

PotatOS

Generated by Doxygen 1.8.11

Contents

| | | |
|----------|---|----------|
| 1 | Data Structure Index | 1 |
| 1.1 | Data Structures | 1 |
| 2 | File Index | 3 |
| 2.1 | File List | 3 |
| 3 | Data Structure Documentation | 5 |
| 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 | COMMAND Struct Reference | 6 |
| 3.3.1 | Detailed Description | 6 |
| 3.4 | control_sequence Struct Reference | 7 |
| 3.4.1 | Detailed Description | 7 |
| 3.5 | fakelong Struct Reference | 7 |
| 3.5.1 | Detailed Description | 8 |
| 3.6 | HELP_PAGES Struct Reference | 8 |
| 3.6.1 | Detailed Description | 8 |
| 3.7 | node Struct Reference | 8 |
| 3.7.1 | Detailed Description | 9 |
| 3.8 | pcb Struct Reference | 9 |
| 3.8.1 | Detailed Description | 10 |
| 3.9 | queue Struct Reference | 10 |
| 3.9.1 | Detailed Description | 10 |
| 3.10 | time Struct Reference | 11 |
| 3.10.1 | Detailed Description | 11 |

| | |
|--|-----------|
| 4 File Documentation | 13 |
| 4.1 alarmWrangler.h File Reference | 13 |
| 4.1.1 Detailed Description | 14 |
| 4.1.2 Function Documentation | 14 |
| 4.1.2.1 alarmProcess() | 14 |
| 4.1.2.2 check() | 14 |
| 4.1.2.3 insertAlarm(char *message, char *dateIn) | 14 |
| 4.1.2.4 listAlarms() | 15 |
| 4.1.2.5 removeAlarm(const char *message) | 15 |
| 4.2 command_handler.h File Reference | 15 |
| 4.2.1 Detailed Description | 15 |
| 4.3 commands.h File Reference | 15 |
| 4.3.1 Detailed Description | 17 |
| 4.3.2 Function Documentation | 17 |
| 4.3.2.1 cmd_alarm(char *params) | 17 |
| 4.3.2.2 cmd_alias(char *params) | 17 |
| 4.3.2.3 cmd_blockPCB(char *params) | 17 |
| 4.3.2.4 cmd_clear(char *params) | 17 |
| 4.3.2.5 cmd_create_pcb(char *params) | 17 |
| 4.3.2.6 cmd_date(char *params) | 18 |
| 4.3.2.7 cmd_delete_pcb(char *params) | 18 |
| 4.3.2.8 cmd_help(char *params) | 18 |
| 4.3.2.9 cmd_infinity(char *params) | 19 |
| 4.3.2.10 cmd_loadr3(char *params) | 19 |
| 4.3.2.11 cmd_resume(char *params) | 19 |
| 4.3.2.12 cmd_set_priority_pcb(char *params) | 19 |
| 4.3.2.13 cmd_shutdown(char *params) | 19 |
| 4.3.2.14 cmd_suspend(char *params) | 20 |
| 4.3.2.15 cmd_time(char *params) | 20 |
| 4.3.2.16 cmd_unblock_pcb(char *params) | 20 |

| | | |
|----------|--|----|
| 4.3.2.17 | cmd_version(char *params) | 20 |
| 4.3.2.18 | cmd_yield(char *params) | 21 |
| 4.4 | commandUtils.h File Reference | 21 |
| 4.4.1 | Detailed Description | 23 |
| 4.4.2 | Macro Definition Documentation | 23 |
| 4.4.2.1 | A_FLAG | 23 |
| 4.4.2.2 | alphanum | 23 |
| 4.4.2.3 | B_FLAG | 23 |
| 4.4.2.4 | C_FLAG | 23 |
| 4.4.2.5 | CMD_SIZE | 23 |
| 4.4.2.6 | D_FLAG | 24 |
| 4.4.2.7 | E_FLAG | 24 |
| 4.4.2.8 | F_FLAG | 24 |
| 4.4.2.9 | G_FLAG | 24 |
| 4.4.2.10 | H_FLAG | 24 |
| 4.4.2.11 | I_FLAG | 24 |
| 4.4.2.12 | J_FLAG | 24 |
| 4.4.2.13 | K_FLAG | 24 |
| 4.4.2.14 | L_FLAG | 24 |
| 4.4.2.15 | M_FLAG | 24 |
| 4.4.2.16 | N_FLAG | 25 |
| 4.4.2.17 | NO_FLAG | 25 |
| 4.4.2.18 | O_FLAG | 25 |
| 4.4.2.19 | P_FLAG | 25 |
| 4.4.2.20 | Q_FLAG | 25 |
| 4.4.2.21 | R_FLAG | 25 |
| 4.4.2.22 | S_FLAG | 25 |
| 4.4.2.23 | T_FLAG | 25 |
| 4.4.2.24 | U_FLAG | 25 |
| 4.4.2.25 | V_FLAG | 25 |

