

PotatOS

Generated by Doxygen 1.8.11



# Contents

<b>1</b>	<b>Data Structure Index</b>	<b>1</b>
1.1	Data Structures . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Data Structure Documentation</b>	<b>5</b>
3.1	ALARM Struct Reference . . . . .	5
3.1.1	Detailed Description . . . . .	5
3.2	ALIAS Struct Reference . . . . .	6
3.2.1	Detailed Description . . . . .	6
3.3	cmcb Struct Reference . . . . .	6
3.3.1	Detailed Description . . . . .	7
3.4	COMMAND Struct Reference . . . . .	7
3.4.1	Detailed Description . . . . .	7
3.5	control_sequence Struct Reference . . . . .	8
3.5.1	Detailed Description . . . . .	8
3.6	fakelong Struct Reference . . . . .	8
3.6.1	Detailed Description . . . . .	8
3.7	HELP_PAGES Struct Reference . . . . .	9
3.7.1	Detailed Description . . . . .	9
3.8	lmc_b Struct Reference . . . . .	9
3.8.1	Detailed Description . . . . .	10
3.9	node Struct Reference . . . . .	10
3.9.1	Detailed Description . . . . .	11
3.10	pcb Struct Reference . . . . .	11
3.10.1	Detailed Description . . . . .	11
3.11	queue Struct Reference . . . . .	12
3.11.1	Detailed Description . . . . .	12
3.12	time Struct Reference . . . . .	12
3.12.1	Detailed Description . . . . .	13

<b>4 File Documentation</b>	<b>15</b>
4.1 alarmWrangler.h File Reference	15
4.1.1 Detailed Description	16
4.1.2 Function Documentation	16
4.1.2.1 alarmProcess()	16
4.1.2.2 check()	16
4.1.2.3 insertAlarm(char *message, char *dateIn)	16
4.1.2.4 listAlarms()	17
4.1.2.5 removeAlarm(const char *message)	17
4.2 command_handler.c File Reference	17
4.2.1 Detailed Description	18
4.2.2 Macro Definition Documentation	18
4.2.2.1 CMDSIZE	18
4.3 command_handler.h File Reference	18
4.3.1 Detailed Description	18
4.4 commands.h File Reference	18
4.4.1 Detailed Description	20
4.4.2 Function Documentation	20
4.4.2.1 cmd_alarm(char *params)	20
4.4.2.2 cmd_alias(char *params)	20
4.4.2.3 cmd_blockPCB(char *params)	20
4.4.2.4 cmd_clear(char *params)	20
4.4.2.5 cmd_create_pcb(char *params)	21
4.4.2.6 cmd_date(char *params)	21
4.4.2.7 cmd_delete_pcb(char *params)	21
4.4.2.8 cmd_help(char *params)	22
4.4.2.9 cmd_history(char *params)	22
4.4.2.10 cmd_infinity(char *params)	22
4.4.2.11 cmd_loadr3(char *params)	22
4.4.2.12 cmd_resume(char *params)	22

4.4.2.13	<a href="#">cmd_set_priority_pcb(char *params)</a>	23
4.4.2.14	<a href="#">cmd_shutdown(char *params)</a>	23
4.4.2.15	<a href="#">cmd_suspend(char *params)</a>	23
4.4.2.16	<a href="#">cmd_time(char *params)</a>	23
4.4.2.17	<a href="#">cmd_unblock_pcb(char *params)</a>	24
4.4.2.18	<a href="#">cmd_version(char *params)</a>	24
4.4.2.19	<a href="#">cmd_yield(char *params)</a>	24
4.5	<a href="#">commandUtils.h File Reference</a>	24
4.5.1	<a href="#">Detailed Description</a>	26
4.5.2	<a href="#">Macro Definition Documentation</a>	26
4.5.2.1	<a href="#">A_FLAG</a>	26
4.5.2.2	<a href="#">alphanum</a>	26
4.5.2.3	<a href="#">B_FLAG</a>	27
4.5.2.4	<a href="#">C_FLAG</a>	27
4.5.2.5	<a href="#">CMD_SIZE</a>	27
4.5.2.6	<a href="#">D_FLAG</a>	27
4.5.2.7	<a href="#">E_FLAG</a>	27
4.5.2.8	<a href="#">F_FLAG</a>	27
4.5.2.9	<a href="#">G_FLAG</a>	27
4.5.2.10	<a href="#">H_FLAG</a>	27
4.5.2.11	<a href="#">I_FLAG</a>	27
4.5.2.12	<a href="#">J_FLAG</a>	28
4.5.2.13	<a href="#">K_FLAG</a>	28
4.5.2.14	<a href="#">L_FLAG</a>	28
4.5.2.15	<a href="#">M_FLAG</a>	28
4.5.2.16	<a href="#">N_FLAG</a>	28
4.5.2.17	<a href="#">NO_FLAG</a>	28
4.5.2.18	<a href="#">O_FLAG</a>	28
4.5.2.19	<a href="#">P_FLAG</a>	28
4.5.2.20	<a href="#">Q_FLAG</a>	28

4.5.2.21	<a href="#">R_FLAG</a>	28
4.5.2.22	<a href="#">S_FLAG</a>	29
4.5.2.23	<a href="#">T_FLAG</a>	29
4.5.2.24	<a href="#">U_FLAG</a>	29
4.5.2.25	<a href="#">V_FLAG</a>	29
4.5.2.26	<a href="#">W_FLAG</a>	29
4.5.2.27	<a href="#">X_FLAG</a>	29
4.5.2.28	<a href="#">Y_FLAG</a>	29
4.5.2.29	<a href="#">Z_FLAG</a>	29
4.5.3	<a href="#">Function Documentation</a>	29
4.5.3.1	<a href="#">get_command_array()</a>	29
4.5.3.2	<a href="#">get_pvalue(char c)</a>	29
4.5.3.3	<a href="#">search_commands(char *)</a>	30
4.5.3.4	<a href="#">set_flags(char *paramstr, int *flag, int num_aliases,...)</a>	30
4.5.3.5	<a href="#">set_flags_search_alias(char *alias, int num_aliases, ALIAS aliases[])</a>	30
4.6	<a href="#">memEnv.h File Reference</a>	31
4.6.1	<a href="#">Detailed Description</a>	32
4.6.2	<a href="#">Macro Definition Documentation</a>	32
4.6.2.1	<a href="#">HEAP_SIZE</a>	32
4.6.3	<a href="#">Function Documentation</a>	32
4.6.3.1	<a href="#">internal_free(void *data)</a>	32
4.6.3.2	<a href="#">internal_malloc(u32int size)</a>	32
4.6.3.3	<a href="#">mem_init()</a>	32
4.6.3.4	<a href="#">show_mem_state()</a>	33
4.7	<a href="#">memory_wrangler.h File Reference</a>	33
4.7.1	<a href="#">Detailed Description</a>	34
4.7.2	<a href="#">Function Documentation</a>	34
4.7.2.1	<a href="#">get_remaining_free()</a>	34
4.7.2.2	<a href="#">internal_free(void *data)</a>	34
4.7.2.3	<a href="#">internal_malloc(u32int size)</a>	35

4.7.2.4	<code>mem_init()</code> . . . . .	35
4.7.2.5	<code>print_both(cmcb *c, lmcb *l, char *msg)</code> . . . . .	35
4.7.2.6	<code>print_cmcb(cmcb *)</code> . . . . .	35
4.7.2.7	<code>print_lmcb(lmcb *)</code> . . . . .	35
4.7.2.8	<code>show_alloc_mem_state()</code> . . . . .	35
4.7.2.9	<code>show_free_mem_state()</code> . . . . .	35
4.7.2.10	<code>show_mem_state()</code> . . . . .	36
4.8	<code>pcb_constants.h</code> File Reference . . . . .	36
4.8.1	Detailed Description . . . . .	37
4.8.2	Typedef Documentation . . . . .	37
4.8.2.1	<code>node_t</code> . . . . .	37
4.9	<code>pcb_queue.h</code> File Reference . . . . .	38
4.9.1	Detailed Description . . . . .	39
4.9.2	Function Documentation . . . . .	39
4.9.2.1	<code>construct_queue()</code> . . . . .	39
4.9.2.2	<code>dequeue(queue_t *queue)</code> . . . . .	39
4.9.2.3	<code>destruct_queue(queue_t *queue)</code> . . . . .	39
4.9.2.4	<code>enqueue(queue_t *que, pcb_t *data)</code> . . . . .	39
4.9.2.5	<code>priority_enqueue(queue_t *cue, pcb_t *data)</code> . . . . .	40
4.10	<code>pcb_utils.h</code> File Reference . . . . .	40
4.10.1	Detailed Description . . . . .	41
4.10.2	Function Documentation . . . . .	41
4.10.2.1	<code>allocate_pcb(char *)</code> . . . . .	41
4.10.2.2	<code>find_pcb(char *pname)</code> . . . . .	41
4.10.2.3	<code>free_pcb(pcb_t *)</code> . . . . .	42
4.10.2.4	<code>get_blocked_queue()</code> . . . . .	42
4.10.2.5	<code>get_process_class_string(PROCESS_CLASS process_class)</code> . . . . .	42
4.10.2.6	<code>get_process_state_string(PROCESS_STATE process_state)</code> . . . . .	42
4.10.2.7	<code>get_ready_queue()</code> . . . . .	43
4.10.2.8	<code>get_suspended_blocked_queue()</code> . . . . .	43

4.10.2.9	<code>get_suspended_ready_queue()</code>	43
4.10.2.10	<code>init_queue()</code>	43
4.10.2.11	<code>insert_pcb(pcb_t *)</code>	43
4.10.2.12	<code>kill_it___kill_it_all()</code>	44
4.10.2.13	<code>print_pcb_info(const pcb_t *pcb)</code>	44
4.10.2.14	<code>remove_pcb(char *pname)</code>	44
4.10.2.15	<code>setup_pcb(char *, PROCESS_CLASS, int priority)</code>	44
4.11	<code>pcb_wrangler.h</code> File Reference	44
4.11.1	Detailed Description	45
4.12	<code>poll_input.c</code> File Reference	45
4.12.1	Detailed Description	46
4.12.2	Function Documentation	46
4.12.2.1	<code>get_command_history())[11]</code>	46
4.12.2.2	<code>get_history_length()</code>	46
4.12.2.3	<code>get_key()</code>	46
4.12.2.4	<code>input_available()</code>	46
4.12.2.5	<code>memcpy(char *destination, const char *source, int n)</code>	46
4.12.2.6	<code>move_cursor(int n)</code>	47
4.12.2.7	<code>poll_input(char *buffer, int *length)</code>	47
4.12.2.8	<code>print_after_cursor(const char *str)</code>	47
4.12.2.9	<code>wait_for_input(int timeout)</code>	47
4.12.3	Variable Documentation	48
4.12.3.1	<code>control_sequences</code>	48
4.12.3.2	<code>TOLERANCE</code>	48
4.13	<code>poll_input.h</code> File Reference	49
4.13.1	Detailed Description	50
4.13.2	Typedef Documentation	50
4.13.2.1	<code>ControlSequence</code>	50
4.13.3	Function Documentation	50
4.13.3.1	<code>get_command_history())[11]</code>	50



4.13.3.2	<a href="#">get_history_length()</a>	50
4.13.3.3	<a href="#">poll_input(char *buffer, int *length)</a>	50
4.14	<a href="#">procsr3.h File Reference</a>	51
4.15	<a href="#">splash.h File Reference</a>	51
4.15.1	<a href="#">Detailed Description</a>	51
4.16	<a href="#">stdio.c File Reference</a>	52
4.16.1	<a href="#">Detailed Description</a>	52
4.16.2	<a href="#">Function Documentation</a>	52
4.16.2.1	<a href="#">printf(char *form,...)</a>	52
4.16.2.2	<a href="#">puts(char *buff)</a>	53
4.17	<a href="#">stdio.h File Reference</a>	53
4.17.1	<a href="#">Detailed Description</a>	54
4.17.2	<a href="#">Function Documentation</a>	54
4.17.2.1	<a href="#">printf(char *form,...)</a>	54
4.17.2.2	<a href="#">puts(char *buffer)</a>	54
4.18	<a href="#">string.c File Reference</a>	54
4.18.1	<a href="#">Detailed Description</a>	56
4.18.2	<a href="#">Function Documentation</a>	56
4.18.2.1	<a href="#">isdigit(char c)</a>	56
4.18.2.2	<a href="#">itoa(int num, char *str, int base)</a>	56
4.18.2.3	<a href="#">reverse(char *str, int end)</a>	57
4.18.2.4	<a href="#">sprintf(char *buffer, char *format,...)</a>	57
4.18.2.5	<a href="#">sprintf_internal(char *buffer, char *format, va_list valist)</a>	57
4.18.2.6	<a href="#">sprintf_pad_helper(char *buffer, char pad, int fNum, int n, BYTE doAction)</a>	58
4.18.2.7	<a href="#">tolower(char c)</a>	58
4.18.2.8	<a href="#">toupper(char c)</a>	58
4.18.2.9	<a href="#">trim(char *str)</a>	59
4.18.2.10	<a href="#">utoa(u32int num, char *str, int base)</a>	59
4.19	<a href="#">string.h File Reference</a>	59
4.19.1	<a href="#">Detailed Description</a>	60

