OS Allstars' MPX Programmer's Manual

Generated by Doxygen 1.9.2

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 date_time Struct Reference

`#include <system.h>`

Data Fields

- int sec
- int min
- int hour
- int day_w
- int day_m
- int day_y
- int mon
- int year

3.1.1 Field Documentation

3.1.1.1 day_m

`int day_m`

3.1.1.2 day_w

`int day_w`

3.1.1.3 day_y

```
int day_y
```

3.1.1.4 hour

```
int hour
```

3.1.1.5 min

```
int min
```

3.1.1.6 mon

```
int mon
```

3.1.1.7 sec

```
int sec
```

3.1.1.8 year

```
int year
```

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

- include/system.h

3.2 footer Struct Reference

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

Collaboration diagram for footer:

Data Fields

- header head

3.2.1 Field Documentation

3.2.1.1 head

```
header head
```

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

- include/mem/heap.h

3.3 gdt_descriptor_struct Struct Reference

```
#include <tables.h>
```

Data Fields

- u16int limit
- u32int base

3.3.1 Field Documentation

3.3.1.1 base

```
u32int base
```

3.3.1.2 limit

```
u16int limit
```

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

- include/core/tables.h

3.4 gdt_entry_struct Struct Reference

`#include <tables.h>`

Data Fields

- u16int limit_low
- u16int base_low
- u8int base_mid
- u8int access
- u8int flags
- u8int base_high

3.4.1 Field Documentation

3.4.1.1 access

`u8int access`

3.4.1.2 base_high

`u8int base_high`

3.4.1.3 base_low

`u16int base_low`

3.4.1.4 base_mid

`u8int base_mid`

3.4.1.5 flags

`u8int flags`

3.4.1.6 limit_low

```
u16int limit_low
```

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

- include/core/tables.h

3.5 header Struct Reference

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

Data Fields

- int size
- int index_id

3.5.1 Field Documentation

3.5.1.1 index_id

```
int index_id
```

3.5.1.2 size

```
int size
```

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

- include/mem/heap.h

3.6 heap Struct Reference

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

Collaboration diagram for heap:

Data Fields

- index_table index
- u32int base
- u32int max_size
- u32int min_size

3.6.1 Field Documentation

3.6.1.1 base

```
u32int base
```

3.6.1.2 index

```
index_table index
```

3.6.1.3 max_size

```
u32int max_size
```

3.6.1.4 min_size

```
u32int min_size
```

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

- include/mem/heap.h

3.7 idt_entry_struct Struct Reference

```
#include <tables.h>
```

Data Fields

- u16int base_low
- u16int sselect
- u8int zero
- u8int flags
- u16int base_high

3.7.1 Field Documentation

3.7.1.1 base_high

```
u16int base_high
```

3.7.1.2 base_low

```
u16int base_low
```

3.7.1.3 flags

```
u8int flags
```

3.7.1.4 sselect

```
u16int sselect
```

3.7.1.5 zero

```
u8int zero
```

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

- include/core/tables.h

3.8 idt_struct Struct Reference

`#include <tables.h>`

Data Fields

- u16int limit
- u32int base

3.8.1 Field Documentation

3.8.1.1 base

`u32int base`

3.8.1.2 limit

`u16int limit`

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

- include/core/tables.h

3.9 index_entry Struct Reference

`#include <heap.h>`

Data Fields

- int size
- int empty
- u32int block

3.9.1 Field Documentation

3.9.1.1 block

```
u32int block
```

3.9.1.2 empty

```
int empty
```

3.9.1.3 size

```
int size
```

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

- include/mem/heap.h

3.10 index_table Struct Reference

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

Collaboration diagram for index_table:

Data Fields

- index_entry table [TABLE_SIZE]
- int id

3.10.1 Field Documentation

3.10.1.1 id

```
int id
```

3.10.1.2 table

```
index_entry table[TABLE_SIZE]
```

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

- include/mem/heap.h

3.11 page_dir Struct Reference

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

Collaboration diagram for page_dir:

Data Fields

- page_table * tables [1024]
- u32int tables_phys [1024]

3.11.1 Field Documentation

3.11.1.1 tables

```
page_table* tables[1024]
```

3.11.1.2 tables_phys

```
u32int tables_phys[1024]
```

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

- include/mem/paging.h

3.12 page_entry Struct Reference

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

Data Fields

- u32int present: 1
- u32int writeable: 1
- u32int usermode: 1
- u32int accessed: 1
- u32int dirty: 1
- u32int reserved: 7
- u32int frameaddr: 20

3.12.1 Field Documentation

3.12.1.1 accessed

```
u32int accessed
```

3.12.1.2 dirty

```
u32int dirty
```

3.12.1.3 frameaddr

```
u32int frameaddr
```

3.12.1.4 present

```
u32int present
```

3.12.1.5 reserved

```
u32int reserved
```

3.12.1.6 usermode

```
u32int usermode
```

3.12.1.7 writeable

```
u32int writeable
```

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

- include/mem/paging.h

3.13 page_table Struct Reference

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

Collaboration diagram for page_table:

Data Fields

- page_entry pages [1024]

3.13.1 Field Documentation

3.13.1.1 pages

```
page_entry pages[1024]
```

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

- include/mem/paging.h

3.14 param Struct Reference

```
#include <mpx_supt.h>
```

Data Fields

- int op_code
- int device_id
- char * buffer_ptr
- int * count_ptr

3.14.1 Field Documentation

3.14.1.1 buffer_ptr

```
char* buffer_ptr
```

3.14.1.2 count_ptr

```
int* count_ptr
```

3.14.1.3 device_id

```
int device_id
```

3.14.1.4 op_code

```
int op_code
```

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

- modules/mpx_supt.h

Chapter 4

File Documentation

4.1 include/core/asm.h File Reference

```
#include <system.h>
#include <tables.h>
```

Include dependency graph for asm.h:

