MPX Thunder Krakens

R6

Generated by Doxygen 1.8.6

Sat Apr 23 2016 18:01:29

Contents

Chapter 1

Main Page

Welcome to the Programmer's manual for the Thunder Kracken's MPX Operating system. This document catalogues all of the information one may need to know regarding the use and modification of this Operating system and its contents. Included is a complete API of every method created for the operating system which includes all inputs and outputs as well as a brief summary of the purpose of each method. This will give you a more in depth look at all of the ordinary user commands as well as the internal commands used to perform functions that normal users cannot access. Most likely these commands will be the most important for making new programs on the operating system. This document also lists the documentation for the files files in the operating system. This includes all of the variables and methods used in each file. These will help direct you as to where certain functions are defined. For general usage tips, please refer to the user manual. We hope you find working with the Thunder Kracken's MPX Operating System as enjoyable as we do and we thank you for using our product.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 context Struct Reference

Context structure that holds the 15 CPU register values to begin and resume process execution.

`#include <context.h>`

Data Fields

- u32int gs

  Segment register.
- u32int fs

  Segment register.
- u32int es

  Segment register.
- u32int ds

  Segment register.
- u32int edi

  General-purpose register.
- u32int esi

  General-purpose register.
- u32int ebp

  General-purpose register.
- u32int esp

  General-purpose register.
- u32int ebx

  General-purpose register.
- u32int edx

  General-purpose register.
- u32int ecx

  General-purpose register.
- u32int eax

  General-purpose register.
- u32int eip

Status and control register.

- u32int cs

  Status and control register.

- u32int eflags

  Status and control register.

4.1.1 Detailed Description

Context structure that holds the 15 CPU register values to begin and resume process execution.

4.1.2 Field Documentation

4.1.2.1 u32int context::cs

Status and control register.

4.1.2.2 u32int context::ds

Segment register.

4.1.2.3 u32int context::eax

General-purpose register.

4.1.2.4 u32int context::ebp

General-purpose register.

4.1.2.5 u32int context::ebx

General-purpose register.

4.1.2.6 u32int context::ecx

General-purpose register.

4.1.2.7 u32int context::edi

General-purpose register.

4.1.2.8 u32int context::edx

General-purpose register.

4.1.2.9 u32int context::eflags

Status and control register.

4.1.2.10 u32int context::eip

Status and control register.

4.1.2.11 u32int context::es

Segment register.

4.1.2.12 u32int context::esi

General-purpose register.

4.1.2.13 u32int context::esp

General-purpose register.

4.1.2.14 u32int context::fs

Segment register.

4.1.2.15 u32int context::gs

Segment register.

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

- modules/r3/context.h

4.2 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

4.2.1 Field Documentation

4.2.1.1 int date_time::day_m

4.2.1.2 int date_time::day_w

4.2.1.3 int date_time::day_y

4.2.1.4 int date_time::hour

4.2.1.5 int date_time::min

4.2.1.6 int date_time::mon

4.2.1.7 int date_time::sec

4.2.1.8 int date_time::year

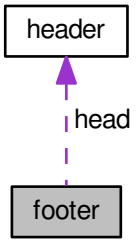
The documentation for this struct was generated from the following file:

- include/system.h

4.3 footer Struct Reference

`#include <heap.h>`

Collaboration diagram for footer:



Data Fields

- header head

4.3.1 Field Documentation

4.3.1.1 header footer::head

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

- include/mem/heap.h

4.4 gdt_descriptor_struct Struct Reference

`#include <tables.h>`

Data Fields

- u16int limit
- u32int base

4.4.1 Field Documentation

4.4.1.1 u32int gdt_descriptor_struct::base

4.4.1.2 u16int gdt_descriptor_struct::limit

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

- include/core/tables.h

4.5 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

4.5.1 Field Documentation

4.5.1.1 u8int gdt_entry_struct::access

4.5.1.2 u8int gdt_entry_struct::base_high

4.5.1.3 u16int gdt_entry_struct::base_low

4.5.1.4 u8int gdt_entry_struct::base_mid

4.5.1.5 u8int gdt_entry_struct::flags

4.5.1.6 u16int gdt_entry_struct::limit_low

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

- include/core/tables.h

4.6 header Struct Reference

`#include <heap.h>`

Data Fields

- int size
- int index_id

4.6.1 Field Documentation

4.6.1.1 int header::index_id

4.6.1.2 int header::size

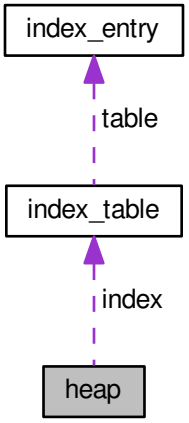
The documentation for this struct was generated from the following file:

- include/mem/heap.h

4.7 heap Struct Reference

`#include <heap.h>`

Collaboration diagram for heap:



Data Fields

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

4.7.1 Field Documentation

4.7.1.1 u32int heap::base

4.7.1.2 index_table heap::index

4.7.1.3 u32int heap::max_size

4.7.1.4 u32int heap::min_size

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

- include/mem/heap.h

4.8 idt_entry_struct Struct Reference

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

Data Fields

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

4.8.1 Field Documentation

4.8.1.1 u16int idt_entry_struct::base_high

4.8.1.2 u16int idt_entry_struct::base_low

4.8.1.3 u8int idt_entry_struct::flags

4.8.1.4 u16int idt_entry_struct::sselect

4.8.1.5 u8int idt_entry_struct::zero

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

- include/core/tables.h

4.9 idt_struct Struct Reference

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

Data Fields

- u16int limit
- u32int base

4.9.1 Field Documentation

4.9.1.1 u32int idt_struct::base

4.9.1.2 u16int idt_struct::limit

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

- include/core/tables.h

4.10 index_entry Struct Reference

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

Data Fields

- int size
- int empty
- u32int block

4.10.1 Field Documentation

4.10.1.1 u32int index_entry::block

4.10.1.2 int index_entry::empty

4.10.1.3 int index_entry::size

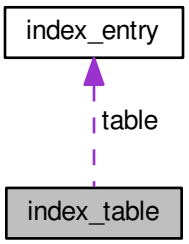
The documentation for this struct was generated from the following file:

- include/mem/heap.h

4.11 index_table Struct Reference

`#include <heap.h>`

Collaboration diagram for index_table:



Data Fields

- index_entry table [TABLE_SIZE]
- int id

4.11.1 Field Documentation

4.11.1.1 int index_table::id

4.11.1.2 index_entry index_table::table[TABLE_SIZE]

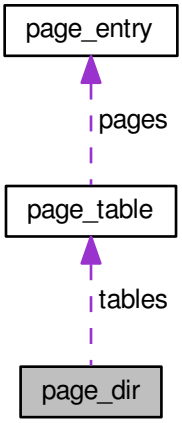
The documentation for this struct was generated from the following file:

- include/mem/heap.h

4.12 page_dir Struct Reference

`#include <paging.h>`

Collaboration diagram for page_dir:



Data Fields

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

4.12.1 Field Documentation

4.12.1.1 page_table* page_dir::tables[1024]

4.12.1.2 u32int page_dir::tables_phys[1024]

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

- include/mem/paging.h

4.13 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

4.13.1 Field Documentation

4.13.1.1 u32int page_entry::accessed

4.13.1.2 u32int page_entry::dirty

4.13.1.3 u32int page_entry::frameaddr

4.13.1.4 u32int page_entry::present

4.13.1.5 u32int page_entry::reserved

4.13.1.6 u32int page_entry::usermode

4.13.1.7 u32int page_entry::writeable

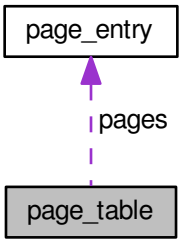
The documentation for this struct was generated from the following file:

- include/mem/paging.h

4.14 page_table Struct Reference

`#include <paging.h>`

Collaboration diagram for page_table:



Data Fields

- page_entry pages [1024]

4.14.1 Field Documentation

4.14.1.1 page_entry page_table::pages[1024]

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

- include/mem/paging.h

4.15 param Struct Reference

A structure to represent interrupt.

`#include <mpx_supt.h>`

Data Fields

- int op_code

  interrupt's operation
- int device_id

  interrupt's device

4.15.1 Detailed Description

A structure to represent interrupt.

