

MPX Thunder Krakens
R2

Generated by Doxygen 1.8.9.1

Thu Feb 25 2016 01:47:53

Contents

1	Main Page	1
2	Data Structure Index	3
2.1	Data Structures	3
3	File Index	5
3.1	File List	5
4	Data Structure Documentation	7
4.1	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	pcb_queue Struct Reference	8
4.2.1	Detailed Description	8
4.2.2	Field Documentation	8
4.2.2.1	count	8
4.2.2.2	head	9
4.2.2.3	tail	9
4.3	pcb_struct Struct Reference	9
4.3.1	Detailed Description	10
4.3.2	Field Documentation	10
4.3.2.1	class	10
4.3.2.2	is_suspended	10
4.3.2.3	name	10
4.3.2.4	next	10

4.3.2.5	prev	10
4.3.2.6	priority	10
4.3.2.7	running_state	10
4.3.2.8	stack_base	10
4.3.2.9	stack_top	10
5	File Documentation	11
5.1	documentation/mainpage.dox File Reference	11
5.2	include/core/serial.h File Reference	11
5.2.1	Detailed Description	12
5.2.2	Macro Definition Documentation	12
5.2.2.1	COM1	12
5.2.2.2	COM2	12
5.2.2.3	COM3	12
5.2.2.4	COM4	12
5.2.2.5	WithEcho	12
5.2.2.6	WithoutEcho	12
5.2.3	Function Documentation	12
5.2.3.1	get_input_line	13
5.2.3.2	init_serial	13
5.2.3.3	serial_print	13
5.2.3.4	serial_println	13
5.2.3.5	set_serial_in	13
5.2.3.6	set_serial_out	13
5.3	include/string.h File Reference	13
5.3.1	Detailed Description	17
5.3.2	Function Documentation	17
5.3.2.1	atoi	18
5.3.2.2	isspace	18
5.3.2.3	memset	19
5.3.2.4	printf	19
5.3.2.5	sprintf	20
5.3.2.6	strcat	20
5.3.2.7	strcmp	21
5.3.2.8	strcpy	22
5.3.2.9	strlen	23
5.3.2.10	strtok	23

5.4	lib/string.c File Reference	24
5.4.1	Detailed Description	27
5.4.2	Function Documentation	28
5.4.2.1	atoi	28
5.4.2.2	isspace	28
5.4.2.3	memset	29
5.4.2.4	printf	29
5.4.2.5	sprintf	30
5.4.2.6	strcat	30
5.4.2.7	strcmp	31
5.4.2.8	strcpy	32
5.4.2.9	strlen	33
5.4.2.10	strtok	33
5.5	modules/errno.h File Reference	34
5.5.1	Detailed Description	34
5.5.2	Macro Definition Documentation	35
5.5.2.1	E_FREEMEM	35
5.5.2.2	E_INVPARA	35
5.5.2.3	E_INVSTRF	35
5.5.2.4	E_INVUSRI	35
5.5.2.5	E_NOERROR	35
5.5.2.6	E_NULL_PTR	35
5.5.2.7	E_PROGERR	35
5.5.3	Typedef Documentation	35
5.5.3.1	error_t	35
5.6	modules/r1/r1.c File Reference	35
5.6.1	Detailed Description	38
5.6.2	Macro Definition Documentation	38
5.6.2.1	COMPLETION	38
5.6.2.2	MAX_ARGC	38
5.6.2.3	MOD_VERSION	38
5.6.2.4	USER_INPUT_BUFFER_SIZE	38
5.6.3	Enumeration Type Documentation	38
5.6.3.1	CommandPaserStat	39
5.6.4	Function Documentation	39
5.6.4.1	__attribute__	39
5.6.4.2	command_line_parser	39

5.6.4.3	commhand	39
5.6.4.4	help_usages	39
5.6.4.5	print_help	39
5.6.5	Variable Documentation	40
5.6.5.1	DoubleQuoteWriting	40
5.6.5.2	NormalWriting	40
5.6.5.3	NotWriting	41
5.6.5.4	SingleQuoteWriting	41
5.7	modules/r1/r1.h File Reference	41
5.7.1	Detailed Description	42
5.7.2	Macro Definition Documentation	43
5.7.2.1	BLOCKPCB	43
5.7.2.2	CREATEPCB	43
5.7.2.3	DELPCB	43
5.7.2.4	GETDATE	43
5.7.2.5	GETTIME	43
5.7.2.6	HELP	43
5.7.2.7	NUM_OF_FUNCTIONS	43
5.7.2.8	RESUMEPCB	43
5.7.2.9	SETDATE	43
5.7.2.10	SETPCBPRIO	43
5.7.2.11	SETTIME	43
5.7.2.12	SHOWPCB	43
5.7.2.13	SHUTDOWN	43
5.7.2.14	SUSPDPCB	43
5.7.2.15	UNBLKPCB	43
5.7.2.16	VERSION	43
5.7.3	Enumeration Type Documentation	43
5.7.3.1	comm_type	43
5.7.4	Function Documentation	43
5.7.4.1	__attribute__	43
5.7.4.2	command_line_parser	43
5.7.4.3	commhand	43
5.7.4.4	help_usages	44
5.7.4.5	print_help	44
5.7.5	Variable Documentation	45
5.7.5.1	help	45

5.7.5.2	mpx	45
5.7.5.3	pcb	46
5.8	modules/r1/sys_clock.c File Reference	46
5.8.1	Detailed Description	50
5.8.2	Macro Definition Documentation	51
5.8.2.1	RTC_INDEX_DAY_MONTH	51
5.8.2.2	RTC_INDEX_DAY_WEEK	51
5.8.2.3	RTC_INDEX_HOUR	51
5.8.2.4	RTC_INDEX_HOUR_ALARM	51
5.8.2.5	RTC_INDEX_MINUTE	51
5.8.2.6	RTC_INDEX_MINUTE_ALARM	51
5.8.2.7	RTC_INDEX_MONTH	51
5.8.2.8	RTC_INDEX_SECOND	51
5.8.2.9	RTC_INDEX_SECOND_ALARM	51
5.8.2.10	RTC_INDEX_YEAR	51
5.8.3	Function Documentation	51
5.8.3.1	get_date	51
5.8.3.2	get_date_main	52
5.8.3.3	get_time	52
5.8.3.4	get_time_main	53
5.8.3.5	set_date	53
5.8.3.6	set_date_main	54
5.8.3.7	set_date_str	54
5.8.3.8	set_time	55
5.8.3.9	set_time_main	56
5.8.3.10	set_time_str	56
5.9	modules/r1/sys_clock.h File Reference	57
5.9.1	Detailed Description	60
5.9.2	Function Documentation	60
5.9.2.1	get_date	60
5.9.2.2	get_date_main	61
5.9.2.3	get_time	61
5.9.2.4	get_time_main	62
5.9.2.5	set_date	62
5.9.2.6	set_date_main	63
5.9.2.7	set_date_str	63
5.9.2.8	set_time	64

5.9.2.9	set_time_main	65
5.9.2.10	set_time_str	65
5.10	modules/r2/pcb.c File Reference	66
5.10.1	Detailed Description	71
5.10.2	Enumeration Type Documentation	71
5.10.2.1	process_state	71
5.10.2.2	process_suspended	71
5.10.3	Function Documentation	71
5.10.3.1	__attribute__	71
5.10.3.2	allocate_pcb	71
5.10.3.3	block_pcb	72
5.10.3.4	find_pcb	72
5.10.3.5	free_pcb	73
5.10.3.6	insert_pcb	74
5.10.3.7	pcb_init	74
5.10.3.8	remove_pcb	74
5.10.3.9	resume_pcb	75
5.10.3.10	set_pcb_priority	75
5.10.3.11	setup_pcb	76
5.10.3.12	show_all_processes	76
5.10.3.13	show_blocked_processes	77
5.10.3.14	show_pcb	78
5.10.3.15	show_ready_processes	78
5.10.3.16	suspend_pcb	79
5.10.3.17	unblock_pcb	79
5.10.4	Variable Documentation	80
5.10.4.1	__attribute__	80
5.10.4.2	blocked	80
5.10.4.3	false	80
5.10.4.4	ready	80
5.10.4.5	running	80
5.10.4.6	true	80
5.11	modules/r2/pcb.h File Reference	80
5.11.1	Detailed Description	85
5.11.2	Macro Definition Documentation	86
5.11.2.1	SIZE_OF_PCB_NAME	86
5.11.2.2	SIZE_OF_STACK	86

5.11.3	Enumeration Type Documentation	86
5.11.3.1	process_class	86
5.11.4	Function Documentation	86
5.11.4.1	__attribute__	86
5.11.4.2	allocate_pcb	86
5.11.4.3	block_pcb	86
5.11.4.4	find_pcb	87
5.11.4.5	free_pcb	88
5.11.4.6	insert_pcb	89
5.11.4.7	pcb_init	89
5.11.4.8	remove_pcb	89
5.11.4.9	resume_pcb	90
5.11.4.10	set_pcb_priority	90
5.11.4.11	setup_pcb	91
5.11.4.12	show_all_processes	91
5.11.4.13	show_blocked_processes	92
5.11.4.14	show_pcb	93
5.11.4.15	show_ready_processes	93
5.11.4.16	suspend_pcb	94
5.11.4.17	unblock_pcb	94
5.11.5	Variable Documentation	95
5.11.5.1	pcb_class_app	95
5.11.5.2	pcb_class_sys	95
5.12	modules/r2/pcb_comm.c File Reference	95
5.12.1	Detailed Description	98
5.12.2	Function Documentation	98
5.12.2.1	block_pcb_main	98
5.12.2.2	create_pcb_main	99
5.12.2.3	delete_pcb_main	99
5.12.2.4	resume_pcb_main	100
5.12.2.5	set_pcb_priority_main	100
5.12.2.6	show_pcb_main	101
5.12.2.7	suspend_pcb_main	101
5.12.2.8	unblock_pcb_main	102
5.13	modules/r2/pcb_comm.h File Reference	102
5.13.1	Detailed Description	105
5.13.2	Function Documentation	105

5.13.2.1	block_pcb_main	105
5.13.2.2	create_pcb_main	106
5.13.2.3	delete_pcb_main	106
5.13.2.4	resume_pcb_main	107
5.13.2.5	set_pcb_priority_main	107
5.13.2.6	show_pcb_main	108
5.13.2.7	suspend_pcb_main	108
5.13.2.8	unblock_pcb_main	109

Index**111**

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:

function_name	A structure to represent each function	7
pcb_queue	Queue structure that will store PCBs	8
pcb_struct	Struct that will describe PCB Processes	9

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

include/ string.h	Many usefull functions that used for handling string	13
include/core/ serial.h	Serial - Header	11
lib/ string.c	Many usefull functions that used for handling string	24
modules/ errno.h	This file contains the type of errors. The error can be from invalid paramter passed to a function, or invalid input format	34
modules/r1/ r1.c	The commandhander and functions associations for Module R1	35
modules/r1/ r1.h	The commandhander and functions associations for Module R1	41
modules/r1/ sys_clock.c	The main file that manipulates and controls the system's clock	46
modules/r1/ sys_clock.h	The main file that manipulates and controls the system's clock	57
modules/r2/ pcb.c	The Process Control Block	66
modules/r2/ pcb.h	The Process Control Block	80
modules/r2/ pcb_comm.c	The main functions that manipulate the PCB	95
modules/r2/ pcb_comm.h	The main functions that manipulate the PCB	102

Chapter 4

Data Structure Documentation

4.1 function_name Struct Reference

A structure to represent each function.

Data Fields

- char * [nameStr](#)
function'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

function's name

4.1.2.4 char* function_name::usage

function's usage or use cases

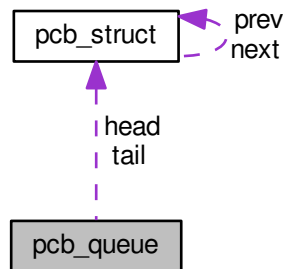
The documentation for this struct was generated from the following file:

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

4.2 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.

4.2.1 Detailed Description

Queue structure that will store PCBs.

4.2.2 Field Documentation

4.2.2.1 int pcb_queue::count

The length of the queue.

4.2.2.2 struct pcb_struct* pcb_queue::head

Pointer to the start/head of the queue.

4.2.2.3 struct pcb_struct* pcb_queue::tail

Pointer to the end/tail of the queue.

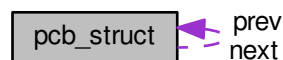
The documentation for this struct was generated from the following file:

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

4.3 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.

4.3.1 Detailed Description

Struct that will describe PCB Processes.

4.3.2 Field Documentation

4.3.2.1 `enum process_class pcb_struct::class`

PCB's class is an application or system process.

4.3.2.2 `enum process_suspended pcb_struct::is_suspended`

PCB process is either suspended or not suspended.

4.3.2.3 `char pcb_struct::name[SIZE_OF_PCB_NAME]`

PCB's name.

4.3.2.4 `struct pcb_struct* pcb_struct::next`

Pointer to the next PCB in the queue.

4.3.2.5 `struct pcb_struct* pcb_struct::prev`

Pointer to the previous PCB in the queue.

4.3.2.6 `unsigned char pcb_struct::priority`

PCB's priority an integer between 0 and 9.

Processes with higher priority values execute before lower priority processes.

4.3.2.7 `enum process_state pcb_struct::running_state`

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

4.3.2.8 `unsigned char* pcb_struct::stack_base`

Pointer to base of the stack.

4.3.2.9 `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

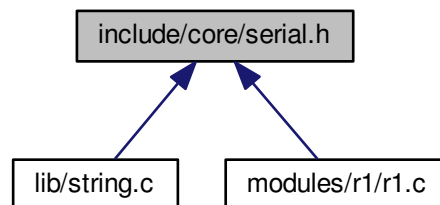
File Documentation

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

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 keyboard.

