

MPX Thunder Krakens

R1

Generated by Doxygen 1.8.11

Contents

1	Main Page	1
2	Data Structure Index	3
2.1	Data Structures	3
3	File Index	5
3.1	File List	5
4	Data Structure Documentation	7
4.1	function_name Struct Reference	7
4.1.1	Detailed Description	7
4.1.2	Field Documentation	7
4.1.2.1	function	7
4.1.2.2	help	7
4.1.2.3	nameStr	8
4.1.2.4	usage	8

5 File Documentation	9
5.1 documentation/mainpage.dox File Reference	9
5.2 include/core/serial.h File Reference	9
5.2.1 Detailed Description	10
5.2.2 Macro Definition Documentation	10
5.2.2.1 COM1	10
5.2.2.2 COM2	10
5.2.2.3 COM3	11
5.2.2.4 COM4	11
5.2.2.5 WithEcho	11
5.2.2.6 WithoutEcho	11
5.2.3 Function Documentation	11
5.2.3.1 get_input_line(char *buffer, const int buffer_size, const int bWithEcho)	11
5.2.3.2 init_serial(int device)	11
5.2.3.3 serial_print(const char *msg)	11
5.2.3.4 serial_println(const char *msg)	12
5.2.3.5 set_serial_in(int device)	12
5.2.3.6 set_serial_out(int device)	12
5.3 include/string.h File Reference	12
5.3.1 Detailed Description	16
5.3.2 Function Documentation	16
5.3.2.1 atoi(const char *s)	16
5.3.2.2 isspace(const char *c)	17
5.3.2.3 memset(void *s, int c, size_t n)	17
5.3.2.4 printf(const char *format,...)	17
5.3.2.5 sprintf(char *str, const char *format,...)	18
5.3.2.6 strcat(char *s1, const char *s2)	18
5.3.2.7 strcmp(const char *s1, const char *s2)	18

5.3.2.8	strcpy(char *s1, const char *s2)	19
5.3.2.9	strlen(const char *s)	19
5.3.2.10	strtok(char *s1, const char *s2)	20
5.4	lib/string.c File Reference	20
5.4.1	Detailed Description	24
5.4.2	Function Documentation	24
5.4.2.1	atoi(const char *s)	24
5.4.2.2	isspace(const char *c)	25
5.4.2.3	memset(void *s, int c, size_t n)	25
5.4.2.4	printf(const char *format,...)	25
5.4.2.5	sprintf(char *str, const char *format,...)	26
5.4.2.6	strcat(char *s1, const char *s2)	26
5.4.2.7	strcmp(const char *s1, const char *s2)	26
5.4.2.8	strcpy(char *s1, const char *s2)	27
5.4.2.9	strlen(const char *s)	27
5.4.2.10	strtok(char *s1, const char *s2)	28
5.5	modules/errno.h File Reference	28
5.5.1	Detailed Description	29
5.5.2	Macro Definition Documentation	29
5.5.2.1	E_INVPARA	29
5.5.2.2	E_INVSTRF	29
5.5.2.3	E_INVUSRI	29
5.5.2.4	E_NOERROR	29
5.5.3	Typedef Documentation	29
5.5.3.1	error_t	29
5.6	modules/r1/r1.c File Reference	29
5.6.1	Detailed Description	33
5.6.2	Macro Definition Documentation	33

5.6.2.1	COMPLETION	33
5.6.2.2	MAX_ARGC	33
5.6.2.3	MOD_VERSION	33
5.6.2.4	USER_INPUT_BUFFER_SIZE	33
5.6.3	Enumeration Type Documentation	33
5.6.3.1	CommandPaserStat	33
5.6.4	Function Documentation	34
5.6.4.1	command_line_parser(const char *CmdStr, int *argc, char **argv, const int MaxArgNum, const int MaxStrLen)	34
5.6.4.2	commhand()	35
5.6.4.3	print_help(const int function_index)	35
5.7	modules/r1/r1.h File Reference	36
5.7.1	Detailed Description	37
5.7.2	Macro Definition Documentation	38
5.7.2.1	GETDATE	38
5.7.2.2	GETTIME	38
5.7.2.3	HELP	38
5.7.2.4	NUM_OF_FUNCTIONS	38
5.7.2.5	SETDATE	38
5.7.2.6	SETTIME	38
5.7.2.7	SHUTDOWN	38
5.7.2.8	VERSION	38
5.7.3	Function Documentation	38
5.7.3.1	command_line_parser(const char *CmdStr, int *argc, char **argv, const int MaxArgNum, const int MaxStrLen)	38
5.7.3.2	commhand()	39
5.7.3.3	print_help(const int function_index)	40
5.8	modules/r1/sys_clock.c File Reference	40
5.8.1	Detailed Description	44

