eta-os

Generated by Doxygen 1.9.6

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

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', ';', '\", '`', 0, '\\', 'y', '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_adress
- 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_adress

```
void* Framebuffer::base_adress
```

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

- GDTEntry null
- GDTEntry kernel_code
- GDTEntry kernel_data
- GDTEntry user_null
- GDTEntry user_code
- GDTEntry user_data

5.7.1 Member Data Documentation

5.7.1.1 kernel_code

```
GDTEntry GDT::kernel_code
```

5.7.1.2 kernel_data

```
GDTEntry GDT::kernel_data
```

5.7.1.3 null

```
GDTEntry GDT::null
```

5.7.1.4 user_code

```
GDTEntry GDT::user_code
```

5.7.1.5 user_data

```
GDTEntry GDT::user_data
```

5.7.1.6 user_null

```
GDTEntry 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 GDTEntry::base0
```

5.9.1.3 base1

```
uint8_t GDTEntry::base1
```

5.9.1.4 base2

```
uint8_t GDTEntry::base2
```

5.9.1.5 limit0

```
uint16_t GDTEntry::limit0
```

5.9.1.6 limit1_flags

```
uint8_t GDTEntry::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_adrr

```
uint64_t ACPI::RSDP2::xsdt_adrr
```

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.

```
#pragma once

#include <stdint.h>

namespace ACPI
{
    struct RSDP2
    {
        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];
    }__attribute__((packed));

    struct SDTHeader
    {
        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;
    }__attribute__((packed));

    struct MCFGHeader // contains information about the pci bus
    {
        SDTHeader header;
        uint64_t reserved;
    }__attribute__((packed));

    struct DeviceConfig
    {
        uint64_t base_addr;
        uint16_t pci_seg_group;
        uint8_t start_bus;
        uint8_t end_bus;
        uint32_t reserved;
    }__attribute__((packed));

    void* find_table(SDTHeader* sdt_header, char* signature);
}
```

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.

```
#pragma once

#include <stdint.h>

#include "math_util.h"
#include "framebuffer.h"
#include "simple_font.h"

class BasicRenderer
{
public:
    BasicRenderer(Framebuffer* target_framebuffer, PSF1_FONT* psf1_font, unsigned int color)
    {
        this->target_framebuffer = target_framebuffer;
        this->psf1_font = psf1_font;
        this->color = color;
        this->cursor_position = { 0,0 };
    };

    Point cursor_position;
    Framebuffer* target_framebuffer;
    PSF1_FONT* psf1_font;
    unsigned int color;
    unsigned int clear_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();
};

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.

```
#pragma once

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

class Bitmap
{
public:
    size_t size;
    uint8_t* buffer;
    bool operator[](uint64_t index);
    bool set(uint64_t index, bool value);
};
```

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.

```
#pragma once

#include <stdint.h>

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);
```

6.13 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.

```
#pragma once

#include <stdint.h>

struct EFI_MEMORY_DESCRIPTOR
{
    uint32_t type;
    void* phys_addr;
    void* virt_addr;
    uint64_t num_pages;
    uint64_t attrib;
};

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.

```
#pragma once

#include <stddef.h>

struct Framebuffer
{
    void* base_adress;
    size_t buffer_size;
    unsigned int width;
    unsigned int height;
    unsigned int pixels_per_scanline;
};
```

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 GDTEntry
- 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
- GDTEntry null
- GDTEntry kernel_code
- GDTEntry kernel_data
- GDTEntry user_null
- GDTEntry user_code
- GDTEntry 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

```
GDTEntry user_code
```

6.19.2.14 user_data

```
GDTEntry user_data
```

6.19.2.15 user_null

```
GDTEntry user_null
```

6.20 gdt.h

Go to the documentation of this file.

```
#pragma once

#include <stdint.h>

struct GDTDescriptor
{
    uint16_t size;
    uint64_t offset;
}__attribute__((packed));

struct GDTEntry
{
    uint16_t limit0;
    uint16_t base0;
    uint8_t base1;
    uint8_t access_byte;
    uint8_t limit1_flags;
    uint8_t base2;
}__attribute__((packed));

struct GDT
{
    GDTEntry null;
    GDTEntry kernel_code;
    GDTEntry kernel_data;
    GDTEntry user_null;
    GDTEntry user_code;
    GDTEntry user_data;
}__attribute__((packed))
__attribute__((aligned(4096)));

extern GDT default_gdt;

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.

```
#pragma once

#include <stdint.h>

namespace QWERTZKeyboard
{
    #define LEFT_SHIFT 0x2a
    #define RIGHT_SHIFT 0x36
    #define ENTER 0x1c
    #define BACKSPACE 0x0e
    #define SPACE 0x39

    extern const char ascii_table[];
    char translate(uint8_t scancode, bool uppercase);
}
```

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.

```
#pragma once

#include <stdint.h>

#include "kb_scancode_trans.h"
#include "../basic_renderer.h"

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.

