

# MPX Thunder Krakens

## R4

Generated by Doxygen 1.8.6

Fri Mar 18 2016 02:02:45



# Contents

<b>1</b>	<b>Main Page</b>	<b>1</b>
<b>2</b>	<b>Data Structure Index</b>	<b>3</b>
2.1	Data Structures . . . . .	3
<b>3</b>	<b>File Index</b>	<b>5</b>
3.1	File List . . . . .	5
<b>4</b>	<b>Data Structure Documentation</b>	<b>7</b>
4.1	function_name Struct Reference . . . . .	7
4.1.1	Detailed Description . . . . .	7
4.1.2	Field Documentation . . . . .	7
4.1.2.1	function . . . . .	7
4.1.2.2	help . . . . .	7
4.1.2.3	nameStr . . . . .	7
4.1.2.4	usage . . . . .	8
4.2	param Struct Reference . . . . .	8
4.2.1	Detailed Description . . . . .	8
4.2.2	Field Documentation . . . . .	8
4.2.2.1	device_id . . . . .	8
4.2.2.2	op_code . . . . .	8
4.3	pcb_queue Struct Reference . . . . .	8
4.3.1	Detailed Description . . . . .	9
4.3.2	Field Documentation . . . . .	9
4.3.2.1	count . . . . .	9
4.3.2.2	head . . . . .	9
4.3.2.3	tail . . . . .	9
4.4	pcb_struct Struct Reference . . . . .	10
4.4.1	Detailed Description . . . . .	10

4.4.2	Field Documentation	10
4.4.2.1	class	10
4.4.2.2	is_suspended	11
4.4.2.3	name	11
4.4.2.4	next	11
4.4.2.5	prev	11
4.4.2.6	priority	11
4.4.2.7	running_state	11
4.4.2.8	stack_base	11
4.4.2.9	stack_top	11
<b>5</b>	<b>File Documentation</b>	<b>13</b>
5.1	documentation/mainpage.dox File Reference	13
5.2	include/core/serial.h File Reference	13
5.2.1	Detailed Description	14
5.2.2	Macro Definition Documentation	14
5.2.2.1	COM1	14
5.2.2.2	COM2	14
5.2.2.3	COM3	14
5.2.2.4	COM4	14
5.2.2.5	USER_INPUT_BUFFER_SIZE	14
5.2.2.6	WithEcho	14
5.2.2.7	WithoutEcho	15
5.2.3	Function Documentation	15
5.2.3.1	get_input_line	15
5.2.3.2	init_serial	15
5.2.3.3	serial_print	15
5.2.3.4	serial_println	15
5.2.3.5	set_serial_in	15
5.2.3.6	set_serial_out	15
5.3	include/string.h File Reference	15
5.3.1	Detailed Description	19
5.3.2	Function Documentation	19
5.3.2.1	atoi	20
5.3.2.2	isspace	20
5.3.2.3	memset	21
5.3.2.4	printf	21

5.3.2.5	<a href="#">sprintf</a>	22
5.3.2.6	<a href="#">strcat</a>	22
5.3.2.7	<a href="#">strcmp</a>	23
5.3.2.8	<a href="#">strcpy</a>	24
5.3.2.9	<a href="#">strlen</a>	24
5.3.2.10	<a href="#">strtok</a>	25
5.4	<a href="#">lib/string.c File Reference</a>	25
5.4.1	<a href="#">Detailed Description</a>	29
5.4.2	<a href="#">Function Documentation</a>	29
5.4.2.1	<a href="#">atoi</a>	29
5.4.2.2	<a href="#">isspace</a>	30
5.4.2.3	<a href="#">memset</a>	30
5.4.2.4	<a href="#">printf</a>	31
5.4.2.5	<a href="#">sprintf</a>	31
5.4.2.6	<a href="#">strcat</a>	31
5.4.2.7	<a href="#">strcmp</a>	32
5.4.2.8	<a href="#">strcpy</a>	33
5.4.2.9	<a href="#">strlen</a>	33
5.4.2.10	<a href="#">strtok</a>	34
5.5	<a href="#">modules/errno.h File Reference</a>	34
5.5.1	<a href="#">Detailed Description</a>	35
5.5.2	<a href="#">Macro Definition Documentation</a>	35
5.5.2.1	<a href="#">E_EMPTPCB</a>	35
5.5.2.2	<a href="#">E_FREEMEM</a>	35
5.5.2.3	<a href="#">E_INVPARA</a>	35
5.5.2.4	<a href="#">E_INVSTRF</a>	35
5.5.2.5	<a href="#">E_INVUSRI</a>	35
5.5.2.6	<a href="#">E_NOERROR</a>	35
5.5.2.7	<a href="#">E_NULL_PTR</a>	35
5.5.2.8	<a href="#">E_PROGERR</a>	35
5.5.3	<a href="#">Typedef Documentation</a>	35
5.5.3.1	<a href="#">error_t</a>	36
5.6	<a href="#">modules/mpx_supt.c File Reference</a>	36
5.6.1	<a href="#">Function Documentation</a>	36
5.6.1.1	<a href="#">get_op_code</a>	36
5.6.1.2	<a href="#">idle</a>	37
5.6.1.3	<a href="#">mpx_init</a>	37

5.6.1.4	<a href="#">sys_alloc_mem</a>	37
5.6.1.5	<a href="#">sys_free_mem</a>	37
5.6.1.6	<a href="#">sys_req</a>	38
5.6.1.7	<a href="#">sys_set_free</a>	38
5.6.1.8	<a href="#">sys_set_malloc</a>	38
5.6.2	<a href="#">Variable Documentation</a>	38
5.6.2.1	<a href="#">current_module</a>	38
5.6.2.2	<a href="#">params</a>	38
5.6.2.3	<a href="#">student_free</a>	38
5.6.2.4	<a href="#">student_malloc</a>	38
5.7	<a href="#">modules/mpx_supt.h File Reference</a>	38
5.7.1	<a href="#">Detailed Description</a>	41
5.7.2	<a href="#">Macro Definition Documentation</a>	41
5.7.2.1	<a href="#">EXIT</a>	41
5.7.2.2	<a href="#">IDLE</a>	41
5.7.2.3	<a href="#">MODULE_R1</a>	41
5.7.2.4	<a href="#">MODULE_R2</a>	41
5.7.2.5	<a href="#">MODULE_R3</a>	41
5.7.2.6	<a href="#">MODULE_R4</a>	41
5.7.2.7	<a href="#">MODULE_R5</a>	41
5.7.2.8	<a href="#">READ</a>	41
5.7.2.9	<a href="#">WRITE</a>	41
5.7.3	<a href="#">Function Documentation</a>	41
5.7.3.1	<a href="#">get_op_code</a>	41
5.7.3.2	<a href="#">idle</a>	42
5.7.3.3	<a href="#">mpx_init</a>	42
5.7.3.4	<a href="#">sys_alloc_mem</a>	42
5.7.3.5	<a href="#">sys_free_mem</a>	42
5.7.3.6	<a href="#">sys_req</a>	43
5.7.3.7	<a href="#">sys_set_free</a>	43
5.7.3.8	<a href="#">sys_set_malloc</a>	43
5.7.4	<a href="#">Variable Documentation</a>	43
5.7.4.1	<a href="#">__attribute__</a>	43
5.8	<a href="#">modules/r1/r1.c File Reference</a>	43
5.8.1	<a href="#">Detailed Description</a>	46
5.8.2	<a href="#">Macro Definition Documentation</a>	46
5.8.2.1	<a href="#">COMPLETION</a>	46

5.8.2.2	MAX_ARGC	46
5.8.2.3	MAX_HISTORY	46
5.8.2.4	MOD_VERSION	46
5.8.3	Enumeration Type Documentation	46
5.8.3.1	CommandPaserStat	46
5.8.4	Function Documentation	46
5.8.4.1	__attribute__	46
5.8.4.2	command_line_parser	47
5.8.4.3	commhand	47
5.8.4.4	help_usages	47
5.8.4.5	print_help	47
5.8.5	Variable Documentation	48
5.8.5.1	DoubleQuoteWriting	48
5.8.5.2	NormalWriting	48
5.8.5.3	NotWriting	48
5.8.5.4	SingleQuoteWriting	48
5.9	modules/r1/r1.h File Reference	48
5.9.1	Detailed Description	50
5.9.2	Macro Definition Documentation	51
5.9.2.1	GETDATE	51
5.9.2.2	GETTIME	51
5.9.2.3	HELP	51
5.9.2.4	LOADR3	51
5.9.2.5	NUM_MPX_FUNCTIONS	51
5.9.2.6	NUM_OF_FUNCTIONS	51
5.9.2.7	POS_OF_MPX	51
5.9.2.8	POS_OF_PCB	51
5.9.2.9	RESUMEPCB	51
5.9.2.10	SETDATE	51
5.9.2.11	SETPCBPRIO	51
5.9.2.12	SETTIME	51
5.9.2.13	SHOWPCB	51
5.9.2.14	SHUTDOWN	51
5.9.2.15	SUSPDPCB	51
5.9.2.16	VERSION	51
5.9.2.17	YIELD	51
5.9.3	Enumeration Type Documentation	51

5.9.3.1	<code>comm_type</code>	51
5.9.4	Function Documentation	51
5.9.4.1	<code>__attribute__</code>	51
5.9.4.2	<code>command_line_parser</code>	52
5.9.4.3	<code>commhand</code>	52
5.9.4.4	<code>help_usages</code>	52
5.9.4.5	<code>print_help</code>	52
5.9.5	Variable Documentation	53
5.9.5.1	<code>help</code>	53
5.9.5.2	<code>mpx</code>	53
5.9.5.3	<code>pcb</code>	53
5.10	<code>modules/r1/sys_clock.c</code> File Reference	53
5.10.1	Detailed Description	57
5.10.2	Macro Definition Documentation	57
5.10.2.1	<code>RTC_INDEX_DAY_MONTH</code>	57
5.10.2.2	<code>RTC_INDEX_DAY_WEEK</code>	57
5.10.2.3	<code>RTC_INDEX_HOUR</code>	57
5.10.2.4	<code>RTC_INDEX_HOUR_ALARM</code>	57
5.10.2.5	<code>RTC_INDEX_MINUTE</code>	57
5.10.2.6	<code>RTC_INDEX_MINUTE_ALARM</code>	57
5.10.2.7	<code>RTC_INDEX_MONTH</code>	58
5.10.2.8	<code>RTC_INDEX_SECOND</code>	58
5.10.2.9	<code>RTC_INDEX_SECOND_ALARM</code>	58
5.10.2.10	<code>RTC_INDEX_YEAR</code>	58
5.10.3	Function Documentation	58
5.10.3.1	<code>get_date</code>	58
5.10.3.2	<code>get_date_main</code>	58
5.10.3.3	<code>get_time</code>	59
5.10.3.4	<code>get_time_main</code>	59
5.10.3.5	<code>set_date</code>	59
5.10.3.6	<code>set_date_main</code>	60
5.10.3.7	<code>set_date_str</code>	60
5.10.3.8	<code>set_time</code>	61
5.10.3.9	<code>set_time_main</code>	62
5.10.3.10	<code>set_time_str</code>	62
5.11	<code>modules/r1/sys_clock.h</code> File Reference	63
5.11.1	Detailed Description	66



5.11.2	Function Documentation	66
5.11.2.1	get_date	66
5.11.2.2	get_date_main	67
5.11.2.3	get_time	67
5.11.2.4	get_time_main	68
5.11.2.5	set_date	68
5.11.2.6	set_date_main	69
5.11.2.7	set_date_str	69
5.11.2.8	set_time	70
5.11.2.9	set_time_main	71
5.11.2.10	set_time_str	71
5.12	modules/r2/pcb.c File Reference	72
5.12.1	Detailed Description	78
5.12.2	Enumeration Type Documentation	78
5.12.2.1	process_state	78
5.12.2.2	process_suspended	78
5.12.3	Function Documentation	78
5.12.3.1	__attribute__	78
5.12.3.2	allocate_pcb	78
5.12.3.3	block_pcb	79
5.12.3.4	find_pcb	79
5.12.3.5	free_pcb	80
5.12.3.6	get_running_process	81
5.12.3.7	get_stack_base	81
5.12.3.8	get_stack_top	81
5.12.3.9	insert_pcb	81
5.12.3.10	pcb_init	81
5.12.3.11	remove_pcb	82
5.12.3.12	resume_pcb	82
5.12.3.13	save_running_process	82
5.12.3.14	set_pcb_priority	83
5.12.3.15	setup_pcb	83
5.12.3.16	show_all_processes	84
5.12.3.17	show_blocked_processes	84
5.12.3.18	show_pcb	85
5.12.3.19	show_ready_processes	85
5.12.3.20	shutdown_pcb	86

5.12.3.21	<a href="#">suspend_pcb</a>	86
5.12.3.22	<a href="#">unblock_pcb</a>	87
5.12.4	<a href="#">Variable Documentation</a>	87
5.12.4.1	<a href="#">__attribute__</a>	87
5.12.4.2	<a href="#">blocked</a>	87
5.12.4.3	<a href="#">false</a>	87
5.12.4.4	<a href="#">ready</a>	87
5.12.4.5	<a href="#">running</a>	87
5.12.4.6	<a href="#">true</a>	87
5.13	<a href="#">modules/r2/pcb.h File Reference</a>	87
5.13.1	<a href="#">Detailed Description</a>	93
5.13.2	<a href="#">Macro Definition Documentation</a>	94
5.13.2.1	<a href="#">SIZE_OF_PCB_NAME</a>	94
5.13.2.2	<a href="#">SIZE_OF_STACK</a>	94
5.13.3	<a href="#">Enumeration Type Documentation</a>	94
5.13.3.1	<a href="#">process_class</a>	94
5.13.4	<a href="#">Function Documentation</a>	94
5.13.4.1	<a href="#">__attribute__</a>	94
5.13.4.2	<a href="#">allocate_pcb</a>	94
5.13.4.3	<a href="#">block_pcb</a>	95
5.13.4.4	<a href="#">find_pcb</a>	95
5.13.4.5	<a href="#">free_pcb</a>	96
5.13.4.6	<a href="#">get_running_process</a>	97
5.13.4.7	<a href="#">get_stack_base</a>	97
5.13.4.8	<a href="#">get_stack_top</a>	97
5.13.4.9	<a href="#">insert_pcb</a>	97
5.13.4.10	<a href="#">pcb_init</a>	97
5.13.4.11	<a href="#">remove_pcb</a>	98
5.13.4.12	<a href="#">resume_pcb</a>	98
5.13.4.13	<a href="#">save_running_process</a>	98
5.13.4.14	<a href="#">set_pcb_priority</a>	99
5.13.4.15	<a href="#">setup_pcb</a>	99
5.13.4.16	<a href="#">show_all_processes</a>	100
5.13.4.17	<a href="#">show_blocked_processes</a>	100
5.13.4.18	<a href="#">show_pcb</a>	101
5.13.4.19	<a href="#">show_ready_processes</a>	101
5.13.4.20	<a href="#">shutdown_pcb</a>	102

5.13.4.21	suspend_pcb	102
5.13.4.22	unblock_pcb	103
5.13.5	Variable Documentation	103
5.13.5.1	pcb_class_app	103
5.13.5.2	pcb_class_sys	103
5.14	modules/r2/pcb_comm.c File Reference	103
5.14.1	Detailed Description	105
5.14.2	Function Documentation	106
5.14.2.1	resume_pcb_main	106
5.14.2.2	set_pcb_priority_main	107
5.14.2.3	show_pcb_main	107
5.14.2.4	suspend_pcb_main	108
5.15	modules/r2/pcb_comm.h File Reference	108
5.15.1	Detailed Description	111
5.15.2	Function Documentation	111
5.15.2.1	block_pcb_main	111
5.15.2.2	create_pcb_main	111
5.15.2.3	delete_pcb_main	111
5.15.2.4	resume_pcb_main	111
5.15.2.5	set_pcb_priority_main	112
5.15.2.6	show_pcb_main	112
5.15.2.7	suspend_pcb_main	113
5.15.2.8	unblock_pcb_main	113



# 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 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:

<a href="#">function_name</a>	A structure to represent each function . . . . .	7
<a href="#">param</a>	A structure to represent interrupt . . . . .	8
<a href="#">pcb_queue</a>	Queue structure that will store PCBs . . . . .	8
<a href="#">pcb_struct</a>	Struct that will describe PCB Processes . . . . .	10





## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

include/ <a href="#">string.h</a>	Many usefull functions that used for handling string . . . . .	15
include/core/ <a href="#">serial.h</a>	Serial - Header . . . . .	13
lib/ <a href="#">string.c</a>	Many usefull functions that used for handling string . . . . .	25
modules/ <a href="#">errno.h</a>	This file contains the type of errors. The error can be from invalid paramter passed to a function, or invalid input format . . . . .	34
modules/ <a href="#">mpx_supt.c</a>		36
modules/ <a href="#">mpx_supt.h</a>	MPX System Supplementaries . . . . .	38
modules/r1/ <a href="#">r1.c</a>	The commandhandler and functions associations for Module R1 . . . . .	43
modules/r1/ <a href="#">r1.h</a>	The command handler and functions associations for Module R1 . . . . .	48
modules/r1/ <a href="#">sys_clock.c</a>	The main file that manipulates and controls the system's clock . . . . .	53
modules/r1/ <a href="#">sys_clock.h</a>	The main file that manipulates and controls the system's clock . . . . .	63
modules/r2/ <a href="#">pcb.c</a>	The Process Control Block . . . . .	72
modules/r2/ <a href="#">pcb.h</a>	The Process Control Block . . . . .	87
modules/r2/ <a href="#">pcb_comm.c</a>	The main functions that manipulate the PCB . . . . .	103
modules/r2/ <a href="#">pcb_comm.h</a>	The main functions that manipulate the PCB . . . . .	108



## Chapter 4

# Data Structure Documentation

### 4.1 function\_name Struct Reference

A structure to represent each function.

#### Data Fields

- char \* [nameStr](#)  
*fuction's name*
- int(\* [function](#) )(int argc, char \*\*argv)  
*the function*
- char \* [usage](#)  
*function's usage or use cases*
- char \* [help](#)  
*function's help information*

#### 4.1.1 Detailed Description

A structure to represent each function.

#### 4.1.2 Field Documentation

##### 4.1.2.1 int(\* function\_name::function)(int argc, char \*\*argv)

the function

##### 4.1.2.2 char\* function\_name::help

function's help information

##### 4.1.2.3 char\* function\_name::nameStr

fuction's name

#### 4.1.2.4 `char* function_name::usage`

function's usage or use cases

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

- [modules/r1/r1.c](#)

## 4.2 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*

### 4.2.1 Detailed Description

A structure to represent interrupt.