4.15.2 Field Documentation

4.15.2.1 int param::device_id

interrupt's device

4.15.2.2 int param::op_code

interrupt's operation

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

- modules/mpx_supt.h

Chapter 5

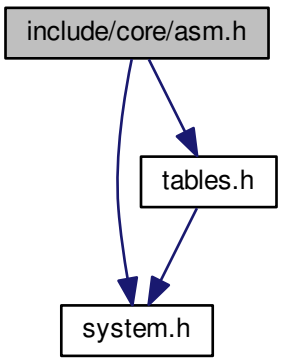
File Documentation

5.1 documentation/mainpage.dox File Reference

5.2 include/core/asm.h File Reference

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

Include dependency graph for asm.h:



5.3 include/core/interrupts.h File Reference

Functions

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

5.3.1 Function Documentation

5.3.1.1 void init_irq (void)

5.3.1.2 void init_pic (void)

5.4 include/core/io.h File Reference

Macros

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

5.4.1 Macro Definition Documentation

5.4.1.1 #define inb(*port*)

Value:

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

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

5.5 include/core/serial.h File Reference

Serial - Header.

Macros

- #define COM1 0x3f8
- #define COM2 0x2f8
- #define COM3 0x3e8
- #define COM4 0x2e8
- #define WithoutEcho 0
- #define WithEcho 1
- #define USER_INPUT_BUFFER_SIZE 100

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)

get_input_line

Get user's input from keyborad.

Parameters

buffer	*The pointer to the buffer where store the user's input.*
buffer_size	*The size of that buffer.*
bWithEcho	*With echo or not*

Returns

VOID

- void get_input_line (char ∗buffer, const int bWithEcho)

5.5.1 Detailed Description

Serial - Header.

Author

Thunder Krakens

Date

February 2nd, 2016

Version

R1

5.5.2 Macro Definition Documentation

5.5.2.1 #define COM1 0x3f8

5.5.2.2 #define COM2 0x2f8

5.5.2.3 #define COM3 0x3e8

5.5.2.4 #define COM4 0x2e8

5.5.2.5 #define USER_INPUT_BUFFER_SIZE 100

5.5.2.6 #define WithEcho 1

5.5.2.7 #define WithoutEcho 0

5.5.3 Function Documentation

5.5.3.1 void get_input_line (char ∗ *buffer,* const int *bWithEcho*)

5.5.3.2 int init_serial (int *device*)

5.5.3.3 int serial_print (const char ∗ *msg*)

5.5.3.4 int serial_println (const char * *msg*)

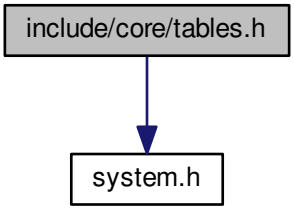
5.5.3.5 int set_serial_in (int *device*)

5.5.3.6 int set_serial_out (int *device*)