5.8.2	Macro Definition Documentation	45
5.8.2.1	RTC_INDEX_DAY_MONTH	45
5.8.2.2	RTC_INDEX_DAY_WEEK	45
5.8.2.3	RTC_INDEX_HOUR	45
5.8.2.4	RTC_INDEX_HOUR_ALARM	45
5.8.2.5	RTC_INDEX_MINUTE	45
5.8.2.6	RTC_INDEX_MINUTE_ALARM	45
5.8.2.7	RTC_INDEX_MONTH	45
5.8.2.8	RTC_INDEX_SECOND	45
5.8.2.9	RTC_INDEX_SECOND_ALARM	45
5.8.2.10	RTC_INDEX_YEAR	45
5.8.3	Function Documentation	45
5.8.3.1	get_date(date_time *dateTimeValues)	45
5.8.3.2	get_date_main(int argc, char **argv)	46
5.8.3.3	get_time(date_time *dateTimeValues)	46
5.8.3.4	get_time_main(int argc, char **argv)	47
5.8.3.5	set_date(const date_time *dateTimeValues)	47
5.8.3.6	set_date_main(int argc, char **argv)	48
5.8.3.7	set_date_str(const char *str)	49
5.8.3.8	set_time(const date_time *dateTimeValues)	49
5.8.3.9	set_time_main(int argc, char **argv)	50
5.8.3.10	set_time_str(const char *timeStr)	51
5.9	modules/r1/sys_clock.h File Reference	51
5.9.1	Detailed Description	55
5.9.2	Function Documentation	55
5.9.2.1	get_date(date_time *dateTimeValues)	56
5.9.2.2	get_date_main(int argc, char **argv)	56
5.9.2.3	get_time(date_time *dateTimeValues)	57
5.9.2.4	get_time_main(int argc, char **argv)	57
5.9.2.5	set_date(const date_time *dateTimeValues)	58
5.9.2.6	set_date_main(int argc, char **argv)	58
5.9.2.7	set_date_str(const char *str)	59
5.9.2.8	set_time(const date_time *dateTimeValues)	59
5.9.2.9	set_time_main(int argc, char **argv)	60
5.9.2.10	set_time_str(const char *timeStr)	61

Chapter 1

Main Page

Welcome to the Programmer's manual for the Thunder Kracken's MPX Operating system. This document catalogues all of the information one may need to know regarding the use and modification of this Operating system and its contents. Included is a complete API of every method created for the operating system which includes all inputs and outputs as well as a brief summary of the purpose of each method. This will give you a more in depth look at all of the ordinary user commands as well as the internal commands used to perform functions that normal users cannot access. Most likely these commands will be the most important for making new programs on the operating system. This document also lists the documentation for the files files in the operating system. This includes all of the variables and methods used in each file. These will help direct you as to where certain functions are defined. For general usage tips, please refer to the user manual. We hope you find working with the Thunder Kracken's MPX Operating System as enjoyable as we do and we thank you for using our product.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

function_name	
A structure to represent each function	7

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

include/ string.h	Many usefull functions that used for handling string	12
include/core/ serial.h	Serial - Header	9
lib/ string.c	Many usefull functions that used for handling string	20
modules/ errno.h	This file contains the type of errors. The error can be from invalid paramter passed to a function, or invalid input format	28
modules/r1/ r1.c	The commandhander and functions associations for Module R1	29
modules/r1/ r1.h	The commandhander and functions associations for Module R1	36
modules/r1/ sys_clock.c	The main file that manipulates and controls the system's clock	40
modules/r1/ sys_clock.h	The main file that manipulates and controls the system's clock	51

Chapter 4

Data Structure Documentation

4.1 function_name Struct Reference

A structure to represent each function.

Data Fields

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

4.1.1 Detailed Description

A structure to represent each function.