| | | |
|----------|---|----|
| 4.4.2.26 | W_FLAG | 26 |
| 4.4.2.27 | X_FLAG | 26 |
| 4.4.2.28 | Y_FLAG | 26 |
| 4.4.2.29 | Z_FLAG | 26 |
| 4.4.3 | Function Documentation | 26 |
| 4.4.3.1 | get_pvalue(char c) | 26 |
| 4.4.3.2 | search_commands(char *) | 26 |
| 4.4.3.3 | set_flags(char *paramstr, int *flag, int num_aliases,...) | 27 |
| 4.4.3.4 | set_flags_search_alias(char *alias, int num_aliases, ALIAS aliases[]) | 27 |
| 4.5 | pcb_constants.h File Reference | 27 |
| 4.5.1 | Detailed Description | 29 |
| 4.5.2 | Typedef Documentation | 29 |
| 4.5.2.1 | node_t | 29 |
| 4.6 | pcb_queue.h File Reference | 29 |
| 4.6.1 | Detailed Description | 30 |
| 4.6.2 | Function Documentation | 30 |
| 4.6.2.1 | construct_queue() | 30 |
| 4.6.2.2 | dequeue(queue_t *queue) | 30 |
| 4.6.2.3 | destruct_queue(queue_t *queue) | 31 |
| 4.6.2.4 | enqueue(queue_t *que, pcb_t *data) | 31 |
| 4.6.2.5 | priority_enqueue(queue_t *cue, pcb_t *data) | 31 |
| 4.7 | pcb_utils.h File Reference | 32 |
| 4.7.1 | Detailed Description | 33 |
| 4.7.2 | Function Documentation | 33 |
| 4.7.2.1 | allocate_pcb() | 33 |
| 4.7.2.2 | find_pcb(char *pname) | 33 |
| 4.7.2.3 | free_pcb(pcb_t *) | 33 |
| 4.7.2.4 | get_blocked_queue() | 34 |
| 4.7.2.5 | get_process_class_string(PROCESS_CLASS process_class) | 34 |
| 4.7.2.6 | get_process_state_string(PROCESS_STATE process_state) | 34 |

| | | |
|----------|--|----|
| 4.7.2.7 | get_ready_queue() | 34 |
| 4.7.2.8 | get_suspended_blocked_queue() | 34 |
| 4.7.2.9 | get_suspended_ready_queue() | 35 |
| 4.7.2.10 | kill_it___kill_it_all() | 35 |
| 4.7.2.11 | print_pcb_info(const pcb_t *pcb) | 35 |
| 4.7.2.12 | remove_pcb(char *pname) | 35 |
| 4.7.2.13 | setup_pcb(char *, PROCESS_CLASS, int priority) | 35 |
| 4.8 | pcb_wrangler.h File Reference | 35 |
| 4.8.1 | Detailed Description | 36 |
| 4.9 | poll_input.c File Reference | 36 |
| 4.9.1 | Detailed Description | 37 |
| 4.9.2 | Function Documentation | 37 |
| 4.9.2.1 | get_key() | 37 |
| 4.9.2.2 | input_available() | 37 |
| 4.9.2.3 | memcpy(char *destination, const char *source, int n) | 37 |
| 4.9.2.4 | move_cursor(int n) | 38 |
| 4.9.2.5 | poll_input(char *buffer, int *length) | 38 |
| 4.9.2.6 | print_after_cursor(const char *str) | 38 |
| 4.9.2.7 | wait_for_input(int timeout) | 38 |
| 4.9.3 | Variable Documentation | 39 |
| 4.9.3.1 | control_sequences | 39 |
| 4.9.3.2 | TOLERANCE | 39 |
| 4.10 | poll_input.h File Reference | 40 |
| 4.10.1 | Detailed Description | 40 |
| 4.10.2 | Typedef Documentation | 41 |
| 4.10.2.1 | ControlSequence | 41 |
| 4.10.3 | Function Documentation | 41 |
| 4.10.3.1 | poll_input(char *buffer, int *length) | 41 |
| 4.11 | splash.h File Reference | 41 |
| 4.11.1 | Detailed Description | 41 |

| | | |
|-----------|--|----|
| 4.12 | stdio.c File Reference | 42 |
| 4.12.1 | Detailed Description | 42 |
| 4.12.2 | Function Documentation | 42 |
| 4.12.2.1 | printf(char *form,...) | 42 |
| 4.12.2.2 | puts(char *buff) | 43 |
| 4.13 | stdio.h File Reference | 43 |
| 4.13.1 | Detailed Description | 44 |
| 4.13.2 | Function Documentation | 44 |
| 4.13.2.1 | printf(char *form,...) | 44 |
| 4.13.2.2 | puts(char *buffer) | 44 |
| 4.14 | string.c File Reference | 44 |
| 4.14.1 | Detailed Description | 46 |
| 4.14.2 | Function Documentation | 46 |
| 4.14.2.1 | isdigit(char c) | 46 |
| 4.14.2.2 | itoa(int num, char *str, int base) | 46 |
| 4.14.2.3 | reverse(char *str, int end) | 47 |
| 4.14.2.4 | sprintf(char *buffer, char *format,...) | 47 |
| 4.14.2.5 | sprintf_internal(char *buffer, char *format, va_list valist) | 47 |
| 4.14.2.6 | sprintf_pad_helper(char *buffer, char pad, int fNum, int n, BYTE doAction) | 48 |
| 4.14.2.7 | tolower(char c) | 48 |
| 4.14.2.8 | toupper(char c) | 48 |
| 4.14.2.9 | trim(char *str) | 49 |
| 4.14.2.10 | utoa(u32int num, char *str, int base) | 49 |
| 4.15 | string.h File Reference | 49 |
| 4.15.1 | Detailed Description | 50 |
| 4.15.2 | Function Documentation | 50 |
| 4.15.2.1 | isdigit(char c) | 50 |
| 4.15.2.2 | itoa(int num, char *str, int base) | 51 |
| 4.15.2.3 | reverse(char *str, int j) | 51 |
| 4.15.2.4 | sprintf(char *buffer, char *format,...) | 51 |