4.2 include/core/interrupts.h File Reference

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

Functions

- void init_irq (void)
- void init_pic (void)

4.2.1 Function Documentation

4.2.1.1 init_irq()

```
void init_irq (
            void  )
```

4.2.1.2 init_pic()

```
void init_pic (
            void  )
```

4.3 include/core/io.h File Reference

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

Macros

- #define outb(port, data) asm volatile ("outb %%al,%%dx" : : "a" (data), "d" (port))
- #define inb(port)

4.3.1 Macro Definition Documentation

4.3.1.1 inb

```
#define inb(
              port )
```

Value:

```
({                          \
unsigned char r;                        \
asm volatile ("inb %%dx,%%al": "=a" (r): "d" (port)); \
r;                                      \
})
```

4.3.1.2 outb

```
#define outb(
              port,
              data )  asm volatile ("outb %%al,%%dx" :  :  "a" (data), "d" (port))
```

4.4 include/core/serial.h File Reference

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

Macros

- #define COM1 0x3f8
- #define COM2 0x2f8
- #define COM3 0x3e8
- #define COM4 0x2e8

Functions

- int init_serial (int device)
- int serial_println (const char *msg)
- int serial_print (const char *msg)
- int set_serial_out (int device)
- int set_serial_in (int device)
- int * polling (char *buffer, int *count)

4.4.1 Macro Definition Documentation

4.4.1.1 COM1

```
#define COM1 0x3f8
```

4.4.1.2 COM2

```
#define COM2 0x2f8
```

4.4.1.3 COM3

```
#define COM3 0x3e8
```

4.4.1.4 COM4

```
#define COM4 0x2e8
```

4.4.2 Function Documentation

4.4.2.1 init_serial()

```
int init_serial (
            int device )
```

4.4.2.2 polling()

```
int* polling (
            char * buffer,
            int * count )
```

This function is used to navigate the user interface, by taking in keyboard inputs, wrties them to the console and stores the input in a buffer

Parameters

beffer	the buffer is a pointer to the character array in the command handler. The character array stores character input from the user
count	pointer to a integer size of the buffer used in sys_req

Return values

count	point to integer size of the buffer used in sys_req

4.4.2.3 serial_print()

```
int serial_print (
            const char * msg )
```

4.4.2.4 serial_println()

```
int serial_println (
            const char * msg )
```

4.4.2.5 set_serial_in()

```
int set_serial_in (
            int device )
```

4.4.2.6 set_serial_out()

```
int set_serial_out (
            int device )
```

4.5 include/core/tables.h File Reference

```
#include "system.h"
```

Include dependency graph for tables.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct idt_entry_struct
- struct idt_struct
- struct gdt_descriptor_struct
- struct gdt_entry_struct

Functions

- struct idt_entry_struct __attribute__ ((packed)) idt_entry
- void idt_set_gate (u8int idx, u32int base, u16int sel, u8int flags)
- void gdt_init_entry (int idx, u32int base, u32int limit, u8int access, u8int flags)
- void init_idt ()
- void init_gdt ()

Variables

- u16int base_low
- u16int sselect
- u8int zero
- u8int flags
- u16int base_high
- u16int limit
- u32int base
- u16int limit_low
- u8int base_mid
- u8int access

4.5.1 Function Documentation

4.5.1.1 __attribute__()

```
struct gdt_entry_struct __attribute__ (
             (packed)  )
```

4.5.1.2 gdt_init_entry()

```
void gdt_init_entry (
             int idx,
             u32int base,
             u32int limit,
             u8int access,
             u8int flags )
```

4.5.1.3 idt_set_gate()

```
void idt_set_gate (
             u8int idx,
             u32int base,
             u16int sel,
             u8int flags )
```

4.5.1.4 init_gdt()

```
void init_gdt ( )
```

4.5.1.5 init_idt()

```
void init_idt ( )
```

4.5.2 Variable Documentation

4.5.2.1 access

```
u8int access
```

4.5.2.2 base

```
u32int base
```

4.5.2.3 base_high

```
u8int base_high
```

4.5.2.4 base_low

```
u16int base_low
```

4.5.2.5 base_mid

```
u8int base_mid
```

4.5.2.6 flags

```
u8int flags
```

4.5.2.7 limit

```
u16int limit
```

4.5.2.8 limit_low

```
u16int limit_low
```

4.5.2.9 sselect

```
u16int sselect
```

4.5.2.10 zero

```
u8int zero
```

4.6 include/mem/heap.h File Reference

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

Data Structures

- struct header
- struct footer
- struct index_entry
- struct index_table
- struct heap

Macros

- #define TABLE_SIZE 0x1000
- #define KHEAP_BASE 0xD000000
- #define KHEAP_MIN 0x10000
- #define KHEAP_SIZE 0x1000000

Functions

- u32int _kmalloc (u32int size, int align, u32int *phys_addr)
- u32int kmalloc (u32int size)
- u32int kfree ()
- void init_kheap ()
- u32int alloc (u32int size, heap *hp, int align)
- heap * make_heap (u32int base, u32int max, u32int min)

4.6.1 Macro Definition Documentation

4.6.1.1 KHEAP_BASE

```
#define KHEAP_BASE 0xD000000
```

4.6.1.2 KHEAP_MIN

```
#define KHEAP_MIN 0x10000
```

4.6.1.3 KHEAP_SIZE

```
#define KHEAP_SIZE 0x1000000
```

4.6.1.4 TABLE_SIZE

```
#define TABLE_SIZE 0x1000
```

4.6.2 Function Documentation

4.6.2.1 _kmalloc()

```
u32int _kmalloc (
            u32int size,
            int align,
            u32int * phys_addr )
```

4.6.2.2 alloc()

```
u32int alloc (
            u32int size,
            heap * hp,
            int align )
```