4.1.2 Field Documentation

4.1.2.1 int(* function_name::function) (int argc, char **argv)

the function

4.1.2.2 char* function_name::help

function's help information

4.1.2.3 `char* function_name::nameStr`

function's name

4.1.2.4 `char* function_name::usage`

function's usage or use cases

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

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

Chapter 5

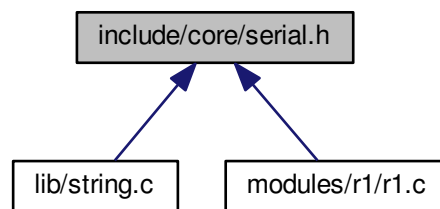
File Documentation

5.1 documentation/mainpage.dox File Reference

5.2 include/core/serial.h File Reference

Serial - Header.

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



Macros

- #define **COM1** 0x3f8
- #define **COM2** 0x2f8
- #define **COM3** 0x3e8
- #define **COM4** 0x2e8
- #define **WithoutEcho** 0
- #define **WithEcho** 1

Functions

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

get_input_line

Get user's input from keyboard.

Parameters

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

Returns

VOID

- void [get_input_line](#) (char *buffer, const int buffer_size, const int bWithEcho)

5.2.1 Detailed Description

Serial - Header.

Author

Thunder Krakens

Date

February 2nd, 2016

Version

R1

5.2.2 Macro Definition Documentation

5.2.2.1 `#define COM1 0x3f8`

5.2.2.2 `#define COM2 0x2f8`

5.2.2.3 `#define COM3 0x3e8`

5.2.2.4 `#define COM4 0x2e8`

5.2.2.5 `#define WithEcho 1`

5.2.2.6 `#define WithoutEcho 0`

5.2.3 Function Documentation

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

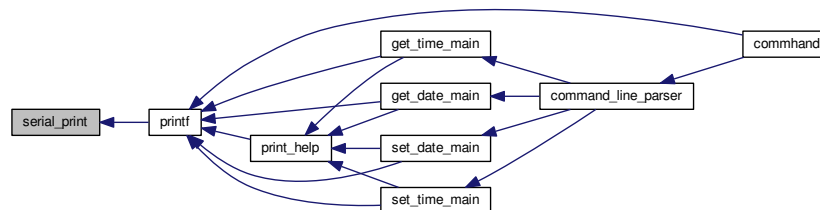
Here is the caller graph for this function:



5.2.3.2 `int init_serial (int device)`

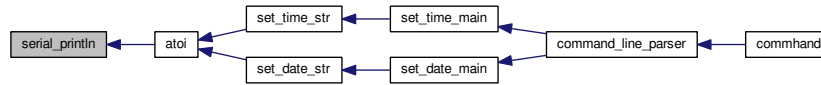
5.2.3.3 `int serial_print (const char * msg)`

Here is the caller graph for this function:



5.2.3.4 int serial_println (const char * msg)

Here is the caller graph for this function:



5.2.3.5 int set_serial_in (int device)

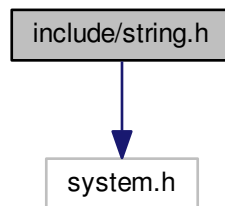
5.2.3.6 int set_serial_out (int device)

5.3 include/string.h File Reference

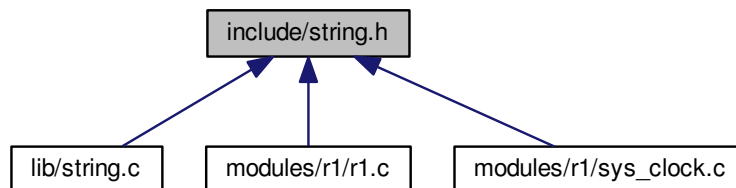
Many usefull functions that used for handling string.

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

Include dependency graph for string.h:



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



Functions

isspace.

Identifies if its space

Parameters

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

Returns

1 if it is space, otherwise return 0.

- int `isspace` (const char *c)

memset.

Sets region of memory

Parameters

s	destination
c	byte to write
n	count

Returns

the pointer to the memory space.

- void * `memset` (void *s, int c, size_t n)

strcpy.

Copies one string to another.

Parameters

s1	Destination string
s2	Source string

Returns

pointer to the destination String

- char * `strcpy` (char *s1, const char *s2)

strcat.

Concatenate the contents of one string onto another.

Parameters

s1	Destination string
s2	Source string

Returns

pointer to destination String

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

strlen.

Returns the length of a string.

Parameters

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

Returns

count Length of the String

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

strcmp.

String comparison.

