

HanOS

0.1.1.220310

THINK BIG, DO SMALL

1 Todo List	1
2 Data Structure Index	3
2.1 Data Structures	3
3 File Index	5
3.1 File List	5
4 Data Structure Documentation	7
4.1 acpi_gas_t Struct Reference	7
4.2 acpi_sdt_hdr_t Struct Reference	7
4.3 acpi_sdt_t Struct Reference	8
4.4 addrspace_t Struct Reference	8
4.5 cmos_rtc_t Struct Reference	8
4.6 cpu_t Struct Reference	9
4.7 cpuid_feature_t Struct Reference	9
4.8 gdt_entry_t Struct Reference	10
4.9 gdt_register_t Struct Reference	10
4.10 gdt_table_t Struct Reference	10
4.11 hpet_sdt_t Struct Reference	11
4.12 hpet_t Struct Reference	12
4.13 hpet_timer_t Struct Reference	12
4.14 idt_entry_t Struct Reference	13
4.15 idt_register_t Struct Reference	13
4.16 klog_info_t Struct Reference	13
4.17 lock_t Struct Reference	14
4.18 madt_record_hdr_t Struct Reference	14
4.19 madt_record_ioapic_t Struct Reference	14
4.20 madt_record_iso_t Struct Reference	15
4.21 madt_record_lapic_t Struct Reference	16
4.22 madt_record_nmi_t Struct Reference	16
4.23 madt_t Struct Reference	17
4.24 mem_info_t Struct Reference	17
4.25 metadata_t Struct Reference	18
4.26 rsdp_t Struct Reference	18
4.27 smp_info_t Struct Reference	18
4.28 symbol_t Struct Reference	19
4.29 sys_seg_desc_t Struct Reference	19
4.30 task_list_t Struct Reference	20
4.31 task_regs_t Struct Reference	20
4.32 task_t Struct Reference	21
4.33 timeval_t Struct Reference	21
4.34 timezone_t Struct Reference	22

4.35 tm_t Struct Reference	22
4.36 tss_t Struct Reference	22
5 File Documentation	23
5.1 kernel/core/acpi.c File Reference	23
5.1.1 Detailed Description	23
5.2 kernel/core/acpi.h File Reference	24
5.2.1 Detailed Description	24
5.3 kernel/core/apic.c File Reference	25
5.3.1 Detailed Description	25
5.4 kernel/core/apic.h File Reference	26
5.4.1 Detailed Description	27
5.5 kernel/core/cmos.c File Reference	27
5.5.1 Detailed Description	28
5.6 kernel/core/cmos.h File Reference	28
5.6.1 Detailed Description	29
5.7 kernel/core/cpu.c File Reference	30
5.7.1 Detailed Description	30
5.8 kernel/core/cpu.h File Reference	30
5.8.1 Detailed Description	31
5.8.2 Macro Definition Documentation	32
5.8.2.1 read_cr	32
5.8.2.2 write_cr	32
5.9 kernel/core/gdt.c File Reference	32
5.9.1 Detailed Description	33
5.10 kernel/core/gdt.h File Reference	33
5.10.1 Detailed Description	34
5.11 kernel/core/hpet.c File Reference	34
5.11.1 Detailed Description	35
5.12 kernel/core/hpet.h File Reference	35
5.12.1 Detailed Description	36
5.13 kernel/core/idt.c File Reference	36
5.13.1 Detailed Description	37
5.14 kernel/core/idt.h File Reference	38
5.14.1 Detailed Description	39
5.15 kernel/core/isr.c File Reference	39
5.15.1 Detailed Description	39
5.16 kernel/core/isr_base.h File Reference	40
5.16.1 Detailed Description	41
5.17 kernel/core/madt.c File Reference	41
5.17.1 Detailed Description	42
5.18 kernel/core/madt.h File Reference	43

5.18.1 Detailed Description	44
5.19 kernel/core/mm.c File Reference	44
5.19.1 Detailed Description	45
5.20 kernel/core/mm.h File Reference	45
5.20.1 Detailed Description	46
5.21 kernel/core/panic.h File Reference	47
5.21.1 Detailed Description	47
5.21.2 Macro Definition Documentation	48
5.21.2.1 kpanic	48
5.22 kernel/core/pit.c File Reference	48
5.22.1 Detailed Description	48
5.23 kernel/core/pit.h File Reference	49
5.23.1 Detailed Description	49
5.24 kernel/core/smp.c File Reference	50
5.24.1 Detailed Description	51
5.25 kernel/core/smp.h File Reference	51
5.25.1 Detailed Description	52
5.26 kernel/core/timer.c File Reference	52
5.26.1 Detailed Description	53
5.27 kernel/core/timer.h File Reference	54
5.27.1 Detailed Description	55
5.28 kernel/kmain.c File Reference	55
5.28.1 Detailed Description	56
5.29 kernel/lib/klog.c File Reference	56
5.29.1 Detailed Description	57
5.30 kernel/lib/klog.h File Reference	57
5.30.1 Detailed Description	59
5.31 kernel/lib/kmalloc.c File Reference	59
5.31.1 Detailed Description	59
5.32 kernel/lib/kmalloc.h File Reference	60
5.32.1 Detailed Description	60
5.33 kernel/lib/lock.h File Reference	60
5.33.1 Detailed Description	61
5.33.2 Macro Definition Documentation	61
5.33.2.1 lock_lock	62
5.33.2.2 lock_release	62
5.34 kernel/lib/memutils.c File Reference	62
5.34.1 Detailed Description	63
5.35 kernel/lib/memutils.h File Reference	63
5.35.1 Detailed Description	64
5.36 kernel/lib/string.c File Reference	64
5.36.1 Detailed Description	64

5.37 kernel/lib/string.h File Reference	65
5.37.1 Detailed Description	65
5.38 kernel/lib/time.c File Reference	65
5.38.1 Detailed Description	66
5.39 kernel/lib/time.h File Reference	66
5.39.1 Detailed Description	67
5.40 kernel/proc/sched.c File Reference	67
5.40.1 Detailed Description	68
5.41 kernel/proc/sched.h File Reference	68
5.41.1 Detailed Description	69
5.42 kernel/proc/task.c File Reference	70
5.42.1 Detailed Description	70
5.43 kernel/proc/task.h File Reference	71
5.43.1 Detailed Description	72
5.44 kernel/proc/tlist.c File Reference	73
5.44.1 Detailed Description	73
5.45 kernel/proc/tlist.h File Reference	74
5.45.1 Detailed Description	75
5.46 kernel/symbols.h File Reference	75
5.46.1 Detailed Description	76
Index	77

Chapter 1

Todo List

File [kmalloc.c](#)

Memory allocation should be improved for better efficiency.

File [pit.c](#)

Rewrite pit_wait() function

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

acpi_gas_t	7
acpi_sdt_hdr_t	7
acpi_sdt_t	8
addrspace_t	8
cmos_rtc_t	8
cpu_t	9
cpuid_feature_t	9
gdt_entry_t	10
gdt_register_t	10
gdt_table_t	10
hpet_sdt_t	11
hpet_t	12
hpet_timer_t	12
idt_entry_t	13
idt_register_t	13
klog_info_t	13
lock_t	14
madt_record_hdr_t	14
madt_record_ioapic_t	14
madt_record_iso_t	15
madt_record_lapic_t	16
madt_record_nmi_t	16
madt_t	17
mem_info_t	17
metadata_t	18
rsdp_t	18
smp_info_t	18
symbol_t	19
sys_seg_desc_t	19
task_list_t	20
task_regs_t	20
task_t	21
timeval_t	21
timezone_t	22
tm_t	22
tss_t	22

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

kernel/ kmain.c	55
Entry function of HanOS kernel	
kernel/ symbols.h	75
Definition of symbol related data structures	
kernel/core/ acpi.c	23
Implementation of ACPI (Advanced Configuration and Power Management Interface) functions	
kernel/core/ acpi.h	24
Definition of ACPI (Advanced Configuration and Power Management Interface) related data structures	
kernel/core/ apic.c	25
Implementation of APIC (Advanced Programmable Interrupt Controller) functions	
kernel/core/ apic.h	26
Definition of APIC (Advanced Programmable Interrupt Controller) related data structures	
kernel/core/ cmos.c	27
Implementation of CMOS related functions	
kernel/core/ cmos.h	28
Definition of CMOS related data structures	
kernel/core/ cpu.c	30
Implementation of CPU related functions	
kernel/core/ cpu.h	30
Definition of CPU related data structures and macros	
kernel/core/ gdt.c	32
Implementation of GDT related functions	
kernel/core/ gdt.h	33
Definition of GDT related data structures	
kernel/core/ hpet.c	34
Implementation of HPET (High Precision Event Timer) functions	
kernel/core/ hpet.h	35
Definition of HPET (High Precision Event Timer) related data structures	
kernel/core/ idt.c	36
Implementation of idt related functions	
kernel/core/ idt.h	38
Definition of IDT related data structures	
kernel/core/ isr.c	39
Implementation of ISR related functions	

kernel/core/isr_base.h	
Definition of ISR related data structures	40
kernel/core/madt.c	
Implementation of ACPI MADT (Multiple APIC Description Table) functions	41
kernel/core/madt.h	
Definition of ACPI MADT (Multiple APIC Description Table) related data structures	43
kernel/core/mm.c	
Implementation of memory management functions	44
kernel/core/mm.h	
Definition of memory management related data structures	45
kernel/core/panic.h	
Implementation of panic related functions	47
kernel/core/pit.c	
Implementation of PIT related functions	48
kernel/core/pit.h	
Definition of PIT related data structures	49
kernel/core/smp.c	
Implementation of SMP related functions	50
kernel/core/smp.h	
Definition of SMP related data structures	51
kernel/core/timer.c	
Implementation of APIC timer related functions	52
kernel/core/timer.h	
Definition of APIC timer related macros, data structures, functions	54
kernel/lib/klog.c	
Implementation of kernel log related functions	56
kernel/lib/klog.h	
Definition of kernel log related functions	57
kernel/lib/kmalloc.c	
Implementation of memory allocation related functions	59
kernel/lib/kmalloc.h	
Definition of memory allocation related functions	60
kernel/lib/lock.h	
Definition of lock related data structures and functions	60
kernel/lib/memutils.c	
Implementation of memory operation related functions	62
kernel/lib/memutils.h	
Definition of memory operation related functions	63
kernel/lib/string.c	
Implementation of string operation related functions	64
kernel/lib/string.h	
Definition of string operation related functions	65
kernel/lib/time.c	
Implementation of time related functions	65
kernel/lib/time.h	
Definition of time related data structures and functions	66
kernel/proc/sched.c	
Implementation of scheduling related functions	67
kernel/proc/sched.h	
Definition of scheduling related functions	68
kernel/proc/task.c	
Implementation of task related functions	70
kernel/proc/task.h	
Definition of task related functions	71
kernel/proc/tlist.c	
Implementation of task list related functions	73
kernel/proc/tlist.h	
Definition of task list related functions	74

Chapter 4

Data Structure Documentation

4.1 `acpi_gas_t` Struct Reference

Data Fields

- `uint8_t addr_space_id`
- `uint8_t reg_bit_width`
- `uint8_t reg_bit_offset`
- `uint8_t reserved`
- `uint64_t address`

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

- `kernel/core/acpi.h`

4.2 `acpi_sdt_hdr_t` Struct Reference

Data Fields

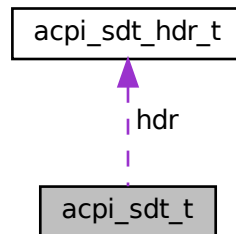
- `char sign [4]`
- `uint32_t length`
- `uint8_t rev`
- `uint8_t checksum`
- `char oem_id [6]`
- `char oem_table_id [8]`
- `uint32_t oem_rev`
- `uint32_t creator_id`
- `uint32_t creator_rev`

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

- `kernel/core/acpi.h`

4.3 acpi_sdt_t Struct Reference

Collaboration diagram for `acpi_sdt_t`:



Data Fields

- [acpi_sdt_hdr_t](#) `hdr`
- `uint8_t data []`

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

- [kernel/core/acpi.h](#)

4.4 addrspace_t Struct Reference

Data Fields

- `uint64_t * PML4`

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

- [kernel/core/mm.h](#)

4.5 cmos_rtc_t Struct Reference

Data Fields

- `uint8_t seconds`
- `uint8_t minutes`
- `uint8_t hours`
- `uint8_t weekdays`
- `uint8_t day`
- `uint8_t month`
- `uint16_t year`
- `uint8_t century`

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

- [kernel/core/cmos.h](#)

4.6 `cpu_t` Struct Reference

Collaboration diagram for `cpu_t`:



Data Fields

- `tss_t` **tss**
- `uint16_t` **cpu_id**
- `uint16_t` **lapic_id**
- `bool` **is_bsp**
- `uint8_t` **reserved** [3]

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

- `kernel/core/smp.h`

4.7 `cpuid_feature_t` Struct Reference

Public Types

- `enum` { **CPUID_REG_EAX**, **CPUID_REG_EBX**, **CPUID_REG_ECX**, **CPUID_REG_EDX** }

Data Fields

- `uint32_t` **func**
- `uint32_t` **param**
- `enum` `cpuid_feature_t`:: { ... } **reg**
- `uint32_t` **mask**

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

- `kernel/core/cpu.h`

4.8 gdt_entry_t Struct Reference

Data Fields

- uint16_t **limit**
- uint16_t **base_low**
- uint8_t **base_mid**
- uint8_t **access**
- uint8_t **granularity**
- uint8_t **base_high**

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

- [kernel/core/gdt.h](#)

4.9 gdt_register_t Struct Reference

Data Fields

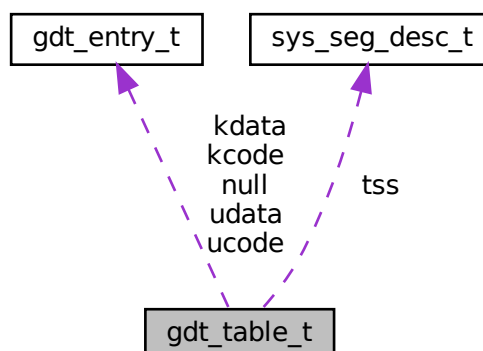
- uint16_t **size**
- uint64_t **offset**

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

- [kernel/core/gdt.h](#)

4.10 gdt_table_t Struct Reference

Collaboration diagram for gdt_table_t:



Data Fields

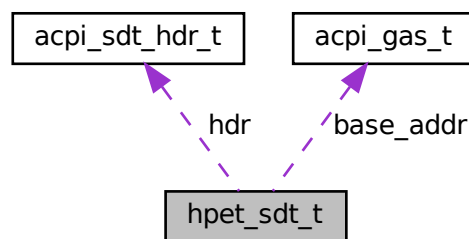
- [gdt_entry_t](#) **null**
- [gdt_entry_t](#) **kcode**
- [gdt_entry_t](#) **kdata**
- [gdt_entry_t](#) **ucode**
- [gdt_entry_t](#) **udata**
- [sys_seg_desc_t](#) **tss**

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

- [kernel/core/gdt.h](#)

4.11 hpet_sdt_t Struct Reference

Collaboration diagram for hpet_sdt_t:



Data Fields

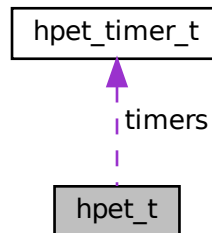
- [acpi_sdt_hdr_t](#) **hdr**
- `uint8_t` **hardware_rev_id**
- `uint8_t` **comparator_count**: 5
- `uint8_t` **counter_size**: 1
- `uint8_t` **reserved**: 1
- `uint8_t` **legacy_replace**: 1
- `uint16_t` **pci_vendor_id**
- [acpi_gas_t](#) **base_addr**
- `uint8_t` **hpet_number**
- `uint16_t` **minimum_tick**
- `uint8_t` **page_protection**

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

- [kernel/core/hpet.h](#)

4.12 hpet_t Struct Reference

Collaboration diagram for hpet_t:



Data Fields

- volatile uint64_t **general_capabilities**
- volatile uint64_t **unused0**
- volatile uint64_t **general_configuration**
- volatile uint64_t **unused1**
- volatile uint64_t **general_int_status**
- volatile uint64_t **unused2**
- volatile uint64_t **unused3** [2][12]
- volatile uint64_t **main_counter_value**
- volatile uint64_t **unused4**
- [hpet_timer_t](#) **timers** []

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

- [kernel/core/hpet.h](#)

4.13 hpet_timer_t Struct Reference

Data Fields

- volatile uint64_t **config_and_capabilities**
- volatile uint64_t **comparator_value**
- volatile uint64_t **fsb_interrupt_route**
- volatile uint64_t **unused**

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

- [kernel/core/hpet.h](#)

4.14 idt_entry_t Struct Reference

Data Fields

- uint16_t **offset_1**
- uint16_t **selector**
- uint8_t **ist**
- uint8_t **type_attributes**
- uint16_t **offset_2**
- uint32_t **offset_3**
- uint32_t **zero**

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

- kernel/core/[idt.h](#)

4.15 idt_register_t Struct Reference

Data Fields

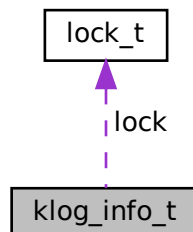
- uint16_t **size**
- uint64_t **offset**

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

- kernel/core/[idt.h](#)

4.16 klog_info_t Struct Reference

Collaboration diagram for klog_info_t:



Data Fields

- `uint8_t buff` [KLOG_BUFFER_SIZE]
- `int start`
- `int end`
- `term_info_t * term`
- `lock_t lock`

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

- `kernel/lib/klog.h`

4.17 lock_t Struct Reference

Data Fields

- `int lock`
- `uint64_t rflags`

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

- `kernel/lib/lock.h`

4.18 madt_record_hdr_t Struct Reference

Data Fields

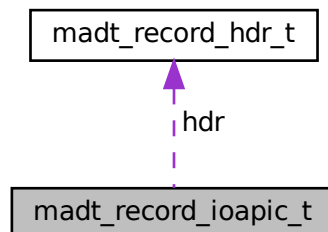
- `uint8_t type`
- `uint8_t len`

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

- `kernel/core/madt.h`

4.19 madt_record_ioapic_t Struct Reference

Collaboration diagram for `madt_record_ioapic_t`:



Data Fields

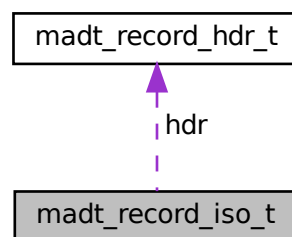
- [madt_record_hdr_t](#) **hdr**
- `uint8_t` **id**
- `uint8_t` **reserved**
- `uint32_t` **addr**
- `uint32_t` **gsi_base**

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

- `kernel/core/madt.h`

4.20 madt_record_iso_t Struct Reference

Collaboration diagram for madt_record_iso_t:



Data Fields

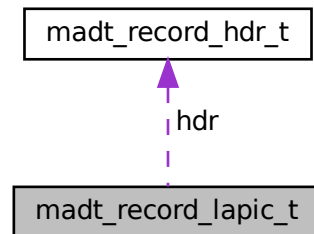
- [madt_record_hdr_t](#) **hdr**
- `uint8_t` **bus_src**
- `uint8_t` **irq_src**
- `uint32_t` **gsi**
- `uint16_t` **flags**

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

- `kernel/core/madt.h`

4.21 madt_record_lapic_t Struct Reference

Collaboration diagram for madt_record_lapic_t:



Data Fields

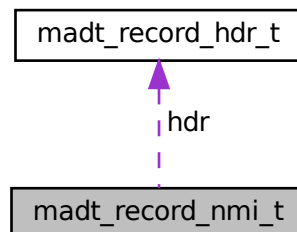
- [madt_record_hdr_t](#) `hdr`
- `uint8_t` `proc_id`
- `uint8_t` `apic_id`
- `uint32_t` `flags`

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

- `kernel/core/madt.h`

4.22 madt_record_nmi_t Struct Reference

Collaboration diagram for madt_record_nmi_t:



Data Fields

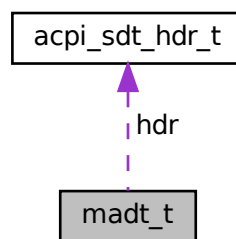
- [madt_record_hdr_t](#) **hdr**
- `uint8_t` **proc_id**
- `uint16_t` **flags**
- `uint8_t` **lint**

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

- `kernel/core/madt.h`

4.23 madt_t Struct Reference

Collaboration diagram for madt_t:



Data Fields

- [acpi_sdt_hdr_t](#) **hdr**
- `uint32_t` **lapic_addr**
- `uint32_t` **flags**
- `uint8_t` **records** []

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

- `kernel/core/madt.h`

4.24 mem_info_t Struct Reference

Data Fields

- `uint64_t` **phys_limit**
- `uint64_t` **total_size**
- `uint64_t` **free_size**
- `uint8_t *` **bitmap**

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

- `kernel/core/mm.h`

4.25 metadata_t Struct Reference

Data Fields

- size_t **numpages**
- size_t **size**

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

- kernel/lib/[kmallocc.c](#)

4.26 rsdp_t Struct Reference

Data Fields

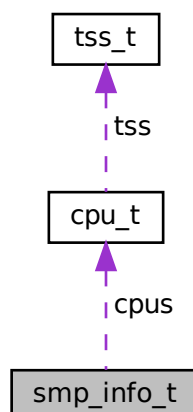
- char **sign** [8]
- uint8_t **chksum**
- char **oem_id** [6]
- uint8_t **revision**
- uint32_t **rsdt_addr**
- uint32_t **length**
- uint64_t **xsdt_addr**
- uint8_t **chksum_ext**
- uint8_t **reserved** [3]

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

- kernel/core/[acpi.h](#)

4.27 smp_info_t Struct Reference

Collaboration diagram for smp_info_t:



Data Fields

- `cpu_t cpus` [CPU_MAX]
- `uint16_t num_cpus`

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

- `kernel/core/smp.h`

4.28 `symbol_t` Struct Reference

Data Fields

- `uint64_t addr`
- `char * name`

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

- `kernel/symbols.h`

4.29 `sys_seg_desc_t` Struct Reference

Data Fields

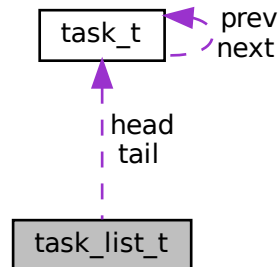
- `uint32_t segment_limit_low`: 16
- `uint32_t segment_base_low`: 16
- `uint32_t segment_base_mid`: 8
- `uint32_t segment_type`: 4
- `uint32_t zero_1`: 1
- `uint32_t segment_dpl`: 2
- `uint32_t segment_present`: 1
- `uint32_t segment_limit_high`: 4
- `uint32_t segment_avail`: 1
- `uint32_t zero_2`: 2
- `uint32_t segment_gran`: 1
- `uint32_t segment_base_mid2`: 8
- `uint32_t segment_base_high`: 32
- `uint32_t reserved_1`: 8
- `uint32_t zero_3`: 5
- `uint32_t reserved_2`: 19

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

- `kernel/core/gdt.h`

4.30 task_list_t Struct Reference

Collaboration diagram for task_list_t:



Data Fields

- [task_t](#) * head
- [task_t](#) * tail

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

- [kernel/proc/tlist.h](#)

4.31 task_regs_t Struct Reference

Data Fields

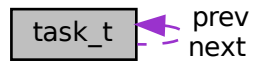
- uint64_t r15
- uint64_t r14
- uint64_t r13
- uint64_t r12
- uint64_t r11
- uint64_t r10
- uint64_t r9
- uint64_t r8
- uint64_t rbp
- uint64_t rdi
- uint64_t rsi
- uint64_t rdx
- uint64_t rcx
- uint64_t rbx
- uint64_t rax
- uint64_t rip
- uint64_t cs
- uint64_t rflags
- uint64_t rsp
- uint64_t ss

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

- [kernel/proc/task.h](#)

4.32 task_t Struct Reference

Collaboration diagram for task_t:



Data Fields

- void * **kstack_top**
- void * **kstack_limit**
- void * **ustack_top**
- void * **ustack_limit**
- task_id_t **tid**
- task_priority_t **priority**
- uint64_t **last_tick**
- task_status_t **status**
- task_mode_t **mode**
- struct [task_t](#) * **next**
- struct [task_t](#) * **prev**

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

- kernel/proc/[task.h](#)

4.33 timeval_t Struct Reference

Data Fields

- uint64_t **sec**
- uint64_t **usec**

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

- kernel/lib/[time.h](#)

4.34 `timezone_t` Struct Reference

Data Fields

- int **minuteswest**
- int **dstime**

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

- kernel/lib/[time.h](#)

4.35 `tm_t` Struct Reference

Data Fields

- int **sec**
- int **min**
- int **hour**
- int **mday**
- int **mon**
- int **year**
- int **wday**
- int **yday**
- int **isdst**

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

- kernel/lib/[time.h](#)

4.36 `tss_t` Struct Reference

Data Fields

- uint32_t **reserved**
- uint64_t **rsp0**
- uint64_t **rsp1**
- uint64_t **rsp2**
- uint32_t **reserved_1**
- uint32_t **reserved_2**
- uint64_t **ist1**
- uint64_t **ist2**
- uint64_t **ist3**
- uint64_t **ist4**
- uint64_t **ist5**
- uint64_t **ist6**
- uint64_t **ist7**
- uint64_t **reserved_3**
- uint16_t **reserved_4**
- uint16_t **io_bitmap_offset**

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

- kernel/core/[smp.h](#)

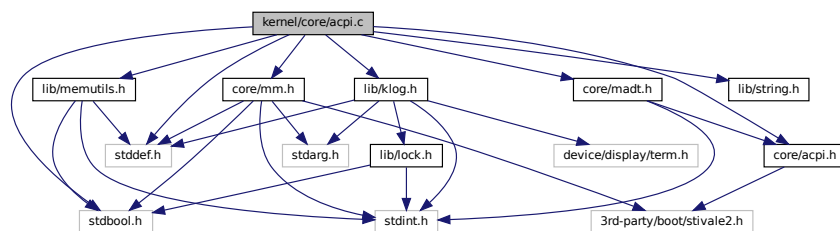
Chapter 5

File Documentation

5.1 kernel/core/acpi.c File Reference

Implementation of ACPI (Advanced Configuration and Power Management Interface) functions.

```
#include <stdbool.h>
#include <stddef.h>
#include <core/acpi.h>
#include <core/madt.h>
#include <core/mm.h>
#include <lib/klog.h>
#include <lib/memutils.h>
#include <lib/string.h>
Include dependency graph for acpi.c:
```



Functions

- `acpi_sdt_t * acpi_get_sdt (const char *sign)`
- `void acpi_init (struct stivale2_struct_tag_rsdp *rsdp_info)`

5.1.1 Detailed Description

Implementation of ACPI (Advanced Configuration and Power Management Interface) functions.

This module includes implementation of RSDT/XSDT initialization and "MADT/HPET" parsing.

Author

JW

Date

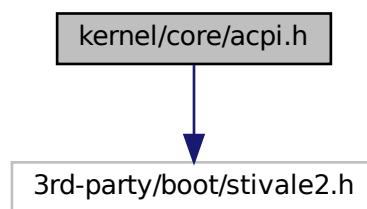
DEC 12, 2021

5.2 kernel/core/acpi.h File Reference

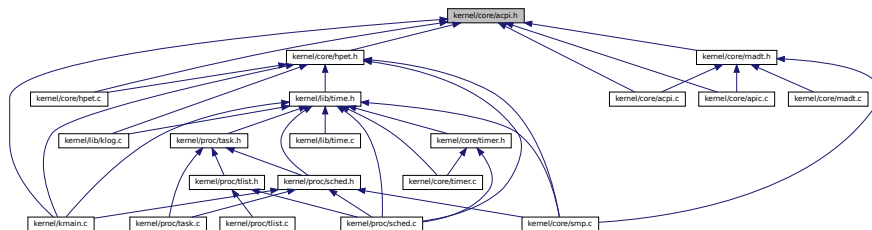
Definition of ACPI (Advanced Configuration and Power Management Interface) related data structures.

```
#include <3rd-party/boot/stivale2.h>
```

Include dependency graph for acpi.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [rsdp_t](#)
- struct [acpi_sdt_hdr_t](#)
- struct [acpi_sdt_t](#)
- struct [acpi_gas_t](#)

Functions

- void [acpi_init](#) (struct stivale2_struct_tag_rsdp *)
- [acpi_sdt_t](#) * [acpi_get_sdt](#) (const char *sign)

5.2.1 Detailed Description

Definition of ACPI (Advanced Configuration and Power Management Interface) related data structures.