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 buffer_size, 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 WithEcho 1**

5.2.2.6 **#define WithoutEcho 0**

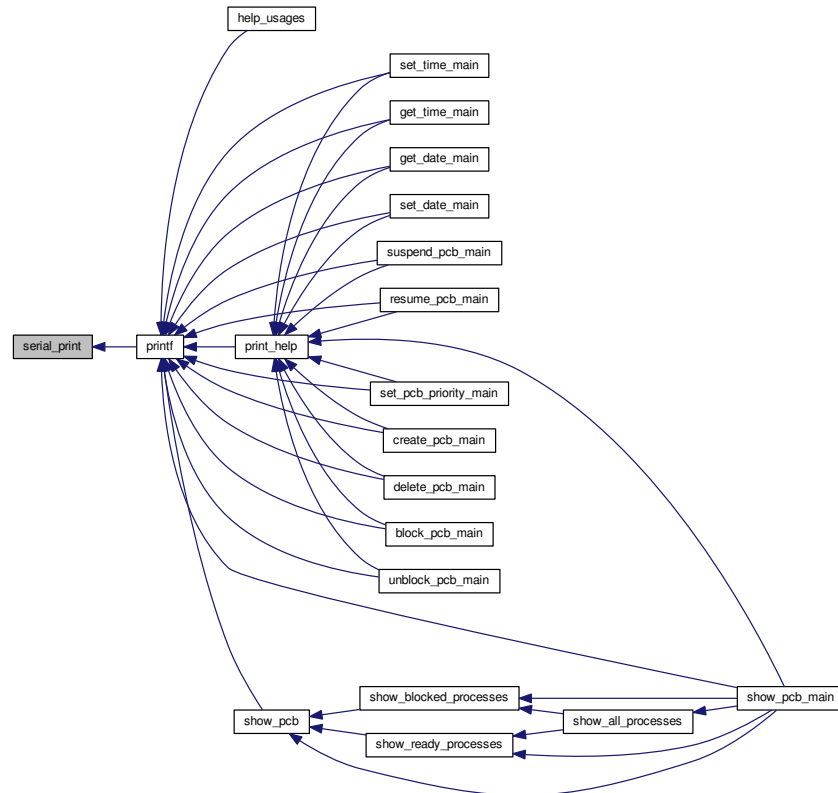
5.2.3 Function Documentation

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

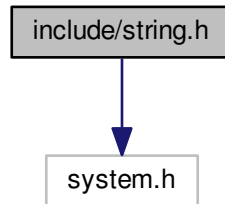
5.2.3.6 int `set_serial_out` (int *device*)

