

OS Allstars' MPX Project

Generated by Doxygen 1.9.2

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

alarm	This struct supports the alarm process	??
alarm_list	This struct stores user created alarms	??
context	??
date_time	??
footer	??
gdt_descriptor_struct	??
gdt_entry_struct	??
header	??
heap	??
idt_entry_struct	??
idt_struct	??
index_entry	??
index_table	??
page_dir	??
page_entry	??
page_table	??
param	??
pcb	??
queue	??

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

include/system.h	??
include/core/interrupts.h	??
include/core/io.h	??
include/core/serial.h	??
include/core/tables.h	??
include/mem/paging.h	??
kernel/core/kmain.c	??
kernel/core/serial.c	??
kernel/core/system.c	??
kernel/core/tables.c	??
kernel/mem/paging.c	??
modules/cmd_handler.h	??
modules/internal_procedures.c	??
modules/internal_procedures.h	??
modules/mpx_supt.c	??
modules/mpx_supt.h	??
modules/pcb_temp_commands.c	??
modules/pcb_temp_commands.h	??
modules/pcb_user_commands.c	??
modules/pcb_user_commands.h	??
modules/procsr3.c	??
modules/procsr3.h	??
modules/R4processes.c	??
modules/R4processes.h	??
modules/structs.h	??
modules/sys_call.c	??
modules/sys_call.h	??
modules/userR3Commands.c	??
modules/userR3Commands.h	??

Chapter 3

Data Structure Documentation

3.1 alarm Struct Reference

This struct supports the alarm process.

```
#include <structs.h>
```

Collaboration diagram for alarm:

Data Fields

- char [alarm_time](#) [10]
- char [alarm_msg](#) [50]
- struct [alarm](#) * [next](#)
- struct [alarm](#) * [prev](#)

3.1.1 Detailed Description

This struct supports the alarm process.

3.1.2 Field Documentation

3.1.2.1 alarm_msg

```
char alarm_msg[50]
```

3.1.2.2 alarm_time

```
char alarm_time[10]
```

3.1.2.3 next

```
struct alarm* next
```

3.1.2.4 prev

```
struct alarm* prev
```

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

- modules/[structs.h](#)

3.2 alarm_list Struct Reference

This struct stores user created alarms.

```
#include <structs.h>
```

Collaboration diagram for alarm_list:

Data Fields

- int [count](#)
- struct [alarm](#) * [head](#)
- struct [alarm](#) * [tail](#)

3.2.1 Detailed Description

This struct stores user created alarms.

3.2.2 Field Documentation

3.2.2.1 count

```
int count
```

3.2.2.2 head

```
struct alarm* head
```

3.2.2.3 tail

```
struct alarm* tail
```

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

- [modules/structs.h](#)

3.3 context Struct Reference

```
#include <structs.h>
```

Data Fields

- [u32int gs](#)
- [u32int fs](#)
- [u32int es](#)
- [u32int ds](#)
- [u32int edi](#)
- [u32int esi](#)
- [u32int ebp](#)
- [u32int esp](#)
- [u32int ebx](#)
- [u32int edx](#)
- [u32int ecx](#)
- [u32int eax](#)
- [u32int eip](#)
- [u32int cs](#)
- [u32int eflags](#)

3.3.1 Field Documentation

3.3.1.1 cs

`u32int` cs

3.3.1.2 ds

`u32int` ds

3.3.1.3 eax

`u32int` eax

3.3.1.4 ebp

`u32int` ebp

3.3.1.5 ebx

`u32int` ebx

3.3.1.6 ecx

`u32int` ecx

3.3.1.7 edi

`u32int` edi

3.3.1.8 edx

`u32int` edx

3.3.1.9 eflags

`u32int` eflags

3.3.1.10 eip

`u32int` eip

3.3.1.11 es

`u32int` es

3.3.1.12 esi

`u32int` esi

3.3.1.13 esp

`u32int` esp

3.3.1.14 fs

`u32int` fs

3.3.1.15 gs

`u32int` gs

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

- [modules/structs.h](#)

3.4 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](#)

3.4.1 Field Documentation

3.4.1.1 day_m

```
int day_m
```

3.4.1.2 day_w

```
int day_w
```

3.4.1.3 day_y

```
int day_y
```

3.4.1.4 hour

```
int hour
```

3.4.1.5 min

```
int min
```

3.4.1.6 mon

```
int mon
```

3.4.1.7 sec

```
int sec
```

3.4.1.8 year

```
int year
```

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

- [include/system.h](#)

3.5 footer Struct Reference

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

Collaboration diagram for footer:

Data Fields

- [header head](#)

3.5.1 Field Documentation

3.5.1.1 head

```
header head
```

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

- [include/mem/heap.h](#)

3.6 gdt_descriptor_struct Struct Reference

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

Data Fields

- [u16int limit](#)
- [u32int base](#)

3.6.1 Field Documentation

3.6.1.1 base

[u32int](#) base

3.6.1.2 limit

[u16int](#) limit

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

- include/core/[tables.h](#)

3.7 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](#)

3.7.1 Field Documentation

3.7.1.1 access

```
u8int access
```

3.7.1.2 base_high

```
u8int base_high
```

3.7.1.3 base_low

```
u16int base_low
```

3.7.1.4 base_mid

```
u8int base_mid
```

3.7.1.5 flags

```
u8int flags
```

3.7.1.6 limit_low

```
u16int limit_low
```

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

- [include/core/tables.h](#)