ACPI (Advanced Configuration and Power Interface) is a Power Management and configuration standard for the PC, developed by Intel, Microsoft and Toshiba. ACPI allows the operating system to control the amount of power each device is given (allowing it to put certain devices on standby or power-off for example). It is also used to control and/or check thermal zones (temperature sensors, fan speeds, etc), battery levels, PCI IRQ routing, CPUs, NUMA domains and many other things.

There are 2 main parts to ACPI. The first part is the tables used by the OS for configuration during boot (these include things like how many CPUs, APIC details, NUMA memory ranges, etc). The second part is the run time ACPI environment, which consists of AML code (a platform independent OOP language that comes from the BIOS and devices) and the ACPI SMM (System Management Mode) code.

Author

JW

Date

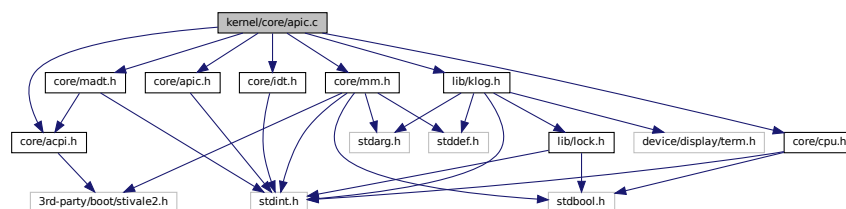
DEC 12, 2021

5.3 kernel/core/apic.c File Reference

Implementation of APIC (Advanced Programmable Interrupt Controller) functions.

```
#include <core/acpi.h>
#include <core/apic.h>
#include <core/madt.h>
#include <core/mm.h>
#include <core/cpu.h>
#include <core/idt.h>
#include <lib/klog.h>
```

Include dependency graph for apic.c:



Functions

- uint32_t **apic_read_reg** (uint16_t offset)
- void **apic_write_reg** (uint16_t offset, uint32_t val)
- void **apic_send_eoi** ()
- void **apic_send_ipi** (uint8_t dest, uint8_t vector, uint32_t mtype)
- void **apic_enable** ()
- void **apic_init** ()

5.3.1 Detailed Description

Implementation of APIC (Advanced Programmable Interrupt Controller) functions.

APIC ("Advanced Programmable Interrupt Controller") is the updated Intel standard for the older PIC. It is used in multiprocessor systems and is an integral part of all recent Intel (and compatible) processors. The APIC is used for sophisticated interrupt redirection, and for sending interrupts between processors.

Ref: <https://wiki.osdev.org/APIC>

Author

JW

Date

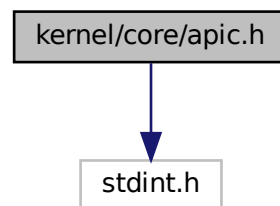
DEC 12, 2021

5.4 kernel/core/apic.h File Reference

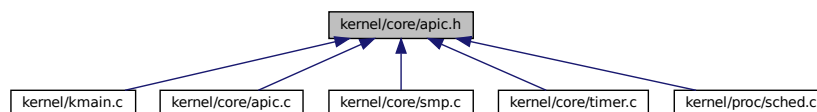
Definition of APIC (Advanced Programmable Interrupt Controller) related data structures.

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

Include dependency graph for apic.h:



This graph shows which files directly or indirectly include this file:



Macros

- `#define APIC_REG_ID 0x20`
- `#define APIC_REG_VERSION 0x30`
- `#define APIC_REG_SPURIOUS_INT 0xF0`
- `#define APIC_REG_EOI 0xB0`
- `#define APIC_REG_ICR_LOW 0x300`
- `#define APIC_REG_ICR_HIGH 0x310`
- `#define APIC_SPURIOUS_VECTOR_NUM 0xFF`
- `#define APIC_FLAG_ENABLE (1 << 8)`
- `#define APIC_IPI_TYPE_INIT 0b101`
- `#define APIC_IPI_TYPE_STARTUP 0b110`

Functions

- void **apic_init** (void)
- void **apic_enable** (void)
- uint32_t **apic_read_reg** (uint16_t offset)
- void **apic_write_reg** (uint16_t offset, uint32_t val)
- void **apic_send_eoi** (void)
- void **apic_send_ipi** (uint8_t dest, uint8_t vector, uint32_t mtype)

5.4.1 Detailed Description

Definition of APIC (Advanced Programmable Interrupt Controller) related data structures.

APIC is used in multiprocessor systems and is an integral part of all recent Intel (and compatible) processors. The APIC is used for sophisticated interrupt redirection, and for sending interrupts between processors.

Author

JW

Date

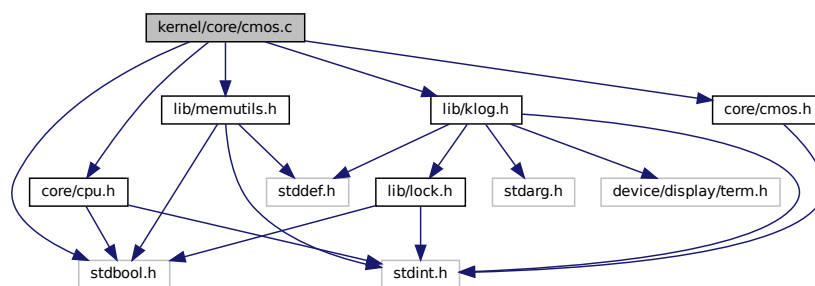
DEC 12, 2021

5.5 kernel/core/cmos.c File Reference

Implementation of CMOS related functions.

```
#include <stdbool.h>
#include <core/cmos.h>
#include <core/cpu.h>
#include <lib/memutils.h>
#include <lib/klog.h>
```

Include dependency graph for cmos.c:



Macros

- `#define CURRENT_YEAR 2022 /* Change this each year! */`

Functions

- uint8_t **get_rtc_register** (uint8_t reg)
 - bool **update_in_progress** ()
 - bool **rtc_values_are_not_equal** (cmos_rtc_t c1, cmos_rtc_t c2)
-

- uint64_t **secs_of_month** (uint64_t months, uint64_t year)
- uint64_t **secs_of_years** (uint64_t years)
- void **cmos_init** ()
- uint64_t **cmos_boot_time** ()
- [cmos_rtc_t](#) **cmos_read_rtc** ()

Variables

- int **century_register** = 0x00

5.5.1 Detailed Description

Implementation of CMOS related functions.

"CMOS" is a tiny bit of very low power static memory that lives on the same chip as the Real-Time Clock (RTC). It was introduced to IBM PC AT in 1984 which used Motorola MC146818A RTC.

Ref: <https://wiki.osdev.org/CMOS>

Author

JW

Date

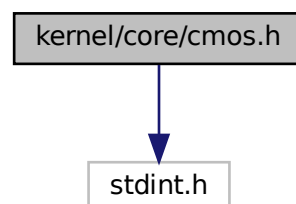
Jan 2, 2022

5.6 kernel/core/cmos.h File Reference

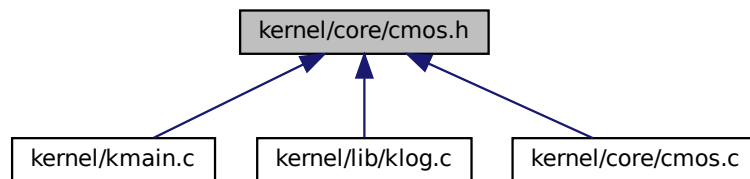
Definition of CMOS related data structures.

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

Include dependency graph for cmos.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [cmos_rtc_t](#)

Macros

- #define **CMOS_COMMAND_PORT** 0x70
- #define **CMOS_DATA_PORT** 0x71
- #define **CMOS_REG_SECONDS** 0x00
- #define **CMOS_REG_MINUTES** 0x02
- #define **CMOS_REG_HOURS** 0x04
- #define **CMOS_REG_WEEKDAYS** 0x06
- #define **CMOS_REG_DAY** 0x07
- #define **CMOS_REG_MONTH** 0x08
- #define **CMOS_REG_YEAR** 0x09
- #define **CMOS_REG_CENTURY** 0x32
- #define **CMOS_REG_STATUS_A** 0x0A
- #define **CMOS_REG_STATUS_B** 0x0B

Functions

- void **cmos_init** ()
- [cmos_rtc_t](#) **cmos_read_rtc** ()
- uint64_t **cmos_boot_time** ()

5.6.1 Detailed Description

Definition of CMOS related data structures.

"CMOS" is a tiny bit of very low power static memory that lives on the same chip as the Real-Time Clock (RTC). It was introduced to IBM PC AT in 1984 which used Motorola MC146818A RTC.

Author

JW

Date

Jan 2, 2022

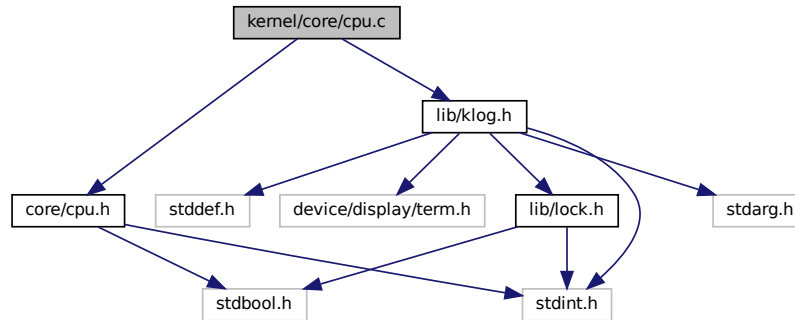
5.7 kernel/core/cpu.c File Reference

Implementation of CPU related functions.

```
#include <core/cpu.h>
```

```
#include <lib/klog.h>
```

Include dependency graph for cpu.c:



Functions

- void **cpuid** (uint32_t func, uint32_t param, uint32_t *eax, uint32_t *ebx, uint32_t *ecx, uint32_t *edx)
- bool **cpuid_check_feature** ([cpuid_feature_t](#) feature)
- void **cpu_init** ()

5.7.1 Detailed Description

Implementation of CPU related functions.

e.g., CPU initialization...

Author

JW

Date

Jan 2, 2022

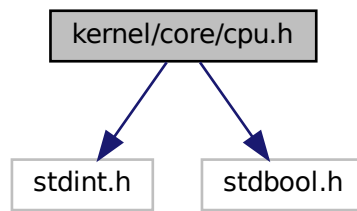
5.8 kernel/core/cpu.h File Reference

Definition of CPU related data structures and macros.

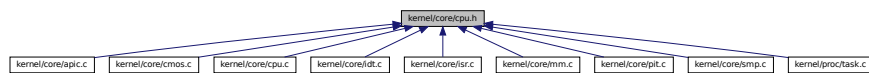
```
#include <stdint.h>
```

```
#include <stdbool.h>
```

Include dependency graph for cpu.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [cpuid_feature_t](#)

Macros

- #define **MSR_PAT** 0x0277
- #define **MSR_GS_BASE** 0xC0000102
- #define **MSR_EFER** 0xC0000080
- #define **MSR_STAR** 0xC0000081
- #define **MSR_LSTAR** 0xC0000082
- #define **MSR_SFMASK** 0xC0000084
- #define **read_cr**(cr, n)
- #define **write_cr**(cr, n)

Functions

- void **cpu_init** ()
- bool **cpuid_check_feature** ([cpuid_feature_t](#) feature)

5.8.1 Detailed Description

Definition of CPU related data structures and macros.

e.g., Read & write control registers, model specific registers and port input & output.

Author

JW

Date

Jan 2, 2022

5.8.2 Macro Definition Documentation

5.8.2.1 read_cr

```
#define read_cr(  
    cr,  
    n )
```

Value:

```
asm volatile("mov %%" cr ", %%rax;" \
"mov %%rax, %0" \
: "=g" (* (n)) \
: "rax");
```

5.8.2.2 write_cr

```
#define write_cr(  
    cr,  
    n )
```

Value:

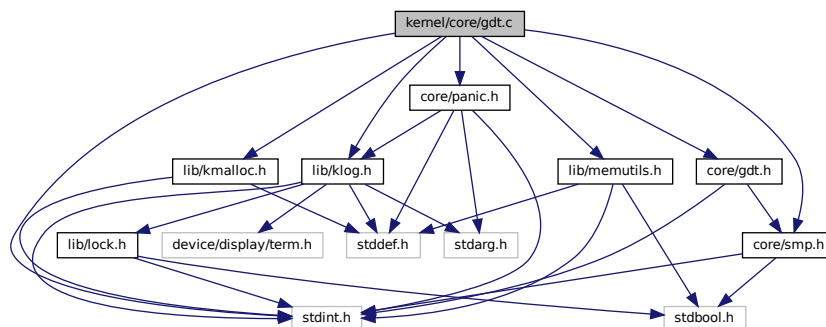
```
asm volatile("mov %0, %%rax;" \
"mov %%rax, %%" cr \
: \
: "g" (n) \
: "rax");
```

5.9 kernel/core/gdt.c File Reference

Implementation of GDT related functions.

```
#include <stdint.h>  
#include <lib/klog.h>  
#include <lib/kmalloc.h>  
#include <lib/memutils.h>  
#include <core/gdt.h>  
#include <core/panic.h>  
#include <core/smp.h>
```

Include dependency graph for gdt.c:



Functions

- void **gdt_init** ([cpu_t](#) *cpuinfo)
- void **gdt_install_tss** ([cpu_t](#) *cpuinfo)

5.9.1 Detailed Description

Implementation of GDT related functions.

The Global Descriptor Table (GDT) contains entries telling the CPU about memory segments.

In HanOS, GDT initialization is very simple. Only memory protection is used. Two ring-0 and two ring-3 segment descriptor are defined. The memory regions are from 0 to 4GB.

Author

JW

Date

Nov 27, 2021

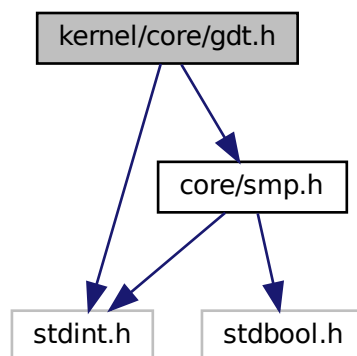
5.10 kernel/core/gdt.h File Reference

Definition of GDT related data structures.