4.19.2	Function Documentation	60
4.19.2.1	isdigit(char c)	60
4.19.2.2	itoa(int num, char *str, int base)	61
4.19.2.3	reverse(char *str, int j)	61
4.19.2.4	sprintf(char *buffer, char *format,...)	61
4.19.2.5	sprintf_internal(char *buffer, char *format, va_list valist)	62
4.19.2.6	tolower(char c)	62
4.19.2.7	toupper(char c)	62
4.19.2.8	trim(char *str)	63
4.19.2.9	utoa(u32int num, char *str, int base)	63
4.20	time.h File Reference	63
4.20.1	Detailed Description	65
4.20.2	Function Documentation	65
4.20.2.1	bcd_to_decimal(int bcd)	65
4.20.2.2	format_time(char *dest, time_h *t)	65
4.20.2.3	get_current_time()	65
4.20.2.4	set_current_time(time_h time)	66
4.21	utility.c File Reference	66
4.21.1	Detailed Description	67
4.21.2	Function Documentation	67
4.21.2.1	isnullorspace(char test)	67
4.22	utility.h File Reference	67
4.22.1	Detailed Description	67
4.22.2	Function Documentation	68
4.22.2.1	isnullorspace(char test)	68
<b>Index</b>		<b>71</b>

# Chapter 1

## Data Structure Index

### 1.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">ALARM</a>	Struct to hold alarm information . . . . .	5
<a href="#">ALIAS</a>	A struct to hold command aliases . . . . .	6
<a href="#">cmcb</a>	CMCB struct . . . . .	6
<a href="#">COMMAND</a>	A struct to hold commands . . . . .	7
<a href="#">control_sequence</a>	A struct to hold key mappings . . . . .	8
<a href="#">fakelong</a>	Fake 64 bit integer . . . . .	8
<a href="#">HELP_PAGES</a>	A struct to hold help outputs . . . . .	9
<a href="#">lmcb</a>	LMCB struct . . . . .	9
<a href="#">node</a>	One element within the pcb queue . . . . .	10
<a href="#">pcb</a>	Struct that contains all information related to a pcb . . . . .	11
<a href="#">queue</a>	Contains all the data needed to use/modify a queue . . . . .	12
<a href="#">time</a>	A struct to all the time and date elements . . . . .	12



## Chapter 2

# File Index

### 2.1 File List

Here is a list of all documented files with brief descriptions:

<a href="#">alarmWrangler.h</a>	Contains all alarm processes and internal structures . . . . .	15
<a href="#">command_handler.c</a>	The primary command handler for the Operating System . . . . .	17
<a href="#">command_handler.h</a>	The header file for the command handler for the Operating System . . . . .	18
<a href="#">commands.h</a>	The header file for commands.c . . . . .	18
<a href="#">commandUtils.h</a>	Utilites that apply to all command files . . . . .	24
<a href="#">memEnv.h</a>	The header file for the memory env files . . . . .	31
<a href="#">memory_wrangler.h</a>	The header file for the memory control and memory functions . . . . .	33
<a href="#">pcb_constants.h</a>	Contains all shared resources amongst all PCBs . . . . .	36
<a href="#">pcb_queue.h</a>	File to hold all queue functions . . . . .	38
<a href="#">pcb_utils.h</a>	Utility functions for all PCBs . . . . .	40
<a href="#">pcb_wrangler.h</a>	Initiates the creation of all queues . . . . .	44
<a href="#">poll_input.c</a>	The polling input file that allows user input . . . . .	45
<a href="#">poll_input.h</a>	The header file for the polling input . . . . .	49
<a href="#">procsr3.h</a>		51
<a href="#">splash.h</a>	File to hold the splash screen . . . . .	51
<a href="#">stdio.c</a>	Holds all implementation of standard I/O functions . . . . .	52
<a href="#">stdio.h</a>	Holds all prototypes of standard I/O functions . . . . .	53
<a href="#">string.c</a>	Holds all utility functions used to modify strings . . . . .	54

<a href="#">string.h</a>	Holds all utility prototypes used to modify strings . . . . .	<a href="#">59</a>
<a href="#">time.h</a>	The header file for the date and time functions . . . . .	<a href="#">63</a>
<a href="#">utility.c</a>	Holds utility function implementations for this project . . . . .	<a href="#">66</a>
<a href="#">utility.h</a>	Holds utility function prototypes for this project . . . . .	<a href="#">67</a>

## Chapter 3

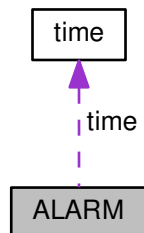
# Data Structure Documentation

### 3.1 ALARM Struct Reference

Struct to hold alarm information.

```
#include <alarmWrangler.h>
```

Collaboration diagram for ALARM:



#### Data Fields

- char \* **message**
- [time\\_h](#) time

#### 3.1.1 Detailed Description

Struct to hold alarm information.

message The alarm message time The alarm execution time

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

- [alarmWrangler.h](#)

## 3.2 ALIAS Struct Reference

A struct to hold command aliases.

```
#include <commandUtils.h>
```

### Data Fields

- char **c**
- char \* **val**

### 3.2.1 Detailed Description

A struct to hold command aliases.

The [ALIAS](#) Struct is a custom struct that is designed to hold aliases for commands

#### Parameters

<i>c</i>	A string that will hold the initial command name
<i>val</i>	A string pointer that will point to the original command name

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

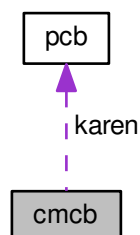
- [commandUtils.h](#)

## 3.3 cmcb Struct Reference

CMCB struct.

```
#include <memEnv.h>
```

Collaboration diagram for cmcb:





## Data Fields

- [pcb\\_t](#) \* **karen**
- unsigned int **size**
- [MEMTYPE](#) type
- char \* **karen**

### 3.3.1 Detailed Description

CMCB struct.

CMCB Struct.

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

- [memEnv.h](#)
- [memory\\_wrangler.h](#)

## 3.4 COMMAND Struct Reference

A struct to hold commands.

```
#include <commandUtils.h>
```

## Data Fields

- char \* **str**
- int(\* **func** )(char \*)
- char \* **alias**

### 3.4.1 Detailed Description

A struct to hold commands.

The [COMMAND](#) Struct is a custom struct that is designed to hold custom commands

#### Parameters

<i>str</i>	A string type to hold the name of the command
<i>CommandPointer</i>	A pointer to a command so that we can pass commands

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

- [commandUtils.h](#)

## 3.5 control\_sequence Struct Reference

A struct to hold key mappings.

```
#include <poll_input.h>
```

### Data Fields

- char **code** [8]
- int **id**

### 3.5.1 Detailed Description

A struct to hold key mappings.

The [control\\_sequence](#) Struct is a custom struct that is designed to hold mappings between control sequence codes used to encode arrow keys. It also holds other special buttons.

#### Parameters

<i>code</i>	The special keyboard code name
<i>id</i>	The keyboard code value

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

- [poll\\_input.h](#)

## 3.6 fakelong Struct Reference

Fake 64 bit integer.

```
#include <time.h>
```

### Data Fields

- unsigned long int **lower**
- unsigned long int **upper**

### 3.6.1 Detailed Description

Fake 64 bit integer.

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

- [time.h](#)

## 3.7 HELP\_PAGES Struct Reference

A struct to hold help outputs.

### Data Fields

- char \* **command\_name**
- char \* **command\_help\_page**

### 3.7.1 Detailed Description

A struct to hold help outputs.

The [COMMAND](#) Struct is a custom struct that is designed to hold custom commands

#### Parameters

<i>str</i>	A string type to hold the name of the command
<i>command_help_page</i>	A string that holds the actual help page

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

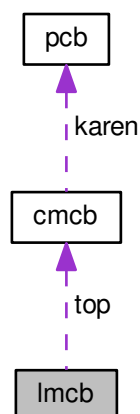
- cmdHelp.c

## 3.8 Imcb Struct Reference

LMCB struct.

```
#include <memEnv.h>
```

Collaboration diagram for Imcb:



## Data Fields

- [cmcb](#) \* **top**
- [MEMTYPE](#) **type**

### 3.8.1 Detailed Description

LMCB struct.

LMCB Struct.

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

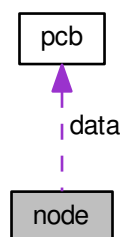
- [memEnv.h](#)
- [memory\\_wrangler.h](#)

## 3.9 node Struct Reference

One element within the pcb queue.

```
#include <pcb_constants.h>
```

Collaboration diagram for node:



## Data Fields

- [pcb\\_t](#) \* **data**
- void \* **next**
- void \* **prev**

### 3.9.1 Detailed Description

One element within the pcb queue.

This allows us to abbreviate code elsewhere... probably

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

- [pcb\\_constants.h](#)

## 3.10 pcb Struct Reference

Struct that contains all information related to a pcb.

```
#include <memEnv.h>
```

### Data Fields

- char \* **process\_name**
- u32int **process\_class**
- u32int **priority**
- u32int **last\_time\_run**
- u32int **state**
- char **stack** [2048]
- unsigned char \* **stacktop**

### 3.10.1 Detailed Description

Struct that contains all information related to a pcb.

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

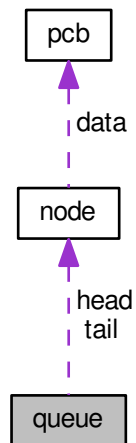
- [memEnv.h](#)
- [pcb\\_constants.h](#)

### 3.11 queue Struct Reference

Contains all the data needed to use/modify a queue.

```
#include <pcb_constants.h>
```

Collaboration diagram for queue:



#### Data Fields

- int **size**
- struct [node](#) \* **head**
- struct [node](#) \* **tail**

#### 3.11.1 Detailed Description

Contains all the data needed to use/modify a queue.

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

- [pcb\\_constants.h](#)

### 3.12 time Struct Reference

A struct to all the time and date elements.

```
#include <time.h>
```

## Data Fields

- int **seconds**
- int **minutes**
- int **hours**
- int **day\_of\_month**
- int **month**
- int **year**

### 3.12.1 Detailed Description

A struct to all the time and date elements.

The time Struct is a custom struct that is designed to hold all the elements necessary for time and date.

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

- [time.h](#)





## Chapter 4

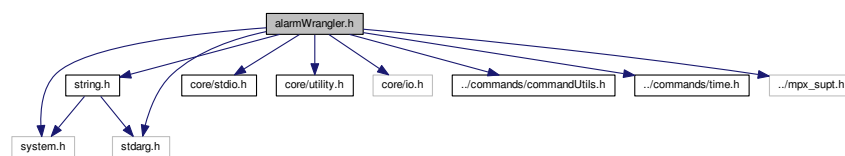
# File Documentation

### 4.1 alarmWrangler.h File Reference

Contains all alarm processes and internal structures.

```
#include <string.h>
#include <core/stdio.h>
#include <core/utility.h>
#include <core/io.h>
#include <stdarg.h>
#include <system.h>
#include "../commands/commandUtils.h"
#include "../commands/time.h"
#include "../mpx_supt.h"
```

Include dependency graph for alarmWrangler.h:



### Data Structures

- struct [ALARM](#)  
*Struct to hold alarm information.*

### Macros

- `#define MAX_ALARM 10`

## Functions

- int `listAlarms` ()  
*List all alarms.*
- int `insertAlarm` (char \*message, char \*dateIn)  
*Insert an alarm into the array of alarms.*
- int `removeAlarm` (const char \*message)  
*Remove an alarm from the array of alarms.*
- int `check` ()  
*Checks to see if any alarm has passed time and needs to send notification.*
- void `alarmProcess` ()  
*The Alarm Process that is initiated in Kmain.*

### 4.1.1 Detailed Description

Contains all alarm processes and internal structures.

### 4.1.2 Function Documentation

#### 4.1.2.1 void alarmProcess ( )

The Alarm Process that is initiated in Kmain.