4.6.2.3 init_kheap()

```
void init_kheap ( )
```

4.6.2.4 kfree()

```
u32int kfree ( )
```

4.6.2.5 kmalloc()

```
u32int kmalloc (
            u32int size )
```

4.6.2.6 make_heap()

```
heap* make_heap (
            u32int base,
            u32int max,
            u32int min )
```

4.7 include/mem/paging.h File Reference

`#include <system.h>`
Include dependency graph for paging.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct page_entry
- struct page_table
- struct page_dir

Macros

- #define PAGE_SIZE 0x1000

Functions

- void set_bit (u32int addr)
- void clear_bit (u32int addr)
- u32int get_bit (u32int addr)
- u32int first_free ()
- void init_paging ()
- void load_page_dir (page_dir *new_page_dir)
- page_entry * get_page (u32int addr, page_dir *dir, int make_table)
- void new_frame (page_entry *page)

4.7.1 Macro Definition Documentation

4.7.1.1 PAGE_SIZE

```
#define PAGE_SIZE 0x1000
```

4.7.2 Function Documentation

4.7.2.1 clear_bit()

```
void clear_bit (
            u32int addr )
```

4.7.2.2 first_free()

```
u32int first_free ( )
```

4.7.2.3 get_bit()

```
u32int get_bit (
            u32int addr )
```

4.7.2.4 get_page()

```
page_entry* get_page (
            u32int addr,
            page_dir * dir,
            int make_table )
```

4.7.2.5 init_paging()

```
void init_paging ( )
```

4.7.2.6 load_page_dir()

```
void load_page_dir (
            page_dir * new_page_dir )
```

4.7.2.7 new_frame()

```
void new_frame (
            page_entry * page )
```

4.7.2.8 set_bit()

```
void set_bit (
            u32int addr )
```

4.8 include/string.h File Reference

`#include <system.h>`
Include dependency graph for string.h: This graph shows which files directly or indirectly include this file:

Functions

- int isspace (const char *c)
- void * memset (void *s, int c, size_t n)
- char * strcpy (char *s1, const char *s2)
- char * strcat (char *s1, const char *s2)
- int strlen (const char *s)
- int strcmp (const char *s1, const char *s2)
- char * strtok (char *s1, const char *s2)
- int atoi (const char *s)
- void swap (char *x, char *y)

  Swap two characters within two distinct string, created for use within itoa() Design for this function came from two websites: Title: Implement itoa() function in C Last Updated : 29 May, 2017 Availability: techiedelight.com/implement-itoa-function-in-c/ & geeksforgeeks.org/implement-itoa/.
- char * reverse (char *buffer, int length)

  Reverse the order of characters in an array, created for use within itoa() Design for this function came from two websites: Title: Implement itoa() function in C Last Updated : 29 May, 2017 Availability: techiedelight.com/implement-itoa-function-in-c/ & geeksforgeeks.org/implement-itoa/.
- char * itoa (int value, char *buffer, int base)

  Convert an integer to an ASCII string Design for this function came from two websites: Title: Implement itoa() function in C Last Updated : 29 May, 2017 Availability: techiedelight.com/implement-itoa-function-in-c/ & geeksforgeeks.org/implement-itoa/.

4.8.1 Function Documentation

4.8.1.1 atoi()

```
int atoi (
            const char * s )
```

4.8.1.2 isspace()

```
int isspace (
            const char * c )
```

4.8.1.3 itoa()

```
char* itoa (
            int value,
            char * buffer,
            int base )
```

Convert an integer to an ASCII string Design for this function came from two websites: Title: Implement itoa() function in C Last Updated : 29 May, 2017 Availability: techiedelight.com/implement-itoa-function-in-c/ & geeksforgeeks.org/implement-itoa/.

Parameters

int	value: int data type to be converted
char∗	buffer: pointer to destination for converted string
int	base: number base to convert to (2 for binary, 10 for decimal, etc.)

Return values

buffer	converted string

4.8.1.4 memset()

```
void* memset (
            void * s,
            int c,
            size_t n )
```

4.8.1.5 reverse()

```
char* reverse (
            char * buffer,
            int length )
```

Reverse the order of characters in an array, created for use within itoa() Design for this function came from two websites: Title: Implement itoa() function in C Last Updated : 29 May, 2017 Availability: techiedelight.↩ com/implement-itoa-function-in-c/ & geeksforgeeks.org/implement-itoa/.

Parameters

char	∗buffer: pointer to buffer to be reversed in order
int	length: length of buffer

Return values

buffer	buffer in reversed order

4.8.1.6 strcat()

```
char* strcat (
            char * s1,
            const char * s2 )
```

4.8.1.7 strcmp()

```
int strcmp (
            const char * s1,
            const char * s2 )
```

4.8.1.8 strcpy()

```
char* strcpy (
            char * s1,
            const char * s2 )
```

4.8.1.9 strlen()

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

4.8.1.10 strtok()

```
char* strtok (
            char * s1,
            const char * s2 )
```

4.8.1.11 swap()

```
void swap (
            char * x,
            char * y )  [inline]
```

Swap two characters within two distinct string, created for use within itoa() Design for this function came from two websites: Title: Implement itoa() function in C Last Updated : 29 May, 2017 Availability: techiedelight.↩ com/implement-itoa-function-in-c/ & geeksforgeeks.org/implement-itoa/.