3.8 header Struct Reference

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

Data Fields

- int [size](#)
- int [index_id](#)

3.8.1 Field Documentation

3.8.1.1 `index_id`

```
int index_id
```

3.8.1.2 `size`

```
int size
```

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

- [include/mem/heap.h](#)

3.9 heap Struct Reference

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

Collaboration diagram for heap:

Data Fields

- [index_table](#) index
- [u32int](#) base
- [u32int](#) max_size
- [u32int](#) min_size

3.9.1 Field Documentation

3.9.1.1 `base`

```
u32int base
```


3.9.1.2 index

`index_table` index

3.9.1.3 max_size

`u32int` max_size

3.9.1.4 min_size

`u32int` min_size

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

- `include/mem/heap.h`

3.10 idt_entry_struct Struct Reference

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

Data Fields

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

3.10.1 Field Documentation

3.10.1.1 base_high

`u16int` base_high

3.10.1.2 base_low

`u16int` base_low

3.10.1.3 flags

`u8int` flags

3.10.1.4 sselect

`u16int` sselect

3.10.1.5 zero

`u8int` zero

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

- include/core/[tables.h](#)

3.11 idt_struct Struct Reference

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

Data Fields

- `u16int` limit
- `u32int` base

3.11.1 Field Documentation

3.11.1.1 base

`u32int` base

3.11.1.2 limit

```
u16int limit
```

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

- [include/core/tables.h](#)

3.12 index_entry Struct Reference

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

Data Fields

- int [size](#)
- int [empty](#)
- u32int [block](#)

3.12.1 Field Documentation

3.12.1.1 block

```
u32int block
```

3.12.1.2 empty

```
int empty
```

3.12.1.3 size

```
int size
```

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

- [include/mem/heap.h](#)

3.13 index_table Struct Reference

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

Collaboration diagram for index_table:

Data Fields

- [index_entry table](#) [TABLE_SIZE]
- [int id](#)

3.13.1 Field Documentation

3.13.1.1 id

```
int id
```

3.13.1.2 table

```
index\_entry table[TABLE_SIZE]
```

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

- [include/mem/heap.h](#)

3.14 page_dir Struct Reference

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

Collaboration diagram for page_dir:

Data Fields

- [page_table * tables](#) [1024]
- [u32int tables_phys](#) [1024]

3.14.1 Field Documentation

3.14.1.1 tables

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

3.14.1.2 tables_phys

```
u32int tables_phys[1024]
```

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

- include/mem/paging.h

3.15 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

3.15.1 Field Documentation

3.15.1.1 accessed

```
u32int accessed
```

3.15.1.2 dirty

```
u32int dirty
```

3.15.1.3 frameaddr

`u32int` frameaddr

3.15.1.4 present

`u32int` present

3.15.1.5 reserved

`u32int` reserved

3.15.1.6 usermode

`u32int` usermode

3.15.1.7 writeable

`u32int` writeable

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

- `include/mem/paging.h`

3.16 page_table Struct Reference

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

Collaboration diagram for `page_table`:

Data Fields

- `page_entry` `pages` [1024]

3.16.1 Field Documentation

3.16.1.1 pages

```
page_entry pages[1024]
```

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

- [include/mem/paging.h](#)

3.17 param Struct Reference

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

Data Fields

- int [op_code](#)
- int [device_id](#)
- char * [buffer_ptr](#)
- int * [count_ptr](#)

3.17.1 Field Documentation

3.17.1.1 buffer_ptr

```
char* buffer_ptr
```

3.17.1.2 count_ptr

```
int* count_ptr
```

3.17.1.3 device_id

```
int device_id
```

3.17.1.4 op_code

```
int op_code
```

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

- [modules/mpx_supt.h](#)

3.18 pcb Struct Reference

```
#include <structs.h>
```

Collaboration diagram for pcb:

Data Fields

- char [name](#) [10]
- int [class](#)
- int [priority](#)
- int [state](#)
- unsigned char [stack](#) [2048]
- unsigned char * [top](#)
- unsigned char * [base](#)
- struct [pcb](#) * [next](#)
- struct [pcb](#) * [prev](#)

3.18.1 Field Documentation

3.18.1.1 base

```
unsigned char* base
```

3.18.1.2 class

```
int class
```

3.18.1.3 name

```
char name[10]
```


3.18.1.4 next

```
struct pcb* next
```

3.18.1.5 prev

```
struct pcb* prev
```

3.18.1.6 priority

```
int priority
```

3.18.1.7 stack

```
unsigned char stack[2048]
```

3.18.1.8 state

```
int state
```

3.18.1.9 top

```
unsigned char* top
```

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

- [modules/structs.h](#)

3.19 queue Struct Reference

```
#include <structs.h>
```

Collaboration diagram for queue:

Data Fields

- int [count](#)
- struct [pcb](#) * [head](#)
- struct [pcb](#) * [tail](#)

3.19.1 Field Documentation

3.19.1.1 count

```
int count
```

3.19.1.2 head

```
struct pcb* head
```

3.19.1.3 tail

```
struct pcb* tail
```

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

- modules/[structs.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

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

Return values

<i>count</i>	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

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

Return values

<i>buffer</i>	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

<i>char</i>	*buffer: pointer to buffer to be reversed in order
<i>int</i>	length: length of buffer

