

MPX Thunder Krakens

R4

Generated by Doxygen 1.8.6

Thu Mar 24 2016 22:01:04

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	context Struct Reference	7
4.1.1	Detailed Description	8
4.1.2	Field Documentation	8
4.1.2.1	cs	8
4.1.2.2	ds	8
4.1.2.3	eax	8
4.1.2.4	ebp	8
4.1.2.5	ebx	8
4.1.2.6	ecx	8
4.1.2.7	edi	8
4.1.2.8	edx	8
4.1.2.9	eflags	9
4.1.2.10	eip	9
4.1.2.11	es	9
4.1.2.12	esi	9
4.1.2.13	esp	9
4.1.2.14	fs	9
4.1.2.15	gs	9
4.2	function_name Struct Reference	9
4.2.1	Detailed Description	10

4.2.2	Field Documentation	10
4.2.2.1	function	10
4.2.2.2	help	10
4.2.2.3	nameStr	10
4.2.2.4	usage	10
4.3	param Struct Reference	10
4.3.1	Detailed Description	10
4.3.2	Field Documentation	10
4.3.2.1	device_id	10
4.3.2.2	op_code	11
4.4	pcb_queue Struct Reference	11
4.4.1	Detailed Description	11
4.4.2	Field Documentation	11
4.4.2.1	count	11
4.4.2.2	head	12
4.4.2.3	tail	12
4.5	pcb_struct Struct Reference	12
4.5.1	Detailed Description	13
4.5.2	Field Documentation	13
4.5.2.1	class	13
4.5.2.2	is_suspended	13
4.5.2.3	name	13
4.5.2.4	next	13
4.5.2.5	prev	13
4.5.2.6	priority	13
4.5.2.7	running_state	13
4.5.2.8	stack_base	13
4.5.2.9	stack_top	13
5	File Documentation	15
5.1	documentation/mainpage.dox File Reference	15
5.2	include/core/serial.h File Reference	15
5.2.1	Detailed Description	16
5.2.2	Macro Definition Documentation	16
5.2.2.1	COM1	16
5.2.2.2	COM2	16
5.2.2.3	COM3	16

5.2.2.4	COM4	16
5.2.2.5	USER_INPUT_BUFFER_SIZE	16
5.2.2.6	WithEcho	16
5.2.2.7	WithoutEcho	17
5.2.3	Function Documentation	17
5.2.3.1	get_input_line	17
5.2.3.2	init_serial	17
5.2.3.3	serial_print	17
5.2.3.4	serial_println	17
5.2.3.5	set_serial_in	17
5.2.3.6	set_serial_out	17
5.3	include/string.h File Reference	17
5.3.1	Detailed Description	21
5.3.2	Function Documentation	21
5.3.2.1	atoi	22
5.3.2.2	isspace	22
5.3.2.3	memset	23
5.3.2.4	printf	23
5.3.2.5	sprintf	24
5.3.2.6	strcat	24
5.3.2.7	strcmp	25
5.3.2.8	strcpy	25
5.3.2.9	strlen	26
5.3.2.10	strtok	26
5.4	lib/string.c File Reference	26
5.4.1	Detailed Description	30
5.4.2	Function Documentation	30
5.4.2.1	atoi	30
5.4.2.2	isspace	31
5.4.2.3	memset	32
5.4.2.4	printf	32
5.4.2.5	sprintf	33
5.4.2.6	strcat	33
5.4.2.7	strcmp	34
5.4.2.8	strcpy	34
5.4.2.9	strlen	35
5.4.2.10	strtok	35

5.5	modules/errno.h File Reference	35
5.5.1	Detailed Description	36
5.5.2	Macro Definition Documentation	37
5.5.2.1	E_EMPTPCB	37
5.5.2.2	E_FREEMEM	37
5.5.2.3	E_INVPARA	37
5.5.2.4	E_INVSTRF	37
5.5.2.5	E_INVUSRI	37
5.5.2.6	E_NOERROR	37
5.5.2.7	E_NULL_PTR	37
5.5.2.8	E_PCB_SYS	37
5.5.2.9	E_PROGERR	37
5.5.3	Typedef Documentation	37
5.5.3.1	error_t	37
5.6	modules/mpx_supt.c File Reference	37
5.6.1	Function Documentation	38
5.6.1.1	get_op_code	39
5.6.1.2	idle	39
5.6.1.3	mpx_init	39
5.6.1.4	sys_alloc_mem	39
5.6.1.5	sys_free_mem	40
5.6.1.6	sys_req	40
5.6.1.7	sys_set_free	40
5.6.1.8	sys_set_malloc	40
5.6.2	Variable Documentation	40
5.6.2.1	current_module	40
5.6.2.2	params	40
5.6.2.3	student_free	40
5.6.2.4	student_malloc	40
5.7	modules/mpx_supt.h File Reference	41
5.7.1	Detailed Description	43
5.7.2	Macro Definition Documentation	43
5.7.2.1	EXIT	43
5.7.2.2	IDLE	43
5.7.2.3	MODULE_R1	43
5.7.2.4	MODULE_R2	43
5.7.2.5	MODULE_R3	44

5.7.2.6	MODULE_R4	44
5.7.2.7	MODULE_R5	44
5.7.2.8	READ	44
5.7.2.9	WRITE	44
5.7.3	Function Documentation	44
5.7.3.1	get_op_code	44
5.7.3.2	idle	44
5.7.3.3	mpx_init	44
5.7.3.4	sys_alloc_mem	45
5.7.3.5	sys_free_mem	45
5.7.3.6	sys_req	45
5.7.3.7	sys_set_free	45
5.7.3.8	sys_set_malloc	45
5.7.4	Variable Documentation	46
5.7.4.1	__attribute__	46
5.8	modules/r1/r1.c File Reference	46
5.8.1	Detailed Description	49
5.8.2	Macro Definition Documentation	49
5.8.2.1	COMPLETION	49
5.8.2.2	MAX_ARGC	49
5.8.2.3	MAX_HISTORY	49
5.8.2.4	MOD_VERSION	49
5.8.3	Enumeration Type Documentation	49
5.8.3.1	CommandPaserStat	49
5.8.4	Function Documentation	49
5.8.4.1	__attribute__	49
5.8.4.2	command_line_parser	49
5.8.4.3	commhand	50
5.8.4.4	help_usages	50
5.8.4.5	print_help	50
5.8.5	Variable Documentation	51
5.8.5.1	DoubleQuoteWriting	51
5.8.5.2	NormalWriting	51
5.8.5.3	NotWriting	51
5.8.5.4	SingleQuoteWriting	51
5.9	modules/r1/r1.h File Reference	52
5.9.1	Detailed Description	54

5.9.2	Macro Definition Documentation	55
5.9.2.1	GETDATE	55
5.9.2.2	GETTIME	55
5.9.2.3	HELP	55
5.9.2.4	LOADR3	55
5.9.2.5	NUM_MPX_FUNCTIONS	55
5.9.2.6	NUM_OF_FUNCTIONS	55
5.9.2.7	POS_OF_MPX	55
5.9.2.8	POS_OF_PCB	55
5.9.2.9	RESUMEPCB	55
5.9.2.10	SETDATE	55
5.9.2.11	SETPCBPRIO	55
5.9.2.12	SETTIME	55
5.9.2.13	SHOWPCB	55
5.9.2.14	SHUTDOWN	55
5.9.2.15	SUSPDCB	55
5.9.2.16	VERSION	55
5.9.2.17	YIELD	55
5.9.3	Enumeration Type Documentation	55
5.9.3.1	comm_type	55
5.9.4	Function Documentation	55
5.9.4.1	__attribute__	55
5.9.4.2	command_line_parser	56
5.9.4.3	commhand	56
5.9.4.4	help_usages	56
5.9.4.5	print_help	56
5.9.5	Variable Documentation	57
5.9.5.1	help	57
5.9.5.2	mpx	57
5.9.5.3	pcb	57
5.10	modules/r1/sys_clock.c File Reference	57
5.10.1	Detailed Description	61
5.10.2	Macro Definition Documentation	62
5.10.2.1	RTC_INDEX_DAY_MONTH	62
5.10.2.2	RTC_INDEX_DAY_WEEK	62
5.10.2.3	RTC_INDEX_HOUR	62
5.10.2.4	RTC_INDEX_HOUR_ALARM	62

5.10.2.5	RTC_INDEX_MINUTE	62
5.10.2.6	RTC_INDEX_MINUTE_ALARM	62
5.10.2.7	RTC_INDEX_MONTH	62
5.10.2.8	RTC_INDEX_SECOND	62
5.10.2.9	RTC_INDEX_SECOND_ALARM	62
5.10.2.10	RTC_INDEX_YEAR	62
5.10.3	Function Documentation	62
5.10.3.1	get_date	62
5.10.3.2	get_date_main	63
5.10.3.3	get_time	63
5.10.3.4	get_time_main	64
5.10.3.5	set_date	64
5.10.3.6	set_date_main	65
5.10.3.7	set_date_str	65
5.10.3.8	set_time	66
5.10.3.9	set_time_main	67
5.10.3.10	set_time_str	67
5.11	modules/r1/sys_clock.h File Reference	68
5.11.1	Detailed Description	71
5.11.2	Function Documentation	71
5.11.2.1	get_date	71
5.11.2.2	get_date_main	72
5.11.2.3	get_time	72
5.11.2.4	get_time_main	73
5.11.2.5	set_date	73
5.11.2.6	set_date_main	74
5.11.2.7	set_date_str	74
5.11.2.8	set_time	75
5.11.2.9	set_time_main	76
5.11.2.10	set_time_str	76
5.12	modules/r2/pcb.c File Reference	77
5.12.1	Detailed Description	83
5.12.2	Enumeration Type Documentation	83
5.12.2.1	process_state	83
5.12.2.2	process_suspended	83
5.12.3	Function Documentation	83
5.12.3.1	__attribute__	83

5.12.3.2	allocate_pcb	83
5.12.3.3	block_pcb	84
5.12.3.4	find_pcb	84
5.12.3.5	free_pcb	85
5.12.3.6	get_running_process	86
5.12.3.7	get_stack_base	86
5.12.3.8	get_stack_top	87
5.12.3.9	insert_pcb	87
5.12.3.10	pcb_init	87
5.12.3.11	remove_pcb	88
5.12.3.12	resume_pcb	88
5.12.3.13	save_running_process	88
5.12.3.14	set_pcb_priority	89
5.12.3.15	setup_pcb	90
5.12.3.16	show_all_processes	90
5.12.3.17	show_blocked_processes	91
5.12.3.18	show_pcb	92
5.12.3.19	show_ready_processes	92
5.12.3.20	shutdown_pcb	93
5.12.3.21	suspend_pcb	93
5.12.3.22	unblock_pcb	94
5.12.4	Variable Documentation	94
5.12.4.1	__attribute__	94
5.12.4.2	blocked	94
5.12.4.3	false	94
5.12.4.4	ready	94
5.12.4.5	running	94
5.12.4.6	true	94
5.13	modules/r2/pcb.h File Reference	94
5.13.1	Detailed Description	100
5.13.2	Macro Definition Documentation	101
5.13.2.1	SIZE_OF_PCB_NAME	101
5.13.2.2	SIZE_OF_STACK	101
5.13.3	Enumeration Type Documentation	101
5.13.3.1	process_class	101
5.13.4	Function Documentation	101
5.13.4.1	__attribute__	101

5.13.4.2	allocate_pcb	101
5.13.4.3	block_pcb	102
5.13.4.4	find_pcb	102
5.13.4.5	free_pcb	103
5.13.4.6	get_running_process	103
5.13.4.7	get_stack_base	104
5.13.4.8	get_stack_top	104
5.13.4.9	insert_pcb	105
5.13.4.10	pcb_init	105
5.13.4.11	remove_pcb	105
5.13.4.12	resume_pcb	106
5.13.4.13	save_running_process	106
5.13.4.14	set_pcb_priority	107
5.13.4.15	setup_pcb	107
5.13.4.16	show_all_processes	108
5.13.4.17	show_blocked_processes	108
5.13.4.18	show_pcb	109
5.13.4.19	show_ready_processes	110
5.13.4.20	shutdown_pcb	110
5.13.4.21	suspend_pcb	111
5.13.4.22	unblock_pcb	111
5.13.5	Variable Documentation	111
5.13.5.1	pcb_class_app	111
5.13.5.2	pcb_class_sys	111
5.14	modules/r2/pcb_comm.c File Reference	111
5.14.1	Detailed Description	113
5.14.2	Function Documentation	114
5.14.2.1	resume_pcb_main	114
5.14.2.2	set_pcb_priority_main	114
5.14.2.3	show_pcb_main	115
5.14.2.4	suspend_pcb_main	115
5.15	modules/r2/pcb_comm.h File Reference	115
5.15.1	Detailed Description	118
5.15.2	Function Documentation	118
5.15.2.1	block_pcb_main	118
5.15.2.2	create_pcb_main	118
5.15.2.3	delete_pcb_main	118

5.15.2.4	resume_pcb_main	119
5.15.2.5	set_pcb_priority_main	119
5.15.2.6	show_pcb_main	120
5.15.2.7	suspend_pcb_main	120
5.15.2.8	unblock_pcb_main	120
5.16	modules/r3/context.c File Reference	120
5.16.1	Detailed Description	122
5.16.2	Function Documentation	123
5.16.2.1	load_process	123
5.16.2.2	load_r3_main	124
5.16.2.3	sys_call	124
5.16.2.4	yield_main	125
5.16.3	Variable Documentation	125
5.16.3.1	cop	125
5.16.3.2	old_context	125
5.17	modules/r3/context.h File Reference	125
5.17.1	Detailed Description	128
5.17.2	Function Documentation	128
5.17.2.1	load_process	128
5.17.2.2	load_r3_main	129
5.17.2.3	sys_call	130
5.17.2.4	yield_main	130
5.17.3	Variable Documentation	130
5.17.3.1	cop	130
5.17.3.2	old_context	130

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:

context	Context structure that holds the 15 CPU register values to begin and resume process execution . . .	7
function_name	A structure to represent each function	9
param	A structure to represent interrupt	10
pcb_queue	Queue structure that will store PCBs	11
pcb_struct	Struct that will describe PCB Processes	12

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	17
include/core/ serial.h	
Serial - Header	15
lib/ string.c	
Many usefull functions that used for handling string	26
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	35
modules/ mpx_supt.c	37
modules/ mpx_supt.h	
MPX System Supplementaries	41
modules/r1/ r1.c	
The commandhander and functions associations for Module R1	46
modules/r1/ r1.h	
The command handler and functions associations for Module R1	52
modules/r1/ sys_clock.c	
The main file that manipulates and controls the system's clock	57
modules/r1/ sys_clock.h	
The main file that manipulates and controls the system's clock	68
modules/r2/ pcb.c	
The Process Control Block	77
modules/r2/ pcb.h	
The Process Control Block	94
modules/r2/ pcb_comm.c	
The main functions that manipulate the PCB	111
modules/r2/ pcb_comm.h	
The main functions that manipulate the PCB	115
modules/r3/ context.c	
Context Switching	120
modules/r3/ context.h	
Context Switching	125

Chapter 4

Data Structure Documentation

4.1 context Struct Reference

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

```
#include <context.h>
```

Data Fields

- u32int [gs](#)
Segment register.
- u32int [fs](#)
Segment register.
- u32int [es](#)
Segment register.
- u32int [ds](#)
Segment register.
- u32int [edi](#)
General-purpose register.
- u32int [esi](#)
General-purpose register.
- u32int [ebp](#)
General-purpose register.
- u32int [esp](#)
General-purpose register.
- u32int [ebx](#)
General-purpose register.
- u32int [edx](#)
General-purpose register.
- u32int [ecx](#)
General-purpose register.
- u32int [eax](#)
General-purpose register.
- u32int [eip](#)

Status and control register.

- u32int [cs](#)

Status and control register.

- u32int [eflags](#)

Status and control register.

4.1.1 Detailed Description

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

4.1.2 Field Documentation

4.1.2.1 u32int context::cs

Status and control register.

4.1.2.2 u32int context::ds

Segment register.

4.1.2.3 u32int context::eax

General-purpose register.

4.1.2.4 u32int context::ebp

General-purpose register.

4.1.2.5 u32int context::ebx

General-purpose register.

4.1.2.6 u32int context::ecx

General-purpose register.

4.1.2.7 u32int context::edi

General-purpose register.

4.1.2.8 u32int context::edx

General-purpose register.

4.1.2.9 u32int context::eflags

Status and control register.

4.1.2.10 u32int context::eip

Status and control register.

4.1.2.11 u32int context::es

Segment register.

4.1.2.12 u32int context::esi

General-purpose register.

4.1.2.13 u32int context::esp

General-purpose register.

4.1.2.14 u32int context::fs

Segment register.

4.1.2.15 u32int context::gs

Segment register.

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

- [modules/r3/context.h](#)

4.2 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

4.2.1 Detailed Description

A structure to represent each function.

4.2.2 Field Documentation

4.2.2.1 `int(* function_name::function)(int argc, char **argv)`

the function

4.2.2.2 `char* function_name::help`

function's help information

4.2.2.3 `char* function_name::nameStr`

fuction's name

4.2.2.4 `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`

4.3 param Struct Reference

A structure to represent interrupt.

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

Data Fields

- `int op_code`
interrupt's operation
- `int device_id`
interrupt's device

4.3.1 Detailed Description

A structure to represent interrupt.

4.3.2 Field Documentation

4.3.2.1 `int param::device_id`

interrupt's device

4.3.2.2 int param::op_code

interrupt's operation

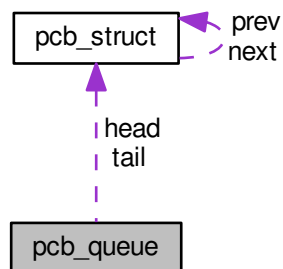
The documentation for this struct was generated from the following file:

- [modules/mpx_supt.h](#)

4.4 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.

4.4.1 Detailed Description

Queue structure that will store PCBs.

4.4.2 Field Documentation

4.4.2.1 int pcb_queue::count

The length of the queue.

4.4.2.2 struct `pcb_struct`* `pcb_queue::head`

Pointer to the start/head of the queue.

4.4.2.3 struct `pcb_struct`* `pcb_queue::tail`

Pointer to the end/tail of the queue.

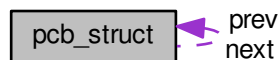
The documentation for this struct was generated from the following file:

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

4.5 `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.

4.5.1 Detailed Description

Struct that will describe PCB Processes.

4.5.2 Field Documentation

4.5.2.1 enum process_class pcb_struct::class

PCB's class is an application or system process.

4.5.2.2 enum process_suspended pcb_struct::is_suspended

PCB process is either suspended or not suspended.

4.5.2.3 char pcb_struct::name[SIZE_OF_PCB_NAME]

PCB's name.

4.5.2.4 struct pcb_struct* pcb_struct::next

Pointer to the next PCB in the queue.

4.5.2.5 struct pcb_struct* pcb_struct::prev

Pointer to the previous PCB in the queue.

4.5.2.6 unsigned char pcb_struct::priority

PCB's priority an integer between 0 and 9.

Processes with higher priority values execute before lower priority processes.

4.5.2.7 enum process_state pcb_struct::running_state

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

4.5.2.8 unsigned char* pcb_struct::stack_base

Pointer to base of the stack.

4.5.2.9 unsigned char* pcb_struct::stack_top

Pointer to top of the stack.

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

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

Chapter 5

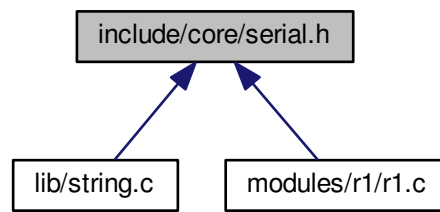
File Documentation

5.1 documentation/mainpage.dox File Reference

5.2 include/core/serial.h File Reference

Serial - Header.

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



Macros

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

Functions

- int [init_serial](#) (int device)
- int [serial_println](#) (const char *msg)
- int [serial_print](#) (const char *msg)
- int [set_serial_out](#) (int device)
- int [set_serial_in](#) (int device)

get_input_line

Get user's input from 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 bWithEcho)

5.2.1 Detailed Description

Serial - Header.