### 4.2.2 Field Documentation

#### 4.2.2.1 int param::device\_id

interrupt's device

#### 4.2.2.2 int param::op\_code

interrupt's operation

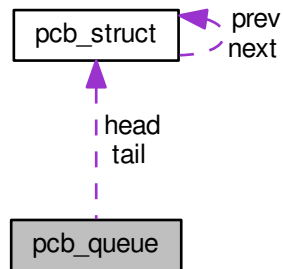
The documentation for this struct was generated from the following file:

- [modules/mpx\\_supt.h](#)

## 4.3 pcb\_queue Struct Reference

Queue structure that will store PCBs.

Collaboration diagram for pcb\_queue:



## Data Fields

- int `count`  
*The length of the queue.*
- struct `pcb_struct * head`  
*Pointer to the start/head of the queue.*
- struct `pcb_struct * tail`  
*Pointer to the end/tail of the queue.*

### 4.3.1 Detailed Description

Queue structure that will store PCBs.

### 4.3.2 Field Documentation

#### 4.3.2.1 int `pcb_queue::count`

The length of the queue.

#### 4.3.2.2 struct `pcb_struct* pcb_queue::head`

Pointer to the start/head of the queue.

#### 4.3.2.3 struct `pcb_struct* pcb_queue::tail`

Pointer to the end/tail of the queue.

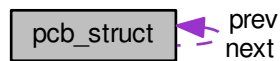
The documentation for this struct was generated from the following file:

- `modules/r2/pcb.c`

## 4.4 pcb\_struct Struct Reference

Struct that will describe PCB Processes.

Collaboration diagram for pcb\_struct:



### Data Fields

- char [name](#) [[SIZE\\_OF\\_PCB\\_NAME](#)]  
*PCB's name.*
- enum [process\\_class](#) [class](#)  
*PCB's class is an application or system process.*
- unsigned char [priority](#)  
*PCB's priority an integer between 0 and 9.*
- enum [process\\_state](#) [running\\_state](#)  
*PCB's states are ready, running, or blocked.*
- enum [process\\_suspended](#) [is\\_suspended](#)  
*PCB process is either suspended or not suspended.*
- unsigned char \* [stack\\_top](#)  
*Pointer to top of the stack.*
- unsigned char \* [stack\\_base](#)  
*Pointer to base of the stack.*
- struct [pcb\\_struct](#) \* [prev](#)  
*Pointer to the previous PCB in the queue.*
- struct [pcb\\_struct](#) \* [next](#)  
*Pointer to the next PCB in the queue.*

### 4.4.1 Detailed Description

Struct that will describe PCB Processes.

### 4.4.2 Field Documentation

#### 4.4.2.1 enum [process\\_class](#) [pcb\\_struct::class](#)

PCB's class is an application or system process.

#### 4.4.2.2 enum process\_suspended pcb\_struct::is\_suspended

PCB process is either suspended or not suspended.

#### 4.4.2.3 char pcb\_struct::name[SIZE\_OF\_PCB\_NAME]

PCB's name.

#### 4.4.2.4 struct pcb\_struct\* pcb\_struct::next

Pointer to the next PCB in the queue.

#### 4.4.2.5 struct pcb\_struct\* pcb\_struct::prev

Pointer to the previous PCB in the queue.

#### 4.4.2.6 unsigned char pcb\_struct::priority

PCB's priority an integer between 0 and 9.

Processes with higher priority values execute before lower priority processes.

#### 4.4.2.7 enum process\_state pcb\_struct::running\_state

PCB's states are ready, running, or blocked.

#### 4.4.2.8 unsigned char\* pcb\_struct::stack\_base

Pointer to base of the stack.

#### 4.4.2.9 unsigned char\* pcb\_struct::stack\_top

Pointer to top of the stack.

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

- [modules/r2/pcb.c](#)





## Chapter 5

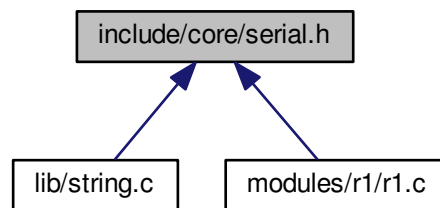
# File Documentation

### 5.1 documentation/mainpage.dox File Reference

### 5.2 include/core/serial.h File Reference

Serial - Header.

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



### 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	<i>The pointer to the buffer where store the user's input.</i>
buffer_size	<i>The size of that buffer.</i>
bWithEcho	<i>With echo or not</i>

#### **Returns**

*VOID*

- void [get\\_input\\_line](#) (char \*buffer, const int bWithEcho)

## 5.2.1 Detailed Description

Serial - Header.

#### **Author**

Thunder Krakens

#### **Date**

February 2nd, 2016

#### **Version**

R1

## 5.2.2 Macro Definition Documentation

5.2.2.1 **#define COM1 0x3f8**

5.2.2.2 **#define COM2 0x2f8**

5.2.2.3 **#define COM3 0x3e8**

5.2.2.4 **#define COM4 0x2e8**

5.2.2.5 **#define USER\_INPUT\_BUFFER\_SIZE 100**

5.2.2.6 **#define WithEcho 1**

5.2.2.7 `#define WithoutEcho 0`

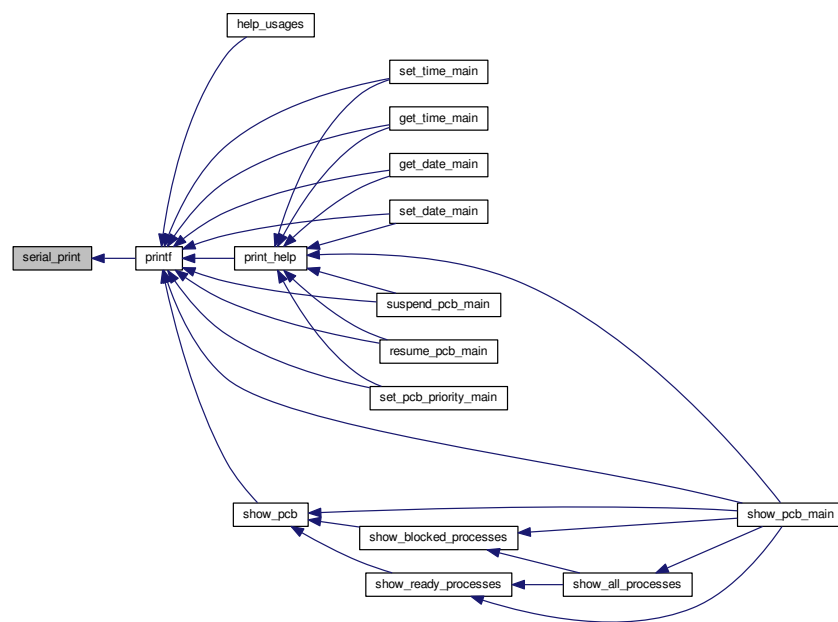
## 5.2.3 Function Documentation

5.2.3.1 `void get_input_line ( char * buffer, const int bWithEcho )`

5.2.3.2 `int init_serial ( int device )`

5.2.3.3 `int serial_print ( const char * msg )`

Here is the caller graph for this function:



5.2.3.4 `int serial_println ( const char * msg )`

5.2.3.5 `int set_serial_in ( int device )`

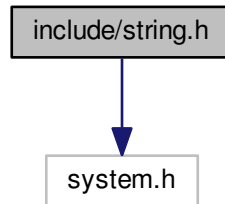
5.2.3.6 `int set_serial_out ( int device )`