Return values

<i>buffer</i>	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

<i>char</i>	*x: pointer to first character to be swapped
<i>char</i>	*y: pointer to second character to be swapped

Return values

<i>none</i>	
-------------	--

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 [sys_call_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 ( )
```

4.10.2.40 sys_call_isr()

```
void sys_call_isr ( )
```

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.h"
#include "modules/structs.h"
#include "modules/internal_procedures.h"
#include "modules/pcb_user_commands.h"
#include "modules/R4processes.h"
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

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

Return values

<i>count</i>	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

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

Return values

<i>buffer</i>	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

<i>char</i>	*buffer: pointer to buffer to be reversed in order
<i>int</i>	length: length of buffer

Return values

<i>buffer</i>	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

<i>char</i>	*x: pointer to first character to be swapped
<i>char</i>	*y: pointer to second character to be swapped

Return values

<i>none</i>	
-------------	--

4.18 modules/cmd_handler.c File Reference

```
#include <string.h>
#include <core/serial.h>
#include <core/io.h>
#include "mpx_supt.h"
#include "cmd_handler.h"
#include "pcb_temp_commands.h"
#include "pcb_user_commands.h"
#include "userR3Commands.h"
#include "internal_procedures.h"
#include "structs.h"
#include "R4processes.h"
```

Include dependency graph for cmd_handler.c:

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 [optional_cmd_handler](#) (char *cmd_buffer)
This function is a supplementary function to [cmd_handler\(\)](#) that specifically handles commands with user input and optional clauses. Splits cmd_buffer into various tokens.
- void [help](#) ()
This function provides functionality for the help user command.
- 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–.

Variables

- int [buffer_size](#) = 99

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

<i>none</i>	
-------------	--

Return values

<i>none</i>	
-------------	--

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

<i>None</i>	
-------------	--

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

<i>None</i>	
-------------	--

Returns

None

4.18.1.4 help()

```
void help ( )
```

This function provides functionality for the help user command.

Parameters

<i>none</i>	
-------------	--

Return values

<i>none</i>	
-------------	--

4.18.1.5 optional_cmd_handler()

```
void optional_cmd_handler (
    char * cmd_buffer )
```

This function is a supplementary function to [cmd_handler\(\)](#) that specifically handles commands with user input and optional clauses. Splits cmd_buffer into various tokens.

Parameters

<i>cmd_buffer</i>	the buffer that is passed from cmd_buffer() to this function
-------------------	--

Return values

<i>none</i>	
-------------	--

4.18.1.6 setdate()

```
void setdate (
```

```
char * date_buffer,  
int date_buffer_size )
```

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

Parameters

<i>date_buffer</i>	Full string representation of the date taken, unparsed or changed
<i>date_buffer_size</i>	Size of the input string

4.18.1.7 settime()

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

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

Parameters

<i>date_buffer</i>	Full string representation of the time taken, unparsed or changed
<i>date_buffer_size</i>	Size of the input string

4.18.2 Variable Documentation

4.18.2.1 buffer_size

```
int buffer_size = 99
```

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: 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 [optional_cmd_handler](#) (char *cmd_buffer)
This function is a supplementary function to [cmd_handler\(\)](#) that specifically handles commands with user input and optional clauses. Splits cmd_buffer into various tokens.
- void [help](#) ()
This function provides functionality for the help user command.
- 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

<i>None</i>	
-------------	--

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

<i>None</i>	
-------------	--

Returns

None

4.19.1.4 help()

```
void help ( )
```

This function provides functionality for the help user command.

Parameters

<i>none</i>	
-------------	--

Return values

<i>none</i>	
-------------	--

4.19.1.5 optional_cmd_handler()

```
void optional_cmd_handler (
    char * cmd_buffer )
```

This function is a supplementary function to [cmd_handler\(\)](#) that specifically handles commands with user input and optional clauses. Splits cmd_buffer into various tokens.

Parameters

<i>cmd_buffer</i>	the buffer that is passed from cmd_buffer() to this function
-------------------	--

Return values

<i>none</i>	
-------------	--

4.19.1.6 setdate()

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

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

Parameters

<i>date_buffer</i>	Full string representation of the date taken, unparsed or changed
<i>date_buffer_size</i>	Size of the input string

4.19.1.7 settime()

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

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

Parameters

<i>date_buffer</i>	Full string representation of the time taken, unparsed or changed
<i>date_buffer_size</i>	Size of the input string

4.20 modules/internal_procedures.c File Reference

```
#include "mpx_supt.h"
#include "structs.h"
#include <string.h>
#include <core/serial.h>
Include dependency graph for internal_procedures.c:
```

Functions

- struct `pcb` * `AllocatePCB` ()
- struct `pcb` * `FindPCB` (char *processName)
- void `FreePCB` (struct `pcb` *PCB)
- void `InsertPCB` (struct `pcb` *PCB)
- void `RemovePCB` (struct `pcb` *PCB)
- struct `pcb` * `SetupPCB` (char *processName, int class, int priority)

Variables

- struct `queue ready_suspended`
- struct `queue ready_not_suspended`
- struct `queue blocked_suspended`
- struct `queue blocked_not_suspended`

4.20.1 Function Documentation

4.20.1.1 AllocatePCB()