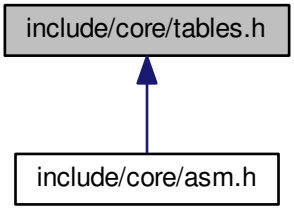
5.6 include/core/tables.h File Reference

```
#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

5.6.1 Function Documentation

5.6.1.1 struct idt_entry_struct __attribute__ ((packed))

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

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

5.6.1.4 void init_gdt ( )

5.6.1.5 void init_idt ( )

5.6.2 Variable Documentation

5.6.2.1 u8int access

5.6.2.2 u32int base

5.6.2.3 u8int base_high

5.6.2.4 u16int base_low

5.6.2.5 u8int base_mid

5.6.2.6 u8int flags

5.6.2.7 u16int limit

5.6.2.8 u16int limit_low

5.6.2.9 u16int sselect

5.6.2.10 u8int zero

5.7 include/mem/heap.h File Reference

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)

Variables

- typedef __attribute__

5.7.1 Macro Definition Documentation

5.7.1.1 #define KHEAP_BASE 0xD000000

5.7.1.2 #define KHEAP_MIN 0x10000

5.7.1.3 #define KHEAP_SIZE 0x1000000

5.7.1.4 #define TABLE_SIZE 0x1000

5.7.2 Function Documentation

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

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

5.7.2.3 void init_kheap ( )

5.7.2.4 u32int kfree ( )

5.7.2.5 u32int kmalloc (u32int *size*)

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

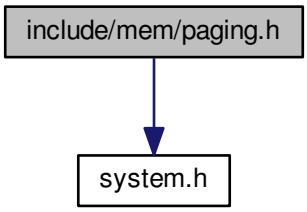
5.7.3 Variable Documentation

5.7.3.1 struct gdt_entry_struct __attribute__

5.8 include/mem/paging.h File Reference

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

Include dependency graph for paging.h:



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)

5.8.1 Macro Definition Documentation

5.8.1.1 #define PAGE_SIZE 0x1000

5.8.2 Function Documentation

5.8.2.1 void clear_bit (u32int *addr*)

5.8.2.2 u32int first_free ( )

5.8.2.3 u32int get_bit (u32int *addr*)

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

5.8.2.5 void init_paging ( )

5.8.2.6 void load_page_dir (page_dir * *new_page_dir*)

5.8.2.7 void new_frame (page_entry * *page*)

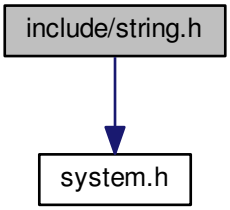
5.8.2.8 void set_bit (u32int *addr*)

5.9 include/string.h File Reference

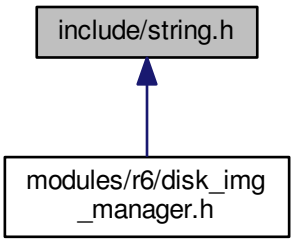
Many usefull functions that used for handling string.

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

Include dependency graph for string.h:



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



Functions

isspace.

Identifies if its space

Parameters

A	*constant character*

Returns

1 if it is space, otherwise return 0.

- int isspace (const char ∗c)

memset.

Sets region of memory

Parameters

s	*destination*
c	*byte to write*
n	*count*

Returns

the pointer to the memory space.

- void ∗ memset (void ∗s, int c, size_t n)

strcpy.

Copies one string to another.

Parameters

s1	*Destination string*
s2	*Source string*

Returns

pointer to the destination String

- char ∗ strcpy (char ∗s1, const char ∗s2)

strcat.

Concatenate the contents of one string onto another.

Parameters

s1	*Destination string*
s2	*Source string*

Returns

pointer to destination String

- char ∗ strcat (char ∗s1, const char ∗s2)

strlen.

Returns the length of a string.

Parameters

s	*String input.*

Returns

count Length of the String

- int strlen (const char ∗s)

strcmp.

String comparison.

Parameters

s1	*First string to use for the compare.*
s2	*Second string to use for the compare.*

Returns

whether they are the same or not.

- int strcmp (const char ∗s1, const char ∗s2)

strtok.

Split string into tokens.

Parameters

s1	*String*
s2	*Delimiter*

Returns

the pointer to the token.

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

atoi.

Convert an ASCII string to an integer.

Parameters

s	*String.*

Returns

The converted integer.

- int atoi (const char *s)

sprintf.

Generate a formatted string.

%[-x]c output a character, '-' - align right, x - the output width

%[-x]s output a string, '-' - align right, x - the output width

%[{-,+}x]d output a character, '-' - align right, '+' - align right and display '+' sign, x - the output width

%[-x]X (capital 'X') output a hexadecimal number, '-' - align right, x - the output width

note: Output width will be ignored if width is smaller than actual length.

Parameters

str	*- Output string.*
format	*- The format of the string.*
...	*- All of the additional parameters.*

Returns

vsprintf(str, format, ap) - Return the string with its format and pointer.

- int sprintf (char *str, const char *format,...)

printf.

Print out a formatted string.

%[-x]c output a character, '-' - align right, x - the output width

%[-x]s output a string, '-' - align right, x - the output width

%[{-,+}x]d output a character, '-' - align right, '+' - align right and display '+' sign, x - the output width

%[-x]X (capital 'X') output a hexadecimal number, '-' - align right, x - the output width

note: Output width will be ignored if width is smaller than actual length.

Parameters

str	*- Output string.*
format	*- The format of the string.*
...	*- All of the additional parameters.*

Returns

vsprintf(str, format, ap) - Return the string with its format and pointer.

- int printf (const char *format,...)

5.9.1 Detailed Description

Many usefull functions that used for handling string.

Author

Thunder Krakens

Date

February 2nd, 2016

Version

R1

5.9.2 Function Documentation

5.9.2.1 int atoi (const char * *s*)

5.9.2.2 int isspace (const char * *c*)

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

5.9.2.4 int printf (const char * *format*, ...)

5.9.2.5 int sprintf (char * *str*, const char * *format*, ...)

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

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

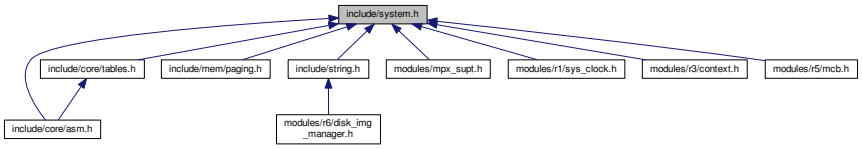
5.9.2.8 char* strcpy (char * *s1*, const char * *s2*)

5.9.2.9 int strlen (const char * *s*)

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

5.10 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)

5.10.1 Macro Definition Documentation

5.10.1.1 #define asm __asm__

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

5.10.1.3 #define GDT_CS_ID 0x01

5.10.1.4 #define GDT_DS_ID 0x02

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

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

5.10.1.7 #define no_warn(*p*) if (p) while (1) break

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

5.10.1.9 #define NULL 0

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

5.10.1.11 #define volatile __volatile__

5.10.2 Typedef Documentation

5.10.2.1 typedef unsigned int size_t

5.10.2.2 typedef unsigned short u16int

5.10.2.3 typedef unsigned long u32int

5.10.2.4 typedef unsigned char u8int

5.10.3 Function Documentation

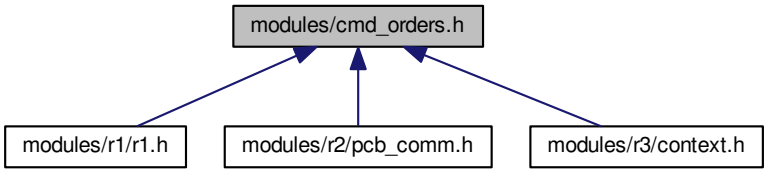
5.10.3.1 void klogv (const char * *msg*)

5.10.3.2 void kpanic (const char * *msg*)

5.11 modules/cmd_orders.h File Reference

This file contains orders & index of all the commands.

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



Macros

- #define WITH_R2_TEMP_CMD 0
- #define WITH_R3_TEMP_CMD 0
- #define WITH_R5_TEMP_CMD 1
- #define FUNCTIONS_BEGIN 0

- #define HELP 0
- #define MPX_FUNCTIONS_BEGIN 1

- #define VERSION MPX_FUNCTIONS_BEGIN+0
- #define GETTIME MPX_FUNCTIONS_BEGIN+1
- #define SETTIME MPX_FUNCTIONS_BEGIN+2
- #define GETDATE MPX_FUNCTIONS_BEGIN+3
- #define SETDATE MPX_FUNCTIONS_BEGIN+4
- #define SHUTDOWN MPX_FUNCTIONS_BEGIN+5
- #define LOADR3 MPX_FUNCTIONS_BEGIN+6
- #define MPX_FUNC_END MPX_FUNCTIONS_BEGIN+6
- #define PCB_FUNCTIONS_BEGIN MPX_FUNC_END+1

- #define SUSPDPCB PCB_FUNCTIONS_BEGIN+0
- #define RESUMEPCB PCB_FUNCTIONS_BEGIN+1
- #define SETPCBPRIO PCB_FUNCTIONS_BEGIN+2
- #define SHOWPCB PCB_FUNCTIONS_BEGIN+3
- #define PCB_FUNC_END PCB_FUNCTIONS_BEGIN+3
- #define MCB_FUNCTIONS_BEGIN PCB_FUNC_END+1

- #define SHOWMCB MCB_FUNCTIONS_BEGIN+0
- #define INITMCB MCB_FUNCTIONS_BEGIN+1
- #define ALLOCMCB MCB_FUNCTIONS_BEGIN+2
- #define FREEMCB MCB_FUNCTIONS_BEGIN+3
- #define ISMCBEMPT MCB_FUNCTIONS_BEGIN+4
- #define MCB_FUNC_END MCB_FUNCTIONS_BEGIN+4
- #define NUM_OF_FUNCTIONS MCB_FUNC_END+1

5.11.1 Detailed Description

This file contains orders & index of all the commands.

Author

Thunder Krakens

Date

February 7nd, 2016

Version

R5

5.11.2 Macro Definition Documentation

5.11.2.1 #define ALLOCMCB MCB_FUNCTIONS_BEGIN+2

5.11.2.2 #define FREEMCB MCB_FUNCTIONS_BEGIN+3

5.11.2.3 #define FUNCTIONS_BEGIN 0

5.11.2.4 #define GETDATE MPX_FUNCTIONS_BEGIN+3

5.11.2.5 #define GETTIME MPX_FUNCTIONS_BEGIN+1

5.11.2.6 #define HELP 0

5.11.2.7 #define INITMCB MCB_FUNCTIONS_BEGIN+1

5.11.2.8 #define ISMCBEMPT MCB_FUNCTIONS_BEGIN+4

5.11.2.9 #define LOADR3 MPX_FUNCTIONS_BEGIN+6

5.11.2.10 #define MCB_FUNC_END MCB_FUNCTIONS_BEGIN+4

5.11.2.11 #define MCB_FUNCTIONS_BEGIN PCB_FUNC_END+1

5.11.2.12 #define MPX_FUNC_END MPX_FUNCTIONS_BEGIN+6

5.11.2.13 #define MPX_FUNCTIONS_BEGIN 1

5.11.2.14 #define NUM_OF_FUNCTIONS MCB_FUNC_END+1

5.11.2.15 #define PCB_FUNC_END PCB_FUNCTIONS_BEGIN+3

5.11.2.16 #define PCB_FUNCTIONS_BEGIN MPX_FUNC_END+1

5.11.2.17 #define RESUMEPCB PCB_FUNCTIONS_BEGIN+1

5.11.2.18 #define SETDATE MPX_FUNCTIONS_BEGIN+4

5.11.2.19 #define SETPCBPRIO PCB_FUNCTIONS_BEGIN+2

5.11.2.20 #define SETTIME MPX_FUNCTIONS_BEGIN+2

5.11.2.21 #define SHOWMCB MCB_FUNCTIONS_BEGIN+0

5.11.2.22 #define SHOWPCB PCB_FUNCTIONS_BEGIN+3

5.11.2.23 #define SHUTDOWN MPX_FUNCTIONS_BEGIN+5

5.11.2.24 #define SUSPDPCB PCB_FUNCTIONS_BEGIN+0

5.11.2.25 #define VERSION MPX_FUNCTIONS_BEGIN+0

5.11.2.26 #define WITH_R2_TEMP_CMD 0

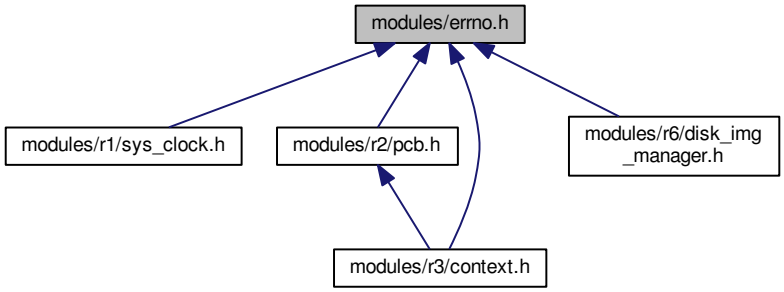
5.11.2.27 #define WITH_R3_TEMP_CMD 0

5.11.2.28 #define WITH_R5_TEMP_CMD 1

5.12 modules/errno.h File Reference

This file contains the type of errors. The error can be from invalid paramter passed to a function, or invalid input format.

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



Macros

- #define E_NOERROR 0
- #define E_INVPARA 1
- #define E_INVSTRF 2
- #define E_INVUSRI 3
- #define E_FREEMEM 4

  Error we cannot actually free the memory space since the student_free had not been implemented before R5.
- #define E_NULL_PTR 5

  A NULL Pointer Error.
- #define E_EMPTPCB 6

  The pcb queue is empty.
- #define E_PCB_SYS 7
- #define E_FILE_NF 8

  The file was not found.
- #define E_PROGERR 99

Typedefs

error_t.

The datetype that holds the error code.

- typedef unsigned int error_t

5.12.1 Detailed Description

This file contains the type of errors. The error can be from invalid paramter passed to a function, or invalid input format.

Author

Thunder Krakens

Date

February 7nd, 2016

Version

R2

5.12.2 Macro Definition Documentation

5.12.2.1 #define E_EMPTPCB 6

The pcb queue is empty.

5.12.2.2 #define E_FILE_NF 8

The file was not found.

5.12.2.3 #define E_FREEMEM 4

Error we cannot actually free the memory space since the student_free had not been implemented before R5.

5.12.2.4 #define E_INVPARA 1

5.12.2.5 #define E_INVSTRF 2

5.12.2.6 #define E_INVUSRI 3

5.12.2.7 #define E_NOERROR 0

5.12.2.8 #define E_NULL_PTR 5

A NULL Pointer Error.

5.12.2.9 #define E_PCB_SYS 7

5.12.2.10 #define E_PROGERR 99

5.12.3 Typedef Documentation

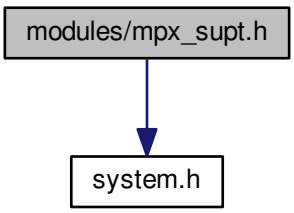
5.12.3.1 typedef unsigned int error_t

5.13 modules/mpx_supt.h File Reference

MPX System Supplementaries.

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

Include dependency graph for mpx_supt.h:



Data Structures

- struct param

  A structure to represent interrupt.

Macros

- #define EXIT 0
- #define IDLE 1
- #define READ 2
- #define WRITE 3
- #define MODULE_R1 0
- #define MODULE_R2 1
- #define MODULE_R3 2
- #define MODULE_R4 4
- #define MODULE_R5 8

Functions

sys_req

Generate interrupt 60H

Parameters

int	*op_code (IDLE)*

- int sys_req (int op_code)

mpx_init

Initialize MPX support software

Parameters

int	*cur_mod (symbolic constants MODULE_R1, MODULE_R2, etc*

- void mpx_init (int cur_mod)

set_malloc

Sets the memory allocation function for sys_alloc_mem

Parameters

Function	*pointer*

- void sys_set_malloc (u32int(∗func)(u32int))

set_free

Sets the memory free function for sys_free_mem

Parameters

s1-destination,s2-source	

- void sys_set_free (int(∗func)(void ∗))

sys_alloc_mem

Allocates a block of memory (similar to malloc)

Parameters

Number	*of bytes to allocate*

- void ∗ sys_alloc_mem (u32int size)

sys_free_mem

Frees memory

Parameters

Pointer	*to block of memory to free*

- int sys_free_mem (void ∗ptr)

idle

The idle process

Parameters

None	

- void idle ()

get_op_code

Returns the interrupt's operation code

Parameters

None	

- int get_op_code ()

5.13.1 Detailed Description

MPX System Supplementaries.

Author

Thunder Krakens

Date

March 18, 2016

Version

R3

5.13.2 Macro Definition Documentation

5.13.2.1 #define EXIT 0

5.13.2.2 #define IDLE 1

5.13.2.3 #define MODULE_R1 0

5.13.2.4 #define MODULE_R2 1

5.13.2.5 #define MODULE_R3 2

5.13.2.6 #define MODULE_R4 4

5.13.2.7 #define MODULE_R5 8

5.13.2.8 #define READ 2

5.13.2.9 #define WRITE 3

5.13.3 Function Documentation

5.13.3.1 int get_op_code ( )

5.13.3.2 void idle ( )

5.13.3.3 void mpx_init (int *cur_mod*)

5.13.3.4 void* sys_alloc_mem (u32int *size*)

5.13.3.5 int sys_free_mem (void * *ptr*)

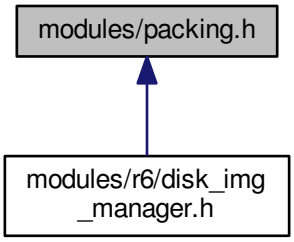
5.13.3.6 int sys_req (int *op_code*)

5.13.3.7 void sys_set_free (int(*)(void *) *func*)

5.13.3.8 void sys_set_malloc (u32int(*)(u32int) *func*)

5.14 modules/packing.h File Reference

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



Macros

- #define PACKED(class_to_pack) __pragma(pack(push, 1)) class_to_pack __pragma(pack(pop))

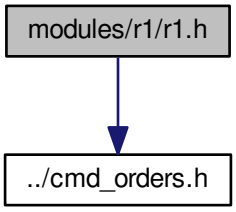
5.14.1 Macro Definition Documentation

5.14.1.1 #define PACKED(*class_to_pack*) __pragma(pack(push, 1)) class_to_pack __pragma(pack(pop))

5.15 modules/r1/r1.h File Reference

The command handler and functions associations for Module R1.