Author

Thunder Krakens

Date

February 2nd, 2016

Version

R1

5.2.2 Macro Definition Documentation

5.2.2.1 **#define** COM1 0x3f8

5.2.2.2 **#define** COM2 0x2f8

5.2.2.3 **#define** COM3 0x3e8

5.2.2.4 **#define** COM4 0x2e8

5.2.2.5 **#define** USER_INPUT_BUFFER_SIZE 100

5.2.2.6 **#define** WithEcho 1

5.2.2.7 #define WithoutEcho 0

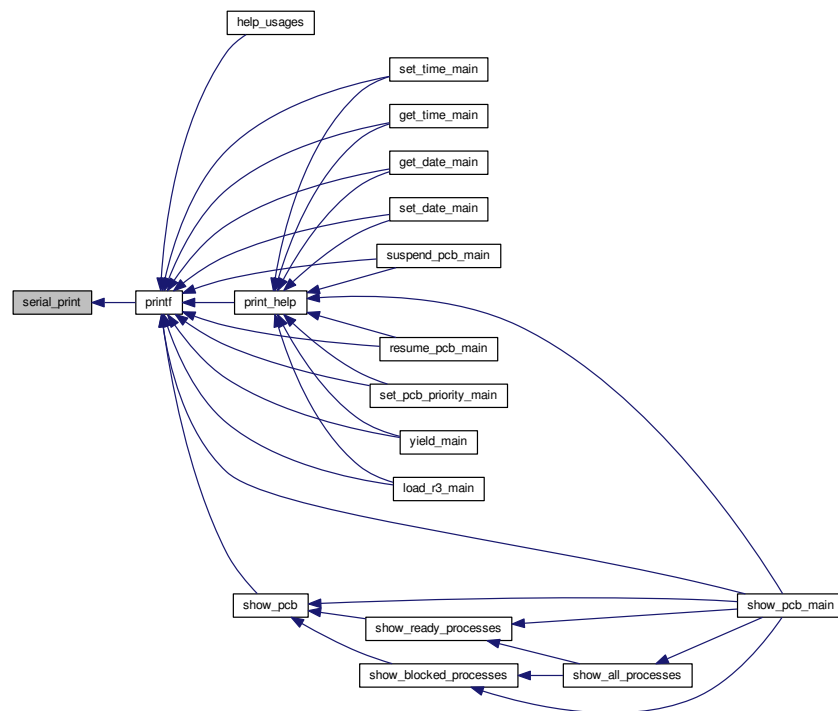
5.2.3 Function Documentation

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

5.2.3.2 int init_serial (int *device*)

5.2.3.3 int serial_print (const char * *msg*)

Here is the caller graph for this function:



5.2.3.4 int serial_println (const char * *msg*)

5.2.3.5 int set_serial_in (int *device*)

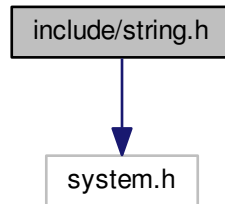
5.2.3.6 int set_serial_out (int *device*)

5.3 include/string.h File Reference

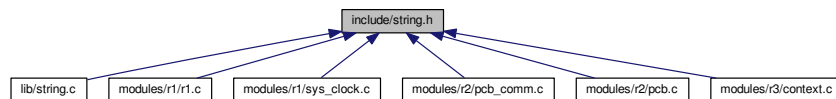
Many usefull functions that used for handling string.

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

Include dependency graph for string.h:



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



Functions

isspace.

Identifies if its space

Parameters

A	constant character
---	--------------------

Returns

1 if it is space, otherwise return 0.

- int `isspace` (const char *c)

memset.

Sets region of memory

Parameters

s	destination
c	byte to write

n	count
---	-------

Returns

the pointer to the memory space.

- void * [memset](#) (void *s, int c, size_t n)

strcpy.

Copies one string to another.

Parameters

s1	Destination string
s2	Source string

Returns

pointer to the destination String

- char * [strcpy](#) (char *s1, const char *s2)

strcat.

Concatenate the contents of one string onto another.

Parameters

s1	Destination string
s2	Source string

Returns

pointer to destination String

- char * [strcat](#) (char *s1, const char *s2)

strlen.

Returns the length of a string.

Parameters

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

Returns

count Length of the String

- int [strlen](#) (const char *s)

strcmp.

String comparison.

Parameters

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

Returns

whether they are the same or not.

- int [strcmp](#) (const char *s1, const char *s2)

strtok.

Split string into tokens.

Parameters

s1	String
s2	Delimiter

Returns

the pointer to the token.

- char * [strtok](#) (char *s1, const char *s2)

atoi.

Convert an ASCII string to an integer.

Parameters

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

Returns

The converted integer.

- int [atoi](#) (const char *s)

sprintf.

Generate a formatted string.

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

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

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

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

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

Parameters

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

Returns

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

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

printf.

Print out a formatted string.

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

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

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

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

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

Parameters

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

Returns

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

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

5.3.1 Detailed Description

Many usefull functions that used for handling string.

Author

Thunder Krakens

Date

February 2nd, 2016

Version

R1

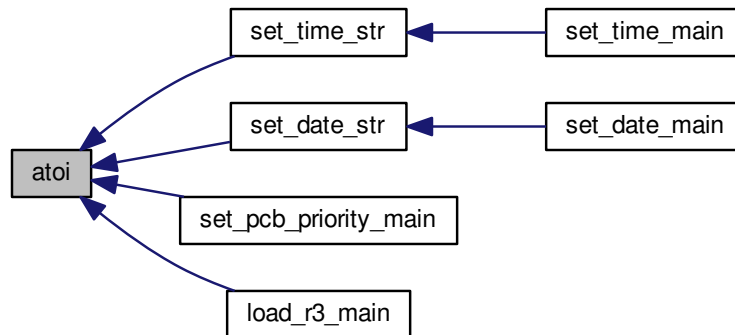
5.3.2 Function Documentation

5.3.2.1 int atoi (const char * s)

Here is the call graph for this function:

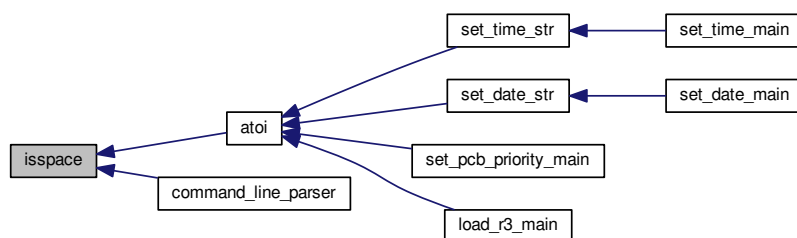


Here is the caller graph for this function:



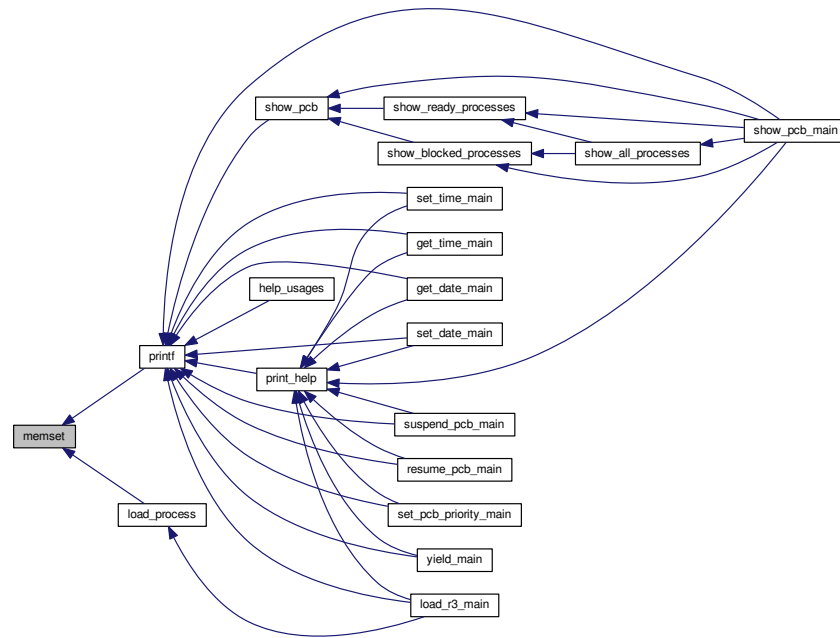
5.3.2.2 int isspace (const char * c)

Here is the caller graph for this function:



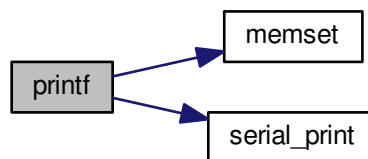
5.3.2.3 void* memset (void * s, int c, size_t n)

Here is the caller graph for this function:

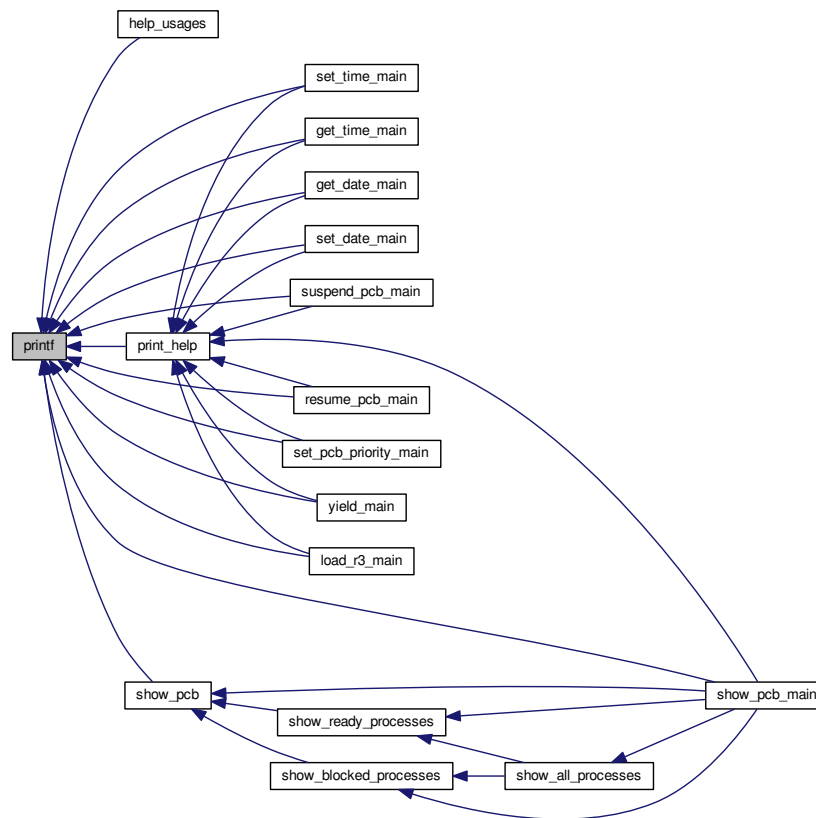


5.3.2.4 int printf (const char * format, ...)

Here is the call graph for this function:



Here is the caller graph for this function:

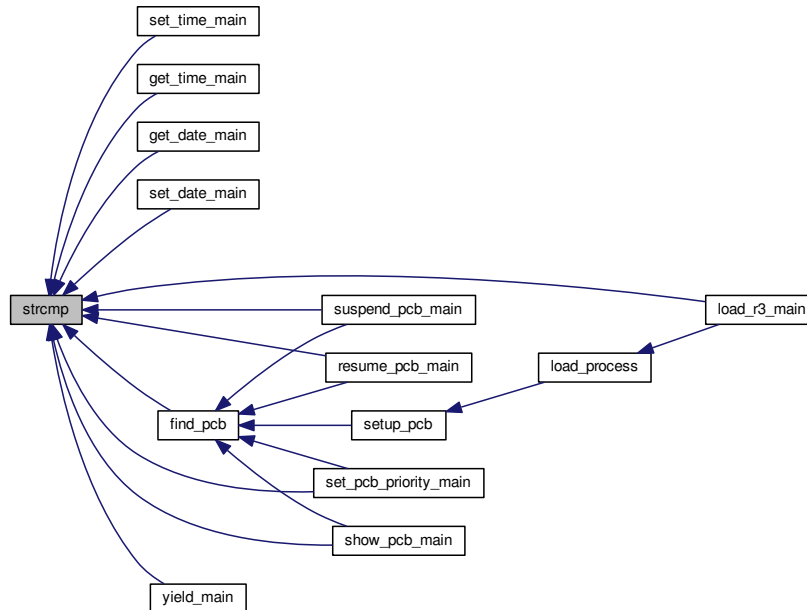


5.3.2.5 `int sprintf (char * str, const char * format, ...)`

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

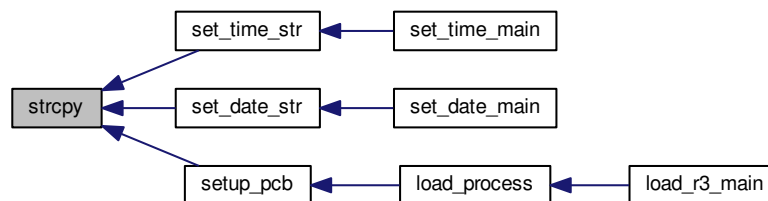
5.3.2.7 int strcmp (const char * s1, const char * s2)

Here is the caller graph for this function:



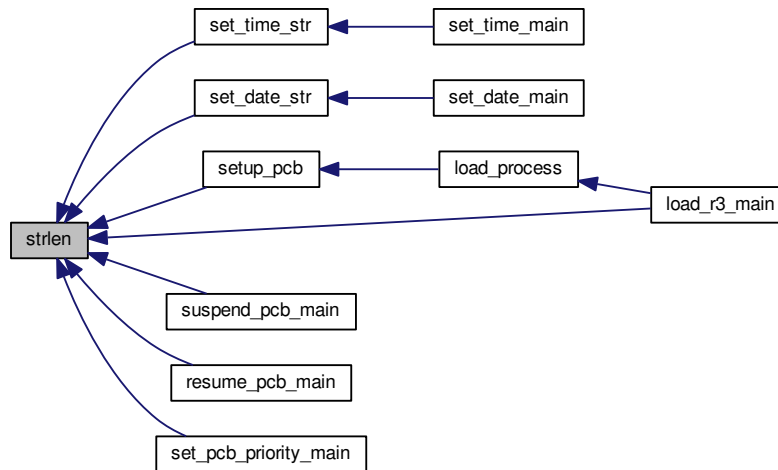
5.3.2.8 char* strcpy (char * s1, const char * s2)

Here is the caller graph for this function:



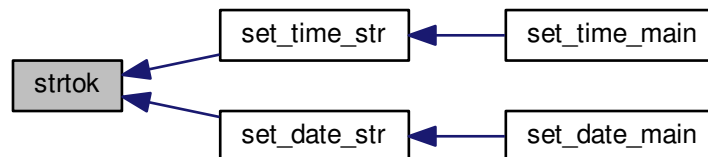
5.3.2.9 int strlen (const char * s)

Here is the caller graph for this function:



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

Here is the caller graph for this function:



5.4 lib/string.c File Reference

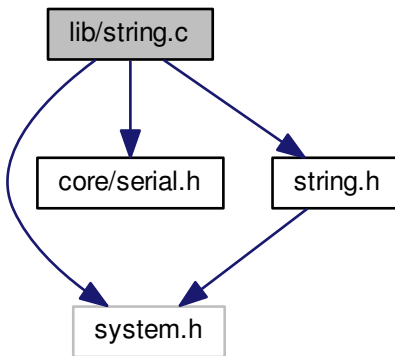
Many usefull functions that used for handling string.

```

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

```

Include dependency graph for string.c:



Functions

strlen.

Returns the length of a string.

Parameters

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

Returns

count Length of the String

- int [strlen](#) (const char *s)

strcpy.

Copies one string to another.

Parameters

s1	Destination string
s2	Source string

Returns

pointer to the destination String

- char * [strcpy](#) (char *s1, const char *s2)

atoi.

Convert an ASCII string to an integer.

Parameters

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

Returns

The converted integer.

- int [atoi](#) (const char *s)

strcmp.

String comparison.

Parameters

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

Returns

whether they are the same or not.

- int [strcmp](#) (const char *s1, const char *s2)

ParsePadding.

Parse the number for padding.

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

Parameters

str	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	In string.
count	Number of whitespace.

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	<i>- Output string.</i>
format	<i>- The format of the string.</i>
...	<i>- All of the additional parameters.</i>

Returns

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

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

printf.

Print out a formatted string.

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

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

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

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

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

Parameters

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

Returns

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

- int [printf](#) (const char *format,...)
- char * [strcat](#) (char *s1, const char *s2)
- int [isspace](#) (const char *c)
- void * [memset](#) (void *s, int c, size_t n)
- char * [strtok](#) (char *s1, const char *s2)

5.4.1 Detailed Description

Many usefull functions that used for handling string.

Author

Thunder Krakens

Date

February 2nd, 2016

Version

R1

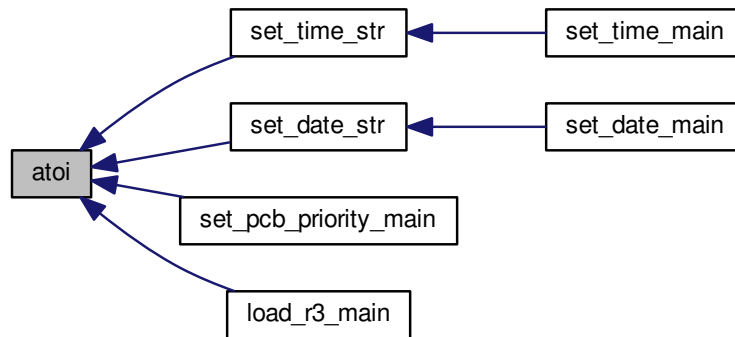
5.4.2 Function Documentation

5.4.2.1 int atoi (const char * s)

Here is the call graph for this function:

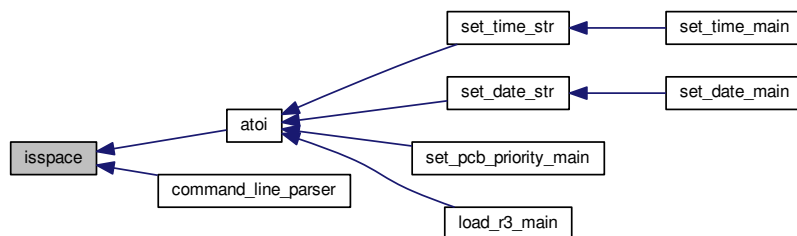


Here is the caller graph for this function:



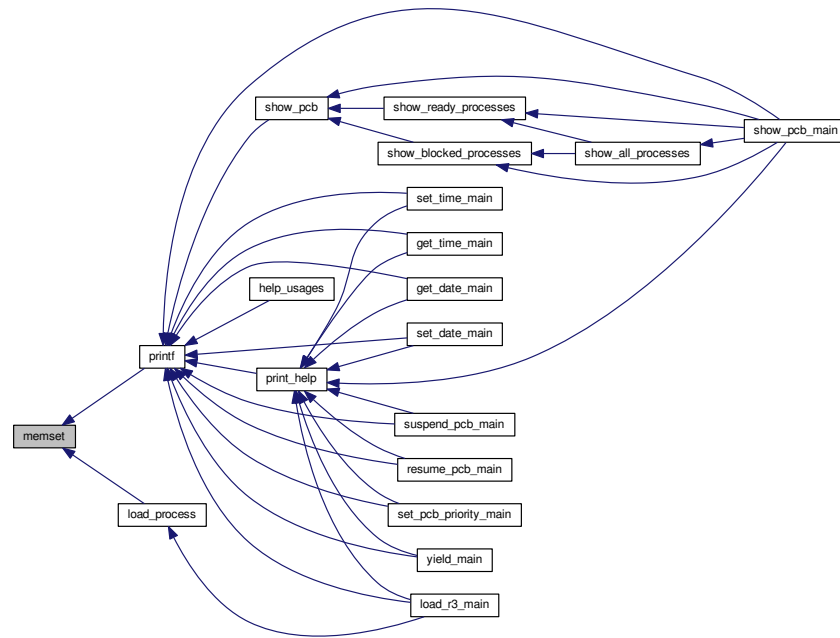
5.4.2.2 `int isspace (const char * c)`

Here is the caller graph for this function:



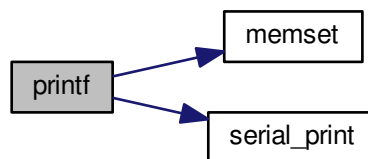
5.4.2.3 void* memset (void * s, int c, size_t n)

Here is the caller graph for this function:

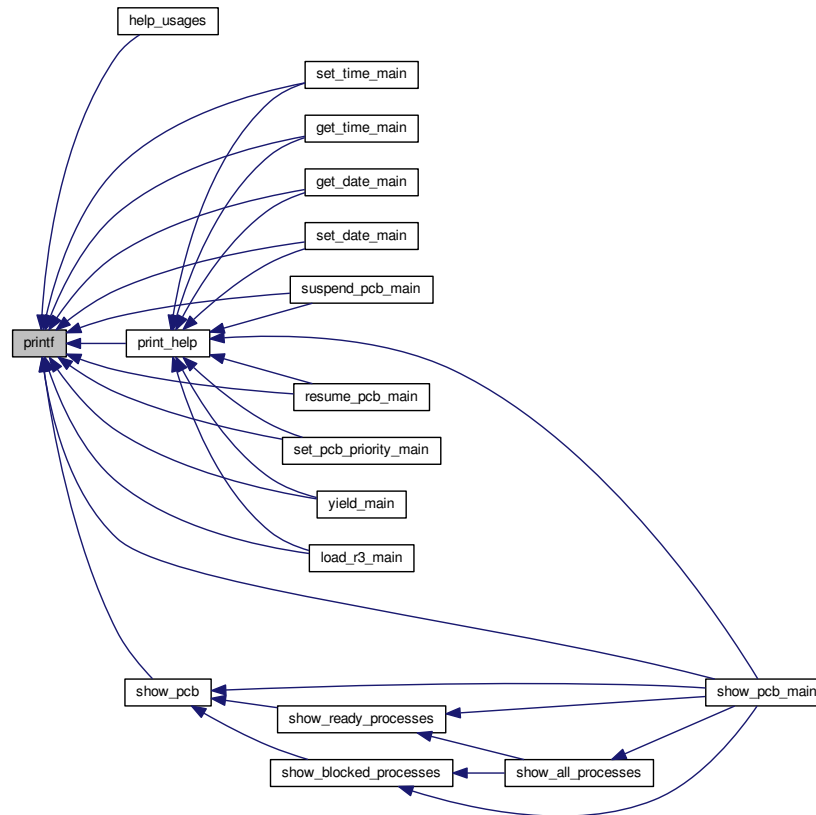


5.4.2.4 int printf (const char * format, ...)