| | | |
|--------------|---|-----------|
| 4.15.2.5 | <code>sprintf_internal(char *buffer, char *format, va_list valist)</code> | 52 |
| 4.15.2.6 | <code>tolower(char c)</code> | 52 |
| 4.15.2.7 | <code>toupper(char c)</code> | 52 |
| 4.15.2.8 | <code>trim(char *str)</code> | 53 |
| 4.15.2.9 | <code>utoa(u32int num, char *str, int base)</code> | 53 |
| 4.16 | <code>time.h</code> File Reference | 53 |
| 4.16.1 | Detailed Description | 55 |
| 4.16.2 | Function Documentation | 55 |
| 4.16.2.1 | <code>bcd_to_decimal(int bcd)</code> | 55 |
| 4.16.2.2 | <code>format_time(char *dest, time_h *t)</code> | 55 |
| 4.16.2.3 | <code>get_current_time()</code> | 55 |
| 4.16.2.4 | <code>set_current_time(time_h time)</code> | 56 |
| 4.17 | <code>utility.c</code> File Reference | 56 |
| 4.17.1 | Detailed Description | 57 |
| 4.17.2 | Function Documentation | 57 |
| 4.17.2.1 | <code>isnullorspace(char test)</code> | 57 |
| 4.18 | <code>utility.h</code> File Reference | 57 |
| 4.18.1 | Detailed Description | 57 |
| 4.18.2 | Function Documentation | 58 |
| 4.18.2.1 | <code>isnullorspace(char test)</code> | 58 |
| Index | | 61 |

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

| | | |
|----------------------------------|---|----|
| ALARM | Struct to hold alarm information | 5 |
| ALIAS | A struct to hold command aliases | 6 |
| COMMAND | A struct to hold commands | 6 |
| control_sequence | A struct to hold key mappings | 7 |
| fakelong | Fake 64 bit integer | 7 |
| HELP_PAGES | A struct to hold help outputs | 8 |
| node | One element within the pcb queue | 8 |
| pcb | Struct that contains all information related to a pcb | 9 |
| queue | Contains all the data needed to use/modify a queue | 10 |
| time | A struct to all the time and date elements | 11 |

Chapter 2

File Index

2.1 File List

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

| | | |
|-----------------------------------|--|----|
| alarmWrangler.h | Contains all alarm processes and internal structures | 13 |
| command_handler.h | The header file for the command handler for the Operating System | 15 |
| commands.h | The header file for commands.c | 15 |
| commandUtils.h | Utilites that apply to all command files | 21 |
| pcb_constants.h | Contains all shared resources amongst all PCBs | 27 |
| pcb_queue.h | File to hold all queue functions | 29 |
| pcb_utils.h | Utility functions for all PCBs | 32 |
| pcb_wrangler.h | Initiates the creation of all queues | 35 |
| poll_input.c | The polling input file that allows user input | 36 |
| poll_input.h | The header file for the polling input | 40 |
| procsr3.h | | ?? |
| splash.h | File to hold the splash screen | 41 |
| stdio.c | Holds all implementation of standard I/O functions | 42 |
| stdio.h | Holds all prototypes of standard I/O functions | 43 |
| string.c | Holds all utility functions used to modify strings | 44 |
| string.h | Holds all utility prototypes used to modify strings | 49 |
| time.h | The header file for the date and time functions | 53 |
| utility.c | Holds utility function implementations for this project | 56 |
| utility.h | Holds utility function prototypes for this project | 57 |

Chapter 3

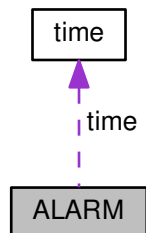
Data Structure Documentation

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 COMMAND Struct Reference

A struct to hold commands.

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

Data Fields

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

3.3.1 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](#)

3.4 control_sequence Struct Reference

A struct to hold key mappings.

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

Data Fields

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

3.4.1 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](#)

3.5 fakelong Struct Reference

Fake 64 bit integer.

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

Data Fields

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

3.5.1 Detailed Description

Fake 64 bit integer.

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

- [time.h](#)

3.6 HELP_PAGES Struct Reference

A struct to hold help outputs.

Data Fields

- char * **command_name**
- char * **command_help_page**

3.6.1 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

3.7 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**
- unsigned char **flag**

3.7.1 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](#)

3.8 pcb Struct Reference

Struct that contains all information related to a pcb.

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

Data Fields

- char * **process_name**
- u32int **process_class**
- u32int **priority**
- u32int **last_time_run**
- u32int **state**
- char **stack** [2048]
- unsigned char * **stacktop**

3.8.1 Detailed Description

Struct that contains all information related to a pcb.

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

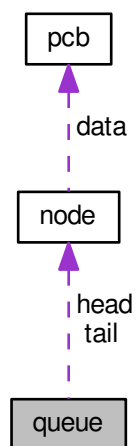
- [pcb_constants.h](#)

3.9 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**

3.9.1 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](#)

3.10 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**

3.10.1 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

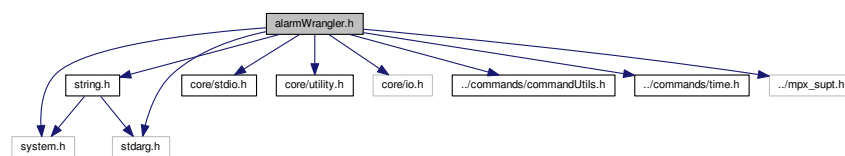
File Documentation

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.h` File Reference

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

Functions

- void [command_handler](#) ()
Entry point for the command handler.