```
#include "../cmd_orders.h"
```
Include dependency graph for r1.h:



Enumerations

- enum comm_type

Functions

- enum comm_type __attribute__ ((packed))

commhand

Accepts and handles commands from the user.

Returns

VOID

- void commhand ()

command_line_parser

Splits the complete command line into tokens by space, single quote, or double quote.

Parameters

CmdStr	*The complete input command.*
argc	*The number of tokens found.*
argv	*The array of tokens.*
MaxArgNum	*The maximum number of tokens that array can hold.*
MaxStrLen	*The maximum length of each token that string can hold.*

Returns

void

- void command_line_parser (const char *CmdStr, int *argc, char **argv, const int MaxArgNum, const int MaxStrLen)

print_help

prints the help message of a certain function that specified by the index number

Parameters

function_index	*The index number of that function.*

Returns

void

- void print_help (const int function_index)
- int help_usages (enum comm_type type)

Variables

- mpx
- pcb
- mcb
- help

5.15.1 Detailed Description

The command handler and functions associations for Module R1.

Author

Thunder Krakens

Date

March 17, 2016

Version

R3 & R4

5.15.2 Enumeration Type Documentation

5.15.2.1 enum comm_type

5.15.3 Function Documentation

5.15.3.1 enum comm_type __attribute__ ((packed))

5.15.3.2 void command_line_parser (const char * *CmdStr,* int * *argc,* char ** *argv,* const int *MaxArgNum,* const int *MaxStrLen*)

5.15.3.3 void commhand ( )

5.15.3.4 int help_usages (enum comm_type *type*)

5.15.3.5 void print_help (const int *function_index*)

5.15.4 Variable Documentation

5.15.4.1 help

5.15.4.2 mcb

5.15.4.3 mpx

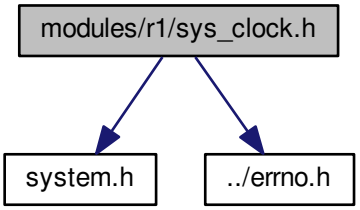
5.15.4.4 pcb

5.16 modules/r1/sys_clock.h File Reference

The main file that manipulates and controls the system's clock.