Parameters

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

Returns

whether they are the same or not.

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

strtok.

Split string into tokens.

Parameters

s1	String
s2	Delimiter

Returns

the pointer to the token.

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

atoi.

Convert an ASCII string to an integer.

Parameters

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

Returns

The converted integer.

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

sprintf.

Generate a formatted string.

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

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

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

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

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

Parameters

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

Returns

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

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

printf.

Print out a formatted string.

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

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

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

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

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

Parameters

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

Returns

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

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

5.3.1 Detailed Description

Many usefull functions that used for handling string.

Author

Thunder Krakens

Date

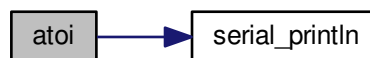
February 2nd, 2016

Version

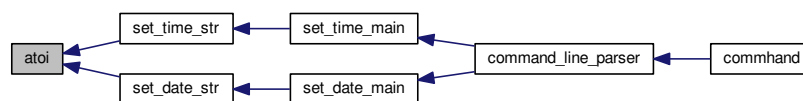
R1

5.3.2 Function Documentation**5.3.2.1 int atoi (const char * s)**

Here is the call graph for this function:



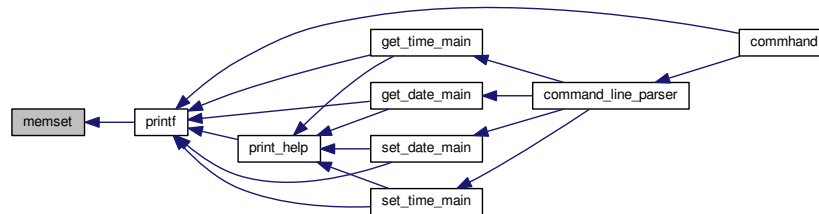
Here is the caller graph for this function:



5.3.2.2 `int isspace (const char * c)`

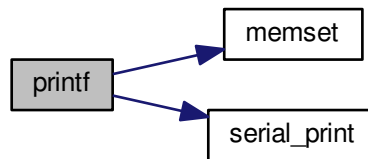
5.3.2.3 `void* memset (void * s, int c, size_t n)`

Here is the caller graph for this function:

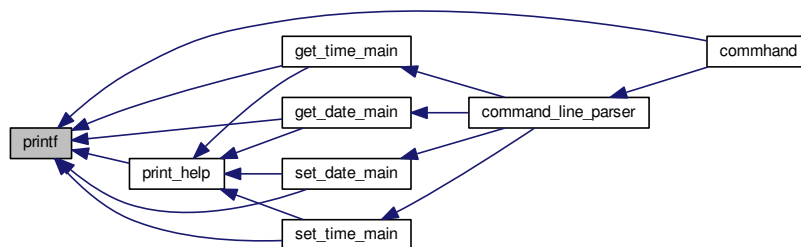


5.3.2.4 `int printf (const char * format, ...)`

Here is the call graph for this function:



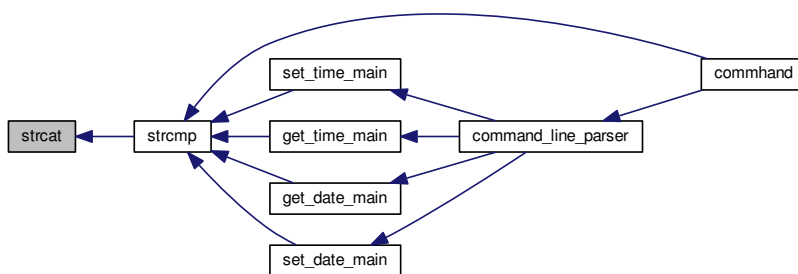
Here is the caller graph for this function:



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

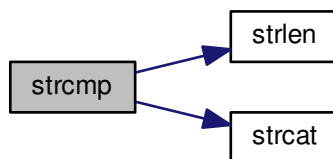
5.3.2.6 `char* strcat (char * s1, const char * s2)`

Here is the caller graph for this function:

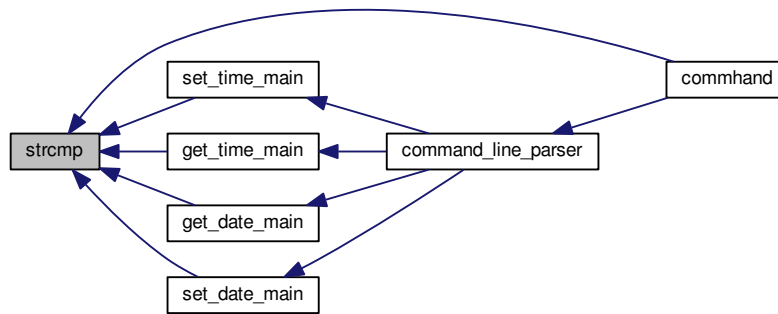


5.3.2.7 `int strcmp (const char * s1, const char * s2)`

Here is the call graph for this function:

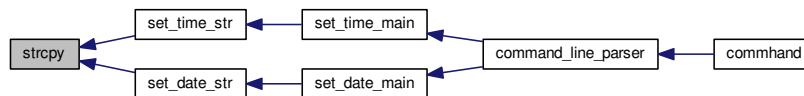


Here is the caller graph for this function:



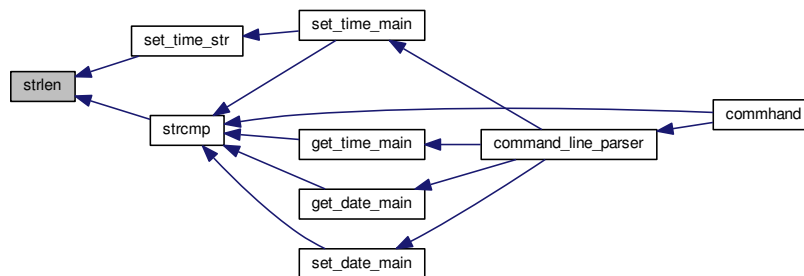
5.3.2.8 `char* strcpy (char * s1, const char * s2)`

Here is the caller graph for this function:



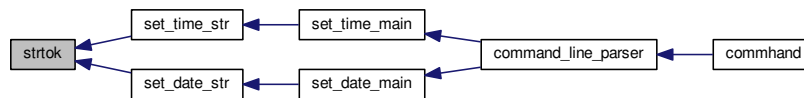
5.3.2.9 `int strlen (const char * s)`

Here is the caller graph for this function:



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

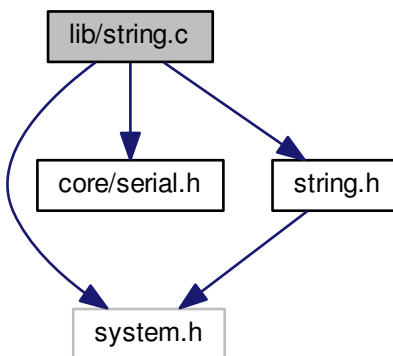
Here is the caller graph for this function:



5.4 lib/string.c File Reference

Many usefull functions that used for handling string.

```
#include <system.h>
#include <core/serial.h>
#include <string.h>
Include dependency graph for string.c:
```



Functions

`strlen.`

Returns the length of a string.

Parameters

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

Returns

count Length of the String

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

strcpy.

Copies one string to another.

Parameters

s1	<i>Destination string</i>
s2	<i>Source string</i>

Returns

pointer to the destination String

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

atoi.

Convert an ASCII string to an integer.

Parameters

s	<i>String.</i>
---	----------------

Returns

The converted integer.

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

strcmp.

String comparison.

Parameters

s1	<i>First string to use for the compare.</i>
s2	<i>Second string to use for the compare.</i>

Returns

whether they are the same or not.

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

ParsePadding.

Parse the number for padding.

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

Parameters

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

Returns

bIsValid Returns the validity.

AddPad.

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

Parameters

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

Returns

VOID

NibbleToChar

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

Parameters

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

Returns

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

bytesToHexString.

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

Parameters

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

Returns

VOID

vsprintf.

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

Parameters

str	Output string.
format	The format of the string.
ap	the pointer of the first additional parameter.

Returns

0

sprintf.

Generate a formatted string.

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

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

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

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

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

Parameters

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

Returns

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

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

printf.

Print out a formatted string.

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

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

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

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

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

Parameters

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

Returns

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

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

5.4.1 Detailed Description

Many usefull functions that used for handling string.