##### Returns

Nothing

#### 4.1.2.2 int check ( )

Checks to see if any alarm has passed time and needs to send notification.

##### Returns

Int SUCCESS/FAILURE

#### 4.1.2.3 int insertAlarm ( char \* message, char \* dateIn )

Insert an alarm into the array of alarms.

message The new alarms message dateIn The date and or time for the alarm to execute

##### Returns

Int SUCCESS/FAILURE

## 4.1.2.4 int listAlarms ( )

List all alarms.

## Returns

Int SUCCESS/FAILURE

## 4.1.2.5 int removeAlarm ( const char \* message )

Remove an alarm from the array of alarms.

message The name of the alarm, message is name, that you are removing

## Returns

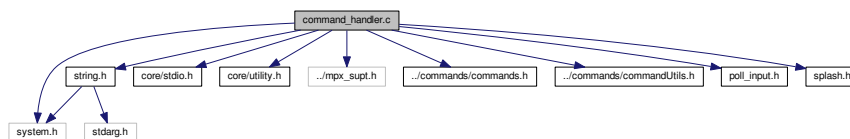
Int SUCCESS/FAILURE

## 4.2 command\_handler.c File Reference

The primary command handler for the Operating System.

```
#include <string.h>
#include <system.h>
#include <core/stdio.h>
#include <core/utility.h>
#include "../mpx_supt.h"
#include "../commands/commands.h"
#include "../commands/commandUtils.h"
#include "poll_input.h"
#include "splash.h"
```

Include dependency graph for command\_handler.c:



### Macros

- #define `CMDSIZE` 100  
The command input buffer.

### Functions

- void `command_handler` ()  
Entry point for the command handler.

### 4.2.1 Detailed Description

The primary command handler for the Operating System.

### 4.2.2 Macro Definition Documentation

#### 4.2.2.1 `#define CMDSIZE 100`

The command input buffer.

This a macro to store the command input buffer Here we can change the ammount of characters we allow to be entered into the command handler at once We currently allow 100 characters

## 4.3 `command_handler.h` File Reference

The header file for the command handler for the Operating System.

### Functions

- void `command_handler` ()  
*Entry point for the command handler.*

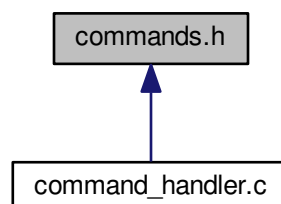
### 4.3.1 Detailed Description

The header file for the command handler for the Operating System.

## 4.4 `commands.h` File Reference

The header file for `commands.c`.

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



## Functions

- int `cmd_help` (char \*params)  
*The help command will show a page to assist users with commands.*
- int `cmd_version` (char \*params)  
*The version command will show the version information.*
- int `cmd_shutdown` (char \*params)  
*shutdown the PotatOS*
- int `cmd_date` (char \*params)  
*The date command will do one of two things. Show the current system date Set a new system date.*
- int `cmd_time` (char \*params)  
*The time command will do one of two things. Show the current system time Set a new system time.*
- int `cmd_clear` (char \*params)  
*clears the screen and sets the pointer at home*
- int `cmd_create_pcb` (char \*params)  
*Create a new pcb.*
- int `cmd_unblock_pcb` (char \*params)  
*Unblock a pcb.*
- int `cmd_blockPCB` (char \*params)  
*command to block PCB by name*
- int `cmd_resume` (char \*params)  
*Resume PCB command.*
- int `cmd_suspend` (char \*params)  
*Suspend PCB command.*
- int `cmd_show_pcb` (char \*params)  
*Show PCB command.*
- int `cmd_show_all_pcbs` (char \*params)  
*Show all PCBs command.*
- int `cmd_show_sus_ready_pcbs` (char \*params)  
*Show all suspended ready PCBs command.*
- int `cmd_show_ready_pcbs` (char \*params)  
*Show ready PCBs command.*
- int `cmd_show_mem` (char \*params)  
*Show the state of memory.*
- int `cmd_show_alloc_mem` (char \*params)  
*Show the state of allocated memory.*
- int `cmd_show_free_mem` (char \*params)  
*Show the state of free memory.*
- int `cmd_show_blocked_pcbs` (char \*params)  
*Show blocked PCBs command.*
- int `cmd_delete_pcb` (char \*params)  
*command to delete PCB by name*
- int `cmd_set_priority_pcb` (char \*params)  
*command to set PCB priority*
- int `cmd_potat` (char \*params)  
*command to draw the potat*
- int `cmd_loadr3` (char \*params)  
*command to load r3 procs*
- int `cmd_yield` (char \*params)  
*command to yield control from commhand*
- int `cmd_alias` (char \*params)

*Command to make an alias for a command.*

- int `cmd_alarm` (char \*params)

*Command to set/delete/list alarms.*

- int `cmd_history` (char \*params)

*Command to show history of commands typed.*

- int `cmd_infinity` (char \*params)

*The infinity alarm for R4.*

#### 4.4.1 Detailed Description

The header file for commands.c.

#### 4.4.2 Function Documentation

##### 4.4.2.1 int `cmd_alarm` ( char \* *params* )

Command to set/delete/list alarms.

Returns

SUCCESS or FAILURE

##### 4.4.2.2 int `cmd_alias` ( char \* *params* )

Command to make an alias for a command.

Returns

SUCCESS

##### 4.4.2.3 int `cmd_blockPCB` ( char \* *params* )

command to block PCB by name

Returns

Success or Failure

##### 4.4.2.4 int `cmd_clear` ( char \* *params* )

clears the screen and sets the pointer at home

## Parameters

<i>params</i>	param string typed by user
---------------	----------------------------

## Returns

SUCCESS or FAILURE

4.4.2.5 int cmd\_create\_pcb ( char \* *params* )

Create a new pcb.

## Parameters

<i>params</i>	string passed from command handler
---------------	------------------------------------

## Returns

SUCCESS or FAILURE

4.4.2.6 int cmd\_date ( char \* *params* )

The date command will do one of two things. Show the current system date Set a new system date.

The date command can be used to query the systems RTC to display the current date. It can also be used to set the systems RTC to a desired date. There is code to check for illegal dates such as Feb 30 on a non leap year.

## Parameters

<i>params</i>	param string typed by user
---------------	----------------------------

## Returns

The current system date

## Warning

The RTC only allows dates between 1700-2999

4.4.2.7 int cmd\_delete\_pcb ( char \* *params* )

command to delete PCB by name

## Returns

Success if the PCB was removed, failure for anything else

#### 4.4.2.8 int cmd\_help ( char \* *params* )

The help command will show a page to assist users with commands.

The help command can be called to do one of two things List all the commands that have help pages Request a help page for a certain command

##### Parameters

<i>params</i>	param string typed by user
---------------	----------------------------

##### Returns

A help page

#### 4.4.2.9 int cmd\_history ( char \* *params* )

Command to show history of commands typed.

##### Returns

SUCCESS or FAILURE

#### 4.4.2.10 int cmd\_infinity ( char \* *params* )

The infinity alarm for R4.

##### Returns

SUCCESS or FAILURE

#### 4.4.2.11 int cmd\_loadr3 ( char \* *params* )

command to load r3 procs

##### Returns

SUCCESS

#### 4.4.2.12 int cmd\_resume ( char \* *params* )

Resume PCB command.

##### Returns

SUCCESS or FAILURE



#### 4.4.2.13 int cmd\_set\_priority\_pcb ( char \* *params* )

command to set PCB priority

##### Returns

Success if the PCB priority was updated, failure for anything else

#### 4.4.2.14 int cmd\_shutdown ( char \* *params* )

shutdown the PotatOS

##### Parameters

<i>params</i>	string passed from command handler
---------------	------------------------------------

##### Returns

SUCCESS or FAILURE

#### 4.4.2.15 int cmd\_suspend ( char \* *params* )

Suspend PCB command.

##### Returns

SUCCESS or FAILURE

#### 4.4.2.16 int cmd\_time ( char \* *params* )

The time command will do one of two things. Show the current system time Set a new system time.

The time command can be used to query the systems RTC to display the current time. It can also be used to set the systems RTC to a desired time. There is code to check for illegal times.

##### Parameters

<i>params</i>	param string typed by user
---------------	----------------------------

##### Returns

The current system time

##### Note

The time is kept in 24 hour time

#### 4.4.2.17 int cmd\_unblock\_pcb ( char \* *params* )

Unblock a pcb.

##### Parameters

<i>params</i>	string passed from command handler
---------------	------------------------------------

##### Returns

SUCCESS or FAILURE

#### 4.4.2.18 int cmd\_version ( char \* *params* )

The version command will show the version information.

The version command can be called to display the version information. The shortened return will just show the short version. The long return will include the current module, the version, and the contributing developers

##### Parameters

<i>params</i>	param string typed by user
---------------	----------------------------

##### Returns

A version page

#### 4.4.2.19 int cmd\_yield ( char \* *params* )

command to yield control from commhand

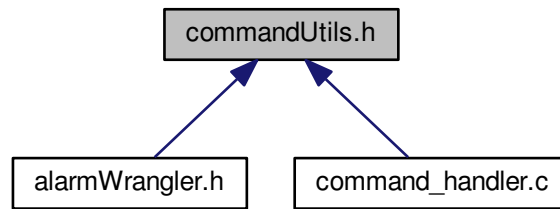
##### Returns

SUCCESS

## 4.5 commandUtils.h File Reference

Utilites that apply to all command files.

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



## Data Structures

- struct [ALIAS](#)  
*A struct to hold command aliases.*
- struct [COMMAND](#)  
*A struct to hold commands.*

## Macros

- #define [CMDSIZE](#) 100  
*The command input buffer.*
- #define [SUCCESS](#) 1  
*Macro to return a 0 on success.*
- #define [FAILURE](#) 0  
*Macro to return a -1 on failure.*
- #define [MAXPARAMCOUNT](#) 10  
*The maximum parameters allowed per command.*
- #define [A\\_FLAG](#) (1 << 0)
- #define [B\\_FLAG](#) (1 << 1)
- #define [C\\_FLAG](#) (1 << 2)
- #define [D\\_FLAG](#) (1 << 3)
- #define [E\\_FLAG](#) (1 << 4)
- #define [F\\_FLAG](#) (1 << 5)
- #define [G\\_FLAG](#) (1 << 6)
- #define [H\\_FLAG](#) (1 << 7)
- #define [I\\_FLAG](#) (1 << 8)
- #define [J\\_FLAG](#) (1 << 9)
- #define [K\\_FLAG](#) (1 << 10)
- #define [L\\_FLAG](#) (1 << 11)
- #define [M\\_FLAG](#) (1 << 12)
- #define [N\\_FLAG](#) (1 << 13)
- #define [O\\_FLAG](#) (1 << 14)
- #define [P\\_FLAG](#) (1 << 15)
- #define [Q\\_FLAG](#) (1 << 16)
- #define [R\\_FLAG](#) (1 << 17)
- #define [S\\_FLAG](#) (1 << 18)

- `#define T_FLAG (1 << 19)`
- `#define U_FLAG (1 << 20)`
- `#define V_FLAG (1 << 21)`
- `#define W_FLAG (1 << 22)`
- `#define Y_FLAG (1 << 23)`
- `#define X_FLAG (1 << 24)`
- `#define Z_FLAG (1 << 25)`
- `#define NO_FLAG (1 << 26)`
- `#define alphanum(c) (('a' <= c && c <= 'z') ? c - 'a' : c - 'A')`  
*A helper macro that will take a letter and return its integer equivalent.*

## Functions

- `int set_flags (char *paramstr, int *flag, int num_aliases,...)`  
*Sets flags based on param string, flags and num aliases.*
- `char * get_pvalue (char c)`  
*Gets value of specific flag.*
- `char set_flags_search_alias (char *alias, int num_aliases, ALIAS aliases[])`  
*Used as a helper function for set\_flags.*
- `COMMAND * get_command_array ()`  
*returns pointer to the command array*
- `COMMAND * search_commands (char *)`  
*search commands with a command name*
- `int showAll ()`

## Variables

- `char gparamstr [CMD_SIZE]`  
*A string to hold the command input up to the max command size.*
- `char * gparams [27]`  
*Will hold all the string pointers.*

### 4.5.1 Detailed Description

Utilites that apply to all command files.

### 4.5.2 Macro Definition Documentation

#### 4.5.2.1 `#define A_FLAG (1 << 0)`

`cmd_help` flags A flag binary bit shift macro

#### 4.5.2.2 `#define alphanum( c ) (('a' <= c && c <= 'z') ? c - 'a' : c - 'A')`

A helper macro that will take a letter and return its integer equivalent.