```
#pragma once

#include <stdint.h>

#define IDT_TA_INTERRUPTGATE 0b10001110
#define IDT_TA_CALLGATE      0b10001100
#define IDT_TA_TRAPGATE      0b10001111

struct IDTDescEntry
{
    uint16_t offset0;
    uint16_t selector;
    uint8_t ist;          // interrupt stack table
    uint8_t type_attr;
    uint16_t offset1;
    uint32_t offset2;
    uint32_t ignore;

    void set_offset(uint64_t offset);
    uint64_t get_offset();
};

struct IDTR
{
    uint16_t limit;
    uint64_t offset;
}__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__()

```
__attribute__ (
              (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__()

```
__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.

```
#pragma once

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

#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

struct InterruptFrame;
__attribute__((interrupt)) void page_fault_handler(InterruptFrame* frame);
__attribute__((interrupt)) void double_fault_handler(InterruptFrame* frame);
__attribute__((interrupt)) void gp_fault_handler(InterruptFrame* frame);
__attribute__((interrupt)) void keyboard_int_handler(InterruptFrame* frame);
__attribute__((interrupt)) void pit_int_handler(InterruptFrame* frame);

void remap_pic();
void pic_end_master();
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.

```
#pragma once

#include <stdint.h>

void outb(uint16_t port, uint8_t value);
uint8_t inb(uint16_t port);
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.

```
#pragma once

#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"

struct BOOT_INFO
{
    Framebuffer* framebuffer;
    PSF1_FONT* psf1_font;
    void* mem_map;
    uint64_t mem_map_size;
    uint64_t mem_map_descriptor_size;
    ACPI::RSDP2* rsdp;
};

struct KernelInfo
{
    PageTableManager* page_table_mgr;
};

KernelInfo init_kernel(BOOT_INFO* boot_info);

extern uint64_t _kernel_start;
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.

```
#pragma once

struct Point
{
    long x;
    long y;
};
```

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.

```
#pragma once

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

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);
```

6.45 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.

```
#pragma once

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

#include "../paging/page_table_manager.h"
#include "../paging/page_frame_allocator.h"

struct HeapSegHeader
{
    size_t length;
    HeapSegHeader* next;
    HeapSegHeader* last;
    bool free;

    void combine_forward();
    void combine_backward();
    HeapSegHeader* split(size_t split_length);
};

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);
```

6.48 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.

```
#pragma once

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

class PageFrameAllocator
{
public:
    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();
    Bitmap page_bitmap;

private:
    void init_bitmap(size_t bitmap_size, void* buffer_addr);
    void reserve_page(void* addr);
    void reserve_pages(void* addr, uint64_t page_count);
    void unreserve_page(void* addr);
    void unreserve_pages(void* addr, uint64_t page_count);
};

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.

```
#pragma once

#include <stdint.h>

class PageMapIndexer
{
public:
    PageMapIndexer(uint64_t virt_addr);
    uint64_t pdp_i; // Page Directory Pointer Index
    uint64_t pd_i;  // Page Directory Index
    uint64_t pt_i;  // Page Table Index
    uint64_t p_i;   // Page Index
};
```

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.

```
#pragma once

#include "paging.h"

class PageTableManager
{
public:
    PageTableManager(PageTable* pml4_addr);
    PageTable* pml4_addr;
    void map_mem(void* virt_mem, void* phys_mem);
};

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.

```
#pragma once

#include <stdint.h>

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 // EXPERIMENTAL !! ONLY ON MODERN SYSTEMS
};

struct PageDirEntry
{
    uint64_t value;
    void set_flag(PT_FLAG flag, bool enabled);
    bool get_flag(PT_FLAG flag);
    void set_address(uint64_t addr);
    uint64_t get_address();
};

struct PageTable
{
    PageDirEntry entries[512];

}__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.

```
#pragma once

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.

```
#pragma once

#include <stdint.h>

#include "acpi.h"
#include "c_str.h"
#include "basic_renderer.h"
#include "paging/page_table_manager.h"

namespace PCI
{
    struct PCIDeviceHeader
    {
        uint16_t vendor_id;
        uint16_t device_id;
        uint16_t command;
        uint16_t status;
        uint8_t revision_id;
        uint8_t prog_if; // program interface
        uint8_t subclass;
        uint8_t class_code;
        uint8_t cache_line_size;
        uint8_t latency_timer;
        uint8_t header_type;
        uint8_t bist;
    };

    void enumerate_pci(ACPI::MCFGHeader* mcfg);

    extern const char* device_classes[];

    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);
}
```

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.

```
#pragma once

#include <stdint.h>

#include "../../io.h"

namespace PIT
{
    extern double time_since_boot;
    const uint64_t base_freq = 1193182;

    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();
}
```

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.

```
#pragma once

struct PSF1_HEADER
{
    unsigned char magic[2];
    unsigned char mode;
    unsigned char charsize;
};

struct PSF1_FONT
{
    PSF1_HEADER* psf1_header;
    void* glyph_buffer;
};
```