## 5.3 include/string.h File Reference

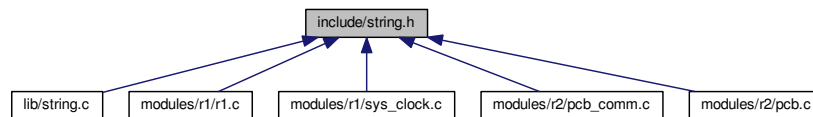
Many usefull functions that used for handling string.

```
#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,...)

### 5.3.1 Detailed Description

Many usefull functions that used for handling string.

**Author**

Thunder Krakens

**Date**

February 2nd, 2016

**Version**

R1

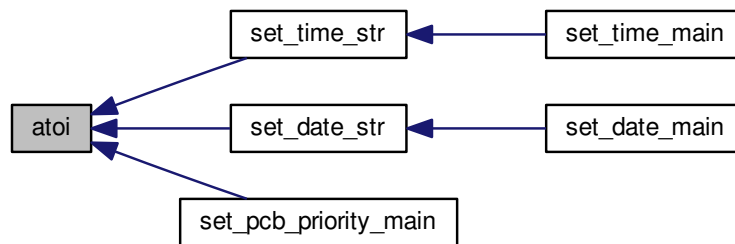
### 5.3.2 Function Documentation

### 5.3.2.1 int atoi ( const char \* s )

Here is the call graph for this function:

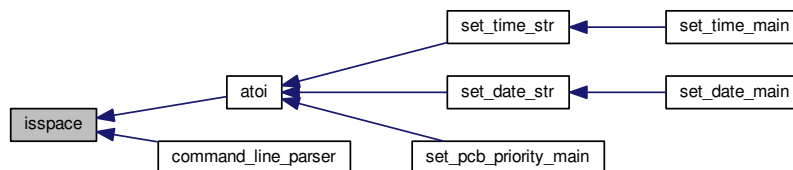


Here is the caller graph for this function:



### 5.3.2.2 int isspace ( const char \* c )

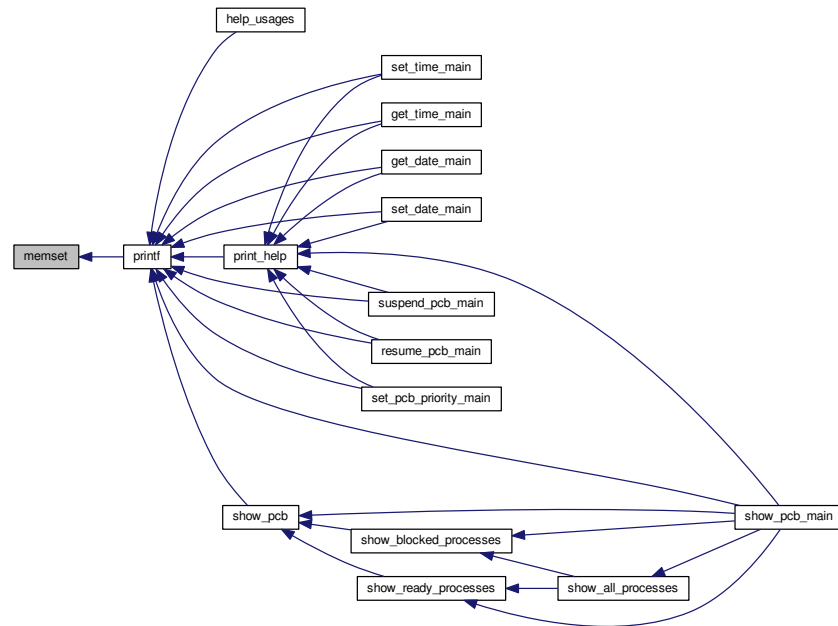
Here is the caller graph for this function:





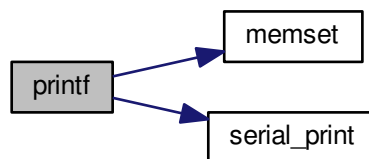
### 5.3.2.3 void\* memset ( void \* *s*, int *c*, size\_t *n* )

Here is the caller graph for this function:

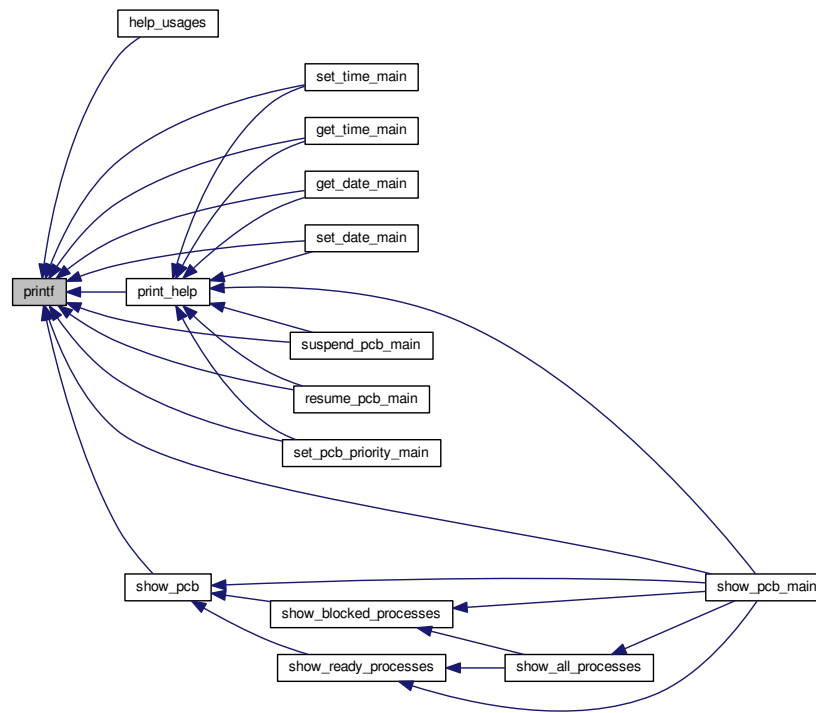


### 5.3.2.4 int printf ( const char \* *format*, ... )

Here is the call graph for this function:



Here is the caller graph for this function:

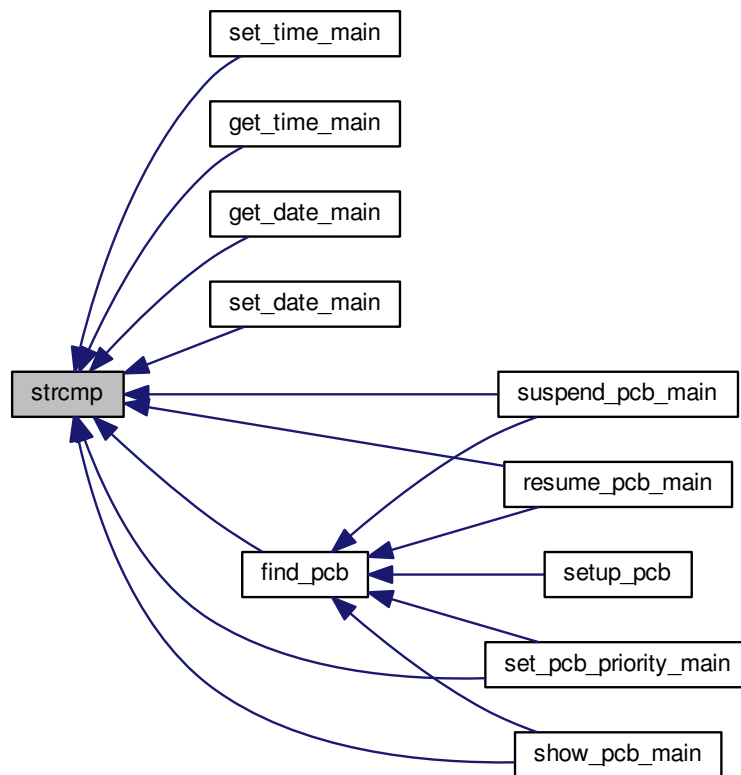


5.3.2.5 `int sprintf ( char * str, const char * format, ... )`

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

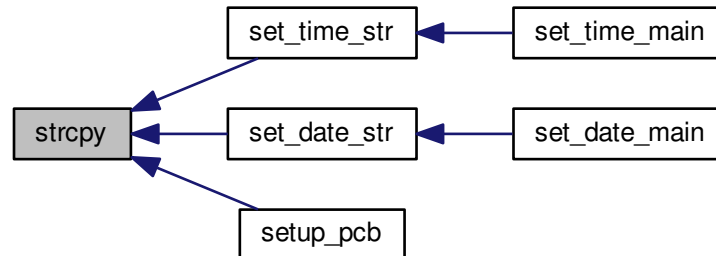
### 5.3.2.7 int strcmp ( const char \* s1, const char \* s2 )

Here is the caller graph for this function:



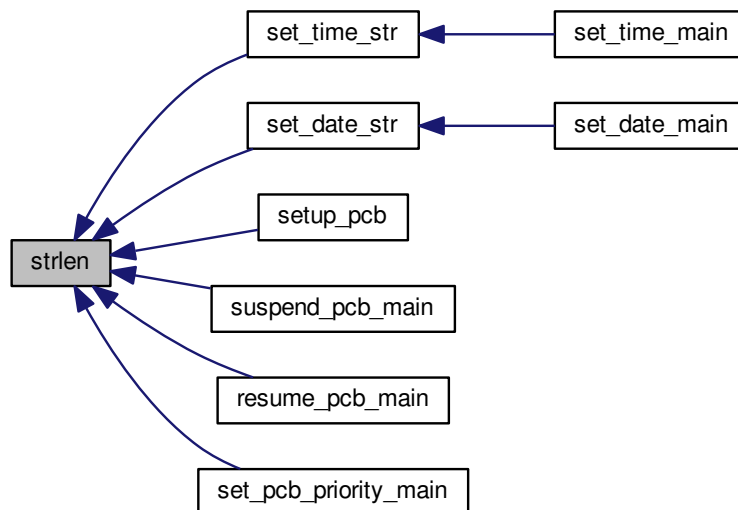
### 5.3.2.8 char\* strcpy ( char \* s1, const char \* s2 )

Here is the caller graph for this function:



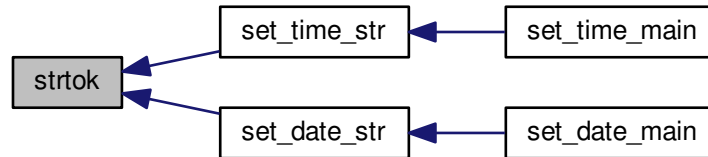
### 5.3.2.9 int strlen ( const char \* s )

Here is the caller graph for this function:



#### 5.3.2.10 char\* strtok ( char \* s1, const char \* s2 )

Here is the caller graph for this function:

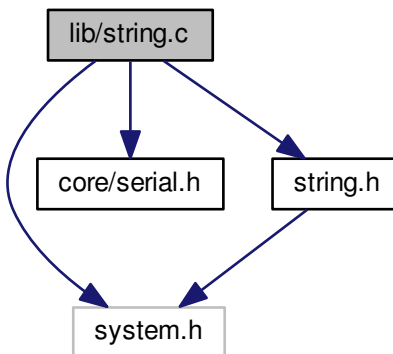


## 5.4 lib/string.c File Reference

Many usefull functions that used for handling string.

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

Include dependency graph for string.c:



## Functions

### **strlen.**

*Returns the length of a string.*

**Parameters**

s	String input.
---	---------------

**Returns**

*count Length of the String*

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

**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)

**atoi.**

*Convert an ASCII string to an integer.*

**Parameters**

s	String.
---	---------

**Returns**

*The converted integer.*

- int `atoi` (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)

**ParsePadding.**

*Parse the number for padding.*

*(static - Only can be access within this file).*

**Parameters**

str	<i>Paddling String</i>
width	<i>Paddling Width</i>
DecWidth	<i>Width of decimal part.</i>
blsRight	<i>Is align right.</i>
bHasSign	<i>Has + / -.</i>

**Returns**

*blsValid Returns the validity.*

**AddPad.**

*Add a certain number of paddings (static - Only can be access within this file).*

**Parameters**

str	<i>In string.</i>
count	<i>Number of whitespace.</i>

**Returns**

*VOID*

**NibbleToChar**

*convert a nibble into a single hexadecimal (static - Only can be access within this file)*

**Parameters**

value	<i>The value of the nibble</i>
-------	--------------------------------

**Returns**

*the character of the Hexadecimal number if valid, otherwise, return '\*'.*

**bytesToHexString.**

*Convert bytes into a hexadecimal string (static - Only can be access within this file).*

**Parameters**

OutStr	<i>Output string.</i>
Value	<i>The value of bytes.</i>

**Returns**

*VOID*

**vsprintf.**

*The actual function that perform the "printf" and "sprintf" function (static - Only can be access within this file).*

**Parameters**

str	<i>Output string.</i>
-----	-----------------------

format	<i>The format of the string.</i>
ap	<i>the pointer of the first additional parameter.</i>

**Returns**

0

**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,...)
- 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)



### 5.4.1 Detailed Description

Many usefull functions that used for handling string.

**Author**

Thunder Krakens

**Date**

February 2nd, 2016

**Version**

R1

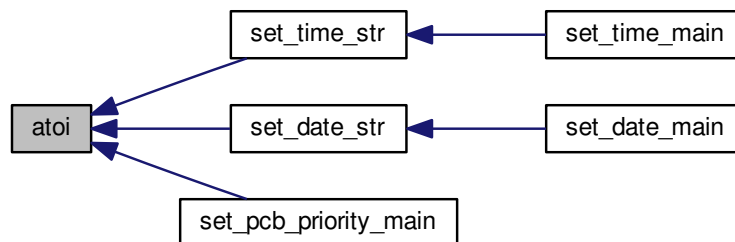
### 5.4.2 Function Documentation

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

Here is the call graph for this function:

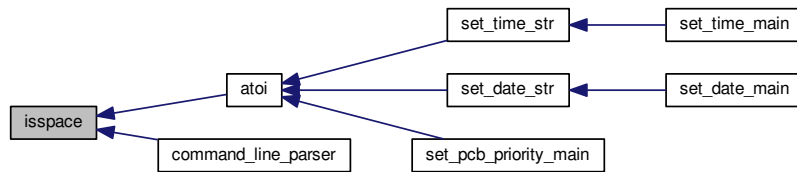


Here is the caller graph for this function:



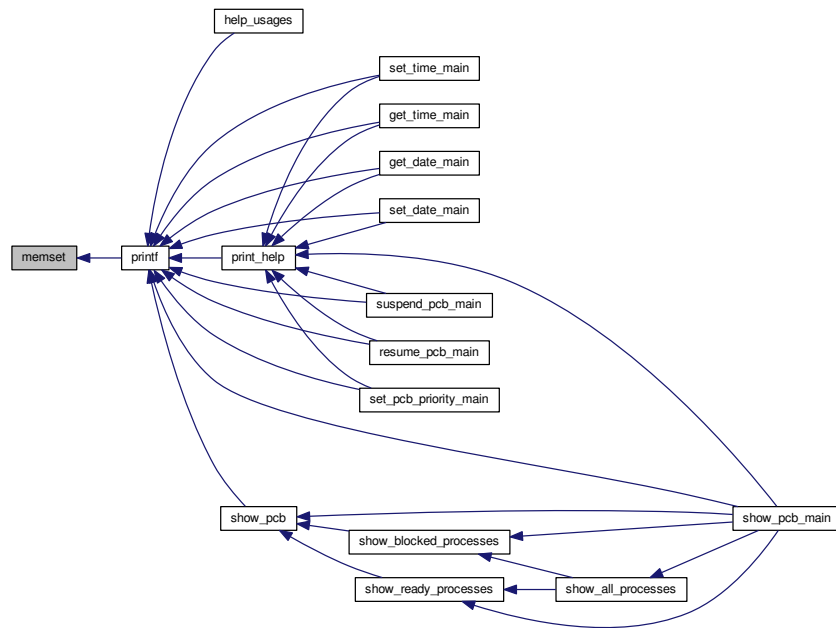
#### 5.4.2.2 int isspace ( const char \* c )

Here is the caller graph for this function:



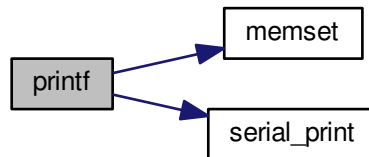
#### 5.4.2.3 void\* memset ( void \* s, int c, size\_t n )

Here is the caller graph for this function:

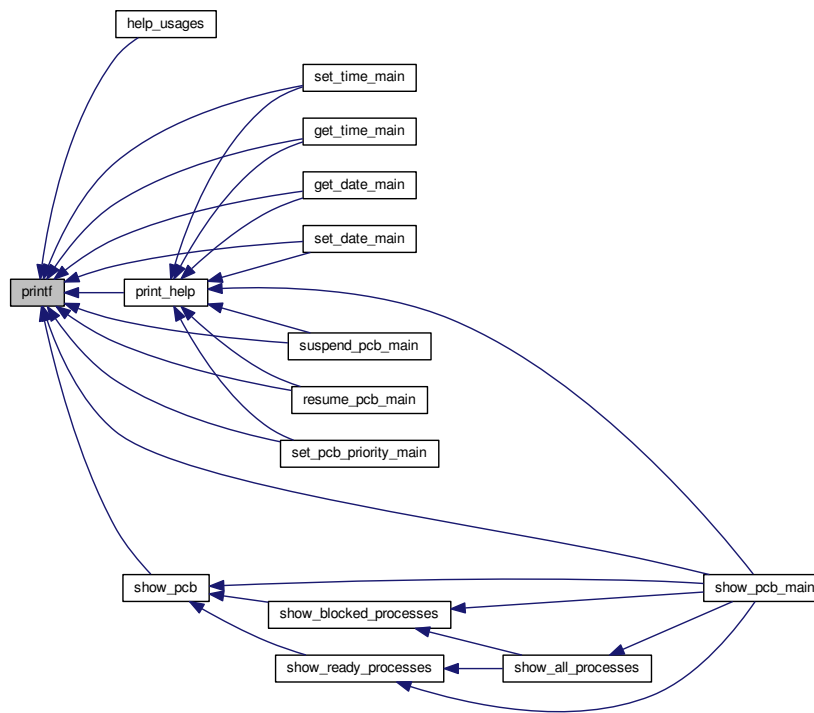


#### 5.4.2.4 int printf ( const char \* *format*, ... )

Here is the call graph for this function:



Here is the caller graph for this function:

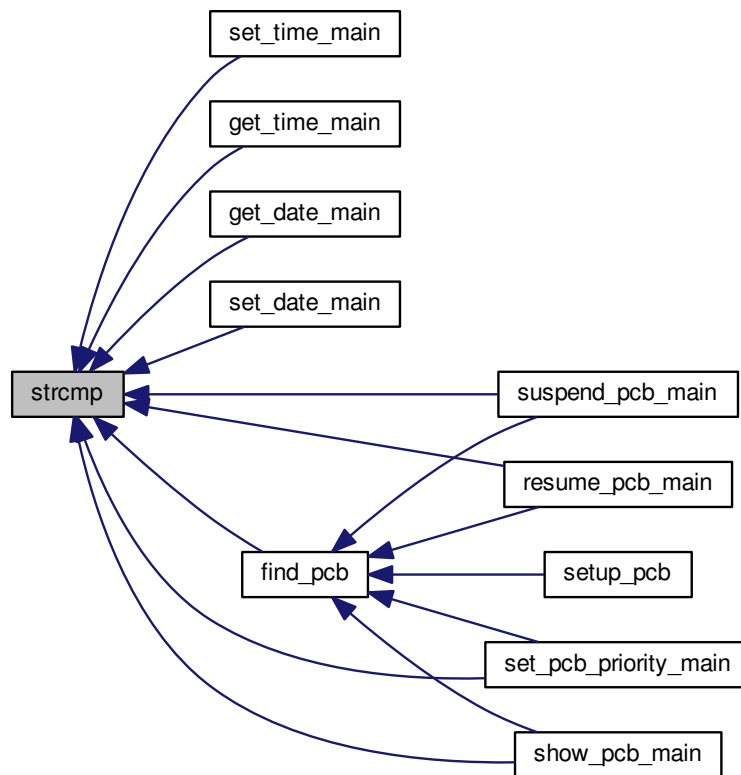


#### 5.4.2.5 int sprintf ( char \* *str*, const char \* *format*, ... )

#### 5.4.2.6 char\* strcat ( char \* *s1*, const char \* *s2* )

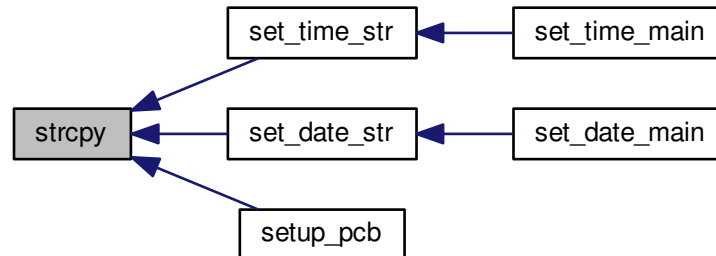
#### 5.4.2.7 int strcmp ( const char \* s1, const char \* s2 )

Here is the caller graph for this function:



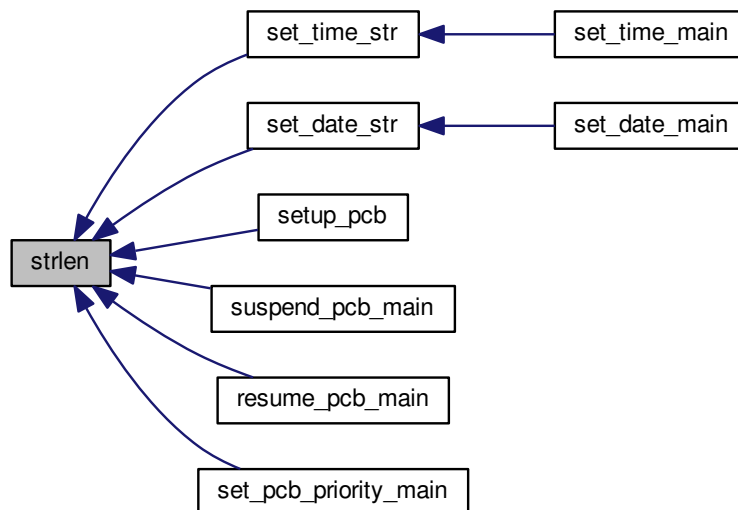
#### 5.4.2.8 char\* strcpy ( char \* s1, const char \* s2 )

Here is the caller graph for this function:



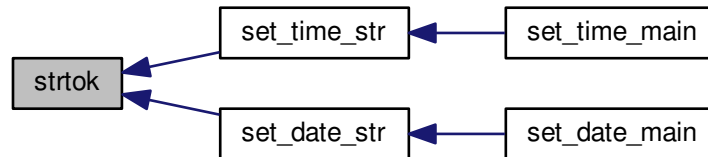
#### 5.4.2.9 int strlen ( const char \* s )

Here is the caller graph for this function:



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

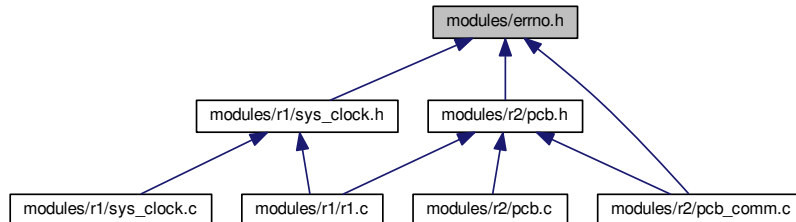
Here is the caller graph for this function:



## 5.5 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_PROGERR 99`

## Typedefs

### **error\_t.**

*The datatype that holds the error code.*

- typedef unsigned int [error\\_t](#)

## 5.5.1 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

## 5.5.2 Macro Definition Documentation

### 5.5.2.1 #define E\_EMPTPCB 6

The pcb queue is empty.

### 5.5.2.2 #define E\_FREEMEM 4

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

### 5.5.2.3 #define E\_INVPARA 1

### 5.5.2.4 #define E\_INVSTRF 2

### 5.5.2.5 #define E\_INVUSRI 3

### 5.5.2.6 #define E\_NOERROR 0

### 5.5.2.7 #define E\_NULL\_PTR 5

A NULL Pointer Error.

### 5.5.2.8 #define E\_PROGERR 99

## 5.5.3 Typedef Documentation

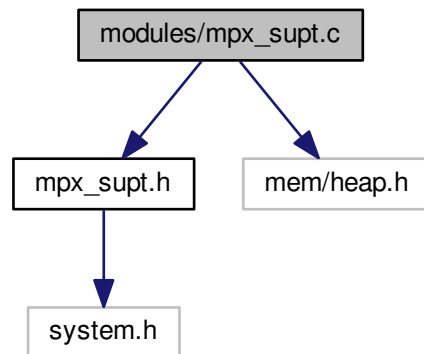
### 5.5.3.1 typedef unsigned int error\_t

## 5.6 modules/mpx\_supt.c File Reference

```
#include "mpx_supt.h"
```

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

Include dependency graph for mpx\_supt.c:



### Functions

- int [sys\\_req](#) (int op\_code)
- 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](#) ()
- int [get\\_op\\_code](#) ()

### Variables

- [param](#) [params](#)
- int [current\\_module](#) = -1
- u32int(\* [student\\_malloc](#) )(u32int)
- int(\* [student\\_free](#) )(void \*)

### 5.6.1 Function Documentation

#### 5.6.1.1 int get\_op\_code ( )

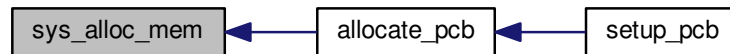


## 5.6.1.2 void idle ( )

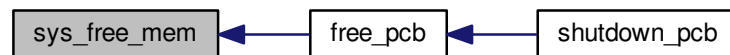
Here is the call graph for this function:

5.6.1.3 void mpx\_init ( int *cur\_mod* )5.6.1.4 void\* sys\_alloc\_mem ( u32int *size* )

Here is the caller graph for this function:

5.6.1.5 int sys\_free\_mem ( void \* *ptr* )

Here is the caller graph for this function:



#### 5.6.1.6 `int sys_req ( int op_code )`

Here is the caller graph for this function:



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

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

### 5.6.2 Variable Documentation

#### 5.6.2.1 `int current_module = -1`

#### 5.6.2.2 `param params`

#### 5.6.2.3 `int(*) student_free)(void *)`

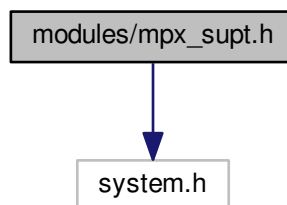
#### 5.6.2.4 `u32int(*) student_malloc)(u32int)`

## 5.7 modules/mpx\_supt.h File Reference

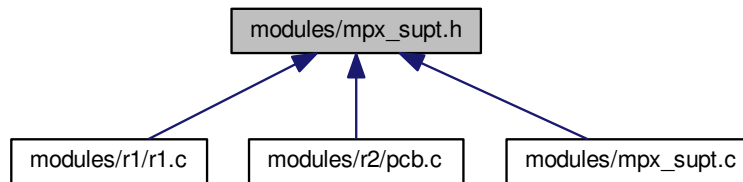
MPX System Supplementaries.

```
#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](#)  
*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	<i>op_code (IDLE)</i>
-----	-----------------------

- int [sys\\_req](#) (int op\_code)

### mpx\_init

*Initialize MPX support software*

*Parameters*

int	<i>cur_mod (symbolic constants MODULE_R1, MODULE_R2, etc</i>
-----	--

- void [mpx\\_init](#) (int cur\_mod)

### set\_malloc

*Sets the memory allocation function for sys\_alloc\_mem*

#### Parameters

Function	<i>pointer</i>
----------	----------------

- 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	<i>of bytes to allocate</i>
--------	-----------------------------

- void \* [sys\\_alloc\\_mem](#) (u32int size)

### sys\_free\_mem

*Frees memory*

#### Parameters

Pointer	<i>to block of memory to free</i>
---------	-----------------------------------

- 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](#) ()

**Variables**

- typedef [\\_\\_attribute\\_\\_](#)

**5.7.1 Detailed Description**

MPX System Supplementaries.

**Author**

Thunder Krakens

**Date**

March 18, 2016

**Version**

R3

**5.7.2 Macro Definition Documentation**

5.7.2.1 `#define EXIT 0`

5.7.2.2 `#define IDLE 1`

5.7.2.3 `#define MODULE_R1 0`

5.7.2.4 `#define MODULE_R2 1`

5.7.2.5 `#define MODULE_R3 2`

5.7.2.6 `#define MODULE_R4 4`

5.7.2.7 `#define MODULE_R5 8`

5.7.2.8 `#define READ 2`

5.7.2.9 `#define WRITE 3`

**5.7.3 Function Documentation**

5.7.3.1 `int get_op_code ( )`

### 5.7.3.2 void idle ( )

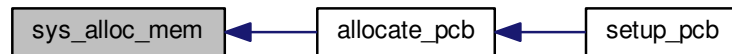
Here is the call graph for this function:



### 5.7.3.3 void mpx\_init ( int *cur\_mod* )

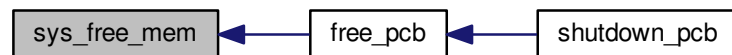
### 5.7.3.4 void\* sys\_alloc\_mem ( u32int *size* )

Here is the caller graph for this function:



### 5.7.3.5 int sys\_free\_mem ( void \* *ptr* )

Here is the caller graph for this function:



## 5.7.3.6 int sys\_req ( int op\_code )

Here is the caller graph for this function:



## 5.7.3.7 void sys\_set\_free ( int(\*) (void \*) func )

## 5.7.3.8 void sys\_set\_malloc ( u32int(\*) (u32int) func )

## 5.7.4 Variable Documentation

## 5.7.4.1 enum process\_suspended \_\_attribute\_\_

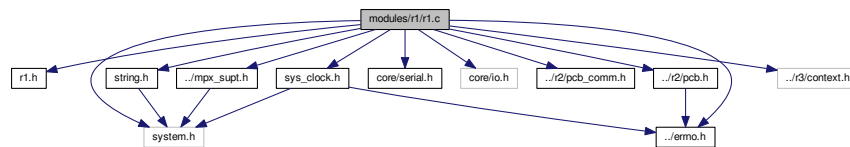
## 5.8 modules/r1/r1.c File Reference

The commandhandler and functions associations for Module R1.

```

#include "r1.h"
#include "sys_clock.h"
#include <string.h>
#include <core/serial.h>
#include <core/io.h>
#include "../r2/pcb_comm.h"
#include "../r2/pcb.h"
#include "../mpx_supt.h"
#include "../r3/context.h"
  
```

Include dependency graph for r1.c:



## Data Structures

- struct [function\\_name](#)

*A structure to represent each function.*

## Macros

- #define `MAX_ARGC` 50
- #define `MOD_VERSION` "R4"
- #define `COMPLETION` "03/18/2016"
- #define `MAX_HISTORY` 10

## Functions

### **exe\_function.**

*Executes the specific function.*

#### **Parameters**

argc	<i>The number of tokens.</i>
argv	<i>The array of tokens.</i>

#### **Returns**

0

### **version**

*displays the version of the system currently running.*

#### **Parameters**

argc	<i>The number of tokens.</i>
argv	<i>The array of tokens.</i>

#### **Returns**

0

### **shutdown**

*Closes all functions, and shuts down the system.*

#### **Parameters**

argc	<i>The number of tokens found.</i>
argv	<i>The array of tokens.</i>

#### **Returns**

0 for shutdown, 1 for keep running.