This is a helper macro that is used in `set_flags` and `get_gparams`. It takes in character and return the integer equivalent of that character.

## Parameters

c	The character to be returned as an int
---	--

4.5.2.3 `#define B_FLAG (1 << 1)`

B flag binary bit shift macro

4.5.2.4 `#define C_FLAG (1 << 2)`

C flag binary bit shift macro

4.5.2.5 `#define CMDSIZE 100`

The command input buffer.

This a macro to store the command input buffer. Here we can change the amount of characters we allow to be entered into the command handler at once. We currently allow 100 characters.

4.5.2.6 `#define D_FLAG (1 << 3)`

D flag binary bit shift macro

4.5.2.7 `#define E_FLAG (1 << 4)`

E flag binary bit shift macro

4.5.2.8 `#define F_FLAG (1 << 5)`

F flag binary bit shift macro

4.5.2.9 `#define G_FLAG (1 << 6)`

G flag binary bit shift macro

4.5.2.10 `#define H_FLAG (1 << 7)`

H flag binary bit shift macro

4.5.2.11 `#define I_FLAG (1 << 8)`

I flag binary bit shift macro

**4.5.2.12 #define J\_FLAG (1 << 9)**

J flag binary bit shift macro

**4.5.2.13 #define K\_FLAG (1 << 10)**

K flag binary bit shift macro

**4.5.2.14 #define L\_FLAG (1 << 11)**

L flag binary bit shift macro

**4.5.2.15 #define M\_FLAG (1 << 12)**

M flag binary bit shift macro

**4.5.2.16 #define N\_FLAG (1 << 13)**

N flag binary bit shift macro

**4.5.2.17 #define NO\_FLAG (1 << 26)**

NO flag binary bit shift macro

**4.5.2.18 #define O\_FLAG (1 << 14)**

O flag binary bit shift macro

**4.5.2.19 #define P\_FLAG (1 << 15)**

P flag binary bit shift macro

**4.5.2.20 #define Q\_FLAG (1 << 16)**

Q flag binary bit shift macro

**4.5.2.21 #define R\_FLAG (1 << 17)**

R flag binary bit shift macro

#### 4.5.2.22 #define S\_FLAG (1 << 18)

S flag binary bit shift macro

#### 4.5.2.23 #define T\_FLAG (1 << 19)

T flag binary bit shift macro

#### 4.5.2.24 #define U\_FLAG (1 << 20)

U flag binary bit shift macro

#### 4.5.2.25 #define V\_FLAG (1 << 21)

V flag binary bit shift macro

#### 4.5.2.26 #define W\_FLAG (1 << 22)

W flag binary bit shift macro

#### 4.5.2.27 #define X\_FLAG (1 << 24)

X flag binary bit shift macro

#### 4.5.2.28 #define Y\_FLAG (1 << 23)

Y flag binary bit shift macro

#### 4.5.2.29 #define Z\_FLAG (1 << 25)

Z flag binary bit shift macro

### 4.5.3 Function Documentation

#### 4.5.3.1 COMMAND\* get\_command\_array ( )

returns pointer to the command array

Returns

pointer to the command array

#### 4.5.3.2 char\* get\_pvalue ( char c )

Gets value of specific flag.

Usage: get\_pvalue('a');

**Parameters**

<i>c</i>	character of flag to get the value from
----------	---

**Returns**

value after the flag specified

**4.5.3.3 `COMMAND* search_commands ( char * cmd )`**

search commands with a command name

**Returns**

pointer to a [COMMAND](#)

search commands with a command name

**Parameters**

<i>cmd</i>	cmd typed by user
------------	-------------------

**4.5.3.4 `int set_flags ( char * paramstr, int * flag, int num_aliases, ... )`**

Sets flags based on param string, flags and num aliases.

Usage: `set_flags(paramstr,&flag,5, 'a',"alpha", 'b',"bravo", 'f',"foxtrot", 'g',"golf", 'r',"whiskey" )`

**Parameters**

<i>paramstr</i>	string that each command gets. Typed by the user
<i>flag</i>	pointer to integer flag
<i>num_aliases</i>	number of aliases specified

**Returns**

success or failure

**Note**

`num_aliases` must be the exact number of parameters. In the example, 5

**4.5.3.5 `char set_flags_search_alias ( char * alias, int num_aliases, ALIAS aliases[ ] )`**

Used as a helper function for `set_flags`.



## Parameters

<i>alias</i>	alias to search for in aliases
<i>num_aliases</i>	number of aliases in aliases
<i>aliases</i>	array of ALIASes to search through

## Returns

character of flag that it found

## 4.6 memEnv.h File Reference

The header file for the memory env files.

### Data Structures

- struct [pcb](#)  
*Struct that contains all information related to a pcb.*
- struct [cmcb](#)  
*CMCB struct.*
- struct [lmcb](#)  
*LMCB struct.*

### Macros

- #define [HEAP\\_SIZE](#) 1000
- #define [MCB\\_PADDING](#) (sizeof([cmcb](#)) + sizeof([lmcb](#)))  
*Macro of the MCB\_Padding.*

### Typedefs

- typedef unsigned int **u32int**
- typedef struct [pcb](#) [pcb\\_t](#)  
*Struct that contains all information related to a pcb.*

### Enumerations

- enum **MEMTYPE** { **FREE**, **ALIVE**, **FREE**, **ALIVE** }

## Functions

- void \* [mem\\_init](#) ()  
*Memory init function.*
- void \* [internal\\_malloc](#) (u32int size)  
*Internal memory allocation function.*
- int [internal\\_free](#) (void \*data)  
*Internal free memory function.*
- u32int [get\\_remaining\\_free](#) ()
- void [print\\_cmcb](#) (cmcb \*)  
*prints out cmcb info*
- void [print\\_lmcb](#) (lmcb \*)  
*prints out lmcb info*
- void [print\\_both](#) (cmcb \*c, lmcb \*l, char \*msg)  
*prints out cmcb info and lmcb info with a message*
- void [show\\_mem\\_state](#) ()

### 4.6.1 Detailed Description

The header file for the memory env files.

### 4.6.2 Macro Definition Documentation

#### 4.6.2.1 #define HEAP\_SIZE 1000

@ brief Macro of the heap size

### 4.6.3 Function Documentation

#### 4.6.3.1 int internal\_free ( void \* data )

Internal free memory function.

data The data to be freed

#### 4.6.3.2 void\* internal\_malloc ( u32int size )

Internal memory allocation function.

size The size of the memory to be allocated

#### 4.6.3.3 void\* mem\_init ( )

Memory init function.

None

## 4.6.3.4 void show\_mem\_state ( )

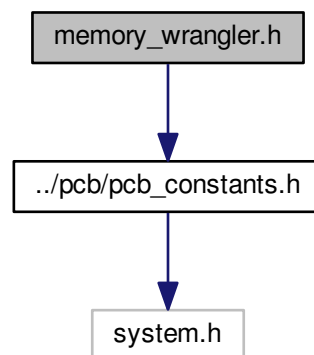
show the entire state of memory visualized

## 4.7 memory\_wrangler.h File Reference

The header file for the memory control and memory functions.

```
#include "../pcb/pcb_constants.h"
```

Include dependency graph for memory\_wrangler.h:



### Data Structures

- struct `cmcb`  
*CMCB struct.*
- struct `lmcb`  
*LMCB struct.*

### Macros

- #define `HEAP_SIZE` 54301  
*The overall Heap Size.*
- #define `MCB_PADDING` (sizeof(`cmcb`) + sizeof(`lmcb`))  
*MCB\_PADDING Macro.*

### Enumerations

- enum `MEMTYPE` { `FREE`, `ALIVE`, `FREE`, `ALIVE` }  
*Enum to specify weather memory is free or alive.*

## Functions

- void \* [mem\\_init](#) ()  
*Memory init function.*
- void \* [internal\\_malloc](#) (u32int size)  
*Internal memory allocation function.*
- void \* **internal\_malloc\_named** (u32int size, char \*karen)
- int [internal\\_free](#) (void \*data)  
*Internal memory free function.*
- u32int [get\\_remaining\\_free](#) ()  
*Get the remaining free memory.*
- void [print\\_cmcb](#) (cmcb \*)  
*prints out cmcb info*
- void [print\\_lmcb](#) (lmcb \*)  
*prints out lmcb info*
- void [print\\_both](#) (cmcb \*c, lmcb \*l, char \*msg)  
*prints out cmcb info and lmcb info with a message*
- void [show\\_mem\\_state](#) ()
- void [show\\_free\\_mem\\_state](#) ()  
*Function to show the free memory state.*
- void [show\\_alloc\\_mem\\_state](#) ()  
*Function to show the allocated memory state.*

### 4.7.1 Detailed Description

The header file for the memory control and memory functions.

### 4.7.2 Function Documentation

#### 4.7.2.1 u32int get\_remaining\_free ( )

Get the remaining free memory.

None

#### 4.7.2.2 int internal\_free ( void \* data )

Internal memory free function.

data The data that needs to be freed

Internal memory free function.

data The data to be freed

#### 4.7.2.3 void\* internal\_malloc ( u32int *size* )

Internal memory allocation function.

size The size of the memory that needs to be allocated

size The size of the memory to be allocated

#### 4.7.2.4 void\* mem\_init ( )

Memory init function.

None

#### 4.7.2.5 void print\_both ( cmcb \* *c*, lmcb \* *l*, char \* *msg* )

prints out cmcb info and lmcb info with a message

c The CMCB requested l The LMCB requested msg The message to be displayed

#### 4.7.2.6 void print\_cmcb ( cmcb \* )

prints out cmcb info

cmcb The requested CMCB

#### 4.7.2.7 void print\_lmcb ( lmcb \* )

prints out lmcb info

lmcb The requested LMCB

#### 4.7.2.8 void show\_alloc\_mem\_state ( )

Function to show the allocated memory state.

None

#### 4.7.2.9 void show\_free\_mem\_state ( )

Function to show the free memory state.

None

#### 4.7.2.10 void show\_mem\_state ( )

show the entire state of memory visualized

None

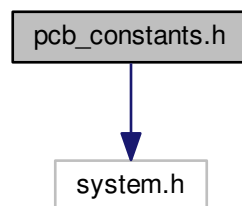
show the entire state of memory visualized

## 4.8 pcb\_constants.h File Reference

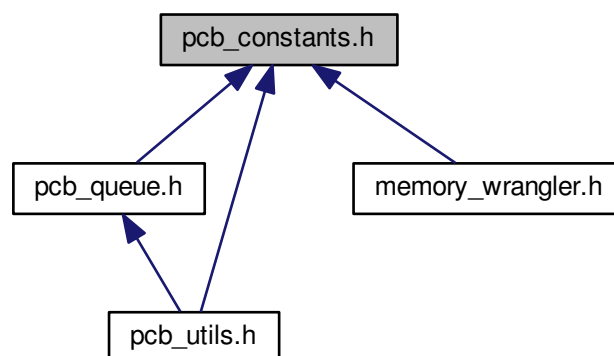
Contains all shared resources amongst all PCBs.

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

Include dependency graph for pcb\_constants.h:



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



## Data Structures

- struct [pcb](#)  
*Struct that contains all information related to a pcb.*
- struct [node](#)  
*One element within the pcb queue.*
- struct [queue](#)  
*Contains all the data needed to use/modify a queue.*

## Macros

- `#define PCB_CONSTANTS_H`
- `#define DEFAULT_PRIORITY 314159265`

## Typedefs

- typedef struct [pcb](#) [pcb\\_t](#)  
*Struct that contains all information related to a pcb.*
- typedef struct [node](#) [node\\_t](#)  
*One element within the pcb queue.*
- typedef struct [queue](#) [queue\\_t](#)  
*Contains all the data needed to use/modify a queue.*

## Enumerations

- enum [PROCESS\\_CLASS](#) { **SYSTEM**, **APPLICATION** }  
*Contains all possible process classes.*
- enum [PROCESS\\_STATE](#) { **RUNNING**, **READY**, **BLOCKED**, **SUSPENDED\_READY**, **SUSPENDED\_BLOCKED** }  
*Contains all possible process states.*

### 4.8.1 Detailed Description

Contains all shared resources amongst all PCBs.

### 4.8.2 Typedef Documentation

#### 4.8.2.1 typedef struct node node\_t

One element within the pcb queue.