```
struct pcb* AllocatePCB ( )
```

This function is used to allocate memory for a `pcb` and initializes the stack to null

Return values

<code>pcb*</code>	returns a <code>pcb</code> pointer
-------------------	------------------------------------

4.20.1.2 FindPCB()

```
struct pcb* FindPCB (
    char * processName )
```

This function is used to search through the 4 queues to find a specific `pcb`

Parameters

<i>processName</i>	The name of the process is passed in as a pointer
--------------------	---

Return values

<i>pcb*</i>	returns a pcb pointer
-------------	-----------------------

4.20.1.3 FreePCB()

```
void FreePCB (
    struct pcb * PCB )
```

This function is used to free a pcb from memory Success is printed if the command is successful if an the pcb is not freed Error is printed

Parameters

<i>PCB</i>	the functions takes in a pcb pointer
------------	--------------------------------------

4.20.1.4 InsertPCB()

```
void InsertPCB (
    struct pcb * PCB )
```

This function is used to insert a pcb into its correct queue

Parameters

<i>PCB</i>	pcb pointer
------------	-------------

4.20.1.5 RemovePCB()

```
void RemovePCB (
    struct pcb * PCB )
```

This function is used to remove a pcb from a queue, Success is printed if th epcb is removed Error is printed if there was an issues removing the pcb

Parameters

<i>PCB</i>	a pointer to a specific pcb
------------	-----------------------------

4.20.1.6 SetupPCB()

```
struct pcb* SetupPCB (  
    char * processName,  
    int class,  
    int priority )
```

This function is used to place a pcb in the memory that has been allocated for it as well as necessary initialization. Inserts into ready not suspended queue

Parameters

<i>processName</i>	a character pointer to what the user would like the pcb to be called
<i>class</i>	an integer indicating whether the pcb is an application or system process
<i>priority</i>	an integer indicating the priority of the pcb

Return values

<i>count</i>	pointer to the pcb that has just been allocated to memory and initialized
--------------	---

4.20.2 Variable Documentation

4.20.2.1 blocked_not_suspended

```
struct queue blocked_not_suspended
```

4.20.2.2 blocked_suspended

```
struct queue blocked_suspended
```

4.20.2.3 ready_not_suspended

```
struct queue ready_not_suspended
```

4.20.2.4 ready_suspended

```
struct queue ready_suspended
```

4.21 modules/internal_procedures.h File Reference

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

Functions

- struct `pcb` * `AllocatePCB` ()
- struct `pcb` * `FindPCB` (char *processName)
- void `FreePCB` (struct `pcb` *PCB)
- void `InsertPCB` ()
- void `RemovePCB` (struct `pcb` *PCB)
- struct `pcb` * `SetupPCB` (char *processName, int class, int priority)

Variables

- struct `queue` `ready_suspended`
- struct `queue` `ready_not_suspended`
- struct `queue` `blocked_suspended`
- struct `queue` `blocked_not_suspended`

4.21.1 Function Documentation

4.21.1.1 AllocatePCB()

```
struct pcb* AllocatePCB ( )
```

This function is used to allocate memory for a pcb and initializes the stack to null

Return values

<code>pcb*</code>	returns a pcb pointer
-------------------	-----------------------

4.21.1.2 FindPCB()

```
struct pcb* FindPCB (
    char * processName )
```

This function is used to search through the 4 queues to find a specific pcb

Parameters

<i>processName</i>	The name of the process is passed in as a pointer
--------------------	---

Return values

<i>pcb*</i>	returns a pcb pointer
-------------	-----------------------

4.21.1.3 FreePCB()

```
void FreePCB (
    struct pcb * PCB )
```

This function is used to free a pcb from memory Success is printed if the command is successful if an the pcb is not freed Error is printed

Parameters

<i>PCB</i>	the functions takes in a pcb pointer
------------	--------------------------------------

4.21.1.4 InsertPCB()

```
void InsertPCB ( )
```

4.21.1.5 RemovePCB()

```
void RemovePCB (
    struct pcb * PCB )
```

This function is used to remove a pcb from a queue, Success is printed if th epcb is removed Error is printed if there was an issues removing the pcb

Parameters

<i>PCB</i>	a pointer to a specific pcb
------------	-----------------------------

4.21.1.6 SetupPCB()

```
struct pcb* SetupPCB (
```

```
char * processName,  
int class,  
int priority )
```

This function is used to place a pcb in the memory that has been allocated for it as well as necessary initialization. Inserts into ready not suspended queue

Parameters

<i>processName</i>	a character pointer to what the user would like the pcb to be called
<i>class</i>	an integer indicating whether the pcb is an application or system process
<i>priority</i>	an integer indicating the priority of the pcb

Return values

<i>count</i>	pointer to the pcb that has just been allocated to memory and initialized
--------------	---

4.21.2 Variable Documentation

4.21.2.1 blocked_not_suspended

```
struct queue blocked_not_suspended [extern]
```

4.21.2.2 blocked_suspended

```
struct queue blocked_suspended [extern]
```

4.21.2.3 ready_not_suspended

```
struct queue ready_not_suspended [extern]
```

4.21.2.4 ready_suspended

```
struct queue ready_suspended [extern]
```

4.22 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](#) ()
- void [infinite_proc](#) ()

This process initiates a identical process to [idle\(\)](#), but is not a system process, and can be deleted if it has already been suspended.

Variables

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

4.22.1 Function Documentation

4.22.1.1 [idle\(\)](#)