```
#include <system.h>
#include "../errno.h"
```

Include dependency graph for sys_clock.h:



Functions

set_time_main.

Sets the time for the system.

Parameters

argc	*The number of tokens found.*
argv	*The array of tokens.*

Returns

0

- int set_time_main (int argc, char **argv)

get_time_main.

Retrieves system's current time.

Parameters

argc	*The number of tokens found.*
argv	*The array of tokens.*

Returns

0

- int get_time_main (int argc, char **argv)

set_time_str.

Sets the time for the system by string.

Parameters

timeStr	*The string type of current Time.*

Returns

0 if there is no error, otherwise return a error code.

- error_t set_time_str (const char *timeStr)

get_time.

Retrieves system's current time and date.

Parameters

dateTimeValues	*The value of current time and date*

Returns

VOID

- void get_time (date_time *dateTimeValues)

set_time.

Sets the time for the system by date_time struct.

Parameters

dateTimeValues	*The struct that holds the time values.*

Returns

0 if there is no error, otherwise return a error code.

- error_t set_time (const date_time *dateTimeValues)

set_date_main.

Sets system's date.

Parameters

argc	*The number of tokens.*
argv	*The array of tokens.*

Returns

0

- int set_date_main (int argc, char **argv)

get_date_main.

Retrieves system's current date.

Parameters

argc	*The number of tokens.*
argv	*The array of tokens.*

Returns

0

- int get_date_main (int argc, char **argv)

get_date.

Retrieves system's current date.

Parameters

dateTimeValues	*The struct that holds the value of current date*

Returns

VOID

- void get_date (date_time *dateTimeValues)

set_date_str.

Sets the date for the system by string.

Parameters

str	*The string type of current date.*

Returns

0 if there is no error, otherwise return a error code.

- int set_date_str (const char *str)

set_date.

Sets the date of the system.

Parameters

dateTimeValues	*The struct that holds the value of date*

Returns

0 if there is no error, otherwise return a error code.

- error_t set_date (const date_time *dateTimeValues)

5.16.1 Detailed Description

The main file that manipulates and controls the system's clock.

Author

Thunder Krakens

Date

February 2nd, 2016

Version

R1

5.16.2 Function Documentation

5.16.2.1 void get_date (date_time * *dateTimeValues*)

5.16.2.2 int get_date_main (int *argc*, char ** *argv*)

5.16.2.3 void get_time (date_time * *dateTimeValues*)

5.16.2.4 int get_time_main (int *argc*, char ** *argv*)

5.16.2.5 error_t set_date (const date_time * *dateTimeValues*)

5.16.2.6 int set_date_main (int *argc*, char ** *argv*)

5.16.2.7 int set_date_str (const char * *str*)

5.16.2.8 error_t set_time (const date_time * *dateTimeValues*)

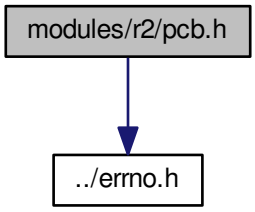
5.16.2.9 int set_time_main (int *argc*, char ** *argv*)

5.16.2.10 error_t set_time_str (const char * *timeStr*)

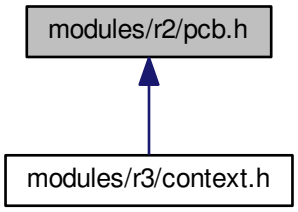
5.17 modules/r2/pcb.h File Reference

The Process Control Block.

```
#include "../errno.h"
```
Include dependency graph for pcb.h:



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



Macros

- #define SIZE_OF_STACK 1024
- #define SIZE_OF_PCB_NAME 10
- #define COMMHAND_PCB_NAME "commhand"
- #define IDLE_PCB_NAME "idle"

Enumerations

- enum process_class

  PCB process class types.

Functions

- enum process_class __attribute__ ((packed))

pcb_init

Initiates the PCB queues

- void pcb_init ()

allocate_pcb

allocate a space for the PCB structure.

Returns

The pointer that point to the PCB structure.

- struct pcb_struct * allocate_pcb ()

free_pcb

Frees all memory associated with given PCB, including the PCB itself, the stack, etc, with sys_free_mem()

Parameters

pcb_ptr	*The pointer to the PCB*

Returns

The error code. Possible error code to be returned: E_NOERROR No error. E_INVPARA The PCB probably had not been removed from queue before free it.

- error_t free_pcb (struct pcb_struct *pcb_ptr)

setup_pcb

allocate a space for the PCB structure, setup the properties of the PCB.

NOTE: pName must less than 10 character, pClass should be either "application" or "system" , and pPriority must within the range of [0, 9].

Parameters

pName	*Process Name (length $<$ 10).*
pClass	*Process class (system or application).*
pPriority	*Process priority (0 $\sim$ 9).*

Returns

NULL if error occured, otherwise, the pointer that point to the PCB structure.

- struct pcb_struct * setup_pcb (const char *pName, const enum process_class pClass, const unsigned char pPriority)

find_pcb

Will search all queues for a process named pName

Parameters

pName	*The char pointer to the desired searched name*

Returns

PCB pointer if found, NULL if PCB is not found

- struct pcb_struct * find_pcb (const char *pName)

insert_pcb

Inserts PCB into the appropriate queue.

Parameters

pcb_ptr	*The pointer to the PCB*

Returns

The error code. Possible error code to be returned: E_NOERROR No error. E_NULL_PTR Null pointer error. E_INVPARA The given PCB has running status or abnormal data members.

- error_t insert_pcb (struct pcb_struct *pcb_ptr)

remove_pcb

Removes PCB from the queue it is currently in.

Parameters

pcb_ptr	*The pointer to the PCB*

Returns

The error code. Possible error code to be returned: E_NOERROR No error. E_NULL_PTR Null pointer error. E_INVPARA The given PCB has abnormal data members.

- error_t remove_pcb (struct pcb_struct *pcb_ptr)

suspend_pcb

Suspends the specific PCB.

Parameters

pcb_ptr	*The pointer to the PCB*

Returns

The error code. Possible error code to be returned: E_NOERROR No error. E_NULL_PTR Null pointer error.

- error_t suspend_pcb (struct pcb_struct *pcb_ptr)

resume_pcb

Resumes the specific PCB.

Parameters

pcb_ptr	*The pointer to the PCB*

Returns

The error code. Possible error code to be returned: E_NOERROR No error. E_NULL_PTR Null pointer error.

- error_t resume_pcb (struct pcb_struct *pcb_ptr)

set_pcb_priority

Sets the priority of the selected PCB

Parameters

pcb_ptr	*The PCB pointer.*
pPriorty	*The assigned priorirty*

Returns

The error code. Possible error code to be returned: E_NOERROR No error. E_NULL_PTR Null pointer error. E_INVPARA The pPriority is out of range. Or, the given PCB has abnormal data members (By "remove_pcb" or "insert_pcb").

- error_t set_pcb_priority (struct pcb_struct *pcb_ptr, const unsigned char pPriority)

show_pcb

Displays the name, class, state, suspend status, and priority of a PCB.

Parameters

pName	*The PCB pointer.*

Returns

The error code. Possible error code to be returned: E_NOERROR No error. E_NULL_PTR Null pointer error.

- error_t show_pcb (struct pcb_struct *pcb_ptr)

show_all_processes

Displays all of the processes and their attributes.

Returns

VOID.

- void show_all_processes ()

show_ready_processes

Displays all of the ready processes and their attributes.

Returns

VOID.

- void show_ready_processes ()

show_blocked_processes

displays all blocked processes and their attributes

Returns

VOID.

- void show_blocked_processes ()

block_pcb

puts the given pcb into the blocked state and places it into the correct queue

Parameters

pcb_ptr	*The pointer to the PCB*

Returns

The error code. Possible error code to be returned: E_NOERROR No error. E_NULL_PTR Null pointer error. E_INVPARA The given PCB has abnormal data members (By "remove_pcb" or "insert_pcb").

- error_t block_pcb (struct pcb_struct *pcb_ptr)

unblock_pcb

puts the given pcb into the unblocked state and places it into the correct queue

Parameters

pcb_ptr	*The pointer to the PCB*

Returns

The error code. Possible error code to be returned: E_NOERROR No error. E_NULL_PTR Null pointer error. E_INVPARA The given PCB has abnormal data members (By "remove_pcb" or "insert_pcb").

- error_t unblock_pcb (struct pcb_struct *pcb_ptr)

get_running_process

gets a unsuspended and unblocked process from the front of the queue, and sets it to running state.

Parameters

None	

Returns

NULL if there is no process available, otherwise, the pointer that point to the PCB structure.

- struct pcb_struct * get_running_process ()

save_running_process

sets the running process to ready state, and inserts it to the ready queue.

Parameters

pcb_ptr	*The pointer to the PCB.*
new_stack_top	*The pointer to the new stack top.*

Returns

The error code. Possible error code to be returned: E_NOERROR No error. E_NULL_PTR Null pointer error. E_INVPARA The given PCB has abnormal data members (By "insert_pcb").

- error_t save_running_process (struct pcb_struct *pcb_ptr, struct context *new_stack_top)

get_stack_top

gets the pointer to the stack top of the specific PCB.

Parameters

pcb_ptr	*The pointer to the PCB.*

Returns

NULL if the pcb_ptr is NULL, otherwise, the pointer that point to the stack top of the specific PCB.

- unsigned char * get_stack_top (struct pcb_struct *pcb_ptr)

get_stack_base

gets the pointer to the stack base of the specific PCB.

Parameters

pcb_ptr	*The pointer to the PCB.*

Returns

NULL if the pcb_ptr is NULL, otherwise, the pointer that point to the stack base of the specific PCB.

- unsigned char * get_stack_base (struct pcb_struct *pcb_ptr)

shutdown_pcb

called when system is going to shutdown, removes all PCBs, free all PCBs.

Returns

VOID

- void shutdown_pcb ()

Variables

- pcb_class_app

  Process is an application process.
- pcb_class_sys

  < Process is a system process.

5.17.1 Detailed Description

The Process Control Block.

Author

Thunder Krakens

Date

February 7th, 2016

Version

R3

5.17.2 Macro Definition Documentation

5.17.2.1 #define COMMHAND_PCB_NAME "commhand"

5.17.2.2 #define IDLE_PCB_NAME "idle"

5.17.2.3 #define SIZE_OF_PCB_NAME 10

5.17.2.4 #define SIZE_OF_STACK 1024

5.17.3 Enumeration Type Documentation

5.17.3.1 enum process_class

PCB process class types.

5.17.4 Function Documentation

5.17.4.1 enum process_class __attribute__ ((packed))

5.17.4.2 struct pcb_struct* allocate_pcb ( )

5.17.4.3 error_t block_pcb (struct pcb_struct * *pcb_ptr*)

5.17.4.4 struct pcb_struct* find_pcb (const char * *pName*)

5.17.4.5 error_t free_pcb (struct pcb_struct * *pcb_ptr*)

5.17.4.6 struct pcb_struct* get_running_process ( )

5.17.4.7 unsigned char* get_stack_base (struct pcb_struct * *pcb_ptr*)

5.17.4.8 unsigned char* get_stack_top (struct pcb_struct * *pcb_ptr*)

5.17.4.9 error_t insert_pcb (struct pcb_struct * *pcb_ptr*)

5.17.4.10 void pcb_init ( )

5.17.4.11 error_t remove_pcb (struct pcb_struct * *pcb_ptr*)

5.17.4.12 error_t resume_pcb (struct pcb_struct * *pcb_ptr*)

5.17.4.13 error_t save_running_process (struct pcb_struct * *pcb_ptr,* struct context * *new_stack_top*)

5.17.4.14 error_t set_pcb_priority (struct pcb_struct * *pcb_ptr,* const unsigned char *pPriority*)

5.17.4.15 struct pcb_struct* setup_pcb (const char * *pName,* const enum process_class *pClass,* const unsigned char *pPriority*)

5.17.4.16 void show_all_processes ( )

5.17.4.17 void show_blocked_processes ( )

5.17.4.18 error_t show_pcb (struct pcb_struct * *pcb_ptr*)

5.17.4.19 void show_ready_processes ( )

5.17.4.20 void shutdown_pcb ( )

5.17.4.21 error_t suspend_pcb (struct pcb_struct * *pcb_ptr*)

5.17.4.22 error_t unblock_pcb (struct pcb_struct * *pcb_ptr*)

5.17.5 Variable Documentation

5.17.5.1 pcb_class_app

Process is an application process.

5.17.5.2 pcb_class_sys

< Process is a system process.

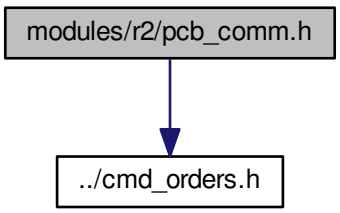
Process is a system process.

5.18 modules/r2/pcb_comm.h File Reference

The main functions that manipulate the PCB.