Here is the call graph for this function:



Here is the caller graph for this function:

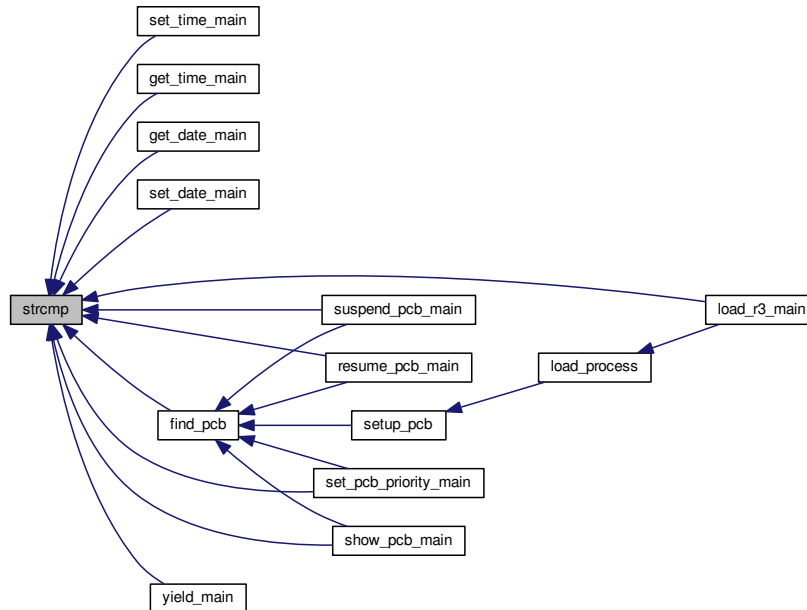


5.4.2.5 `int sprintf (char * str, const char * format, ...)`

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

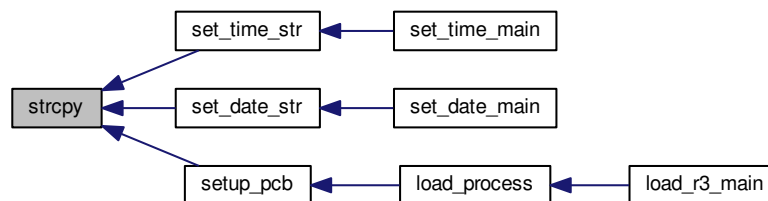
5.4.2.7 int strcmp (const char * s1, const char * s2)

Here is the caller graph for this function:



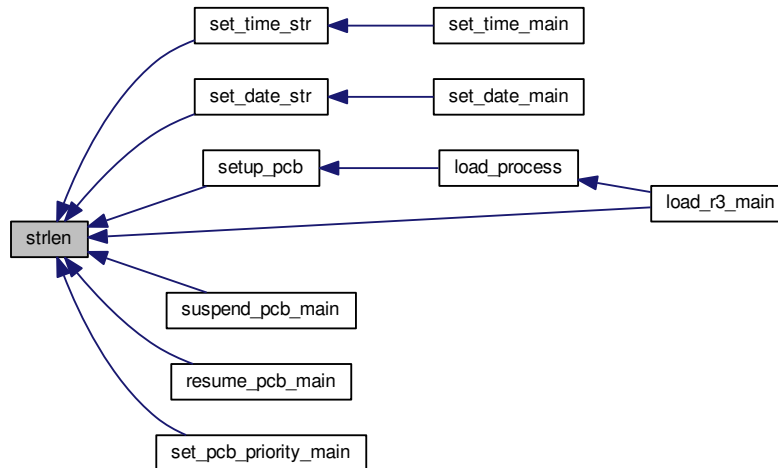
5.4.2.8 char* strcpy (char * s1, const char * s2)

Here is the caller graph for this function:



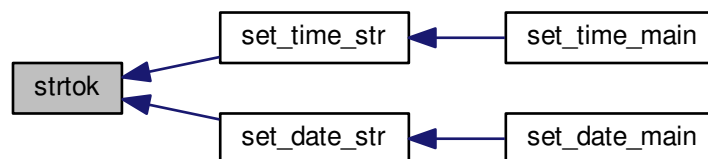
5.4.2.9 int strlen (const char * s)

Here is the caller graph for this function:



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

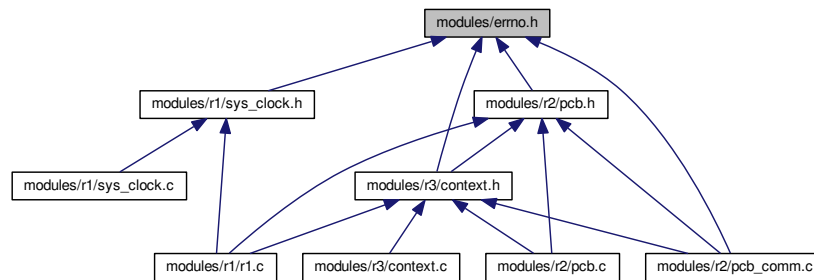
Here is the caller graph for this function:



5.5 modules/errno.h File Reference

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

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



Macros

- `#define E_NOERROR 0`
- `#define E_INVPARA 1`
- `#define E_INVSTRF 2`
- `#define E_INVUSRI 3`
- `#define E_FREEMEM 4`
Error we cannot actually free the memory space since the student_free had not been implemented before R5.
- `#define E_NULL_PTR 5`
A NULL Pointer Error.
- `#define E_EMPTPCB 6`
The pcb queue is empty.
- `#define E_PCB_SYS 7`
- `#define E_PROGERR 99`

Typedefs

`error_t.`

The datatype that holds the error code.

- `typedef unsigned int error_t`

5.5.1 Detailed Description

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

Author

Thunder Krakens

Date

February 7nd, 2016

Version

R2

5.5.2 Macro Definition Documentation

5.5.2.1 #define E_EMPTPCB 6

The pcb queue is empty.

5.5.2.2 #define E_FREEMEM 4

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

5.5.2.3 #define E_INVPARA 1

5.5.2.4 #define E_INVSTRF 2

5.5.2.5 #define E_INVUSRI 3

5.5.2.6 #define E_NOERROR 0

5.5.2.7 #define E_NULL_PTR 5

A NULL Pointer Error.

5.5.2.8 #define E_PCB_SYS 7

5.5.2.9 #define E_PROGERR 99

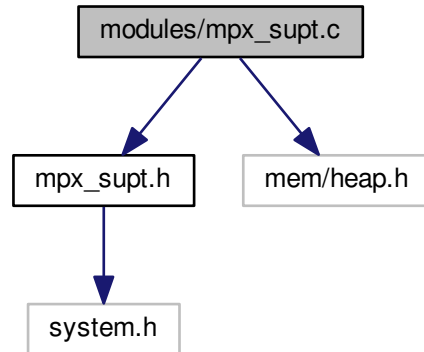
5.5.3 Typedef Documentation

5.5.3.1 typedef unsigned int error_t

5.6 modules/mpx_supt.c File Reference

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

Include dependency graph for mpx_supt.c:



Functions

- int [sys_req](#) (int op_code)
- void [mpx_init](#) (int cur_mod)
- void [sys_set_malloc](#) (u32int(*func)(u32int))
- void [sys_set_free](#) (int(*func)(void *))
- void * [sys_alloc_mem](#) (u32int size)
- int [sys_free_mem](#) (void *ptr)
- void [idle](#) ()
- int [get_op_code](#) ()

Variables

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

5.6.1 Function Documentation

5.6.1.1 int get_op_code ()

Here is the caller graph for this function:



5.6.1.2 void idle ()

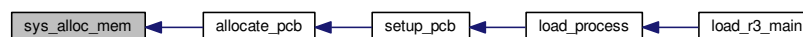
Here is the call graph for this function:



5.6.1.3 void mpx_init (int cur_mod)

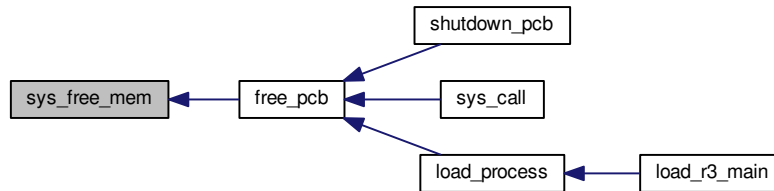
5.6.1.4 void* sys_alloc_mem (u32int size)

Here is the caller graph for this function:



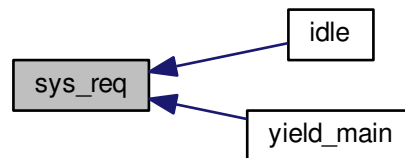
5.6.1.5 `int sys_free_mem (void * ptr)`

Here is the caller graph for this function:



5.6.1.6 `int sys_req (int op_code)`

Here is the caller graph for this function:



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

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

5.6.2 Variable Documentation

5.6.2.1 `int current_module = -1`

5.6.2.2 `param params`

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

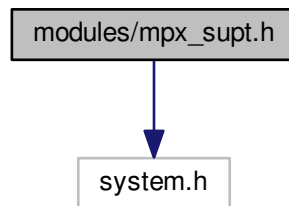
5.6.2.4 `u32int(* student_malloc)(u32int)`

5.7 modules/mpx_supt.h File Reference

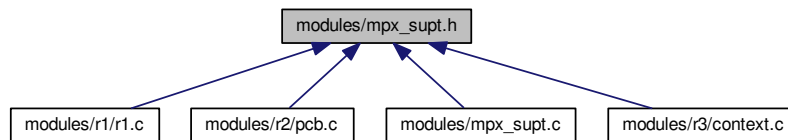
MPX System Supplementaries.

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

Include dependency graph for mpx_supt.h:



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



Data Structures

- struct [param](#)

A structure to represent interrupt.

Macros

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

Functions

sys_req

Generate interrupt 60H

Parameters

int	<i>op_code (IDLE)</i>
-----	-----------------------

- int [sys_req](#) (int op_code)

mpx_init

Initialize MPX support software

Parameters

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

- void [mpx_init](#) (int cur_mod)

set_malloc

Sets the memory allocation function for sys_alloc_mem

Parameters

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

- void [sys_set_malloc](#) (u32int>(*func)(u32int))

set_free

Sets the memory free function for sys_free_mem

Parameters

s1- destination,s2- source	
----------------------------------	--

- void [sys_set_free](#) (int(*func)(void *))

sys_alloc_mem

Allocates a block of memory (similar to malloc)

Parameters

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

- void * [sys_alloc_mem](#) (u32int size)

sys_free_mem

Frees memory

Parameters

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

- int `sys_free_mem` (void *ptr)

idle

The idle process

Parameters

None	
------	--

- void `idle` ()

get_op_code

Returns the interrupt's operation code

Parameters

None	
------	--

- int `get_op_code` ()

Variables

- typedef `__attribute__`

5.7.1 Detailed Description

MPX System Supplementaries.

Author

Thunder Krakens

Date

March 18, 2016

Version

R3

5.7.2 Macro Definition Documentation

5.7.2.1 `#define EXIT 0`

5.7.2.2 `#define IDLE 1`

5.7.2.3 `#define MODULE_R1 0`

5.7.2.4 `#define MODULE_R2 1`

5.7.2.5 `#define MODULE_R3 2`

5.7.2.6 `#define MODULE_R4 4`

5.7.2.7 `#define MODULE_R5 8`

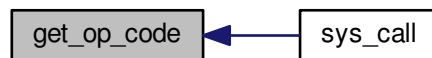
5.7.2.8 `#define READ 2`

5.7.2.9 `#define WRITE 3`

5.7.3 Function Documentation

5.7.3.1 `int get_op_code ()`

Here is the caller graph for this function:



5.7.3.2 `void idle ()`

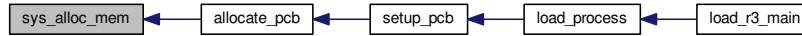
Here is the call graph for this function:



5.7.3.3 `void mpx_init (int cur_mod)`

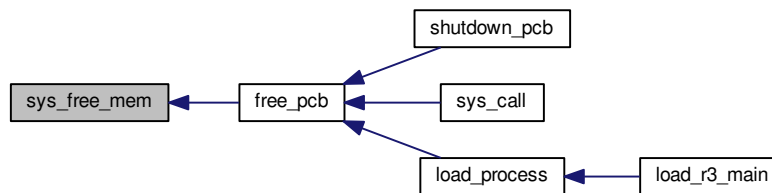
5.7.3.4 void* sys_alloc_mem (u32int size)

Here is the caller graph for this function:



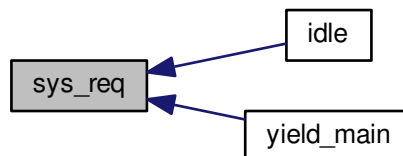
5.7.3.5 int sys_free_mem (void * ptr)

Here is the caller graph for this function:



5.7.3.6 int sys_req (int op_code)

Here is the caller graph for this function:



5.7.3.7 void sys_set_free (int(*) (void *) func)

5.7.3.8 void sys_set_malloc (u32int(*) (u32int) func)

5.7.4 Variable Documentation

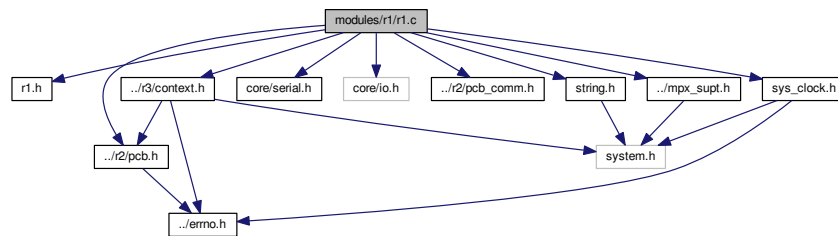
5.7.4.1 enum process_suspended __attribute__

5.8 modules/r1/r1.c File Reference

The commandhandler and functions associations for Module R1.

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

Include dependency graph for r1.c:



Data Structures

- struct [function_name](#)
A structure to represent each function.

Macros

- #define [MAX_ARGC](#) 50
- #define [MOD_VERSION](#) "R4"
- #define [COMPLETION](#) "03/18/2016"
- #define [MAX_HISTORY](#) 10

Functions

exe_function.

Executes the specific function.

Parameters

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

Returns

0

version

displays the version of the system currently running.

Parameters

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

Returns

0

shutdown

Closes all functions, and shuts down the system.

Parameters

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

Returns

0 for shutdown, 1 for keep running.

help_usages

shows usage message for each function.

Parameters

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

Returns

0

- int [help_usages](#) (enum [comm_type](#) type)

help_function

displays help text for all functions.

Parameters

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

Returns

0

commhand

Accepts and handles commands from the user.

Returns

0

- void [commhand](#) ()

command_line_parser

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

Parameters

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

Returns

void

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

print_help

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

Parameters

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

Returns

void

- void [print_help](#) (const int function_index)

Variables

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

CommandParserStat

The status of the command parser

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

5.8.1 Detailed Description

The commandhandler and functions associations for Module R1.

Author

Thunder Krakens

Date

February 2nd, 2016

Version

R4

5.8.2 Macro Definition Documentation

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

5.8.2.2 `#define MAX_ARGC 50`

5.8.2.3 `#define MAX_HISTORY 10`

5.8.2.4 `#define MOD_VERSION "R4"`

5.8.3 Enumeration Type Documentation

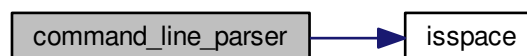
5.8.3.1 `enum CommandPaserStat`

5.8.4 Function Documentation

5.8.4.1 `enum CommandPaserStat __attribute__((packed))`

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

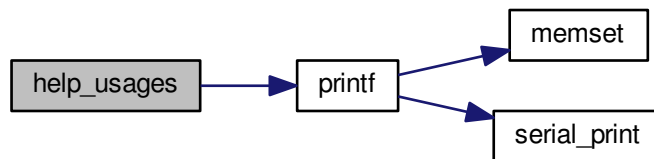
Here is the call graph for this function:



5.8.4.3 void commhand ()

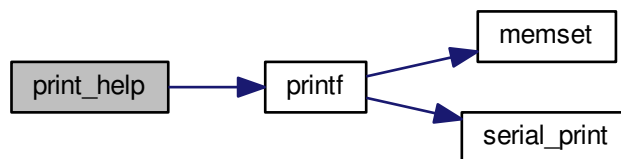
5.8.4.4 int help_usages (enum comm_type type)

Here is the call graph for this function:

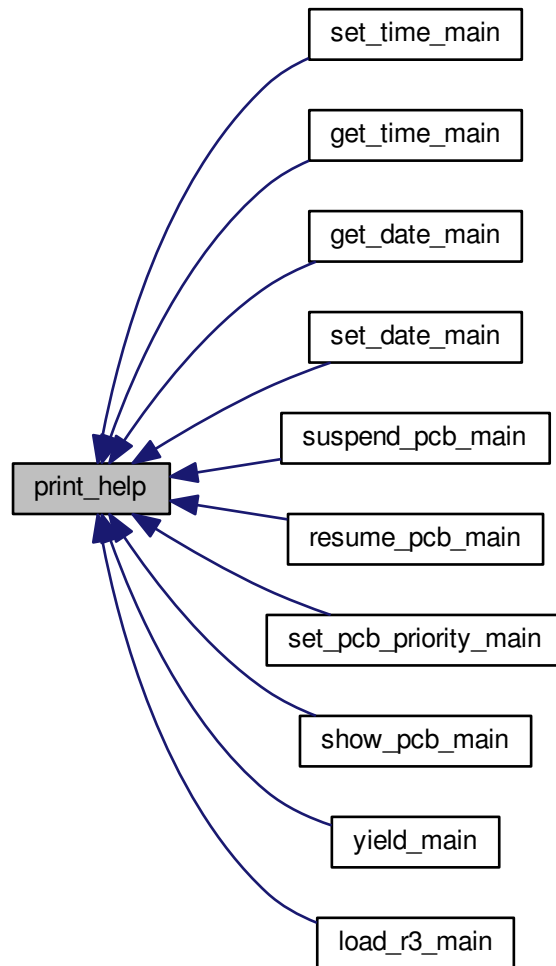


5.8.4.5 void print_help (const int *function_index*)

Here is the call graph for this function:



Here is the caller graph for this function:



5.8.5 Variable Documentation

5.8.5.1 DoubleQuoteWriting

5.8.5.2 NormalWriting

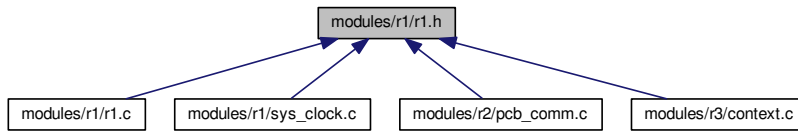
5.8.5.3 NotWriting

5.8.5.4 SingleQuoteWriting

5.9 modules/r1/r1.h File Reference

The command handler and functions associations for Module R1.

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



Macros

- #define [HELP](#) 0
- #define [POS_OF_MPX](#) 1
- #define [VERSION](#) 1
- #define [GETTIME](#) 2
- #define [SETTIME](#) 3
- #define [GETDATE](#) 4
- #define [SETDATE](#) 5
- #define [SHUTDOWN](#) 6
- #define [YIELD](#) 7
- #define [LOADR3](#) 8
- #define [NUM_MPX_FUNCTIONS](#) 9
- #define [POS_OF_PCB](#) 9
- #define [SUSPDPCB](#) 9
- #define [RESUMEPCB](#) 10
- #define [SETPCBPRIO](#) 11
- #define [SHOWPCB](#) 12
- #define [NUM_OF_FUNCTIONS](#) 13

Enumerations

- enum [comm_type](#)

Functions

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

commhand

Accepts and handles commands from the user.

*Returns**VOID*

- void `commhand` ()

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

Parameters

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

Returns

void

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

print_help

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

Parameters

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

Returns

void

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

Variables

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

5.9.1 Detailed Description

The command handler and functions associations for Module R1.