5.3 include/string.h File Reference

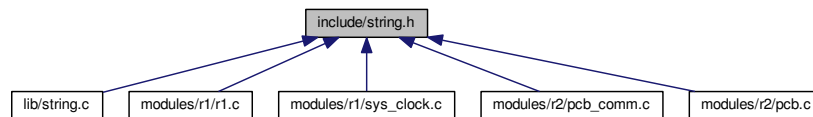
Many usefull functions that used for handling string.

```
#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

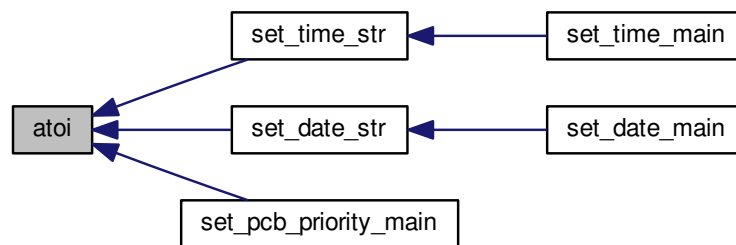
5.3.2 Function Documentation

5.3.2.1 int atoi (const char * s)

Here is the call graph for this function:

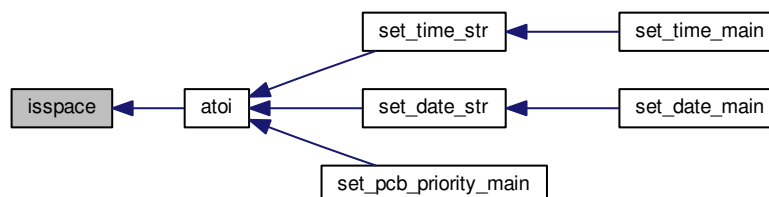


Here is the caller graph for this function:



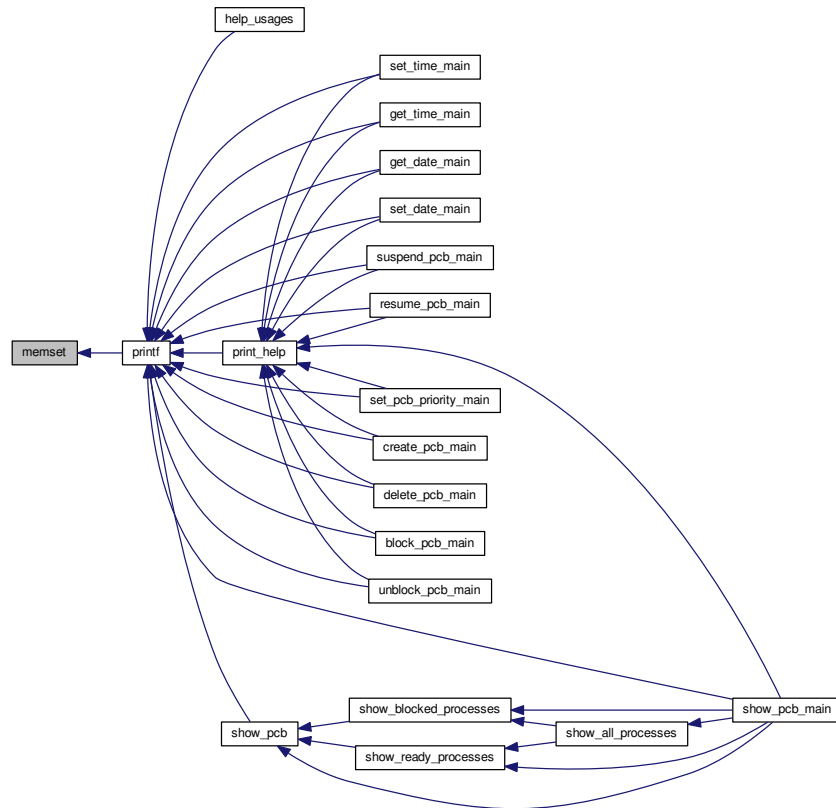
5.3.2.2 int isspace (const char * c)

Here is the caller graph for this function:



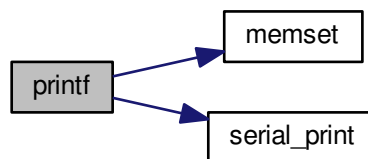
5.3.2.3 void* memset (void * s, int c, size_t n)

Here is the caller graph for this function:

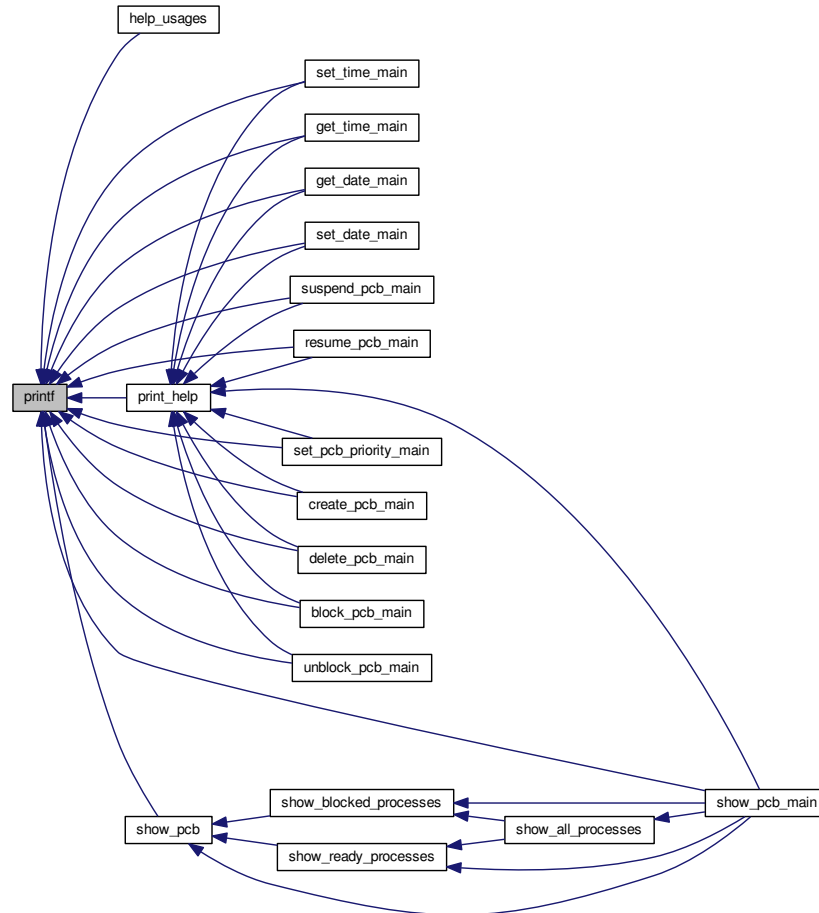


5.3.2.4 int printf (const char * format, ...)

Here is the call graph for this function:



Here is the caller graph for this function:

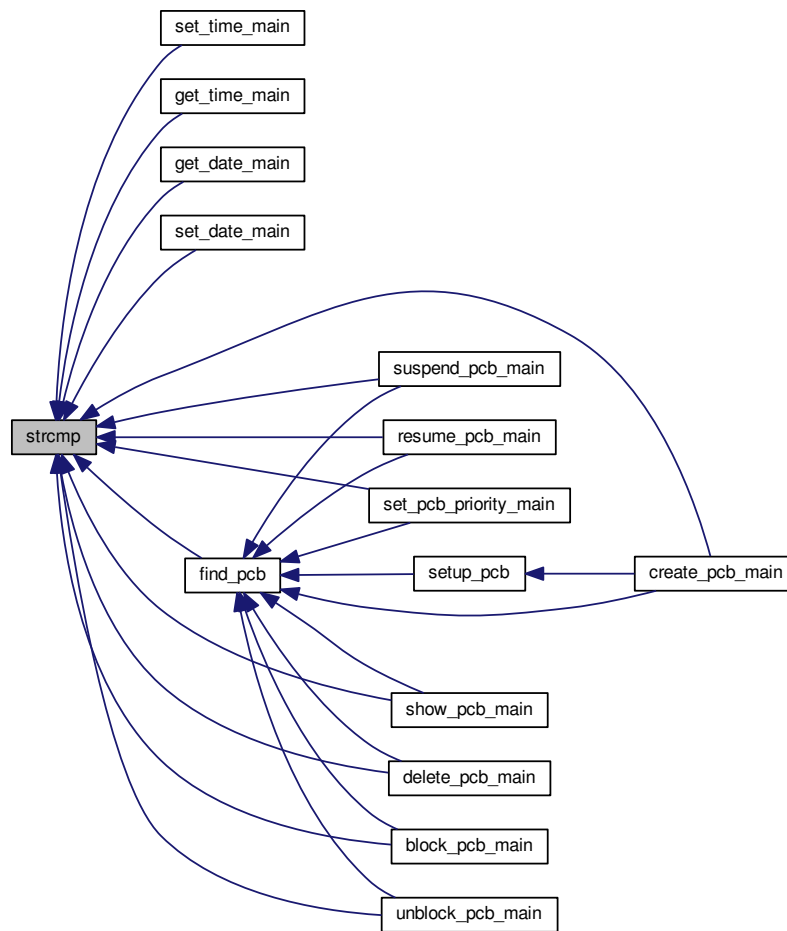


5.3.2.5 `int sprintf (char * str, const char * format, ...)`

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

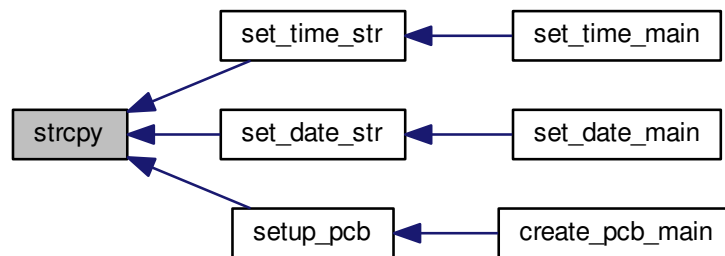
5.3.2.7 int strcmp (const char * s1, const char * s2)

Here is the caller graph for this function:



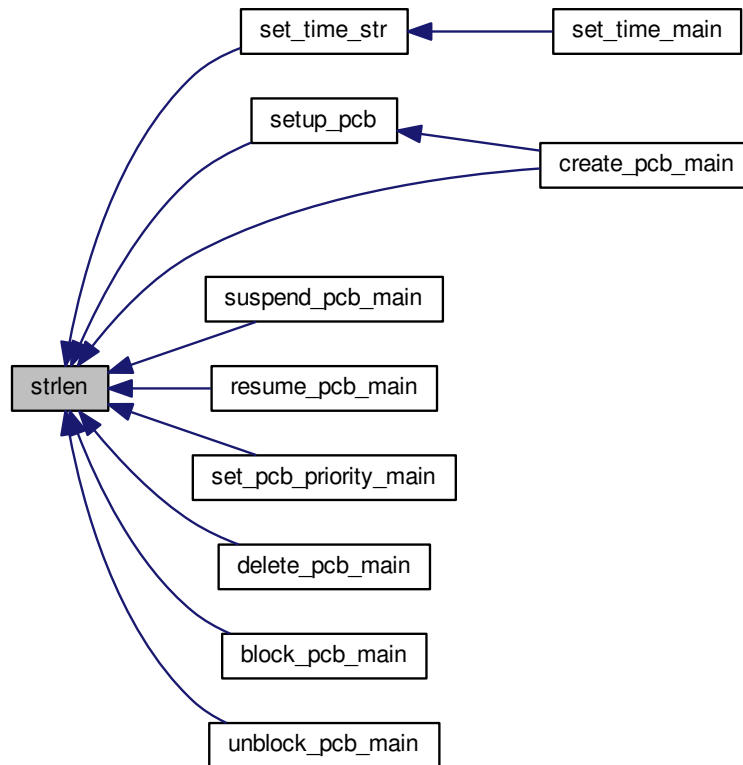
5.3.2.8 char* strcpy (char * s1, const char * s2)

Here is the caller graph for this function:



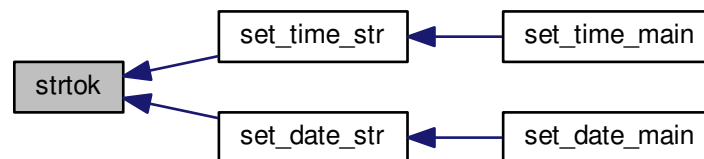
5.3.2.9 int strlen (const char * s)

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:

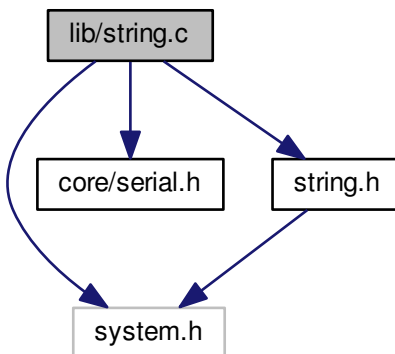


5.4 lib/string.c File Reference

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

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

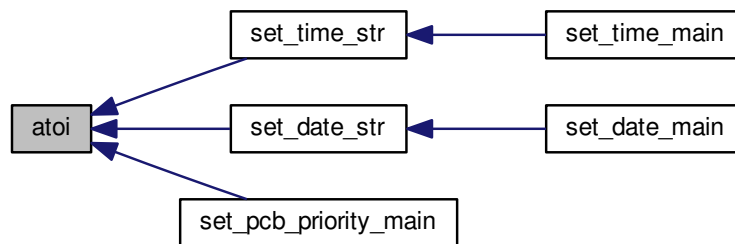
5.4.2 Function Documentation

5.4.2.1 `int atoi (const char * s)`

Here is the call graph for this function:

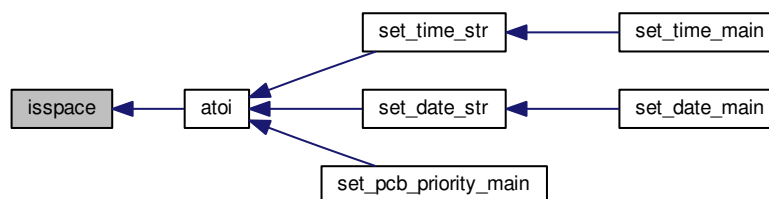


Here is the caller graph for this function:



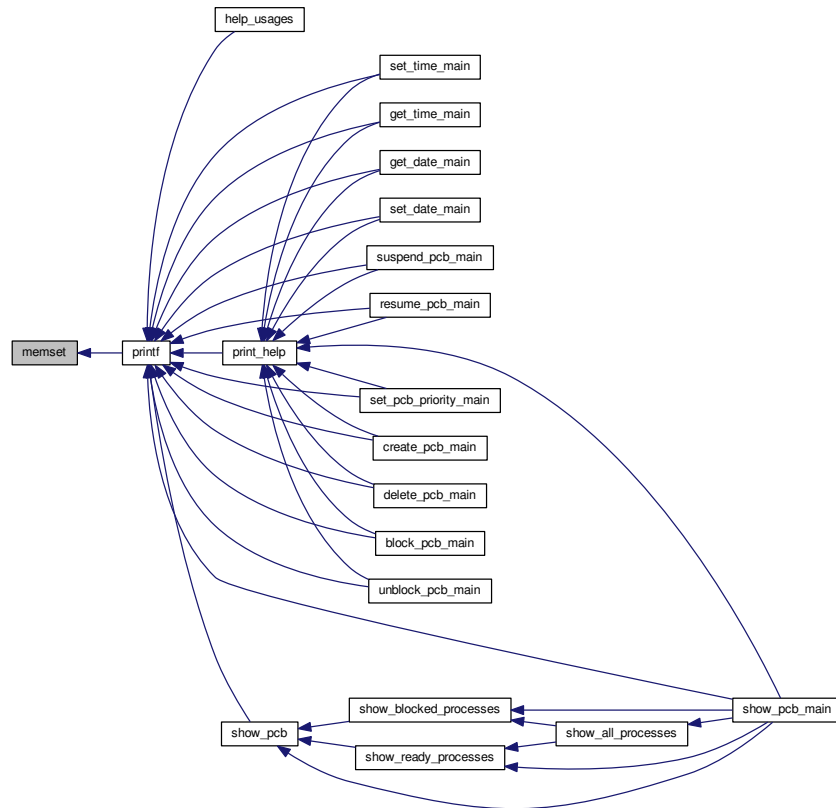
5.4.2.2 `int isspace (const char * c)`

Here is the caller graph for this function:



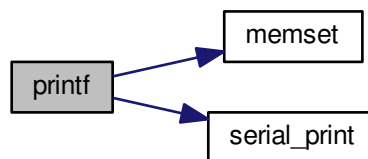
5.4.2.3 void* memset (void * s, int c, size_t n)

Here is the caller graph for this function:

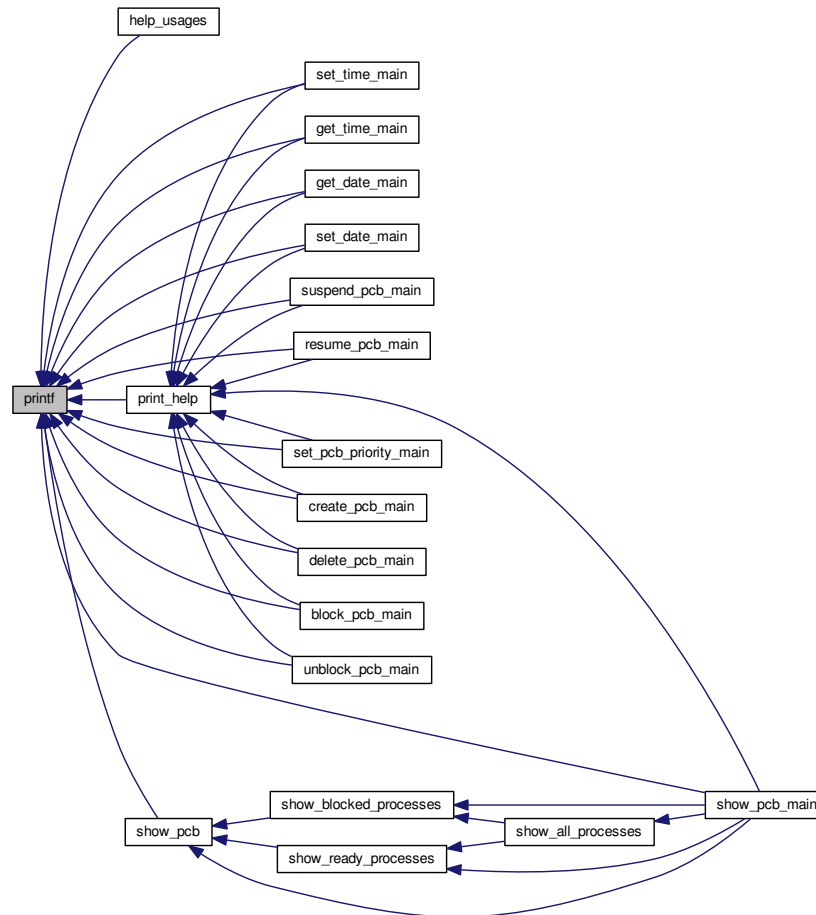


5.4.2.4 int printf (const char * format, ...)

Here is the call graph for this function:



Here is the caller graph for this function:

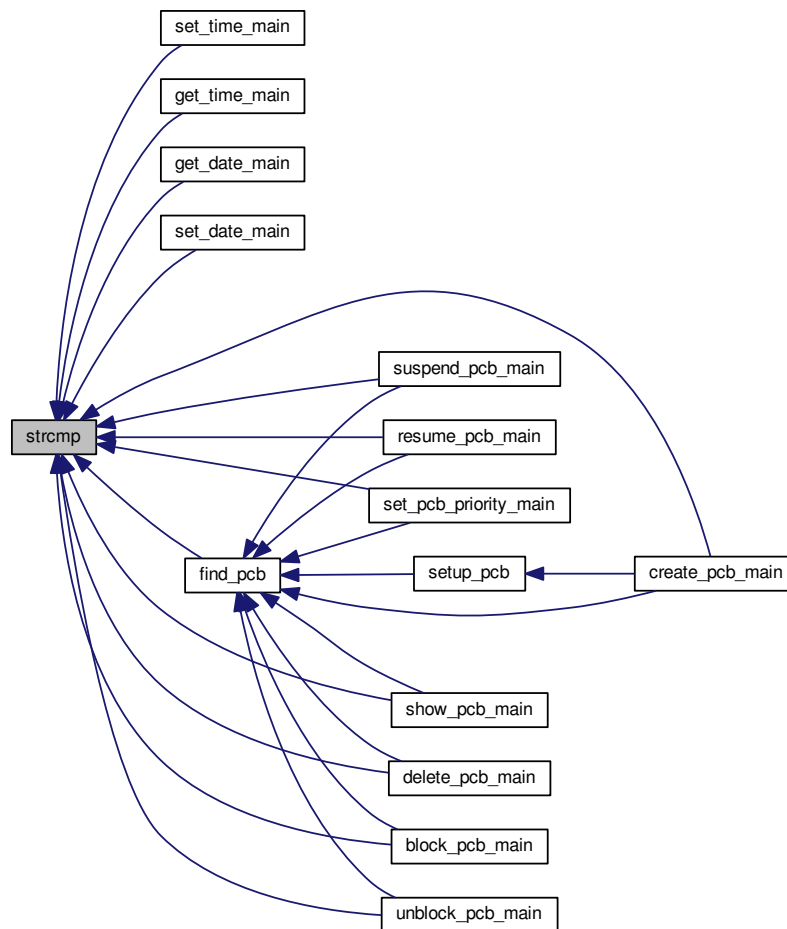


5.4.2.5 `int sprintf (char * str, const char * format, ...)`

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

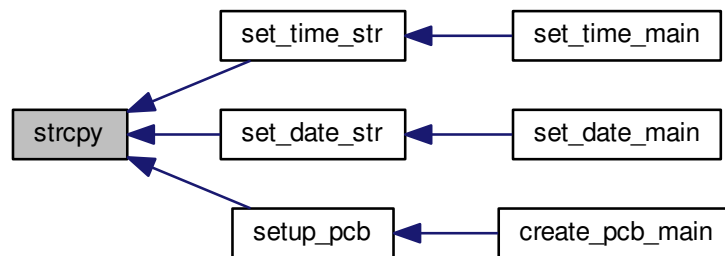
5.4.2.7 int strcmp (const char * s1, const char * s2)

Here is the caller graph for this function:



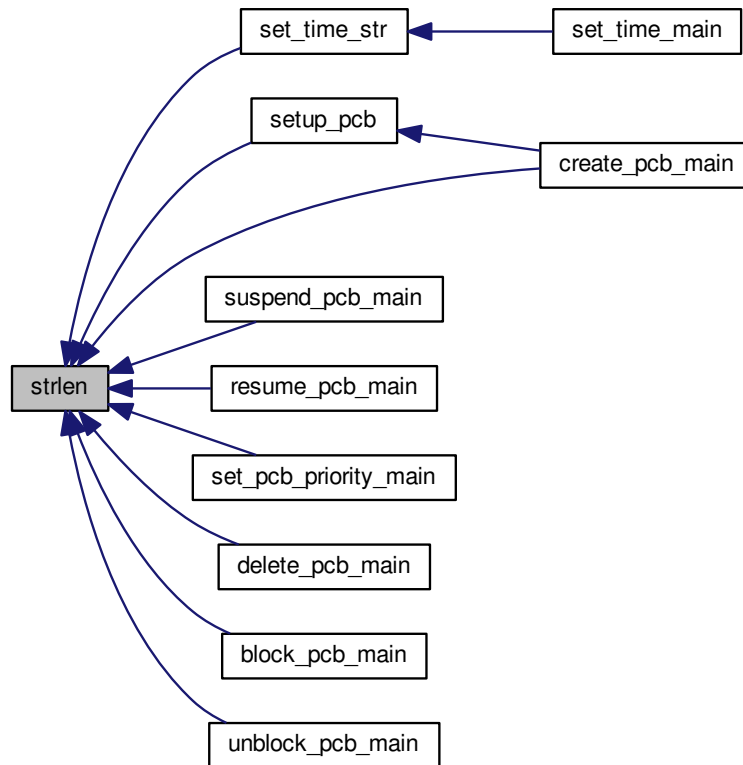
5.4.2.8 char* strcpy (char * s1, const char * s2)

Here is the caller graph for this function:



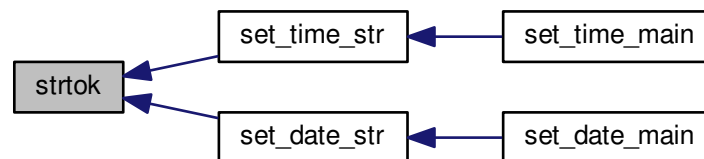
5.4.2.9 int strlen (const char * s)

Here is the caller graph for this function:



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

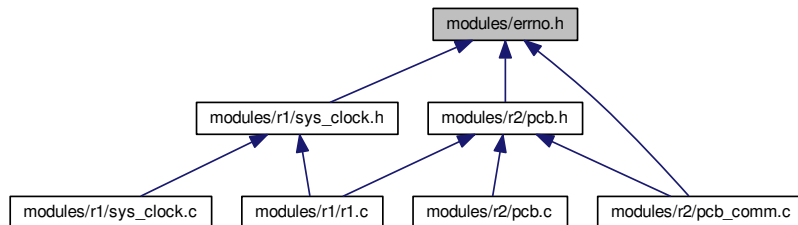
Here is the caller graph for this function:



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
- #define [E_PROGERR](#) 99

A NULL Pointer Error.

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_FREEMEM 4

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

5.5.2.2 #define E_INVPARA 1

5.5.2.3 #define E_INVSTRF 2

5.5.2.4 #define E_INVUSRI 3

5.5.2.5 #define E_NOERROR 0

5.5.2.6 #define E_NULL_PTR 5

A NULL Pointer Error.

5.5.2.7 #define E_PROGERR 99

5.5.3 Typedef Documentation

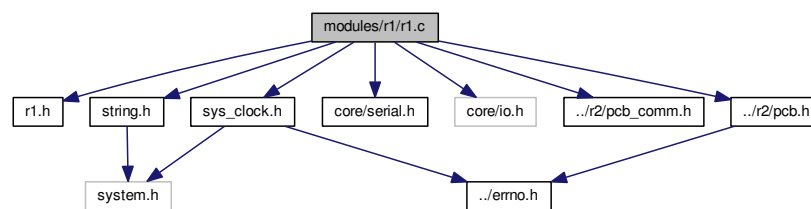
5.5.3.1 typedef unsigned int error_t

5.6 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 dependency graph for r1.c:



Data Structures

- struct `function_name`

A structure to represent each function.

Macros

- #define `USER_INPUT_BUFFER_SIZE` 1000
- #define `MAX_ARGC` 50
- #define `MOD_VERSION` "R2"
- #define `COMPLETION` "02/26/2016"

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

- int [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__\(\(packed\)\)](#)

5.6.1 Detailed Description

The commandhandler and functions associations for Module R1.

Author

Thunder Krakens

Date

February 2nd, 2016

Version

R1

5.6.2 Macro Definition Documentation

5.6.2.1 `#define COMPLETION "02/26/2016"`

5.6.2.2 `#define MAX_ARGC 50`

5.6.2.3 `#define MOD_VERSION "R2"`

5.6.2.4 `#define USER_INPUT_BUFFER_SIZE 1000`

5.6.3 Enumeration Type Documentation

5.6.3.1 enum CommandPaserStat

5.6.4 Function Documentation

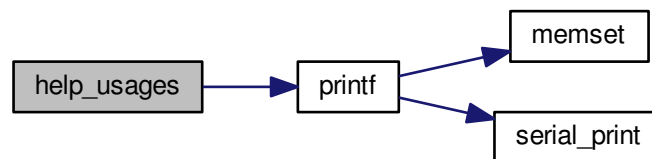
5.6.4.1 enum CommandPaserStat __attribute__((packed))

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

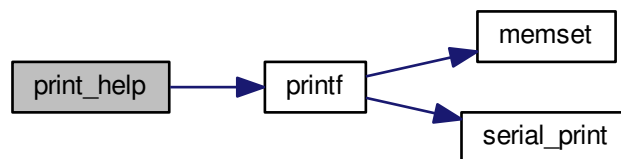
5.6.4.3 int commhand ()

5.6.4.4 int help_usages (enum comm_type *type*)

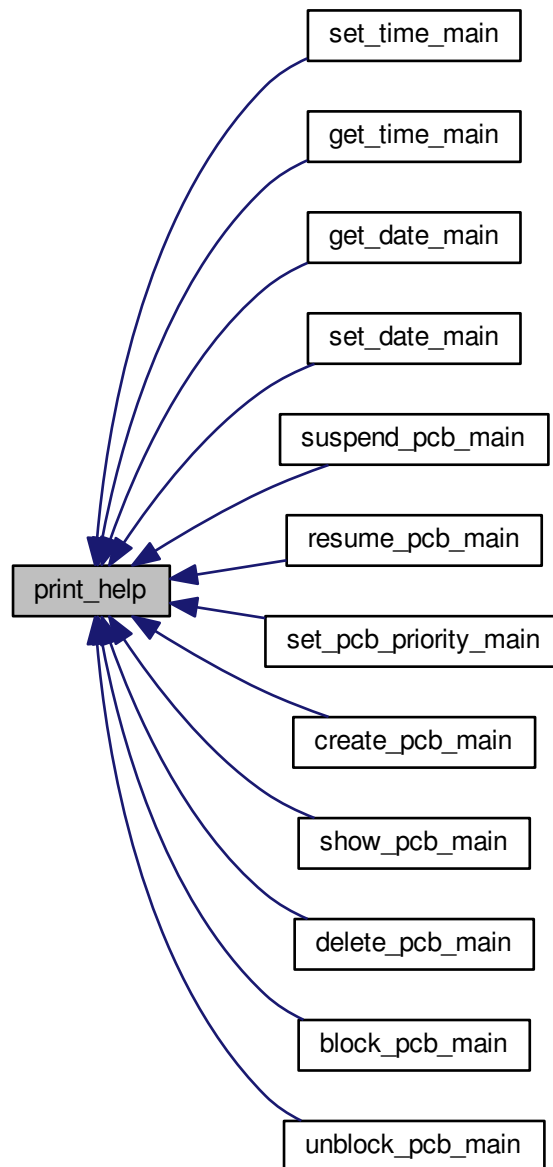
Here is the call graph for this function:

5.6.4.5 void print_help (const int *function_index*)

Here is the call graph for this function:



Here is the caller graph for this function:



5.6.5 Variable Documentation

5.6.5.1 DoubleQuoteWriting

5.6.5.2 NormalWriting

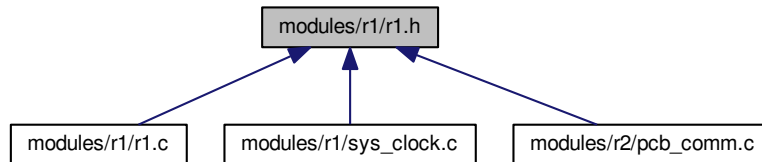
5.6.5.3 NotWriting

5.6.5.4 SingleQuoteWriting

5.7 modules/r1/r1.h File Reference

The commandhandler and functions associations for Module R1.

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



Macros

- #define [HELP](#) 0
- #define [VERSION](#) 1
- #define [GETTIME](#) 2
- #define [SETTIME](#) 3
- #define [GETDATE](#) 4
- #define [SETDATE](#) 5
- #define [SHUTDOWN](#) 6
- #define [CREATEPCB](#) 7
- #define [SHOWPCB](#) 8
- #define [SETPCBPRIO](#) 9
- #define [DELPcb](#) 10
- #define [BLOCKPCB](#) 11
- #define [UNBLKPCB](#) 12
- #define [RESUMEPCB](#) 13
- #define [SUSPDPCB](#) 14
- #define [NUM_OF_FUNCTIONS](#) 15

Enumerations

- enum [comm_type](#)

Functions

- enum [comm_type](#) [__attribute__\(\(packed\)\)](#)

commhand

Accepts and handles commands from the user.

Returns

0

- int [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](#)

5.7.1 Detailed Description

The commandhandler and functions associations for Module R1.

Author

Thunder Krakens

Date

February 2nd, 2016

Version

R1

5.7.2 Macro Definition Documentation

5.7.2.1 `#define BLOCKPCB 11`

5.7.2.2 `#define CREATEPCB 7`

5.7.2.3 `#define DELPCB 10`

5.7.2.4 `#define GETDATE 4`

5.7.2.5 `#define GETTIME 2`

5.7.2.6 `#define HELP 0`

5.7.2.7 `#define NUM_OF_FUNCTIONS 15`

5.7.2.8 `#define RESUMEPCB 13`

5.7.2.9 `#define SETDATE 5`

5.7.2.10 `#define SETPCBPRIOR 9`

5.7.2.11 `#define SETTIME 3`

5.7.2.12 `#define SHOWPCB 8`

5.7.2.13 `#define SHUTDOWN 6`

5.7.2.14 `#define SUSDPCH 14`

5.7.2.15 `#define UNBLKPCB 12`

5.7.2.16 `#define VERSION 1`

5.7.3 Enumeration Type Documentation

5.7.3.1 `enum comm_type`

5.7.4 Function Documentation

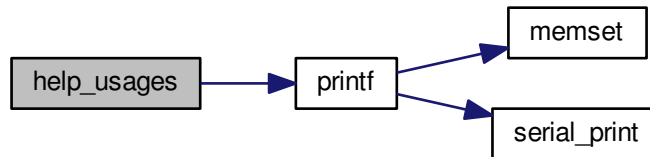
5.7.4.1 `enum comm_type __attribute__((packed))`

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

5.7.4.3 `int commhand ()`

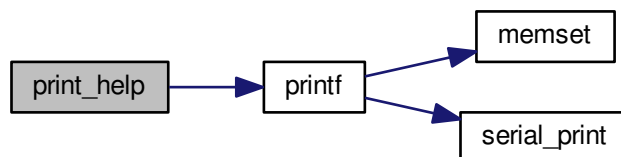
5.7.4.4 `int help_usages (enum comm_type type)`

Here is the call graph for this function:

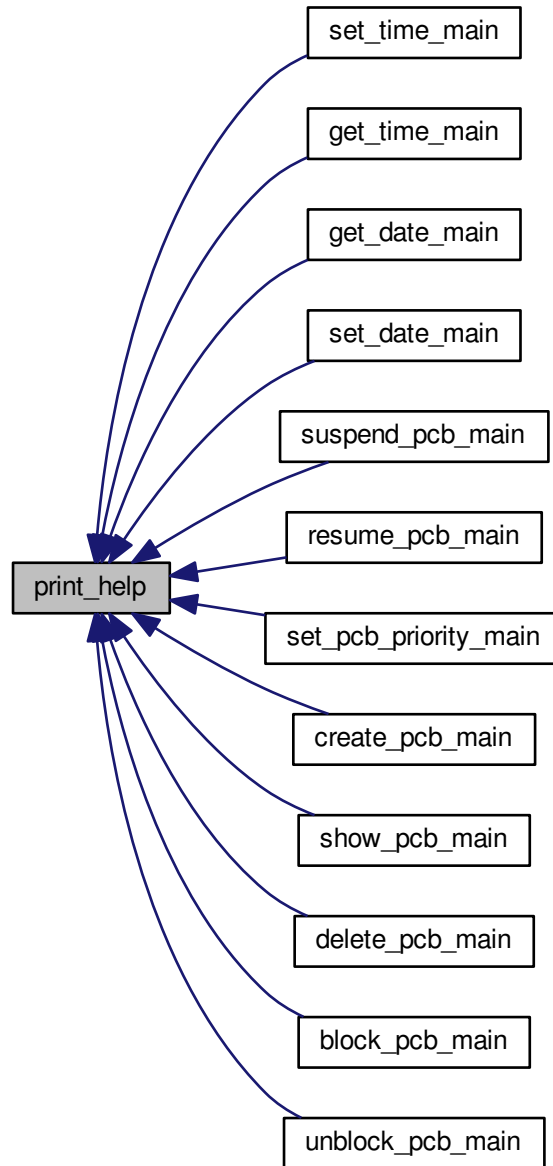


5.7.4.5 `void print_help (const int function_index)`

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.5 Variable Documentation

5.7.5.1 help

5.7.5.2 mpx

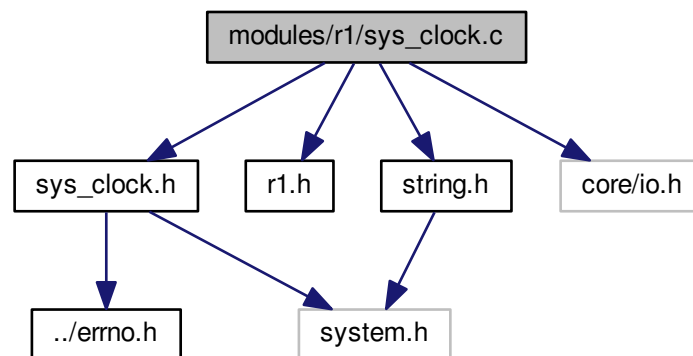
5.7.5.3 pcb

5.8 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

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)

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	<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_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)

5.8.1 Detailed Description

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

Author

Thunder Krakens

Date

February 2nd, 2016

Version

R1

5.8.2 Macro Definition Documentation

5.8.2.1 `#define RTC_INDEX_DAY_MONTH 0x07`

5.8.2.2 `#define RTC_INDEX_DAY_WEEK 0x06`

5.8.2.3 `#define RTC_INDEX_HOUR 0x04`

5.8.2.4 `#define RTC_INDEX_HOUR_ALARM 0x05`

5.8.2.5 `#define RTC_INDEX_MINUTE 0x02`

5.8.2.6 `#define RTC_INDEX_MINUTE_ALARM 0x03`

5.8.2.7 `#define RTC_INDEX_MONTH 0x08`

5.8.2.8 `#define RTC_INDEX_SECOND 0x00`

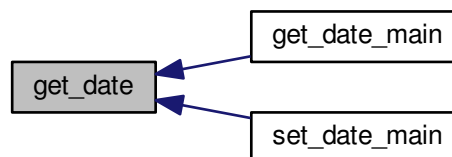
5.8.2.9 `#define RTC_INDEX_SECOND_ALARM 0x01`

5.8.2.10 `#define RTC_INDEX_YEAR 0x09`

5.8.3 Function Documentation

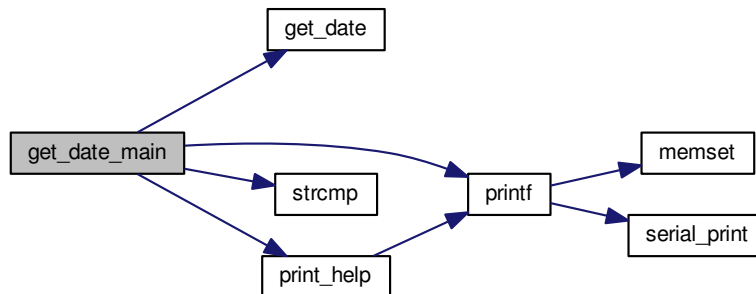
5.8.3.1 `void get_date (date_time * dateTimeValues)`

Here is the caller graph for this function:



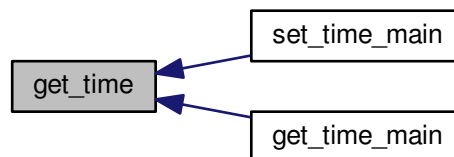
5.8.3.2 int get_date_main (int argc, char ** argv)

Here is the call graph for this function:



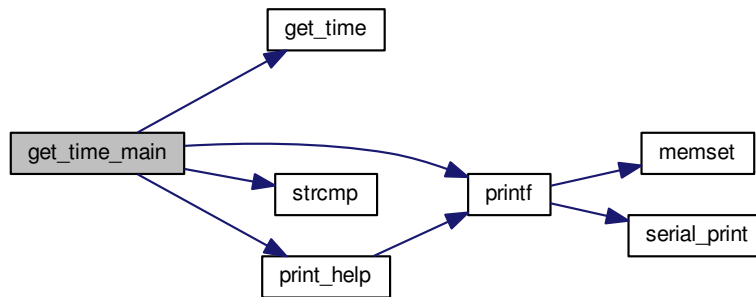
5.8.3.3 void get_time (date_time * dateTimeValues)

Here is the caller graph for this function:



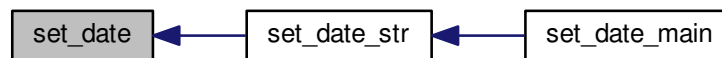
5.8.3.4 int get_time_main (int argc, char ** argv)

Here is the call graph for this function:



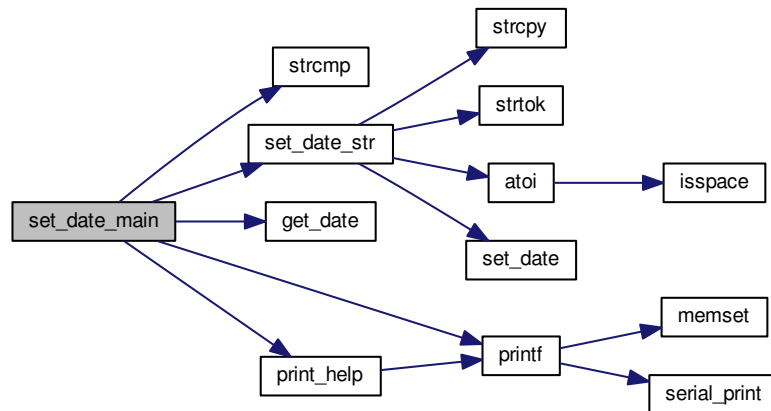
5.8.3.5 error_t set_date (const date_time * dateTimeValues)

Here is the caller graph for this function:



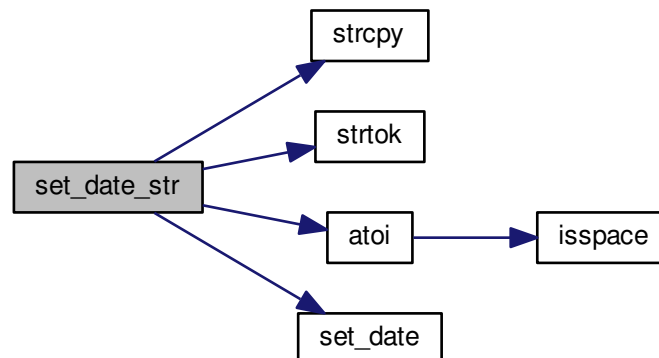
5.8.3.6 int set_date_main (int argc, char ** argv)

Here is the call graph for this function:



5.8.3.7 int set_date_str (const char * str)

Here is the call graph for this function:

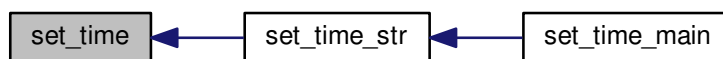


Here is the caller graph for this function:



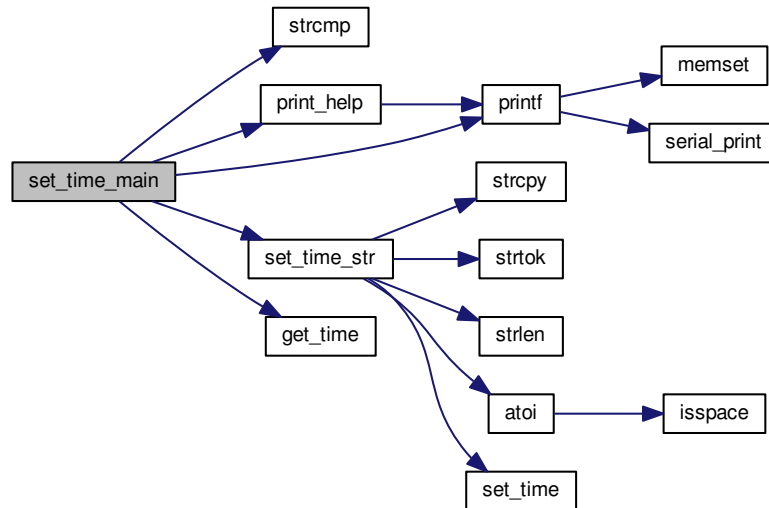
5.8.3.8 `error_t set_time (const date_time * dateTimeValues)`

Here is the caller graph for this function:



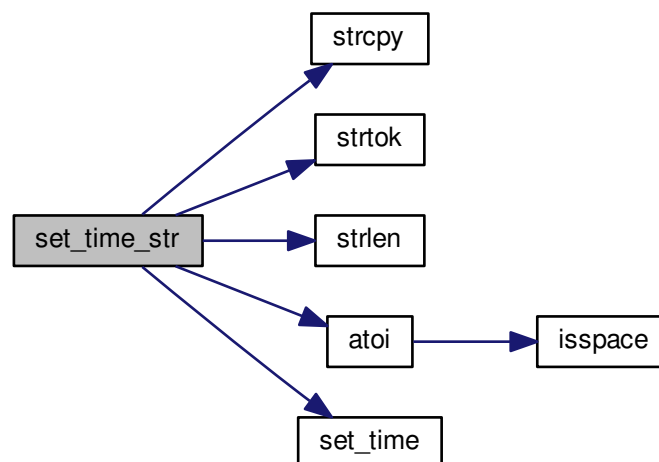
5.8.3.9 `int set_time_main (int argc, char ** argv)`

Here is the call graph for this function:



5.8.3.10 `error_t set_time_str (const char * timeStr)`

Here is the call graph for this function:



Here is the caller graph for this function:

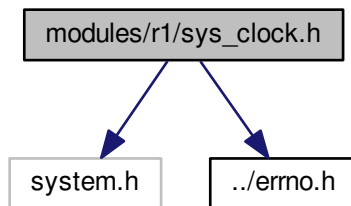


5.9 modules/r1/sys_clock.h File Reference

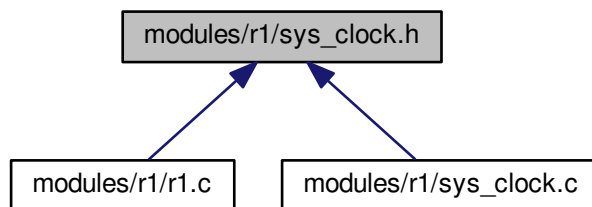
The main file that manipulates and controls the system's clock.