```
#include "../cmd_orders.h"
```

Include dependency graph for pcb_comm.h:



Functions

suspend_pcb_main.

The main function for the "suspend PCB".

Accepted formats: pcb suspend <name> pcb suspend –help

Parameters

argc	*The number of tokens found.*
argv	*The array of tokens.*

Returns

0

- int suspend_pcb_main (int argc, char **argv)

resume_pcb_main.

The main function for the "resume PCB".

Accepted formats: pcb resume <name> pcb resume –help

Parameters

argc	*The number of tokens found.*
argv	*The array of tokens.*

Returns

0

- int resume_pcb_main (int argc, char **argv)

set_pcb_priority_main.

The main function for the "set PCB priority".

Accepted formats: pcb setpriority <name> <priority> pcb setpriority –help

Parameters

argc	*The number of tokens found.*
argv	*The array of tokens.*

Returns

0

- int set_pcb_priority_main (int argc, char **argv)

show_pcb_main.

The main function for the "Show PCB", "Show all Processes", "Show Ready Processes", and "Show Blocked Processes".

Accepted formats: pcb show [name] pcb show -all pcb show -ready pcb show -blocked pcb show –help

Parameters

argc	*The number of tokens found.*
argv	*The array of tokens.*

Returns

0

- int show_pcb_main (int argc, char **argv)

5.18.1 Detailed Description

The main functions that manipulate the PCB.

Author

Thunder Krakens

Date

February 7th, 2016

Version

R2

5.18.2 Function Documentation

5.18.2.1 int resume_pcb_main (int *argc,* char ** *argv*)

5.18.2.2 int set_pcb_priority_main (int *argc,* char ** *argv*)

5.18.2.3 int show_pcb_main (int *argc,* char ** *argv*)

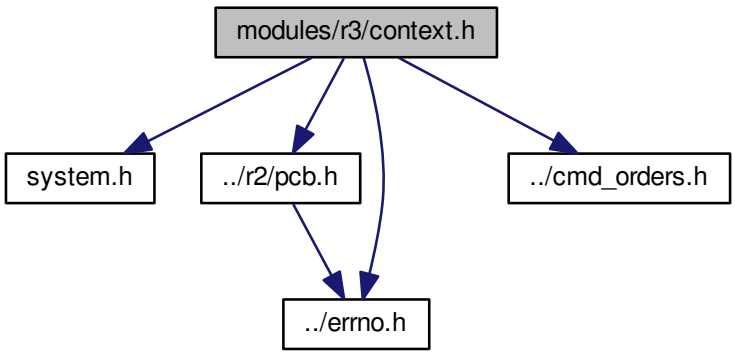
5.18.2.4 int suspend_pcb_main (int *argc,* char ** *argv*)

5.19 modules/r3/context.h File Reference

Context Switching.