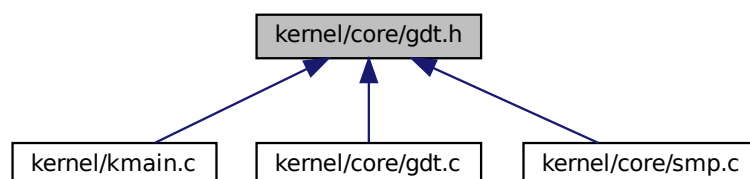
```
#include <stdint.h>
```

```
#include <core/smp.h>
```

Include dependency graph for gdt.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [gdt_entry_t](#)
- struct [sys_seg_desc_t](#)
- struct [gdt_table_t](#)
- struct [gdt_register_t](#)

Functions

- void **gdt_init** ([cpu_t](#) *cpuinfo)
- void **gdt_install_tss** ([cpu_t](#) *cpuinfo)

5.10.1 Detailed Description

Definition of GDT related data structures.

The Global Descriptor Table (GDT) contains entries telling the CPU about memory segments.

Author

JW

Date

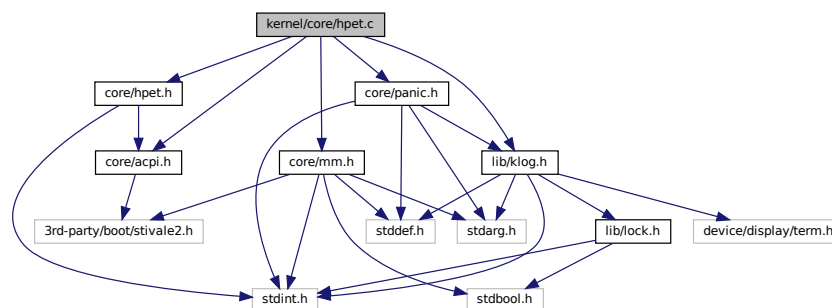
Nov 27, 2021

5.11 kernel/core/hpet.c File Reference

Implementation of HPET (High Precision Event Timer) functions.

```
#include <lib/klog.h>
#include <core/hpet.h>
#include <core/acpi.h>
#include <core/mm.h>
#include <core/panic.h>
```

Include dependency graph for hpet.c:



Functions

- uint64_t **hpet_get_nanos** ()
- void **hpet_nanosleep** (uint64_t nanos)
- void **hpet_init** ()

5.11.1 Detailed Description

Implementation of HPET (High Precision Event Timer) functions.

This module includes implementation of HPET initialization, getting nanos and nanos sleep.

HPET consists of (usually 64-bit) main counter (which counts up), as well as from 3 to 32 32-bit or 64-bit wide comparators. HPET is programmed using memory mapped IO, and the base address of HPET can be found using ACPI.

General initialization:

1. Find HPET base address in 'HPET' ACPI table.
2. Calculate HPET frequency ($f = 10^{15} / \text{period}$).
3. Save minimal tick (either from ACPI table or configuration register).
4. Initialize comparators.
5. Set ENABLE_CNF bit.

Author

JW

Date

DEC 12, 2021

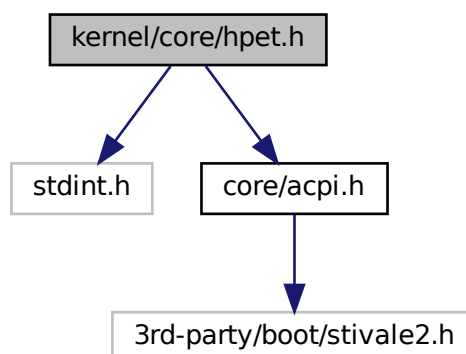
5.12 kernel/core/hpet.h File Reference

Definition of HPET (High Precision Event Timer) related data structures.

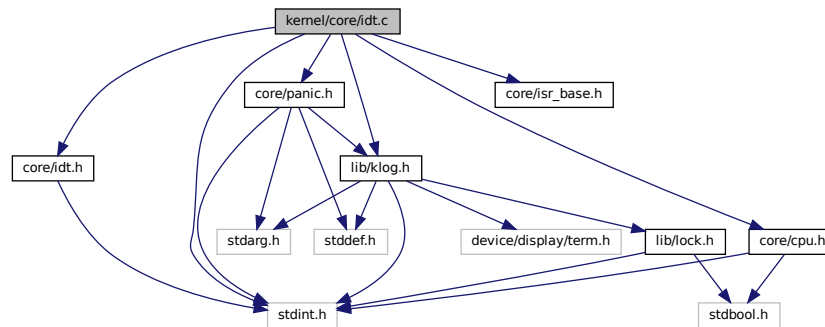
```
#include <stdint.h>
```

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

Include dependency graph for hpet.h:



Include dependency graph for idt.c:



Functions

- void **idt_set_handler** (uint8_t vector, void *handler)
- uint8_t **idt_get_available_vector** (void)
- void **irq_set_mask** (uint8_t line)
- void **irq_clear_mask** (uint8_t line)
- void **idt_init** ()

5.13.1 Detailed Description

Implementation of idt related functions.

The Interrupt Descriptor Table (idt) telling the CPU where the Interrupt Service Routines (ISR) are located (one per interrupt vector). The idt entries are called gates. It can contain Interrupt Gates, Task Gates and Trap Gates. As the first step, only trap gates (exceptions) are implemented.

Ref: https://wiki.osdev.org/Interrupt_Descriptor_Table

Author

JW

Date

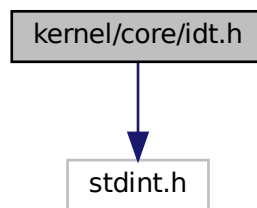
Nov 27, 2021

5.14 kernel/core/idt.h File Reference

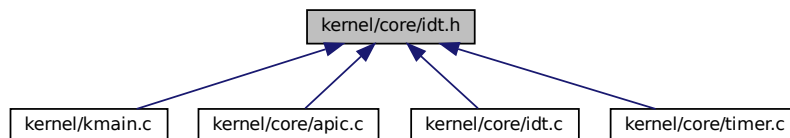
Definition of IDT related data structures.

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

Include dependency graph for idt.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [idt_entry_t](#)
- struct [idt_register_t](#)

Macros

- `#define IDT_ENTRIES 256`
- `#define IDT_DEFAULT_TYPE_ATTRIBUTES 0b10001110`

Functions

- void `idt_init` ()
- void `idt_set_handler` (uint8_t vector, void *handler)
- uint8_t `idt_get_available_vector` (void)
- void `irq_set_mask` (uint8_t line)
- void `irq_clear_mask` (uint8_t line)

5.14.1 Detailed Description

Definition of IDT related data structures.

The Interrupt Descriptor Table (IDT) telling the CPU where the Interrupt Service Routines (ISR) are located (one per interrupt vector).

Author

JW

Date

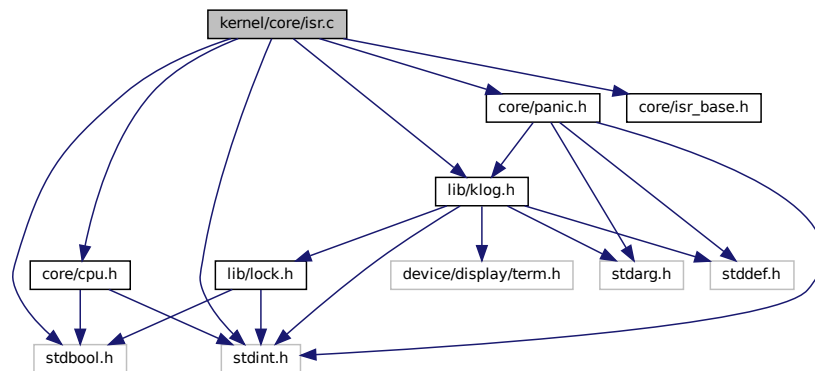
Nov 27, 2021

5.15 kernel/core/isr.c File Reference

Implementation of ISR related functions.

```
#include <stdbool.h>
#include <stdint.h>
#include <lib/klog.h>
#include <core/isr_base.h>
#include <core/panic.h>
#include <core/cpu.h>
```

Include dependency graph for isr.c:



Functions

- void **exc_register_handler** (uint64_t id, exc_handler_t handler)
- void **exc_handler_proc** (uint64_t errcode, uint64_t excno)

5.15.1 Detailed Description

Implementation of ISR related functions.

The x86 architecture is an interrupt driven system. Only a common interrupt handling function is implemented.

Author

JW

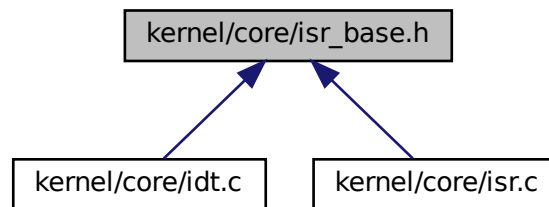
Date

Nov 27, 2021

5.16 kernel/core/isr_base.h File Reference

Definition of ISR related data structures.

This graph shows which files directly or indirectly include this file:



Macros

- `#define PIC1 0x20 /* Master PIC */`
- `#define PIC2 0xA0 /* Slave PIC */`
- `#define PIC1_DATA (PIC1 + 1)`
- `#define PIC2_DATA (PIC2 + 1)`
- `#define PIC_EOI 0x20 /* end of interrupt */`
- `#define IRQ_BASE 0x20`
- `#define IRQ0 32`
- `#define IRQ1 33`
- `#define IRQ2 34`
- `#define IRQ3 35`
- `#define IRQ4 36`
- `#define IRQ5 37`
- `#define IRQ6 38`
- `#define IRQ7 39`
- `#define IRQ8 40`
- `#define IRQ9 41`
- `#define IRQ10 42`
- `#define IRQ11 43`
- `#define IRQ12 44`

Typedefs

- `typedef void(* exc_handler_t) ()`

Functions

- `void exc_register_handler (uint64_t id, exc_handler_t handler)`
- `void exc0 (void *p)`
- `void exc1 (void *p)`
- `void exc2 (void *p)`
- `void exc3 (void *p)`
- `void exc4 (void *p)`
- `void exc5 (void *p)`
- `void exc6 (void *p)`

- void **exc7** (void *p)
- void **exc8** (void *p)
- void **exc10** (void *p)
- void **exc11** (void *p)
- void **exc12** (void *p)
- void **exc13** (void *p)
- void **exc14** (void *p)
- void **exc16** (void *p)
- void **exc17** (void *p)
- void **exc18** (void *p)
- void **exc19** (void *p)
- void **exc20** (void *p)
- void **exc30** (void *p)
- void **irq0** (void *p)
- void **irq1** (void *p)
- void **irq2** (void *p)
- void **irq3** (void *p)
- void **irq4** (void *p)
- void **irq5** (void *p)
- void **irq6** (void *p)
- void **irq7** (void *p)
- void **irq8** (void *p)
- void **irq9** (void *p)
- void **irq10** (void *p)
- void **irq11** (void *p)
- void **irq12** (void *p)

5.16.1 Detailed Description

Definition of ISR related data structures.

The x86 architecture is an interrupt driven system.

Author

JW

Date

Nov 27, 2021

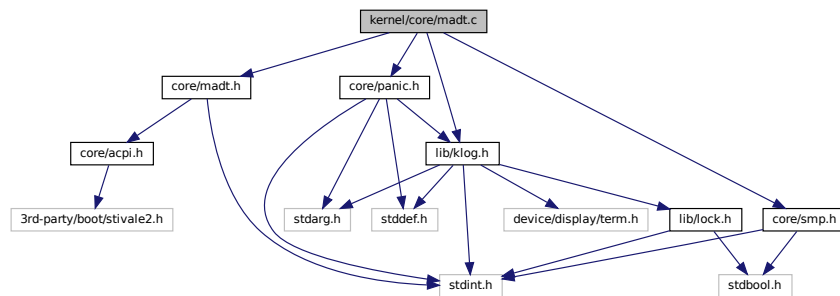
5.17 kernel/core/madt.c File Reference

Implementation of ACPI MADT (Multiple APIC Description Table) functions.

```
#include <core/madt.h>
#include <core/panic.h>
#include <core/smp.h>
```

```
#include <lib/klog.h>
```

Include dependency graph for madt.c:



Functions

- uint32_t **madt_get_num_ioapic** ()
- uint32_t **madt_get_num_lapic** ()
- madt_record_ioapic_t ** **madt_get_ioapics** ()
- madt_record_lapic_t ** **madt_get_lapics** ()
- uint64_t **madt_get_lapic_base** ()
- void **madt_init** ()

5.17.1 Detailed Description

Implementation of ACPI MADT (Multiple APIC Description Table) functions.

The MADT describes all of the interrupt controllers in the system. It can be used to enumerate the processors currently available.

Author

JW

Date

DEC 25, 2021

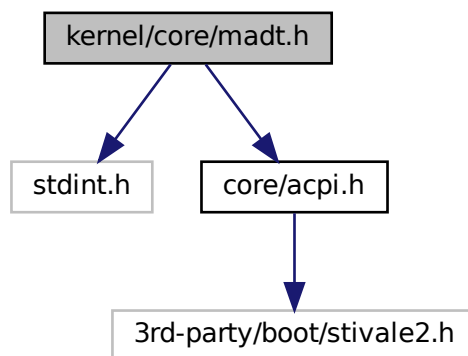
5.18 kernel/core/madt.h File Reference

Definition of ACPI MADT (Multiple APIC Description Table) related data structures.

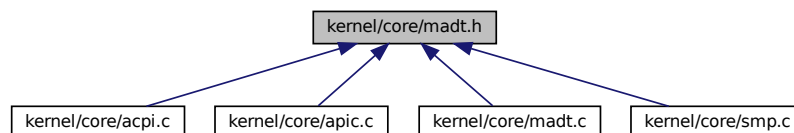
```
#include <stdint.h>
```

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

Include dependency graph for madt.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [madt_record_hdr_t](#)
- struct [madt_record_lapic_t](#)
- struct [madt_record_ioapic_t](#)
- struct [madt_record_iso_t](#)
- struct [madt_record_nmi_t](#)
- struct [madt_t](#)

Macros

- `#define MADT_RECORD_TYPE_LAPIC 0`
- `#define MADT_RECORD_TYPE_IOAPIC 1`
- `#define MADT_RECORD_TYPE_ISO 2`
- `#define MADT_RECORD_TYPE_NMI 4`

- `#define MADT_RECORD_TYPE_LAPIC_AO 5`
- `#define MADT_LAPIC_FLAG_ENABLED (1 << 0)`
- `#define MADT_LAPIC_FLAG_ONLINE_CAPABLE (1 << 1)`

Functions

- `void madt_init ()`
- `uint32_t madt_get_num_ioapic ()`
- `uint32_t madt_get_num_lapic ()`
- `madt_record_ioapic_t** madt_get_ioapics ()`
- `madt_record_lapic_t** madt_get_lapics ()`
- `uint64_t madt_get_lapic_base ()`

5.18.1 Detailed Description

Definition of ACPI MADT (Multiple APIC Description Table) related data structures.