```
#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)

5.9.1 Detailed Description

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

Author

Thunder Krakens

Date

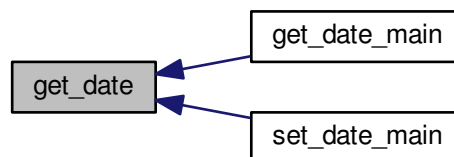
February 2nd, 2016

Version

R1

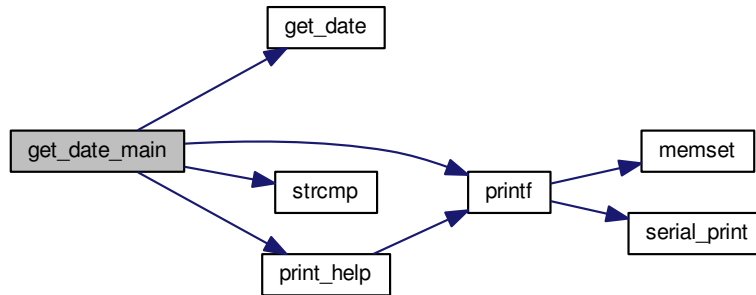
5.9.2 Function Documentation**5.9.2.1 void get_date (date_time * dateTimeValues)**

Here is the caller graph for this function:



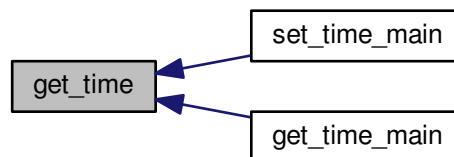
5.9.2.2 int get_date_main (int argc, char ** argv)

Here is the call graph for this function:



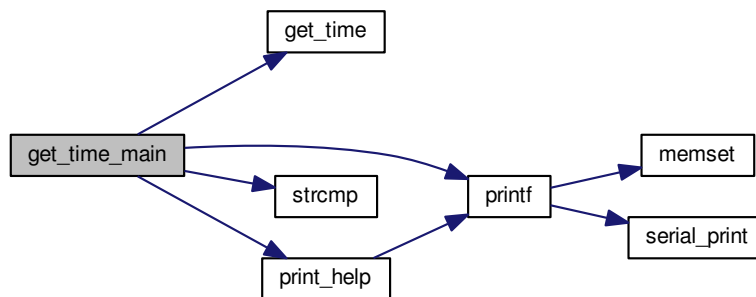
5.9.2.3 void get_time (date_time * dateTimeValues)

Here is the caller graph for this function:



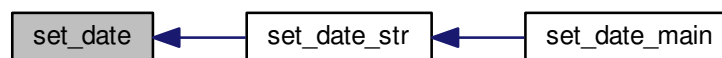
5.9.2.4 `int get_time_main (int argc, char ** argv)`

Here is the call graph for this function:



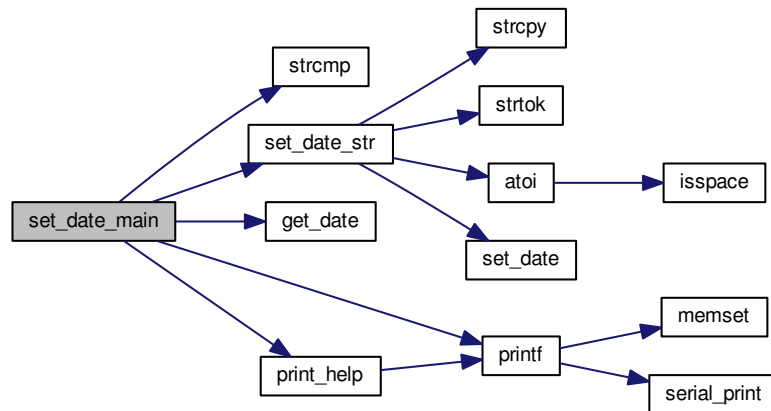
5.9.2.5 `error_t set_date (const date_time * dateTimeValues)`

Here is the caller graph for this function:



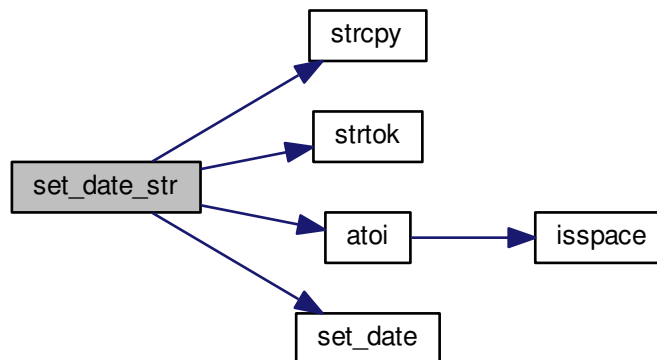
5.9.2.6 int set_date_main (int argc, char ** argv)

Here is the call graph for this function:



5.9.2.7 int set_date_str (const char * str)

Here is the call graph for this function:

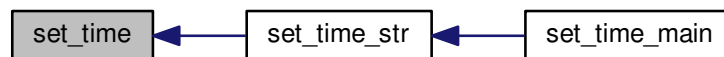


Here is the caller graph for this function:



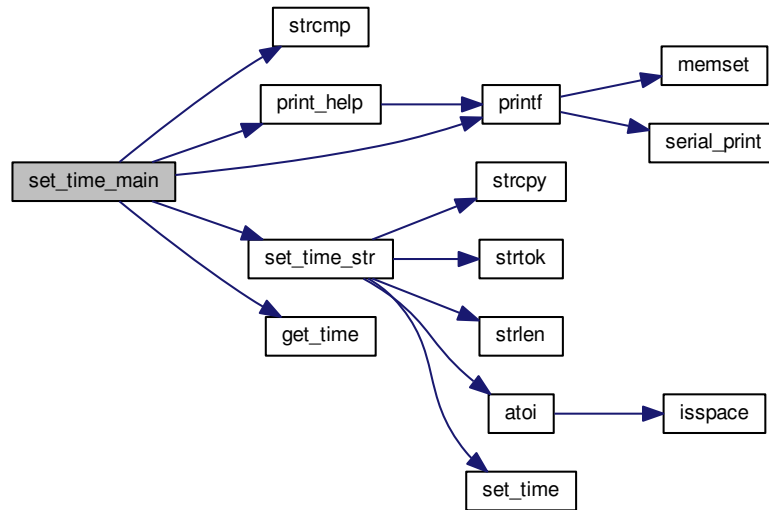
5.9.2.8 `error_t set_time (const date_time * dateTimeValues)`

Here is the caller graph for this function:



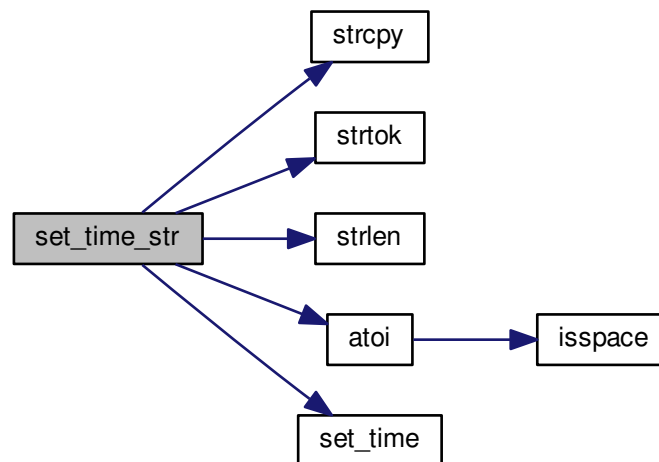
5.9.2.9 `int set_time_main (int argc, char ** argv)`

Here is the call graph for this function:



5.9.2.10 `error_t set_time_str (const char * timeStr)`

Here is the call graph for this function:



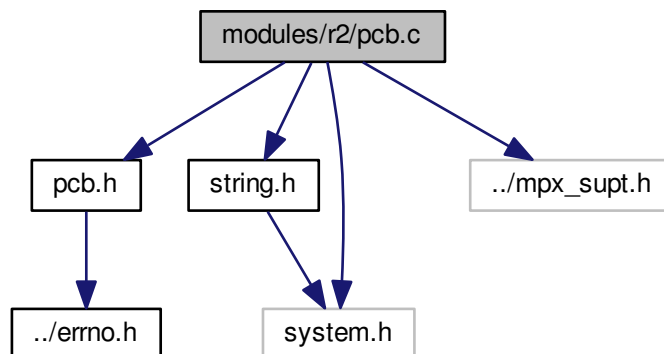
Here is the caller graph for this function:



5.10 modules/r2/pcb.c File Reference

The Process Control Block.

```
#include "pcb.h"
#include <string.h>
#include "../mpx_supt.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

pcb_ptr	<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

pcb_ptr	<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)

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__](#)

5.10.1 Detailed Description

The Process Control Block.

Author

Thunder Krakens

Date

February 7th, 2016

Version

R2

5.10.2 Enumeration Type Documentation

5.10.2.1 enum process_state

PCB process states/statuses.

5.10.2.2 enum process_suspended

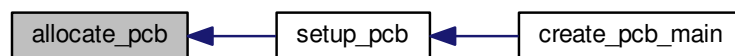
PCB process suspended or not suspended status.

5.10.3 Function Documentation

5.10.3.1 enum process_state __attribute__((packed))

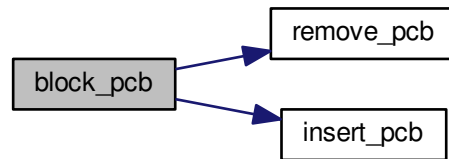
5.10.3.2 struct pcb_struct* allocate_pcb ()

Here is the caller graph for this function:



5.10.3.3 `error_t block_pcb (struct pcb_struct * pcb_ptr)`

Here is the call graph for this function:



Here is the caller graph for this function:

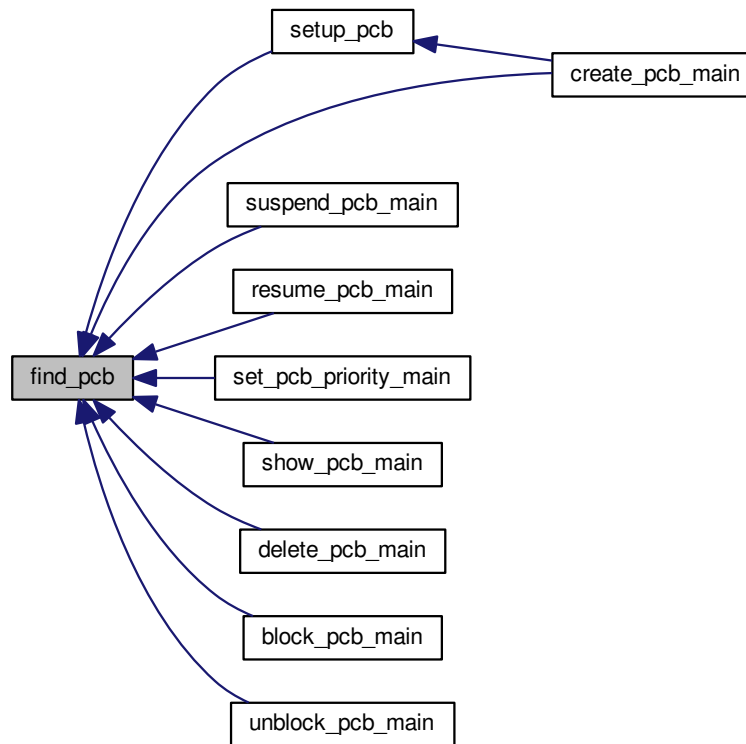


5.10.3.4 `struct pcb_struct* find_pcb (const char * pName)`

Here is the call graph for this function:



Here is the caller graph for this function:



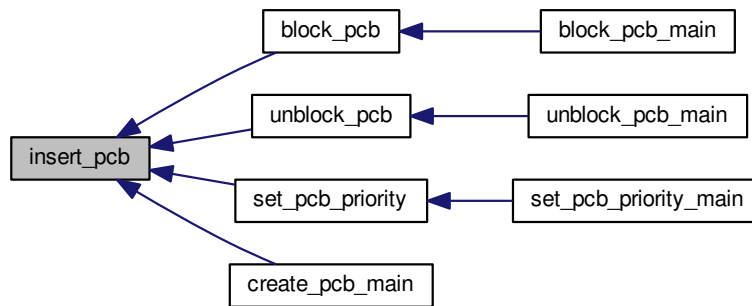
5.10.3.5 `error_t free_pcb (struct pcb_struct * pcb_ptr)`

Here is the caller graph for this function:



5.10.3.6 `error_t insert_pcb (struct pcb_struct * pcb_ptr)`

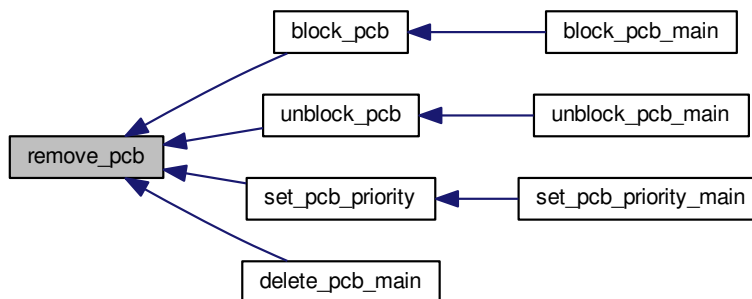
Here is the caller graph for this function:



5.10.3.7 `void pcb_init ()`

5.10.3.8 `error_t remove_pcb (struct pcb_struct * pcb_ptr)`

Here is the caller graph for this function:



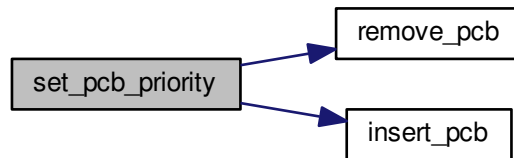
5.10.3.9 `error_t resume_pcb (struct pcb_struct * pcb_ptr)`

Here is the caller graph for this function:



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

Here is the call graph for this function:

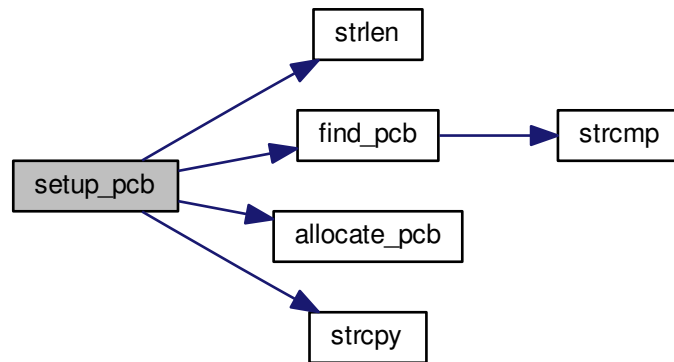


Here is the caller graph for this function:



5.10.3.11 `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:

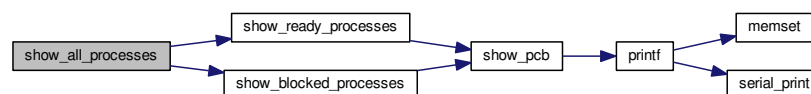


Here is the caller graph for this function:



5.10.3.12 `void show_all_processes ()`

Here is the call graph for this function:

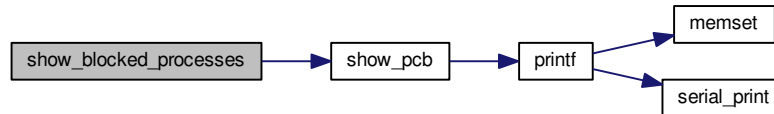


Here is the caller graph for this function:

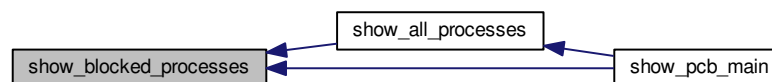


5.10.3.13 void show_blocked_processes ()

Here is the call graph for this function:

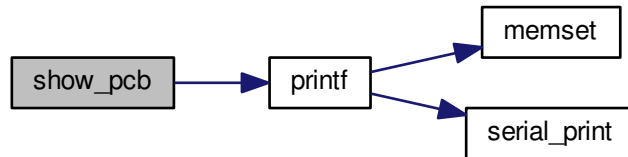


Here is the caller graph for this function:

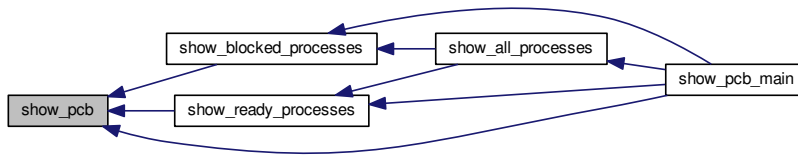


5.10.3.14 `error_t show_pcb (struct pcb_struct * pcb_ptr)`

Here is the call graph for this function:

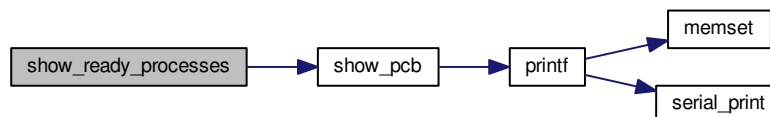


Here is the caller graph for this function:

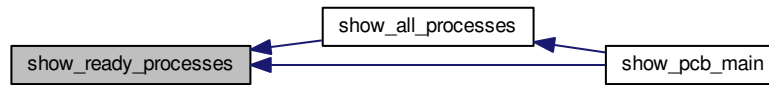


5.10.3.15 `void show_ready_processes ()`

Here is the call graph for this function:



Here is the caller graph for this function:



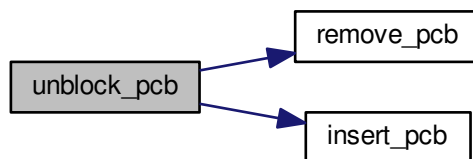
5.10.3.16 `error_t suspend_pcb (struct pcb_struct * pcb_ptr)`

Here is the caller graph for this function:



5.10.3.17 `error_t unblock_pcb (struct pcb_struct * pcb_ptr)`

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.4 Variable Documentation

5.10.4.1 enum process_suspended __attribute__

5.10.4.2 blocked

< PCB in the blocked state.

PCB in the blocked state.

5.10.4.3 false

< PCB process is not suspended.

PCB process is not suspended.

5.10.4.4 ready

PCB in the ready state.

5.10.4.5 running

PCB in the running state.

5.10.4.6 true

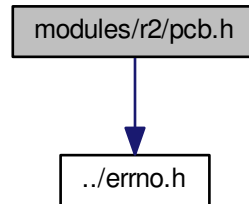
PCB process is suspended.

5.11 modules/r2/pcb.h File Reference

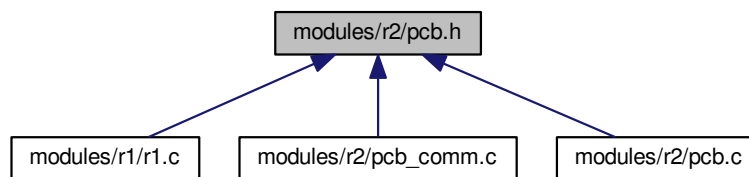
The Process Control Block.