```
#include <system.h>
#include "../r2/pcb.h"
#include "../errno.h"
#include "../cmd_orders.h"
```

Include dependency graph for context.h:



Data Structures

- struct context

  Context structure that holds the 15 CPU register values to begin and resume process execution.

Functions

sys_call

system call interrupt

Parameters

context∗	*registers current registers*

Returns

result if there is no current process running, it will load new context. If the process is still running, it will load its old context.

- u32int ∗ sys_call (struct context ∗registers)

load_process

loads a process into the PCB.

Parameters

pName	*Process Name*
pClass	*Process Class*
pPriority	*Process Priority*
∗function()	*A function pointer*

Returns

new_pcb Returns the values of the new PCB

- struct pcb_struct ∗ load_process (const char ∗pName, const enum process_class pClass, const unsigned char pPriority, void(∗function)())

load_r3_main

Loads the main function of R3.

Parameters

argc	*The number of tokens found.*
argv	*The array of tokens.*

Returns

0

- int load_r3_main (int argc, char ∗∗argv)

Variables

- struct context ∗ old_context
- struct pcb_struct ∗ cop
- struct context __attribute__

5.19.1 Detailed Description

Context Switching.

Author

Thunder Krakens

Date

March 18th, 2016

Version

R3

5.19.2 Function Documentation

5.19.2.1 struct pcb_struct* load_process (const char * *pName,* const enum process_class *pClass,* const unsigned char *pPriority,* void(*)() *function*)

5.19.2.2 int load_r3_main (int *argc,* char ** *argv*)

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

5.19.3 Variable Documentation

5.19.3.1 struct context __attribute__

5.19.3.2 struct pcb_struct* cop

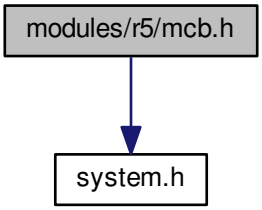
5.19.3.3 struct context* old_context

5.20 modules/r5/mcb.h File Reference

Memory Control Block.

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

Include dependency graph for mcb.h:



Macros

- #define MAX_HEAP_SIZE 5000

Functions

init_heap

Allocates all the memory for MPX

Parameters

size	*Size of heap in bytes*

- void init_heap (u32int size)

mcb_allocate

Allocates a memory block

Parameters

mem_size	*The MCB size to be allocated*

Returns

Address to allocated MCB
NULL if not enough space in free memory found

- void * mcb_allocate (u32int mem_size)

show_mcb

Displays the allocated or free memory block's address, previous and next pointers, and block's size.

Parameters

mcb_ptr	*MCB Pointer*

- void show_mcb (struct mcb *mcb_ptr)

show_free_mcb

Displays all the free memory

- void show_free_mcb ()

show_allocated_mcb

Displays all the allocated MCBs

- void show_allocated_mcb ()

show_all_mcb

Displays all the free and allocated memory

- void show_all_mcb ()

is_mcb_empty

Checks if the heap is empty

Returns

0 or 1 (true or false)

- int is_mcb_empty ()

mcb_free_mpx

Calls mcb_free to free memory block, used as parameter for sys_set_free in kmain.c

Parameters

mem_ptr	*Memory Pointer*

Returns

0

- int mcb_free_mpx (void ∗mem_ptr)

mcb_allocate_mpx

Calls mcb_allocate to allocate memory block, used as parameter for sys_set_malloc in kmain.c

Parameters

size	*Size of block in bytes to allocate*

Returns

Address of allocated MCB

- u32int mcb_allocate_mpx (u32int size)

mcb_allocate_mpx2

MCB allocate MPX

Parameters

mem_size	*Block size to allocate*
name	*name of the pcb process*

Returns

Address pointer to allocated memory only used for testing in commhand for module R5

- void ∗ mcb_allocate_mpx2 (u32int size, const char ∗name)

show_mcb_main.

The function of show MCB for commhand.

Parameters

argc	*The number of tokens found.*
argv	*The array of tokens.*

Returns

0

- int show_mcb_main (int argc, char **argv)

shutdown_mcb.

Shutdown the pcb during the shutdown procedure.

Returns

0

- void shutdown_mcb ()

Variables

- u32int start_of_memory

  Global variable labeling start of memory.

5.20.1 Detailed Description

Memory Control Block.

Author

Thunder Krakens

Date

April 8th, 2016

Version

R5

5.20.2 Macro Definition Documentation

5.20.2.1 #define MAX_HEAP_SIZE 5000

5.20.3 Function Documentation

5.20.3.1 void init_heap (u32int *size*)

5.20.3.2 int is_mcb_empty ( )

5.20.3.3 void* mcb_allocate (u32int *mem_size*)

5.20.3.4 u32int mcb_allocate_mpx (u32int *size*)

5.20.3.5 void* mcb_allocate_mpx2 (u32int *size,* const char * *name*)

5.20.3.6 int mcb_free_mpx (void * *mem_ptr*)

5.20.3.7 void show_all_mcb ( )

5.20.3.8 void show_allocated_mcb ( )

5.20.3.9 void show_free_mcb ( )

5.20.3.10 void show_mcb (struct mcb * *mcb_ptr*)

5.20.3.11 int show_mcb_main (int *argc,* char ** *argv*)

5.20.3.12 void shutdown_mcb ( )

5.20.4 Variable Documentation

5.20.4.1 u32int start_of_memory

Global variable labeling start of memory.

5.21 modules/r6/ansi.h File Reference

Macros

- #define T_RESET ""
- #define T_BOLD ""
- #define T_BOLD_OFF ""
- #define T_ITCS ""
- #define T_ITCS_OFF ""
- #define T_NRM ""
- #define T_RED ""
- #define T_CYAN ""
- #define T_WHT ""
- #define B_NRM ""
- #define B_CYAN ""
- #define T_DIR ""
- #define T_DIR_OFF ""

5.21.1 Macro Definition Documentation

5.21.1.1 #define B_CYAN ""

5.21.1.2 #define B_NRM ""

5.21.1.3 #define T_BOLD ""

5.21.1.4 #define T_BOLD_OFF ""

5.21.1.5 #define T_CYAN ""

5.21.1.6 #define T_DIR ""

5.21.1.7 #define T_DIR_OFF ""

5.21.1.8 #define T_ITCS ""

5.21.1.9 #define T_ITCS_OFF ""

5.21.1.10 #define T_NRM ""

5.21.1.11 #define T_RED ""

5.21.1.12 #define T_RESET ""

5.21.1.13 #define T_WHT ""

5.22 modules/r6/disk_file_manager.h File Reference

Functions

- void type_file (struct dir_entry_info *file_entry_ptr)
- void extract_file (struct dir_entry_info *file_entry_ptr, const char *out_file_path)
- void import_file (const char *in_file_path, struct dir_entry_info *dest_dir)
- void move_file (struct dir_entry_info *file_entry, struct dir_entry_info *dest_dir)

5.22.1 Function Documentation

5.22.1.1 void extract_file (struct dir_entry_info * *file_entry_ptr,* const char * *out_file_path*)

5.22.1.2 void import_file (const char * *in_file_path,* struct dir_entry_info * *dest_dir*)

5.22.1.3 void move_file (struct dir_entry_info * *file_entry,* struct dir_entry_info * *dest_dir*)

5.22.1.4 void type_file (struct dir_entry_info * *file_entry_ptr*)

5.23 modules/r6/disk_folder_manager.h File Reference

Macros

- #define FOLDER_STACK_SIZE 1000

Functions

- void folder_manager_init ()
- void push_folder (struct dir_entry_info *child_folder_ptr)
- void pop_folder ()
- void print_dir_entry_info (struct dir_entry_info *folder_ptr)
- void list_dir_entry_report ()
- void list_dir_entry_short ()

- void print_curr_path ()
- void rename_entry (struct dir_entry_info *folder_ptr, const char *new_name)
- struct dir_entry_info * get_entry_simple (const char *nameStr)
- struct dir_entry_info * get_entry (char *full_path)
- void change_dir (char *full_path)

5.23.1 Macro Definition Documentation

5.23.1.1 #define FOLDER_STACK_SIZE 1000

5.23.2 Function Documentation

5.23.2.1 void change_dir (char * *full_path*)

5.23.2.2 void folder_manager_init ( )

5.23.2.3 struct dir_entry_info* get_entry (char * *full_path*)

5.23.2.4 struct dir_entry_info* get_entry_simple (const char * *nameStr*)

5.23.2.5 void list_dir_entry_report ( )

5.23.2.6 void list_dir_entry_short ( )

5.23.2.7 void pop_folder ( )

5.23.2.8 void print_curr_path ( )

5.23.2.9 void print_dir_entry_info (struct dir_entry_info * *folder_ptr*)

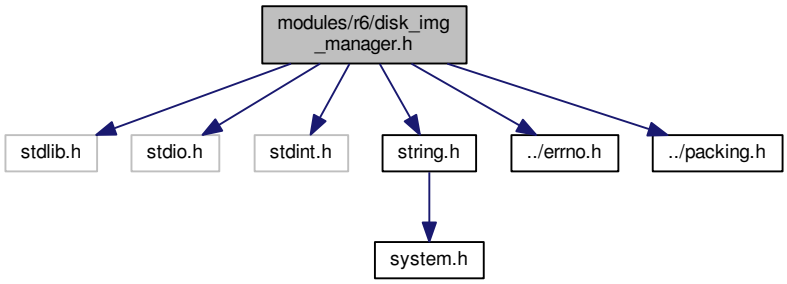
5.23.2.10 void push_folder (struct dir_entry_info * *child_folder_ptr*)

5.23.2.11 void rename_entry (struct dir_entry_info * *folder_ptr,* const char * *new_name*)

5.24 modules/r6/disk_img_manager.h File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <stdint.h>
#include <string.h>
#include "../errno.h"
#include "../packing.h"
```