The MADT describes all of the interrupt controllers in the system. It can be used to enumerate the processors currently available.

Author

JW

Date

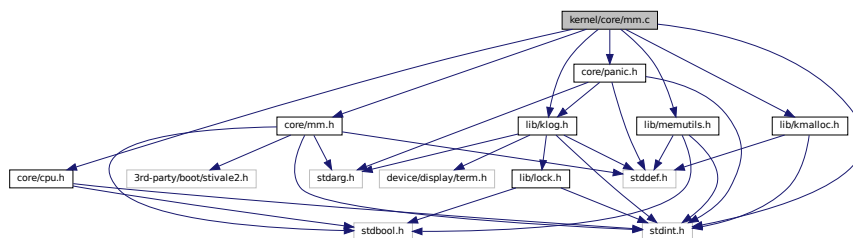
DEC 25, 2021

5.19 kernel/core/mm.c File Reference

Implementation of memory management functions.

```
#include <stdint.h>
#include <core/cpu.h>
#include <core/mm.h>
#include <core/panic.h>
#include <lib/memutils.h>
#include <lib/klog.h>
#include <lib/kmalloc.h>
```

Include dependency graph for mm.c:



Macros

- `#define MAKE_TABLE_ENTRY(address, flags) ((address & ~(0xfff)) | flags)`

Functions

- void **pmm_free** (uint64_t addr, uint64_t numpages)
- bool **pmm_alloc** (uint64_t addr, uint64_t numpages)
- uint64_t **pmm_get** (uint64_t numpages, uint64_t baseaddr)
- void **pmm_init** (struct stivale2_struct_tag_memmap *map)
- void **pmm_dump_usage** (void)
- void **vmm_unmap** (uint64_t vaddr, uint64_t np)
- void **vmm_map** (uint64_t vaddr, uint64_t paddr, uint64_t np, uint64_t flags)
- void **vmm_init** ()

5.19.1 Detailed Description

Implementation of memory management functions.

Memory management is a critical part of any operating system kernel. Providing a quick way for programs to allocate and free memory on a regular basis is a major responsibility of the kernel.

High Half Kernel: To setup a higher half kernel, you have to map your kernel to the appropriate virtual address. Without a boot loader help, you'll need a small trampoline code which runs in lower half, sets up higher half paging and jumps.

If page protection is not enabled, virtual address is equal with physical address. The highest bit of CR0 indicates whether paging is enabled or not: `mov cr0,80000000` can enable paging.

PMM: The method behind PMM is very simple. The memories with type - STIVALE2_MMAP_USABLE are divided into 4K-size pages. A bitmap array is used for indicated whether it is free or not. One bit for one page in bitmap array.

Author

JW

Date

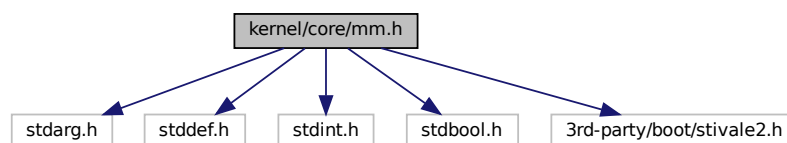
Nov 27, 2021

5.20 kernel/core/mm.h File Reference

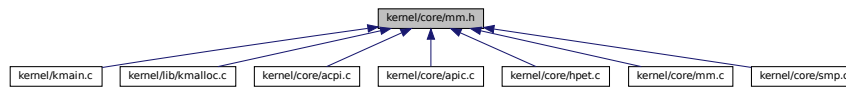
Definition of memory management related data structures.

```
#include <stdarg.h>
#include <stddef.h>
#include <stdint.h>
#include <stdbool.h>
#include <3rd-party/boot/stivale2.h>
```

Include dependency graph for mm.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [mem_info_t](#)
- struct [addrspace_t](#)

Macros

- #define **PAGE_SIZE** 4096
- #define **BMP_PAGES_PER_BYTE** 8
- #define **USERSPACE_OFFSET** 0x20000000
- #define **HIGHERHALF_OFFSET** 0xffffffff80000000
- #define **NUM_PAGES**(num) (((num) + PAGE_SIZE - 1) / PAGE_SIZE)
- #define **PAGE_ALIGN_UP**(num) (NUM_PAGES(num) * PAGE_SIZE)
- #define **VMM_FLAG_PRESENT** (1 << 0)
- #define **VMM_FLAG_READWRITE** (1 << 1)
- #define **VMM_FLAG_USER** (1 << 2)
- #define **VMM_FLAG_WRITETHROUGH** (1 << 3)
- #define **VMM_FLAG_CACHE_DISABLE** (1 << 4)
- #define **VMM_FLAG_WRITECOMBINE** (1 << 7)
- #define **VMM_FLAGS_DEFAULT** (VMM_FLAG_PRESENT | VMM_FLAG_READWRITE)
- #define **VMM_FLAGS_MMIO** (VMM_FLAGS_DEFAULT | VMM_FLAG_CACHE_DISABLE)
- #define **VMM_FLAGS_USERMODE** (VMM_FLAGS_DEFAULT | VMM_FLAG_USER)
- #define **MEM_VIRT_OFFSET** 0xffff800000000000
- #define **VIRT_TO_PHYS**(a) (((uint64_t)(a)) - MEM_VIRT_OFFSET)
- #define **PHYS_TO_VIRT**(a) (((uint64_t)(a)) + MEM_VIRT_OFFSET)

Functions

- void **pmm_init** (struct stivale2_struct_tag_memmap *map)
- uint64_t **pmm_get** (uint64_t numpages, uint64_t baseaddr)
- bool **pmm_alloc** (uint64_t addr, uint64_t numpages)
- void **pmm_free** (uint64_t addr, uint64_t numpages)
- void **pmm_dump_usage** (void)
- void **vmm_init** ()
- void **vmm_map** (uint64_t vaddr, uint64_t paddr, uint64_t np, uint64_t flags)
- void **vmm_unmap** (uint64_t vaddr, uint64_t np)

5.20.1 Detailed Description

Definition of memory management related data structures.

Memory management is a critical part of any operating system kernel. Providing a quick way for programs to allocate and free memory on a regular basis is a major responsibility of the kernel.

Author

JW

Date

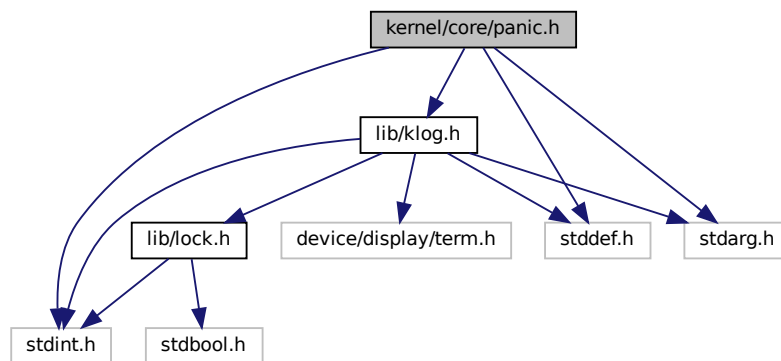
Nov 27, 2021

5.21 kernel/core/panic.h File Reference

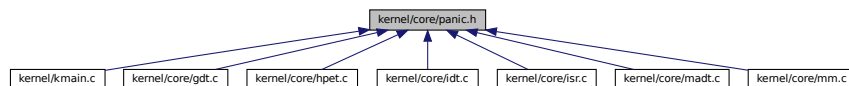
Implementation of panic related functions.

```
#include <stdint.h>
#include <stddef.h>
#include <stdarg.h>
#include <lib/klog.h>
```

Include dependency graph for panic.h:



This graph shows which files directly or indirectly include this file:



Macros

- `#define kpanic(s, ...)`

Functions

- `void dump_backtrace ()`

5.21.1 Detailed Description

Implementation of panic related functions.

Definition of panic related data structures.

A kernel panic is one of several boot issues. In basic terms, it is a situation when the kernel can't load properly and therefore the system fails to boot.

Author

JW

Date

Nov 27, 2021

5.21.2 Macro Definition Documentation

5.21.2.1 kpanic

```
#define kpanic(  
    s,  
    ... )
```

Value:

```
{ \
asm volatile("cli"); \
klog_vprintf(KLOG_LEVEL_ERROR, s, ##__VA_ARGS__); \
dump_backtrace(); \
while (true) \
    asm volatile("hlt"); \
}
```

5.22 kernel/core/pit.c File Reference

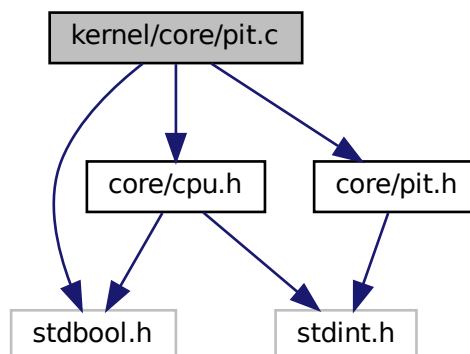
Implementation of PIT related functions.

```
#include <stdbool.h>
```

```
#include <core/pit.h>
```

```
#include <core/cpu.h>
```

Include dependency graph for pit.c:



Functions

- void `pit_wait` (uint64_t ms)

5.22.1 Detailed Description

Implementation of PIT related functions.

This PIT sleep function should be rewritten based on IRQ mechanism, then it can be used in different processes at the same time.

Author

JW

Date

Mar 12, 2022

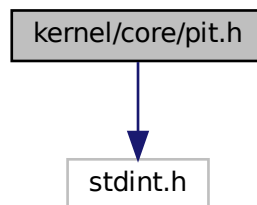
Todo Rewrite pit_wait() function

5.23 kernel/core/pit.h File Reference

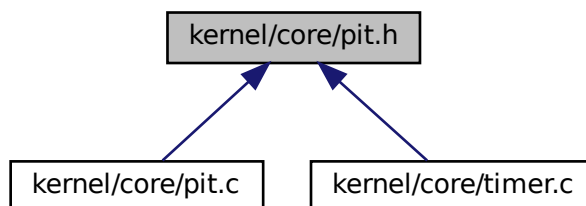
Definition of PIT related data structures.

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

Include dependency graph for pit.h:



This graph shows which files directly or indirectly include this file:



Functions

- void **pit_wait** (uint64_t ms)

5.23.1 Detailed Description

Definition of PIT related data structures.

The Programmable Interval Timer (PIT) chip (Intel 8253/8254) basically consists of an oscillator, a prescaler and 3 independent frequency dividers. Each frequency divider has an output, which is used to allow the timer to control external circuitry (for example, IRQ 0).

Author

JW

Date

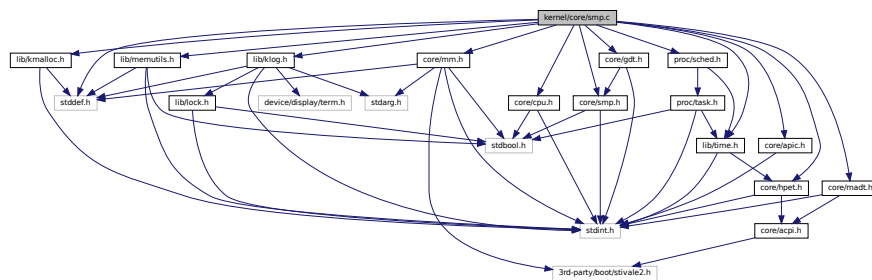
Mar 12, 2022

5.24 kernel/core/smp.c File Reference

Implementation of SMP related functions.

```
#include <stddef.h>
#include <lib/klog.h>
#include <lib/kmalloc.h>
#include <lib/memutils.h>
#include <lib/time.h>
#include <core/mm.h>
#include <core/cpu.h>
#include <core/smp.h>
#include <core/gdt.h>
#include <core/hpet.h>
#include <core/madt.h>
#include <core/apic.h>
#include <proc/sched.h>
```

Include dependency graph for smp.c:



Functions

- const [smp_info_t](#) * **smp_get_info** ()
- [cpu_t](#) * **smp_get_current_cpu** (bool force_read)
- void **init_tss** ([cpu_t](#) *cpuinfo)
- [_Noreturn](#) void **smp_ap_entrpoint** ([cpu_t](#) *cpuinfo)
- void **smp_init** ()

Variables

- uint8_t **smp_trampoline_blob_start**
- uint8_t **smp_trampoline_blob_end**

5.24.1 Detailed Description

Implementation of SMP related functions.

Symmetric Multiprocessing (or SMP) is one method of having multiple processors in one computer system.

Ref: <https://wiki.osdev.org/SMP>

Author

JW

Date

Jan 2, 2022

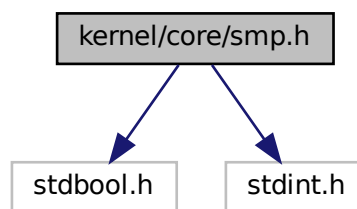
5.25 kernel/core/smp.h File Reference

Definition of SMP related data structures.