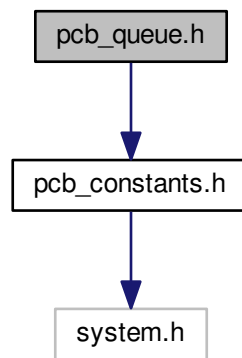
This allows us to abbreviate code elsewhere... probably

## 4.9 pcb\_queue.h File Reference

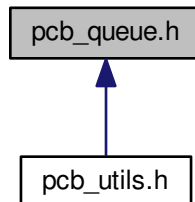
File to hold all queue functions.

```
#include "pcb_constants.h"
```

Include dependency graph for pcb\_queue.h:



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



### Functions

- void `enqueue` (`queue_t` \*que, `pcb_t` \*data)  
*Appends an element to the end of the queue.*
- void `priority_enqueue` (`queue_t` \*cue, `pcb_t` \*data)  
*Appends an element onto the tail of the given queue.*
- `pcb_t` \* `dequeue` (`queue_t` \*queue)  
*Takes the PCB off of the head of the queue and moves head.*
- `queue_t` \* `construct_queue` ()  
*Creates a queue.*
- void `destruct_queue` (`queue_t` \*queue)  
*Destructs a queue and its contents.*



### 4.9.1 Detailed Description

File to hold all queue functions.

### 4.9.2 Function Documentation

#### 4.9.2.1 `queue_t* construct_queue ( )`

Creates a queue.

Creates and allocates a queue and sets all variables correctly for initialization.

##### Returns

A pointer to a newly constructed queue.

#### 4.9.2.2 `pcb_t* dequeue ( queue_t * queue )`

Takes the PCB off of the head of the queue and moves head.

Takes care of freeing the node returns the head PCB

##### Parameters

<i>queue</i>	A pointer to a queue that you want to dequeue the first element from.
--------------	---

##### Returns

A pointer to the dequeued PCB

#### 4.9.2.3 `void destruct_queue ( queue_t * queue )`

Destructs a queue and its contents.

De-allocates a queue and all of the elements within it. This function exists to avoid memory leaks.

##### Parameters

<i>queue</i>	A pointer to the queue you wish to deallocate.
--------------	--

#### 4.9.2.4 `void enqueue ( queue_t * cue, pcb_t * data )`

Appends an element to the end of the queue.

This function searches for the end of the queue and, adds the specified pcb to the end of the list.

## Parameters

<i>que</i>	A pointer to a queue that the PCB will be inserted into.
<i>data</i>	A pointer to the PCB to insert into the queue.

## 4.9.2.5 void priority\_enqueue ( queue\_t \* que, pcb\_t \* data )

Appends an element onto the tail of the given queue.

This function inserts the given data (a PCB) into the queue according to priority.

## Parameters

<i>que</i>	A pointer to the queue to insert the data into.
<i>data</i>	a pointer to the PCB that is to be inserted.

## Note

The data must point to a pcb with a valid priority.

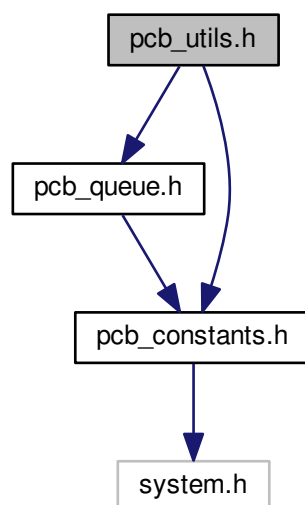
## 4.10 pcb\_utils.h File Reference

Utility functions for all PCBs.

```
#include "pcb_queue.h"
```

```
#include "pcb_constants.h"
```

Include dependency graph for pcb\_utils.h:



## Functions

- void **showReadyQueue** ()
- [pcb\\_t \\*](#) **allocate\_pcb** (char \*)  
*simply allocates space for a pcb and returns that pointer*
- [pcb\\_t \\*](#) **setup\_pcb** (char \*, [PROCESS\\_CLASS](#), int priority)  
*Setup a new PCB and enques that PCB.*
- int **free\_pcb** ([pcb\\_t \\*](#))  
*Frees the space for a pcb.*
- [pcb\\_t \\*](#) **find\_pcb** (char \*pname)  
*Finds a PCB in all queues.*
- int **insert\_pcb** ([pcb\\_t \\*](#))  
*Inserts a PCB.*
- [pcb\\_t \\*](#) **remove\_pcb** (char \*pname)  
*Removes a PCB by process name.*
- void **init\_queue** ()  
*Initializes queues.*
- [queue\\_t \\*](#) **get\_ready\_queue** ()  
*Getter function for the ready queue.*
- [queue\\_t \\*](#) **get\_blocked\_queue** ()  
*Getter function for the blocked queue.*
- [queue\\_t \\*](#) **get\_suspended\_ready\_queue** ()  
*Getter function for the suspended ready queue.*
- [queue\\_t \\*](#) **get\_suspended\_blocked\_queue** ()  
*Getter function for the suspended blocked queue.*
- void **print\_pcb\_info** (const [pcb\\_t \\*](#)pcb)  
*Prints the passed pcb's info in a stylized manner.*
- const char \* **get\_process\_class\_string** ([PROCESS\\_CLASS](#) process\_class)  
*Returns a string corresponding to the process class enum.*
- const char \* **get\_process\_state\_string** ([PROCESS\\_STATE](#) process\_state)  
*Returns a string corresponding to the process state enum.*
- void **kill\_it\_\_kill\_it\_all** ()

### 4.10.1 Detailed Description

Utility functions for all PCBs.

### 4.10.2 Function Documentation

#### 4.10.2.1 [pcb\\_t\\*](#) **allocate\_pcb** ( char \* *pname* )

simply allocates space for a pcb and returns that pointer

#### Returns

pointer to allocated pcb

#### 4.10.2.2 [pcb\\_t\\*](#) **find\_pcb** ( char \* *name* )

Finds a PCB in all queues.

Searches through all the system PCB queues to find a PCB with the specified process name given by pname.

**Parameters**

<i>pname</i>	The name of the process you want to find the PCB of.
--------------	--

**Returns**

A pointer to the pcb with the specified name or 'NULL' for not found.

**4.10.2.3 int free\_pcb ( pcb\_t \* p )**

Frees the space for a pcb.

**Returns**

Success or failure

Frees the space for a pcb.

**Returns**

Success or failure

**4.10.2.4 queue\_t\* get\_blocked\_queue ( )**

Getter function for the blocked queue.

**Returns**

A pointer to the blocked queue

**4.10.2.5 const char\* get\_process\_class\_string ( PROCESS\_CLASS process\_class )**

Returns a string corresponding to the process class enum.

**Parameters**

<i>process_class</i>	An enumeration variant of the PROCESS_CLASS enum
----------------------	--

**Returns**

A char pointer that is the enumeration name

**4.10.2.6 const char\* get\_process\_state\_string ( PROCESS\_STATE process\_state )**

Returns a string corresponding to the process state enum.

## Parameters

<i>process_state</i>	An enumeration variant of the PROCESS_STATE enum
----------------------	--

## Returns

A char pointer that is the enumeration name

4.10.2.7 `queue_t* get_ready_queue ( )`

Getter function for the ready queue.

## Returns

A pointer to the ready queue

4.10.2.8 `queue_t* get_suspended_blocked_queue ( )`

Getter function for the suspended blocked queue.

## Returns

A pointer to the suspended blocked queue

4.10.2.9 `queue_t* get_suspended_ready_queue ( )`

Getter function for the suspended ready queue.

## Returns

A pointer to the suspended ready queue

4.10.2.10 `void init_queue ( )`

Initializes queues.

None

4.10.2.11 `int insert_pcb ( pcb_t * )`

Inserts a PCB.

`pcb_t` The PCB to be inserted

#### 4.10.2.12 void kill\_it\_\_kill\_it\_all ( )

Kills the ready queue after commhand death

#### 4.10.2.13 void print\_pcb\_info ( const pcb\_t \* pcb )

Prints the passed pcb's info in a stylized manner.

Example output

Process Name: PROC1 Process Class: system State: ready Priority: 1 Suspended: true

#### 4.10.2.14 pcb\_t\* remove\_pcb ( char \* name )

Removes a PCB by process name.

Searches through all system queues to find the PCB with the given name.

##### Parameters

<i>pname</i>	Name of the process you want to remove.
--------------	---

##### Returns

Success condition (boolean).

#### 4.10.2.15 pcb\_t\* setup\_pcb ( char \* pname, PROCESS\_CLASS pclass, int priority )

Setup a new PCB and enques that PCB.

PROCESS\_CLASS The Process Class of the new PCB priority The priority of the new PCB

Setup a new PCB and enques that PCB.

##### Returns

Success if the PCB was created, failure for anything else

## 4.11 pcb\_wrangler.h File Reference

Initiates the creation of all queues.

### Functions

- void **init\_process\_queues** ()

### 4.11.1 Detailed Description

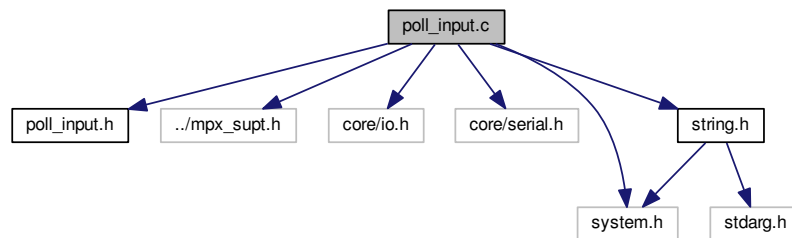
Initiates the creation of all queues.

## 4.12 poll\_input.c File Reference

The polling input file that allows user input.

```
#include "poll_input.h"
#include "../mpx_supt.h"
#include <core/io.h>
#include <core/serial.h>
#include <string.h>
#include <system.h>
```

Include dependency graph for poll\_input.c:



### Macros

- `#define BUFFER_LEN 100`

### Functions

- `int input_available ()`  
*Checks for input on COM1.*
- `int wait_for_input (int timeout)`  
*Loops N times to check for input.*
- `int get_key ()`  
*Receives a key press, whether a full control sequence or simple character.*
- `void move_cursor (int n)`  
*Moves the cursor n characters.*
- `void print_after_cursor (const char *str)`  
*Prints text after the cursor without moving the cursor.*
- `void delete_after_cursor ()`  
*Deletes all text after the cursor.*
- `void memcpy (char *destination, const char *source, int n)`  
*Copies n bytes from one buffer to another.*
- `int get_history_length ()`  
*Getter function for the command history length.*
- `char(* get_command_history ()) [11]`  
*Getter function for the command history array.*
- `int poll_input (char *buffer, int *length)`  
*Polls COM1 for input and puts it into buffer.*

## Variables

- const `ControlSequence control_sequences []`  
*A collection of known control sequences and what they mean.*
- const int `TOLERANCE = 300`  
*Maximum amount of NOP cycles that can occur between two inputs from the same control sequence.*
- const char `ESC = '\x1B'`  
*The escape character.*
- const int `ALT_FLAG = 1 << 8`  
*The bit indicating a key from `get_key` was held with the ALT key.*

### 4.12.1 Detailed Description

The polling input file that allows user input.

### 4.12.2 Function Documentation

#### 4.12.2.1 `char(* get_command_history ( ))[11]`

Getter function for the command history array.

##### Returns

A char pointer to char pointers; an array of commands.

#### 4.12.2.2 `int get_history_length ( )`

Getter function for the command history length.

##### Returns

An int of how long the history currently is. Maxes out at 10

#### 4.12.2.3 `int get_key ( )`

Receives a key press, whether a full control sequence or simple character.

Calls `inb(COM1)` to receive bytes. If a control sequence is detected then it is parsed according to the `control_sequences` array. If it was just a simple character like the A key. Then the char is sent as an int. Arrow keys and other control sequences are special numbers higher than 255 to differentiate themselves from the regular characters. The `KEYS` enum shows the special characters

##### Returns

Returns an int corresponding to the key

#### 4.12.2.4 `int input_available ( )`

Checks for input on COM1.

##### Returns

1 if input is available, 0 if it isn't.

#### 4.12.2.5 `void memcpy ( char * destination, const char * source, int n )`

Copies n bytes from one buffer to another.



## Parameters

<i>destination</i>	Where to copy the bytes to.
<i>source</i>	Where to copy the bytes from.
<i>n</i>	How many bytes to copy.

4.12.2.6 void move\_cursor ( int *n* )

Moves the cursor *n* characters.

## Parameters

<i>n</i>	How many characters to move the character, can be negative.
----------	---

4.12.2.7 int poll\_input ( char \* *buffer*, int \* *length* )

Polls COM1 for input and puts it into buffer.

An internal history is kept so the user can go through past commands

## Parameters

<i>buffer</i>	a pointer to the buffer to put the user input into
<i>length</i>	a pointer to the length of buffer, will be modified to length of input

## Returns

function status

4.12.2.8 void print\_after\_cursor ( const char \* *str* )

Prints text after the cursor without moving the cursor.