4.2.1 Detailed Description

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

4.3 `commands.h` File Reference

The header file for `commands.c`.

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_test` (char *params)
- 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_pcb` (char *params)
Show all PCBs command.
- int `cmd_show_ready_pcb` (char *params)
Show ready PCBs command.
- int `cmd_show_blocked_pcb` (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_infinity` (char *params)
The infinity alarm for R4.

4.3.1 Detailed Description

The header file for commands.c.

4.3.2 Function Documentation

4.3.2.1 int cmd_alarm (char * *params*)

Command to set/delete/list alarms.

Returns

SUCCESS or FAILURE

4.3.2.2 int cmd_alias (char * *params*)

Command to make an alias for a command.

Returns

SUCCESS

4.3.2.3 int cmd_blockPCB (char * *params*)

command to block PCB by name

Returns

Success or Failure

4.3.2.4 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

4.3.2.5 int cmd_create_pcb (char * *params*)

Create a new pcb.

Parameters

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

Returns

SUCCESS or FAILURE

4.3.2.6 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

4.3.2.7 int cmd_delete_pcb (char * *params*)

command to delete PCB by name

Returns

Success if the PCB was removed, failure for anything else

4.3.2.8 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

4.3.2.9 int cmd_infinity (char * *params*)

The infinity alarm for R4.

Returns

SUCCESS or FAILURE

4.3.2.10 int cmd_loadr3 (char * *params*)

command to load r3 procs

Returns

SUCCESS

4.3.2.11 int cmd_resume (char * *params*)

Resume PCB command.

Returns

SUCCESS or FAILURE

4.3.2.12 int cmd_set_priority_pcb (char * *params*)

command to set PCB priority

Returns

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

4.3.2.13 int cmd_shutdown (char * *params*)

shutdown the PotatOS

Parameters

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

Returns

SUCCESS or FAILURE

4.3.2.14 int cmd_suspend (char * *params*)

Suspend PCB command.

Returns

SUCCESS or FAILURE

4.3.2.15 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

4.3.2.16 int cmd_unblock_pcb (char * *params*)

Unblock a pcb.

Parameters

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

Returns

SUCCESS or FAILURE

4.3.2.17 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

4.3.2.18 int cmd_yield (char * *params*)

command to yield control from commhand

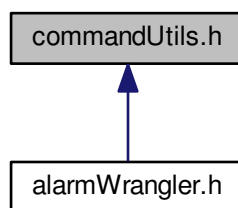
Returns

SUCCESS

4.4 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 * 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.

4.4.1 Detailed Description

Utilites that apply to all command files.

4.4.2 Macro Definition Documentation

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

`cmd_help` flags A flag binary bit shift macro

4.4.2.2 `#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

| | |
|----------------|--|
| <code>c</code> | The character to be returned as an int |
|----------------|--|

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

B flag binary bit shift macro

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

C flag binary bit shift macro

4.4.2.5 `#define CMD_SIZE 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.

4.4.2.6 #define D_FLAG (1 << 3)

D flag binary bit shift macro

4.4.2.7 #define E_FLAG (1 << 4)

E flag binary bit shift macro

4.4.2.8 #define F_FLAG (1 << 5)

F flag binary bit shift macro

4.4.2.9 #define G_FLAG (1 << 6)

G flag binary bit shift macro

4.4.2.10 #define H_FLAG (1 << 7)

H flag binary bit shift macro

4.4.2.11 #define I_FLAG (1 << 8)

I flag binary bit shift macro

4.4.2.12 #define J_FLAG (1 << 9)

J flag binary bit shift macro

4.4.2.13 #define K_FLAG (1 << 10)

K flag binary bit shift macro

4.4.2.14 #define L_FLAG (1 << 11)

L flag binary bit shift macro

4.4.2.15 #define M_FLAG (1 << 12)

M flag binary bit shift macro

4.4.2.16 #define N_FLAG (1 << 13)

N flag binary bit shift macro

4.4.2.17 #define NO_FLAG (1 << 26)

NO flag binary bit shift macro

4.4.2.18 #define O_FLAG (1 << 14)

O flag binary bit shift macro

4.4.2.19 #define P_FLAG (1 << 15)

P flag binary bit shift macro

4.4.2.20 #define Q_FLAG (1 << 16)

Q flag binary bit shift macro

4.4.2.21 #define R_FLAG (1 << 17)

R flag binary bit shift macro

4.4.2.22 #define S_FLAG (1 << 18)

S flag binary bit shift macro

4.4.2.23 #define T_FLAG (1 << 19)

T flag binary bit shift macro

4.4.2.24 #define U_FLAG (1 << 20)

U flag binary bit shift macro

4.4.2.25 #define V_FLAG (1 << 21)

V flag binary bit shift macro

4.4.2.26 `#define W_FLAG (1 << 22)`

W flag binary bit shift macro

4.4.2.27 `#define X_FLAG (1 << 24)`

X flag binary bit shift macro

4.4.2.28 `#define Y_FLAG (1 << 23)`

Y flag binary bit shift macro

4.4.2.29 `#define Z_FLAG (1 << 25)`

Z flag binary bit shift macro

4.4.3 Function Documentation

4.4.3.1 `char* get_pvalue (char c)`

Gets value of specific flag.