Parameters

char	∗x: pointer to first character to be swapped
char	∗y: pointer to second character to be swaped

Return values

none	

4.9 include/system.h File Reference

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

Data Structures

- struct date_time

Macros

- #define NULL 0
- #define no_warn(p) if (p) while (1) break
- #define asm __asm__
- #define volatile __volatile__
- #define sti() asm volatile ("sti"::)
- #define cli() asm volatile ("cli"::)
- #define nop() asm volatile ("nop"::)
- #define hlt() asm volatile ("hlt"::)
- #define iret() asm volatile ("iret"::)
- #define GDT_CS_ID 0x01
- #define GDT_DS_ID 0x02

Typedefs

- typedef unsigned int size_t
- typedef unsigned char u8int
- typedef unsigned short u16int
- typedef unsigned long u32int

Functions

- void klogv (const char *msg)
- void kpanic (const char *msg)

4.9.1 Macro Definition Documentation

4.9.1.1 asm

```
#define asm __asm__
```

4.9.1.2 cli

```
#define cli( ) asm volatile ("cli"::)
```

4.9.1.3 GDT_CS_ID

```
#define GDT_CS_ID 0x01
```

4.9.1.4 GDT_DS_ID

```
#define GDT_DS_ID 0x02
```

4.9.1.5 hlt

```
#define hlt( ) asm volatile ("hlt"::)
```

4.9.1.6 iret

```
#define iret( ) asm volatile ("iret"::)
```

4.9.1.7 no_warn

```
#define no_warn(
              p ) if (p) while (1) break
```

4.9.1.8 nop

```
#define nop( ) asm volatile ("nop"::)
```

4.9.1.9 NULL

```
#define NULL 0
```

4.9.1.10 sti

```
#define sti( ) asm volatile ("sti"::)
```

4.9.1.11 volatile

```
#define volatile __volatile__
```

4.9.2 Typedef Documentation

4.9.2.1 size_t

```
typedef unsigned int size_t
```

4.9.2.2 u16int

```
typedef unsigned short u16int
```

4.9.2.3 u32int

```
typedef unsigned long u32int
```

4.9.2.4 u8int

```
typedef unsigned char u8int
```

4.9.3 Function Documentation

4.9.3.1 klogv()

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

4.9.3.2 kpanic()

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

4.10 kernel/core/interrupts.c File Reference

```
#include <system.h>
#include <core/io.h>
#include <core/serial.h>
#include <core/tables.h>
#include <core/interrupts.h>
```

Include dependency graph for interrupts.c:

Macros

- #define PIC1 0x20
- #define PIC2 0xA0
- #define ICW1 0x11
- #define ICW4 0x01
- #define io_wait() asm volatile ("outb $0x80")

Functions

- void divide_error ()
- void debug ()
- void nmi ()
- void breakpoint ()
- void overflow ()
- void bounds ()
- void invalid_op ()
- void device_not_available ()
- void double_fault ()
- void coprocessor_segment ()
- void invalid_tss ()
- void segment_not_present ()
- void stack_segment ()

- void general_protection ()
- void page_fault ()
- void reserved ()
- void coprocessor ()
- void rtc_isr ()
- void isr0 ()
- void do_isr ()
- void init_irq (void)
- void init_pic (void)
- void do_divide_error ()
- void do_debug ()
- void do_nmi ()
- void do_breakpoint ()
- void do_overflow ()
- void do_bounds ()
- void do_invalid_op ()
- void do_device_not_available ()
- void do_double_fault ()
- void do_coprocessor_segment ()
- void do_invalid_tss ()
- void do_segment_not_present ()
- void do_stack_segment ()
- void do_general_protection ()
- void do_page_fault ()
- void do_reserved ()
- void do_coprocessor ()

Variables

- idt_entry idt_entries [256]