```
#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)

Variables

- [pcb_class_app](#)
Process is an application process.
- [pcb_class_sys](#)
< Process is a system process.

5.11.1 Detailed Description

The Process Control Block.

Author

Thunder Krakens

Date

February 7th, 2016

Version

R2

5.11.2 Macro Definition Documentation

5.11.2.1 `#define SIZE_OF_PCB_NAME 10`

5.11.2.2 `#define SIZE_OF_STACK 1024`

5.11.3 Enumeration Type Documentation

5.11.3.1 `enum process_class`

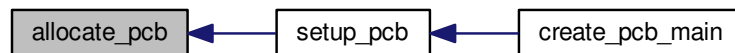
PCB process class types.

5.11.4 Function Documentation

5.11.4.1 `enum process_class __attribute__((packed))`

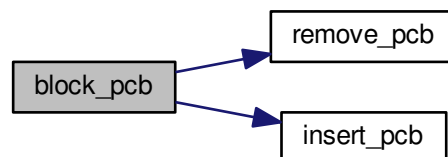
5.11.4.2 `struct pcb_struct* allocate_pcb ()`

Here is the caller graph for this function:



5.11.4.3 `error_t block_pcb (struct pcb_struct * pcb_ptr)`

Here is the call graph for this function:



Here is the caller graph for this function:

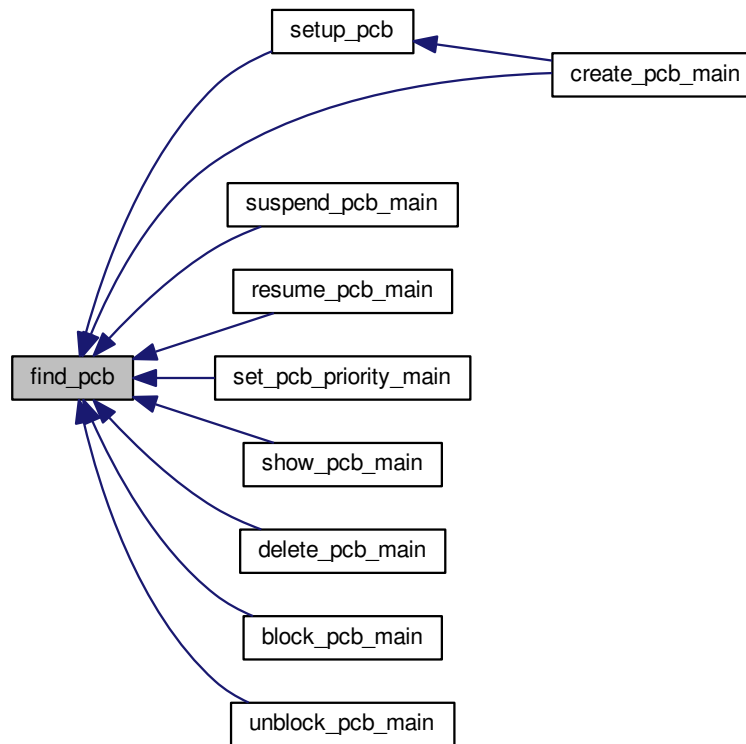


5.11.4.4 struct pcb_struct* find_pcb (const char * *pName*)

Here is the call graph for this function:



Here is the caller graph for this function:



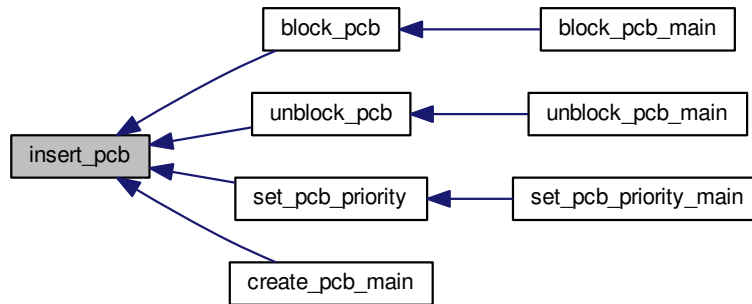
5.11.4.5 `error_t free_pcb (struct pcb_struct * pcb_ptr)`

Here is the caller graph for this function:



5.11.4.6 `error_t insert_pcb (struct pcb_struct * pcb_ptr)`

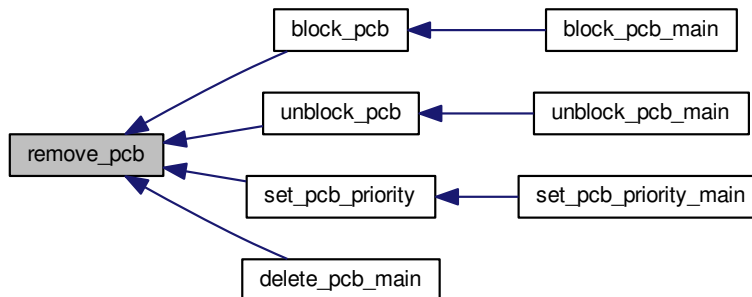
Here is the caller graph for this function:



5.11.4.7 `void pcb_init ()`

5.11.4.8 `error_t remove_pcb (struct pcb_struct * pcb_ptr)`

Here is the caller graph for this function:



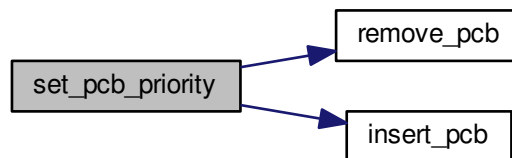
5.11.4.9 `error_t resume_pcb (struct pcb_struct * pcb_ptr)`

Here is the caller graph for this function:



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

Here is the call graph for this function:

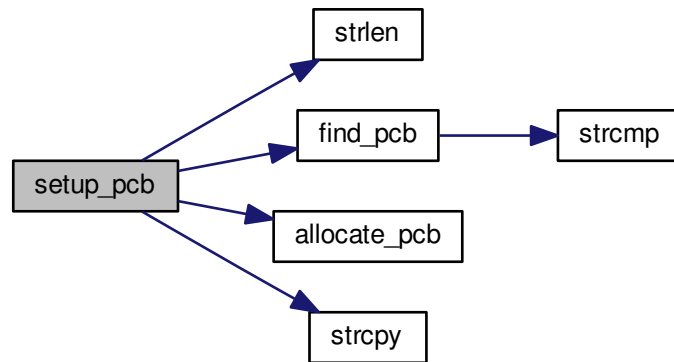


Here is the caller graph for this function:



5.11.4.11 `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:

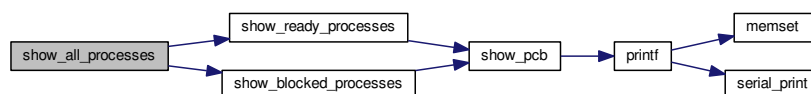


Here is the caller graph for this function:



5.11.4.12 `void show_all_processes ()`

Here is the call graph for this function:

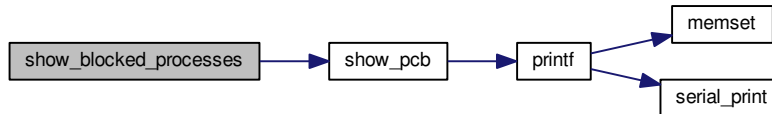


Here is the caller graph for this function:

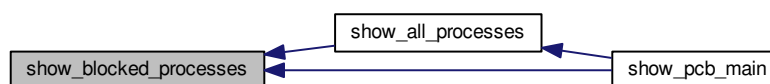


5.11.4.13 void show_blocked_processes ()

Here is the call graph for this function:

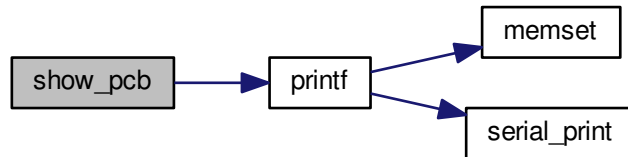


Here is the caller graph for this function:

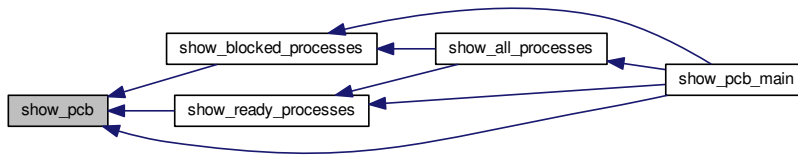


5.11.4.14 `error_t show_pcb (struct pcb_struct * pcb_ptr)`

Here is the call graph for this function:

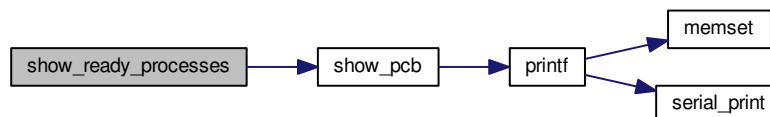


Here is the caller graph for this function:

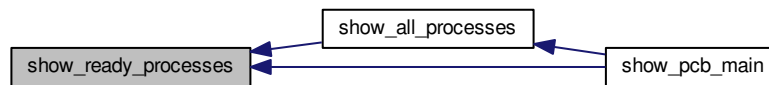


5.11.4.15 `void show_ready_processes ()`

Here is the call graph for this function:



Here is the caller graph for this function:



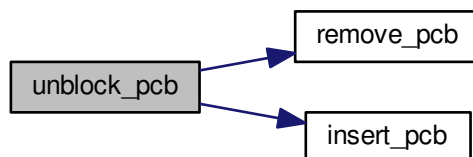
5.11.4.16 `error_t suspend_pcb (struct pcb_struct * pcb_ptr)`

Here is the caller graph for this function:



5.11.4.17 `error_t unblock_pcb (struct pcb_struct * pcb_ptr)`

Here is the call graph for this function:



Here is the caller graph for this function:



5.11.5 Variable Documentation

5.11.5.1 pcb_class_app

Process is an application process.

5.11.5.2 pcb_class_sys

< Process is a system process.

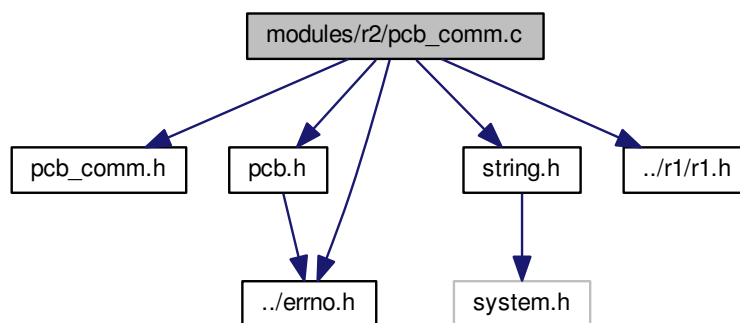
Process is a system process.

5.12 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)

create_pcb_main.

The main function for the "Create PCB".

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

Parameters

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

Returns

0

- int [create_pcb_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)

delete_pcb_main.

The main function for the "Delete PCB".

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

Parameters

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

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

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

Returns

0

- int [unblock_pcb_main](#) (int argc, char **argv)

5.12.1 Detailed Description

The main functions that manipulate the PCB.

Author

Thunder Krakens

Date

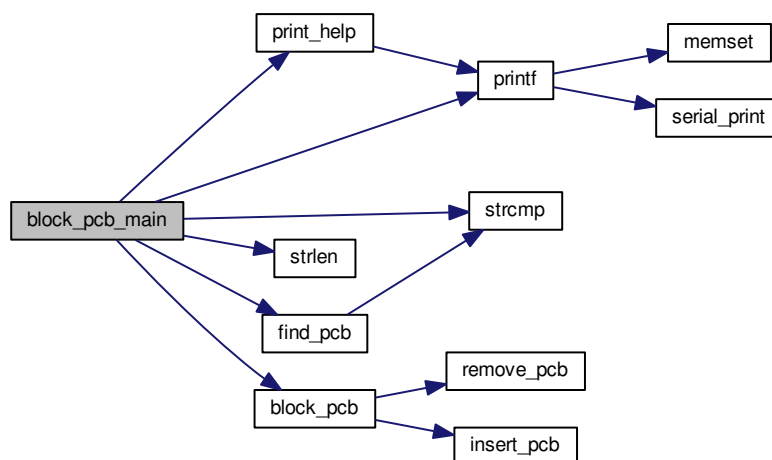
February 7th, 2016

Version

R2

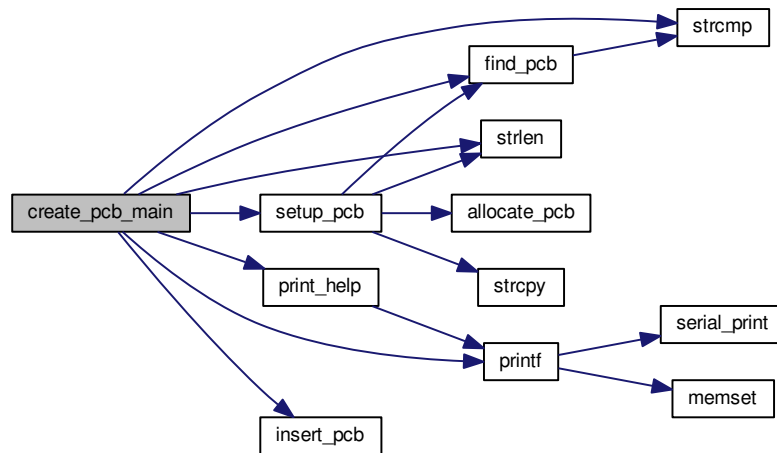
5.12.2 Function Documentation**5.12.2.1 int block_pcb_main (int argc, char ** argv)**

Here is the call graph for this function:



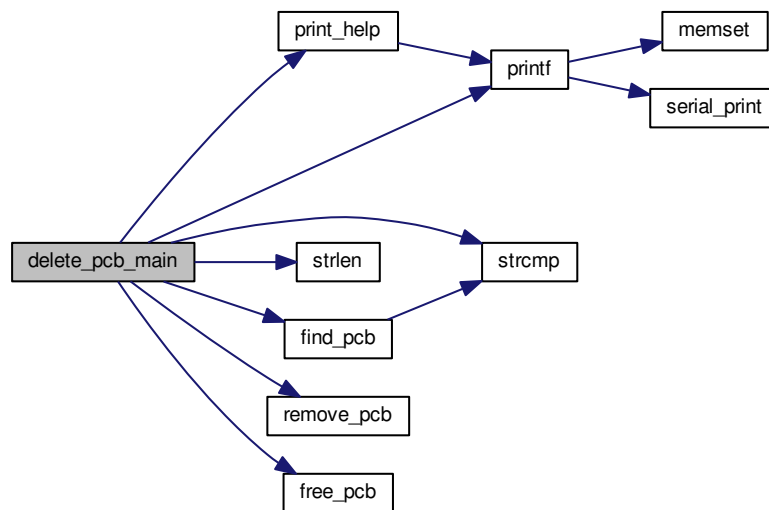
5.12.2.2 int create_pcb_main (int argc, char ** argv)

Here is the call graph for this function:



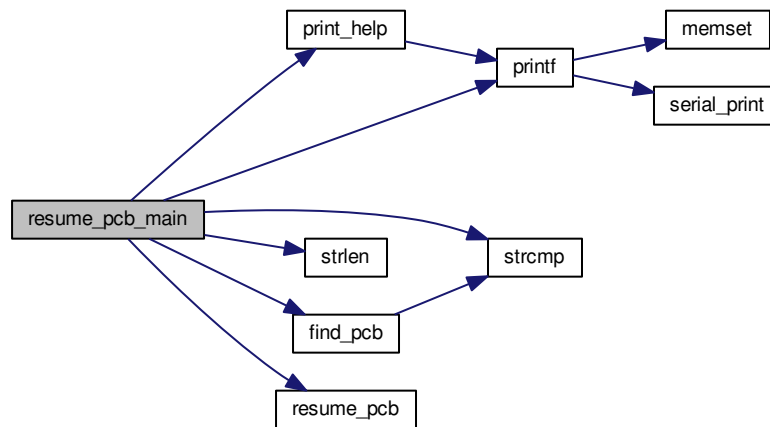
5.12.2.3 int delete_pcb_main (int argc, char ** argv)

Here is the call graph for this function:



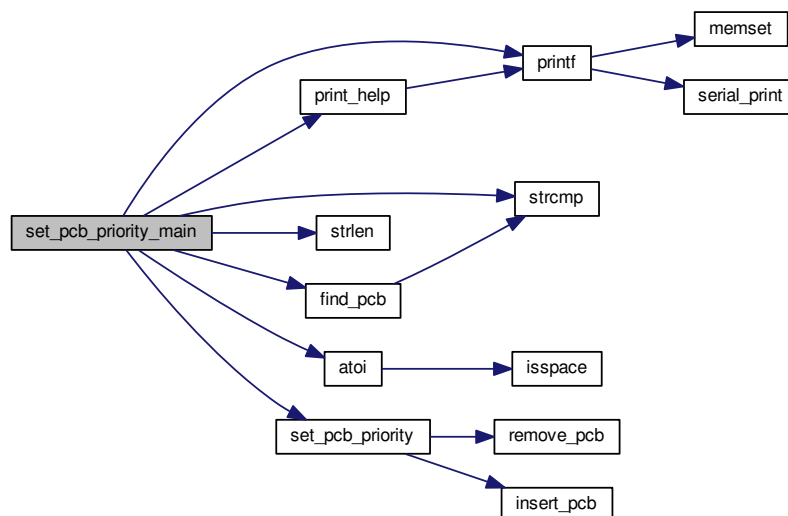
5.12.2.4 int resume_pcb_main (int argc, char ** argv)

Here is the call graph for this function:



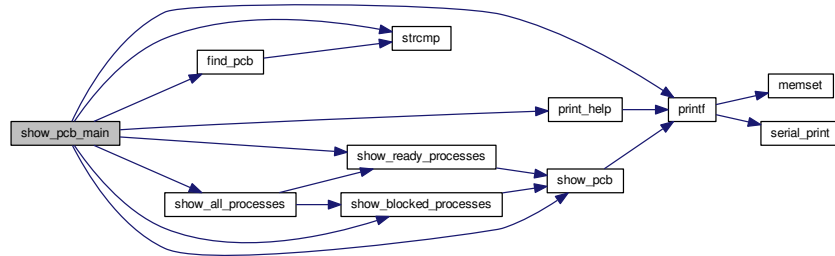
5.12.2.5 int set_pcb_priority_main (int argc, char ** argv)

Here is the call graph for this function:



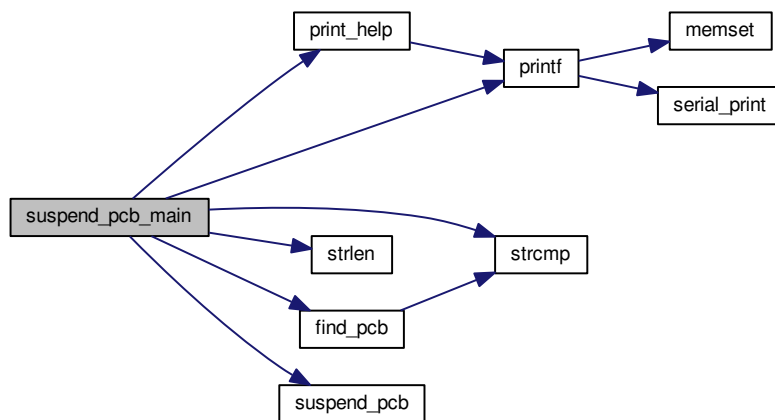
5.12.2.6 int show_pcb_main (int argc, char ** argv)

Here is the call graph for this function:



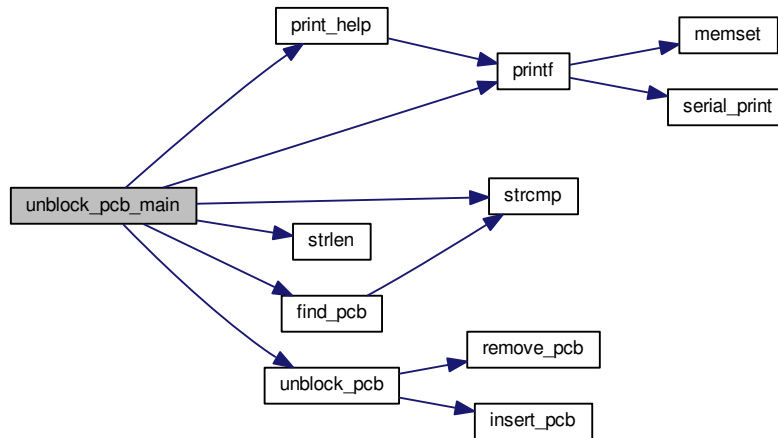
5.12.2.7 int suspend_pcb_main (int argc, char ** argv)

Here is the call graph for this function:



5.12.2.8 `int unblock_pcb_main (int argc, char ** argv)`

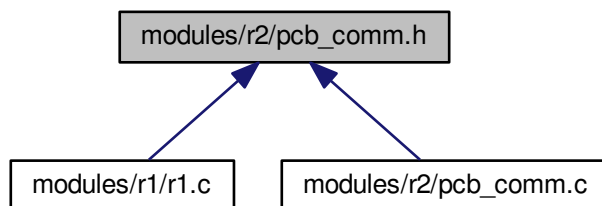
Here is the call graph for this function:



5.13 `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)

5.13.1 Detailed Description

The main functions that manipulate the PCB.

Author

Thunder Krakens

Date

February 7th, 2016

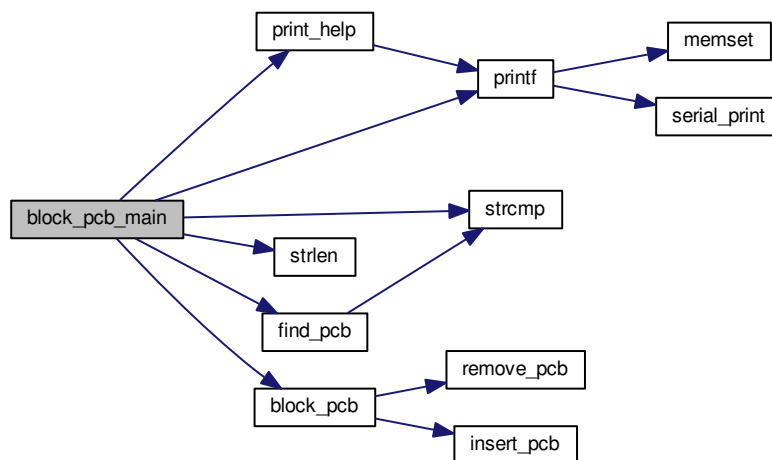
Version

R2

5.13.2 Function Documentation

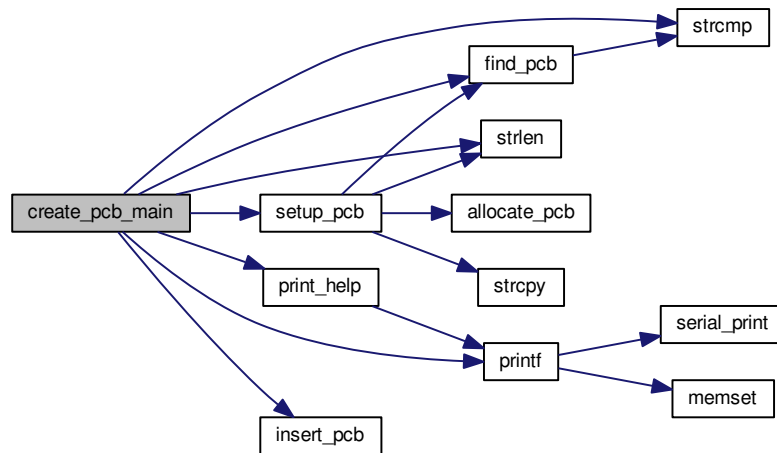
5.13.2.1 `int block_pcb_main (int argc, char ** argv)`

Here is the call graph for this function:



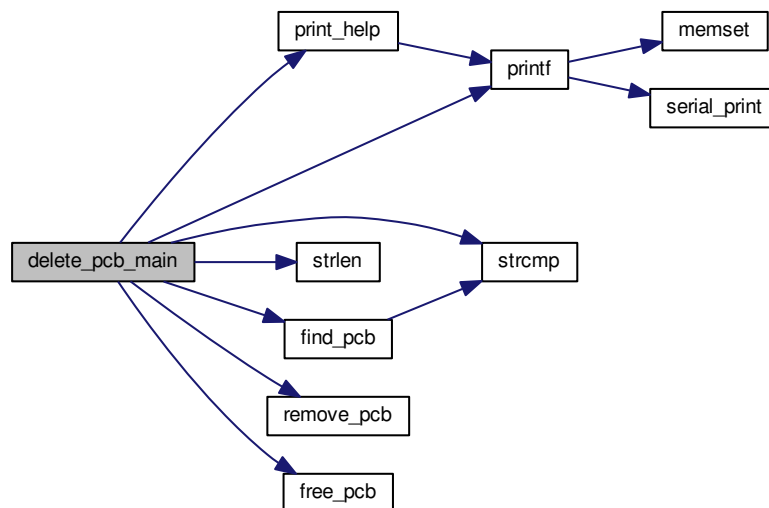
5.13.2.2 int create_pcb_main (int argc, char ** argv)

Here is the call graph for this function:



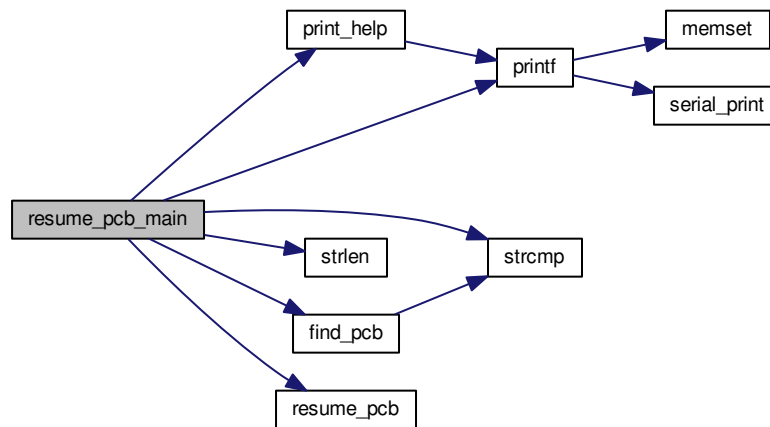
5.13.2.3 int delete_pcb_main (int argc, char ** argv)

Here is the call graph for this function:



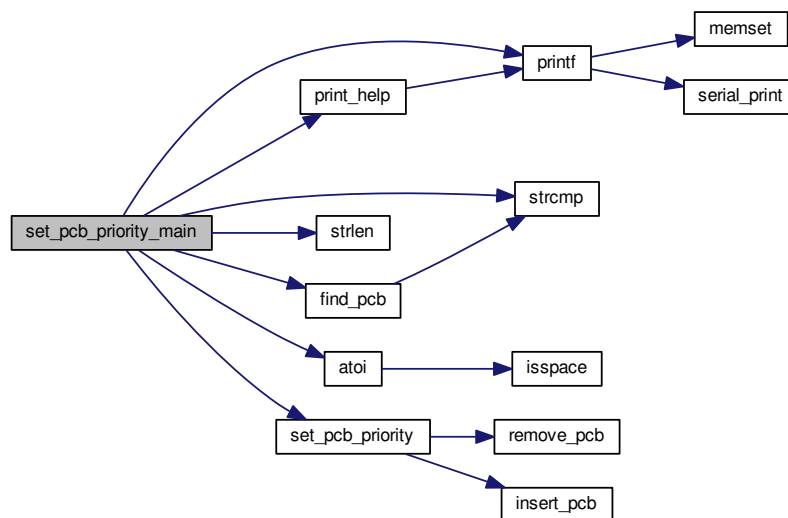
5.13.2.4 int resume_pcb_main (int argc, char ** argv)

Here is the call graph for this function:



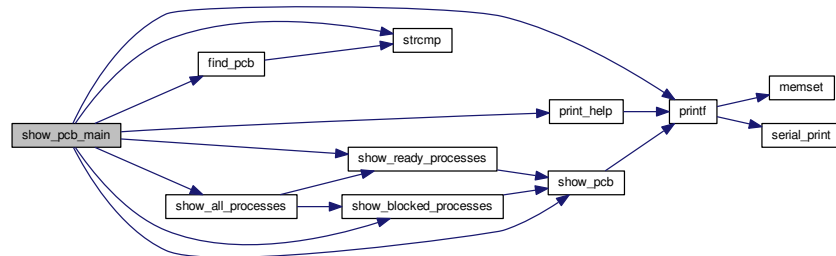
5.13.2.5 int set_pcb_priority_main (int argc, char ** argv)

Here is the call graph for this function:



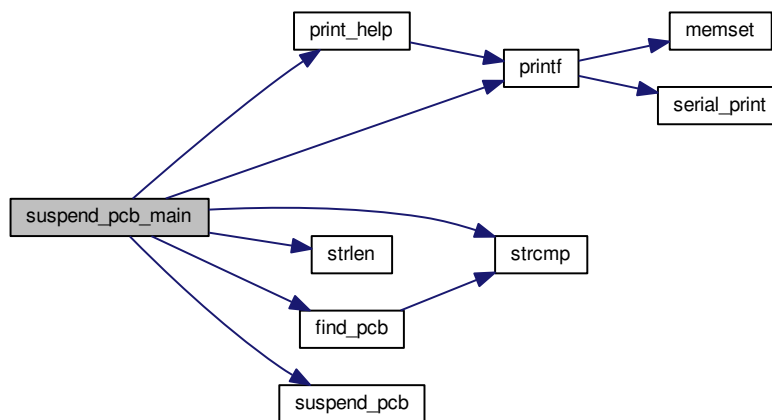
5.13.2.6 int show_pcb_main (int argc, char ** argv)