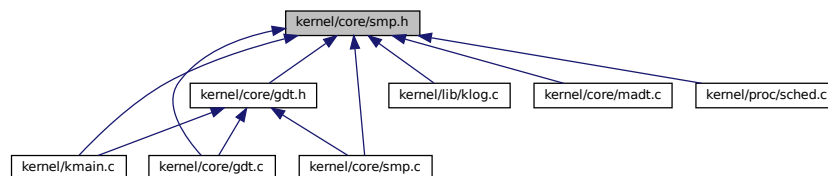
```
#include <stdbool.h>
```

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

Include dependency graph for smp.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct `tss_t`
- struct `cpu_t`
- struct `smp_info_t`

Macros

- `#define SMP_TRAMPOLINE_BLOB_ADDR 0x70000`
- `#define SMP_AP_BOOT_COUNTER_ADDR 0xff0`
- `#define SMP_TRAMPOLINE_ARG_IDTPTR 0xfa0`
- `#define SMP_TRAMPOLINE_ARG_RSP 0xfb0`
- `#define SMP_TRAMPOLINE_ARG_ENTRYPOINT 0xfc0`
- `#define SMP_TRAMPOLINE_ARG_CR3 0xfd0`
- `#define SMP_TRAMPOLINE_ARG_CPUINFO 0xfe0`
- `#define CPU_MAX 256`

Functions

- `void smp_init (void)`
- `const smp_info_t * smp_get_info (void)`
- `cpu_t * smp_get_current_cpu (bool force_read)`

5.25.1 Detailed Description

Definition of SMP related data structures.

Symmetric Multiprocessing (or SMP) is one method of having multiple processors in one computer system.

Author

JW

Date

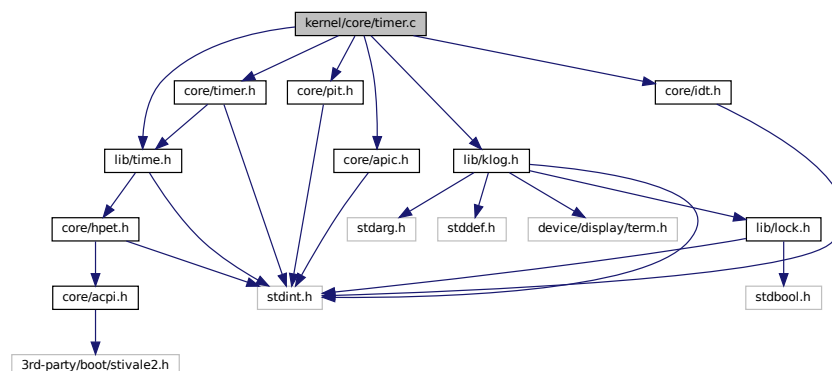
Jan 2, 2022

5.26 kernel/core/timer.c File Reference

Implementation of APIC timer related functions.

```
#include <core/timer.h>
#include <core/apic.h>
#include <core/idt.h>
#include <core/pit.h>
#include <lib/klog.h>
#include <lib/time.h>
```

Include dependency graph for timer.c:



Functions

- void **apic_timer_stop** (void)
- void **apic_timer_start** (void)
- void **apic_timer_set_handler** (void(*h)(void *))
- void **apic_timer_set_frequency** (uint64_t freq)
- void **apic_timer_set_period** (time_t tv)
- uint8_t **apic_timer_get_vector** (void)
- void **apic_timer_set_mode** (apic_timer_mode_t mode)
- void **apic_timer_enable** (void)
- void **apic_timer_init** (void)

5.26.1 Detailed Description

Implementation of APIC timer related functions.

The timer has 2 or 3 modes. The first 2 modes (periodic and one-shot) are supported by all local APICs. The third mode (TSC-Deadline mode) is an extension that is only supported on recent CPUs.

Periodic Mode:

- Generate IRQs at a fixed rate depending on the initial count.

One-Shot Mode:

- Decrement the current count (and generates a timer IRQ when the count reaches zero) in the same way as in periodic mode. However it doesn't reset the current count to the initial count when the current count reaches zero.

TSC-Deadline mode:

- Similar with one-shot mode but using CPU's time stamp counter instead to get higher precision.

Ref: https://wiki.osdev.org/APIC_timer

Author

JW

Date

Jan 8, 2022

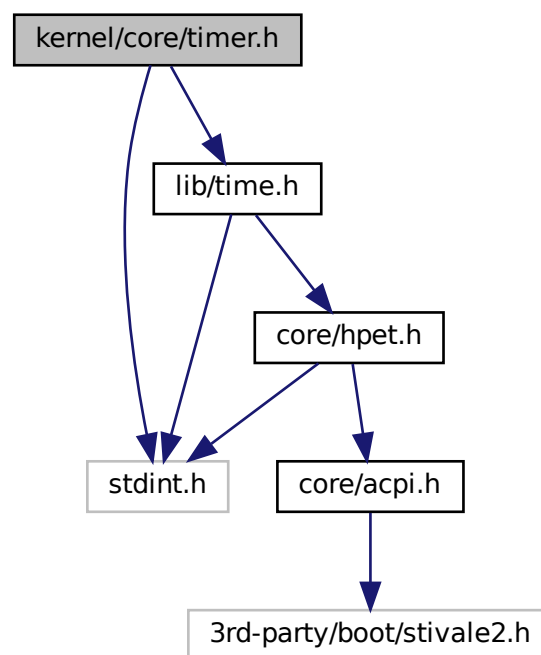
5.27 kernel/core/timer.h File Reference

Definition of APIC timer related macros, data structures, functions.

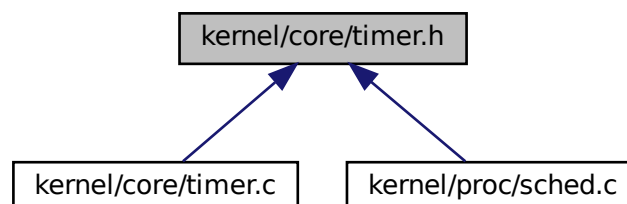
```
#include <stdint.h>
```

```
#include <lib/time.h>
```

Include dependency graph for timer.h:



This graph shows which files directly or indirectly include this file:



Macros

- `#define APIC_REG_TIMER_LVT 0x320`
- `#define APIC_REG_TIMER_ICR 0x380`
- `#define APIC_REG_TIMER_CCR 0x390`
- `#define APIC_REG_TIMER_DCR 0x3e0`
- `#define APIC_TIMER_FLAG_PERIODIC (1 << 17)`
- `#define APIC_TIMER_FLAG_MASKED (1 << 16)`

Enumerations

- `enum apic_timer_mode_t { APIC_TIMER_MODE_PERIODIC, APIC_TIMER_MODE_ONESHOT }`

Functions

- `void apic_timer_init (void)`
- `void apic_timer_enable (void)`
- `void apic_timer_stop (void)`
- `void apic_timer_start (void)`
- `void apic_timer_set_handler (void(*) (void *))`
- `void apic_timer_set_frequency (uint64_t freq)`
- `void apic_timer_set_period (time_t tv)`
- `void apic_timer_set_mode (apic_timer_mode_t mode)`
- `uint8_t apic_timer_get_vector (void)`

5.27.1 Detailed Description

Definition of APIC timer related macros, data structures, functions.

The great benefit of the Local APIC timer is that it is hardwired to each CPU core, unlike the Programmable Interval Timer which is a separate circuit. Because of this, there is no need for any resource management, which makes things easier. The downside is that it's oscillating at (one of) the CPU's frequencies, which varies from machine to machine, while the PIT uses a standard frequency (1,193,182 Hz). To make use of it, you have to know how many interrupts/sec it's capable of.

Author

JW

Date

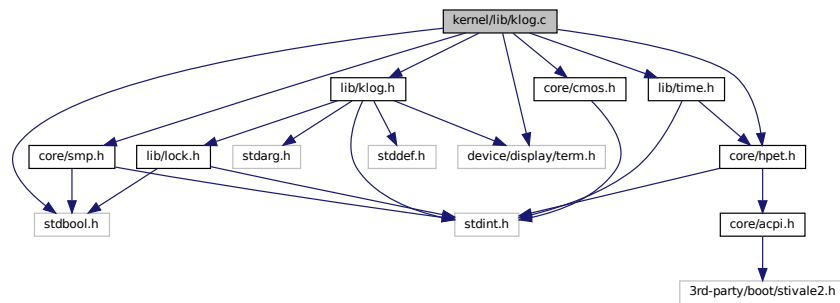
Jan 8, 2022

5.28 kernel/kmain.c File Reference

Entry function of HanOS kernel.

```
#include <stddef.h>
#include <3rd-party/boot/stivale2.h>
#include <lib/time.h>
#include <lib/klog.h>
#include <core/mm.h>
#include <core/gdt.h>
#include <core/idt.h>
#include <core/smp.h>
#include <core/cmos.h>
#include <core/acpi.h>
#include <core/apic.h>
```

Include dependency graph for klog.c:



Functions

- void **klog_init** ()
- void **klog_vprintf_core** (int mode, const char *s, va_list args)
- void **klog_vprintf_wrapper** (int mode, const char *s,...)
- void **klog_lock** (void)
- void **klog_unlock** (void)
- void **klog_vprintf** (klog_level_t level, const char *s,...)
- void **kprintf** (const char *s,...)

5.29.1 Detailed Description

Implementation of kernel log related functions.

A kernel-level log system was implemented. As the first step, it mainly supports information display.

Author

JW

Date

Nov 20, 2021

5.30 kernel/lib/klog.h File Reference

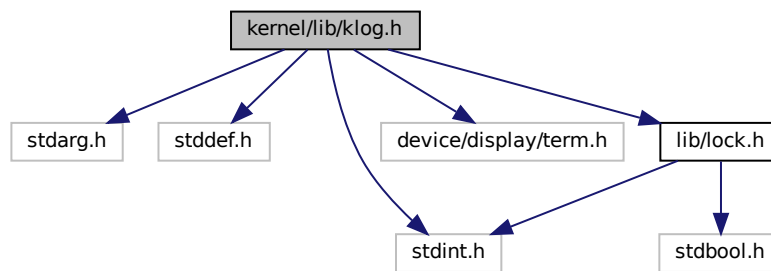
Definition of kernel log related functions.

```

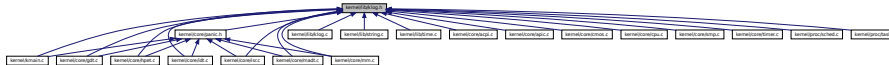
#include <stdarg.h>
#include <stddef.h>
#include <stdint.h>
#include <device/display/term.h>
#include <lib/lock.h>

```

Include dependency graph for klog.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [klog_info_t](#)

Macros

- `#define KLOG_BUFFER_SIZE (UINT16_MAX + 1)`
- `#define klogv(s, ...) klog_vprintf(KLOG_LEVEL_VERBOSE, s, ##__VA_ARGS__)`
- `#define klogd(s, ...) klog_vprintf(KLOG_LEVEL_DEBUG, s, ##__VA_ARGS__)`
- `#define klogi(s, ...) klog_vprintf(KLOG_LEVEL_INFO, s, ##__VA_ARGS__)`
- `#define klogw(s, ...) klog_vprintf(KLOG_LEVEL_WARN, s, ##__VA_ARGS__)`
- `#define kloge(s, ...) klog_vprintf(KLOG_LEVEL_ERROR, s, ##__VA_ARGS__)`
- `#define klogu(s, ...) klog_vprintf(KLOG_LEVEL_UNK, s, ##__VA_ARGS__)`
- `#define klog_printf(s, ...) klog_vprintf(KLOG_LEVEL_INFO, s, ##__VA_ARGS__)`

Enumerations

- enum `klog_level_t` {
`KLOG_LEVEL_VERBOSE`, `KLOG_LEVEL_DEBUG`, `KLOG_LEVEL_INFO`, `KLOG_LEVEL_WARN`,
`KLOG_LEVEL_ERROR`, `KLOG_LEVEL_UNK` }

Functions

- void `klog_init` (void)
- void `klog_vprintf` (klog_level_t level, const char *,...)
- void `klog_lock` (void)
- void `klog_unlock` (void)
- void `kprintf` (const char *,...)

5.30.1 Detailed Description

Definition of kernel log related functions.

A kernel-level log system was implemented. As the first step, it mainly supports information display.

Author

JW

Date

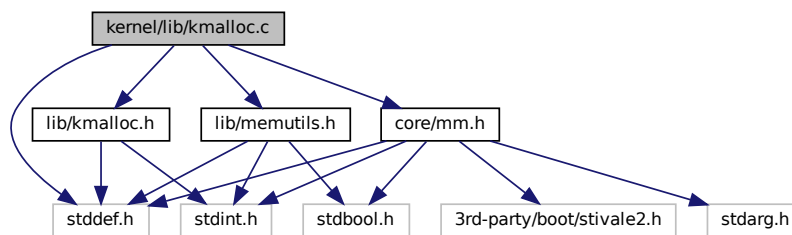
Nov 20, 2021

5.31 kernel/lib/kmalloc.c File Reference

Implementation of memory allocation related functions.

```
#include <stddef.h>
#include <lib/kmalloc.h>
#include <lib/memutils.h>
#include <core/mm.h>
```

Include dependency graph for kmalloc.c:



Data Structures

- struct [metadata_t](#)

Functions

- void * **kmalloc** (uint64_t size)
- void **kfree** (void *addr)
- void * **kmrealloc** (void *addr, size_t newsize)
- void * **umalloc** (uint64_t size)
- void **umfree** (void *addr)

5.31.1 Detailed Description

Implementation of memory allocation related functions.

e.g., malloc, free and realloc.

Author

JW

Date

Jan 2, 2022

Todo Memory allocation should be improved for better efficiency.

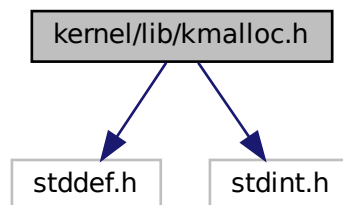
5.32 kernel/lib/kmalloc.h File Reference

Definition of memory allocation related functions.

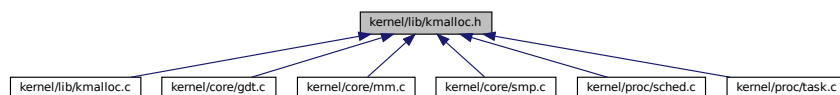
```
#include <stddef.h>
```

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

Include dependency graph for kmalloc.h:



This graph shows which files directly or indirectly include this file:



Functions

- void * **kmalloc** (uint64_t size)
- void **kfree** (void *addr)
- void * **kmrealloc** (void *addr, size_t newsz)
- void * **umalloc** (uint64_t size)
- void **umfree** (void *addr)

5.32.1 Detailed Description

Definition of memory allocation related functions.

e.g., malloc, free and realloc.