4.10.1 Macro Definition Documentation

4.10.1.1 ICW1

```
#define ICW1 0x11
```

4.10.1.2 ICW4

```
#define ICW4 0x01
```

4.10.1.3 io_wait

```
#define io_wait( ) asm volatile ("outb $0x80")
```

4.10.1.4 PIC1

```
#define PIC1 0x20
```

4.10.1.5 PIC2

```
#define PIC2 0xA0
```

4.10.2 Function Documentation

4.10.2.1 bounds()

```
void bounds ( )
```

4.10.2.2 breakpoint()

```
void breakpoint ( )
```

4.10.2.3 coprocessor()

```
void coprocessor ( )
```

4.10.2.4 coprocessor_segment()

```
void coprocessor_segment ( )
```

4.10.2.5 debug()

```
void debug ( )
```

4.10.2.6 device_not_available()

```
void device_not_available ( )
```

4.10.2.7 divide_error()

```
void divide_error ( )
```

4.10.2.8 do_bounds()

```
void do_bounds ( )
```

4.10.2.9 do_breakpoint()

```
void do_breakpoint ( )
```

4.10.2.10 do_coprocessor()

```
void do_coprocessor ( )
```

4.10.2.11 do_coprocessor_segment()

```
void do_coprocessor_segment ( )
```

4.10.2.12 do_debug()

```
void do_debug ( )
```

4.10.2.13 do_device_not_available()

```
void do_device_not_available ( )
```

4.10.2.14 do_divide_error()

```
void do_divide_error ( )
```

4.10.2.15 do_double_fault()

```
void do_double_fault ( )
```

4.10.2.16 do_general_protection()

```
void do_general_protection ( )
```

4.10.2.17 do_invalid_op()

```
void do_invalid_op ( )
```

4.10.2.18 do_invalid_tss()

```
void do_invalid_tss ( )
```

4.10.2.19 do_isr()

```
void do_isr ( )
```

4.10.2.20 do_nmi()

```
void do_nmi ( )
```

4.10.2.21 do_overflow()

```
void do_overflow ( )
```

4.10.2.22 do_page_fault()

```
void do_page_fault ( )
```

4.10.2.23 do_reserved()

```
void do_reserved ( )
```

4.10.2.24 do_segment_not_present()

```
void do_segment_not_present ( )
```

4.10.2.25 do_stack_segment()

```
void do_stack_segment ( )
```

4.10.2.26 double_fault()

```
void double_fault ( )
```

4.10.2.27 general_protection()

```
void general_protection ( )
```

4.10.2.28 init_irq()

```
void init_irq (
             void  )
```

4.10.2.29 init_pic()

```
void init_pic (
            void  )
```

4.10.2.30 invalid_op()

```
void invalid_op ( )
```

4.10.2.31 invalid_tss()

```
void invalid_tss ( )
```

4.10.2.32 isr0()

```
void isr0 ( )
```

4.10.2.33 nmi()

```
void nmi ( )
```

4.10.2.34 overflow()

```
void overflow ( )
```

4.10.2.35 page_fault()

```
void page_fault ( )
```

4.10.2.36 reserved()

```
void reserved ( )
```

4.10.2.37 rtc_isr()

```
void rtc_isr ( )
```

4.10.2.38 segment_not_present()

```
void segment_not_present ( )
```

4.10.2.39 stack_segment()

```
void stack_segment ( )
```

4.10.3 Variable Documentation

4.10.3.1 idt_entries

```
idt_entry idt_entries[256]  [extern]
```

4.11 kernel/core/kmain.c File Reference

```
#include <stdint.h>
#include <string.h>
#include <system.h>
#include <core/io.h>
#include <core/serial.h>
#include <core/tables.h>
#include <core/interrupts.h>
#include <mem/heap.h>
#include <mem/paging.h>
#include "modules/mpx_supt.h"
#include "modules/cmd_handler.c"
```

Include dependency graph for kmain.c:

Functions

- void kmain (void)

4.11.1 Function Documentation

4.11.1.1 kmain()

```
void kmain (
            void  )
```

4.12 kernel/core/serial.c File Reference

```
#include <stdint.h>
#include <string.h>
#include <core/io.h>
#include <core/serial.h>
#include <modules/mpx_supt.h>
```

Include dependency graph for serial.c:

Macros

- #define NO_ERROR 0

Functions

- int init_serial (int device)
- int serial_println (const char *msg)
- int serial_print (const char *msg)
- int set_serial_out (int device)
- int set_serial_in (int device)
- int * polling (char *buffer, int *count)

Variables

- int serial_port_out = 0
- int serial_port_in = 0

4.12.1 Macro Definition Documentation

4.12.1.1 NO_ERROR

```
#define NO_ERROR 0
```

4.12.2 Function Documentation

4.12.2.1 init_serial()

```
int init_serial (
            int device )
```

4.12.2.2 polling()

```
int* polling (
            char * buffer,
            int * count )
```

This function is used to navigate the user interface, by taking in keyboard inputs, wrties them to the console and stores the input in a buffer

Parameters

beffer	the buffer is a pointer to the character array in the command handler. The character array stores character input from the user
count	pointer to a integer size of the buffer used in sys_req

Return values

count	point to integer size of the buffer used in sys_req

4.12.2.3 serial_print()

```
int serial_print (
            const char * msg )
```

4.12.2.4 serial_println()

```
int serial_println (
            const char * msg )
```

4.12.2.5 set_serial_in()

```
int set_serial_in (
            int device )
```

4.12.2.6 set_serial_out()

```
int set_serial_out (
            int device )
```

4.12.3 Variable Documentation

4.12.3.1 serial_port_in

```
int serial_port_in = 0
```

4.12.3.2 serial_port_out

```
int serial_port_out = 0
```

4.13 kernel/core/system.c File Reference

```
#include <string.h>
#include <system.h>
#include <core/serial.h>
```

Include dependency graph for system.c:

Functions

- void klogv (const char *msg)
- void kpanic (const char *msg)

4.13.1 Function Documentation

4.13.1.1 klogv()

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

4.13.1.2 kpanic()

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

4.14 kernel/core/tables.c File Reference

```
#include <string.h>
#include <core/tables.h>
```

Include dependency graph for tables.c:

Functions

- void write_gdt_ptr (u32int, size_t)
- void write_idt_ptr (u32int)
- void idt_set_gate (u8int idx, u32int base, u16int sel, u8int flags)
- void init_idt ()
- void gdt_init_entry (int idx, u32int base, u32int limit, u8int access, u8int flags)
- void init_gdt ()

Variables

- gdt_descriptor gdt_ptr
- gdt_entry gdt_entries [5]
- idt_descriptor idt_ptr
- idt_entry idt_entries [256]

4.14.1 Function Documentation

4.14.1.1 gdt_init_entry()

```
void gdt_init_entry (
            int idx,
            u32int base,
            u32int limit,
            u8int access,
            u8int flags )
```

4.14.1.2 idt_set_gate()

```
void idt_set_gate (
             u8int idx,
             u32int base,
             u16int sel,
             u8int flags )
```

4.14.1.3 init_gdt()

```
void init_gdt ( )
```

4.14.1.4 init_idt()

```
void init_idt ( )
```

4.14.1.5 write_gdt_ptr()

```
void write_gdt_ptr (
             u32int ,
             size_t  )
```

4.14.1.6 write_idt_ptr()

```
void write_idt_ptr (
             u32int  )
```

4.14.2 Variable Documentation

4.14.2.1 gdt_entries

```
gdt_entry gdt_entries[5]
```

4.14.2.2 gdt_ptr

```
gdt_descriptor gdt_ptr
```

4.14.2.3 idt_entries

```
idt_entry idt_entries[256]
```

4.14.2.4 idt_ptr

```
idt_descriptor idt_ptr
```

4.15 kernel/mem/heap.c File Reference

```
#include <system.h>
#include <string.h>
#include <core/serial.h>
#include <mem/heap.h>
#include <mem/paging.h>
```

Include dependency graph for heap.c:

Functions

- u32int _kmalloc (u32int size, int page_align, u32int *phys_addr)
- u32int kmalloc (u32int size)
- u32int alloc (u32int size, heap *h, int align)
- heap * make_heap (u32int base, u32int max, u32int min)

Variables

- heap * kheap = 0
- heap * curr_heap = 0
- page_dir * kdir
- void * end
- void _end
- void __end
- u32int phys_alloc_addr = (u32int)&end

4.15.1 Function Documentation

4.15.1.1 _kmalloc()

```
u32int _kmalloc (
            u32int size,
            int page_align,
            u32int * phys_addr )
```

4.15.1.2 alloc()

```
u32int alloc (
            u32int size,
            heap * h,
            int align )
```