```
void idle ( )
```

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

4.22.1.2 [infinite_proc\(\)](#)

```
void infinite_proc ( )
```

This process initiates a identical process to [idle\(\)](#), but is not a system process, and can be deleted if it has already been suspended.

Parameters

<i>None</i>	
-------------	--

Return values

<i>None</i>	
-------------	--

4.22.1.3 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

4.22.1.4 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

4.22.1.5 sys_free_mem()

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

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

4.22.1.6 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)

4.22.1.7 sys_set_free()

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

4.22.1.8 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

4.22.2 Variable Documentation

4.22.2.1 current_module

```
int current_module = -1
```

4.22.2.2 params

```
param params
```

4.22.2.3 student_free

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

4.22.2.4 student_malloc

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

4.23 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 ()`
- `void infinite_proc ()`

This process initiates a identical process to `idle()`, but is not a system process, and can be deleted if it has already been suspended.

4.23.1 Macro Definition Documentation

4.23.1.1 COM_PORT

```
#define COM_PORT 222
```

4.23.1.2 DEFAULT_DEVICE

```
#define DEFAULT_DEVICE 111
```

4.23.1.3 EXIT

```
#define EXIT 0
```

4.23.1.4 FALSE

```
#define FALSE 0
```

4.23.1.5 IDLE

```
#define IDLE 1
```

4.23.1.6 INVALID_BUFFER

```
#define INVALID_BUFFER 1000
```

4.23.1.7 INVALID_COUNT

```
#define INVALID_COUNT 2000
```

4.23.1.8 INVALID_OPERATION

```
#define INVALID_OPERATION 4
```

4.23.1.9 IO_MODULE

```
#define IO_MODULE 10
```

4.23.1.10 MEM_MODULE

```
#define MEM_MODULE 11
```

4.23.1.11 MODULE_F

```
#define MODULE_F 9
```

4.23.1.12 MODULE_R1

```
#define MODULE_R1 0
```

4.23.1.13 MODULE_R2

```
#define MODULE_R2 1
```

4.23.1.14 MODULE_R3

```
#define MODULE_R3 2
```

4.23.1.15 MODULE_R4

```
#define MODULE_R4 4
```

4.23.1.16 MODULE_R5

```
#define MODULE_R5 8
```

4.23.1.17 READ

```
#define READ 2
```

4.23.1.18 TRUE

```
#define TRUE 1
```

4.23.1.19 WRITE

```
#define WRITE 3
```

4.23.2 Function Documentation

4.23.2.1 idle()

```
void idle ( )
```

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

4.23.2.2 infinite_proc()

```
void infinite_proc ( )
```

This process initiates a identical process to [idle\(\)](#), but is not a system process, and can be deleted if it has already been suspended.

Parameters

None	
------	--

Return values

None	
------	--

4.23.2.3 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

4.23.2.4 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

4.23.2.5 sys_free_mem()

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

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

4.23.2.6 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)

4.23.2.7 sys_set_free()

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

4.23.2.8 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

4.24 modules/pcb_temp_commands.c File Reference

```
#include "internal_procedures.h"
#include "structs.h"
#include "mpx_supt.h"
#include <string.h>
Include dependency graph for pcb_temp_commands.c:
```

Functions

- void [CreatePCB](#) (char *processName, int class, int priority)
This function will create a new PCB by calling the internal function SetupPCB.
- void [DeletePCB](#) (char *processName)
This function will delete a PCB from the queue by calling the internal function RemovePCB.
- void [BlockPCB](#) (char *processName)
This function will remove the PCB from a ready queue and add it to a blocked queue.
- void [UnblockPCB](#) (char *processName)
his function will remove the PCB from a blocked queue and add it to a ready queue

4.24.1 Function Documentation

4.24.1.1 BlockPCB()

```
void BlockPCB (
    char * processName )
```

This function will remove the PCB from a ready queue and add it to a blocked queue.

Parameters

<i>processName</i>	full string representation of the desired process name
--------------------	--

4.24.1.2 CreatePCB()

```
void CreatePCB (
    char * processName,
    int class,
    int priority )
```

This function will create a new PCB by calling the internal function SetupPCB.

Parameters

<i>processName</i>	full string representation of the desired process name
<i>class</i>	identification of the process as either a application or system process
<i>priority</i>	the priority level of the new process for the order it is added to the process queues

4.24.1.3 DeletePCB()

```
void DeletePCB (
    char * processName )
```

This function will delete a PCB from the queue by calling the internal function RemovePCB.

Parameters

<i>processName</i>	full string representation of the desired process name
--------------------	--

4.24.1.4 UnblockPCB()

```
void UnblockPCB (
    char * processName )
```

this function will remove the PCB from a blocked queue and add it to a ready queue

Parameters

<i>processName</i>	full string representation of the desired process name
--------------------	--

4.25 modules/pcb_temp_commands.h File Reference

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

Functions

- void [CreatePCB](#) (char *processName, int class, int priority)
This function will create a new PCB by calling the internal function SetupPCB.
- void [DeletePCB](#) (char *processName)
This function will delete a PCB from the queue by calling the internal function RemovePCB.
- void [BlockPCB](#) (char *processName)
This function will remove the PCB from a ready queue and add it to a blocked queue.
- void [UnblockPCB](#) (char *processName)
this function will remove the PCB from a blocked queue and add it to a ready queue

4.25.1 Function Documentation

4.25.1.1 BlockPCB()