## Parameters

<i>str</i>	A pointer to the string to print out
------------	--------------------------------------

4.12.2.9 int wait\_for\_input ( int *timeout* )

Loops *N* times to check for input.

Calls NOP in a while loop at most *timeout* times until it returns.

**Parameters**

<i>timeout</i>	How many times to loop before we give up
----------------	--

**Returns**

how many times were left in the timeout

**4.12.3 Variable Documentation****4.12.3.1 const ControlSequence control\_sequences[]****Initial value:**

```
= {
    {"A", UP_ARROW},
    {"B", DOWN_ARROW},
    {"C", RIGHT_ARROW},
    {"D", LEFT_ARROW},

    {"1~", HOME},
    {"2~", INSERT},
    {"3~", DELETE},
    {"4~", END},
    {"5~", PAGE_DOWN},
    {"6~", PAGE_UP},

    {"[A", F1},
    {"[B", F2},
    {"[C", F3},
    {"[D", F4},
    {"[E", F5},
    {"17~", F6},
    {"18~", F7},
    {"19~", F8},
    {"20~", F9},
    {"21~", F10},
    {"23~", F11},
    {"24~", F12},

    {"", 0}
}
```

A collection of known control sequences and what they mean.

Control sequences are used to encode special input keys from the keyboard that aren't just a one byte character. They start with ESCAPE [ and then a series of characters. This array holds the series of characters that comes after the bracket, along with the corresponding keyboard input. The keyboard inputs are from the KEYS enum.

**4.12.3.2 const int TOLERANCE = 300**

Maximum amount of NOP cycles that can occur between two inputs from the same control sequence.

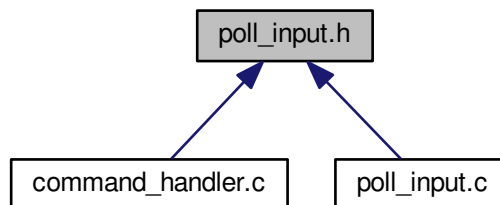
**Note**

This is entirely arbitrary and was just increased until things stopped being weird.

## 4.13 poll\_input.h File Reference

The header file for the polling input.

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



### Data Structures

- struct [control\\_sequence](#)  
*A struct to hold key mappings.*

### Typedefs

- typedef struct [control\\_sequence](#) [ControlSequence](#)  
*A struct to hold key mappings.*

### Enumerations

- enum **KEYS** {  
**BASE = 1024, UP\_ARROW, DOWN\_ARROW, RIGHT\_ARROW,**  
**LEFT\_ARROW, HOME, INSERT, DELETE,**  
**END, PAGE\_UP, PAGE\_DOWN, F1,**  
**F2, F3, F4, F5,**  
**F6, F7, F8, F9,**  
**F10, F11, F12 }**

### Functions

- int [get\\_history\\_length](#) ()  
*Getter function for the command history length.*
- char(\* [get\\_command\\_history](#) ()) [11]  
*Getter function for the command history array.*
- int [poll\\_input](#) (char \*buffer, int \*length)  
*Polls COM1 for input and puts it into buffer.*

### 4.13.1 Detailed Description

The header file for the polling input.

### 4.13.2 Typedef Documentation

#### 4.13.2.1 typedef struct control\_sequence ControlSequence

A struct to hold key mappings.

The [control\\_sequence](#) Struct is a custom struct that is designed to hold mappings between control sequence codes used to encode arrow keys. It also holds other special buttons.

##### Parameters

<i>code</i>	The special keyboard code name
<i>id</i>	The keyboard code value

### 4.13.3 Function Documentation

#### 4.13.3.1 char(\* get\_command\_history ( ))[11]

Getter function for the command history array.

##### Returns

A char pointer to char pointers; an array of commands.

#### 4.13.3.2 int get\_history\_length ( )

Getter function for the command history length.

##### Returns

An int of how long the history currently is. Maxes out at 10

#### 4.13.3.3 int poll\_input ( char \* *buffer*, int \* *length* )

Polls COM1 for input and puts it into buffer.

An internal history is kept so the user can go through past commands

##### Parameters

<i>buffer</i>	a pointer to the buffer to put the user input into
<i>length</i>	a pointer to the length of buffer, will be modified to length of input

## Returns

function status

## 4.14 procsr3.h File Reference

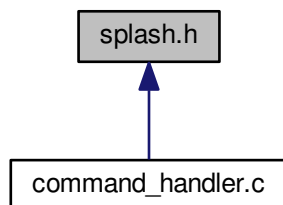
## Functions

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

## 4.15 splash.h File Reference

File to hold the splash screen.

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



## Functions

- void **draw\_splash** ()  
*draw the splash screen*

### 4.15.1 Detailed Description

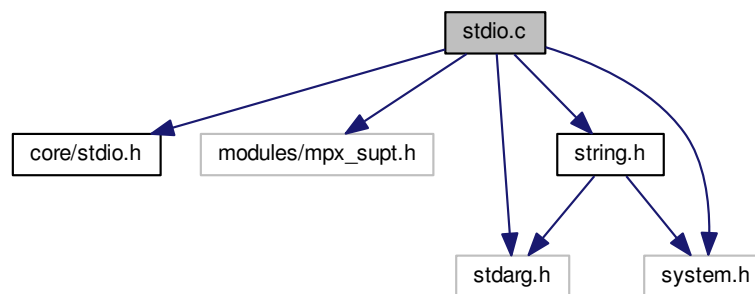
File to hold the splash screen.

## 4.16 stdio.c File Reference

Holds all implementation of standard I/O functions.

```
#include <core/stdio.h>
#include <modules/mpx_supt.h>
#include <stdarg.h>
#include <system.h>
#include <string.h>
```

Include dependency graph for stdio.c:



### Functions

- int **printf** (char \*form,...)  
*takes in a format string and prints it out to the DEFAULT\_DEVICE*
- int **puts** (char \*buff)  
*prints out a string to DEFAULT\_DEVICE*

#### 4.16.1 Detailed Description

Holds all implementation of standard I/O functions.

#### 4.16.2 Function Documentation

##### 4.16.2.1 int printf ( char \* form, ... )

takes in a format string and prints it out to the DEFAULT\_DEVICE

c - character d/i - decimal integer x - hexadecimal integer s - string %% - percent sign Numbers can be included before the format specifier to declare alignment. i.e. %-10c = "c " A pad with 0s can also be added using a '0' directly after the percent i.e. %03c = "00c"

## Parameters

<i>form</i>	character pointer to the format
<i>valist</i>	variadic arguments to match the format (see brief)

## Returns

0 for failure 1 for success

## 4.16.2.2 int puts ( char \* buff )

prints out a string to DEFAULT\_DEVICE

## Parameters

<i>buff</i>	string to print out
-------------	---------------------

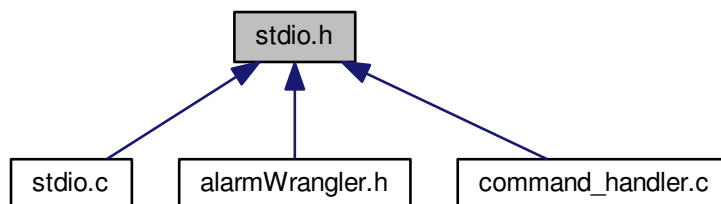
## Returns

1

## 4.17 stdio.h File Reference

Holds all prototypes of standard I/O functions.

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



## Functions

- int [printf](#) (char \*form,...)  
*takes in a format string and prints it out to the DEFAULT\_DEVICE*
- int [puts](#) (char \*buffer)  
*prints out a string to DEFAULT\_DEVICE*

### 4.17.1 Detailed Description

Holds all prototypes of standard I/O functions.

### 4.17.2 Function Documentation

#### 4.17.2.1 `int printf ( char * form, ... )`

takes in a format string and prints it out to the DEFAULT\_DEVICE

c - character d/i - decimal integer x - hexadecimal integer s - string %% - percent sign Numbers can be included before the format specifier to declare alignment. i.e. %-10c = "c " A pad with 0s can also be added using a '0' directly after the percent i.e. %03c = "00c"

##### Parameters

<i>form</i>	character pointer to the format
<i>valist</i>	variadic arguments to match the format (see brief)

##### Returns

0 for failure 1 for success

#### 4.17.2.2 `int puts ( char * buff )`

prints out a string to DEFAULT\_DEVICE

##### Parameters

<i>buff</i>	string to print out
-------------	---------------------

##### Returns

1

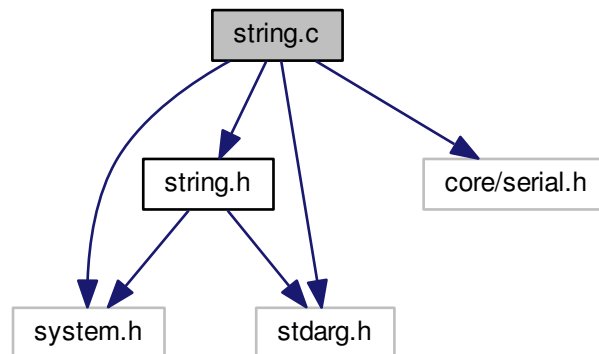
## 4.18 string.c File Reference

Holds all utility functions used to modify strings.

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



Include dependency graph for string.c:



## Macros

- `#define F_MINUS (1 << 0)`
- `#define F_PLUS (1 << 1)`
- `#define F_PERCENT (1 << 2)`
- `#define F_ZERO (1 << 3)`

## Typedefs

- `typedef unsigned char BYTE`

## Functions

- `int strlen (const char *s)`
- `char * strcpy (char *s1, const char *s2)`
- `int atoi (const char *s)`
- `int strcmp (const char *s1, const char *s2)`
- `char * strcat (char *s1, const char *s2)`
- `int isspace (const char *c)`
- `void * memset (void *s, int c, size_t n)`
- `char * strtok (char *s1, const char *s2)`
- `int isdigit (char c)`  
*Checks if char c is a digit.*
- `char * reverse (char *str, int end)`  
*reverse a string from 0 to j*
- `char * itoa (int num, char *str, int base)`  
*Converts signed integer to string.*
- `char * utoa (u32int num, char *str, int base)`  
*Converts unsigned integer to string.*
- `char * sprintf_pad_helper (char *buffer, char pad, int fNum, int n, BYTE doAction)`

- adds spaces where needed for the sprintf function*
- int `sprintf_internal` (char \*buffer, char \*format, va\_list valist)  
*Main implementation of the sprintf function.*
- int `sprintf` (char \*buffer, char \*format,...)  
*Visible representation of the sprintf function.*
- char `tolower` (char c)  
*Returns the lowercase representation of a character.*
- char `toupper` (char c)  
*Returns the uppercase representation of a character.*
- char \* `trim` (char \*str)  
*Returns a string with the beginning and ending whitespaces removed.*

### 4.18.1 Detailed Description

Holds all utility functions used to modify strings.

### 4.18.2 Function Documentation

#### 4.18.2.1 int isdigit ( char c )

Checks if char c is a digit.

##### Parameters

<i>c</i>	character to check
----------	--------------------

##### Returns

is digit: 1; is not digit: 0;

#### 4.18.2.2 char\* itoa ( int num, char \* str, int base )

Converts signed integer to string.

##### Parameters

<i>num</i>	number to convert
<i>str</i>	string to store result in
<i>base</i>	base to convert to

##### Returns

pointer to str

#### 4.18.2.3 char\* reverse ( char \* *str*, int *end* )

reverse a string from 0 to j

##### Parameters

<i>str</i>	string to reverse
<i>j</i>	index to reverse str to

##### Returns

pointer to str

#### 4.18.2.4 int sprintf ( char \* *buffer*, char \* *format*, ... )

Visible representation of the sprintf function.

c - character d/i - decimal integer x - hexadecimal integer s - string %% - percent sign Numbers can be included before the format specifier to declare alignment. i.e. %-10c = "c " A pad with 0s can also be added using a '0' directly after the percent i.e. %03c = "00c"

##### Parameters

<i>buffer</i>	character pointer to store spaces to
<i>format</i>	format string with format specifiers
<i>valist</i>	variadic list with parameters matching the format

##### Returns

pointer to buffer

#### 4.18.2.5 int sprintf\_internal ( char \* *buffer*, char \* *format*, va\_list *valist* )

Main implementation of the sprintf function.

c - character d/i - decimal integer x - hexadecimal integer s - string %% - percent sign Numbers can be included before the format specifier to declare alignment. i.e. %-10c = "c " A pad with 0s can also be added using a '0' directly after the percent i.e. %03c = "00c"

##### Parameters

<i>buffer</i>	character pointer to store spaces to
<i>format</i>	format string with format specifiers
<i>valist</i>	variadic list with parameters matching the format

**Returns**

pointer to buffer

**4.18.2.6 char\* sprintf\_pad\_helper ( char \* *buffer*, char *pad*, int *fNum*, int *n*, BYTE *doAction* )**

adds spaces where needed for the sprintf function

**Parameters**

<i>buffer</i>	character pointer to store spaces to
<i>pad</i>	what character to pad with
<i>fNum</i>	format number from sprintf
<i>n</i>	length of string that has been/will be added
<i>doAction</i>	boolean on whether or not to add the spaces

**Returns**

pointer to buffer

**4.18.2.7 char tolower ( char *c* )**

Returns the lowercase representation of a character.