Usage: `get_pvalue('a');`

Parameters

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

Returns

value after the flag specified

4.4.3.2 `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 |
|------------|-------------------|

4.4.3.3 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

4.4.3.4 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

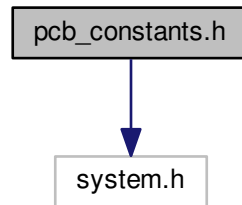
character of flag that it found

4.5 pcb_constants.h File Reference

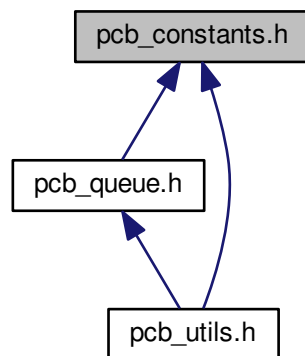
Contains all shared resources amongst all PCBs.

```
#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.

4.5.1 Detailed Description

Contains all shared resources amongst all PCBs.

4.5.2 Typedef Documentation

4.5.2.1 typedef struct [node](#) [node_t](#)

One element within the pcb queue.

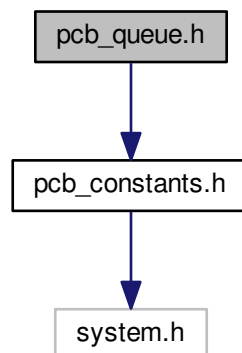
This allows us to abbreviate code elsewhere... probably

4.6 pcb_queue.h File Reference

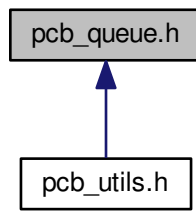
File to hold all queue functions.

```
#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.

4.6.1 Detailed Description

File to hold all queue functions.

4.6.2 Function Documentation

4.6.2.1 `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.

4.6.2.2 `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

4.6.2.3 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. |
|--------------|--|

4.6.2.4 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. |

4.6.2.5 void priority_enqueue (queue_t * cue, 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.

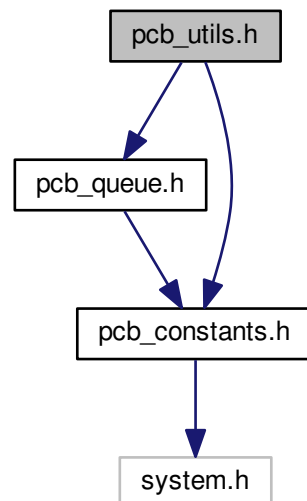
4.7 pcb_utils.h File Reference

Utility functions for all PCBs.

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

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

Include dependency graph for `pcb_utils.h`:



Functions

- `pcb_t * allocate_pcb ()`
simply allocates space for a pcb and returns that pointer
- `pcb_t * setup_pcb (char *, PROCESS_CLASS, int priority)`
command to setup new 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 *)`
- `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 ()`

4.7.1 Detailed Description

Utility functions for all PCBs.

4.7.2 Function Documentation

4.7.2.1 `pcb_t* allocate_pcb ()`

simply allocates space for a pcb and returns that pointer

Returns

pointer to allocated pcb

4.7.2.2 `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.

4.7.2.3 `int free_pcb (pcb_t * p)`

frees the space for a pcb

Returns

Success or failure

4.7.2.4 `queue_t* get_blocked_queue ()`

Getter function for the blocked queue.

Returns

A pointer to the blocked queue

4.7.2.5 `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

4.7.2.6 `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

4.7.2.7 `queue_t* get_ready_queue ()`

Getter function for the ready queue.

Returns

A pointer to the ready queue

4.7.2.8 `queue_t* get_suspended_blocked_queue ()`

Getter function for the suspended blocked queue.

Returns

A pointer to the suspended blocked queue

4.7.2.9 queue_t* get_suspended_ready_queue ()

Getter function for the suspended ready queue.

Returns

A pointer to the suspended ready queue

4.7.2.10 void kill_it__kill_it_all ()

kills the ready queue after commhand death

4.7.2.11 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

4.7.2.12 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).

4.7.2.13 pcb_t* setup_pcb (char * pname, PROCESS_CLASS pclass, int priority)

command to setup new PCB

Returns

Success if the PCB was created, failure for anything else

4.8 pcb_wrangler.h File Reference

Initiates the creation of all queues.

Functions

- void **init_process_queues** ()

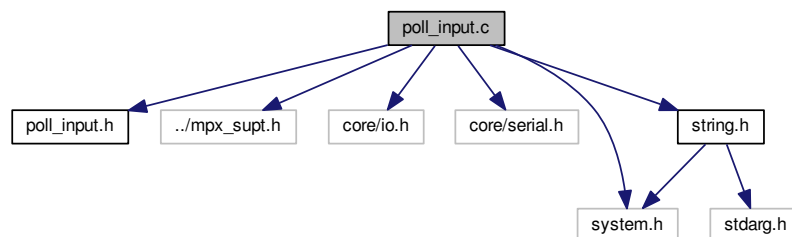
4.8.1 Detailed Description

Initiates the creation of all queues.

4.9 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
- #define **LEAVE_TIME** 700

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 `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.

4.9.1 Detailed Description

The polling input file that allows user input.

4.9.2 Function Documentation

4.9.2.1 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

4.9.2.2 int `input_available` ()

Checks for input on COM1.

Returns

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

4.9.2.3 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. |

4.9.2.4 void move_cursor (int *n*)

Moves the cursor *n* characters.