Author

JW

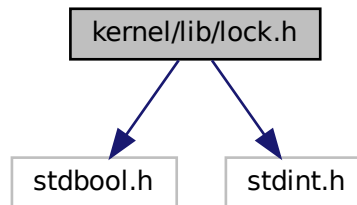
Date

Jan 2, 2022

5.33 kernel/lib/lock.h File Reference

Definition of lock related data structures and functions.

```
#include <stdbool.h>
#include <stdint.h>
Include dependency graph for lock.h:
```



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [lock_t](#)

Macros

- #define **lock_new()** ([lock_t](#)){0, 0}
- #define **lock_lock(s)**
- #define **lock_release(s)**

5.33.1 Detailed Description

Definition of lock related data structures and functions.

e.g., lock new, lock and release.

Author

JW

Date

Jan 2, 2022

5.33.2 Macro Definition Documentation

5.33.2.1 lock_lock

```
#define lock_lock(  
    s )
```

Value:

```
{  
    asm volatile(  
        "pushfq;"  
        "cli;"  
        "lock btsl $0, %[lock];"  
        "jnc 2f;"  
        "1:"  
        "pause;"  
        "btl $0, %[lock];"  
        "jc 1b;"  
        "lock btsl $0, %[lock];"  
        "jc 1b;"  
        "2:"  
        "pop %[flags]"  
        : [lock] "=m"((s)->lock), [flags] "=m"((s)->rflags)  
        :  
        : "memory", "cc");  
}
```

5.33.2.2 lock_release

```
#define lock_release(  
    s )
```

Value:

```
{  
    asm volatile("push %[flags];"  
        "lock btrl $0, %[lock];"  
        "popfq;"  
        : [lock] "=m"((s)->lock)  
        : [flags] "m"((s)->rflags)  
        : "memory", "cc");  
}
```

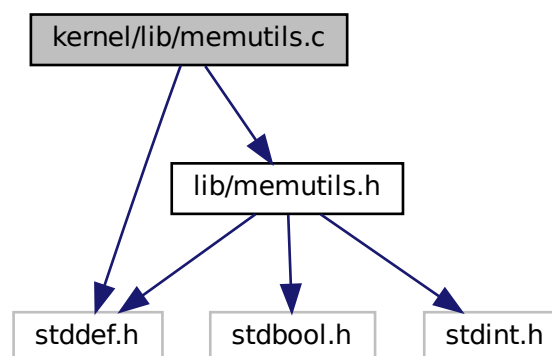
5.34 kernel/lib/memutils.c File Reference

Implementation of memory operation related functions.

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

```
#include <lib/memutils.h>
```

Include dependency graph for memutils.c:



Functions

- void **memcpy** (void *target, const void *src, uint64_t len)
- void **memset** (void *addr, uint8_t val, uint64_t len)
- bool **memcmp** (const void *s1, const void *s2, uint64_t len)

5.34.1 Detailed Description

Implementation of memory operation related functions.

e.g., comparison, set value and copy.

Author

JW

Date

Jan 2, 2022

5.35 kernel/lib/memutils.h File Reference

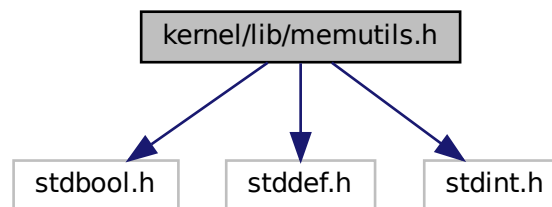
Definition of memory operation related functions.

```
#include <stdbool.h>
```

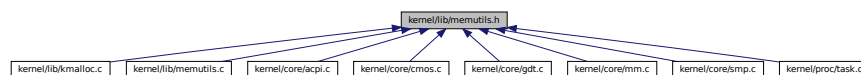
```
#include <stddef.h>
```

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

Include dependency graph for memutils.h:



This graph shows which files directly or indirectly include this file:



Functions

- bool **memcmp** (const void *s1, const void *s2, uint64_t len)
 - void **memset** (void *addr, uint8_t val, uint64_t len)
 - void **memcpy** (void *target, const void *src, uint64_t len)
-

5.35.1 Detailed Description

Definition of memory operation related functions.

e.g., comparison, set value and copy.

Author

JW

Date

Jan 2, 2022

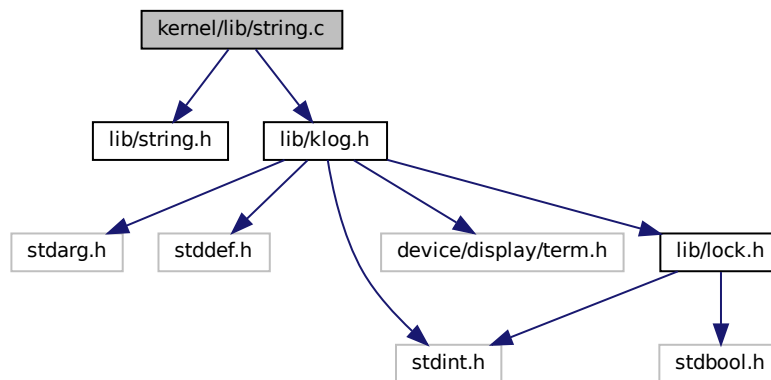
5.36 kernel/lib/string.c File Reference

Implementation of string operation related functions.

```
#include <lib/string.h>
```

```
#include <lib/klog.h>
```

Include dependency graph for string.c:



Functions

- int **strlen** (const char *s)

5.36.1 Detailed Description

Implementation of string operation related functions.

String functions including length, comparison, copy etc.

Author

JW

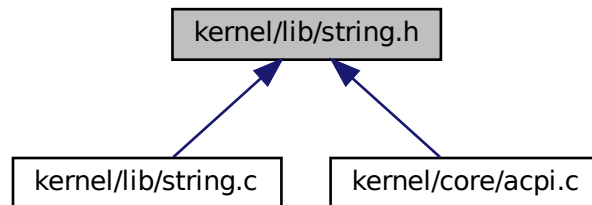
Date

Jan 2, 2022

5.37 kernel/lib/string.h File Reference

Definition of string operation related functions.

This graph shows which files directly or indirectly include this file:



Functions

- int **strlen** (const char *s)

5.37.1 Detailed Description

Definition of string operation related functions.

String functions including length, comparison, copy etc.

Author

JW

Date

Jan 2, 2022

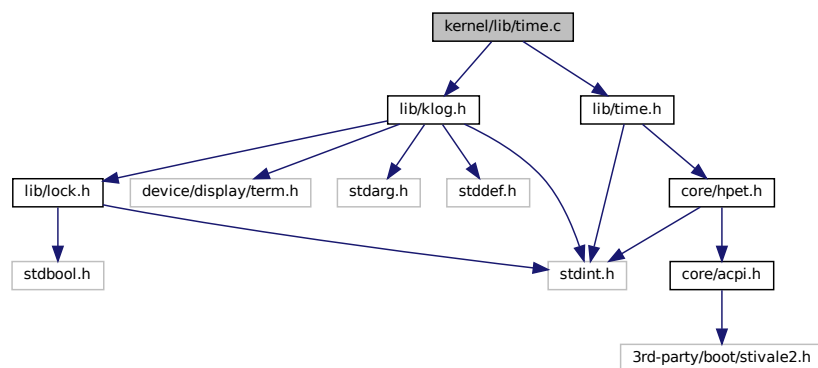
5.38 kernel/lib/time.c File Reference

Implementation of time related functions.

```
#include <lib/klog.h>
```

```
#include <lib/time.h>
```

Include dependency graph for time.c:



Functions

- void **localtime** (const time_t *timep, tm_t *_timevalue)
- time_t **mktime** (tm_t *tm)

5.38.1 Detailed Description

Implementation of time related functions.

e.g., localtime...

Author

JW

Date

Jan 2, 2022

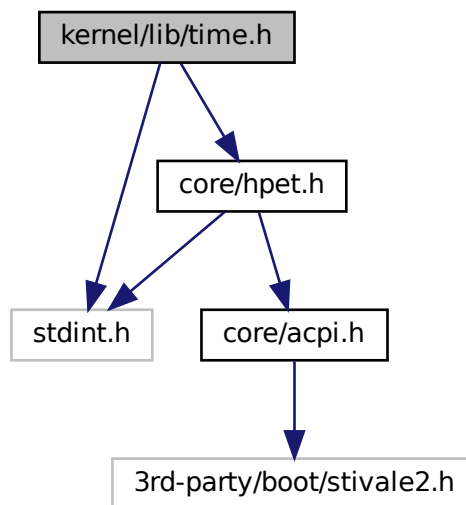
5.39 kernel/lib/time.h File Reference

Definition of time related data structures and functions.

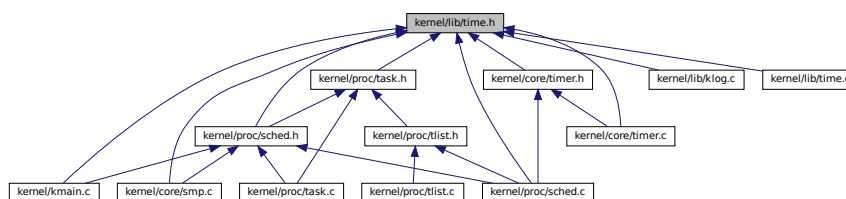
```
#include <stdint.h>
```

```
#include <core/hpet.h>
```

Include dependency graph for time.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [timeval_t](#)
- struct [timezone_t](#)
- struct [tm_t](#)

Macros

- #define **sleep**(x) hpet_nanosleep(MILLIS_TO_NANOS(x))
- #define **SECONDS_TO_NANOS**(x) ((x)*1000000000ULL)
- #define **MILLIS_TO_NANOS**(x) ((x)*1000000ULL)
- #define **MICROS_TO_NANOS**(x) ((x)*1000ULL)
- #define **NANOS_TO_SECONDS**(x) ((x) / 1000000000ULL)
- #define **NANOS_TO_MILLIS**(x) ((x) / 1000000ULL)
- #define **NANOS_TO_MICROS**(x) ((x) / 1000ULL)

Typedefs

- typedef uint64_t **time_t**

Functions

- void **localtime** (const time_t *timep, [tm_t](#) *tm)

5.39.1 Detailed Description

Definition of time related data structures and functions.

e.g., time value, time zone and time information.

Author

JW

Date

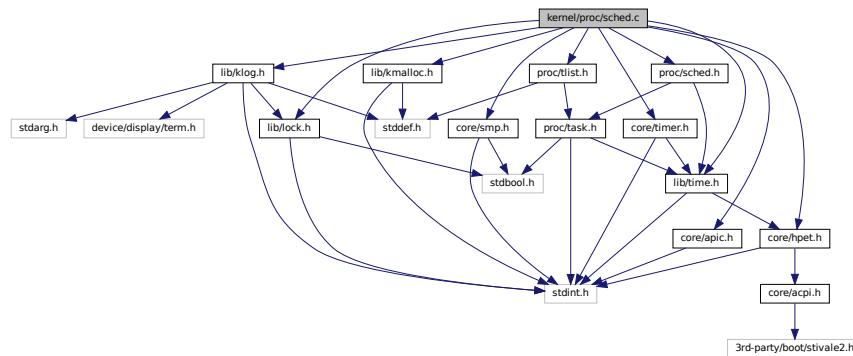
Jan 2, 2022

5.40 kernel/proc/sched.c File Reference

Implementation of scheduling related functions.

```
#include <lib/klog.h>
#include <lib/lock.h>
#include <lib/time.h>
#include <lib/kmalloc.h>
#include <proc/sched.h>
#include <proc/tlist.h>
#include <core/smp.h>
#include <core/timer.h>
#include <core/apic.h>
```

```
#include <core/hpet.h>
Include dependency graph for sched.c:
```



Macros

- `#define TIMESLICE_DEFAULT MILLIS_TO_NANOS(1)`

Functions

- void **enter_context_switch** (void *v)
- void **exit_context_switch** ([task_t](#) *next)
- void **do_context_switch** (void *stack)
- void **sched_init** (uint16_t cpu_id)
- void **sched_add** (void(*entry)(task_id_t))

5.40.1 Detailed Description

Implementation of scheduling related functions.

Include Context Switching, Scheduling Algorithms etc.

Author

JW

Date

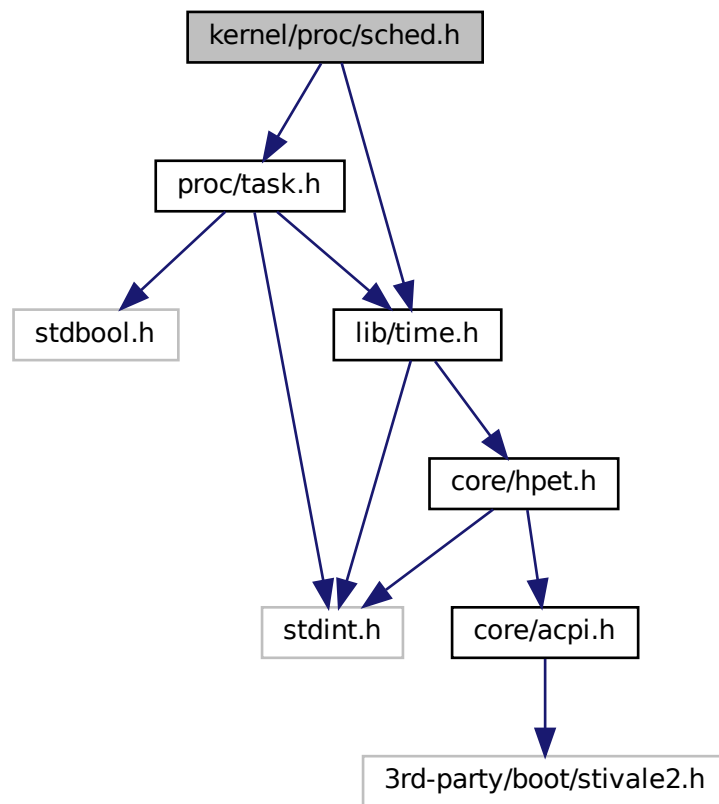
Jan 2, 2022

5.41 kernel/proc/sched.h File Reference

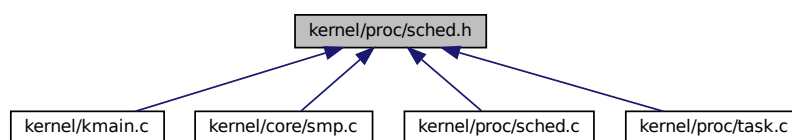
Definition of scheduling related functions.

```
#include <proc/task.h>
#include <lib/time.h>
```

Include dependency graph for sched.h:



This graph shows which files directly or indirectly include this file:



Functions

- void **sched_init** (uint16_t cpu_id)
- void **sched_add** (void(*entry)(task_id_t))

5.41.1 Detailed Description

Definition of scheduling related functions.