Include dependency graph for disk_img_manager.h:



Macros

- #define ATTRIBUTE_READ 0x01
- #define ATTRIBUTE_HIDD 0x02
- #define ATTRIBUTE_SYST 0x04
- #define ATTRIBUTE_VOLL 0x08
- #define ATTRIBUTE_SUBD 0x10
- #define ATTRIBUTE_ARCH 0x20
- #define ATTRIBUTE_UUS1 0x40
- #define ATTRIBUTE_UUS2 0x80

Functions

- PACKED (struct img_boot_sector{uint8_t ignore1[11];uint16_t byte_per_sector;uint8_t sector_per_cluster;uint16_t reserved_sec_num;uint8_t fat_copies_num;uint16_t root_dir_max_num;uint16_t sec_num;uint8_t ignore2;uint16_t sec_per_fat_num;uint16_t sec_per_track;uint16_t head_num;uint32_t ignore3;uint32_t total_sec_fat32;uint16_t ignore4;uint8_t boot_sign;uint32_t vol_id;uint8_t vol_label[11];uint8_t file_sys_type[8];uint8_t ignore5[450];})
- PACKED (struct dir_entry_info{uint8_t file_name[8];uint8_t extension[3];uint8_t attributes;uint16_t reserved;uint16_t create_time;uint16_t create_date;uint16_t last_acc_date;uint16_t ignore1;uint16_t last_wri_time;uint16_t last_wri_date;uint16_t first_log_clu;uint32_t file_size;})
- PACKED (struct data_sector{uint8_t data[512];})
- error_t load_image_file (const char *path_to_file)
- void print_boot_sec_info (const struct img_boot_sector *boot_sec)
- void clean_buffers ()
- void ch_arr_to_str (char *dest, const char *src, const unsigned int size)
- void str_to_ch_arr (char *dest, const char *src, const unsigned int size)
- uint8_t * get_fat_val (const unsigned int copy_index, const unsigned int byte_index)
- void fat (uint16_t *fat_val, const uint16_t cluster_index)

  get the specific meanningful 12-bit value from the FAT array.
- void * get_data_ptr (const uint16_t data_area_sec_index)
- void write_fat (const uint16_t fat_val, const uint16_t cluster_index)
- uint16_t find_unused_fat ()

Variables

- struct img_boot_sector * boot_sec
- struct dir_entry_info * root_dir_file_arr
- struct data_sector * data_area

5.24.1 Macro Definition Documentation

5.24.1.1 #define ATTRIBUTE_ARCH 0x20

5.24.1.2 #define ATTRIBUTE_HIDD 0x02

5.24.1.3 #define ATTRIBUTE_READ 0x01

5.24.1.4 #define ATTRIBUTE_SUBD 0x10

5.24.1.5 #define ATTRIBUTE_SYST 0x04

5.24.1.6 #define ATTRIBUTE_UUS1 0x40

5.24.1.7 #define ATTRIBUTE_UUS2 0x80

5.24.1.8 #define ATTRIBUTE_VOLL 0x08

5.24.2 Function Documentation

5.24.2.1 void ch_arr_to_str (char * *dest,* const char * *src,* const unsigned int *size*)

5.24.2.2 void clean_buffers ( )

5.24.2.3 void fat (uint16_t * *fat_val,* const uint16_t *cluster_index*)

get the specific meanningful 12-bit value from the FAT array.

As said in page 4 of FAT12 File System Format Information. Convert 2*1 byte to 12 bit.

5.24.2.4 uint16_t find_unused_fat ( )

5.24.2.5 void* get_data_ptr (const uint16_t *data_area_sec_index*)

5.24.2.6 uint8_t* get_fat_val (const unsigned int *copy_index,* const unsigned int *byte_index*)

5.24.2.7 error_t load_image_file (const char * *path_to_file*)

5.24.2.8 PACKED (struct img_boot_sector{uint8_t ignore1[11];uint16_t byte_per_sector;uint8_t sector_per_cluster;uint16_t reserved_sec_num;uint8_t fat_copies_num;uint16_t root_dir_max_num;uint16_t sec_num;uint8_t ignore2;uint16_t sec_per_fat_num;uint16_t sec_per_track;uint16_t head_num;uint32_t ignore3;uint32_t total_sec_fat32;uint16_t ignore4;uint8_t boot_sign;uint32_t vol_id;uint8_t vol_label[11];uint8_t file_sys_type[8];uint8_t ignore5[450];})

5.24.2.9 PACKED (struct dir_entry_info{uint8_t file_name[8];uint8_t extension[3];uint8_t attributes;uint16_t reserved;uint16_t create_time;uint16_t create_date;uint16_t last_acc_date;uint16_t ignore1;uint16_t last_wri_time;uint16_t last_wri_date;uint16_t first_log_clu;uint32_t file_size;})

5.24.2.10 PACKED (struct data_sector{uint8_t data[512];})

5.24.2.11 void print_boot_sec_info (const struct img_boot_sector * *boot_sec*)

5.24.2.12 void str_to_ch_arr (char * *dest,* const char * *src,* const unsigned int *size*)

5.24.2.13 void write_fat (const uint16_t *fat_val,* const uint16_t *cluster_index*)

5.24.3 Variable Documentation

5.24.3.1 struct img_boot_sector* boot_sec

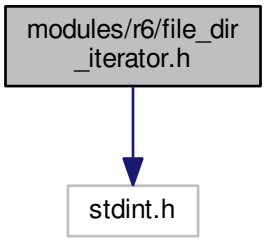
5.24.3.2 struct data_sector* data_area

5.24.3.3 struct dir_entry_info* root_dir_file_arr

5.25 modules/r6/file_dir_iterator.h File Reference

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

Include dependency graph for file_dir_iterator.h:



Macros

- #define ROOT_DIR_SEC_INDEX 0

Functions

- struct file_itr * init_file_itr (const uint16_t sec_index)
- void fitr_begin (struct file_itr *itr_ptr)
- uint8_t fitr_end (struct file_itr *itr_ptr)
- void fitr_next (struct file_itr *itr_ptr)

- struct data_sector * fitr_get (struct file_itr *itr_ptr)
- struct dir_itr * init_dir_itr (const uint16_t sec_index)
- void ditr_set_filter (struct dir_itr *itr_ptr, uint8_t attr_filter)
- void ditr_begin (struct dir_itr *itr_ptr)
- void ditr_set_find_unused (struct dir_itr *itr_ptr)
- uint8_t ditr_end (struct dir_itr *itr_ptr)
- void ditr_next (struct dir_itr *itr_ptr)
- struct dir_entry_info * ditr_get (struct dir_itr *itr_ptr)
- struct img_writer * init_img_writer (struct dir_entry_info *entry_ptr)
- void iw_write (struct img_writer *writer_ptr, const struct data_sector *data)

5.25.1 Macro Definition Documentation

5.25.1.1 #define ROOT_DIR_SEC_INDEX 0

5.25.2 Function Documentation

5.25.2.1 void ditr_begin (struct dir_itr * *itr_ptr*)

5.25.2.2 uint8_t ditr_end (struct dir_itr * *itr_ptr*)

5.25.2.3 struct dir_entry_info* ditr_get (struct dir_itr * *itr_ptr*)

5.25.2.4 void ditr_next (struct dir_itr * *itr_ptr*)

5.25.2.5 void ditr_set_filter (struct dir_itr * *itr_ptr,* uint8_t *attr_filter*)

5.25.2.6 void ditr_set_find_unused (struct dir_itr * *itr_ptr*)

5.25.2.7 void fitr_begin (struct file_itr * *itr_ptr*)

5.25.2.8 uint8_t fitr_end (struct file_itr * *itr_ptr*)

5.25.2.9 struct data_sector* fitr_get (struct file_itr * *itr_ptr*)

5.25.2.10 void fitr_next (struct file_itr * *itr_ptr*)

5.25.2.11 struct dir_itr* init_dir_itr (const uint16_t *sec_index*)

5.25.2.12 struct file_itr* init_file_itr (const uint16_t *sec_index*)

5.25.2.13 struct img_writer* init_img_writer (struct dir_entry_info * *entry_ptr*)

5.25.2.14 void iw_write (struct img_writer * *writer_ptr,* const struct data_sector * *data*)

Index