Parameters

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

4.9.2.5 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

4.9.2.6 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 |
|------------|--------------------------------------|

4.9.2.7 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

4.9.3 Variable Documentation

4.9.3.1 `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.

4.9.3.2 `const int TOLERANCE = 300`

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

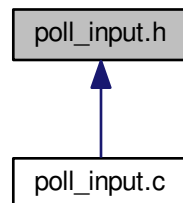
Note

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

4.10 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 [poll_input](#) (char *buffer, int *length)
Polls COM1 for input and puts it into buffer.

4.10.1 Detailed Description

The header file for the polling input.

4.10.2 Typedef Documentation

4.10.2.1 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 |

4.10.3 Function Documentation

4.10.3.1 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

4.11 splash.h File Reference

File to hold the splash screen.

Functions

- void [draw_splash](#) ()
draw the splash screen

4.11.1 Detailed Description

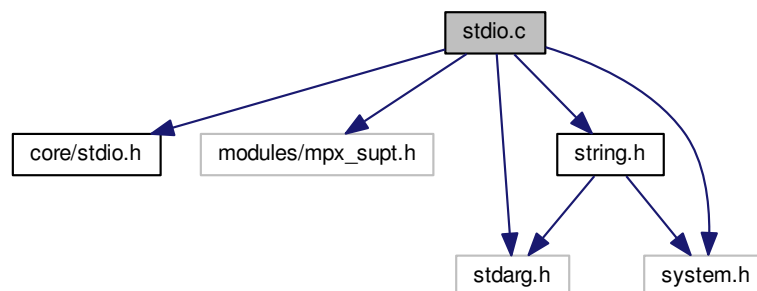
File to hold the splash screen.

4.12 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

4.12.1 Detailed Description

Holds all implementation of standard I/O functions.

4.12.2 Function Documentation

4.12.2.1 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

4.12.2.2 int puts (char * buff)

prints out a string to DEFAULT_DEVICE

Parameters

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

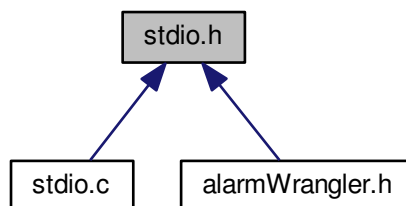
Returns

1

4.13 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

4.13.1 Detailed Description

Holds all prototypes of standard I/O functions.

4.13.2 Function Documentation

4.13.2.1 `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

4.13.2.2 `int puts (char * buff)`

prints out a string to DEFAULT_DEVICE

Parameters

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

Returns

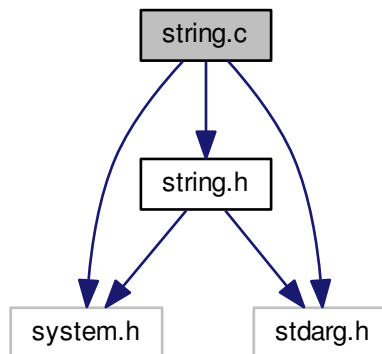
1

4.14 string.c File Reference

Holds all utility functions used to modify strings.

```
#include <string.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.

4.14.1 Detailed Description

Holds all utility functions used to modify strings.

4.14.2 Function Documentation

4.14.2.1 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;

4.14.2.2 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

4.14.2.3 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

4.14.2.4 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

4.14.2.5 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

4.14.2.6 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

4.14.2.7 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

4.14.2.8 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

4.14.2.9 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

4.14.2.10 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*

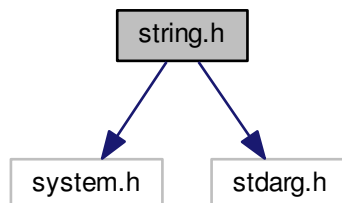
4.15 string.h File Reference

Holds all utility prototypes used to modify strings.

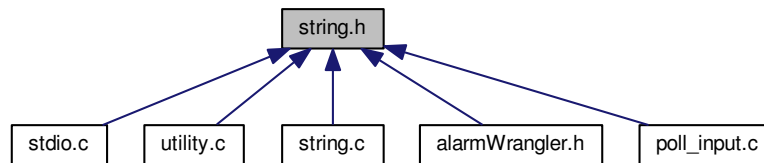
```
#include <system.h>
```

```
#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.*

4.15.1 Detailed Description

Holds all utility prototypes used to modify strings.

4.15.2 Function Documentation

4.15.2.1 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;

4.15.2.2 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*

4.15.2.3 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*

4.15.2.4 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 |
| Generated by Doxygen vars | Variable list with parameters matching the format |

Returns

pointer to buffer

4.15.2.5 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

4.15.2.6 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

4.15.2.7 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

4.15.2.8 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

4.15.2.9 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 |

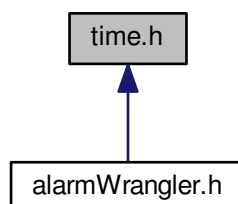
Returns

pointer to *str*

4.16 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 **HOURL_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)

4.16.1 Detailed Description

The header file for the date and time functions.

4.16.2 Function Documentation

4.16.2.1 `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.

4.16.2.2 `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.

4.16.2.3 `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.

4.16.2.4 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.

