
00005 #include "acpi.h"
00006 #include "c_str.h"
00007 #include "basic_renderer.h"
00008 #include "paging/page_table_manager.h"
00009
00010 namespace PCI
00011 {
00012     struct PCIDeviceHeader
00013     {
00014         uint16_t vendor_id;
00015         uint16_t device_id;
00016         uint16_t command;
00017         uint16_t status;
00018         uint8_t revision_id;
00019         uint8_t prog_if; // program interface
00020         uint8_t subclass;
00021         uint8_t class_code;
00022         uint8_t cache_line_size;
00023         uint8_t latency_timer;
00024         uint8_t header_type;
00025         uint8_t bist;
00026     };
00027
00028     void enumerate_pci(ACPI::MCFGHeader* mcfg);
00029
00030     extern const char* device_classes[];
00031
00032     const char* get_vendor_name(uint16_t vendor_id);
00033     const char* get_device_name(uint16_t vendor_id, uint16_t device_id);
00034     const char* get_subclass_name(uint8_t class_code, uint8_t subclass_code);
00035     const char* get_prog_if_name(uint8_t class_code, uint8_t subclass_code, uint8_t prog_if);
00036 }

```

## 6.66 pci\_descriptors.cpp File Reference

```
#include <stdint.h>
#include "c_str.h"
```

### Namespaces

- namespace [PCI](#)

### Functions

- const char \* [PCI::get\\_vendor\\_name](#) (uint16\_t vendor\_id)
- const char \* [PCI::get\\_device\\_name](#) (uint16\_t vendor\_id, uint16\_t device\_id)
- const char \* [PCI::mass\\_storage\\_controller\\_subclass\\_name](#) (uint8\_t subclass\_code)
- const char \* [PCI::get\\_prog\\_if\\_name](#) (uint8\_t class\_code, uint8\_t subclass\_code, uint8\_t prog\_if)
- const char \* [PCI::get\\_subclass\\_name](#) (uint8\_t class\_code, uint8\_t subclass\_code)

## 6.67 pit.cpp File Reference

```
#include "pit.h"
```

### Namespaces

- namespace [PIT](#)

### Functions

- void [PIT::sleepd](#) (double seconds)
- void [PIT::sleep](#) (uint64\_t ms)
- void [PIT::set\\_divisor](#) (uint16\_t divisor)
- uint64\_t [PIT::get\\_freq](#) ()
- void [PIT::set\\_freq](#) (uint64\_t freq)
- void [PIT::tick](#) ()

### Variables

- double [PIT::time\\_since\\_boot](#) = 0
- uint16\_t [PIT::divisor](#) = 65535

## 6.68 pit.h File Reference

```
#include <stdint.h>
#include "../io.h"
```



## Namespaces

- namespace [PIT](#)

## Functions

- void [PIT::sleepd](#) (double seconds)
- void [PIT::sleep](#) (uint64\_t ms)
- void [PIT::set\\_divisor](#) (uint16\_t divisor)
- uint64\_t [PIT::get\\_freq](#) ()
- void [PIT::set\\_freq](#) (uint64\_t freq)
- void [PIT::tick](#) ()

## Variables

- const uint64\_t [PIT::base\\_freq](#) = 1193182

## 6.69 pit.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <stdint.h>
00004
00005 #include "../io.h"
00006
00007 namespace PIT
00008 {
00009     extern double time_since_boot;
00010     const uint64_t base_freq = 1193182;
00011
00012     void sleepd(double seconds);
00013     void sleep(uint64_t ms);
00014
00015     void set_divisor(uint16_t divisor);
00016     uint64_t get_freq();
00017     void set_freq(uint64_t freq);
00018     void tick();
00019 }
```

## 6.70 simple\_font.h File Reference

### Classes

- struct [PSF1\\_HEADER](#)
- struct [PSF1\\_FONT](#)

## 6.71 simple\_font.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 struct PSF1_HEADER
00004 {
00005     unsigned char magic[2];
00006     unsigned char mode;
00007     unsigned char charsize;
00008 };
00009
00010 struct PSF1_FONT
00011 {
00012     PSF1_HEADER* psf1_header;
00013     void* glyph_buffer;
00014 };
```