Author

Thunder Krakens

Date

March 17, 2016

Version

R3 & R4

5.9.2 Macro Definition Documentation

5.9.2.1 `#define GETDATE 4`

5.9.2.2 `#define GETTIME 2`

5.9.2.3 `#define HELP 0`

5.9.2.4 `#define LOADR3 8`

5.9.2.5 `#define NUM_MPX_FUNCTIONS 9`

5.9.2.6 `#define NUM_OF_FUNCTIONS 13`

5.9.2.7 `#define POS_OF_MPX 1`

5.9.2.8 `#define POS_OF_PCB 9`

5.9.2.9 `#define RESUMEPCB 10`

5.9.2.10 `#define SETDATE 5`

5.9.2.11 `#define SETPCBPRI 11`

5.9.2.12 `#define SETTIME 3`

5.9.2.13 `#define SHOWPCB 12`

5.9.2.14 `#define SHUTDOWN 6`

5.9.2.15 `#define SUSPDPCB 9`

5.9.2.16 `#define VERSION 1`

5.9.2.17 `#define YIELD 7`

5.9.3 Enumeration Type Documentation

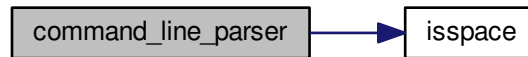
5.9.3.1 `enum comm_type`

5.9.4 Function Documentation

5.9.4.1 `enum comm_type __attribute__((packed))`

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

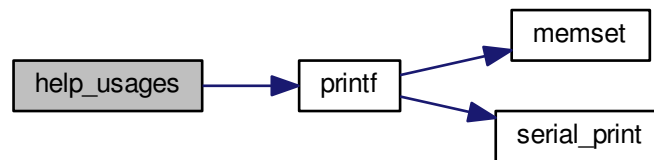
Here is the call graph for this function:



5.9.4.3 void `commhand` ()

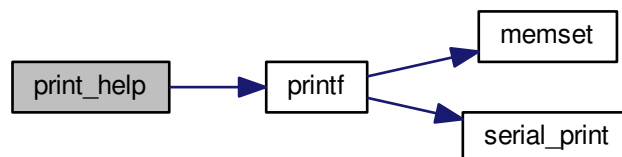
5.9.4.4 int `help_usages` (enum `comm_type` *type*)

Here is the call graph for this function:

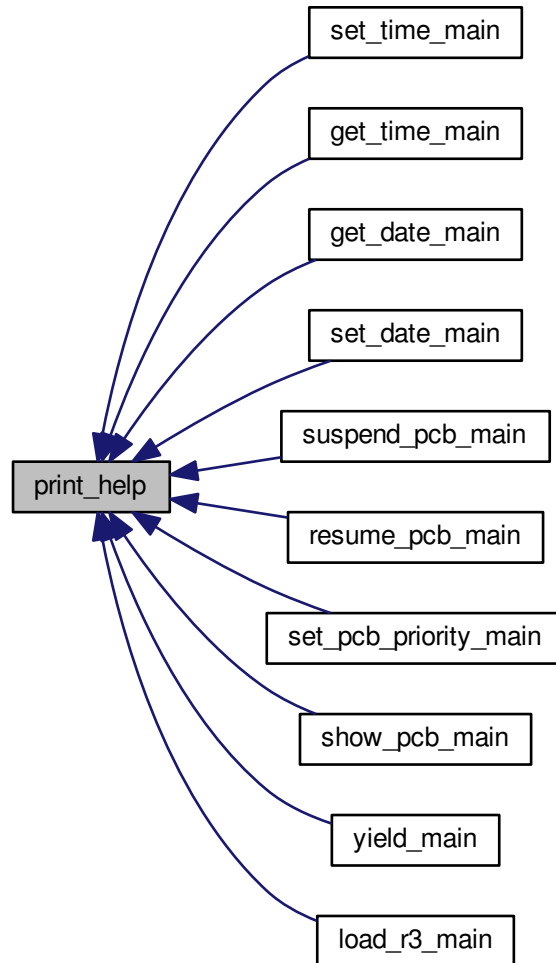


5.9.4.5 void `print_help` (const int *function_index*)

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.5 Variable Documentation

5.9.5.1 help

5.9.5.2 mpx

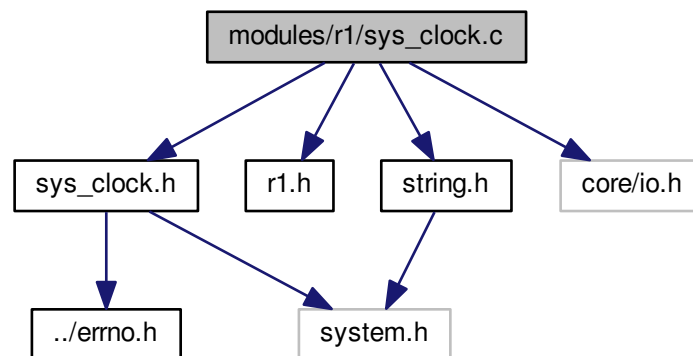
5.9.5.3 pcb

5.10 modules/r1/sys_clock.c File Reference

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

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

Include dependency graph for sys_clock.c:



Macros

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

Functions

set_time_main.

Sets the time for the system.

Parameters

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

Returns

0

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

get_time_main.

Retrieves system's current time.

Parameters

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

Returns

0

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

is_digit

determines if a character represents a digit.

Parameters

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

Returns

1 if it is digit, otherwise returns 0.

set_time_str.

Sets the time for the system by string.

Parameters

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

Returns

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

- [error_t set_time_str](#) (const char *timeStr)

get_time.

Retrieves system's current time and date.

Parameters

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

Returns

VOID

- void [get_time](#) (date_time *dateTimeValues)

set_time.

Sets the time for the system by date_time struct.

Parameters

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

Returns

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

- [error_t set_time](#) (const date_time *dateTimeValues)

get_date.

Retrieves system's current date.

Parameters

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

Returns

VOID

- void [get_date](#) (date_time *dateTimeValues)

is_date_value_valid.

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

Parameters

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

Returns

VOID

set_date.

Sets the date of the system.

Parameters

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

Returns

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

- [error_t set_date](#) (const date_time *dateTimeValues)

get_date_main.

Retrieves system's current date.

Parameters

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

Returns

0

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

set_date_str.

Sets the date for the system by string.

Parameters

str	<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)

5.10.1 Detailed Description

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

Author

Thunder Krakens

Date

February 2nd, 2016

Version

R1

5.10.2 Macro Definition Documentation

5.10.2.1 `#define RTC_INDEX_DAY_MONTH 0x07`

5.10.2.2 `#define RTC_INDEX_DAY_WEEK 0x06`

5.10.2.3 `#define RTC_INDEX_HOUR 0x04`

5.10.2.4 `#define RTC_INDEX_HOUR_ALARM 0x05`

5.10.2.5 `#define RTC_INDEX_MINUTE 0x02`

5.10.2.6 `#define RTC_INDEX_MINUTE_ALARM 0x03`

5.10.2.7 `#define RTC_INDEX_MONTH 0x08`

5.10.2.8 `#define RTC_INDEX_SECOND 0x00`

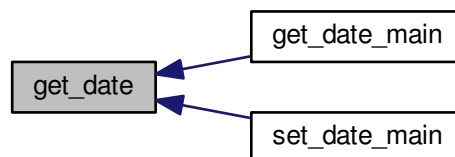
5.10.2.9 `#define RTC_INDEX_SECOND_ALARM 0x01`

5.10.2.10 `#define RTC_INDEX_YEAR 0x09`

5.10.3 Function Documentation

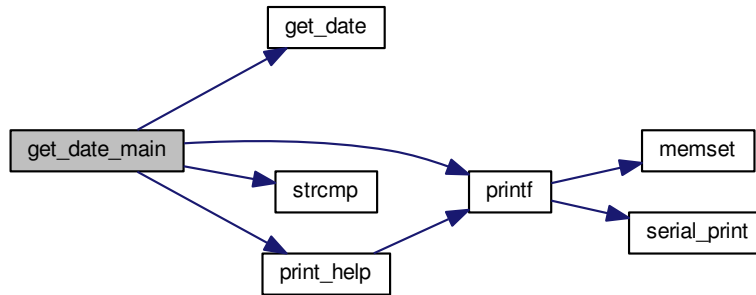
5.10.3.1 `void get_date (date_time * dateTimeValues)`

Here is the caller graph for this function:



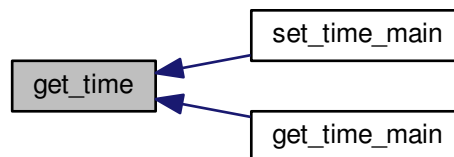
5.10.3.2 int get_date_main (int argc, char ** argv)

Here is the call graph for this function:



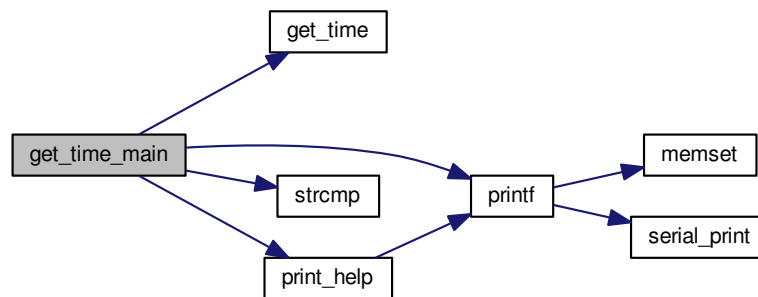
5.10.3.3 void get_time (date_time * dateTimeValues)

Here is the caller graph for this function:



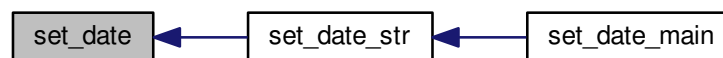
5.10.3.4 `int get_time_main (int argc, char ** argv)`

Here is the call graph for this function:



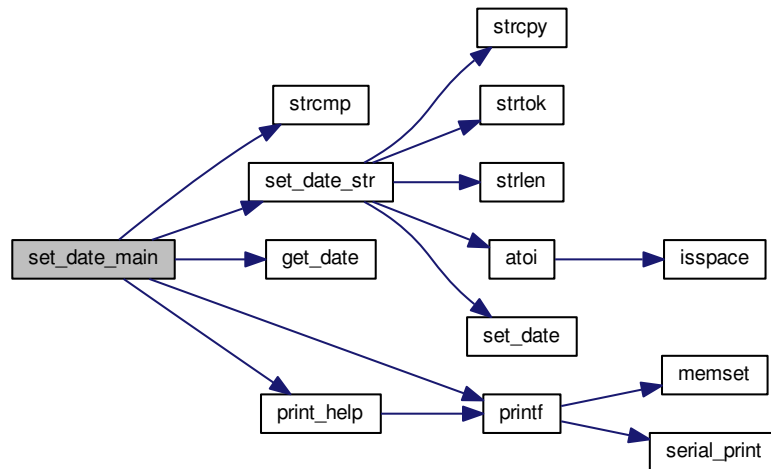
5.10.3.5 `error_t set_date (const date_time * dateTimeValues)`

Here is the caller graph for this function:



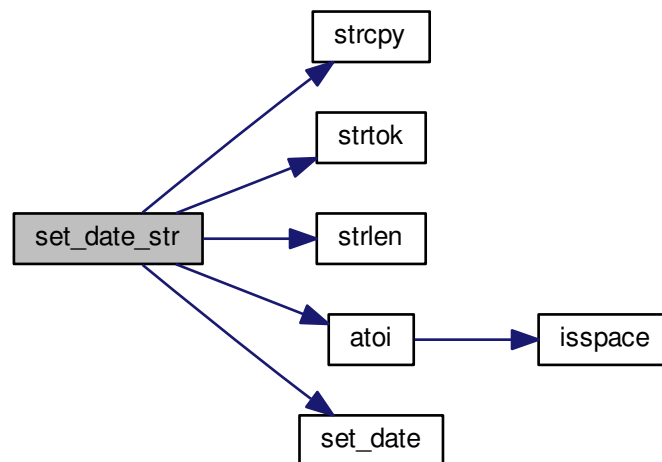
5.10.3.6 int set_date_main (int argc, char ** argv)

Here is the call graph for this function:



5.10.3.7 int set_date_str (const char * str)

Here is the call graph for this function:

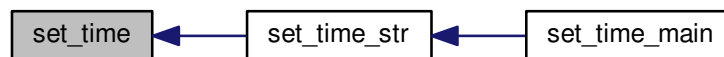


Here is the caller graph for this function:



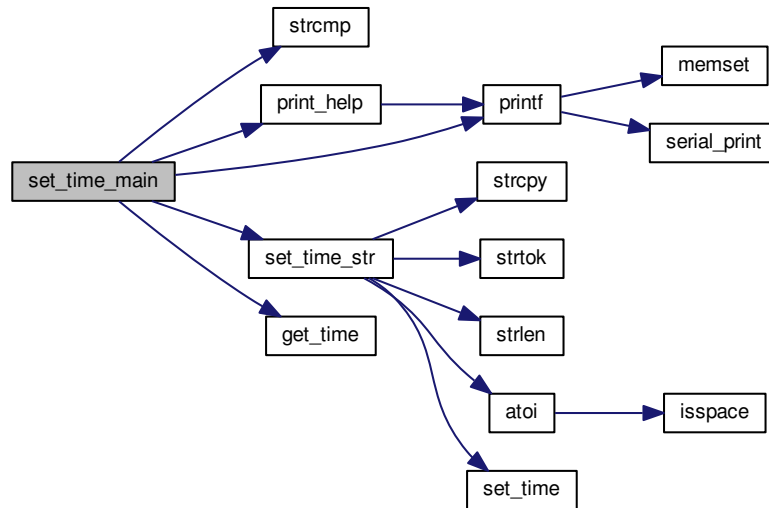
5.10.3.8 `error_t set_time (const date_time * dateTimeValues)`

Here is the caller graph for this function:



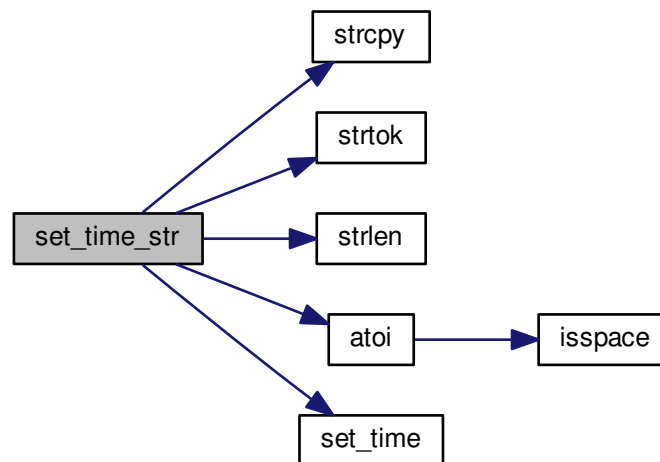
5.10.3.9 int set_time_main (int argc, char ** argv)

Here is the call graph for this function:



5.10.3.10 error_t set_time_str (const char * timeStr)

Here is the call graph for this function:



Here is the caller graph for this function:

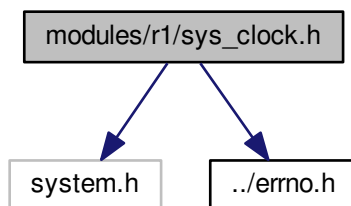


5.11 modules/r1/sys_clock.h File Reference

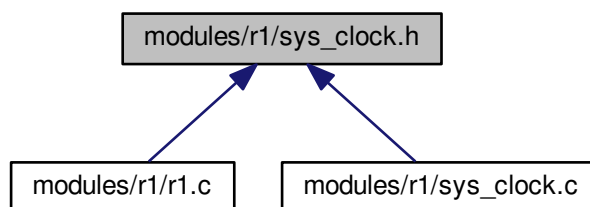
The main file that manipulates and controls the system's clock.

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

Include dependency graph for `sys_clock.h`:



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



Functions

set_time_main.

Sets the time for the system.

Parameters

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

Returns

0

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

get_time_main.

Retrieves system's current time.

Parameters

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

Returns

0

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

set_time_str.

Sets the time for the system by string.

Parameters

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

Returns

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

- [error_t set_time_str](#) (const char *timeStr)

get_time.

Retrieves system's current time and date.

Parameters

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

Returns

VOID

- void [get_time](#) (date_time *dateTimeValues)

set_time.

Sets the time for the system by date_time struct.

Parameters

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

Returns

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

- [error_t set_time](#) (const date_time *dateTimeValues)

set_date_main.

Sets system's date.

Parameters

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

Returns

0

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

get_date_main.

Retrieves system's current date.

Parameters

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

Returns

0

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

get_date.

Retrieves system's current date.

Parameters

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

Returns

VOID

- void [get_date](#) (date_time *dateTimeValues)

set_date_str.

Sets the date for the system by string.

Parameters

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

Returns

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

- int [set_date_str](#) (const char *str)

set_date.

Sets the date of the system.

Parameters

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

Returns

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

- [error_t set_date](#) (const date_time *dateTimeValues)

5.11.1 Detailed Description

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

Author

Thunder Krakens

Date

February 2nd, 2016

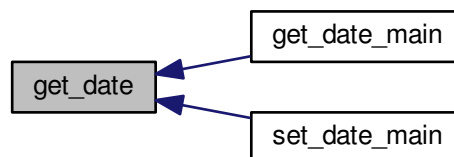
Version

R1

5.11.2 Function Documentation

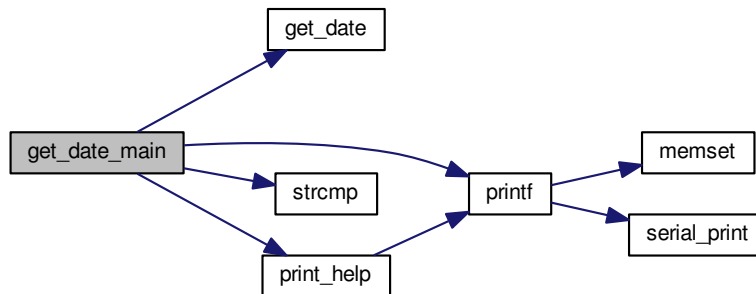
5.11.2.1 void [get_date](#) (date_time * *dateTimeValues*)

Here is the caller graph for this function:



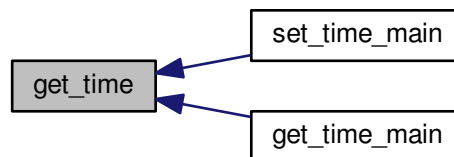
5.11.2.2 `int get_date_main (int argc, char ** argv)`

Here is the call graph for this function:



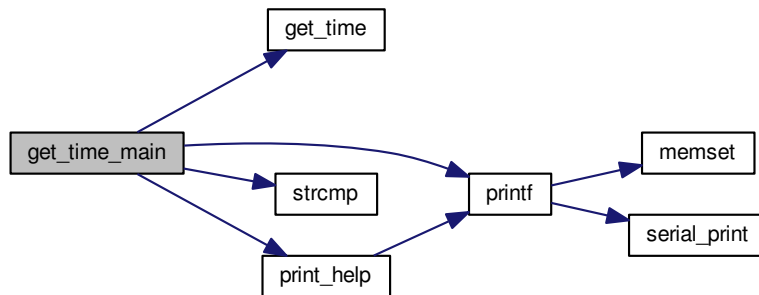
5.11.2.3 `void get_time (date_time * dateTimeValues)`

Here is the caller graph for this function:



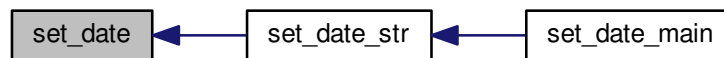
5.11.2.4 `int get_time_main (int argc, char ** argv)`

Here is the call graph for this function:



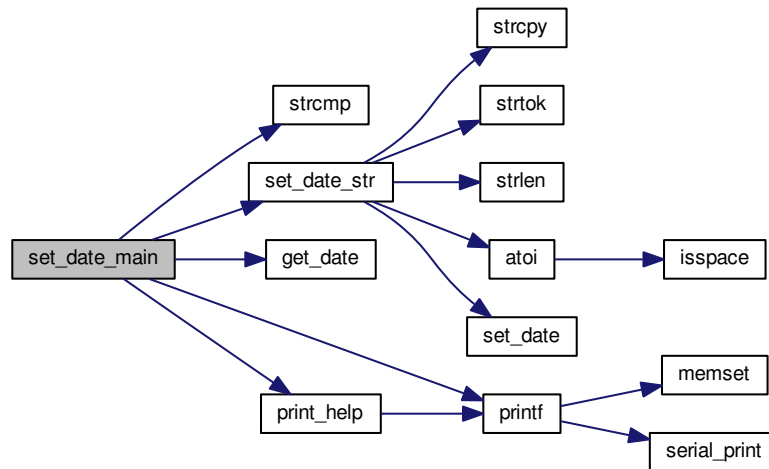
5.11.2.5 `error_t set_date (const date_time * dateTimeValues)`

Here is the caller graph for this function:



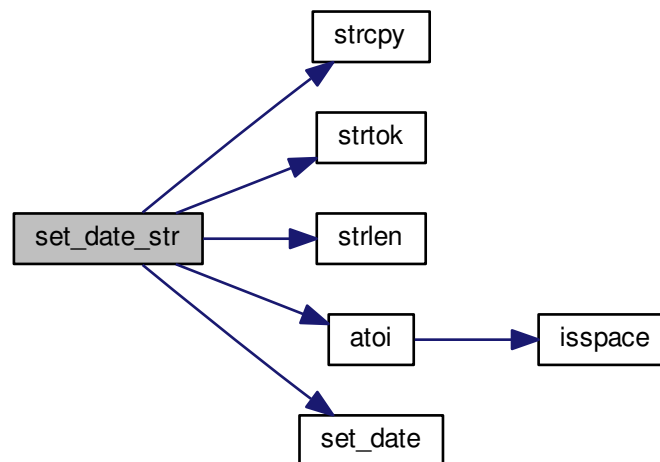
5.11.2.6 `int set_date_main (int argc, char ** argv)`