Author

Thunder Krakens

Date

February 2nd, 2016

Version

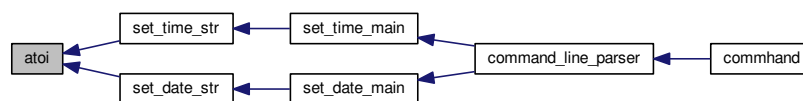
R1

5.4.2 Function Documentation**5.4.2.1 int atoi (const char * s)**

Here is the call graph for this function:



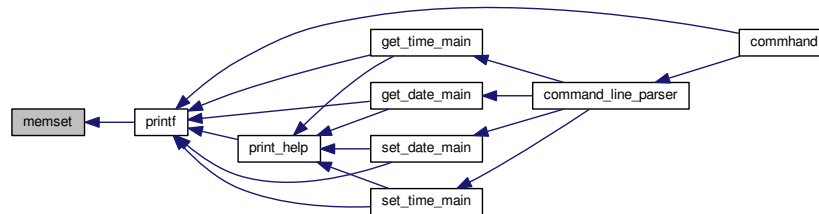
Here is the caller graph for this function:



5.4.2.2 `int isspace (const char * c)`

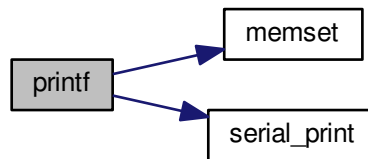
5.4.2.3 `void* memset (void * s, int c, size_t n)`

Here is the caller graph for this function:

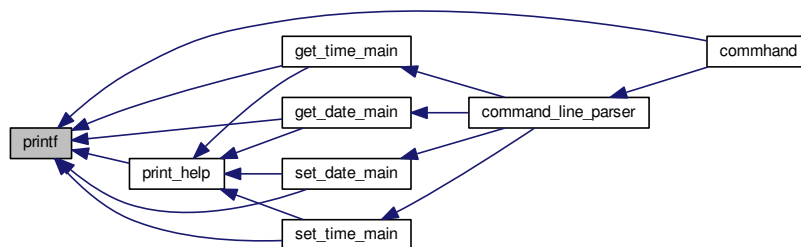


5.4.2.4 `int printf (const char * format, ...)`

Here is the call graph for this function:



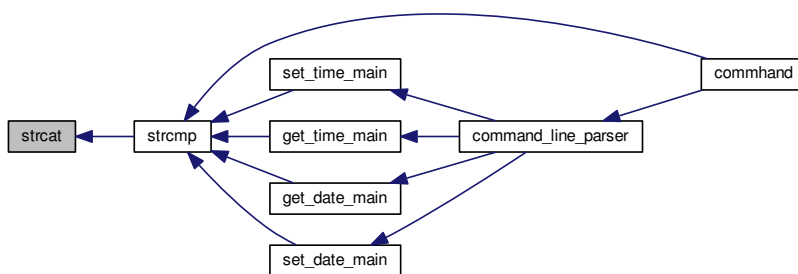
Here is the caller graph for this function:



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

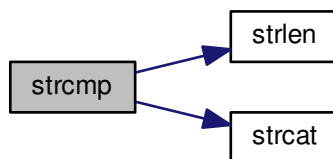
5.4.2.6 `char* strcat (char * s1, const char * s2)`

Here is the caller graph for this function:

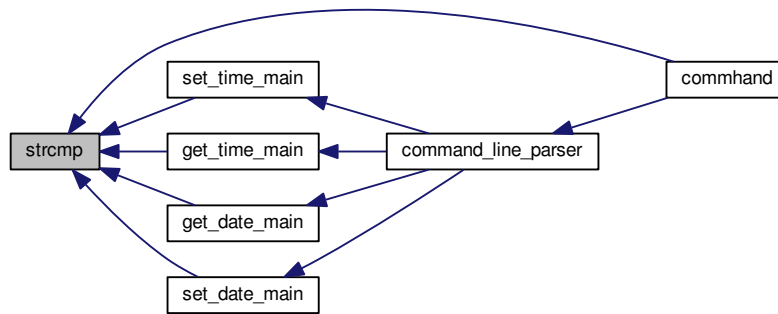


5.4.2.7 `int strcmp (const char * s1, const char * s2)`

Here is the call graph for this function:

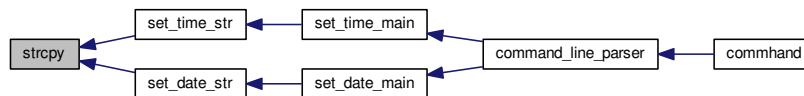


Here is the caller graph for this function:



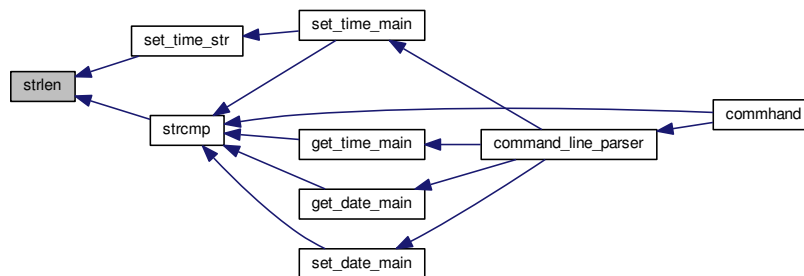
5.4.2.8 char* strcpy (char * s1, const char * s2)

Here is the caller graph for this function:



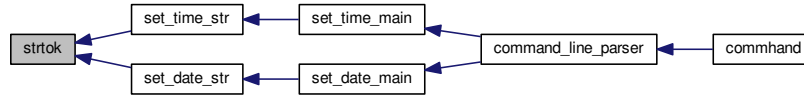
5.4.2.9 int strlen (const char * s)

Here is the caller graph for this function:



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

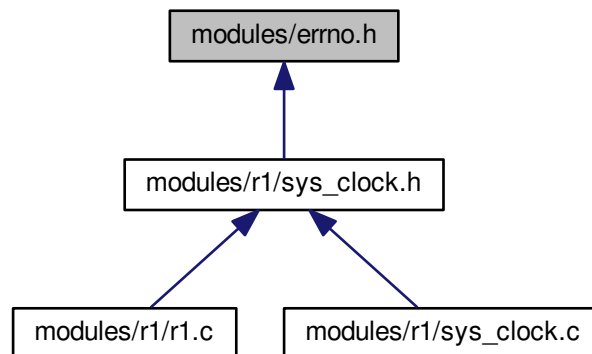
Here is the caller graph for this function:



5.5 modules/errno.h File Reference

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

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



Macros

- `#define E_NOERROR 0`
- `#define E_INVPARA 1`
- `#define E_INVSTRF 2`
- `#define E_INVUSRI 3`

Typedefs

`error_t`.

The datatype that holds the error code.

- `typedef unsigned int error_t`

5.5.1 Detailed Description

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

Author

Thunder Krakens

Date

February 2nd, 2016

Version

R1

5.5.2 Macro Definition Documentation

5.5.2.1 `#define E_INVPARA 1`

5.5.2.2 `#define E_INVSTRF 2`

5.5.2.3 `#define E_INVUSRI 3`

5.5.2.4 `#define E_NOERROR 0`

5.5.3 Typedef Documentation

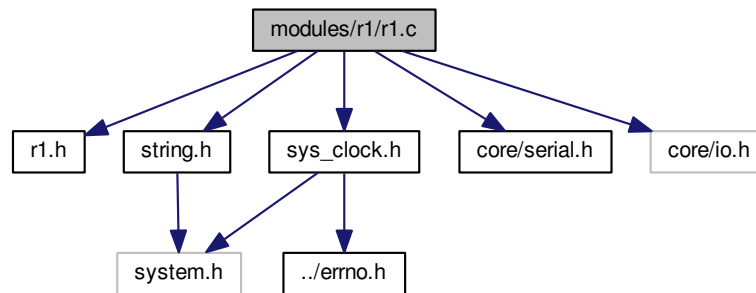
5.5.3.1 `typedef unsigned int error_t`

5.6 modules/r1/r1.c File Reference

The commandhandler and functions associations for Module R1.

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

Include dependency graph for r1.c:



Data Structures

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

Macros

- #define [USER_INPUT_BUFFER_SIZE](#) 1000
- #define [MAX_ARGC](#) 50
- #define [MOD_VERSION](#) "R1"
- #define [COMPLETION](#) "02/05/2016"

Enumerations

CommandParserStat

The status of the command parser

- enum [CommandPaserStat](#) { [NotWriting](#), [NormalWriting](#), [DoubleQuoteWriting](#), [SingleQuoteWriting](#) }

Functions

exe_function.

Executes the specific fuction.

Parameters

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

Returns

0

version

displays the version of the system currently running.

Parameters

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

Returns

0

shutdown

Closes all functions, and shuts down the system.

Parameters

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

Returns

0 for shutdown, 1 for keep running.

help_usages

shows usage message for each function.