### **help\_usages**

*shows usage message for each function.*

#### **Parameters**

start_from	<i>the index of the beginning function.</i>
------------	---

#### **Returns**

0

- int `help_usages` (enum `comm_type` type)

### **help\_function**

*displays help text for all functions.*



**Parameters**

argc	<i>The number of tokens.</i>
argv	<i>The array of tokens.</i>

**Returns**

0

**commhand**

*Accepts and handles commands from the user.*

**Returns**

0

- void [commhand](#) ()

**command\_line\_parser**

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

**Parameters**

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

**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	<i>The index number of that function.</i>
----------------	---

**Returns**

void

- void [print\\_help](#) (const int function\_index)

**Variables**

- [NotWriting](#)
- [NormalWriting](#)
- [DoubleQuoteWriting](#)
- [SingleQuoteWriting](#)

## CommandParserStat

The status of the command parser

- enum [CommandPaserStat](#)
- enum [CommandPaserStat \\_\\_attribute\\_\\_ \(\(packed\)\)](#)

### 5.8.1 Detailed Description

The commandhandler and functions associations for Module R1.

#### Author

Thunder Krakens

#### Date

February 2nd, 2016

#### Version

R4

### 5.8.2 Macro Definition Documentation

5.8.2.1 `#define COMPLETION "03/18/2016"`

5.8.2.2 `#define MAX_ARGC 50`

5.8.2.3 `#define MAX_HISTORY 10`

5.8.2.4 `#define MOD_VERSION "R4"`

### 5.8.3 Enumeration Type Documentation

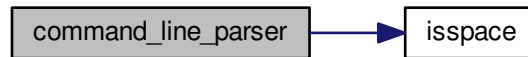
5.8.3.1 enum [CommandPaserStat](#)

### 5.8.4 Function Documentation

5.8.4.1 enum [CommandPaserStat \\_\\_attribute\\_\\_ \( \(packed\) \)](#)

5.8.4.2 void `command_line_parser` ( const char \* *CmdStr*, int \* *argc*, char \*\* *argv*, const int *MaxArgNum*, const int *MaxStrLen* )

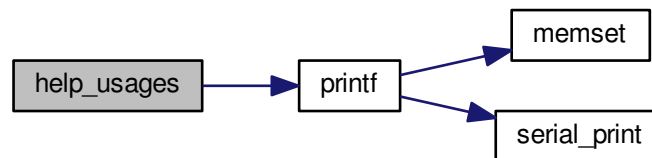
Here is the call graph for this function:



5.8.4.3 void `commhand` ( )

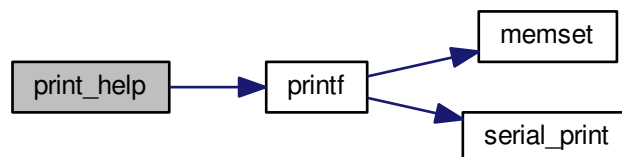
5.8.4.4 int `help_usages` ( enum `comm_type` *type* )

Here is the call graph for this function:

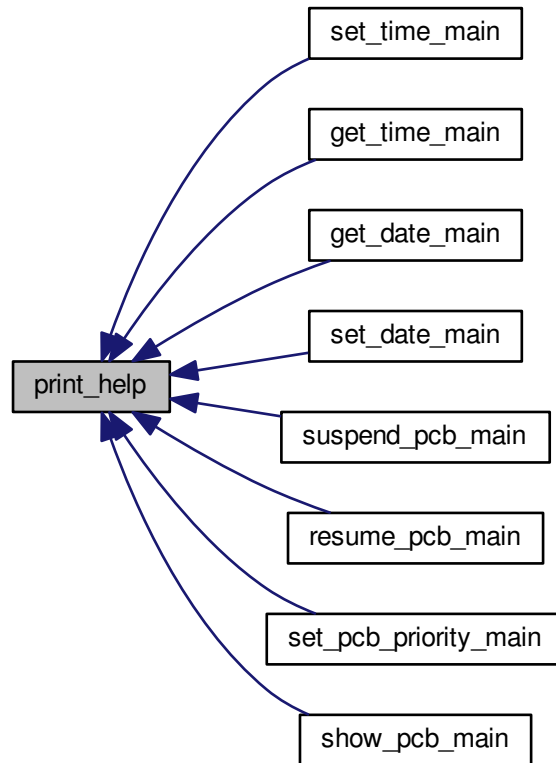


5.8.4.5 void `print_help` ( const int *function\_index* )

Here is the call graph for this function:



Here is the caller graph for this function:



## 5.8.5 Variable Documentation

### 5.8.5.1 DoubleQuoteWriting

### 5.8.5.2 NormalWriting

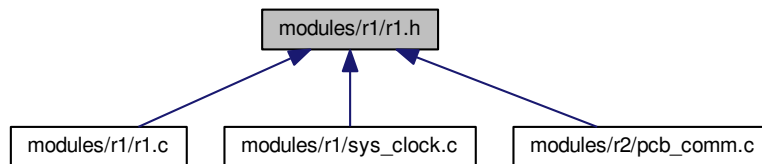
### 5.8.5.3 NotWriting

### 5.8.5.4 SingleQuoteWriting

## 5.9 modules/r1/r1.h File Reference

The command handler and functions associations for Module R1.

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



## Macros

- #define [HELP](#) 0
- #define [POS\\_OF\\_MPX](#) 1
- #define [VERSION](#) 1
- #define [GETTIME](#) 2
- #define [SETTIME](#) 3
- #define [GETDATE](#) 4
- #define [SETDATE](#) 5
- #define [SHUTDOWN](#) 6
- #define [YIELD](#) 7
- #define [LOADR3](#) 8
- #define [NUM\\_MPX\\_FUNCTIONS](#) 9
- #define [POS\\_OF\\_PCB](#) 9
- #define [SUSPDPCB](#) 9
- #define [RESUMEPCB](#) 10
- #define [SETPCBPRIOR](#) 11
- #define [SHOWPCB](#) 12
- #define [NUM\\_OF\\_FUNCTIONS](#) 13

## 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	<i>The complete input command.</i>
argc	<i>The number of tokens found.</i>
argv	<i>The array of tokens.</i>
MaxArgNum	<i>The maximum number of tokens that array can hold.</i>
MaxStrLen	<i>The maximum length of each token that string can hold.</i>

**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	<i>The index number of that function.</i>
----------------	---

**Returns**

*void*

- void [print\\_help](#) (const int function\_index)
- int [help\\_usages](#) (enum [comm\\_type](#) type)

**Variables**

- [mpx](#)
- [pcb](#)
- [help](#)

**5.9.1 Detailed Description**

The command handler and functions associations for Module R1.

**Author**

Thunder Krakens

**Date**

March 17, 2016

**Version**

R3 & R4

## 5.9.2 Macro Definition Documentation

5.9.2.1 `#define GETDATE 4`

5.9.2.2 `#define GETTIME 2`

5.9.2.3 `#define HELP 0`

5.9.2.4 `#define LOADR3 8`

5.9.2.5 `#define NUM_MPX_FUNCTIONS 9`

5.9.2.6 `#define NUM_OF_FUNCTIONS 13`

5.9.2.7 `#define POS_OF_MPX 1`

5.9.2.8 `#define POS_OF_PCB 9`

5.9.2.9 `#define RESUMEPCB 10`

5.9.2.10 `#define SETDATE 5`

5.9.2.11 `#define SETPCBPRI 11`

5.9.2.12 `#define SETTIME 3`

5.9.2.13 `#define SHOWPCB 12`

5.9.2.14 `#define SHUTDOWN 6`

5.9.2.15 `#define SUSPDPCB 9`

5.9.2.16 `#define VERSION 1`

5.9.2.17 `#define YIELD 7`

## 5.9.3 Enumeration Type Documentation

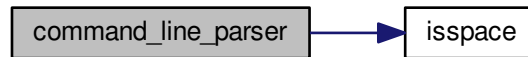
5.9.3.1 `enum comm_type`

## 5.9.4 Function Documentation

5.9.4.1 `enum comm_type __attribute__((packed))`

5.9.4.2 void command\_line\_parser ( const char \* *CmdStr*, int \* *argc*, char \*\* *argv*, const int *MaxArgNum*, const int *MaxStrLen* )

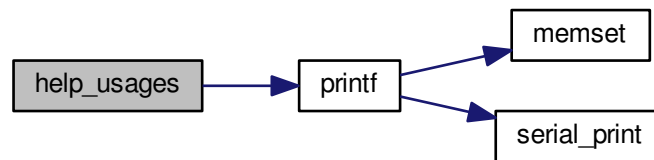
Here is the call graph for this function:



5.9.4.3 void commhand ( )

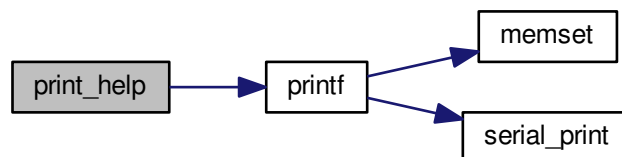
5.9.4.4 int help\_usages ( enum comm\_type *type* )

Here is the call graph for this function:



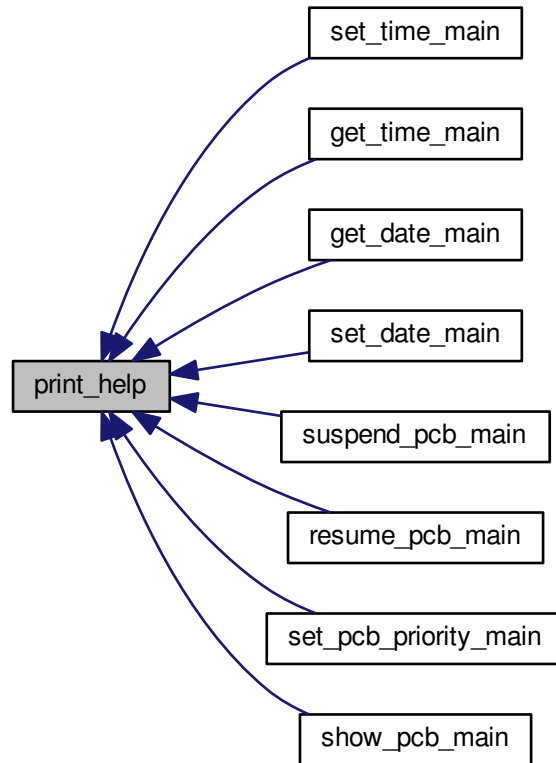
5.9.4.5 void print\_help ( const int *function\_index* )

Here is the call graph for this function:





Here is the caller graph for this function:



### 5.9.5 Variable Documentation

#### 5.9.5.1 help

#### 5.9.5.2 mpx

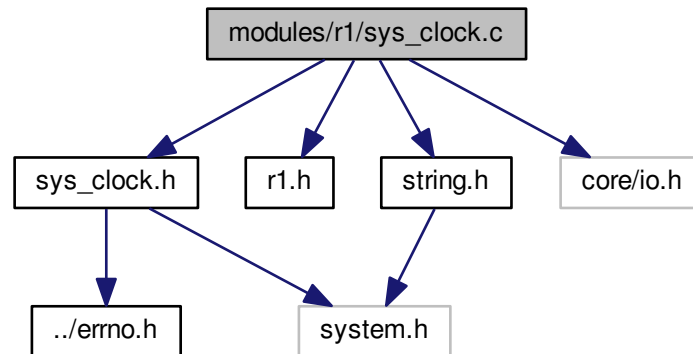
#### 5.9.5.3 pcb

## 5.10 modules/r1/sys\_clock.c File Reference

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

```
#include "sys_clock.h"
#include "r1.h"
#include <string.h>
#include <core/io.h>
```

Include dependency graph for sys\_clock.c:



## Macros

- `#define RTC_INDEX_SECOND 0x00`
- `#define RTC_INDEX_SECOND_ALARM 0x01`
- `#define RTC_INDEX_MINUTE 0x02`
- `#define RTC_INDEX_MINUTE_ALARM 0x03`
- `#define RTC_INDEX_HOUR 0x04`
- `#define RTC_INDEX_HOUR_ALARM 0x05`
- `#define RTC_INDEX_DAY_WEEK 0x06`
- `#define RTC_INDEX_DAY_MONTH 0x07`
- `#define RTC_INDEX_MONTH 0x08`
- `#define RTC_INDEX_YEAR 0x09`

## Functions

### **set\_time\_main.**

*Sets the time for the system.*

#### **Parameters**

<code>argc</code>	<i>The number of tokens found.</i>
<code>argv</code>	<i>The array of tokens.</i>

#### **Returns**

0

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

### **get\_time\_main.**

*Retrieves system's current time.*

**Parameters**

argc	<i>The number of tokens found.</i>
argv	<i>The array of tokens.</i>

**Returns**

0

- int [get\\_time\\_main](#) (int argc, char \*\*argv)

**is\_digit**

*determines if a character represents a digit.*

**Parameters**

ch	<i>The character</i>
----	----------------------

**Returns**

*1 if it is digit, otherwise returns 0.*

**set\_time\_str.**

*Sets the time for the system by string.*

**Parameters**

timeStr	<i>The string type of current Time.</i>
---------	---

**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	<i>The value of current time and date</i>
----------------	---

**Returns**

VOID

- void [get\\_time](#) (date\_time \*dateTimeValues)

**set\_time.**

*Sets the time for the system by date\_time struct.*

**Parameters**

dateTimeValues	<i>The struct that holds the time values.</i>
----------------	---

**Returns**

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

- [error\\_t set\\_time](#) (const date\_time \*dateTimeValues)

**get\_date.**

*Retrieves system's current date.*

**Parameters**

dateTimeValues	<i>The struct that holds the value of current date</i>
----------------	--

**Returns**

VOID

- void [get\\_date](#) (date\_time \*dateTimeValues)

**is\_date\_value\_valid.**

Check if the date specified is valid, which means year should between 1970 ~ 1969, month should between 1 ~ 12, while the range of the day is based on the month and year.

**Parameters**

year	<i>The value of the year</i>
mon	<i>The value of the month</i>
day	<i>The value of the day of month</i>

**Returns**

VOID

**set\_date.**

Sets the date of the system.

**Parameters**

dateTimeValues	<i>The struct that holds the value of date</i>
----------------	--

**Returns**

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

- [error\\_t set\\_date](#) (const date\_time \*dateTimeValues)

**get\_date\_main.**

Retrieves system's current date.

**Parameters**

argc	<i>The number of tokens.</i>
argv	<i>The array of tokens.</i>

**Returns**

0

- int [get\\_date\\_main](#) (int argc, char \*\*argv)

**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\_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)

### 5.10.1 Detailed Description

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

#### Author

Thunder Krakens

#### Date

February 2nd, 2016

#### Version

R1

### 5.10.2 Macro Definition Documentation

5.10.2.1 `#define RTC_INDEX_DAY_MONTH 0x07`

5.10.2.2 `#define RTC_INDEX_DAY_WEEK 0x06`

5.10.2.3 `#define RTC_INDEX_HOUR 0x04`

5.10.2.4 `#define RTC_INDEX_HOUR_ALARM 0x05`

5.10.2.5 `#define RTC_INDEX_MINUTE 0x02`

5.10.2.6 `#define RTC_INDEX_MINUTE_ALARM 0x03`

5.10.2.7 `#define RTC_INDEX_MONTH 0x08`

5.10.2.8 `#define RTC_INDEX_SECOND 0x00`

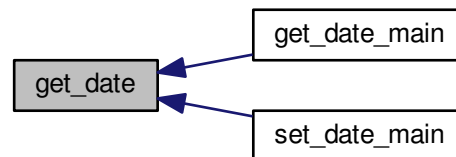
5.10.2.9 `#define RTC_INDEX_SECOND_ALARM 0x01`

5.10.2.10 `#define RTC_INDEX_YEAR 0x09`

### 5.10.3 Function Documentation

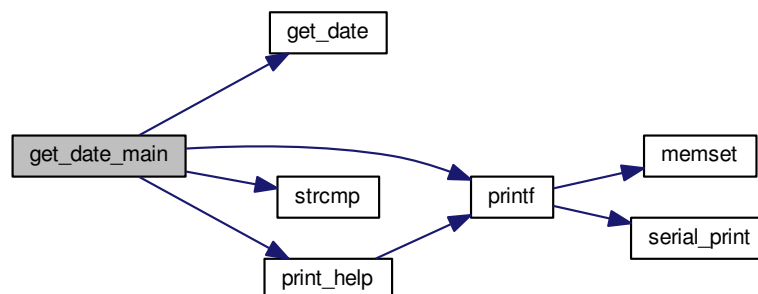
5.10.3.1 `void get_date ( date_time * dateTimeValues )`

Here is the caller graph for this function:



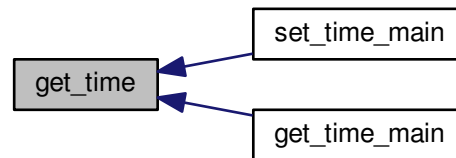
5.10.3.2 `int get_date_main ( int argc, char ** argv )`

Here is the call graph for this function:



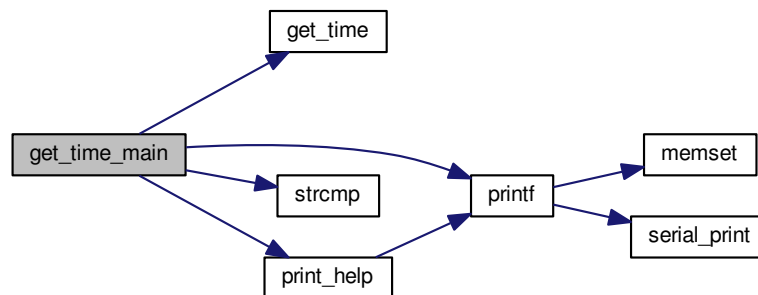
### 5.10.3.3 void get\_time ( date\_time \* dateTimeValues )

Here is the caller graph for this function:



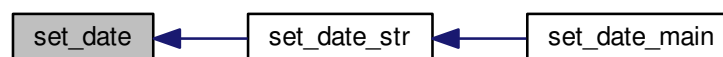
### 5.10.3.4 int get\_time\_main ( int argc, char \*\* argv )

Here is the call graph for this function:



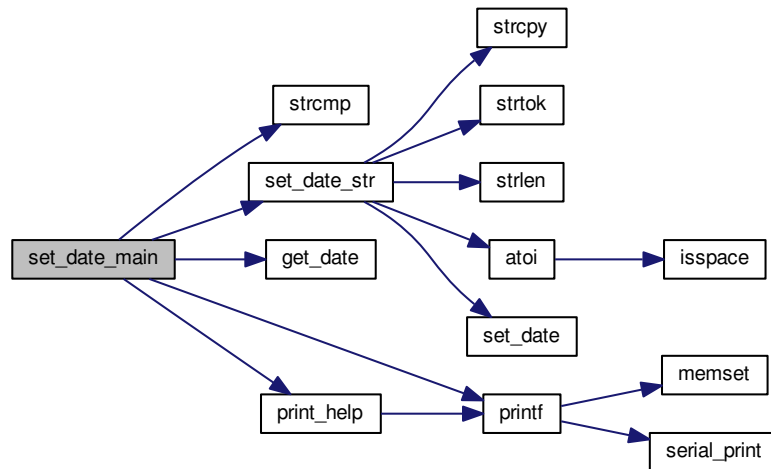
### 5.10.3.5 error\_t set\_date ( const date\_time \* dateTimeValues )

Here is the caller graph for this function:



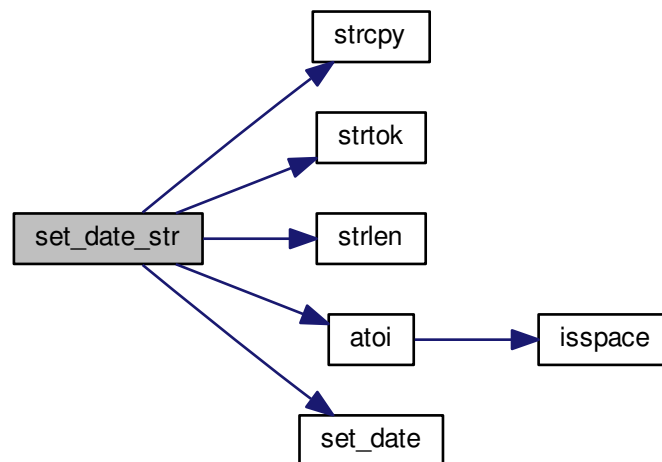
#### 5.10.3.6 int set\_date\_main ( int argc, char \*\* argv )

Here is the call graph for this function:



#### 5.10.3.7 int set\_date\_str ( const char \* str )

Here is the call graph for this function:



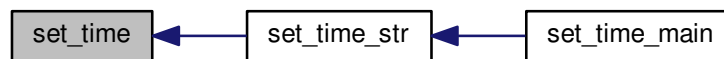


Here is the caller graph for this function:



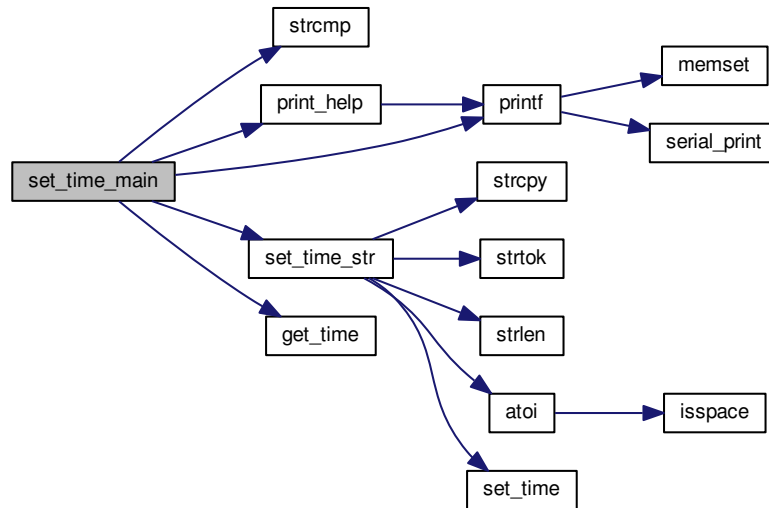
#### 5.10.3.8 `error_t set_time ( const date_time * dateTimeValues )`

Here is the caller graph for this function:



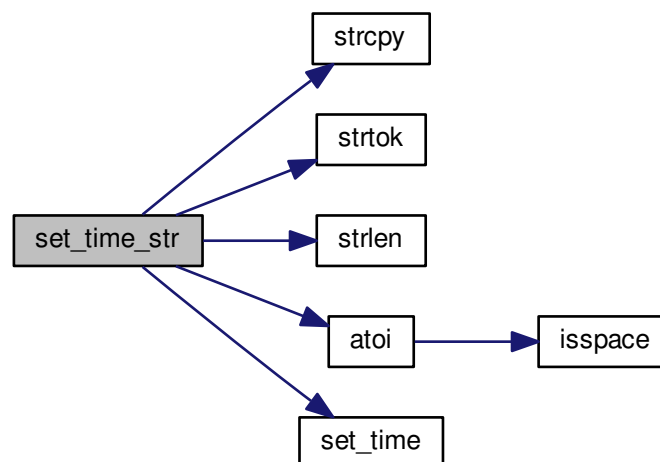
#### 5.10.3.9 `int set_time_main ( int argc, char ** argv )`

Here is the call graph for this function:



#### 5.10.3.10 `error_t set_time_str ( const char * timeStr )`

Here is the call graph for this function:



Here is the caller graph for this function:

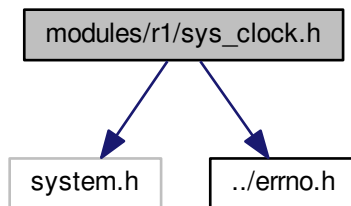


## 5.11 modules/r1/sys\_clock.h File Reference

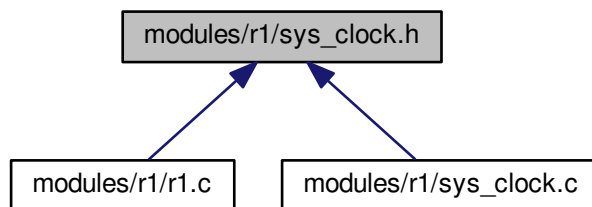
The main file that manipulates and controls the system's clock.

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

Include dependency graph for `sys_clock.h`:



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



## Functions

### **set\_time\_main.**

*Sets the time for the system.*

#### *Parameters*

argc	<i>The number of tokens found.</i>
argv	<i>The array of tokens.</i>

#### *Returns*

0

- int [set\\_time\\_main](#) (int argc, char \*\*argv)

### **get\_time\_main.**

*Retrieves system's current time.*

#### *Parameters*

argc	<i>The number of tokens found.</i>
argv	<i>The array of tokens.</i>

#### *Returns*

0

- int [get\\_time\\_main](#) (int argc, char \*\*argv)

### **set\_time\_str.**

*Sets the time for the system by string.*

#### *Parameters*

timeStr	<i>The string type of current Time.</i>
---------	---

#### *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	<i>The value of current time and date</i>
----------------	---

#### *Returns*

VOID

- void [get\\_time](#) (date\_time \*dateTimeValues)

### **set\_time.**

*Sets the time for the system by date\_time struct.*

**Parameters**

dateTimeValues	<i>The struct that holds the time values.</i>
----------------	---

**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	<i>The number of tokens.</i>
argv	<i>The array of tokens.</i>

**Returns**

0

- int [set\\_date\\_main](#) (int argc, char \*\*argv)

**get\_date\_main.**

*Retrieves system's current date.*

**Parameters**

argc	<i>The number of tokens.</i>
argv	<i>The array of tokens.</i>

**Returns**

0

- int [get\\_date\\_main](#) (int argc, char \*\*argv)

**get\_date.**

*Retrieves system's current date.*

**Parameters**

dateTimeValues	<i>The struct that holds the value of current date</i>
----------------	--

**Returns**

VOID

- void [get\\_date](#) (date\_time \*dateTimeValues)

**set\_date\_str.**

*Sets the date for the system by string.*

**Parameters**

str	<i>The string type of current date.</i>
-----	---

**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	<i>The struct that holds the value of date</i>
----------------	--

**Returns**

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

- [error\\_t set\\_date](#) (const date\_time \*dateTimeValues)

**5.11.1 Detailed Description**

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

**Author**

Thunder Krakens

**Date**

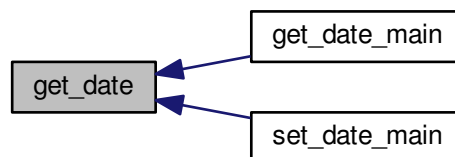
February 2nd, 2016

**Version**

R1

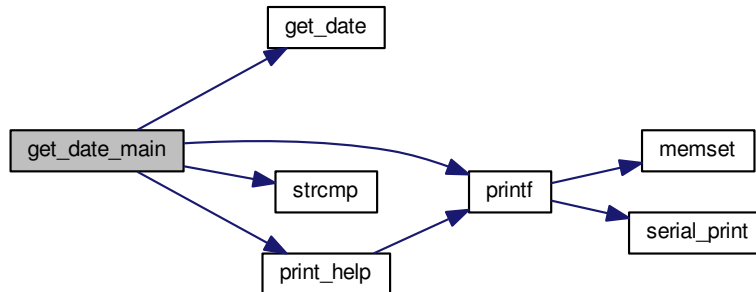
**5.11.2 Function Documentation****5.11.2.1 void get\_date ( date\_time \* dateTimeValues )**

Here is the caller graph for this function:



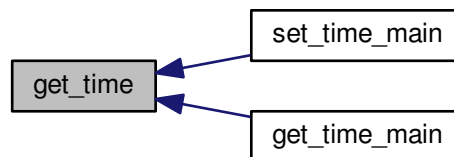
### 5.11.2.2 int get\_date\_main ( int argc, char \*\* argv )

Here is the call graph for this function:



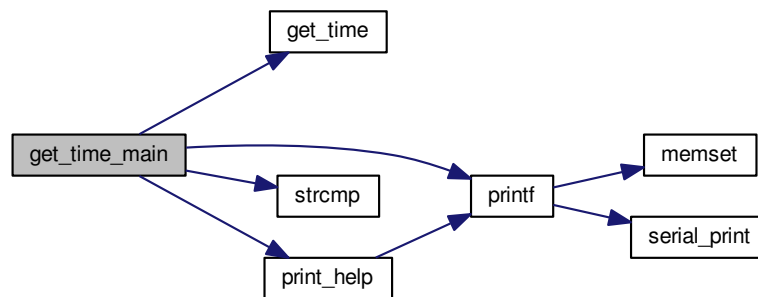
### 5.11.2.3 void get\_time ( date\_time \* dateTimeValues )

Here is the caller graph for this function:



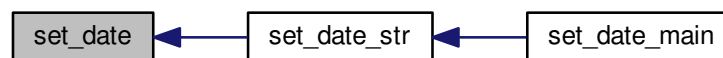
#### 5.11.2.4 `int get_time_main ( int argc, char ** argv )`

Here is the call graph for this function:



#### 5.11.2.5 `error_t set_date ( const date_time * dateTimeValues )`

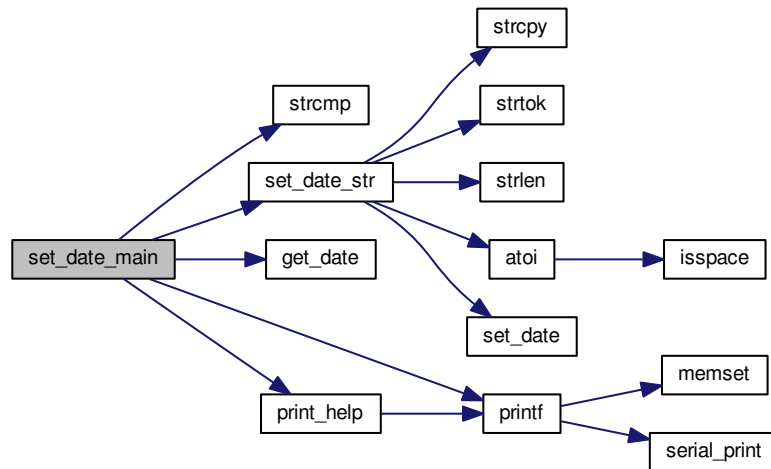
Here is the caller graph for this function:





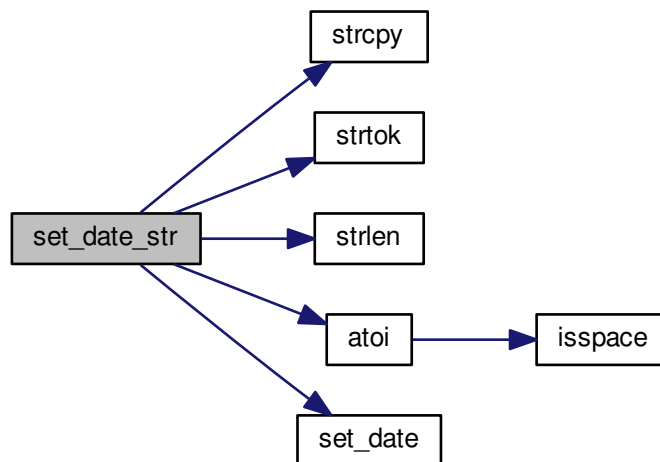
#### 5.11.2.6 int set\_date\_main ( int argc, char \*\* argv )

Here is the call graph for this function:



#### 5.11.2.7 int set\_date\_str ( const char \* str )

Here is the call graph for this function:

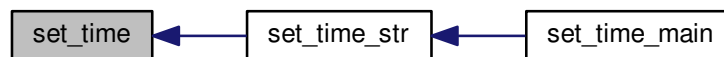


Here is the caller graph for this function:



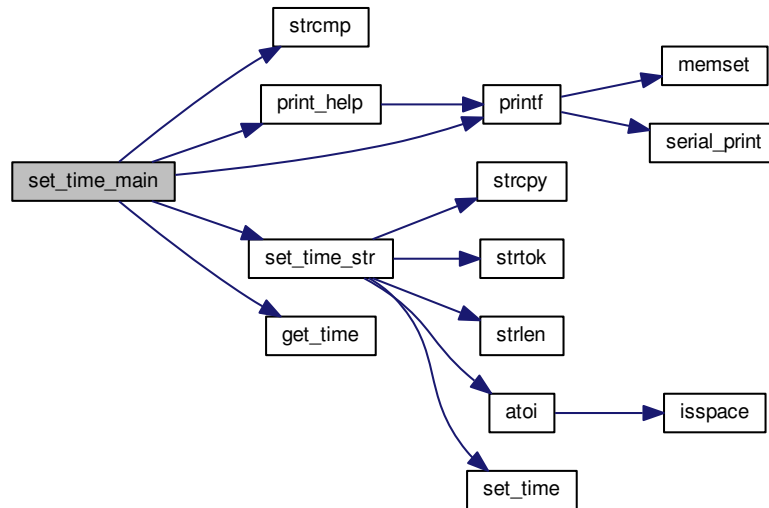
#### 5.11.2.8 `error_t set_time ( const date_time * dateTimeValues )`

Here is the caller graph for this function:



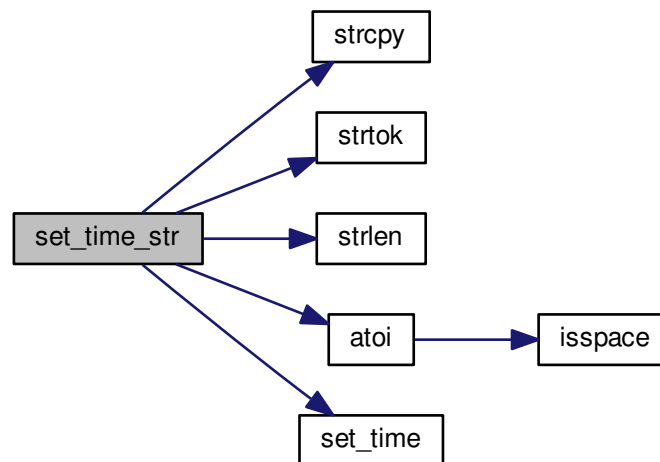
### 5.11.2.9 `int set_time_main ( int argc, char ** argv )`

Here is the call graph for this function:



### 5.11.2.10 `error_t set_time_str ( const char * timeStr )`

Here is the call graph for this function:



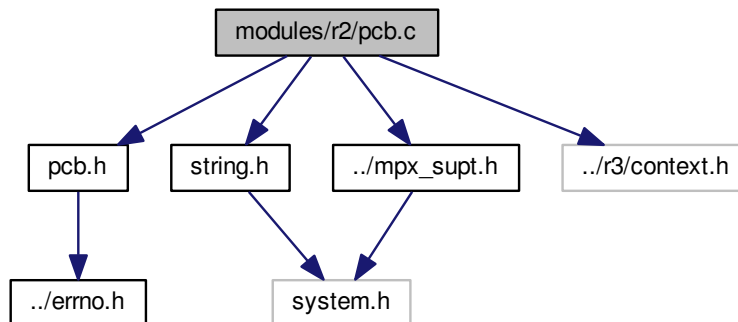
Here is the caller graph for this function:



## 5.12 modules/r2/pcb.c File Reference

The Process Control Block.

```
#include "pcb.h"
#include <string.h>
#include "../mpx_supt.h"
#include "../r3/context.h"
Include dependency graph for pcb.c:
```



## Data Structures

- struct [pcb\\_struct](#)  
*Struct that will describe PCB Processes.*
- struct [pcb\\_queue](#)  
*Queue structure that will store PCBs.*

## Enumerations

- enum [process\\_state](#)

*PCB process states/statuses.*

- enum [process\\_suspended](#)

*PCB process suspended or not suspended status.*

## Functions

- enum [process\\_state\\_\\_attribute\\_\\_](#) ((packed))

### **pcb\_init**

*Initiates the PCB queues*

- void [pcb\\_init](#) ()

### **suspend\_pcb**

*Suspends the specific PCB.*

*Parameters*

<a href="#">pcb_ptr</a>	<i>The pointer to the PCB</i>
-------------------------	-------------------------------

*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*

<a href="#">pcb_ptr</a>	<i>The pointer to the PCB</i>
-------------------------	-------------------------------

*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)

### **allocate\_pcb**

*allocate a space for the PCB structure.*

*Returns*

*The pointer that point to the PCB structure.*

- struct [pcb\\_struct](#) \* [allocate\\_pcb](#) ()

### **setup\_pcb**

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

*NOTE: pName must less than SIZE\_OF\_PCB\_NAME character, pClass should be either "application" or "system", and pPriority must within the range of [0, 9].*

**Parameters**

pName	Process Name (length < SIZE_OF_PCB_NAME).
pClass	Process class (system or application).
pPriority	Process priority (0 ~ 9).