Here is the call graph for this function:



5.11.2.7 `int set_date_str (const char * str)`

Here is the call graph for this function:

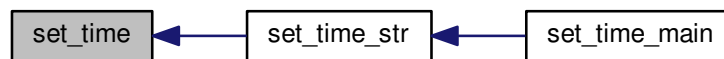


Here is the caller graph for this function:



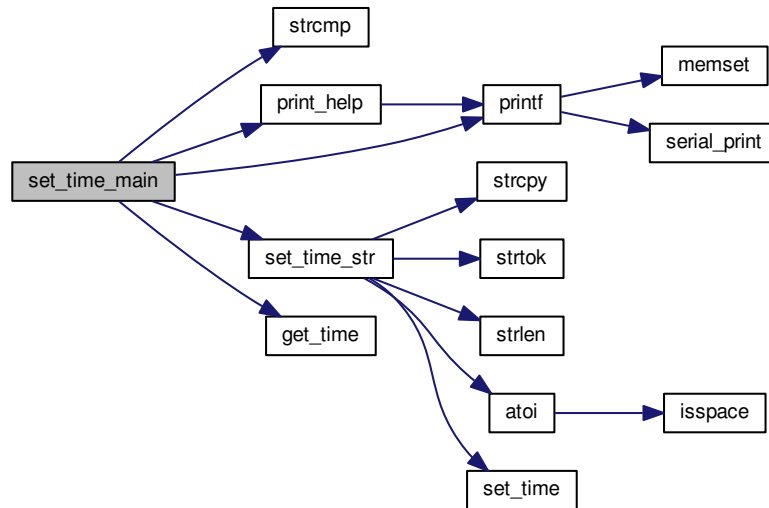
5.11.2.8 `error_t set_time (const date_time * dateTimeValues)`

Here is the caller graph for this function:



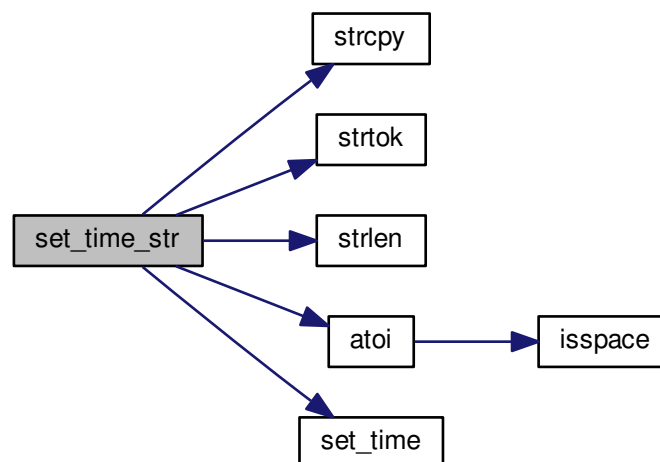
5.11.2.9 `int set_time_main (int argc, char ** argv)`

Here is the call graph for this function:



5.11.2.10 `error_t set_time_str (const char * timeStr)`

Here is the call graph for this function:



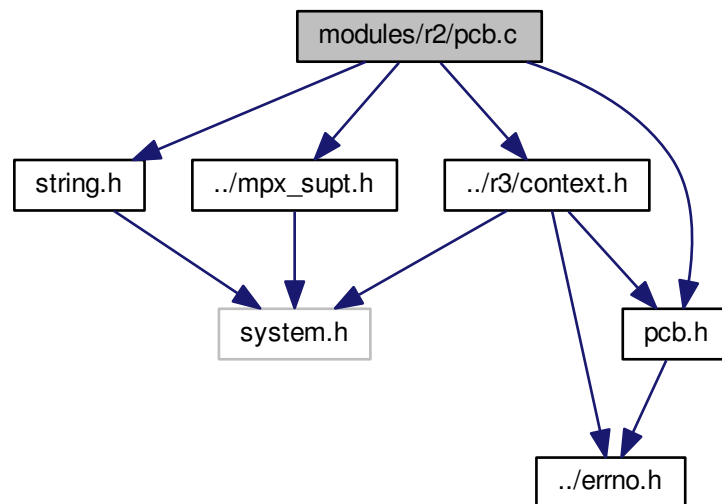
Here is the caller graph for this function:



5.12 modules/r2/pcb.c File Reference

The Process Control Block.

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



Data Structures

- struct [pcb_struct](#)
Struct that will describe PCB Processes.
- struct [pcb_queue](#)

Queue structure that will store PCBs.

Enumerations

- enum `process_state`
PCB process states/statuses.
- enum `process_suspended`
PCB process suspended or not suspended status.

Functions

- enum `process_state __attribute__((packed))`

pcb_init

Initiates the PCB queues

- void `pcb_init ()`

suspend_pcb

Suspends the specific PCB.

Parameters

<code>pcb_ptr</code>	<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

<code>pcb_ptr</code>	<i>The pointer to the PCB</i>
----------------------	-------------------------------

Returns

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

- `error_t resume_pcb (struct pcb_struct *pcb_ptr)`

allocate_pcb

allocate a space for the PCB structure.

Returns

The pointer that point to the PCB structure.

- `struct pcb_struct * allocate_pcb ()`

setup_pcb

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

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

Parameters

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

Returns

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

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

free_pcb

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

Parameters

pcb_ptr	The pointer to the PCB
---------	------------------------

Returns

The error code. Possible error code to be returned: *E_NOERROR* No error. *E_INVPARA* The PCB probably had not been removed from queue before free it. *E_FREEMEM* The memory space cannot be actually free, since the *student_free* had not been implemented yet.

- [error_t](#) [free_pcb](#) (struct [pcb_struct](#) *pcb_ptr)

find_pcb

Will search all queues for a process named pName

Parameters

pName	The char pointer to the desired searched name
-------	---

Returns

PCB pointer if found, NULL if PCB is not found

- struct [pcb_struct](#) * [find_pcb](#) (const char *pName)

insert_pcb

Inserts PCB into the appropriate queue.

Parameters

pcb_ptr	The pointer to the PCB
---------	------------------------

Returns

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

- [error_t](#) [insert_pcb](#) (struct [pcb_struct](#) *pcb_ptr)

remove_pcb

Removes PCB from the queue it is currently in.

Parameters

pcb_ptr	The pointer to the PCB
---------	------------------------

Returns

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

- [error_t remove_pcb](#) (struct [pcb_struct](#) *pcb_ptr)

show_pcb

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

Parameters

pName	The PCB pointer.
-------	------------------

Returns

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

- [error_t show_pcb](#) (struct [pcb_struct](#) *pcb_ptr)

show_blocked_processes

displays all blocked processes and their attributes

Returns

VOID.

- void [show_blocked_processes](#) ()

show_ready_processes

Displays all of the ready processes and their attributes.

Returns

VOID.

- void [show_ready_processes](#) ()

show_all_processes

Displays all of the processes and their attributes.

Returns

VOID.

- void [show_all_processes](#) ()

block_pcb

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

Parameters

pcb_ptr	The pointer to the PCB
---------	------------------------

Returns

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

- [error_t block_pcb](#) (struct [pcb_struct](#) *pcb_ptr)

unblock_pcb

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

Parameters

pcb_ptr	The pointer to the PCB
---------	------------------------

Returns

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

- [error_t unblock_pcb](#) (struct [pcb_struct](#) *pcb_ptr)

set_pcb_priority

Sets the priority of the selected PCB

Parameters

pcb_ptr	The PCB pointer.
pPriority	The assigned priority

Returns

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

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

get_running_process

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

Parameters

None	
------	--

Returns

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

- struct [pcb_struct](#) * [get_running_process](#) ()

save_running_process

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

Parameters

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

Returns

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

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

get_stack_top

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

Parameters

pcb_ptr	The pointer to the PCB.
---------	-------------------------

Returns

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

- unsigned char * [get_stack_top](#) (struct [pcb_struct](#) *pcb_ptr)

get_stack_base

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

Parameters

pcb_ptr	The pointer to the PCB.
---------	-------------------------

Returns

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

- unsigned char * [get_stack_base](#) (struct [pcb_struct](#) *pcb_ptr)

shutdown_pcb

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

Returns

VOID

- void [shutdown_pcb](#) ()

Variables

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

PCB process is suspended.

- `false`

< PCB process is not suspended.

- struct `pcb_struct` `__attribute__`

5.12.1 Detailed Description

The Process Control Block.

Author

Thunder Krakens

Date

March 18th, 2016

Version

R3

5.12.2 Enumeration Type Documentation

5.12.2.1 enum `process_state`

PCB process states/statuses.

5.12.2.2 enum `process_suspended`

PCB process suspended or not suspended status.

5.12.3 Function Documentation

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

5.12.3.2 struct `pcb_struct`* `allocate_pcb` ()

Here is the call graph for this function:

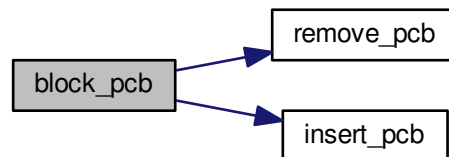


Here is the caller graph for this function:



5.12.3.3 `error_t block_pcb (struct pcb_struct * pcb_ptr)`

Here is the call graph for this function:

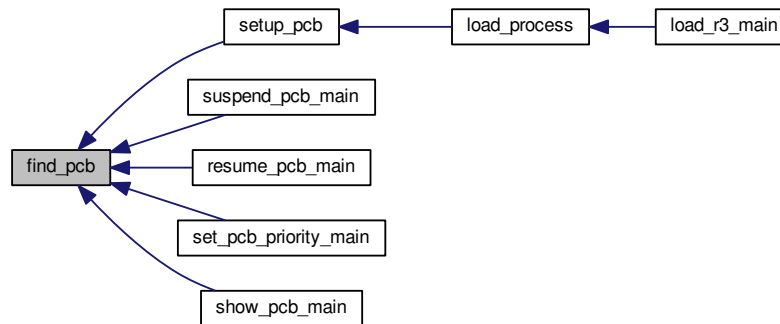


5.12.3.4 `struct pcb_struct* find_pcb (const char * pName)`

Here is the call graph for this function:



Here is the caller graph for this function:

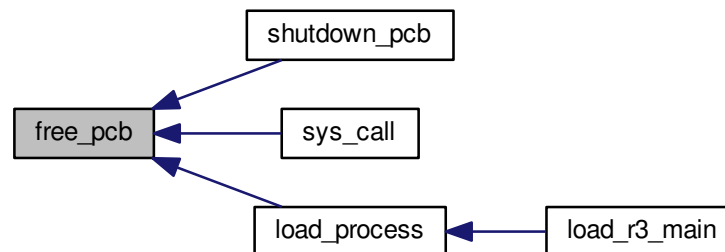


5.12.3.5 `error_t free_pcb (struct pcb_struct * pcb_ptr)`

Here is the call graph for this function:



Here is the caller graph for this function:

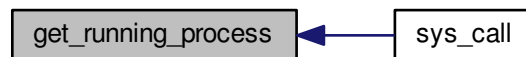


5.12.3.6 `struct pcb_struct* get_running_process ()`

Here is the call graph for this function:

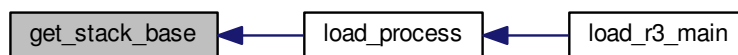


Here is the caller graph for this function:



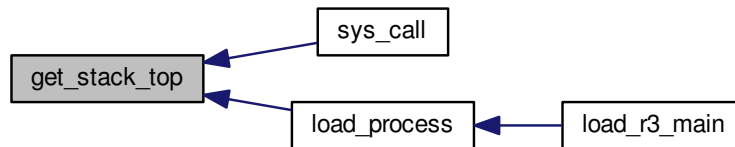
5.12.3.7 `unsigned char* get_stack_base (struct pcb_struct * pcb_ptr)`

Here is the caller graph for this function:



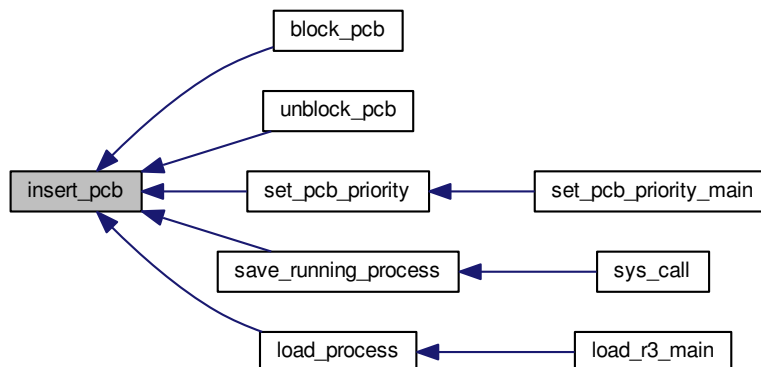
5.12.3.8 unsigned char* get_stack_top (struct pcb_struct * pcb_ptr)

Here is the caller graph for this function:



5.12.3.9 error_t insert_pcb (struct pcb_struct * pcb_ptr)

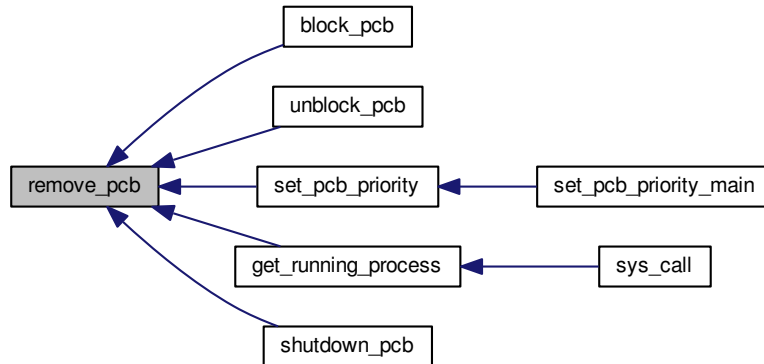
Here is the caller graph for this function:



5.12.3.10 void pcb_init ()

5.12.3.11 `error_t remove_pcb (struct pcb_struct * pcb_ptr)`

Here is the caller graph for this function:



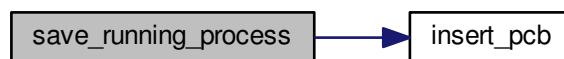
5.12.3.12 `error_t resume_pcb (struct pcb_struct * pcb_ptr)`

Here is the caller graph for this function:

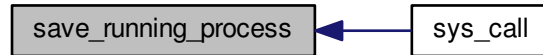


5.12.3.13 `error_t save_running_process (struct pcb_struct * pcb_ptr, struct context * new_stack_top)`

Here is the call graph for this function:

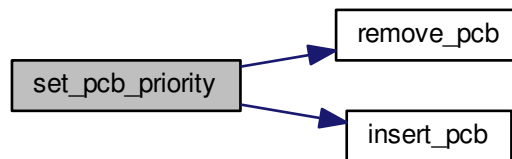


Here is the caller graph for this function:

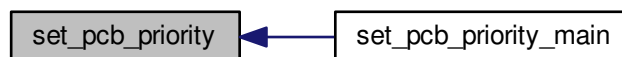


5.12.3.14 `error_t set_pcb_priority (struct pcb_struct * pcb_ptr, const unsigned char pPriority)`

Here is the call graph for this function:

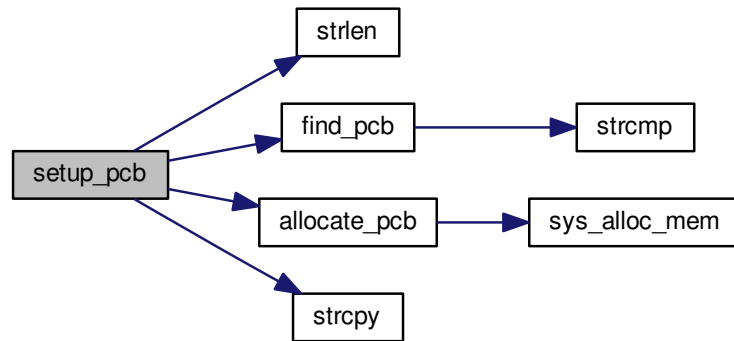


Here is the caller graph for this function:

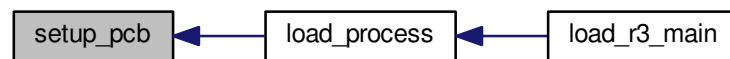


5.12.3.15 `struct pcb_struct* setup_pcb (const char * pName, const enum process_class pClass, const unsigned char pPriority)`

Here is the call graph for this function:

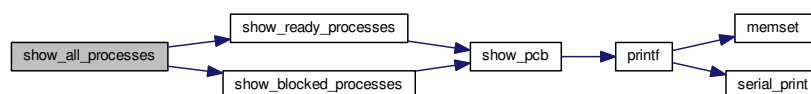


Here is the caller graph for this function:



5.12.3.16 `void show_all_processes ()`

Here is the call graph for this function:

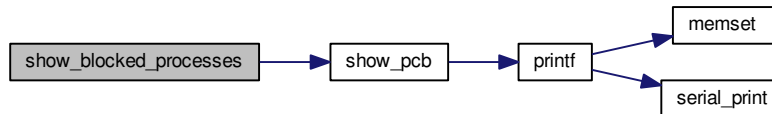


Here is the caller graph for this function:

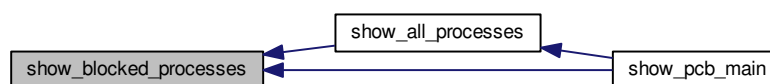


5.12.3.17 void show_blocked_processes ()

Here is the call graph for this function:

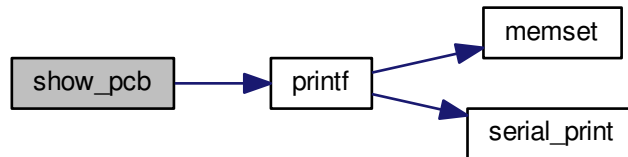


Here is the caller graph for this function:

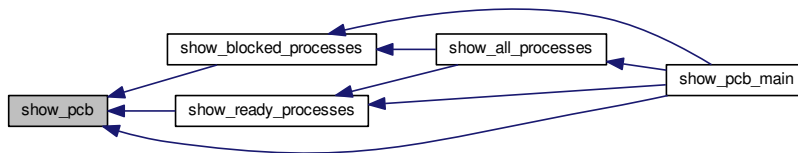


5.12.3.18 `error_t show_pcb (struct pcb_struct * pcb_ptr)`

Here is the call graph for this function:

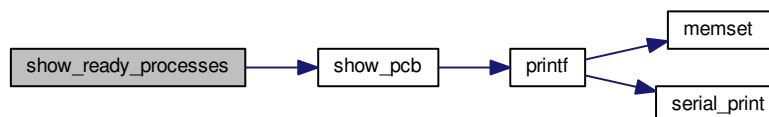


Here is the caller graph for this function:

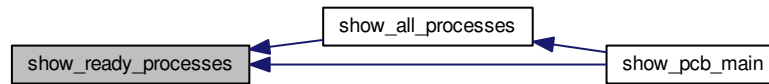


5.12.3.19 `void show_ready_processes ()`

Here is the call graph for this function:

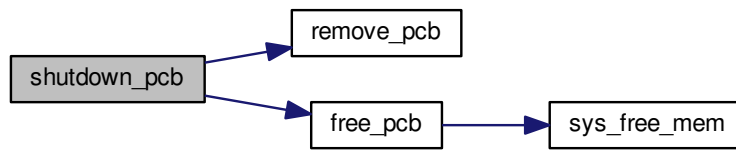


Here is the caller graph for this function:



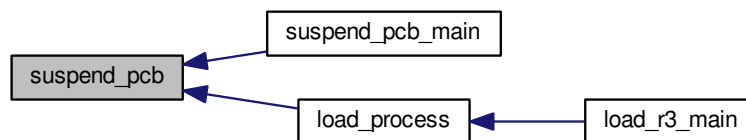
5.12.3.20 void shutdown_pcb ()

Here is the call graph for this function:



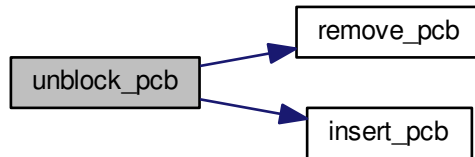
5.12.3.21 error_t suspend_pcb (struct pcb_struct * pcb_ptr)

Here is the caller graph for this function:



5.12.3.22 `error_t unblock_pcb (struct pcb_struct * pcb_ptr)`

Here is the call graph for this function:



5.12.4 Variable Documentation

5.12.4.1 `struct pcb_struct __attribute__`

5.12.4.2 `blocked`

< PCB in the blocked state.

PCB in the blocked state.

5.12.4.3 `false`

< PCB process is not suspended.

PCB process is not suspended.

5.12.4.4 `ready`

PCB in the ready state.

5.12.4.5 `running`

PCB in the running state.

5.12.4.6 `true`

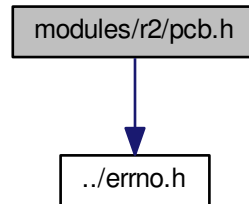
PCB process is suspended.