Here is the call graph for this function:



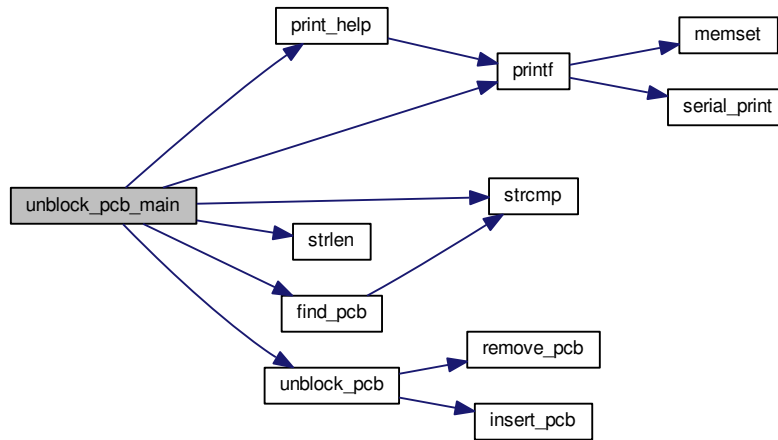
5.13.2.7 int suspend_pcb_main (int argc, char ** argv)

Here is the call graph for this function:



5.13.2.8 int unblock_pcb_main (int argc, char ** argv)

Here is the call graph for this function:



Index

- `__attribute__`
 - `pcb.c`, [71](#), [80](#)
 - `pcb.h`, [86](#)
 - `r1.c`, [39](#)
 - `r1.h`, [43](#)
- `allocate_pcb`
 - `pcb.c`, [71](#)
 - `pcb.h`, [86](#)
- `atoi`
 - `string.c`, [28](#)
 - `string.h`, [17](#)
- `BLOCKPCB`
 - `r1.h`, [43](#)
- `block_pcb`
 - `pcb.c`, [71](#)
 - `pcb.h`, [86](#)
- `block_pcb_main`
 - `pcb_comm.c`, [98](#)
 - `pcb_comm.h`, [105](#)
- `blocked`
 - `pcb.c`, [80](#)
- `COM1`
 - `serial.h`, [12](#)
- `COM2`
 - `serial.h`, [12](#)
- `COM3`
 - `serial.h`, [12](#)
- `COM4`
 - `serial.h`, [12](#)
- `COMPLETION`
 - `r1.c`, [38](#)
- `CREATEPCB`
 - `r1.h`, [43](#)
- `class`
 - `pcb_struct`, [10](#)
- `comm_type`
 - `r1.h`, [43](#)
- `command_line_parser`
 - `r1.c`, [39](#)
 - `r1.h`, [43](#)
- `CommandPaserStat`
 - `r1.c`, [38](#)
- `commhand`
 - `r1.c`, [39](#)
 - `r1.h`, [43](#)
- `count`
 - `pcb_queue`, [8](#)
- `create_pcb_main`
 - `pcb_comm.c`, [98](#)
 - `pcb_comm.h`, [105](#)
- `DELPCB`
 - `r1.h`, [43](#)
- `delete_pcb_main`
 - `pcb_comm.c`, [99](#)
 - `pcb_comm.h`, [106](#)
- `documentation/mainpage.dox`, [11](#)
- `DoubleQuoteWriting`
 - `r1.c`, [40](#)
- `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](#)
- `errno.h`
 - `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`, [80](#)
- `find_pcb`

- pcb.c, 72
- pcb.h, 87
- free_pcb
 - pcb.c, 73
 - pcb.h, 88
- function
 - function_name, 7
- function_name, 7
 - function, 7
 - help, 7
 - nameStr, 7
 - usage, 7
- GETDATE
 - r1.h, 43
- GETTIME
 - r1.h, 43
- get_date
 - sys_clock.c, 51
 - sys_clock.h, 60
- get_date_main
 - sys_clock.c, 51
 - sys_clock.h, 61
- get_input_line
 - serial.h, 12
- get_time
 - sys_clock.c, 52
 - sys_clock.h, 61
- get_time_main
 - sys_clock.c, 52
 - sys_clock.h, 61
- HELP
 - r1.h, 43
- head
 - pcb_queue, 8
- help
 - function_name, 7
 - r1.h, 45
- help_usages
 - r1.c, 39
 - r1.h, 43
- include/core/serial.h, 11
- include/string.h, 13
- init_serial
 - serial.h, 13
- insert_pcb
 - pcb.c, 73
 - pcb.h, 88
- is_suspended
 - pcb_struct, 10
- isspace
 - string.c, 28
 - string.h, 18
- lib/string.c, 24
- MAX_ARGC
 - r1.c, 38
- MOD_VERSION
 - r1.c, 38
- memset
 - string.c, 28
 - string.h, 18
- modules/errno.h, 34
- modules/r1/r1.c, 35
- modules/r1/r1.h, 41
- modules/r1/sys_clock.c, 46
- modules/r1/sys_clock.h, 57
- modules/r2/pcb.c, 66
- modules/r2/pcb.h, 80
- modules/r2/pcb_comm.c, 95
- modules/r2/pcb_comm.h, 102
- mpx
 - r1.h, 45
- NUM_OF_FUNCTIONS
 - r1.h, 43
- name
 - pcb_struct, 10
- nameStr
 - function_name, 7
- next
 - pcb_struct, 10
- NormalWriting
 - r1.c, 40
- NotWriting
 - r1.c, 40
- pcb
 - r1.h, 45
- pcb.c
 - __attribute__, 71, 80
 - allocate_pcb, 71
 - block_pcb, 71
 - blocked, 80
 - false, 80
 - find_pcb, 72
 - free_pcb, 73
 - insert_pcb, 73
 - pcb_init, 74
 - process_state, 71
 - process_suspended, 71
 - ready, 80
 - remove_pcb, 74
 - resume_pcb, 74
 - running, 80
 - set_pcb_priority, 75
 - setup_pcb, 75
 - show_all_processes, 76

- show_blocked_processes, 77
- show_pcb, 77
- show_ready_processes, 78
- suspend_pcb, 79
- true, 80
- unblock_pcb, 79
- pcb.h
 - __attribute__, 86
 - allocate_pcb, 86
 - block_pcb, 86
 - find_pcb, 87
 - free_pcb, 88
 - insert_pcb, 88
 - pcb_class_app, 95
 - pcb_class_sys, 95
 - pcb_init, 89
 - process_class, 86
 - remove_pcb, 89
 - resume_pcb, 89
 - SIZE_OF_PCB_NAME, 86
 - SIZE_OF_STACK, 86
 - set_pcb_priority, 90
 - setup_pcb, 90
 - show_all_processes, 91
 - show_blocked_processes, 92
 - show_pcb, 92
 - show_ready_processes, 93
 - suspend_pcb, 94
 - unblock_pcb, 94
- pcb_class_app
 - pcb.h, 95
- pcb_class_sys
 - pcb.h, 95
- pcb_comm.c
 - block_pcb_main, 98
 - create_pcb_main, 98
 - delete_pcb_main, 99
 - resume_pcb_main, 99
 - set_pcb_priority_main, 100
 - show_pcb_main, 100
 - suspend_pcb_main, 101
 - unblock_pcb_main, 101
- pcb_comm.h
 - block_pcb_main, 105
 - create_pcb_main, 105
 - delete_pcb_main, 106
 - resume_pcb_main, 106
 - set_pcb_priority_main, 107
 - show_pcb_main, 107
 - suspend_pcb_main, 108
 - unblock_pcb_main, 108
- pcb_init
 - pcb.c, 74
 - pcb.h, 89
- pcb_queue, 8
 - count, 8
 - head, 8
 - tail, 9
- pcb_struct, 9
 - class, 10
 - is_suspended, 10
 - name, 10
 - next, 10
 - prev, 10
 - priority, 10
 - running_state, 10
 - stack_base, 10
 - stack_top, 10
- prev
 - pcb_struct, 10
- print_help
 - r1.c, 39
 - r1.h, 44
- printf
 - string.c, 29
 - string.h, 19
- priority
 - pcb_struct, 10
- process_class
 - pcb.h, 86
- process_state
 - pcb.c, 71
- process_suspended
 - pcb.c, 71
- r1.c
 - __attribute__, 39
 - COMPLETION, 38
 - command_line_parser, 39
 - CommandPaserStat, 38
 - commhand, 39
 - DoubleQuoteWriting, 40
 - help_usages, 39
 - MAX_ARGC, 38
 - MOD_VERSION, 38
 - NormalWriting, 40
 - NotWriting, 40
 - print_help, 39
 - SingleQuoteWriting, 41
 - USER_INPUT_BUFFER_SIZE, 38
- r1.h
 - __attribute__, 43
 - BLOCKPCB, 43
 - CREATEPCB, 43
 - comm_type, 43
 - command_line_parser, 43
 - commhand, 43
 - DELPCB, 43

- GETDATE, [43](#)
- GETTIME, [43](#)
- HELP, [43](#)
- help, [45](#)
- help_usages, [43](#)
- mpx, [45](#)
- NUM_OF_FUNCTIONS, [43](#)
- pcb, [45](#)
- print_help, [44](#)
- RESUMEPCB, [43](#)
- SETDATE, [43](#)
- SETPCBPRIO, [43](#)
- SETTIME, [43](#)
- SHOWPCB, [43](#)
- SHUTDOWN, [43](#)
- SUSPDPCB, [43](#)
- UNBLKPCB, [43](#)
- VERSION, [43](#)
- RESUMEPCB
 - [r1.h](#), [43](#)
- RTC_INDEX_DAY_MONTH
 - [sys_clock.c](#), [51](#)
- RTC_INDEX_DAY_WEEK
 - [sys_clock.c](#), [51](#)
- RTC_INDEX_HOUR
 - [sys_clock.c](#), [51](#)
- RTC_INDEX_HOUR_ALARM
 - [sys_clock.c](#), [51](#)
- RTC_INDEX_MINUTE
 - [sys_clock.c](#), [51](#)
- RTC_INDEX_MINUTE_ALARM
 - [sys_clock.c](#), [51](#)
- RTC_INDEX_MONTH
 - [sys_clock.c](#), [51](#)
- RTC_INDEX_SECOND
 - [sys_clock.c](#), [51](#)
- RTC_INDEX_SECOND_ALARM
 - [sys_clock.c](#), [51](#)
- RTC_INDEX_YEAR
 - [sys_clock.c](#), [51](#)
- ready
 - [pcb.c](#), [80](#)
- remove_pcb
 - [pcb.c](#), [74](#)
 - [pcb.h](#), [89](#)
- resume_pcb
 - [pcb.c](#), [74](#)
 - [pcb.h](#), [89](#)
- resume_pcb_main
 - [pcb_comm.c](#), [99](#)
 - [pcb_comm.h](#), [106](#)
- running
 - [pcb.c](#), [80](#)
- running_state
 - [pcb_struct](#), [10](#)
- SETDATE
 - [r1.h](#), [43](#)
- SETPCBPRIO
 - [r1.h](#), [43](#)
- SETTIME
 - [r1.h](#), [43](#)
- SHOWPCB
 - [r1.h](#), [43](#)
- SHUTDOWN
 - [r1.h](#), [43](#)
- SIZE_OF_PCB_NAME
 - [pcb.h](#), [86](#)
- SIZE_OF_STACK
 - [pcb.h](#), [86](#)
- SUSPDPCB
 - [r1.h](#), [43](#)
- serial.h
 - COM1, [12](#)
 - COM2, [12](#)
 - COM3, [12](#)
 - COM4, [12](#)
 - get_input_line, [12](#)
 - init_serial, [13](#)
 - serial_print, [13](#)
 - serial_println, [13](#)
 - set_serial_in, [13](#)
 - set_serial_out, [13](#)
 - WithEcho, [12](#)
 - WithoutEcho, [12](#)
- serial_print
 - [serial.h](#), [13](#)
- serial_println
 - [serial.h](#), [13](#)
- set_date
 - [sys_clock.c](#), [53](#)
 - [sys_clock.h](#), [62](#)
- set_date_main
 - [sys_clock.c](#), [53](#)
 - [sys_clock.h](#), [62](#)
- set_date_str
 - [sys_clock.c](#), [54](#)
 - [sys_clock.h](#), [63](#)
- set_pcb_priority
 - [pcb.c](#), [75](#)
 - [pcb.h](#), [90](#)
- set_pcb_priority_main
 - [pcb_comm.c](#), [100](#)
 - [pcb_comm.h](#), [107](#)
- set_serial_in
 - [serial.h](#), [13](#)
- set_serial_out
 - [serial.h](#), [13](#)

- set_time
 - sys_clock.c, 55
 - sys_clock.h, 64
- set_time_main
 - sys_clock.c, 55
 - sys_clock.h, 64
- set_time_str
 - sys_clock.c, 56
 - sys_clock.h, 65
- setup_pcb
 - pcb.c, 75
 - pcb.h, 90
- show_all_processes
 - pcb.c, 76
 - pcb.h, 91
- show_blocked_processes
 - pcb.c, 77
 - pcb.h, 92
- show_pcb
 - pcb.c, 77
 - pcb.h, 92
- show_pcb_main
 - pcb_comm.c, 100
 - pcb_comm.h, 107
- show_ready_processes
 - pcb.c, 78
 - pcb.h, 93
- SingleQuoteWriting
 - r1.c, 41
- sprintf
 - string.c, 30
 - string.h, 20
- stack_base
 - pcb_struct, 10
- stack_top
 - pcb_struct, 10
- strcat
 - string.c, 30
 - string.h, 20
- strcmp
 - string.c, 30
 - string.h, 20
- strcpy
 - string.c, 31
 - string.h, 21
- string.c
 - atoi, 28
 - isspace, 28
 - memset, 28
 - printf, 29
 - sprintf, 30
 - strcat, 30
 - strcmp, 30
 - strcpy, 31
 - strlen, 32
 - strtok, 33
- string.h
 - atoi, 17
 - isspace, 18
 - memset, 18
 - printf, 19
 - sprintf, 20
 - strcat, 20
 - strcmp, 20
 - strcpy, 21
 - strlen, 22
 - strtok, 23
- strlen
 - string.c, 32
 - string.h, 22
- strtok
 - string.c, 33
 - string.h, 23
- suspend_pcb
 - pcb.c, 79
 - pcb.h, 94
- suspend_pcb_main
 - pcb_comm.c, 101
 - pcb_comm.h, 108
- sys_clock.c
 - get_date, 51
 - get_date_main, 51
 - get_time, 52
 - get_time_main, 52
 - RTC_INDEX_DAY_MONTH, 51
 - RTC_INDEX_DAY_WEEK, 51
 - RTC_INDEX_HOUR, 51
 - RTC_INDEX_HOUR_ALARM, 51
 - RTC_INDEX_MINUTE, 51
 - RTC_INDEX_MINUTE_ALARM, 51
 - RTC_INDEX_MONTH, 51
 - RTC_INDEX_SECOND, 51
 - RTC_INDEX_SECOND_ALARM, 51
 - RTC_INDEX_YEAR, 51
 - set_date, 53
 - set_date_main, 53
 - set_date_str, 54
 - set_time, 55
 - set_time_main, 55
 - set_time_str, 56
- sys_clock.h
 - get_date, 60
 - get_date_main, 61
 - get_time, 61
 - get_time_main, 61
 - set_date, 62
 - set_date_main, 62
 - set_date_str, 63

- set_time, [64](#)
- set_time_main, [64](#)
- set_time_str, [65](#)

tail

- pcb_queue, [9](#)

true

- pcb.c, [80](#)

UNBLKPCB

- r1.h, [43](#)

USER_INPUT_BUFFER_SIZE

- r1.c, [38](#)

unblock_pcb

- pcb.c, [79](#)

- pcb.h, [94](#)

unblock_pcb_main

- pcb_comm.c, [101](#)

- pcb_comm.h, [108](#)

usage

- function_name, [7](#)

VERSION

- r1.h, [43](#)

WithEcho

- serial.h, [12](#)

WithoutEcho

- serial.h, [12](#)