```
void BlockPCB (
    char * processName )
```

This function will remove the PCB from a ready queue and add it to a blocked queue.

Parameters

<i>processName</i>	full string representation of the desired process name
--------------------	--

4.25.1.2 CreatePCB()

```
void CreatePCB (
    char * processName,
    int class,
    int priority )
```

This function will create a new PCB by calling the internal function SetupPCB.

Parameters

<i>processName</i>	full string representation of the desired process name
<i>class</i>	identification of the process as either a application or system process
<i>priority</i>	the priority level of the new process for the order it is added to the process queues

4.25.1.3 DeletePCB()

```
void DeletePCB (
    char * processName )
```

This function will delete a PCB from the queue by calling the internal function RemovePCB.

Parameters

<i>processName</i>	full string representation of the desired process name
--------------------	--

4.25.1.4 UnblockPCB()

```
void UnblockPCB (
    char * processName )
```

his function will remove the PCB from a blocked queue and add it to a ready queue

Parameters

<i>processName</i>	full string representation of the desired process name
--------------------	--

4.26 modules/pcb_user_commands.c File Reference

```
#include <string.h>
#include "internal_procedures.h"
#include "mpx_supt.h"
```

```
#include "structs.h"
```

Include dependency graph for pcb_user_commands.c:

Functions

- void [SuspendPCB](#) (char *processName)
This function changes the state of a user selected PCB to suspended and inserts it into the correct queue.
- void [ResumePCB](#) (char *processName)
This function changes the state of a user selected PCB to unsuspended and inserts it into the correct queue.
- void [SetPCBPRIORITY](#) (char *processName, int priority)
This function displays a user selected PCB to the terminal.
- void [ShowPCB](#) (char *processName)
This function displays a user selected PCB to the terminal.
- void [ShowReady](#) ()
This function displays all currently ready PCBs.
- void [ShowBlocked](#) ()
This function displays all currently blocked PCBs.
- void [ShowAll](#) ()
This function combines the [ShowReady\(\)](#) function and the [ShowBlocked\(\)](#) function to display all existing PCBs.

Variables

- int [buffer_length](#) = 99
- char [input](#) [1]

4.26.1 Function Documentation

4.26.1.1 ResumePCB()

```
void ResumePCB (
    char * processName )
```

This function changes the state of a user selected PCB to unsuspended and inserts it into the correct queue.

Parameters

processName	name of PCB to alter
-----------------------------	----------------------

Return values

none	
----------------------	--

4.26.1.2 SetPCBPriority()

```
void SetPCBPriority (
    char * processName,
    int priority )
```

This function displays a user selected PCB to the terminal.

Parameters

<i>processName</i>	name of PCB to alter
<i>priority</i>	new value to set as PCB priority

Return values

<i>none</i>	
-------------	--

4.26.1.3 ShowAll()

```
void ShowAll ( )
```

This function combines the [ShowReady\(\)](#) function and the [ShowBlocked\(\)](#) function to display all existing PCBs.

Parameters

<i>none</i>	
-------------	--

Return values

<i>none</i>	
-------------	--

4.26.1.4 ShowBlocked()

```
void ShowBlocked ( )
```

This function displays all currently blocked PCBs.

Parameters

<i>none</i>	
-------------	--

Return values

<i>none</i>	
-------------	--

4.26.1.5 ShowPCB()

```
void ShowPCB (
    char * processName )
```

This function displays a user selected PCB to the terminal.

Parameters

<i>processName</i>	name of PCB to display
--------------------	------------------------

Return values

<i>none</i>	
-------------	--

4.26.1.6 ShowReady()

```
void ShowReady ( )
```

This function displays all currently ready PCBs.

Parameters

<i>none</i>	
-------------	--

Return values

<i>none</i>	
-------------	--

4.26.1.7 SuspendPCB()

```
void SuspendPCB (
    char * processName )
```

This function changes the state of a user selected PCB to suspended and inserts it into the correct queue.

Parameters

<i>processName</i>	name of PCB to alter
--------------------	----------------------

Return values

<i>none</i>	
-------------	--

4.26.2 Variable Documentation

4.26.2.1 buffer_length

```
int buffer_length = 99
```

4.26.2.2 input

```
char input[1]
```

4.27 modules/pcb_user_commands.h File Reference

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

Functions

- void [SuspendPCB](#) (char *processName)
This function changes the state of a user selected PCB to suspended and inserts it into the correct queue.
- void [ResumePCB](#) (char *processName)
This function changes the state of a user selected PCB to unsuspended and inserts it into the correct queue.
- void [SetPCBPRIORITY](#) (char *processName, int priority)
This function displays a user selected PCB to the terminal.
- void [ShowPCB](#) (char *processName)
This function displays a user selected PCB to the terminal.
- void [ShowReady](#) ()
This function displays all currently ready PCBs.
- void [ShowBlocked](#) ()
This function displays all currently blocked PCBs.
- void [ShowAll](#) ()
This function combines the [ShowReady\(\)](#) function and the [ShowBlocked\(\)](#) function to display all existing PCBs.

4.27.1 Function Documentation

4.27.1.1 ResumePCB()