**Returns**

NULL if error occurred, 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)

**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. *E\_FREEMEM* The memory space cannot be actually free, since the *student\_free* had not been implemented yet.

- [error\\_t](#) [free\\_pcb](#) (struct [pcb\\_struct](#) \*pcb\_ptr)

**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)

**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\_blocked\_processes**

displays all blocked processes and their attributes

**Returns**

VOID.

- void [show\\_blocked\\_processes](#) ()

**show\_ready\_processes**

Displays all of the ready processes and their attributes.

**Returns**

VOID.

- void [show\\_ready\\_processes](#) ()

**show\_all\_processes**

Displays all of the processes and their attributes.

**Returns**

VOID.

- void [show\\_all\\_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)

**set\_pcb\_priority**

Sets the priority of the selected PCB

**Parameters**

pcb_ptr	The PCB pointer.
pPriority	The assigned priority

**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)

**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**

- [running](#)  
PCB in the running state.
- [ready](#)  
PCB in the ready state.
- [blocked](#)  
< PCB in the blocked state.
- [true](#)

*PCB process is suspended.*

- `false`  
*< PCB process is not suspended.*
- struct `pcb_struct __attribute__`

### 5.12.1 Detailed Description

The Process Control Block.

Author

Thunder Krakens

Date

March 18th, 2016

Version

R3

### 5.12.2 Enumeration Type Documentation

#### 5.12.2.1 `enum process_state`

PCB process states/statuses.

#### 5.12.2.2 `enum process_suspended`

PCB process suspended or not suspended status.

### 5.12.3 Function Documentation

#### 5.12.3.1 `enum process_state __attribute__((packed))`

#### 5.12.3.2 `struct pcb_struct* allocate_pcb ( )`

Here is the call graph for this function:

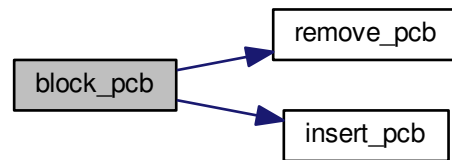


Here is the caller graph for this function:



#### 5.12.3.3 `error_t block_pcb ( struct pcb_struct * pcb_ptr )`

Here is the call graph for this function:

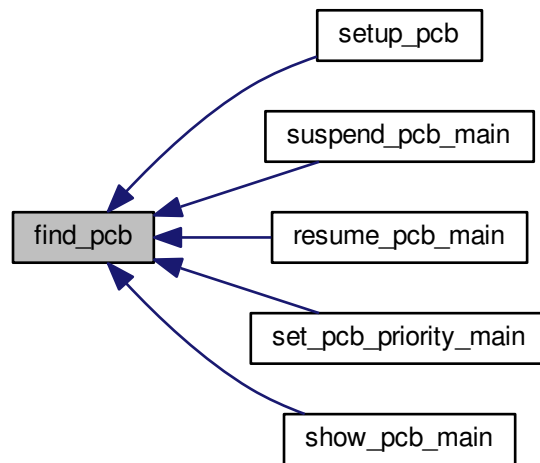


#### 5.12.3.4 `struct pcb_struct* find_pcb ( const char * pName )`

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.12.3.5 `error_t free_pcb ( struct pcb_struct * pcb_ptr )`

Here is the call graph for this function:

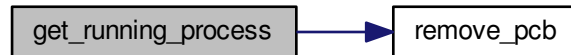


Here is the caller graph for this function:



#### 5.12.3.6 struct pcb\_struct\* get\_running\_process ( )

Here is the call graph for this function:

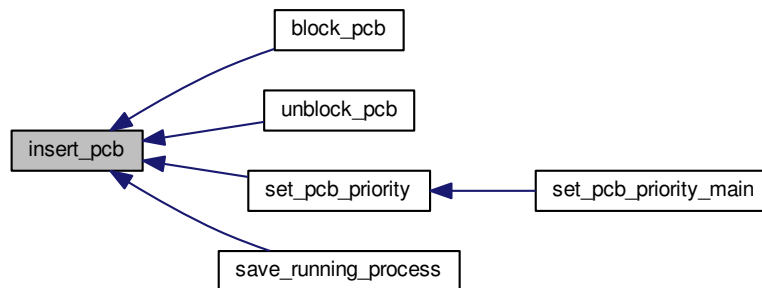


#### 5.12.3.7 unsigned char\* get\_stack\_base ( struct pcb\_struct \* pcb\_ptr )

#### 5.12.3.8 unsigned char\* get\_stack\_top ( struct pcb\_struct \* pcb\_ptr )

#### 5.12.3.9 error\_t insert\_pcb ( struct pcb\_struct \* pcb\_ptr )

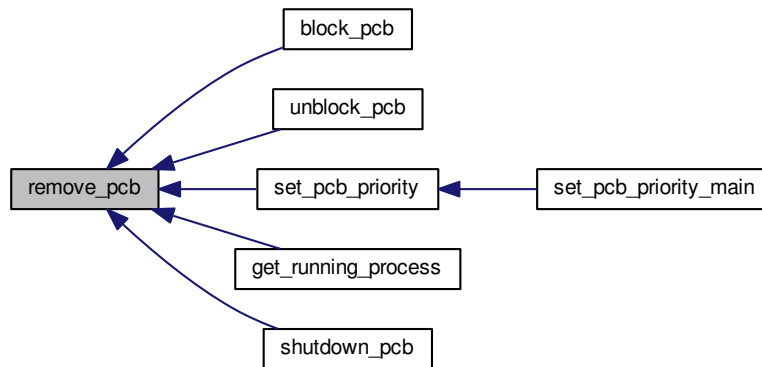
Here is the caller graph for this function:



#### 5.12.3.10 void pcb\_init ( )

#### 5.12.3.11 `error_t remove_pcb ( struct pcb_struct * pcb_ptr )`

Here is the caller graph for this function:



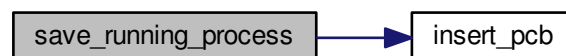
#### 5.12.3.12 `error_t resume_pcb ( struct pcb_struct * pcb_ptr )`

Here is the caller graph for this function:



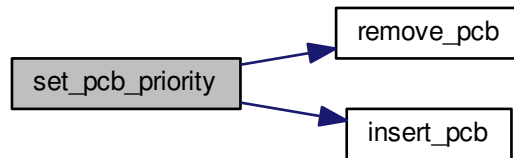
#### 5.12.3.13 `error_t save_running_process ( struct pcb_struct * pcb_ptr, struct context * new_stack_top )`

Here is the call graph for this function:



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

Here is the call graph for this function:

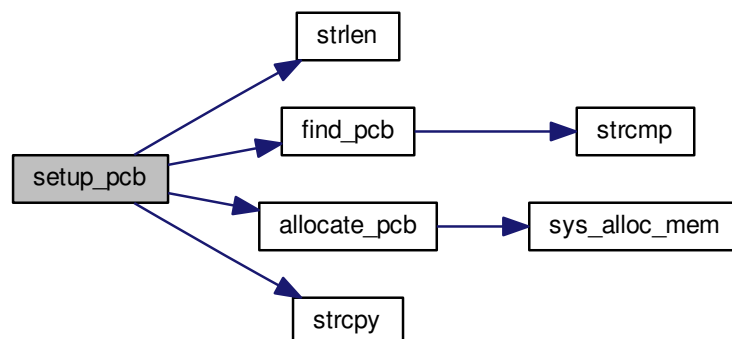


Here is the caller graph for this function:



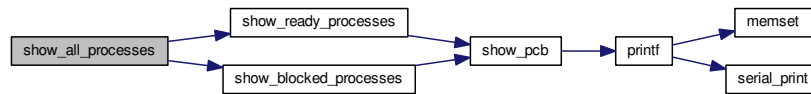
5.12.3.15 `struct pcb_struct* setup_pcb ( const char * pName, const enum process_class pClass, const unsigned char pPriority )`

Here is the call graph for this function:



### 5.12.3.16 void show\_all\_processes ( )

Here is the call graph for this function:

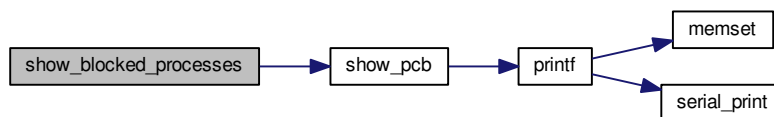


Here is the caller graph for this function:

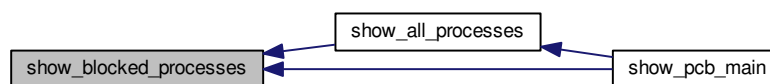


### 5.12.3.17 void show\_blocked\_processes ( )

Here is the call graph for this function:



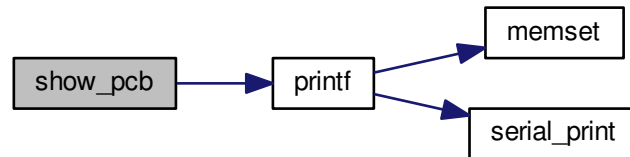
Here is the caller graph for this function:



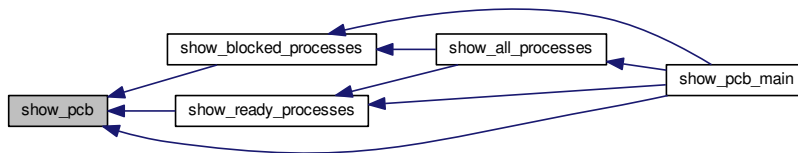


#### 5.12.3.18 `error_t show_pcb ( struct pcb_struct * pcb_ptr )`

Here is the call graph for this function:

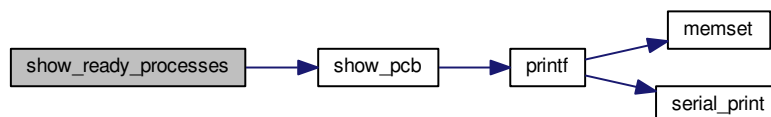


Here is the caller graph for this function:

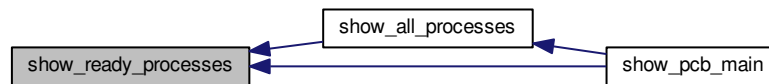


#### 5.12.3.19 `void show_ready_processes ( )`

Here is the call graph for this function:

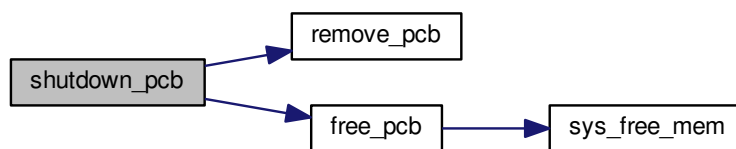


Here is the caller graph for this function:



#### 5.12.3.20 void shutdown\_pcb ( )

Here is the call graph for this function:



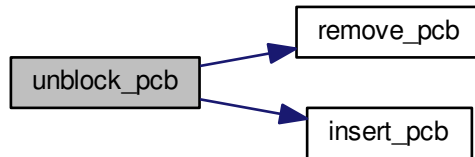
#### 5.12.3.21 error\_t suspend\_pcb ( struct pcb\_struct \* pcb\_ptr )

Here is the caller graph for this function:



#### 5.12.3.22 `error_t unblock_pcb ( struct pcb_struct * pcb_ptr )`

Here is the call graph for this function:



### 5.12.4 Variable Documentation

#### 5.12.4.1 `struct pcb_struct __attribute__`

##### 5.12.4.2 `blocked`

< PCB in the blocked state.

PCB in the blocked state.

##### 5.12.4.3 `false`

< PCB process is not suspended.

PCB process is not suspended.

##### 5.12.4.4 `ready`

PCB in the ready state.

##### 5.12.4.5 `running`

PCB in the running state.

##### 5.12.4.6 `true`

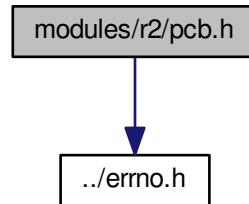
PCB process is suspended.

## 5.13 modules/r2/pcb.h File Reference

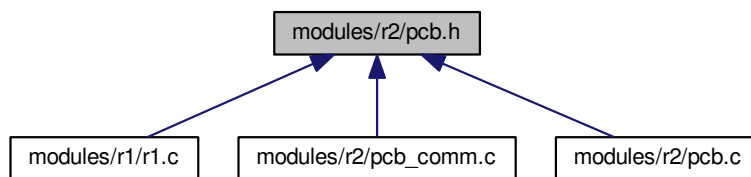
The Process Control Block.

```
#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

## 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 ~ 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.
pPriority	The assigned priority

**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.

**5.13.1 Detailed Description**

The Process Control Block.

**Author**

Thunder Krakens

**Date**

February 7th, 2016

**Version**

R3

### 5.13.2 Macro Definition Documentation

5.13.2.1 `#define SIZE_OF_PCB_NAME 10`

5.13.2.2 `#define SIZE_OF_STACK 1024`

### 5.13.3 Enumeration Type Documentation

5.13.3.1 `enum process_class`

PCB process class types.

### 5.13.4 Function Documentation

5.13.4.1 `enum process_class __attribute__((packed))`

5.13.4.2 `struct pcb_struct* allocate_pcb ( )`

Here is the call graph for this function:

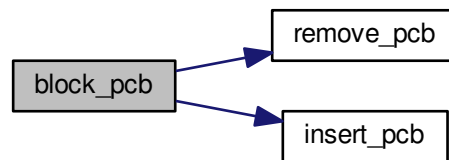


Here is the caller graph for this function:



#### 5.13.4.3 `error_t block_pcb ( struct pcb_struct * pcb_ptr )`

Here is the call graph for this function:

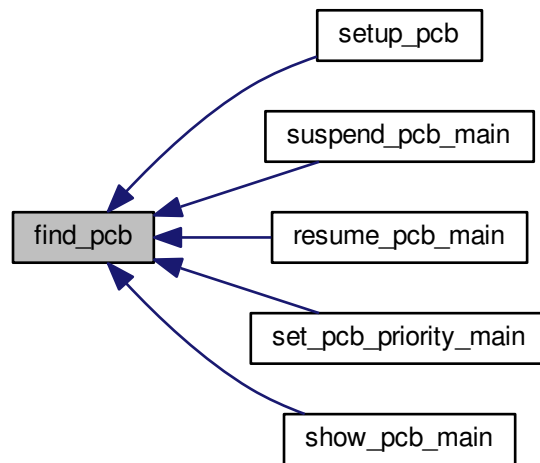


#### 5.13.4.4 `struct pcb_struct* find_pcb ( const char * pName )`

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.13.4.5 `error_t free_pcb ( struct pcb_struct * pcb_ptr )`

Here is the call graph for this function:

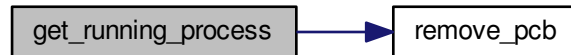


Here is the caller graph for this function:



#### 5.13.4.6 struct pcb\_struct\* get\_running\_process ( )

Here is the call graph for this function:

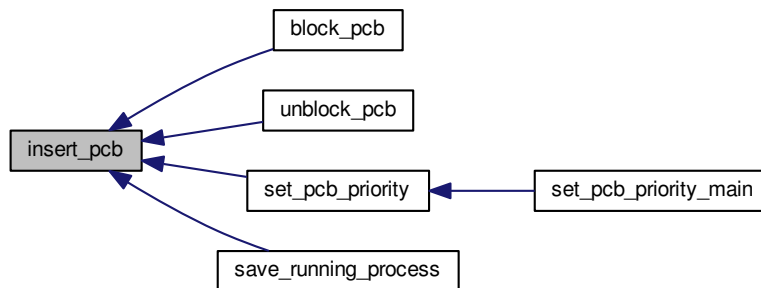


#### 5.13.4.7 unsigned char\* get\_stack\_base ( struct pcb\_struct \* pcb\_ptr )

#### 5.13.4.8 unsigned char\* get\_stack\_top ( struct pcb\_struct \* pcb\_ptr )

#### 5.13.4.9 error\_t insert\_pcb ( struct pcb\_struct \* pcb\_ptr )

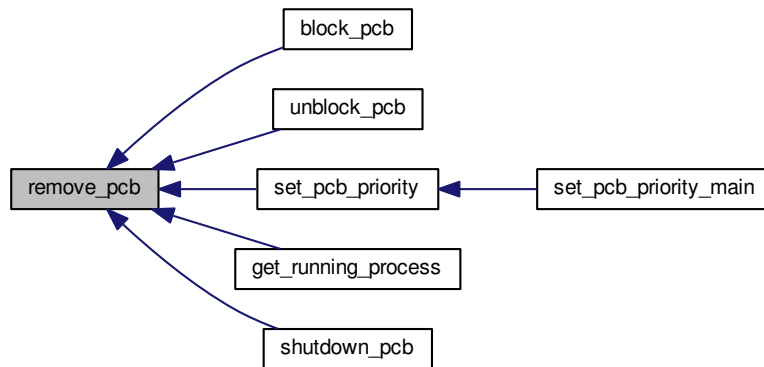
Here is the caller graph for this function:



#### 5.13.4.10 void pcb\_init ( )

#### 5.13.4.11 `error_t remove_pcb ( struct pcb_struct * pcb_ptr )`

Here is the caller graph for this function:



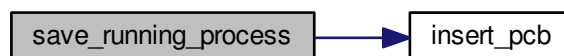
#### 5.13.4.12 `error_t resume_pcb ( struct pcb_struct * pcb_ptr )`

Here is the caller graph for this function:



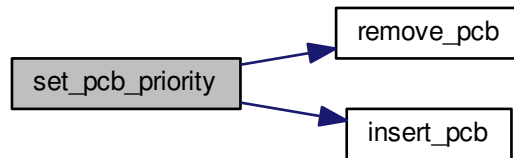
#### 5.13.4.13 `error_t save_running_process ( struct pcb_struct * pcb_ptr, struct context * new_stack_top )`

Here is the call graph for this function:



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

Here is the call graph for this function:

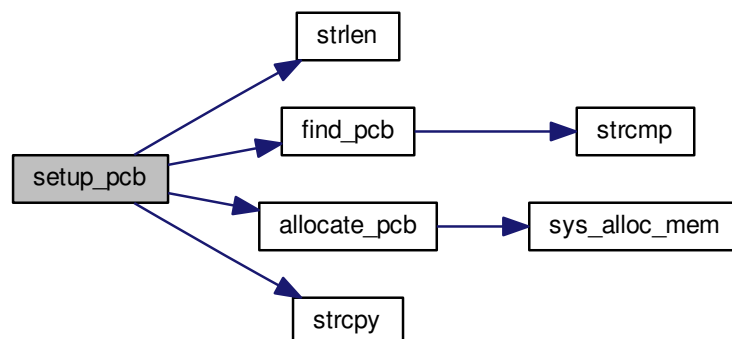


Here is the caller graph for this function:



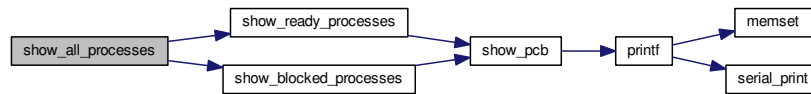
5.13.4.15 `struct pcb_struct* setup_pcb ( const char * pName, const enum process_class pClass, const unsigned char pPriority )`

Here is the call graph for this function:



#### 5.13.4.16 void show\_all\_processes ( )

Here is the call graph for this function:

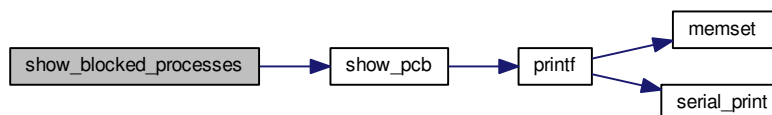


Here is the caller graph for this function:

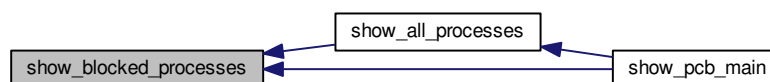


#### 5.13.4.17 void show\_blocked\_processes ( )

Here is the call graph for this function:



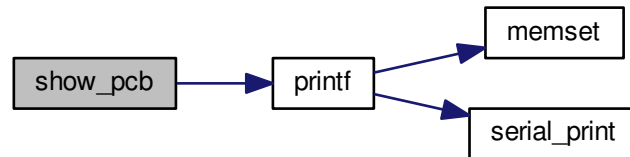
Here is the caller graph for this function:



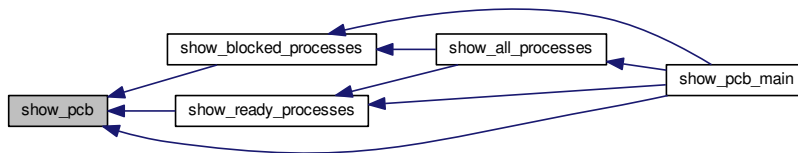


#### 5.13.4.18 `error_t show_pcb ( struct pcb_struct * pcb_ptr )`

Here is the call graph for this function:

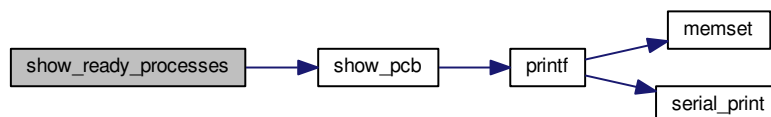


Here is the caller graph for this function:

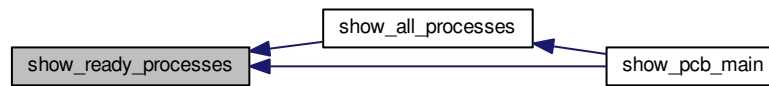


#### 5.13.4.19 `void show_ready_processes ( )`

Here is the call graph for this function:

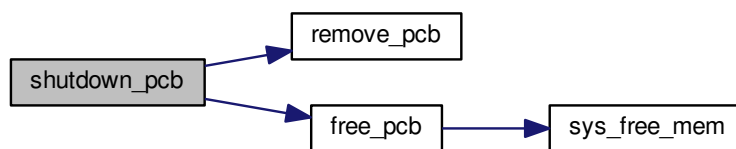


Here is the caller graph for this function:



#### 5.13.4.20 void shutdown\_pcb ( )

Here is the call graph for this function:



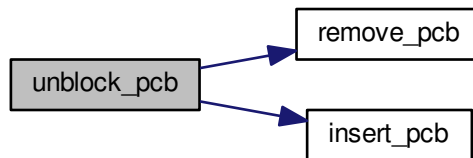
#### 5.13.4.21 error\_t suspend\_pcb ( struct pcb\_struct \* pcb\_ptr )

Here is the caller graph for this function:



#### 5.13.4.22 error\_t unblock\_pcb ( struct pcb\_struct \* pcb\_ptr )

Here is the call graph for this function:



### 5.13.5 Variable Documentation

#### 5.13.5.1 pcb\_class\_app

Process is an application process.

#### 5.13.5.2 pcb\_class\_sys

< Process is a system process.

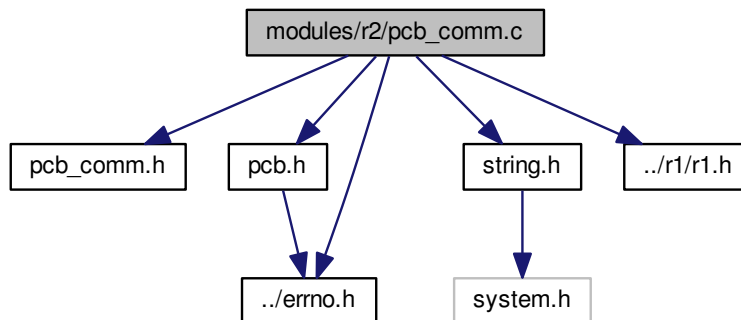
Process is a system process.

## 5.14 modules/r2/pcb\_comm.c File Reference

The main functions that manipulate the PCB.