4.15.1.3 kmalloc()

```
u32int kmalloc (
            u32int size )
```

4.15.1.4 make_heap()

```
heap* make_heap (
            u32int base,
            u32int max,
            u32int min )
```

4.15.2 Variable Documentation

4.15.2.1 __end

```
void __end
```

4.15.2.2 _end

```
void _end
```

4.15.2.3 curr_heap

```
heap* curr_heap = 0
```

4.15.2.4 end

```
void* end  [extern]
```

4.15.2.5 kdir

```
page_dir* kdir  [extern]
```

4.15.2.6 kheap

```
heap* kheap = 0
```

4.15.2.7 phys_alloc_addr

```
u32int phys_alloc_addr = (u32int)&end
```

4.16 kernel/mem/paging.c File Reference

```
#include <system.h>
#include <string.h>
#include "mem/heap.h"
#include "mem/paging.h"
```

Include dependency graph for paging.c:

Functions

- void set_bit (u32int addr)
- void clear_bit (u32int addr)
- u32int get_bit (u32int addr)
- u32int find_free ()
- page_entry * get_page (u32int addr, page_dir *dir, int make_table)
- void init_paging ()
- void load_page_dir (page_dir *new_dir)
- void new_frame (page_entry *page)

Variables

- u32int mem_size = 0x4000000
- u32int page_size = 0x1000
- u32int nframes
- u32int * frames
- page_dir * kdir = 0
- page_dir * cdir = 0
- u32int phys_alloc_addr
- heap * kheap

4.16.1 Function Documentation

4.16.1.1 clear_bit()

```
void clear_bit (
            u32int addr )
```

4.16.1.2 find_free()

```
u32int find_free ( )
```

4.16.1.3 get_bit()

```
u32int get_bit (
            u32int addr )
```

4.16.1.4 get_page()

```
page_entry* get_page (
            u32int addr,
            page_dir * dir,
            int make_table )
```

4.16.1.5 init_paging()

```
void init_paging ( )
```

4.16.1.6 load_page_dir()

```
void load_page_dir (
            page_dir * new_dir )
```

4.16.1.7 new_frame()

```
void new_frame (
            page_entry * page )
```

4.16.1.8 set_bit()

```
void set_bit (
            u32int addr )
```

4.16.2 Variable Documentation

4.16.2.1 cdir

```
page_dir* cdir = 0
```

4.16.2.2 frames

```
u32int* frames
```

4.16.2.3 kdir

```
page_dir* kdir = 0
```

4.16.2.4 kheap

```
heap* kheap  [extern]
```

4.16.2.5 mem_size

```
u32int mem_size = 0x4000000
```

4.16.2.6 nframes

```
u32int nframes
```

4.16.2.7 page_size

```
u32int page_size = 0x1000
```

4.16.2.8 phys_alloc_addr

```
u32int phys_alloc_addr  [extern]
```

4.17 lib/string.c File Reference

```
#include <system.h>
#include <string.h>
```

Include dependency graph for string.c:

Functions

- int strlen (const char *s)
- char * strcpy (char *s1, const char *s2)
- int atoi (const char *s)
- int strcmp (const char *s1, const char *s2)
- char * strcat (char *s1, const char *s2)
- int isspace (const char *c)
- void * memset (void *s, int c, size_t n)
- char * strtok (char *s1, const char *s2)
- void swap (char *x, char *y)

  Swap two characters within two distinct string, created for use within itoa() Design for this function came from two websites: Title: Implement itoa() function in C Last Updated : 29 May, 2017 Availability: techiedelight.com/implement-itoa-function-in-c/ & geeksforgeeks.org/implement-itoa/.
- char * reverse (char *buffer, int length)

  Reverse the order of characters in an array, created for use within itoa() Design for this function came from two websites: Title: Implement itoa() function in C Last Updated : 29 May, 2017 Availability: techiedelight.com/implement-itoa-function-in-c/ & geeksforgeeks.org/implement-itoa/.
- char * itoa (int value, char *buffer, int base)

  Convert an integer to an ASCII string Design for this function came from two websites: Title: Implement itoa() function in C Last Updated : 29 May, 2017 Availability: techiedelight.com/implement-itoa-function-in-c/ & geeksforgeeks.org/implement-itoa/.

4.17.1 Function Documentation

4.17.1.1 atoi()

```
int atoi (
            const char * s )
```

4.17.1.2 isspace()

```
int isspace (
            const char * c )
```

4.17.1.3 itoa()

```
char* itoa (
            int value,
            char * buffer,
            int base )
```

Convert an integer to an ASCII string Design for this function came from two websites: Title: Implement itoa() function in C Last Updated : 29 May, 2017 Availability: techiedelight.com/implement-itoa-function-in-c/ & geeksforgeeks.org/implement-itoa/.

Parameters

int	value: int data type to be converted
*char**	buffer: pointer to destination for converted string
int	base: number base to convert to (2 for binary, 10 for decimal, etc.)

Return values

buffer	converted string

4.17.1.4 memset()

```
void* memset (
            void * s,
            int c,
            size_t n )
```

4.17.1.5 reverse()

```
char* reverse (
            char * buffer,
            int length )
```

Reverse the order of characters in an array, created for use within itoa() Design for this function came from two websites: Title: Implement itoa() function in C Last Updated : 29 May, 2017 Availability: techiedelight.com/implement-itoa-function-in-c/ & geeksforgeeks.org/implement-itoa/.