Include Context Switching, Scheduling Algorithms etc.

Author

JW

Date

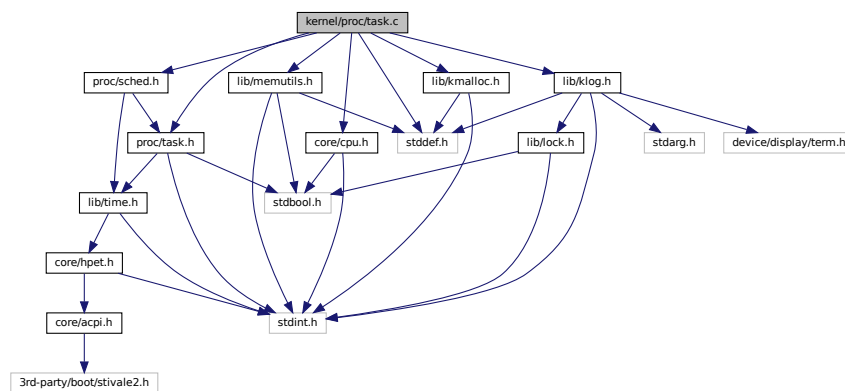
Jan 2, 2022

5.42 kernel/proc/task.c File Reference

Implementation of task related functions.

```
#include <stddef.h>
#include <proc/task.h>
#include <proc/sched.h>
#include <lib/kmalloc.h>
#include <lib/memutils.h>
#include <lib/klog.h>
#include <core/cpu.h>
```

Include dependency graph for task.c:



Functions

- [task_t](#) * **task_make** (void(*entry)(task_id_t), task_priority_t priority, task_mode_t mode)

5.42.1 Detailed Description

Implementation of task related functions.

Create and return task data structure which contains registers and other task related information.

Author

JW

Date

Jan 2, 2022

5.43 kernel/proc/task.h File Reference

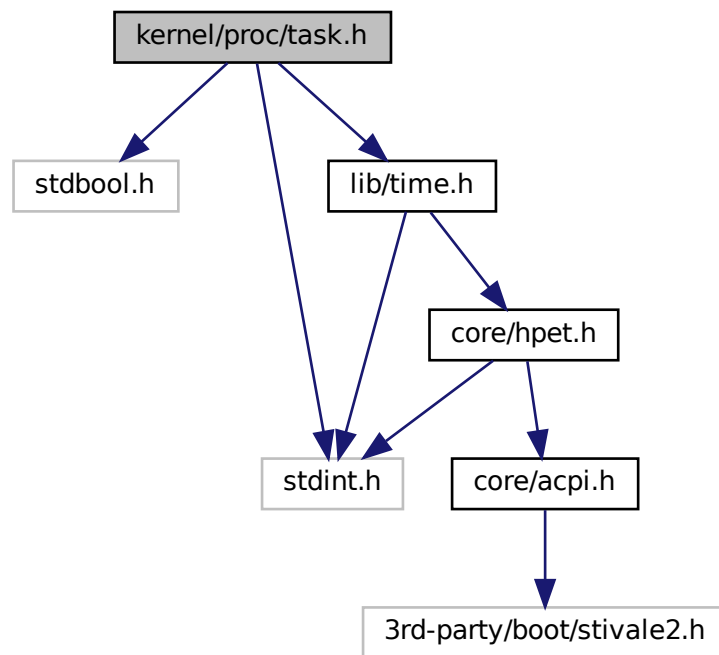
Definition of task related functions.

```
#include <stdbool.h>
```

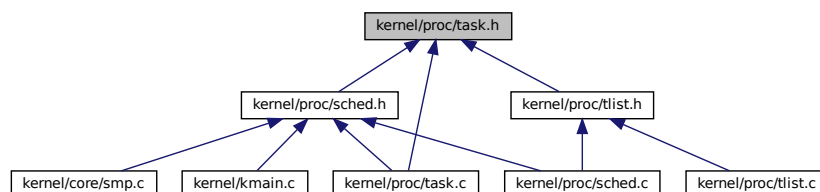
```
#include <stdint.h>
```

```
#include <lib/time.h>
```

Include dependency graph for task.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct `task_regs_t`
- struct `task_t`

Macros

- `#define KSTACK_SIZE 4096`

- `#define DEFAULT_KMODE_CS 0x08`
- `#define DEFAULT_KMODE_SS 0x10`
- `#define DEFAULT_UMODE_CS 0x1b`
- `#define DEFAULT_UMODE_SS 0x23`
- `#define DEFAULT_RFLAGS 0x0202`
- `#define TID_MAX UINT16_MAX`

Typedefs

- `typedef uint16_t task_id_t`
- `typedef uint8_t task_priority_t`
- `typedef struct task_t task_t`

Enumerations

- `enum task_mode_t { TASK_KERNEL_MODE, TASK_USER_MODE }`
- `enum task_status_t { TASK_READY, TASK_RUNNING, TASK_SLEEPING, TASK_DEAD }`

Functions

- `task_t * task_make (void(*entry)(task_id_t), task_priority_t priority, task_mode_t mode)`

5.43.1 Detailed Description

Definition of task related functions.

Create and return task data structure which contains registers and other task related information.

Author

JW

Date

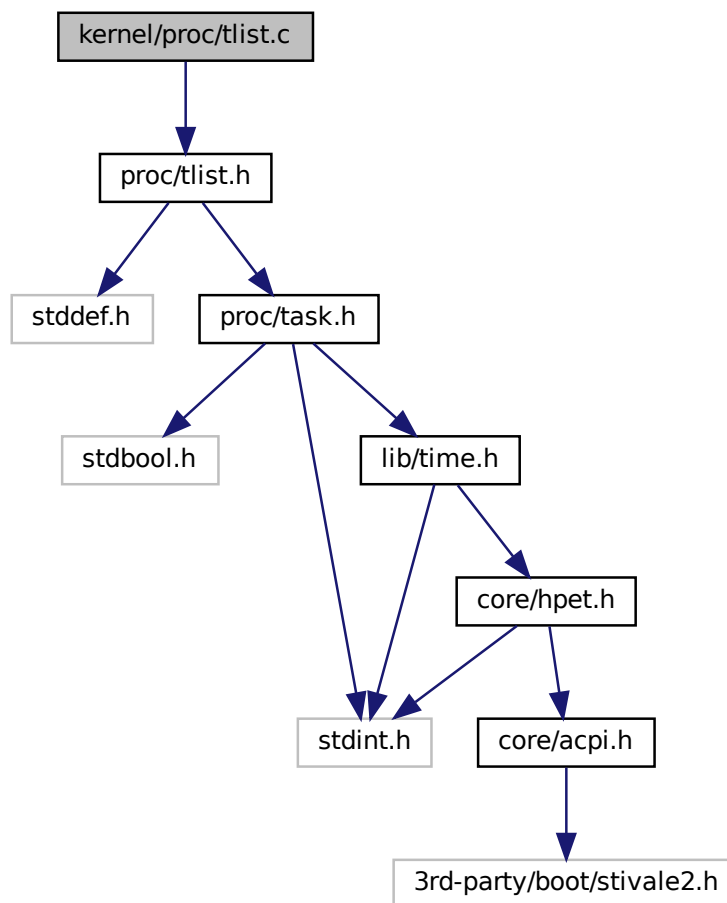
Jan 2, 2022

5.44 kernel/proc/tlist.c File Reference

Implementation of task list related functions.

```
#include <proc/tlist.h>
```

Include dependency graph for tlist.c:



Functions

- `void task_list_push (task_list_t *tl, task_t *t)`
- `task_t * task_list_pop (task_list_t *tl)`

5.44.1 Detailed Description

Implementation of task list related functions.

Push and pop task data structure from task list.

Author

JW

Date

Jan 2, 2022

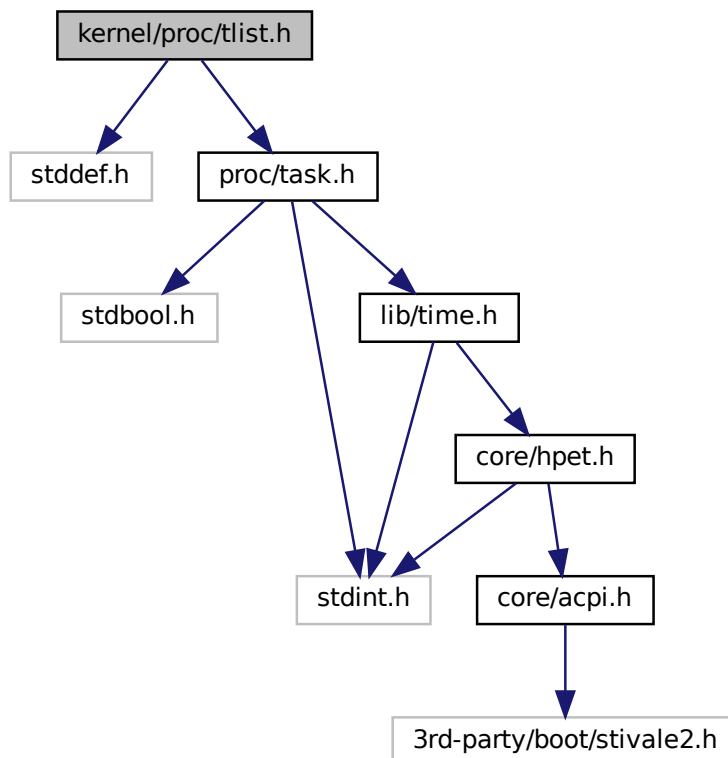
5.45 kernel/proc/tlist.h File Reference

Definition of task list related functions.

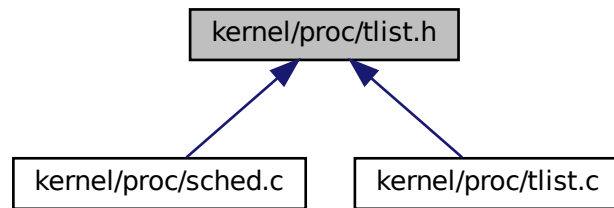
```
#include <stddef.h>
```

```
#include <proc/task.h>
```

Include dependency graph for tlist.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct `task_list_t`

Functions

- `task_t * task_list_pop (task_list_t *tl)`
- `void task_list_push (task_list_t *tl, task_t *t)`

5.45.1 Detailed Description

Definition of task list related functions.

Push and pop task data structure from task list.

Author

JW

Date

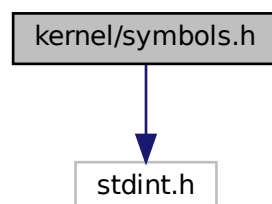
Jan 2, 2022

5.46 kernel/symbols.h File Reference

Definition of symbol related data structures.

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

Include dependency graph for symbols.h:



Data Structures

- struct [symbol_t](#)

Variables

- const [symbol_t](#) `_kernel_symtab []`

5.46.1 Detailed Description

Definition of symbol related data structures.

Symbols are used for backtrace when kernel is crashed. It can help to provide context information for debugging.

Author

JW

Date

Nov 27, 2021

Index

acpi_gas_t, 7
acpi_sdt_hdr_t, 7
acpi_sdt_t, 8
addrspace_t, 8

cmos_rtc_t, 8
cpu.h
 read_cr, 32
 write_cr, 32
cpu_t, 9
cpuid_feature_t, 9

gdt_entry_t, 10
gdt_register_t, 10
gdt_table_t, 10

hpet_sdt_t, 11
hpet_t, 12
hpet_timer_t, 12

idt_entry_t, 13
idt_register_t, 13

kernel/core/acpi.c, 23
kernel/core/acpi.h, 24
kernel/core/apic.c, 25
kernel/core/apic.h, 26
kernel/core/cmos.c, 27
kernel/core/cmos.h, 28
kernel/core/cpu.c, 30
kernel/core/cpu.h, 30
kernel/core/gdt.c, 32
kernel/core/gdt.h, 33
kernel/core/hpet.c, 34
kernel/core/hpet.h, 35
kernel/core/idt.c, 36
kernel/core/idt.h, 38
kernel/core/isr.c, 39
kernel/core/isr_base.h, 40
kernel/core/madt.c, 41
kernel/core/madt.h, 43
kernel/core/mm.c, 44
kernel/core/mm.h, 45
kernel/core/panic.h, 47
kernel/core/pit.c, 48
kernel/core/pit.h, 49
kernel/core/smp.c, 50
kernel/core/smp.h, 51
kernel/core/timer.c, 52
kernel/core/timer.h, 54
kernel/kmain.c, 55

kernel/lib/klog.c, 56
kernel/lib/klog.h, 57
kernel/lib/kmalloc.c, 59
kernel/lib/kmalloc.h, 60
kernel/lib/lock.h, 60
kernel/lib/memutils.c, 62
kernel/lib/memutils.h, 63
kernel/lib/string.c, 64
kernel/lib/string.h, 65
kernel/lib/time.c, 65
kernel/lib/time.h, 66
kernel/proc/sched.c, 67
kernel/proc/sched.h, 68
kernel/proc/task.c, 70
kernel/proc/task.h, 71
kernel/proc/tlist.c, 73
kernel/proc/tlist.h, 74
kernel/symbols.h, 75
klog_info_t, 13
kpanic
 panic.h, 48

lock.h
 lock_lock, 61
 lock_release, 62
lock_lock
 lock.h, 61
lock_release
 lock.h, 62
lock_t, 14

madt_record_hdr_t, 14
madt_record_ioapic_t, 14
madt_record_iso_t, 15
madt_record_lapic_t, 16
madt_record_nmi_t, 16
madt_t, 17
mem_info_t, 17
metadata_t, 18

panic.h
 kpanic, 48

read_cr
 cpu.h, 32
rsdp_t, 18

smp_info_t, 18
symbol_t, 19
sys_seg_desc_t, 19

task_list_t, [20](#)
task_regs_t, [20](#)
task_t, [21](#)
timeval_t, [21](#)
timezone_t, [22](#)
tm_t, [22](#)
tss_t, [22](#)

write_cr
 cpu.h, [32](#)