**Parameters**

<i>c</i>	character to return the lowercase representation of
----------	---

**Returns**

lowercase representation of *c* in ASCII

**4.18.2.8 char toupper ( char *c* )**

Returns the uppercase representation of a character.

**Parameters**

<i>c</i>	character to return the uppercase representation of
----------	---

**Returns**

uppercase representation of *c* in ASCII

#### 4.18.2.9 char\* trim ( char \* *str* )

Returns a string with the beginning and ending whitespaces removed.

##### Parameters

<i>str</i>	the string to have white spaces removed from
------------	--

##### Returns

a sting with the beginning and ending whitespaces removed

#### 4.18.2.10 char\* utoa ( u32int *num*, char \* *str*, int *base* )

Converts unsigned integer to string.

##### Parameters

<i>num</i>	number to convert
<i>str</i>	string to store result in
<i>base</i>	base to convert to

##### Returns

pointer to *str*

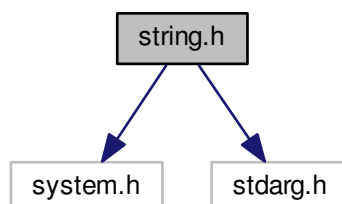
## 4.19 string.h File Reference

Holds all utility prototypes used to modify strings.

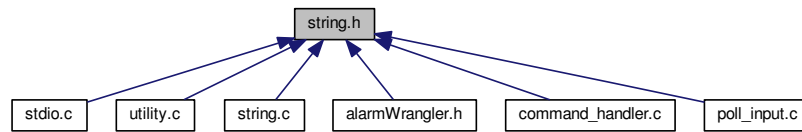
```
#include <system.h>
```

```
#include <stdarg.h>
```

Include dependency graph for string.h:



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



## Functions

- int **isspace** (const char \*c)
- void \* **memset** (void \*s, int c, size\_t n)
- char \* **strcpy** (char \*s1, const char \*s2)
- char \* **strcat** (char \*s1, const char \*s2)
- int **strlen** (const char \*s)
- int **strcmp** (const char \*s1, const char \*s2)
- char \* **strtok** (char \*s1, const char \*s2)
- int **isdigit** (char c)  
*Checks if char c is a digit.*
- char \* **reverse** (char \*str, int j)  
*reverse a string from 0 to j*
- int **atoi** (const char \*s)
- char \* **itoa** (int num, char \*str, int base)  
*Converts signed integer to string.*
- char \* **utoa** (u32int num, char \*str, int base)  
*Converts unsigned integer to string.*
- int **sprintf** (char \*buffer, char \*format,...)  
*Visible representation of the sprintf function.*
- int **sprintf\_internal** (char \*buffer, char \*format, va\_list valist)  
*Main implementation of the sprintf function.*
- char **tolower** (char c)  
*Returns the lowercase representation of a character.*
- char **toupper** (char c)  
*Returns the uppercase representation of a character.*
- char \* **trim** (char \*str)  
*Returns a string with the beginning and ending whitespaces removed.*

### 4.19.1 Detailed Description

Holds all utility prototypes used to modify strings.

### 4.19.2 Function Documentation

#### 4.19.2.1 int isdigit ( char c )

Checks if char c is a digit.

## Parameters

<i>c</i>	character to check
----------	--------------------

## Returns

is digit: 1; is not digit: 0;

4.19.2.2 char\* itoa ( int *num*, char \* *str*, int *base* )

Converts signed integer to string.

## Parameters

<i>num</i>	number to convert
<i>str</i>	string to store result in
<i>base</i>	base to convert to

## Returns

pointer to *str*

4.19.2.3 char\* reverse ( char \* *str*, int *end* )

reverse a string from 0 to *j*

## Parameters

<i>str</i>	string to reverse
<i>j</i>	index to reverse <i>str</i> to

## Returns

pointer to *str*

4.19.2.4 int sprintf ( char \* *buffer*, char \* *format*, ... )

Visible representation of the sprintf function.

*c* - character *d/i* - decimal integer *x* - hexadecimal integer *s* - string *%%* - percent sign Numbers can be included before the format specifier to declare alignment. i.e. %-10c = "c " A pad with 0s can also be added using a '0' directly after the percent i.e. %03c = "00c"

## Parameters

<i>buffer</i>	character pointer to store spaces to
<i>format</i>	format string with format specifiers
Variables	Generated by Doxygen variable list with parameters matching the format

**Returns**

pointer to buffer

**4.19.2.5 int sprintf\_internal ( char \* *buffer*, char \* *format*, va\_list *valist* )**

Main implementation of the sprintf function.

c - character d/i - decimal integer x - hexadecimal integer s - string %% - percent sign Numbers can be included before the format specifier to declare alignment. i.e. %-10c = "c " A pad with 0s can also be added using a '0' directly after the percent i.e. %03c = "00c"

**Parameters**

<i>buffer</i>	character pointer to store spaces to
<i>format</i>	format string with format specifiers
<i>valist</i>	variadic list with parameters matching the format

**Returns**

pointer to buffer

**4.19.2.6 char tolower ( char *c* )**

Returns the lowercase representation of a character.

**Parameters**

<i>c</i>	character to return the lowercase representation of
----------	---

**Returns**

lowercase representation of c in ASCII

**4.19.2.7 char toupper ( char *c* )**

Returns the uppercase representation of a character.

**Parameters**

<i>c</i>	character to return the uppercase representation of
----------	---

**Returns**

uppercase representation of c in ASCII



#### 4.19.2.8 char\* trim ( char \* *str* )

Returns a string with the beginning and ending whitespaces removed.

##### Parameters

<i>str</i>	the string to have white spaces removed from
------------	--

##### Returns

a sting with the beginning and ending whitespaces removed

#### 4.19.2.9 char\* utoa ( u32int *num*, char \* *str*, int *base* )

Converts unsigned integer to string.

##### Parameters

<i>num</i>	number to convert
<i>str</i>	string to store result in
<i>base</i>	base to convert to

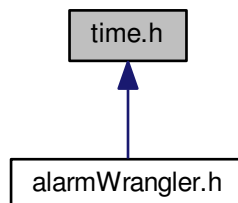
##### Returns

pointer to *str*

## 4.20 time.h File Reference

The header file for the date and time functions.

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



## Data Structures

- struct [time](#)  
*A struct to all the time and date elements.*
- struct [fakelong](#)  
*Fake 64 bit integer.*

## Macros

- #define **SECOND\_REG** 0x00
- #define **MINUTE\_REG** 0x02
- #define **HOURLY\_REG** 0x04
- #define **DAY\_OF\_MONTH\_REG** 0x07
- #define **MONTH\_REG** 0x08
- #define **CENTURY\_REG** 0x32
- #define **YEAR\_REG** 0x09
- #define **INDEX\_REG** 0x70
- #define **DATA\_REG** 0x71
- #define **MIN\_YEAR** 1750
- #define **MAX\_YEAR** 2500

## Typedefs

- typedef struct [time](#) **time\_h**

## Enumerations

- enum **MONTH** {  
    **JANUARY** = 1, **FEBRUARY**, **MARCH**, **APRIL**,  
    **MAY**, **JUNE**, **JULY**, **AGUST**,  
    **SEPTEMBER**, **OCTOBER**, **NOVEMBER**, **DECEMBER** }

## Functions

- void [format\\_time](#) (char \*dest, [time\\_h](#) \*t)  
*Generates a string with a standard format of time.*
- [time\\_h](#) [get\\_current\\_time](#) ()  
*Retrieves the current time in the Real Time Clock(RTC).*
- int [set\\_current\\_time](#) ([time\\_h](#) time)  
*Sets the current time in the RTC.*
- int [bcd\\_to\\_decimal](#) (int bcd)  
*Converts BCD values into decimal.*
- struct [fakelong](#) [rdtsc](#) (void)  
*return clock cycles since reset in a fake long long*
- [time\\_h](#) \* [parseTandD](#) ([time\\_h](#) \*dest, char \*input)
- int **validTime** (char \*hours, char \*minutes, char \*seconds)
- int **validDate** (char \*year, char \*month, char \*day)
- int **compareTime** ([time\\_h](#) timeOne, [time\\_h](#) timeTwo)

### 4.20.1 Detailed Description

The header file for the date and time functions.

### 4.20.2 Function Documentation

#### 4.20.2.1 `int bcd_to_decimal ( int bcd )`

Converts BCD values into decimal.

This function converts BCD values, to be a more code friendly decimal value.

##### Parameters

<i>bcd</i>	Value that is in BCD that needs to be a normal decimal value.
------------	---

##### Returns

The value of the BCD as an integer.

#### 4.20.2.2 `void format_time ( char * dest, time_h * t )`

Generates a string with a standard format of time.

Generates a string that contains all the data contained in a time\_h. This form shows all data from largest timescale to smallest timescale.

##### Parameters

<i>dest</i>	Pointer to a string that is large enough to contain the output string
<i>time</i>	Pointer to the time to write into the destination string.

##### Returns

Return is through the 'dest' pointer.

##### Note

This is merely a convenience, as it is only an sprintf call.

#### 4.20.2.3 `time_h get_current_time ( )`

Retrieves the current time in the Real Time Clock(RTC).

Aquires data from the RTC, packaging the data into a time\_h struct for ease of use.

##### Returns

Returns the current time represented as 6 values in a time\_h struct.

#### 4.20.2.4 int set\_current\_time ( time\_h time )

Sets the current time in the RTC.

Uses a time\_h struct to set the data members of the RTC. This function also does error checking on valid times, including leap-years, valid days of months, etc., to ensure the given time is valid.

##### Parameters

<i>time</i>	A time_h struct containing the new time, as defined by the user.
-------------	--

##### Returns

If the operation was successful in boolean format (1 = true, 0 = false).

##### Note

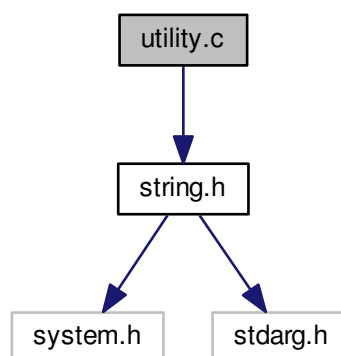
This function also ensures that the date will be set in the correct order within the RTC.  
Setting a value in the input struct to a '-1' will skip the value in setting the time. Essentially, keeping the value as it was before. This is demonstrated in the commands.c file.

## 4.21 utility.c File Reference

Holds utility function implementations for this project.

```
#include <string.h>
```

Include dependency graph for utility.c:



## Functions

- int [isnullorspace](#) (char test)

*Determines if a passed character is a null or space.*

### 4.21.1 Detailed Description

Holds utility function implementations for this project.

### 4.21.2 Function Documentation

#### 4.21.2.1 `int isnullorspace ( char test )`

Determines if a passed character is a null or space.

##### Parameters

<code>test</code>	character to test
-------------------	-------------------

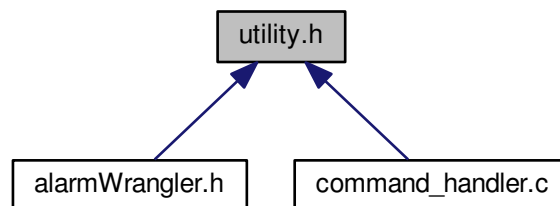
##### Returns

1 if space or null, 0 otherwise

## 4.22 utility.h File Reference

Holds utility function prototypes for this project.

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



### Functions

- `int isnullorspace (char test)`  
*Determines if a passed character is a null or space.*

### 4.22.1 Detailed Description

Holds utility function prototypes for this project.

## 4.22.2 Function Documentation

### 4.22.2.1 `int isnullorspace ( char test )`

Determines if a passed character is a null or space.