Parameters

char	*buffer: pointer to buffer to be reversed in order
int	length: length of buffer

Return values

buffer	buffer in reversed order

4.17.1.6 strcat()

```
char* strcat (
            char * s1,
            const char * s2 )
```

4.17.1.7 strcmp()

```
int strcmp (
            const char * s1,
            const char * s2 )
```

4.17.1.8 strcpy()

```
char* strcpy (
            char * s1,
            const char * s2 )
```

4.17.1.9 strlen()

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

4.17.1.10 strtok()

```
char* strtok (
            char * s1,
            const char * s2 )
```

4.17.1.11 swap()

```
void swap (
            char * x,
            char * y )  [inline]
```

Swap two characters within two distinct string, created for use within itoa() Design for this function came from two websites: Title: Implement itoa() function in C Last Updated : 29 May, 2017 Availability: techiedelight.com/implement-itoa-function-in-c/ & geeksforgeeks.org/implement-itoa/.

Parameters

char	∗x: pointer to first character to be swapped
char	∗y: pointer to second character to be swaped

Return values

none	

4.18 modules/cmd_handler.c File Reference

```
#include <string.h>
#include <core/serial.h>
#include <core/io.h>
```

Include dependency graph for cmd_handler.c: This graph shows which files directly or indirectly include this file:

Functions

- void settime (char ∗time_buffer, int time_buffer_size)

  This function is used to set the processor RTC's current time.
- void gettime ()

  This function is used to get the processor RTC's current time and print it to the window.
- void setdate (char ∗date_buffer, int date_buffer_size)

  This function is used to set the processor RTC's current date.
- void getdate ()

  This function is used to get the processor RTC's current date and print it to the window.
- void cmd_handler ()

  This function has a loop to continuously handle specific user commands. As commands increase in quantity and complexity this function will eventually call a host of other functions to handle tasks. User commands are entered in a fashion similar to Linux command line. For example–.

4.18.1 Function Documentation

4.18.1.1 cmd_handler()

```
void cmd_handler ( )
```

This function has a loop to continuously handle specific user commands. As commands increase in quantity and complexity this function will eventually call a host of other functions to handle tasks. User commands are entered in a fashion similar to Linux command line. For example–.
»help

would be the correct way to issue to "help command". Currently implemented commands: –help –version: provides user with current version of MPX –shutdown: begins shutdown of MPX –settime: sets a user entered time to MPX registers –gettime: prints the current time, according to MPX registers –setdate: sets a user entered date to MPX registers –getdate: prints the current time, according to MPX registers

Parameters

none	

Return values

none	

4.18.1.2 getdate()

```
void getdate ( )
```

This function is used to get the processor RTC's current date and print it to the window.

Parameters

None	

Returns

None

4.18.1.3 gettime()

```
void gettime ( )
```

This function is used to get the processor RTC's current time and print it to the window.

Parameters

None	

Returns

None

4.18.1.4 setdate()

```
void setdate (
            char * date_buffer,
            int date_buffer_size )
```

This function is used to set the processor RTC's current date.

Parameters

date_buffer	Full string representation of the date taken, unparsed or changed
date_buffer_size	Size of the input string

4.18.1.5 settime()

```
void settime (
            char * time_buffer,
            int time_buffer_size )
```

This function is used to set the processor RTC's current time.

Parameters

date_buffer	Full string representation of the time taken, unparsed or changed
date_buffer_size	Size of the input string

4.19 modules/cmd_handler.h File Reference

```
#include <string.h>
#include <core/serial.h>
#include <core/io.h>
```

Include dependency graph for cmd_handler.h:

Functions

- void settime (char *time_buffer, int time_buffer_size)

  This function is used to set the processor RTC's current time.
- void gettime ()

  This function is used to get the processor RTC's current time and print it to the window.
- void setdate (char *date_buffer, int date_buffer_size)

  This function is used to set the processor RTC's current date.
- void getdate ()

  This function is used to get the processor RTC's current date and print it to the window.
- void cmd_handler ()

  This function has a loop to continuously handle specific user commands. As commands increase in quantity and complexity this function will eventually call a host of other functions to handle tasks. User commands are entered in a fashion similar to Linux command line. For example–.

4.19.1 Function Documentation

4.19.1.1 cmd_handler()

```
void cmd_handler ( )
```

This function has a loop to continuously handle specific user commands. As commands increase in quantity and complexity this function will eventually call a host of other functions to handle tasks. User commands are entered in a fashion similar to Linux command line. For example–.
```
»help
```

would be the correct way to issue to "help command". Currently implemented commands: –help –version: provides user with current version of MPX –shutdown: begins shutdown of MPX –settime: sets a user entered time to MPX registers –gettime: prints the current time, according to MPX registers –setdate: sets a user entered date to MPX registers –getdate: prints the current time, according to MPX registers

Parameters

none	

Return values

none	

4.19.1.2 getdate()

```
void getdate ( )
```

This function is used to get the processor RTC's current date and print it to the window.

Parameters

None	

Returns

None

4.19.1.3 gettime()

```
void gettime ( )
```

This function is used to get the processor RTC's current time and print it to the window.

Parameters

None	

Returns

None

4.19.1.4 setdate()

```
void setdate (
            char * date_buffer,
            int date_buffer_size )
```

This function is used to set the processor RTC's current date.

Parameters

date_buffer	Full string representation of the date taken, unparsed or changed
date_buffer_size	Size of the input string

4.19.1.5 settime()

```
void settime (
            char * time_buffer,
            int time_buffer_size )
```

This function is used to set the processor RTC's current time.