```
void ResumePCB (
    char * processName )
```

This function changes the state of a user selected PCB to unsuspended and inserts it into the correct queue.

Parameters

<i>processName</i>	name of PCB to alter
--------------------	----------------------

Return values

<i>none</i>	
-------------	--

4.27.1.2 SetPCBPRIORITY()

```
void SetPCBPRIORITY (
    char * processName,
    int priority )
```

This function displays a user selected PCB to the terminal.

Parameters

<i>processName</i>	name of PCB to alter
<i>priority</i>	new value to set as PCB priority

Return values

<i>none</i>	
-------------	--

4.27.1.3 ShowAll()

```
void ShowAll ( )
```

This function combines the [ShowReady\(\)](#) function and the [ShowBlocked\(\)](#) function to display all existing PCBs.

Parameters

<i>none</i>	
-------------	--

Return values

<i>none</i>	
-------------	--

4.27.1.4 ShowBlocked()

```
void ShowBlocked ( )
```

This function displays all currently blocked PCBs.

Parameters

<i>none</i>	
-------------	--

Return values

<i>none</i>	
-------------	--

4.27.1.5 ShowPCB()

```
void ShowPCB (
    char * processName )
```

This function displays a user selected PCB to the terminal.

Parameters

<i>processName</i>	name of PCB to display
--------------------	------------------------

Return values

<i>none</i>	
-------------	--

4.27.1.6 ShowReady()

```
void ShowReady ( )
```

This function displays all currently ready PCBs.

Parameters

<i>none</i>	
-------------	--

Return values

<i>none</i>	
-------------	--

4.27.1.7 SuspendPCB()

```
void SuspendPCB (
    char * processName )
```

This function changes the state of a user selected PCB to suspended and inserts it into the correct queue.

Parameters

<i>processName</i>	name of PCB to alter
--------------------	----------------------

Return values

<i>none</i>	
-------------	--

4.28 modules/procsr3.c File Reference

```
#include <system.h>
#include <core/serial.h>
#include "mpx_supt.h"
#include "procsr3.h"
Include dependency graph for procsr3.c:
```

Macros

- #define [RC_1](#) 1
- #define [RC_2](#) 2
- #define [RC_3](#) 3
- #define [RC_4](#) 4
- #define [RC_5](#) 5

Functions

- void [proc1](#) ()
- void [proc2](#) ()
- void [proc3](#) ()
- void [proc4](#) ()
- void [proc5](#) ()

Variables

- char * `msg1` = "proc1 dispatched"
- char * `msg2` = "proc2 dispatched"
- char * `msg3` = "proc3 dispatched"
- char * `msg4` = "proc4 dispatched"
- char * `msg5` = "proc5 dispatched"
- int `msgSize` = 17
- char * `er1` = "proc1 ran after it was terminated"
- char * `er2` = "proc2 ran after it was terminated"
- char * `er3` = "proc3 ran after it was terminated"
- char * `er4` = "proc4 ran after it was terminated"
- char * `er5` = "proc5 ran after it was terminated"
- int `erSize` = 34

4.28.1 Macro Definition Documentation

4.28.1.1 RC_1

```
#define RC_1 1
```

4.28.1.2 RC_2

```
#define RC_2 2
```

4.28.1.3 RC_3

```
#define RC_3 3
```

4.28.1.4 RC_4

```
#define RC_4 4
```

4.28.1.5 RC_5

```
#define RC_5 5
```

4.28.2 Function Documentation

4.28.2.1 proc1()

```
void proc1 ( )
```

4.28.2.2 proc2()

```
void proc2 ( )
```

4.28.2.3 proc3()

```
void proc3 ( )
```

4.28.2.4 proc4()

```
void proc4 ( )
```

4.28.2.5 proc5()

```
void proc5 ( )
```

4.28.3 Variable Documentation

4.28.3.1 er1

```
char* er1 = "proc1 ran after it was terminated"
```

4.28.3.2 er2

```
char* er2 = "proc2 ran after it was terminated"
```

4.28.3.3 er3

```
char* er3 = "proc3 ran after it was terminated"
```

4.28.3.4 er4

```
char* er4 = "proc4 ran after it was terminated"
```

4.28.3.5 er5

```
char* er5 = "proc5 ran after it was terminated"
```

4.28.3.6 erSize

```
int erSize = 34
```

4.28.3.7 msg1

```
char* msg1 = "proc1 dispatched"
```

4.28.3.8 msg2

```
char* msg2 = "proc2 dispatched"
```

4.28.3.9 msg3

```
char* msg3 = "proc3 dispatched"
```

4.28.3.10 msg4

```
char* msg4 = "proc4 dispatched"
```

4.28.3.11 msg5

```
char* msg5 = "proc5 dispatched"
```

4.28.3.12 msgSize

```
int msgSize = 17
```

4.29 modules/procsr3.h File Reference

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

Macros

- `#define _PROCSR3_H` value

Functions

- void `proc1` ()
- void `proc2` ()
- void `proc3` ()
- void `proc4` ()
- void `proc5` ()

4.29.1 Macro Definition Documentation

4.29.1.1 _PROCSR3_H