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

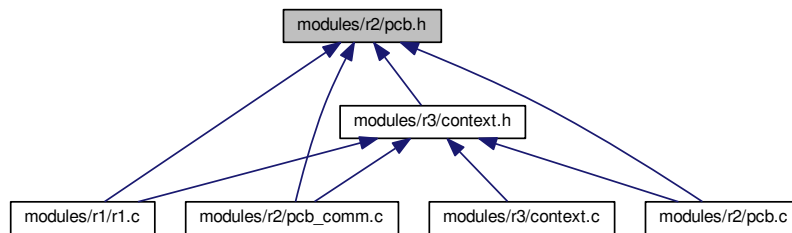
The Process Control Block.

```
#include "../errno.h"
```

Include dependency graph for pcb.h:



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



Macros

- #define `SIZE_OF_STACK` 1024
- #define `SIZE_OF_PCB_NAME` 10

Enumerations

- enum `process_class`
PCB process class types.

Functions

- enum `process_class __attribute__((packed))`

`pcb_init`

Initiates the PCB queues

- void [pcb_init](#) ()

allocate_pcb

allocate a space for the PCB structure.

Returns

The pointer that point to the PCB structure.

- struct [pcb_struct](#) * [allocate_pcb](#) ()

free_pcb

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

Parameters

pcb_ptr	<i>The pointer to the PCB</i>
-------------------------	-------------------------------

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	<i>Process Name (length < 10).</i>
pClass	<i>Process class (system or application).</i>
pPriority	<i>Process priority (0 ~ 9).</i>

Returns

NULL if error ocured, 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	<i>The char pointer to the desired searched name</i>
-----------------------	--

Returns

PCB pointer if found, NULL if PCB is not found

- struct [pcb_struct](#) * [find_pcb](#) (const char *[pName](#))

insert_pcb

Inserts PCB into the appropriate queue.

Parameters

pcb_ptr	The pointer to the PCB
---------	------------------------

Returns

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

- [error_t insert_pcb](#) (struct [pcb_struct](#) *pcb_ptr)

remove_pcb

Removes PCB from the queue it is currently in.

Parameters

pcb_ptr	The pointer to the PCB
---------	------------------------

Returns

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

- [error_t remove_pcb](#) (struct [pcb_struct](#) *pcb_ptr)

suspend_pcb

Suspends the specific PCB.

Parameters

pcb_ptr	The pointer to the PCB
---------	------------------------

Returns

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

- [error_t suspend_pcb](#) (struct [pcb_struct](#) *pcb_ptr)

resume_pcb

Resumes the specific PCB.

Parameters

pcb_ptr	The pointer to the PCB
---------	------------------------

Returns

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

- [error_t resume_pcb](#) (struct [pcb_struct](#) *pcb_ptr)

set_pcb_priority

Sets the priority of the selected PCB

Parameters

pcb_ptr	The PCB pointer.
pPriority	The assigned priority

Returns

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

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

show_pcb

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

Parameters

pName	The PCB pointer.
-------	------------------

Returns

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

- [error_t show_pcb](#) (struct [pcb_struct](#) *pcb_ptr)

show_all_processes

Displays all of the processes and their attributes.

Returns

VOID.

- void [show_all_processes](#) ()

show_ready_processes

Displays all of the ready processes and their attributes.

Returns

VOID.

- void [show_ready_processes](#) ()

show_blocked_processes

displays all blocked processes and their attributes

Returns

VOID.

- void [show_blocked_processes](#) ()

block_pcb

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

Parameters

pcb_ptr	The pointer to the PCB
---------	------------------------

Returns

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

- [error_t block_pcb](#) (struct [pcb_struct](#) *pcb_ptr)

unblock_pcb

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

Parameters

pcb_ptr	The pointer to the PCB
---------	------------------------

Returns

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

- [error_t unblock_pcb](#) (struct [pcb_struct](#) *pcb_ptr)

get_running_process

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

Parameters

None	
------	--

Returns

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

- struct [pcb_struct](#) * [get_running_process](#) ()

save_running_process

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

Parameters

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

Returns

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

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

get_stack_top

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

Parameters

pcb_ptr	The pointer to the PCB.
---------	-------------------------

Returns

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

- unsigned char * [get_stack_top](#) (struct [pcb_struct](#) *pcb_ptr)

get_stack_base

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

Parameters

pcb_ptr	The pointer to the PCB.
---------	-------------------------

Returns

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

- unsigned char * [get_stack_base](#) (struct [pcb_struct](#) *pcb_ptr)

shutdown_pcb

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

Returns

VOID

- void [shutdown_pcb](#) ()

Variables

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

5.13.1 Detailed Description

The Process Control Block.

Author

Thunder Krakens

Date

February 7th, 2016

Version

R3

5.13.2 Macro Definition Documentation

5.13.2.1 `#define SIZE_OF_PCB_NAME 10`

5.13.2.2 `#define SIZE_OF_STACK 1024`

5.13.3 Enumeration Type Documentation

5.13.3.1 `enum process_class`

PCB process class types.

5.13.4 Function Documentation

5.13.4.1 `enum process_class __attribute__((packed))`

5.13.4.2 `struct pcb_struct* allocate_pcb ()`

Here is the call graph for this function:

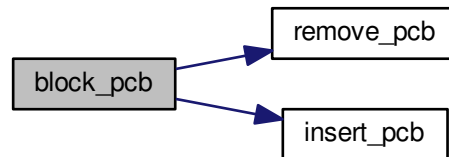


Here is the caller graph for this function:



5.13.4.3 `error_t block_pcb (struct pcb_struct * pcb_ptr)`

Here is the call graph for this function:

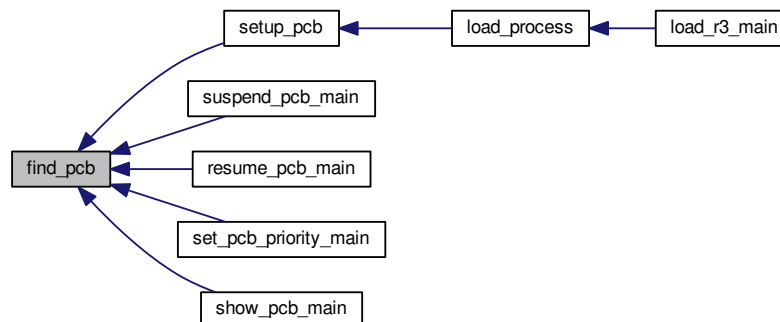


5.13.4.4 `struct pcb_struct* find_pcb (const char * pName)`

Here is the call graph for this function:



Here is the caller graph for this function:

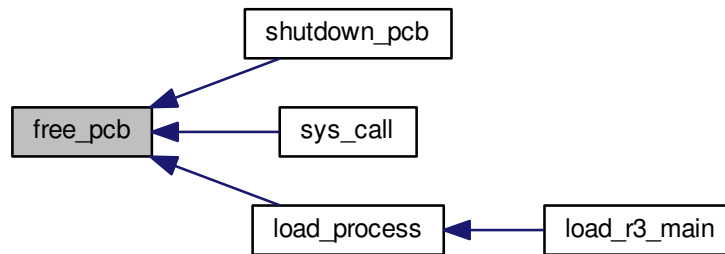


5.13.4.5 `error_t free_pcb (struct pcb_struct * pcb_ptr)`

Here is the call graph for this function:

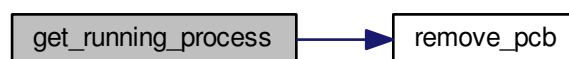


Here is the caller graph for this function:

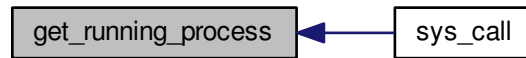


5.13.4.6 `struct pcb_struct* get_running_process ()`

Here is the call graph for this function:

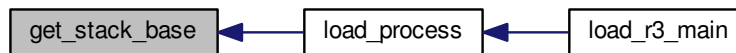


Here is the caller graph for this function:



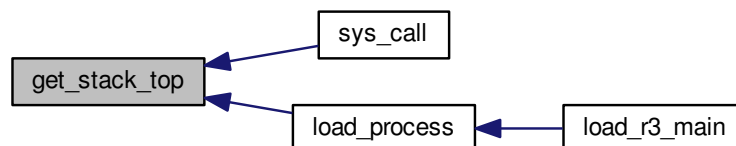
5.13.4.7 `unsigned char* get_stack_base (struct pcb_struct * pcb_ptr)`

Here is the caller graph for this function:



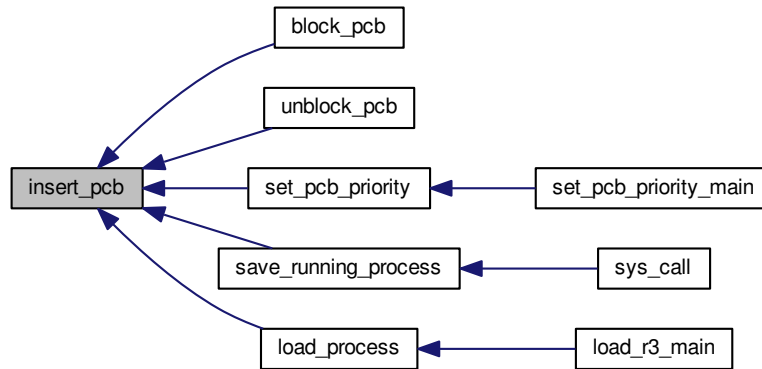
5.13.4.8 `unsigned char* get_stack_top (struct pcb_struct * pcb_ptr)`

Here is the caller graph for this function:



5.13.4.9 `error_t insert_pcb (struct pcb_struct * pcb_ptr)`

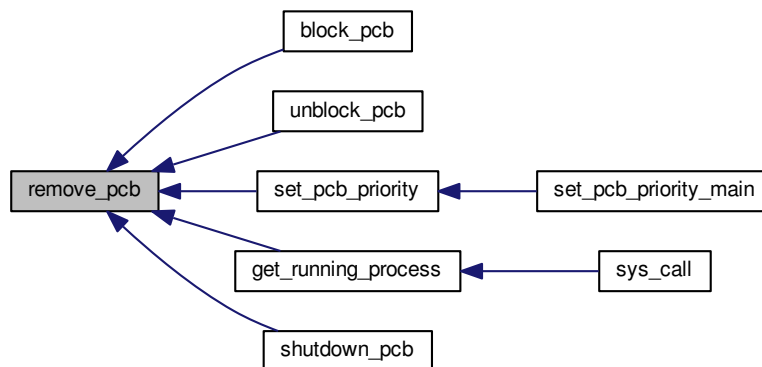
Here is the caller graph for this function:



5.13.4.10 `void pcb_init ()`

5.13.4.11 `error_t remove_pcb (struct pcb_struct * pcb_ptr)`

Here is the caller graph for this function:



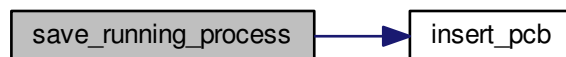
5.13.4.12 `error_t resume_pcb (struct pcb_struct * pcb_ptr)`

Here is the caller graph for this function:

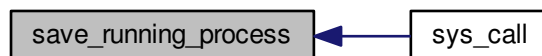


5.13.4.13 `error_t save_running_process (struct pcb_struct * pcb_ptr, struct context * new_stack_top)`

Here is the call graph for this function:

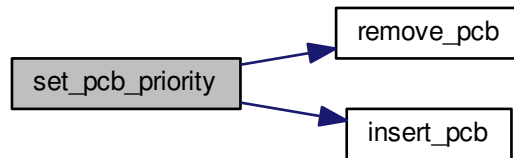


Here is the caller graph for this function:



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

Here is the call graph for this function:

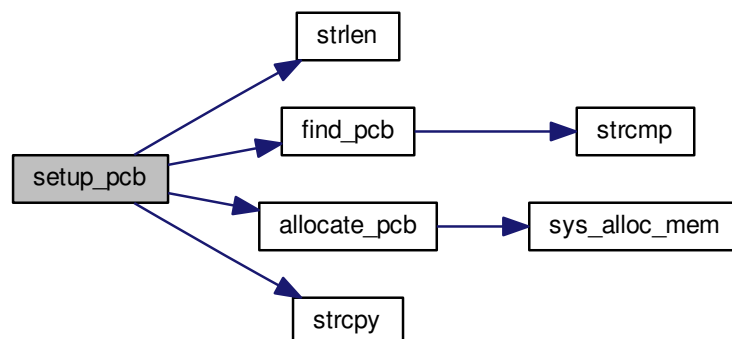


Here is the caller graph for this function:

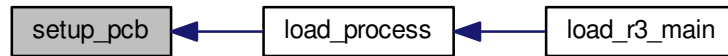


5.13.4.15 `struct pcb_struct* setup_pcb (const char * pName, const enum process_class pClass, const unsigned char pPriority)`

Here is the call graph for this function:

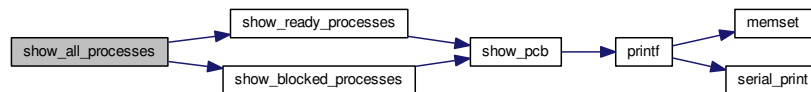


Here is the caller graph for this function:



5.13.4.16 void show_all_processes ()

Here is the call graph for this function:

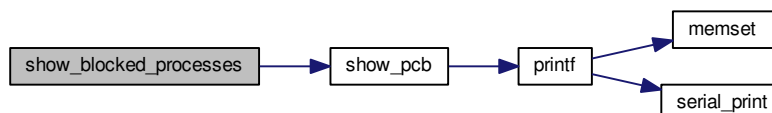


Here is the caller graph for this function:

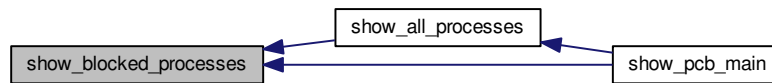


5.13.4.17 void show_blocked_processes ()

Here is the call graph for this function:

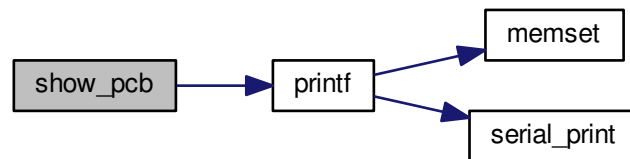


Here is the caller graph for this function:

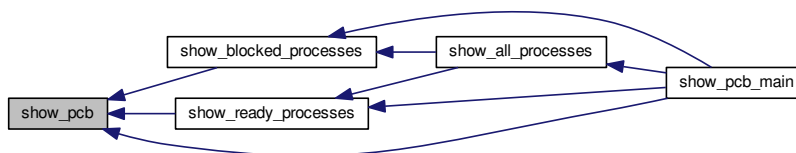


5.13.4.18 `error_t show_pcb (struct pcb_struct * pcb_ptr)`

Here is the call graph for this function:

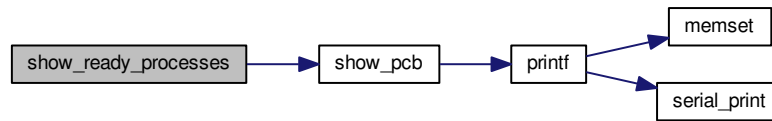


Here is the caller graph for this function:

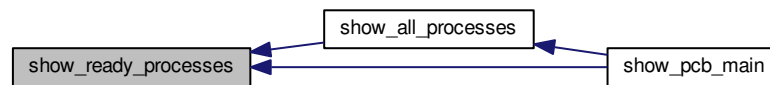


5.13.4.19 void show_ready_processes ()

Here is the call graph for this function:

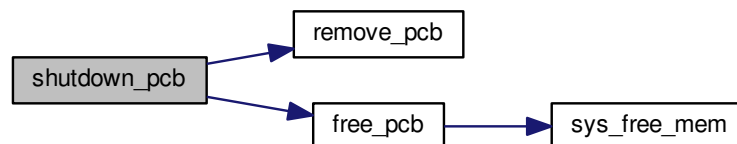


Here is the caller graph for this function:



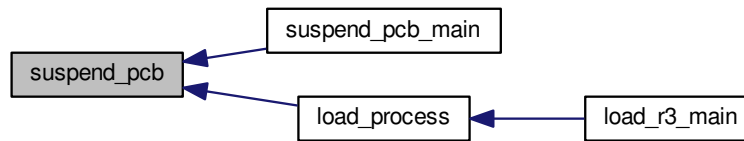
5.13.4.20 void shutdown_pcb ()

Here is the call graph for this function:

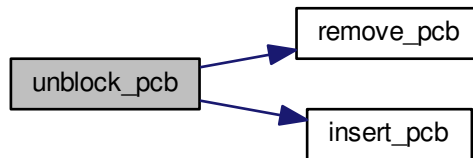


5.13.4.21 `error_t suspend_pcb (struct pcb_struct * pcb_ptr)`

Here is the caller graph for this function:

5.13.4.22 `error_t unblock_pcb (struct pcb_struct * pcb_ptr)`

Here is the call graph for this function:



5.13.5 Variable Documentation

5.13.5.1 `pcb_class_app`

Process is an application process.

5.13.5.2 `pcb_class_sys`

< Process is a system process.

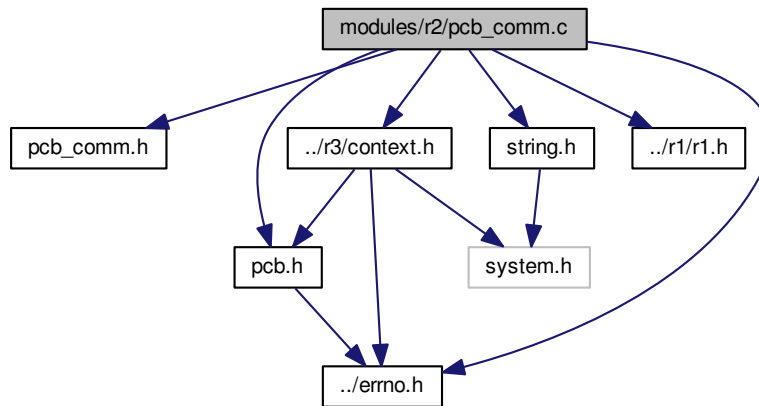
Process is a system process.