Parameters

date_buffer	Full string representation of the time taken, unparsed or changed
date_buffer_size	Size of the input string

4.20 modules/mpx_supt.c File Reference

```
#include "mpx_supt.h"
#include <mem/heap.h>
#include <string.h>
#include <core/serial.h>
#include <core/io.h>
```

Include dependency graph for mpx_supt.c:

Functions

- int sys_req (int op_code, int device_id, char *buffer_ptr, int *count_ptr)
- void mpx_init (int cur_mod)
- void sys_set_malloc (u32int(*func)(u32int))
- void sys_set_free (int(*func)(void *))
- void * sys_alloc_mem (u32int size)
- int sys_free_mem (void *ptr)
- void idle ()

Variables

- param params
- int current_module = -1
- u32int(* student_malloc)(u32int)
- int(* student_free)(void *)

4.20.1 Function Documentation

4.20.1.1 idle()

```
void idle ( )
```

Procedure..: idle Description..: The idle process Params..: None

4.20.1.2 mpx_init()

```
void mpx_init (
            int cur_mod )
```

Procedure..: mpx_init Description..: Initialize MPX support software Params..: int cur_mod (symbolic constants MODULE_R1, MODULE_R2, etc

4.20.1.3 sys_alloc_mem()

```
void* sys_alloc_mem (
            u32int size )
```

Procedure..: sys_alloc_mem Description..: Allocates a block of memory (similar to malloc) Params..: Number of bytes to allocate

4.20.1.4 sys_free_mem()

```
int sys_free_mem (
            void * ptr )
```

Procedure..: sys_free_mem Description..: Frees memory Params..: Pointer to block of memory to free

4.20.1.5 sys_req()

```
int sys_req (
            int op_code,
            int device_id,
            char * buffer_ptr,
            int * count_ptr )
```

Procedure..: sys_req Description..: Generate interrupt 60H Params..: int op_code one of (IDLE, EXIT, READ, WRITE)

4.20.1.6 sys_set_free()

```
void sys_set_free (
            int(*)(void *) func )
```

4.20.1.7 sys_set_malloc()

```
void sys_set_malloc (
            u32int(*)(u32int) func )
```

Procedure..: sys_set_malloc Description..: Sets the memory allocation function for sys_alloc_mem Params..↩ : Function pointer

4.20.2 Variable Documentation

4.20.2.1 current_module

```
int current_module = -1
```

4.20.2.2 params

```
param params
```

4.20.2.3 student_free

```
int(* student_free) (void *) (
              void *  )
```

4.20.2.4 student_malloc

```
u32int(* student_malloc) (u32int) (
              u32int  )
```

4.21 modules/mpx_supt.h File Reference

```
#include <system.h>
```

Include dependency graph for mpx_supt.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct param

Macros

- #define EXIT 0
- #define IDLE 1
- #define READ 2
- #define WRITE 3
- #define INVALID_OPERATION 4
- #define TRUE 1
- #define FALSE 0
- #define MODULE_R1 0
- #define MODULE_R2 1
- #define MODULE_R3 2
- #define MODULE_R4 4
- #define MODULE_R5 8
- #define MODULE_F 9
- #define IO_MODULE 10
- #define MEM_MODULE 11
- #define INVALID_BUFFER 1000
- #define INVALID_COUNT 2000
- #define DEFAULT_DEVICE 111
- #define COM_PORT 222

Functions

- int sys_req (int op_code, int device_id, char *buffer_ptr, int *count_ptr)
- void mpx_init (int cur_mod)
- void sys_set_malloc (u32int(*func)(u32int))
- void sys_set_free (int(*func)(void *))
- void * sys_alloc_mem (u32int size)
- int sys_free_mem (void *ptr)
- void idle ()

4.21.1 Macro Definition Documentation

4.21.1.1 COM_PORT

```
#define COM_PORT 222
```

4.21.1.2 DEFAULT_DEVICE

```
#define DEFAULT_DEVICE 111
```

4.21.1.3 EXIT

```
#define EXIT 0
```

4.21.1.4 FALSE

```
#define FALSE 0
```

4.21.1.5 IDLE

```
#define IDLE 1
```

4.21.1.6 INVALID_BUFFER

```
#define INVALID_BUFFER 1000
```

4.21.1.7 INVALID_COUNT

```
#define INVALID_COUNT 2000
```

4.21.1.8 INVALID_OPERATION

```
#define INVALID_OPERATION 4
```

4.21.1.9 IO_MODULE

```
#define IO_MODULE 10
```

4.21.1.10 MEM_MODULE

```
#define MEM_MODULE 11
```

4.21.1.11 MODULE_F

```
#define MODULE_F 9
```

4.21.1.12 MODULE_R1

```
#define MODULE_R1 0
```

4.21.1.13 MODULE_R2

```
#define MODULE_R2 1
```

4.21.1.14 MODULE_R3

```
#define MODULE_R3 2
```

4.21.1.15 MODULE_R4

```
#define MODULE_R4 4
```

4.21.1.16 MODULE_R5

```
#define MODULE_R5 8
```

4.21.1.17 READ

```
#define READ 2
```

4.21.1.18 TRUE

```
#define TRUE 1
```

4.21.1.19 WRITE

```
#define WRITE 3
```

4.21.2 Function Documentation

4.21.2.1 idle()

```
void idle ( )
```

Procedure..: idle Description..: The idle process Params..: None

4.21.2.2 mpx_init()

```
void mpx_init (
            int cur_mod )
```

Procedure..: mpx_init Description..: Initialize MPX support software Params..: int cur_mod (symbolic constants MODULE_R1, MODULE_R2, etc

4.21.2.3 sys_alloc_mem()

```
void* sys_alloc_mem (
            u32int size )
```

Procedure..: sys_alloc_mem Description..: Allocates a block of memory (similar to malloc) Params..: Number of bytes to allocate

4.21.2.4 sys_free_mem()

```
int sys_free_mem (
            void * ptr )
```

Procedure..: sys_free_mem Description..: Frees memory Params..: Pointer to block of memory to free

4.21.2.5 sys_req()

```
int sys_req (
            int op_code,
            int device_id,
            char * buffer_ptr,
            int * count_ptr )
```

Procedure..: sys_req Description..: Generate interrupt 60H Params..: int op_code one of (IDLE, EXIT, READ, WRITE)

4.21.2.6 sys_set_free()

```
void sys_set_free (
            int(*)(void *) func )
```

4.21.2.7 sys_set_malloc()

```
void sys_set_malloc (
            u32int(*)(u32int) func )
```

Procedure..: sys_set_malloc Description..: Sets the memory allocation function for sys_alloc_mem Params..↩ : Function pointer