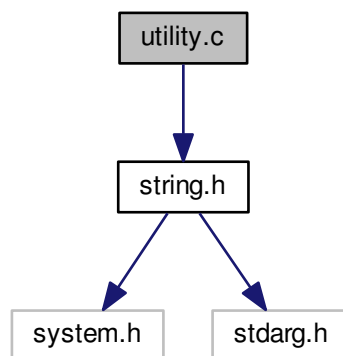
Setting a value in the input struct to a '-1' will skip the value in setting the time. Essentially, keeping the value as it was before. This is demonstrated in the commands.c file.

4.17 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.

4.17.1 Detailed Description

Holds utility function implementations for this project.

4.17.2 Function Documentation

4.17.2.1 `int isnullorspace (char test)`

Determines if a passed character is a null or space.

Parameters

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

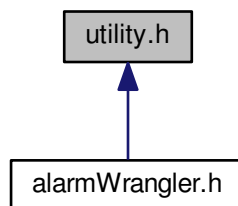
Returns

1 if space or null, 0 otherwise

4.18 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.

4.18.1 Detailed Description

Holds utility function prototypes for this project.

4.18.2 Function Documentation

4.18.2.1 `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, [23](#)
- ALARM, [5](#)
- ALIAS, [6](#)
- alarmProcess
 - alarmWrangler.h, [14](#)
- alarmWrangler.h, [13](#)
 - alarmProcess, [14](#)
 - check, [14](#)
 - insertAlarm, [14](#)
 - listAlarms, [14](#)
 - removeAlarm, [15](#)
- allocate_pcb
 - pcb_utils.h, [33](#)
- alphanum
 - commandUtils.h, [23](#)
- B_FLAG
 - commandUtils.h, [23](#)
- bcd_to_decimal
 - time.h, [55](#)
- C_FLAG
 - commandUtils.h, [23](#)
- CMDSIZE
 - commandUtils.h, [23](#)
- COMMAND, [6](#)
- check
 - alarmWrangler.h, [14](#)
- cmd_alarm
 - commands.h, [17](#)
- cmd_alias
 - commands.h, [17](#)
- cmd_blockPCB
 - commands.h, [17](#)
- cmd_clear
 - commands.h, [17](#)
- cmd_create_pcb
 - commands.h, [17](#)
- cmd_date
 - commands.h, [18](#)
- cmd_delete_pcb
 - commands.h, [18](#)
- cmd_help
 - commands.h, [18](#)
- cmd_infinity
 - commands.h, [19](#)
- cmd_loadr3
 - commands.h, [19](#)
- cmd_resume
 - commands.h, [19](#)
- cmd_set_priority_pcb
 - commands.h, [19](#)
- cmd_shutdown
 - commands.h, [19](#)
- cmd_suspend
 - commands.h, [20](#)
- cmd_time
 - commands.h, [20](#)
- cmd_unblock_pcb
 - commands.h, [20](#)
- cmd_version
 - commands.h, [20](#)
- cmd_yield
 - commands.h, [21](#)
- command_handler.h, [15](#)
- commandUtils.h, [21](#)
 - A_FLAG, [23](#)
 - alphanum, [23](#)
 - B_FLAG, [23](#)
 - C_FLAG, [23](#)
 - CMDSIZE, [23](#)
 - D_FLAG, [23](#)
 - E_FLAG, [24](#)
 - F_FLAG, [24](#)
 - G_FLAG, [24](#)
 - get_pvalue, [26](#)
 - H_FLAG, [24](#)
 - I_FLAG, [24](#)
 - J_FLAG, [24](#)
 - K_FLAG, [24](#)
 - L_FLAG, [24](#)
 - M_FLAG, [24](#)
 - N_FLAG, [24](#)
 - NO_FLAG, [25](#)
 - O_FLAG, [25](#)
 - P_FLAG, [25](#)
 - Q_FLAG, [25](#)
 - R_FLAG, [25](#)
 - S_FLAG, [25](#)
 - search_commands, [26](#)
 - set_flags, [27](#)
 - set_flags_search_alias, [27](#)
 - T_FLAG, [25](#)
 - U_FLAG, [25](#)
 - V_FLAG, [25](#)
 - W_FLAG, [25](#)
 - X_FLAG, [26](#)
 - Y_FLAG, [26](#)