5.14 modules/r2/pcb_comm.c File Reference

The main functions that manipulate the PCB.

```
#include "pcb_comm.h"
#include "pcb.h"
#include <string.h>
#include "../errno.h"
#include "../r1/r1.h"
#include "../r3/context.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	The number of tokens found.
argv	The array of tokens.

Returns

0

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

resume_pcb_main.

The main function for the "resume PCB".

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

Parameters

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

Returns

0

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

set_pcb_priority_main.

The main function for the "set PCB priority".

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

Parameters

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

Returns

0

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

show_pcb_main.

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

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

Parameters

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

Returns

0

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

5.14.1 Detailed Description

The main functions that manipulate the PCB.

Author

Thunder Krakens

Date

February 7th, 2016

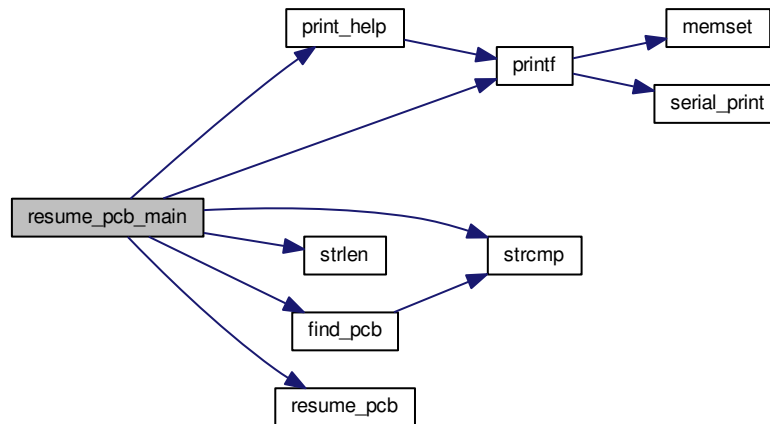
Version

R2

5.14.2 Function Documentation

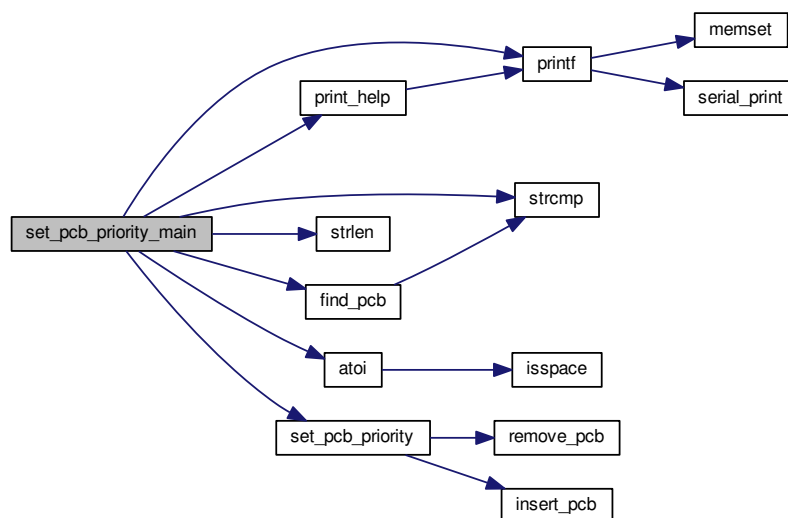
5.14.2.1 `int resume_pcb_main (int argc, char ** argv)`

Here is the call graph for this function:



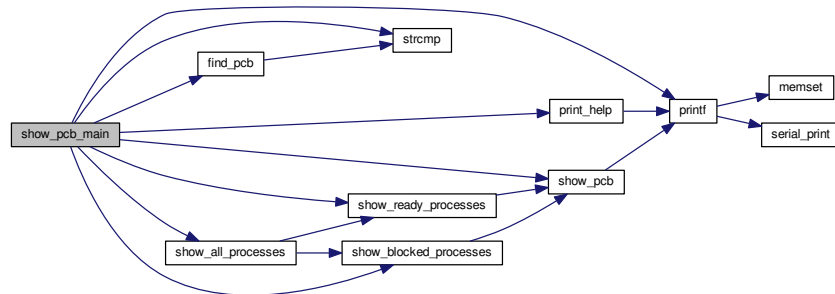
5.14.2.2 `int set_pcb_priority_main (int argc, char ** argv)`

Here is the call graph for this function:



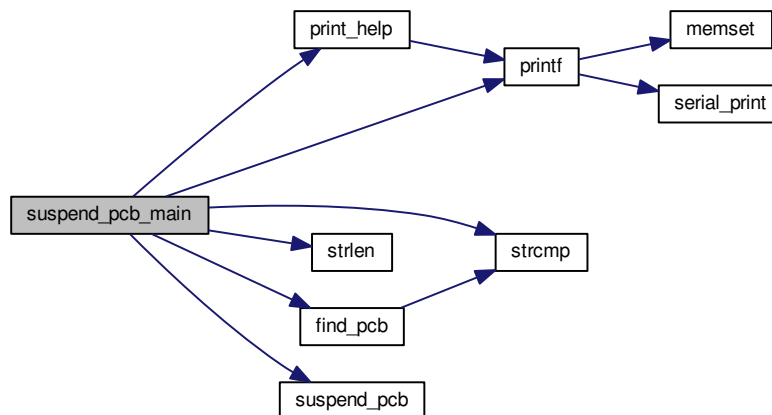
5.14.2.3 int show_pcb_main (int argc, char ** argv)

Here is the call graph for this function:



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

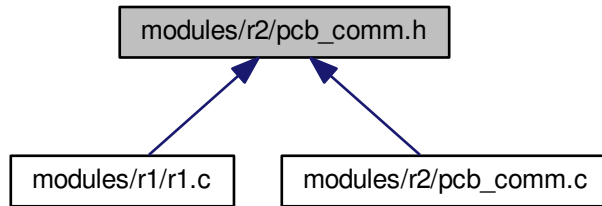
Here is the call graph for this function:



5.15 modules/r2/pcb_comm.h File Reference

The main functions that manipulate the PCB.

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



Functions

suspend_pcb_main.

The main function for the "suspend PCB".

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

Parameters

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

Returns

0

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

resume_pcb_main.

The main function for the "resume PCB".

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

Parameters

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

Returns

0

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

set_pcb_priority_main.

The main function for the "set PCB priority".

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

Parameters

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

Returns

0

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

show_pcb_main.

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

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

Parameters

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

Returns

0

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

create_pcb_main.

The main function for the "Create PCB".

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

Parameters

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

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	<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)

5.15.1 Detailed Description

The main functions that manipulate the PCB.

Author

Thunder Krakens

Date

February 7th, 2016

Version

R2

5.15.2 Function Documentation

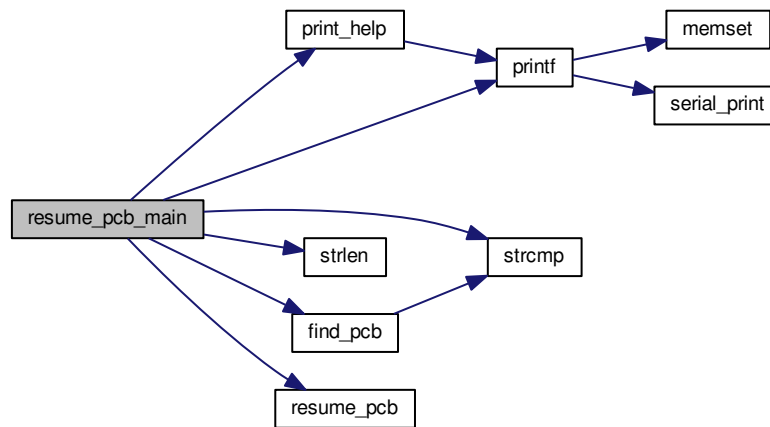
5.15.2.1 int [block_pcb_main](#) (int *argc*, char ** *argv*)

5.15.2.2 int [create_pcb_main](#) (int *argc*, char ** *argv*)

5.15.2.3 int [delete_pcb_main](#) (int *argc*, char ** *argv*)

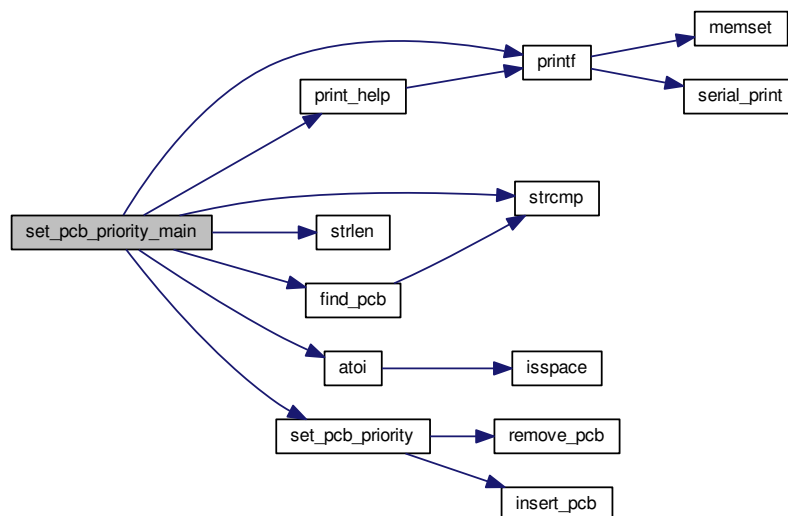
5.15.2.4 int resume_pcb_main (int argc, char ** argv)

Here is the call graph for this function:



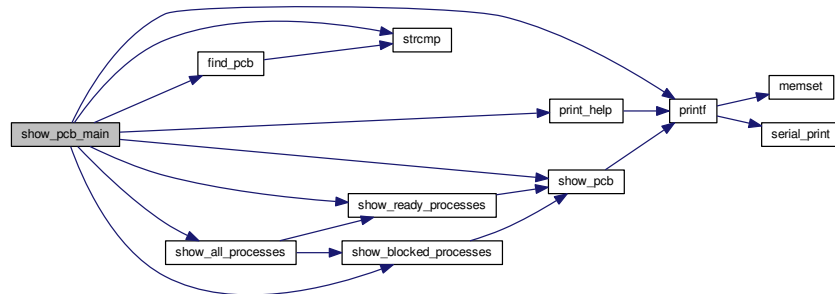
5.15.2.5 int set_pcb_priority_main (int argc, char ** argv)

Here is the call graph for this function:



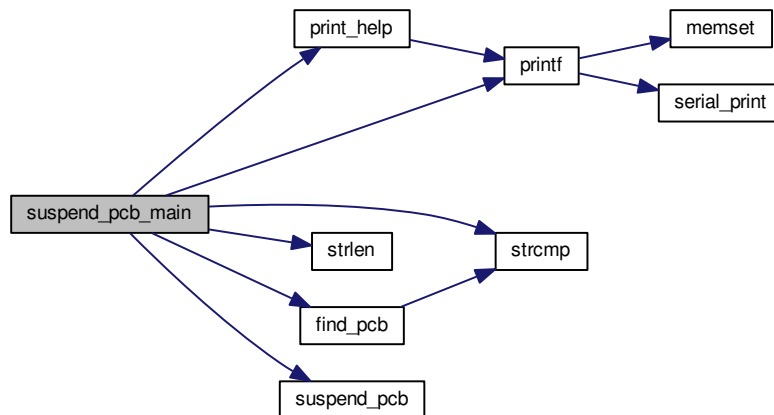
5.15.2.6 int show_pcb_main (int argc, char ** argv)

Here is the call graph for this function:



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

Here is the call graph for this function:

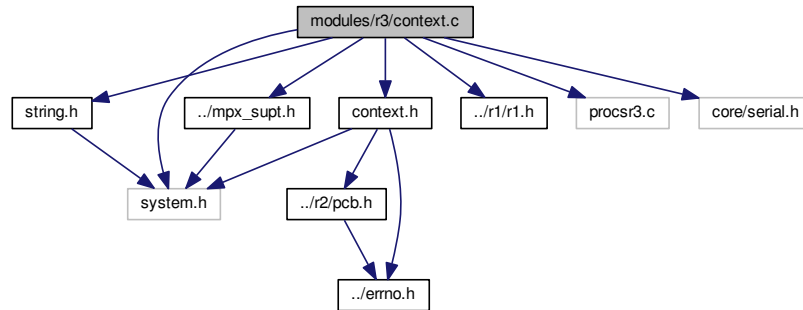


5.15.2.8 int unblock_pcb_main (int argc, char ** argv)

5.16 modules/r3/context.c File Reference

Context Switching.