```
#define _PROCSR3_H value
```

4.29.2 Function Documentation

4.29.2.1 `proc1()`

```
void proc1 ( )
```

4.29.2.2 `proc2()`

```
void proc2 ( )
```

4.29.2.3 `proc3()`

```
void proc3 ( )
```

4.29.2.4 `proc4()`

```
void proc4 ( )
```

4.29.2.5 `proc5()`

```
void proc5 ( )
```

4.30 `modules/R4processes.c` File Reference

```
#include "structs.h"  
#include "userR3Commands.h"  
#include "procsr3.h"  
#include "internal_procedures.h"  
#include "mpx_supt.h"  
#include <string.h>  
#include <core/io.h>
```

Include dependency graph for `R4processes.c`:

4.31 `modules/R4processes.h` File Reference

```
#include "structs.h"  
#include "userR3Commands.h"  
#include "procsr3.h"  
#include "internal_procedures.h"  
#include "mpx_supt.h"  
#include <string.h>  
#include <core/serial.h>  
#include <core/io.h>
```

Include dependency graph for `R4processes.h`: This graph shows which files directly or indirectly include this file:

Functions

- void [infinite_proc](#) ()
This process initiates a identical process to [idle\(\)](#), but is not a system process, and can be deleted if it has already been suspended.
- void [add_alarm](#) (char *alarm_time, char *alarm_msg)
This function add an alarm into a list for the system to keep track of and display a message at the specified time.
- void [alarm_proc](#) ()
This function has the functionality for the alarm, will display and exit the process when the alarm time comes.

Variables

- struct [alarm_list](#) alarms

4.31.1 Function Documentation

4.31.1.1 add_alarm()

```
void add_alarm (
    char * alarm_time,
    char * alarm_msg )
```

This function add an alarm into a list for the system to keep track of and display a message at the specified time.

Parameters

<i>alarm_time</i>	the time the user specifies the alarm to go off
<i>alarm_msg</i>	message that the user specifies that will be displayed at the alarm

Return values

<i>none</i>	
-------------	--

4.31.1.2 alarm_proc()

```
void alarm_proc ( )
```

This function has the functionality for the alarm, will display and exit the process when the alarm time comes.

Parameters

<i>none</i>	
-------------	--

Return values

<i>none</i>	
-------------	--

4.31.1.3 infinite_proc()

```
void infinite_proc ( )
```

This process initiates a identical process to [idle\(\)](#), but is not a system process, and can be deleted if it has already been suspended.

Parameters

<i>None</i>	
-------------	--

Return values

<i>None</i>	
-------------	--

4.31.2 Variable Documentation**4.31.2.1 alarms**

```
struct alarm_list alarms
```

4.32 modules/structs.h File Reference

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

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

Data Structures

- struct [queue](#)
- struct [pcb](#)
- struct [context](#)
- struct [alarm](#)

This struct supports the alarm process.

- struct [alarm_list](#)

This struct stores user created alarms.

4.33 modules/sys_call.c File Reference

```
#include "mpx_supt.h"
#include "structs.h"
#include "internal_procedures.h"
Include dependency graph for sys_call.c:
```

Functions

- `u32int * sys_call` (struct `context` *registers)

Variables

- struct `pcb` * `cop`
- struct `context` * `reference`

4.33.1 Function Documentation

4.33.1.1 sys_call()

```
u32int* sys_call (
    struct context * registers )
```

4.33.2 Variable Documentation

4.33.2.1 cop

```
struct pcb* cop
```

4.33.2.2 reference

```
struct context* reference
```

4.34 modules/sys_call.h File Reference

Functions

- `u32int * sys_call` (struct `context` *registers)

4.34.1 Function Documentation

4.34.1.1 sys_call()

```
u32int* sys_call (
    struct context * registers )
```

4.35 modules/userR3Commands.c File Reference

```
#include "structs.h"
#include "userR3Commands.h"
#include "procsr3.h"
#include "internal_procedures.h"
#include "pcb_user_commands.h"
#include <string.h>
Include dependency graph for userR3Commands.c:
```

Functions

- void `yield` ()
This function will trigger the interrupt 60 and casue the command handler to yield to other processes.
- void `loadr3` ()
This function will create and insert all r3 processes into the suspended ready queue.

4.35.1 Function Documentation

4.35.1.1 loadr3()

```
void loadr3 ( )
```

This function will create and insert all r3 processes into the suspended ready queue.

4.35.1.2 yield()

```
void yield ( )
```

This function will trigger the interrupt 60 and casue the command handler to yield to other processes.

4.36 modules/userR3Commands.h File Reference

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

Macros

- `#define _USERR3COMMANDS_H` value

Functions

- void `yield` ()
This function will trigger the interrupt 60 and cause the command handler to yield to other processes.
- void `loadr3` ()
This function will create and insert all r3 processes into the suspended ready queue.

4.36.1 Macro Definition Documentation

4.36.1.1 _USERR3COMMANDS_H

```
#define _USERR3COMMANDS_H value
```

4.36.2 Function Documentation

4.36.2.1 loadr3()

```
void loadr3 ( )
```

This function will create and insert all r3 processes into the suspended ready queue.

4.36.2.2 yield()

```
void yield ( )
```

This function will trigger the interrupt 60 and cause the command handler to yield to other processes.