```
#include "pcb_comm.h"
#include "pcb.h"
#include <string.h>
#include "../errno.h"
#include "../r1/r1.h"
```

Include dependency graph for pcb\_comm.c:



## Functions

### **suspend\_pcb\_main.**

*The main function for the "suspend PCB".*

*Accepted formats: `pcb suspend <name>` `pcb suspend -help`*

#### **Parameters**

argc	<i>The number of tokens found.</i>
argv	<i>The array of tokens.</i>

#### **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	<i>The number of tokens found.</i>
argv	<i>The array of tokens.</i>

#### **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	<i>The number of tokens found.</i>
argv	<i>The array of tokens.</i>

**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 [name] pcb show -all pcb show -ready pcb show -blocked pcb show -help*

**Parameters**

argc	<i>The number of tokens found.</i>
argv	<i>The array of tokens.</i>

**Returns**

0

- int [show\\_pcb\\_main](#) (int argc, char \*\*argv)

### 5.14.1 Detailed Description

The main functions that manipulate the PCB.

**Author**

Thunder Krakens

**Date**

February 7th, 2016

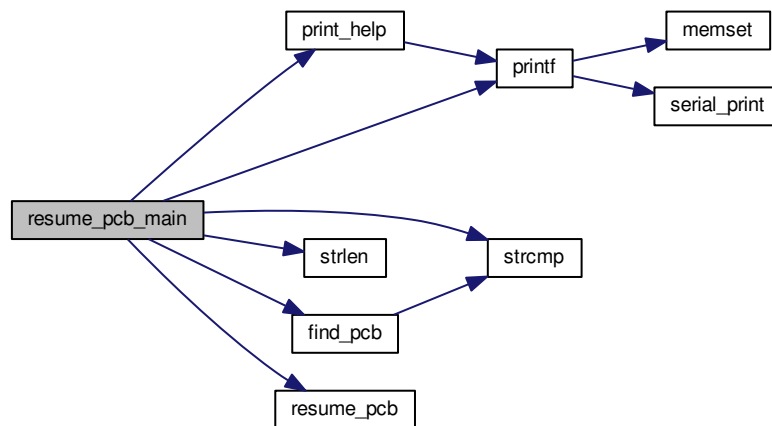
Version

R2

## 5.14.2 Function Documentation

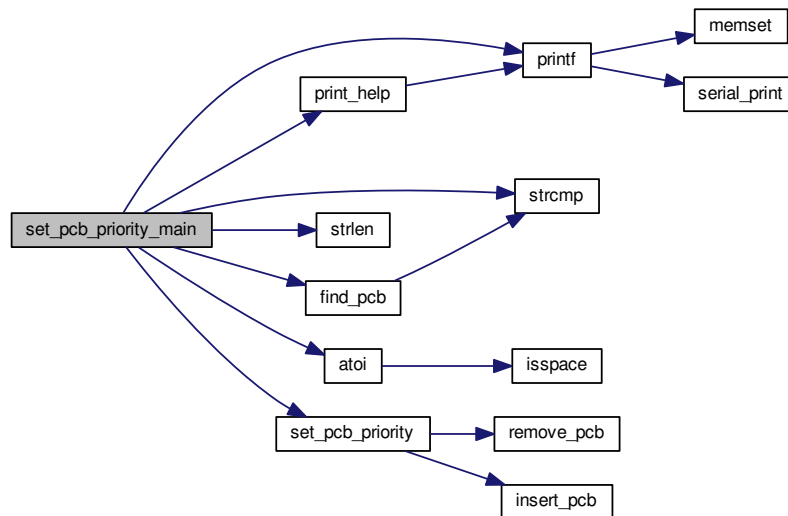
5.14.2.1 `int resume_pcb_main ( int argc, char ** argv )`

Here is the call graph for this function:



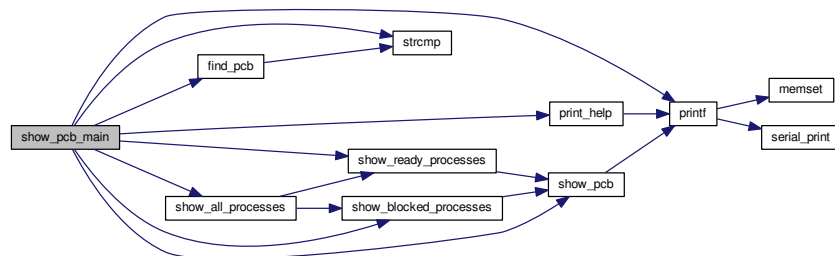
## 5.14.2.2 int set\_pcb\_priority\_main ( int argc, char \*\* argv )

Here is the call graph for this function:



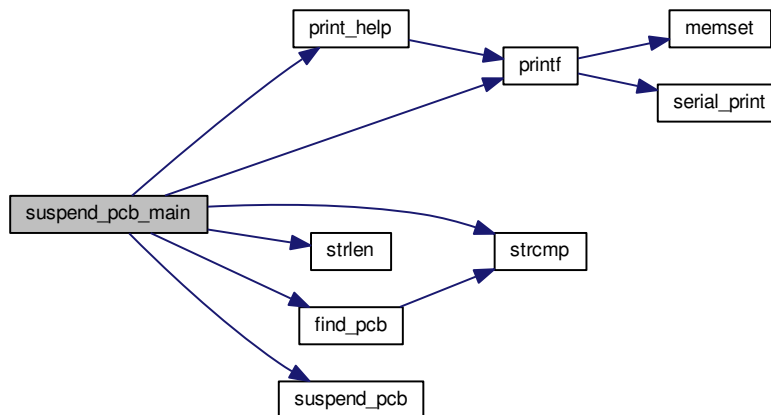
## 5.14.2.3 int show\_pcb\_main ( int argc, char \*\* argv )

Here is the call graph for this function:



#### 5.14.2.4 `int suspend_pcb_main ( int argc, char ** argv )`

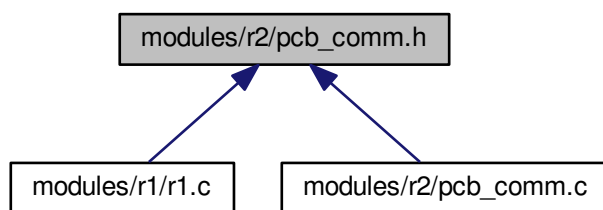
Here is the call graph for this function:



### 5.15 modules/r2/pcb\_comm.h File Reference

The main functions that manipulate the PCB.

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



## Functions

### **`suspend_pcb_main.`**

*The main function for the "suspend PCB".*

*Accepted formats: `pcb suspend <name>` `pcb suspend -help`*



**Parameters**

argc	<i>The number of tokens found.</i>
argv	<i>The array of tokens.</i>

**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	<i>The number of tokens found.</i>
argv	<i>The array of tokens.</i>

**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	<i>The number of tokens found.</i>
argv	<i>The array of tokens.</i>

**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	<i>The number of tokens found.</i>
argv	<i>The array of tokens.</i>

**Returns**

0

- int [show\\_pcb\\_main](#) (int argc, char \*\*argv)

**create\_pcb\_main.**

*The main function for the "Create PCB".*

*Accepted formats: pcb create <name> <type> <priority> pcb create -help*

**Parameters**

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

**Returns**

0

- int [create\\_pcb\\_main](#) (int argc, char \*\*argv)

**delete\_pcb\_main.**

The main function for the "Delete PCB".

Accepted formats: `pcb del <name>` `pcb del -help`

**Parameters**

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

**Returns**

0

- int [delete\\_pcb\\_main](#) (int argc, char \*\*argv)

**block\_pcb\_main.**

The main function for the "block PCB".

Accepted formats: `pcb block <name>` `pcb block -help`

**Parameters**

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

**Returns**

0

- int [block\\_pcb\\_main](#) (int argc, char \*\*argv)

**unblock\_pcb\_main.**

The main function for the "unblock PCB".

Accepted formats: `pcb unblock <name>` `pcb unblock -help`

**Parameters**

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

**Returns**

0

- int [unblock\\_pcb\\_main](#) (int argc, char \*\*argv)

### 5.15.1 Detailed Description

The main functions that manipulate the PCB.

#### Author

Thunder Krakens

#### Date

February 7th, 2016

#### Version

R2

### 5.15.2 Function Documentation

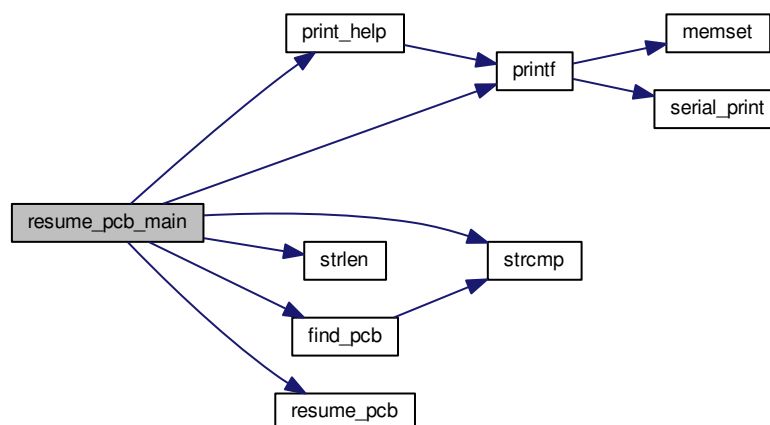
5.15.2.1 `int block_pcb_main ( int argc, char ** argv )`

5.15.2.2 `int create_pcb_main ( int argc, char ** argv )`

5.15.2.3 `int delete_pcb_main ( int argc, char ** argv )`

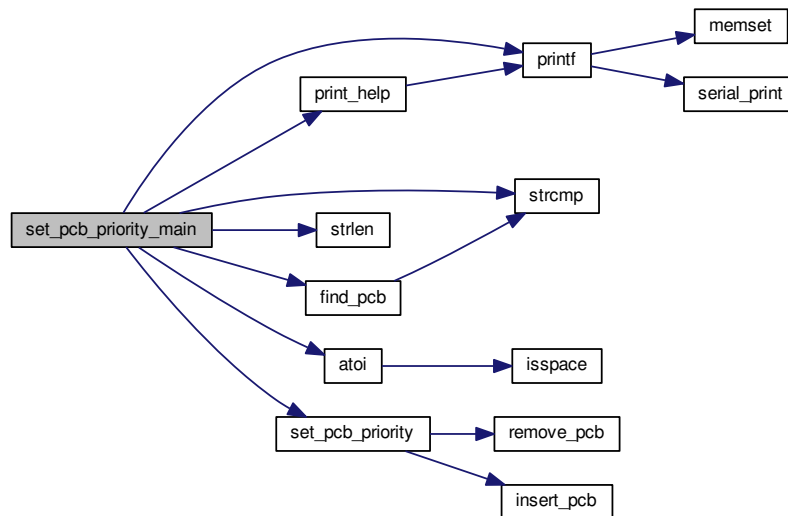
5.15.2.4 `int resume_pcb_main ( int argc, char ** argv )`

Here is the call graph for this function:



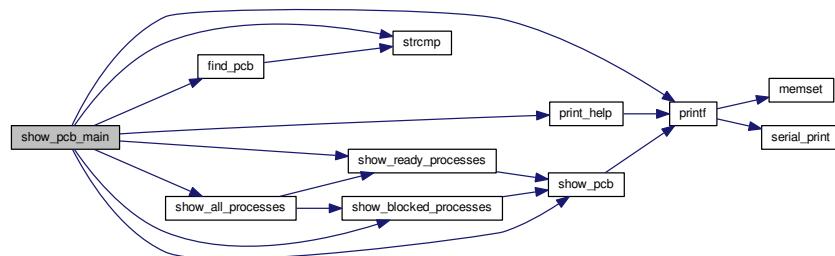
### 5.15.2.5 int set\_pcb\_priority\_main ( int argc, char \*\* argv )

Here is the call graph for this function:



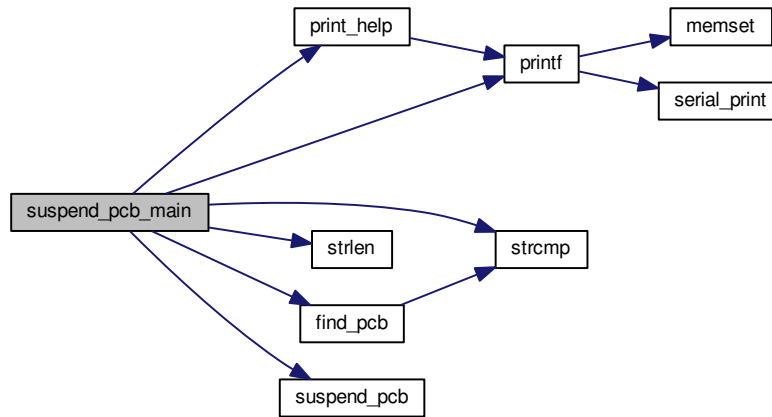
### 5.15.2.6 int show\_pcb\_main ( int argc, char \*\* argv )

Here is the call graph for this function:



#### 5.15.2.7 int suspend\_pcb\_main ( int *argc*, char \*\* *argv* )

Here is the call graph for this function:



#### 5.15.2.8 int unblock\_pcb\_main ( int *argc*, char \*\* *argv* )

# Index

- `__attribute__`
    - `mpx_supt.h`, [43](#)
    - `pcb.c`, [78](#), [87](#)
    - `pcb.h`, [94](#)
    - `r1.c`, [46](#)
    - `r1.h`, [51](#)
- `allocate_pcb`
  - `pcb.c`, [78](#)
  - `pcb.h`, [94](#)
- `atoi`
  - `string.c`, [29](#)
  - `string.h`, [19](#)
- `block_pcb`
  - `pcb.c`, [79](#)
  - `pcb.h`, [94](#)
- `block_pcb_main`
  - `pcb_comm.h`, [111](#)
- `blocked`
  - `pcb.c`, [87](#)
- `COM1`
  - `serial.h`, [14](#)
- `COM2`
  - `serial.h`, [14](#)
- `COM3`
  - `serial.h`, [14](#)
- `COM4`
  - `serial.h`, [14](#)
- `COMPLETION`
  - `r1.c`, [46](#)
- `class`
  - `pcb_struct`, [10](#)
- `comm_type`
  - `r1.h`, [51](#)
- `command_line_parser`
  - `r1.c`, [46](#)
  - `r1.h`, [51](#)
- `CommandPaserStat`
  - `r1.c`, [46](#)
- `commhand`
  - `r1.c`, [47](#)
  - `r1.h`, [52](#)
- `count`
  - `pcb_queue`, [9](#)
- `create_pcb_main`
  - `pcb_comm.h`, [111](#)
- `current_module`
  - `mpx_supt.c`, [38](#)
- `delete_pcb_main`
  - `pcb_comm.h`, [111](#)
- `device_id`
  - `param`, [8](#)
- `documentation/mainpage.dox`, [13](#)
- `DoubleQuoteWriting`
  - `r1.c`, [48](#)
- `E_EMPTPCB`
  - `errno.h`, [35](#)
- `E_FREEMEM`
  - `errno.h`, [35](#)
- `E_INVPARA`
  - `errno.h`, [35](#)
- `E_INVSTRF`
  - `errno.h`, [35](#)
- `E_INVUSRI`
  - `errno.h`, [35](#)
- `E_NOERROR`
  - `errno.h`, [35](#)
- `E_NULL_PTR`
  - `errno.h`, [35](#)
- `E_PROGERR`
  - `errno.h`, [35](#)
- `EXIT`
  - `mpx_supt.h`, [41](#)
- `errno.h`
  - `E_EMPTPCB`, [35](#)
  - `E_FREEMEM`, [35](#)
  - `E_INVPARA`, [35](#)
  - `E_INVSTRF`, [35](#)
  - `E_INVUSRI`, [35](#)
  - `E_NOERROR`, [35](#)
  - `E_NULL_PTR`, [35](#)
  - `E_PROGERR`, [35](#)
  - `error_t`, [35](#)
- `error_t`
  - `errno.h`, [35](#)
- `false`
  - `pcb.c`, [87](#)

find\_pcb  
    pcb.c, 79  
    pcb.h, 95  
free\_pcb  
    pcb.c, 80  
    pcb.h, 96  
function  
    function\_name, 7  
function\_name, 7  
    function, 7  
    help, 7  
    nameStr, 7  
    usage, 7  
  
GETDATE  
    r1.h, 51  
GETTIME  
    r1.h, 51  
get\_date  
    sys\_clock.c, 58  
    sys\_clock.h, 66  
get\_date\_main  
    sys\_clock.c, 58  
    sys\_clock.h, 67  
get\_input\_line  
    serial.h, 15  
get\_op\_code  
    mpx\_supt.c, 36  
    mpx\_supt.h, 41  
get\_running\_process  
    pcb.c, 80  
    pcb.h, 96  
get\_stack\_base  
    pcb.c, 81  
    pcb.h, 97  
get\_stack\_top  
    pcb.c, 81  
    pcb.h, 97  
get\_time  
    sys\_clock.c, 58  
    sys\_clock.h, 67  
get\_time\_main  
    sys\_clock.c, 59  
    sys\_clock.h, 67  
  
HELP  
    r1.h, 51  
head  
    pcb\_queue, 9  
help  
    function\_name, 7  
    r1.h, 53  
help\_usages  
    r1.c, 47  
    r1.h, 52  
  
IDLE  
    mpx\_supt.h, 41  
idle  
    mpx\_supt.c, 36  
    mpx\_supt.h, 41  
include/core/serial.h, 13  
include/string.h, 15  
init\_serial  
    serial.h, 15  
insert\_pcb  
    pcb.c, 81  
    pcb.h, 97  
is\_suspended  
    pcb\_struct, 10  
isspace  
    string.c, 29  
    string.h, 20  
  
LOADR3  
    r1.h, 51  
lib/string.c, 25  
  
MAX\_ARGC  
    r1.c, 46  
MAX\_HISTORY  
    r1.c, 46  
MOD\_VERSION  
    r1.c, 46  
MODULE\_R1  
    mpx\_supt.h, 41  
MODULE\_R2  
    mpx\_supt.h, 41  
MODULE\_R3  
    mpx\_supt.h, 41  
MODULE\_R4  
    mpx\_supt.h, 41  
MODULE\_R5  
    mpx\_supt.h, 41  
memset  
    string.c, 30  
    string.h, 20  
modules/errno.h, 34  
modules/mpx\_supt.c, 36  
modules/mpx\_supt.h, 38  
modules/r1/r1.c, 43  
modules/r1/r1.h, 48  
modules/r1/sys\_clock.c, 53  
modules/r1/sys\_clock.h, 63  
modules/r2/pcb.c, 72  
modules/r2/pcb.h, 87  
modules/r2/pcb\_comm.c, 103  
modules/r2/pcb\_comm.h, 108  
mpx  
    r1.h, 53  
mpx\_init

- mpx\_supt.c, 37
- mpx\_supt.h, 42
- mpx\_supt.c
  - current\_module, 38
  - get\_op\_code, 36
  - idle, 36
  - mpx\_init, 37
  - params, 38
  - student\_free, 38
  - student\_malloc, 38
  - sys\_alloc\_mem, 37
  - sys\_free\_mem, 37
  - sys\_req, 37
  - sys\_set\_free, 38
  - sys\_set\_malloc, 38
- mpx\_supt.h
  - \_\_attribute\_\_, 43
  - EXIT, 41
  - get\_op\_code, 41
  - IDLE, 41
  - idle, 41
  - MODULE\_R1, 41
  - MODULE\_R2, 41
  - MODULE\_R3, 41
  - MODULE\_R4, 41
  - MODULE\_R5, 41
  - mpx\_init, 42
  - READ, 41
  - sys\_alloc\_mem, 42
  - sys\_free\_mem, 42
  - sys\_req, 42
  - sys\_set\_free, 43
  - sys\_set\_malloc, 43
  - WRITE, 41
- NUM\_MPX\_FUNCTIONS
  - r1.h, 51
- NUM\_OF\_FUNCTIONS
  - r1.h, 51
- name
  - pcb\_struct, 11
- nameStr
  - function\_name, 7
- next
  - pcb\_struct, 11
- NormalWriting
  - r1.c, 48
- NotWriting
  - r1.c, 48
- op\_code
  - param, 8
- POS\_OF\_MPX
  - r1.h, 51
- POS\_OF\_PCB
  - r1.h, 51
- param, 8
  - device\_id, 8
  - op\_code, 8
- params
  - mpx\_supt.c, 38
- pcb
  - r1.h, 53
- pcb.c
  - \_\_attribute\_\_, 78, 87
  - allocate\_pcb, 78
  - block\_pcb, 79
  - blocked, 87
  - false, 87
  - find\_pcb, 79
  - free\_pcb, 80
  - get\_running\_process, 80
  - get\_stack\_base, 81
  - get\_stack\_top, 81
  - insert\_pcb, 81
  - pcb\_init, 81
  - process\_state, 78
  - process\_suspended, 78
  - ready, 87
  - remove\_pcb, 81
  - resume\_pcb, 82
  - running, 87
  - save\_running\_process, 82
  - set\_pcb\_priority, 82
  - setup\_pcb, 83
  - show\_all\_processes, 84
  - show\_blocked\_processes, 84
  - show\_pcb, 84
  - show\_ready\_processes, 85
  - shutdown\_pcb, 86
  - suspend\_pcb, 86
  - true, 87
  - unblock\_pcb, 86
- pcb.h
  - \_\_attribute\_\_, 94
  - allocate\_pcb, 94
  - block\_pcb, 94
  - find\_pcb, 95
  - free\_pcb, 96
  - get\_running\_process, 96
  - get\_stack\_base, 97
  - get\_stack\_top, 97
  - insert\_pcb, 97
  - pcb\_class\_app, 103
  - pcb\_class\_sys, 103
  - pcb\_init, 97
  - process\_class, 94
  - remove\_pcb, 97



- resume\_pcb, 98
- SIZE\_OF\_PCB\_NAME, 94
- SIZE\_OF\_STACK, 94
- save\_running\_process, 98
- set\_pcb\_priority, 98
- setup\_pcb, 99
- show\_all\_processes, 100
- show\_blocked\_processes, 100
- show\_pcb, 100
- show\_ready\_processes, 101
- shutdown\_pcb, 102
- suspend\_pcb, 102
- unblock\_pcb, 102
- pcb\_class\_app
  - pcb.h, 103
- pcb\_class\_sys
  - pcb.h, 103
- pcb\_comm.c
  - resume\_pcb\_main, 106
  - set\_pcb\_priority\_main, 106
  - show\_pcb\_main, 107
  - suspend\_pcb\_main, 107
- pcb\_comm.h
  - block\_pcb\_main, 111
  - create\_pcb\_main, 111
  - delete\_pcb\_main, 111
  - resume\_pcb\_main, 111
  - set\_pcb\_priority\_main, 111
  - show\_pcb\_main, 112
  - suspend\_pcb\_main, 112
  - unblock\_pcb\_main, 113
- pcb\_init
  - pcb.c, 81
  - pcb.h, 97
- pcb\_queue, 8
  - count, 9
  - head, 9
  - tail, 9
- pcb\_struct, 10
  - class, 10
  - is\_suspended, 10
  - name, 11
  - next, 11
  - prev, 11
  - priority, 11
  - running\_state, 11
  - stack\_base, 11
  - stack\_top, 11
- prev
  - pcb\_struct, 11
- print\_help
  - r1.c, 47
  - r1.h, 52
- printf
  - string.c, 30
  - string.h, 21
- priority
  - pcb\_struct, 11
- process\_class
  - pcb.h, 94
- process\_state
  - pcb.c, 78
- process\_suspended
  - pcb.c, 78
- r1.c
  - \_\_attribute\_\_, 46
  - COMPLETION, 46
  - command\_line\_parser, 46
  - CommandPaserStat, 46
  - commhand, 47
  - DoubleQuoteWriting, 48
  - help\_usages, 47
  - MAX\_ARGC, 46
  - MAX\_HISTORY, 46
  - MOD\_VERSION, 46
  - NormalWriting, 48
  - NotWriting, 48
  - print\_help, 47
  - SingleQuoteWriting, 48
- r1.h
  - \_\_attribute\_\_, 51
  - comm\_type, 51
  - command\_line\_parser, 51
  - commhand, 52
  - GETDATE, 51
  - GETTIME, 51
  - HELP, 51
  - help, 53
  - help\_usages, 52
  - LOADR3, 51
  - mpx, 53
  - NUM\_MPX\_FUNCTIONS, 51
  - NUM\_OF\_FUNCTIONS, 51
  - POS\_OF\_MPX, 51
  - POS\_OF\_PCB, 51
  - pcb, 53
  - print\_help, 52
  - RESUMEPCB, 51
  - SETDATE, 51
  - SETPCBPRIO, 51
  - SETTIME, 51
  - SHOWPCB, 51
  - SHUTDOWN, 51
  - SUSPDCB, 51
  - VERSION, 51
  - YIELD, 51
- READ

- mpx\_supt.h, [41](#)
- RESUMEPCB
  - r1.h, [51](#)
- RTC\_INDEX\_HOUR
  - sys\_clock.c, [57](#)
- RTC\_INDEX\_MINUTE
  - sys\_clock.c, [57](#)
- RTC\_INDEX\_MONTH
  - sys\_clock.c, [57](#)
- RTC\_INDEX\_SECOND
  - sys\_clock.c, [58](#)
- RTC\_INDEX\_YEAR
  - sys\_clock.c, [58](#)
- ready
  - pcb.c, [87](#)
- remove\_pcb
  - pcb.c, [81](#)
  - pcb.h, [97](#)
- resume\_pcb
  - pcb.c, [82](#)
  - pcb.h, [98](#)
- resume\_pcb\_main
  - pcb\_comm.c, [106](#)
  - pcb\_comm.h, [111](#)
- running
  - pcb.c, [87](#)
- running\_state
  - pcb\_struct, [11](#)
- SETDATE
  - r1.h, [51](#)
- SETPCBPRIO
  - r1.h, [51](#)
- SETTIME
  - r1.h, [51](#)
- SHOWPCB
  - r1.h, [51](#)
- SHUTDOWN
  - r1.h, [51](#)
- SIZE\_OF\_PCB\_NAME
  - pcb.h, [94](#)
- SIZE\_OF\_STACK
  - pcb.h, [94](#)
- SUSPDPCB
  - r1.h, [51](#)
- save\_running\_process
  - pcb.c, [82](#)
  - pcb.h, [98](#)
- serial.h
  - COM1, [14](#)
  - COM2, [14](#)
  - COM3, [14](#)
  - COM4, [14](#)
  - get\_input\_line, [15](#)
  - init\_serial, [15](#)
  - serial\_print, [15](#)
  - serial\_println, [15](#)
  - set\_serial\_in, [15](#)
  - set\_serial\_out, [15](#)
  - WithEcho, [14](#)
  - WithoutEcho, [14](#)
- serial\_print
  - serial.h, [15](#)
- serial\_println
  - serial.h, [15](#)
- set\_date
  - sys\_clock.c, [59](#)
  - sys\_clock.h, [68](#)
- set\_date\_main
  - sys\_clock.c, [60](#)
  - sys\_clock.h, [68](#)
- set\_date\_str
  - sys\_clock.c, [60](#)
  - sys\_clock.h, [69](#)
- set\_pcb\_priority
  - pcb.c, [82](#)
  - pcb.h, [98](#)
- set\_pcb\_priority\_main
  - pcb\_comm.c, [106](#)
  - pcb\_comm.h, [111](#)
- set\_serial\_in
  - serial.h, [15](#)
- set\_serial\_out
  - serial.h, [15](#)
- set\_time
  - sys\_clock.c, [61](#)
  - sys\_clock.h, [70](#)
- set\_time\_main
  - sys\_clock.c, [61](#)
  - sys\_clock.h, [70](#)
- set\_time\_str
  - sys\_clock.c, [62](#)
  - sys\_clock.h, [71](#)
- setup\_pcb
  - pcb.c, [83](#)
  - pcb.h, [99](#)
- show\_all\_processes
  - pcb.c, [84](#)
  - pcb.h, [100](#)
- show\_blocked\_processes
  - pcb.c, [84](#)
  - pcb.h, [100](#)
- show\_pcb
  - pcb.c, [84](#)
  - pcb.h, [100](#)
- show\_pcb\_main
  - pcb\_comm.c, [107](#)
  - pcb\_comm.h, [112](#)

show\_ready\_processes  
    pcb.c, 85  
    pcb.h, 101  
shutdown\_pcb  
    pcb.c, 86  
    pcb.h, 102  
SingleQuoteWriting  
    r1.c, 48  
sprintf  
    string.c, 31  
    string.h, 22  
stack\_base  
    pcb\_struct, 11  
stack\_top  
    pcb\_struct, 11  
strcat  
    string.c, 31  
    string.h, 22  
strcmp  
    string.c, 31  
    string.h, 22  
strcpy  
    string.c, 32  
    string.h, 23  
string.c  
    atoi, 29  
    isspace, 29  
    memset, 30  
    printf, 30  
    sprintf, 31  
    strcat, 31  
    strcmp, 31  
    strcpy, 32  
    strlen, 33  
    strtok, 33  
string.h  
    atoi, 19  
    isspace, 20  
    memset, 20  
    printf, 21  
    sprintf, 22  
    strcat, 22  
    strcmp, 22  
    strcpy, 23  
    strlen, 24  
    strtok, 24  
strlen  
    string.c, 33  
    string.h, 24  
strtok  
    string.c, 33  
    string.h, 24  
student\_free  
    mpx\_supt.c, 38  
student\_malloc  
    mpx\_supt.c, 38  
suspend\_pcb  
    pcb.c, 86  
    pcb.h, 102  
suspend\_pcb\_main  
    pcb\_comm.c, 107  
    pcb\_comm.h, 112  
sys\_alloc\_mem  
    mpx\_supt.c, 37  
    mpx\_supt.h, 42  
sys\_clock.c  
    get\_date, 58  
    get\_date\_main, 58  
    get\_time, 58  
    get\_time\_main, 59  
    RTC\_INDEX\_HOUR, 57  
    RTC\_INDEX\_MINUTE, 57  
    RTC\_INDEX\_MONTH, 57  
    RTC\_INDEX\_SECOND, 58  
    RTC\_INDEX\_YEAR, 58  
    set\_date, 59  
    set\_date\_main, 60  
    set\_date\_str, 60  
    set\_time, 61  
    set\_time\_main, 61  
    set\_time\_str, 62  
sys\_clock.h  
    get\_date, 66  
    get\_date\_main, 67  
    get\_time, 67  
    get\_time\_main, 67  
    set\_date, 68  
    set\_date\_main, 68  
    set\_date\_str, 69  
    set\_time, 70  
    set\_time\_main, 70  
    set\_time\_str, 71  
sys\_free\_mem  
    mpx\_supt.c, 37  
    mpx\_supt.h, 42  
sys\_req  
    mpx\_supt.c, 37  
    mpx\_supt.h, 42  
sys\_set\_free  
    mpx\_supt.c, 38  
    mpx\_supt.h, 43  
sys\_set\_malloc  
    mpx\_supt.c, 38  
    mpx\_supt.h, 43  
tail  
    pcb\_queue, 9  
true

pcb.c, [87](#)

unblock\_pcb

pcb.c, [86](#)

pcb.h, [102](#)

unblock\_pcb\_main

pcb\_comm.h, [113](#)

usage

function\_name, [7](#)

VERSION

r1.h, [51](#)

WRITE

mpx\_supt.h, [41](#)

WithEcho

serial.h, [14](#)

WithoutEcho

serial.h, [14](#)

YIELD

r1.h, [51](#)