```
#include <string.h>
#include "context.h"
#include "../mpx_supt.h"
#include "../r1/r1.h"
#include "procsr3.c"
Include dependency graph for context.c:
```



Functions

is_digit

Checks if the character is a digit.

Parameters

ch	character selected.
----	---------------------

Returns

a digit between 0 and 9.

sys_call

system call interrupt

Parameters

context*	registers current registers
----------	-----------------------------

Returns

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

- u32int * [sys_call](#) (struct [context](#) *registers)

load_process

loads a process into the PCB.

Parameters

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

Returns

new_pcb Returns the values of the new PCB

- struct [pcb_struct](#) * [load_process](#) (const char *pName, const enum [process_class](#) pClass, const unsigned char pPriority, void(*function)())

yield_main

Requests an IDLE interrupt.

Parameters

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

Returns

0

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

load_r3_main

Loads the main function of R3.

Parameters

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

Returns

0

- int [load_r3_main](#) (int argc, char **argv)

Variables

- struct [pcb_struct](#) * [cop](#) = NULL
- struct [context](#) * [old_context](#) = NULL

5.16.1 Detailed Description

Context Switching.

Author

Thunder Krakens

Date

March 18th, 2016

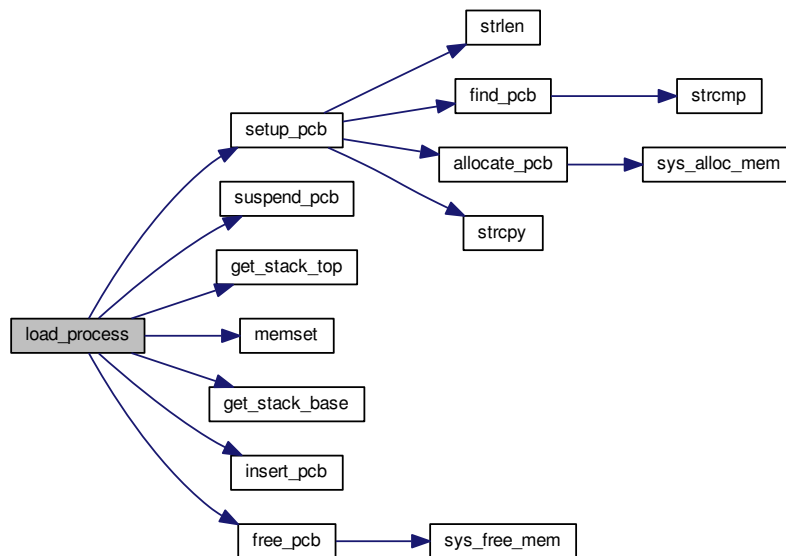
Version

R3

5.16.2 Function Documentation

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

Here is the call graph for this function:

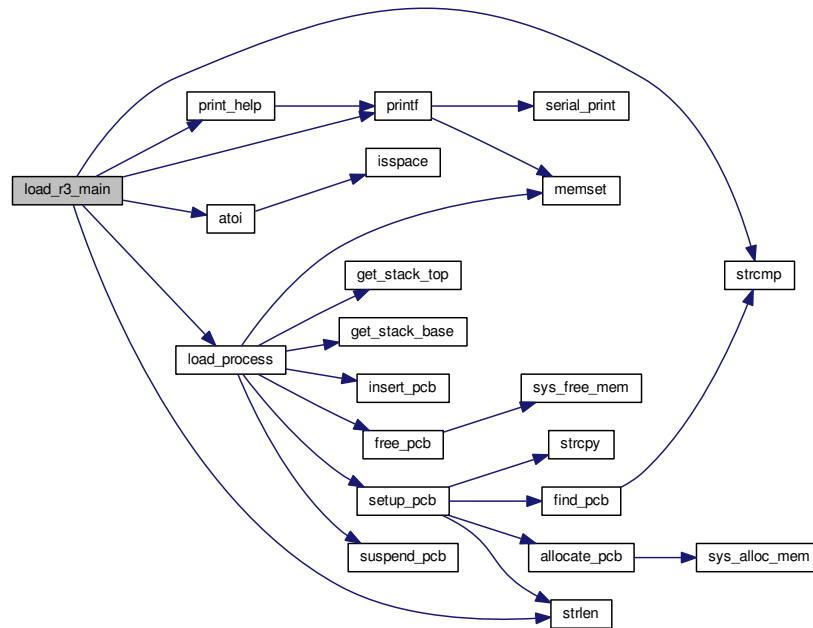


Here is the caller graph for this function:



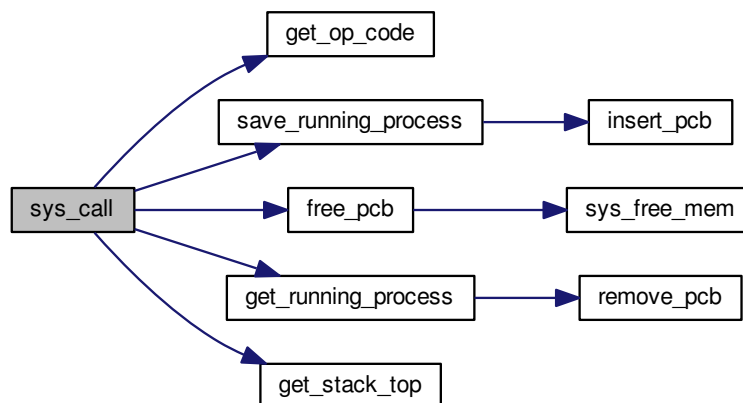
5.16.2.2 int load_r3_main (int argc, char ** argv)

Here is the call graph for this function:



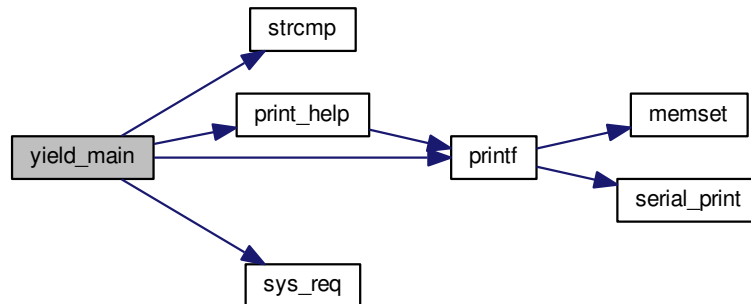
5.16.2.3 u32int* sys_call (struct context * registers)

Here is the call graph for this function:



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

Here is the call graph for this function:



5.16.3 Variable Documentation

5.16.3.1 struct pcb_struct* cop = NULL

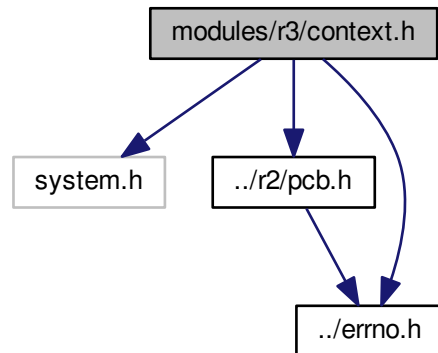
5.16.3.2 struct context* old_context = NULL

5.17 modules/r3/context.h File Reference

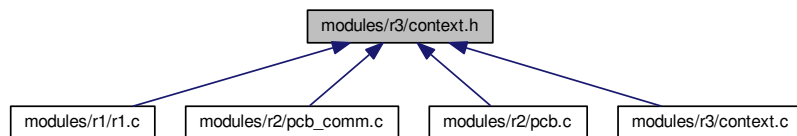
Context Switching.

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

Include dependency graph for context.h:



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



Data Structures

- struct [context](#)

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

Functions

sys_call

system call interrupt

Parameters

context*	<i>registers current registers</i>
----------	------------------------------------

Returns

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

- u32int * [sys_call](#) (struct [context](#) *registers)

load_process

loads a process into the PCB.

Parameters

pName	<i>Process Name</i>
pClass	<i>Process Class</i>
pPriority	<i>Process Priority</i>
*function()	<i>A function pointer</i>

Returns

new_pcb Returns the values of the new PCB

- struct [pcb_struct](#) * [load_process](#) (const char *pName, const enum [process_class](#) pClass, const unsigned char pPriority, void(*function)())

yield_main

Requests an IDLE interrupt.

Parameters

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

Returns

0

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

load_r3_main

Loads the main function of R3.

Parameters

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

Returns

0

- int [load_r3_main](#) (int argc, char **argv)

Variables

- struct [context](#) * [old_context](#)
- struct [pcb_struct](#) * [cop](#)

5.17.1 Detailed Description

Context Switching.

Author

Thunder Krakens

Date

March 18th, 2016

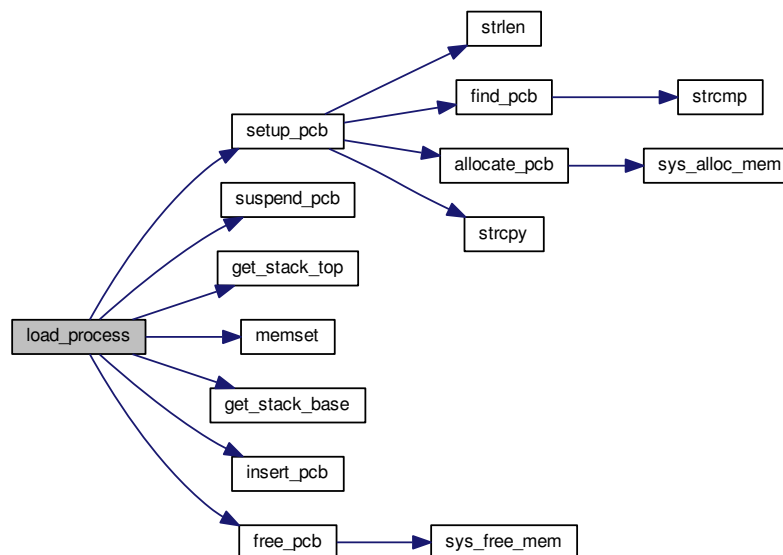
Version

R3

5.17.2 Function Documentation

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

Here is the call graph for this function:

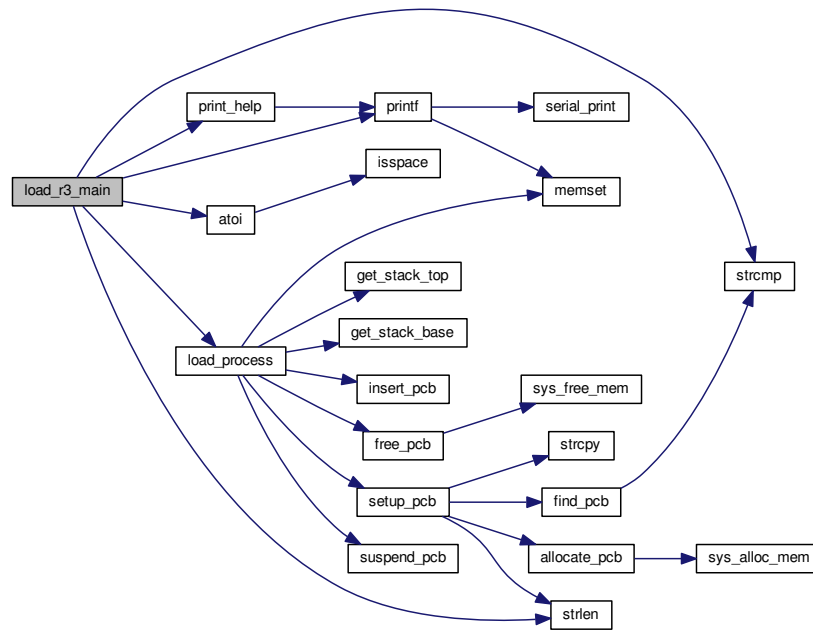


Here is the caller graph for this function:



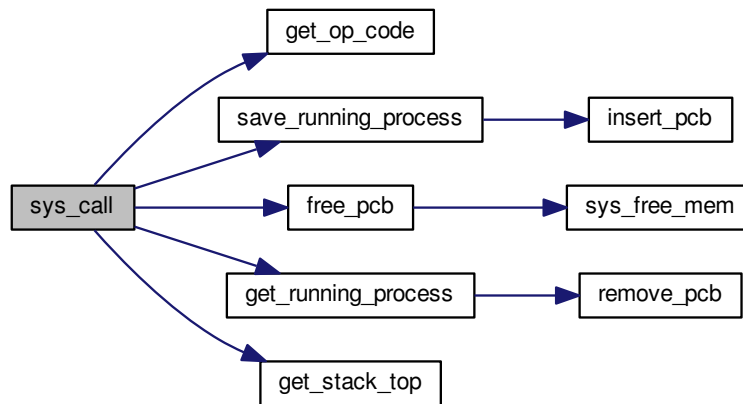
5.17.2.2 int load_r3_main (int argc, char ** argv)

Here is the call graph for this function:



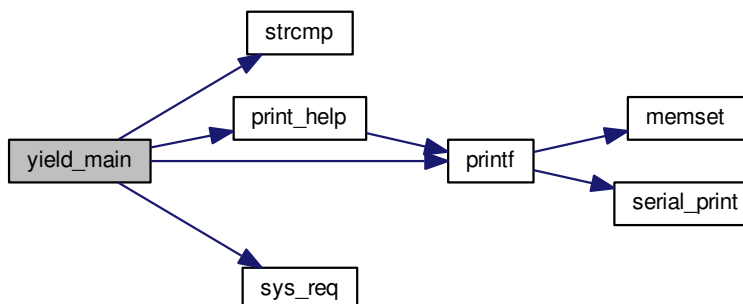
5.17.2.3 `u32int* sys_call (struct context * registers)`

Here is the call graph for this function:



5.17.2.4 `int yield_main (int argc, char ** argv)`

Here is the call graph for this function:



5.17.3 Variable Documentation

5.17.3.1 `struct pcb_struct* cop`

5.17.3.2 `struct context* old_context`

Index

- `__attribute__`
 - `mpx_supt.h`, 46
 - `pcb.c`, 83, 94
 - `pcb.h`, 101
 - `r1.c`, 49
 - `r1.h`, 55
- `allocate_pcb`
 - `pcb.c`, 83
 - `pcb.h`, 101
- `atoi`
 - `string.c`, 30
 - `string.h`, 21
- `block_pcb`
 - `pcb.c`, 84
 - `pcb.h`, 101
- `block_pcb_main`
 - `pcb_comm.h`, 118
- `blocked`
 - `pcb.c`, 94
- `COM1`
 - `serial.h`, 16
- `COM2`
 - `serial.h`, 16
- `COM3`
 - `serial.h`, 16
- `COM4`
 - `serial.h`, 16
- `COMPLETION`
 - `r1.c`, 49
- `class`
 - `pcb_struct`, 13
- `comm_type`
 - `r1.h`, 55
- `command_line_parser`
 - `r1.c`, 49
 - `r1.h`, 55
- `CommandPaserStat`
 - `r1.c`, 49
- `commhand`
 - `r1.c`, 49
 - `r1.h`, 56
- `context`, 7
 - `cs`, 8
 - `ds`, 8
 - `eax`, 8
 - `ebp`, 8
 - `ebx`, 8
 - `ecx`, 8
 - `edi`, 8
 - `edx`, 8
 - `eflags`, 8
 - `eip`, 9
 - `es`, 9
 - `esi`, 9
 - `esp`, 9
 - `fs`, 9
 - `gs`, 9
- `context.c`
 - `cop`, 125
 - `load_process`, 123
 - `load_r3_main`, 123
 - `old_context`, 125
 - `sys_call`, 124
 - `yield_main`, 125
- `context.h`
 - `cop`, 130
 - `load_process`, 128
 - `load_r3_main`, 129
 - `old_context`, 130
 - `sys_call`, 129
 - `yield_main`, 130
- `cop`
 - `context.c`, 125
 - `context.h`, 130
- `count`
 - `pcb_queue`, 11
- `create_pcb_main`
 - `pcb_comm.h`, 118
- `cs`
 - `context`, 8
- `current_module`
 - `mpx_supt.c`, 40
- `delete_pcb_main`
 - `pcb_comm.h`, 118
- `device_id`
 - `param`, 10
- `documentation/mainpage.dox`, 15
- `DoubleQuoteWriting`

- r1.c, [51](#)
- ds
 - context, [8](#)
- E_EMPTPCB
 - errno.h, [37](#)
- E_FREEMEM
 - errno.h, [37](#)
- E_INVPARA
 - errno.h, [37](#)
- E_INVSTRF
 - errno.h, [37](#)
- E_INVUSRI
 - errno.h, [37](#)
- E_NOERROR
 - errno.h, [37](#)
- E_NULL_PTR
 - errno.h, [37](#)
- E_PCB_SYS
 - errno.h, [37](#)
- E_PROGERR
 - errno.h, [37](#)
- EXIT
 - mpx_supt.h, [43](#)
- eax
 - context, [8](#)
- ebp
 - context, [8](#)
- ebx
 - context, [8](#)
- ecx
 - context, [8](#)
- edi
 - context, [8](#)
- edx
 - context, [8](#)
- eflags
 - context, [8](#)
- eip
 - context, [9](#)
- errno.h
 - E_EMPTPCB, [37](#)
 - E_FREEMEM, [37](#)
 - E_INVPARA, [37](#)
 - E_INVSTRF, [37](#)
 - E_INVUSRI, [37](#)
 - E_NOERROR, [37](#)
 - E_NULL_PTR, [37](#)
 - E_PCB_SYS, [37](#)
 - E_PROGERR, [37](#)
 - error_t, [37](#)
- error_t
 - errno.h, [37](#)
- es
 - context, [9](#)
- esi
 - context, [9](#)
- esp
 - context, [9](#)
- false
 - pcb.c, [94](#)
- find_pcb
 - pcb.c, [84](#)
 - pcb.h, [102](#)
- free_pcb
 - pcb.c, [85](#)
 - pcb.h, [102](#)
- fs
 - context, [9](#)
- function
 - function_name, [10](#)
- function_name, [9](#)
 - function, [10](#)
 - help, [10](#)
 - nameStr, [10](#)
 - usage, [10](#)
- GETDATE
 - r1.h, [55](#)
- GETTIME
 - r1.h, [55](#)
- get_date
 - sys_clock.c, [62](#)
 - sys_clock.h, [71](#)
- get_date_main
 - sys_clock.c, [62](#)
 - sys_clock.h, [72](#)
- get_input_line
 - serial.h, [17](#)
- get_op_code
 - mpx_supt.c, [38](#)
 - mpx_supt.h, [44](#)
- get_running_process
 - pcb.c, [85](#)
 - pcb.h, [103](#)
- get_stack_base
 - pcb.c, [86](#)
 - pcb.h, [104](#)
- get_stack_top
 - pcb.c, [86](#)
 - pcb.h, [104](#)
- get_time
 - sys_clock.c, [63](#)
 - sys_clock.h, [72](#)
- get_time_main
 - sys_clock.c, [63](#)
 - sys_clock.h, [72](#)
- gs

- context, 9
- HELP
 - r1.h, 55
- head
 - pcb_queue, 11
- help
 - function_name, 10
 - r1.h, 57
- help_usages
 - r1.c, 50
 - r1.h, 56
- IDLE
 - mpx_supt.h, 43
- idle
 - mpx_supt.c, 39
 - mpx_supt.h, 44
- include/core/serial.h, 15
- include/string.h, 17
- init_serial
 - serial.h, 17
- insert_pcb
 - pcb.c, 87
 - pcb.h, 104
- is_suspended
 - pcb_struct, 13
- isspace
 - string.c, 31
 - string.h, 22
- LOADR3
 - r1.h, 55
- lib/string.c, 26
- load_process
 - context.c, 123
 - context.h, 128
- load_r3_main
 - context.c, 123
 - context.h, 129
- MAX_ARGC
 - r1.c, 49
- MAX_HISTORY
 - r1.c, 49
- MOD_VERSION
 - r1.c, 49
- MODULE_R1
 - mpx_supt.h, 43
- MODULE_R2
 - mpx_supt.h, 43
- MODULE_R3
 - mpx_supt.h, 43
- MODULE_R4
 - mpx_supt.h, 44
- MODULE_R5
 - mpx_supt.h, 44
- memset
 - string.c, 31
 - string.h, 22
- modules/errno.h, 35
- modules/mpx_supt.c, 37
- modules/mpx_supt.h, 41
- modules/r1/r1.c, 46
- modules/r1/r1.h, 52
- modules/r1/sys_clock.c, 57
- modules/r1/sys_clock.h, 68
- modules/r2/pcb.c, 77
- modules/r2/pcb.h, 94
- modules/r2/pcb_comm.c, 111
- modules/r2/pcb_comm.h, 115
- modules/r3/context.c, 120
- modules/r3/context.h, 125
- mpx
 - r1.h, 57
- mpx_init
 - mpx_supt.c, 39
 - mpx_supt.h, 44
- mpx_supt.c
 - current_module, 40
 - get_op_code, 38
 - idle, 39
 - mpx_init, 39
 - params, 40
 - student_free, 40
 - student_malloc, 40
 - sys_alloc_mem, 39
 - sys_free_mem, 39
 - sys_req, 40
 - sys_set_free, 40
 - sys_set_malloc, 40
- mpx_supt.h
 - __attribute__, 46
 - EXIT, 43
 - get_op_code, 44
 - IDLE, 43
 - idle, 44
 - MODULE_R1, 43
 - MODULE_R2, 43
 - MODULE_R3, 43
 - MODULE_R4, 44
 - MODULE_R5, 44
 - mpx_init, 44
 - READ, 44
 - sys_alloc_mem, 44
 - sys_free_mem, 45
 - sys_req, 45
 - sys_set_free, 45
 - sys_set_malloc, 45

WRITE, 44
 NUM_MPX_FUNCTIONS
 r1.h, 55
 NUM_OF_FUNCTIONS
 r1.h, 55
 name
 pcb_struct, 13
 nameStr
 function_name, 10
 next
 pcb_struct, 13
 NormalWriting
 r1.c, 51
 NotWriting
 r1.c, 51
 old_context
 context.c, 125
 context.h, 130
 op_code
 param, 10
 POS_OF_MPX
 r1.h, 55
 POS_OF_PCB
 r1.h, 55
 param, 10
 device_id, 10
 op_code, 10
 params
 mpx_supt.c, 40
 pcb
 r1.h, 57
 pcb.c
 __attribute__, 83, 94
 allocate_pcb, 83
 block_pcb, 84
 blocked, 94
 false, 94
 find_pcb, 84
 free_pcb, 85
 get_running_process, 85
 get_stack_base, 86
 get_stack_top, 86
 insert_pcb, 87
 pcb_init, 87
 process_state, 83
 process_suspended, 83
 ready, 94
 remove_pcb, 87
 resume_pcb, 88
 running, 94
 save_running_process, 88
 set_pcb_priority, 89
 setup_pcb, 89
 show_all_processes, 90
 show_blocked_processes, 91
 show_pcb, 91
 show_ready_processes, 92
 shutdown_pcb, 93
 suspend_pcb, 93
 true, 94
 unblock_pcb, 93
 pcb.h
 __attribute__, 101
 allocate_pcb, 101
 block_pcb, 101
 find_pcb, 102
 free_pcb, 102
 get_running_process, 103
 get_stack_base, 104
 get_stack_top, 104
 insert_pcb, 104
 pcb_class_app, 111
 pcb_class_sys, 111
 pcb_init, 105
 process_class, 101
 remove_pcb, 105
 resume_pcb, 105
 SIZE_OF_PCB_NAME, 101
 SIZE_OF_STACK, 101
 save_running_process, 106
 set_pcb_priority, 106
 setup_pcb, 107
 show_all_processes, 108
 show_blocked_processes, 108
 show_pcb, 109
 show_ready_processes, 109
 shutdown_pcb, 110
 suspend_pcb, 110
 unblock_pcb, 111
 pcb_class_app
 pcb.h, 111
 pcb_class_sys
 pcb.h, 111
 pcb_comm.c
 resume_pcb_main, 114
 set_pcb_priority_main, 114
 show_pcb_main, 114
 suspend_pcb_main, 115
 pcb_comm.h
 block_pcb_main, 118
 create_pcb_main, 118
 delete_pcb_main, 118
 resume_pcb_main, 118
 set_pcb_priority_main, 119
 show_pcb_main, 119
 suspend_pcb_main, 120

- unblock_pcb_main, 120
- pcb_init
 - pcb.c, 87
 - pcb.h, 105
- pcb_queue, 11
 - count, 11
 - head, 11
 - tail, 12
- pcb_struct, 12
 - class, 13
 - is_suspended, 13
 - name, 13
 - next, 13
 - prev, 13
 - priority, 13
 - running_state, 13
 - stack_base, 13
 - stack_top, 13
- prev
 - pcb_struct, 13
- print_help
 - r1.c, 50
 - r1.h, 56
- printf
 - string.c, 32
 - string.h, 23
- priority
 - pcb_struct, 13
- process_class
 - pcb.h, 101
- process_state
 - pcb.c, 83
- process_suspended
 - pcb.c, 83
- r1.c
 - __attribute__, 49
 - COMPLETION, 49
 - command_line_parser, 49
 - CommandPaserStat, 49
 - commhand, 49
 - DoubleQuoteWriting, 51
 - help_usages, 50
 - MAX_ARGC, 49
 - MAX_HISTORY, 49
 - MOD_VERSION, 49
 - NormalWriting, 51
 - NotWriting, 51
 - print_help, 50
 - SingleQuoteWriting, 51
- r1.h
 - __attribute__, 55
 - comm_type, 55
 - command_line_parser, 55
 - commhand, 56
 - GETDATE, 55
 - GETTIME, 55
 - HELP, 55
 - help, 57
 - help_usages, 56
 - LOADR3, 55
 - mpx, 57
 - NUM_MPX_FUNCTIONS, 55
 - NUM_OF_FUNCTIONS, 55
 - POS_OF_MPX, 55
 - POS_OF_PCB, 55
 - pcb, 57
 - print_help, 56
 - RESUMEPCB, 55
 - SETDATE, 55
 - SETPCBPRIO, 55
 - SETTIME, 55
 - SHOWPCB, 55
 - SHUTDOWN, 55
 - SUSPDPCB, 55
 - VERSION, 55
 - YIELD, 55
- READ
 - mpx_supt.h, 44
- RESUMEPCB
 - r1.h, 55
- RTC_INDEX_HOUR
 - sys_clock.c, 62
- RTC_INDEX_MINUTE
 - sys_clock.c, 62
- RTC_INDEX_MONTH
 - sys_clock.c, 62
- RTC_INDEX_SECOND
 - sys_clock.c, 62
- RTC_INDEX_YEAR
 - sys_clock.c, 62
- ready
 - pcb.c, 94
- remove_pcb
 - pcb.c, 87
 - pcb.h, 105
- resume_pcb
 - pcb.c, 88
 - pcb.h, 105
- resume_pcb_main
 - pcb_comm.c, 114
 - pcb_comm.h, 118
- running
 - pcb.c, 94
- running_state
 - pcb_struct, 13
- SETDATE

- r1.h, [55](#)
- SETPCBPRIO
 - r1.h, [55](#)
- SETTIME
 - r1.h, [55](#)
- SHOWPCB
 - r1.h, [55](#)
- SHUTDOWN
 - r1.h, [55](#)
- SIZE_OF_PCB_NAME
 - pcb.h, [101](#)
- SIZE_OF_STACK
 - pcb.h, [101](#)
- SUSPDCB
 - r1.h, [55](#)
- save_running_process
 - pcb.c, [88](#)
 - pcb.h, [106](#)
- serial.h
 - COM1, [16](#)
 - COM2, [16](#)
 - COM3, [16](#)
 - COM4, [16](#)
 - get_input_line, [17](#)
 - init_serial, [17](#)
 - serial_print, [17](#)
 - serial_println, [17](#)
 - set_serial_in, [17](#)
 - set_serial_out, [17](#)
 - WithEcho, [16](#)
 - WithoutEcho, [16](#)
- serial_print
 - serial.h, [17](#)
- serial_println
 - serial.h, [17](#)
- set_date
 - sys_clock.c, [64](#)
 - sys_clock.h, [73](#)
- set_date_main
 - sys_clock.c, [64](#)
 - sys_clock.h, [73](#)
- set_date_str
 - sys_clock.c, [65](#)
 - sys_clock.h, [74](#)
- set_pcb_priority
 - pcb.c, [89](#)
 - pcb.h, [106](#)
- set_pcb_priority_main
 - pcb_comm.c, [114](#)
 - pcb_comm.h, [119](#)
- set_serial_in
 - serial.h, [17](#)
- set_serial_out
 - serial.h, [17](#)
- set_time
 - sys_clock.c, [66](#)
 - sys_clock.h, [75](#)
- set_time_main
 - sys_clock.c, [66](#)
 - sys_clock.h, [75](#)
- set_time_str
 - sys_clock.c, [67](#)
 - sys_clock.h, [76](#)
- setup_pcb
 - pcb.c, [89](#)
 - pcb.h, [107](#)
- show_all_processes
 - pcb.c, [90](#)
 - pcb.h, [108](#)
- show_blocked_processes
 - pcb.c, [91](#)
 - pcb.h, [108](#)
- show_pcb
 - pcb.c, [91](#)
 - pcb.h, [109](#)
- show_pcb_main
 - pcb_comm.c, [114](#)
 - pcb_comm.h, [119](#)
- show_ready_processes
 - pcb.c, [92](#)
 - pcb.h, [109](#)
- shutdown_pcb
 - pcb.c, [93](#)
 - pcb.h, [110](#)
- SingleQuoteWriting
 - r1.c, [51](#)
- sprintf
 - string.c, [33](#)
 - string.h, [24](#)
- stack_base
 - pcb_struct, [13](#)
- stack_top
 - pcb_struct, [13](#)
- strcat
 - string.c, [33](#)
 - string.h, [24](#)
- strcmp
 - string.c, [33](#)
 - string.h, [24](#)
- strcpy
 - string.c, [34](#)
 - string.h, [25](#)
- string.c
 - atoi, [30](#)
 - isspace, [31](#)
 - memset, [31](#)
 - printf, [32](#)
 - sprintf, [33](#)

- strcat, [33](#)
- strcmp, [33](#)
- strcpy, [34](#)
- strlen, [34](#)
- strtok, [35](#)
- string.h
 - atoi, [21](#)
 - isspace, [22](#)
 - memset, [22](#)
 - printf, [23](#)
 - sprintf, [24](#)
 - strcat, [24](#)
 - strcmp, [24](#)
 - strcpy, [25](#)
 - strlen, [25](#)
 - strtok, [26](#)
- strlen
 - string.c, [34](#)
 - string.h, [25](#)
- strtok
 - string.c, [35](#)
 - string.h, [26](#)
- student_free
 - mpx_supt.c, [40](#)
- student_malloc
 - mpx_supt.c, [40](#)
- suspend_pcb
 - pcb.c, [93](#)
 - pcb.h, [110](#)
- suspend_pcb_main
 - pcb_comm.c, [115](#)
 - pcb_comm.h, [120](#)
- sys_alloc_mem
 - mpx_supt.c, [39](#)
 - mpx_supt.h, [44](#)
- sys_call
 - context.c, [124](#)
 - context.h, [129](#)
- sys_clock.c
 - get_date, [62](#)
 - get_date_main, [62](#)
 - get_time, [63](#)
 - get_time_main, [63](#)
 - RTC_INDEX_HOUR, [62](#)
 - RTC_INDEX_MINUTE, [62](#)
 - RTC_INDEX_MONTH, [62](#)
 - RTC_INDEX_SECOND, [62](#)
 - RTC_INDEX_YEAR, [62](#)
 - set_date, [64](#)
 - set_date_main, [64](#)
 - set_date_str, [65](#)
 - set_time, [66](#)
 - set_time_main, [66](#)
 - set_time_str, [67](#)
- sys_clock.h
 - get_date, [71](#)
 - get_date_main, [72](#)
 - get_time, [72](#)
 - get_time_main, [72](#)
 - set_date, [73](#)
 - set_date_main, [73](#)
 - set_date_str, [74](#)
 - set_time, [75](#)
 - set_time_main, [75](#)
 - set_time_str, [76](#)
- sys_free_mem
 - mpx_supt.c, [39](#)
 - mpx_supt.h, [45](#)
- sys_req
 - mpx_supt.c, [40](#)
 - mpx_supt.h, [45](#)
- sys_set_free
 - mpx_supt.c, [40](#)
 - mpx_supt.h, [45](#)
- sys_set_malloc
 - mpx_supt.c, [40](#)
 - mpx_supt.h, [45](#)
- tail
 - pcb_queue, [12](#)
- true
 - pcb.c, [94](#)
- unblock_pcb
 - pcb.c, [93](#)
 - pcb.h, [111](#)
- unblock_pcb_main
 - pcb_comm.h, [120](#)
- usage
 - function_name, [10](#)
- VERSION
 - r1.h, [55](#)
- WRITE
 - mpx_supt.h, [44](#)
- WithEcho
 - serial.h, [16](#)
- WithoutEcho
 - serial.h, [16](#)
- YIELD
 - r1.h, [55](#)
- yield_main
 - context.c, [125](#)
 - context.h, [130](#)