#### Parameters

<i>test</i>	character to test
-------------	-------------------

#### Returns

1 if space or null, 0 otherwise





# Index

- A\_FLAG
  - commandUtils.h, [26](#)
- ALARM, [5](#)
- ALIAS, [6](#)
- alarmProcess
  - alarmWrangler.h, [16](#)
- alarmWrangler.h, [15](#)
  - alarmProcess, [16](#)
  - check, [16](#)
  - insertAlarm, [16](#)
  - listAlarms, [16](#)
  - removeAlarm, [17](#)
- allocate\_pcb
  - pcb\_utils.h, [41](#)
- alphanum
  - commandUtils.h, [26](#)
- B\_FLAG
  - commandUtils.h, [27](#)
- bcd\_to\_decimal
  - time.h, [65](#)
- C\_FLAG
  - commandUtils.h, [27](#)
- CMDSIZE
  - command\_handler.c, [18](#)
  - commandUtils.h, [27](#)
- COMMAND, [7](#)
- check
  - alarmWrangler.h, [16](#)
- cmcb, [6](#)
- cmd\_alarm
  - commands.h, [20](#)
- cmd\_alias
  - commands.h, [20](#)
- cmd\_blockPCB
  - commands.h, [20](#)
- cmd\_clear
  - commands.h, [20](#)
- cmd\_create\_pcb
  - commands.h, [21](#)
- cmd\_date
  - commands.h, [21](#)
- cmd\_delete\_pcb
  - commands.h, [21](#)
- cmd\_help
  - commands.h, [21](#)
- cmd\_history
  - commands.h, [22](#)
- cmd\_infinity
  - commands.h, [22](#)
- cmd\_loadr3
  - commands.h, [22](#)
- cmd\_resume
  - commands.h, [22](#)
- cmd\_set\_priority\_pcb
  - commands.h, [22](#)
- cmd\_shutdown
  - commands.h, [23](#)
- cmd\_suspend
  - commands.h, [23](#)
- cmd\_time
  - commands.h, [23](#)
- cmd\_unblock\_pcb
  - commands.h, [23](#)
- cmd\_version
  - commands.h, [24](#)
- cmd\_yield
  - commands.h, [24](#)
- command\_handler.c, [17](#)
  - CMDSIZE, [18](#)
- command\_handler.h, [18](#)
- commandUtils.h, [24](#)
  - A\_FLAG, [26](#)
  - alphanum, [26](#)
  - B\_FLAG, [27](#)
  - C\_FLAG, [27](#)
  - CMDSIZE, [27](#)
  - D\_FLAG, [27](#)
  - E\_FLAG, [27](#)
  - F\_FLAG, [27](#)
  - G\_FLAG, [27](#)
  - get\_command\_array, [29](#)
  - get\_pvalue, [29](#)
  - H\_FLAG, [27](#)
  - I\_FLAG, [27](#)
  - J\_FLAG, [27](#)
  - K\_FLAG, [28](#)
  - L\_FLAG, [28](#)
  - M\_FLAG, [28](#)
  - N\_FLAG, [28](#)
  - NO\_FLAG, [28](#)
  - O\_FLAG, [28](#)
  - P\_FLAG, [28](#)
  - Q\_FLAG, [28](#)
  - R\_FLAG, [28](#)
  - S\_FLAG, [28](#)
  - search\_commands, [30](#)
  - set\_flags, [30](#)

- set\_flags\_search\_alias, 30
- T\_FLAG, 29
- U\_FLAG, 29
- V\_FLAG, 29
- W\_FLAG, 29
- X\_FLAG, 29
- Y\_FLAG, 29
- Z\_FLAG, 29
- commands.h, 18
  - cmd\_alarm, 20
  - cmd\_alias, 20
  - cmd\_blockPCB, 20
  - cmd\_clear, 20
  - cmd\_create\_pcb, 21
  - cmd\_date, 21
  - cmd\_delete\_pcb, 21
  - cmd\_help, 21
  - cmd\_history, 22
  - cmd\_infinity, 22
  - cmd\_loadr3, 22
  - cmd\_resume, 22
  - cmd\_set\_priority\_pcb, 22
  - cmd\_shutdown, 23
  - cmd\_suspend, 23
  - cmd\_time, 23
  - cmd\_unblock\_pcb, 23
  - cmd\_version, 24
  - cmd\_yield, 24
- construct\_queue
  - pcb\_queue.h, 39
- control\_sequence, 8
- control\_sequences
  - poll\_input.c, 48
- ControlSequence
  - poll\_input.h, 50
- D\_FLAG
  - commandUtils.h, 27
- dequeue
  - pcb\_queue.h, 39
- destruct\_queue
  - pcb\_queue.h, 39
- E\_FLAG
  - commandUtils.h, 27
- enqueue
  - pcb\_queue.h, 39
- F\_FLAG
  - commandUtils.h, 27
- fakelong, 8
- find\_pcb
  - pcb\_utils.h, 41
- format\_time
  - time.h, 65
- free\_pcb
  - pcb\_utils.h, 42
- G\_FLAG
  - commandUtils.h, 27
- get\_blocked\_queue
  - pcb\_utils.h, 42
- get\_command\_array
  - commandUtils.h, 29
- get\_command\_history
  - poll\_input.c, 46
  - poll\_input.h, 50
- get\_current\_time
  - time.h, 65
- get\_history\_length
  - poll\_input.c, 46
  - poll\_input.h, 50
- get\_key
  - poll\_input.c, 46
- get\_process\_class\_string
  - pcb\_utils.h, 42
- get\_process\_state\_string
  - pcb\_utils.h, 42
- get\_pvalue
  - commandUtils.h, 29
- get\_ready\_queue
  - pcb\_utils.h, 43
- get\_remaining\_free
  - memory\_wrangler.h, 34
- get\_suspended\_blocked\_queue
  - pcb\_utils.h, 43
- get\_suspended\_ready\_queue
  - pcb\_utils.h, 43
- H\_FLAG
  - commandUtils.h, 27
- HEAP\_SIZE
  - memEnv.h, 32
- HELP\_PAGES, 9
- I\_FLAG
  - commandUtils.h, 27
- init\_queue
  - pcb\_utils.h, 43
- input\_available
  - poll\_input.c, 46
- insert\_pcb
  - pcb\_utils.h, 43
- insertAlarm
  - alarmWrangler.h, 16
- internal\_free
  - memEnv.h, 32
  - memory\_wrangler.h, 34
- internal\_malloc
  - memEnv.h, 32
  - memory\_wrangler.h, 34
- isdigit
  - string.c, 56
  - string.h, 60
- isnullorspace
  - utility.c, 67
  - utility.h, 68
- itoa

- string.c, 56
- string.h, 61
- J\_FLAG
  - commandUtils.h, 27
- K\_FLAG
  - commandUtils.h, 28
- kill\_it\_\_kill\_it\_all
  - pcb\_utils.h, 43
- L\_FLAG
  - commandUtils.h, 28
- listAlarms
  - alarmWrangler.h, 16
- lmcb, 9
- M\_FLAG
  - commandUtils.h, 28
- mem\_init
  - memEnv.h, 32
  - memory\_wrangler.h, 35
- memEnv.h, 31
  - HEAP\_SIZE, 32
  - internal\_free, 32
  - internal\_malloc, 32
  - mem\_init, 32
  - show\_mem\_state, 32
- memcpy
  - poll\_input.c, 46
- memory\_wrangler.h, 33
  - get\_remaining\_free, 34
  - internal\_free, 34
  - internal\_malloc, 34
  - mem\_init, 35
  - print\_both, 35
  - print\_cmcb, 35
  - print\_lmcb, 35
  - show\_alloc\_mem\_state, 35
  - show\_free\_mem\_state, 35
  - show\_mem\_state, 35
- move\_cursor
  - poll\_input.c, 47
- N\_FLAG
  - commandUtils.h, 28
- NO\_FLAG
  - commandUtils.h, 28
- node, 10
- node\_t
  - pcb\_constants.h, 37
- O\_FLAG
  - commandUtils.h, 28
- P\_FLAG
  - commandUtils.h, 28
- pcb, 11
- pcb\_constants.h, 36
  - node\_t, 37
- pcb\_queue.h, 38
  - construct\_queue, 39
  - dequeue, 39
  - destruct\_queue, 39
  - enqueue, 39
  - priority\_enqueue, 40
- pcb\_utils.h, 40
  - allocate\_pcb, 41
  - find\_pcb, 41
  - free\_pcb, 42
  - get\_blocked\_queue, 42
  - get\_process\_class\_string, 42
  - get\_process\_state\_string, 42
  - get\_ready\_queue, 43
  - get\_suspended\_blocked\_queue, 43
  - get\_suspended\_ready\_queue, 43
  - init\_queue, 43
  - insert\_pcb, 43
  - kill\_it\_\_kill\_it\_all, 43
  - print\_pcb\_info, 44
  - remove\_pcb, 44
  - setup\_pcb, 44
- pcb\_wrangler.h, 44
- poll\_input
  - poll\_input.c, 47
  - poll\_input.h, 50
- poll\_input.c, 45
  - control\_sequences, 48
  - get\_command\_history, 46
  - get\_history\_length, 46
  - get\_key, 46
  - input\_available, 46
  - memcpy, 46
  - move\_cursor, 47
  - poll\_input, 47
  - print\_after\_cursor, 47
  - TOLERANCE, 48
  - wait\_for\_input, 47
- poll\_input.h, 49
  - ControlSequence, 50
  - get\_command\_history, 50
  - get\_history\_length, 50
  - poll\_input, 50
- print\_after\_cursor
  - poll\_input.c, 47
- print\_both
  - memory\_wrangler.h, 35
- print\_cmcb
  - memory\_wrangler.h, 35
- print\_lmcb
  - memory\_wrangler.h, 35
- print\_pcb\_info
  - pcb\_utils.h, 44
- printf
  - stdio.c, 52
  - stdio.h, 54
- priority\_enqueue
  - pcb\_queue.h, 40

- procsr3.h, 51
- puts
  - stdio.c, 53
  - stdio.h, 54
- Q\_FLAG
  - commandUtils.h, 28
- queue, 12
- R\_FLAG
  - commandUtils.h, 28
- remove\_pcb
  - pcb\_utils.h, 44
- removeAlarm
  - alarmWrangler.h, 17
- reverse
  - string.c, 56
  - string.h, 61
- S\_FLAG
  - commandUtils.h, 28
- search\_commands
  - commandUtils.h, 30
- set\_current\_time
  - time.h, 65
- set\_flags
  - commandUtils.h, 30
- set\_flags\_search\_alias
  - commandUtils.h, 30
- setup\_pcb
  - pcb\_utils.h, 44
- show\_alloc\_mem\_state
  - memory\_wrangler.h, 35
- show\_free\_mem\_state
  - memory\_wrangler.h, 35
- show\_mem\_state
  - memEnv.h, 32
  - memory\_wrangler.h, 35
- splash.h, 51
- sprintf
  - string.c, 57
  - string.h, 61
- sprintf\_internal
  - string.c, 57
  - string.h, 62
- sprintf\_pad\_helper
  - string.c, 58
- stdio.c, 52
  - printf, 52
  - puts, 53
- stdio.h, 53
  - printf, 54
  - puts, 54
- string.c, 54
  - isdigit, 56
  - itoa, 56
  - reverse, 56
  - sprintf, 57
  - sprintf\_internal, 57
  - sprintf\_pad\_helper, 58
  - tolower, 58
  - toupper, 58
  - trim, 58
  - utoa, 59
- string.h, 59
  - isdigit, 60
  - itoa, 61
  - reverse, 61
  - sprintf, 61
  - sprintf\_internal, 62
  - tolower, 62
  - toupper, 62
  - trim, 62
  - utoa, 63
- T\_FLAG
  - commandUtils.h, 29
- TOLERANCE
  - poll\_input.c, 48
- time, 12
- time.h, 63
  - bcd\_to\_decimal, 65
  - format\_time, 65
  - get\_current\_time, 65
  - set\_current\_time, 65
- tolower
  - string.c, 58
  - string.h, 62
- toupper
  - string.c, 58
  - string.h, 62
- trim
  - string.c, 58
  - string.h, 62
- U\_FLAG
  - commandUtils.h, 29
- utility.c, 66
  - isnullorspace, 67
- utility.h, 67
  - isnullorspace, 68
- utoa
  - string.c, 59
  - string.h, 63
- V\_FLAG
  - commandUtils.h, 29
- W\_FLAG
  - commandUtils.h, 29
- wait\_for\_input
  - poll\_input.c, 47
- X\_FLAG
  - commandUtils.h, 29
- Y\_FLAG
  - commandUtils.h, 29

Z\_FLAG

commandUtils.h, [29](#)