- Z_FLAG, 26
- commands.h, 15
 - cmd_alarm, 17
 - cmd_alias, 17
 - cmd_blockPCB, 17
 - cmd_clear, 17
 - cmd_create_pcb, 17
 - cmd_date, 18
 - cmd_delete_pcb, 18
 - cmd_help, 18
 - cmd_infinity, 19
 - cmd_loadr3, 19
 - cmd_resume, 19
 - cmd_set_priority_pcb, 19
 - cmd_shutdown, 19
 - cmd_suspend, 20
 - cmd_time, 20
 - cmd_unblock_pcb, 20
 - cmd_version, 20
 - cmd_yield, 21
- construct_queue
 - pcb_queue.h, 30
- control_sequence, 7
- control_sequences
 - poll_input.c, 39
- ControlSequence
 - poll_input.h, 41
- D_FLAG
 - commandUtils.h, 23
- dequeue
 - pcb_queue.h, 30
- destruct_queue
 - pcb_queue.h, 31
- E_FLAG
 - commandUtils.h, 24
- enqueue
 - pcb_queue.h, 31
- F_FLAG
 - commandUtils.h, 24
- fakelong, 7
- find_pcb
 - pcb_utils.h, 33
- format_time
 - time.h, 55
- free_pcb
 - pcb_utils.h, 33
- G_FLAG
 - commandUtils.h, 24
- get_blocked_queue
 - pcb_utils.h, 33
- get_current_time
 - time.h, 55
- get_key
 - poll_input.c, 37
- get_process_class_string
 - pcb_utils.h, 34
- get_process_state_string
 - pcb_utils.h, 34
- get_pvalue
 - commandUtils.h, 26
- get_ready_queue
 - pcb_utils.h, 34
- get_suspended_blocked_queue
 - pcb_utils.h, 34
- get_suspended_ready_queue
 - pcb_utils.h, 34
- H_FLAG
 - commandUtils.h, 24
- HELP_PAGES, 8
- I_FLAG
 - commandUtils.h, 24
- input_available
 - poll_input.c, 37
- insertAlarm
 - alarmWrangler.h, 14
- isdigit
 - string.c, 46
 - string.h, 50
- isnullorspace
 - utility.c, 57
 - utility.h, 58
- itoa
 - string.c, 46
 - string.h, 51
- J_FLAG
 - commandUtils.h, 24
- K_FLAG
 - commandUtils.h, 24
- kill_it__kill_it_all
 - pcb_utils.h, 35
- L_FLAG
 - commandUtils.h, 24
- listAlarms
 - alarmWrangler.h, 14
- M_FLAG
 - commandUtils.h, 24
- memcpy
 - poll_input.c, 37
- move_cursor
 - poll_input.c, 38
- N_FLAG
 - commandUtils.h, 24
- NO_FLAG
 - commandUtils.h, 25
- node, 8
- node_t
 - pcb_constants.h, 29

O_FLAG
 commandUtils.h, 25

P_FLAG
 commandUtils.h, 25

pcb, 9

pcb_constants.h, 27
 node_t, 29

pcb_queue.h, 29
 construct_queue, 30
 dequeue, 30
 destruct_queue, 31
 enqueue, 31
 priority_enqueue, 31

pcb_utils.h, 32
 allocate_pcb, 33
 find_pcb, 33
 free_pcb, 33
 get_blocked_queue, 33
 get_process_class_string, 34
 get_process_state_string, 34
 get_ready_queue, 34
 get_suspended_blocked_queue, 34
 get_suspended_ready_queue, 34
 kill_it__kill_it_all, 35
 print_pcb_info, 35
 remove_pcb, 35
 setup_pcb, 35

pcb_wrangler.h, 35

poll_input
 poll_input.c, 38
 poll_input.h, 41

poll_input.c, 36
 control_sequences, 39
 get_key, 37
 input_available, 37
 memcpy, 37
 move_cursor, 38
 poll_input, 38
 print_after_cursor, 38
 TOLERANCE, 39
 wait_for_input, 38

poll_input.h, 40
 ControlSequence, 41
 poll_input, 41

print_after_cursor
 poll_input.c, 38

print_pcb_info
 pcb_utils.h, 35

printf
 stdio.c, 42
 stdio.h, 44

priority_enqueue
 pcb_queue.h, 31

puts
 stdio.c, 43
 stdio.h, 44

Q_FLAG
 commandUtils.h, 25

queue, 10

R_FLAG
 commandUtils.h, 25

remove_pcb
 pcb_utils.h, 35

removeAlarm
 alarmWrangler.h, 15

reverse
 string.c, 46
 string.h, 51

S_FLAG
 commandUtils.h, 25

search_commands
 commandUtils.h, 26

set_current_time
 time.h, 55

set_flags
 commandUtils.h, 27

set_flags_search_alias
 commandUtils.h, 27

setup_pcb
 pcb_utils.h, 35

splash.h, 41

sprintf
 string.c, 47
 string.h, 51

sprintf_internal
 string.c, 47
 string.h, 52

sprintf_pad_helper
 string.c, 48

stdio.c, 42
 printf, 42
 puts, 43

stdio.h, 43
 printf, 44
 puts, 44

string.c, 44
 isdigit, 46
 itoa, 46
 reverse, 46
 sprintf, 47
 sprintf_internal, 47
 sprintf_pad_helper, 48
 tolower, 48
 toupper, 48
 trim, 48
 utoa, 49

string.h, 49
 isdigit, 50
 itoa, 51
 reverse, 51
 sprintf, 51
 sprintf_internal, 52
 tolower, 52
 toupper, 52

- trim, [52](#)
- utoa, [53](#)
- T_FLAG
 - commandUtils.h, [25](#)
- TOLERANCE
 - poll_input.c, [39](#)
- time, [11](#)
- time.h, [53](#)
 - bcd_to_decimal, [55](#)
 - format_time, [55](#)
 - get_current_time, [55](#)
 - set_current_time, [55](#)
- tolower
 - string.c, [48](#)
 - string.h, [52](#)
- toupper
 - string.c, [48](#)
 - string.h, [52](#)
- trim
 - string.c, [48](#)
 - string.h, [52](#)
- U_FLAG
 - commandUtils.h, [25](#)
- utility.c, [56](#)
 - isnullorspace, [57](#)
- utility.h, [57](#)
 - isnullorspace, [58](#)
- utoa
 - string.c, [49](#)
 - string.h, [53](#)
- V_FLAG
 - commandUtils.h, [25](#)
- W_FLAG
 - commandUtils.h, [25](#)
- wait_for_input
 - poll_input.c, [38](#)
- X_FLAG
 - commandUtils.h, [26](#)
- Y_FLAG
 - commandUtils.h, [26](#)
- Z_FLAG
 - commandUtils.h, [26](#)