Parameters

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

Returns

0

help_function

displays help text for all functions.

Parameters

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

Returns

0

commhand

Accepts and handles commands from the user.

Returns

0

- int [commhand](#) ()

command_line_parser

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

Parameters

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

Returns

void

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

print_help

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

Parameters

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

Returns

void

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

5.6.1 Detailed Description

The commandhandler and functions associations for Module R1.

Author

Thunder Krakens

Date

February 2nd, 2016

Version

R1

5.6.2 Macro Definition Documentation

5.6.2.1 `#define COMPLETION "02/05/2016"`

5.6.2.2 `#define MAX_ARGC 50`

5.6.2.3 `#define MOD_VERSION "R1"`

5.6.2.4 `#define USER_INPUT_BUFFER_SIZE 1000`

5.6.3 Enumeration Type Documentation

5.6.3.1 `enum CommandPaserStat`

Enumerator

NotWriting

NormalWriting

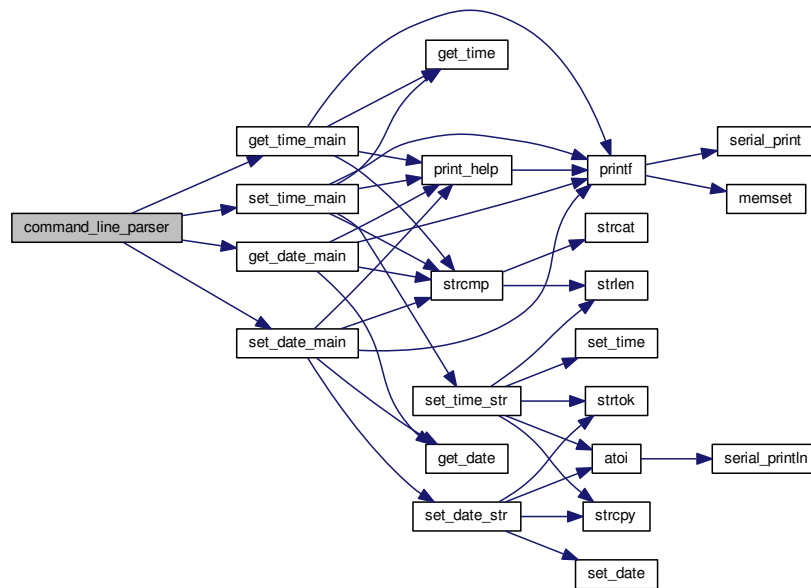
DoubleQuoteWriting

SingleQuoteWriting

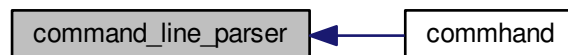
5.6.4 Function Documentation

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

Here is the call graph for this function:

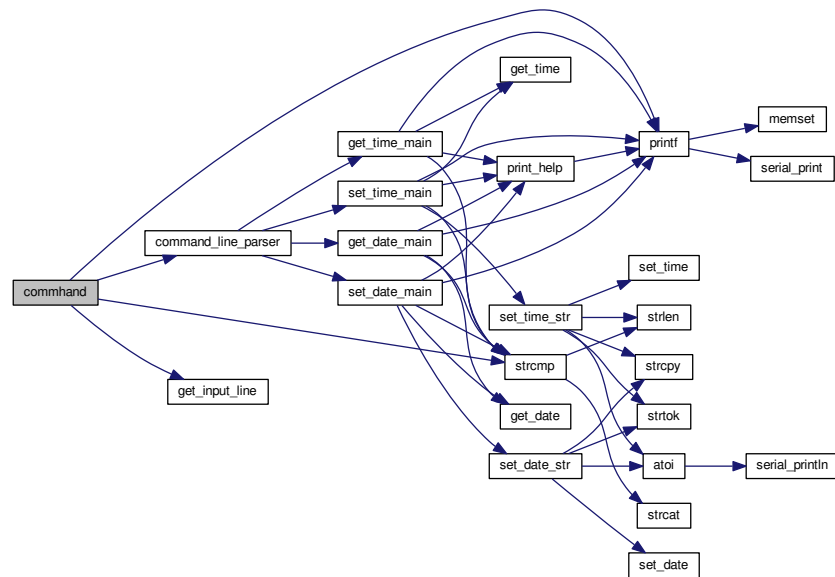


Here is the caller graph for this function:

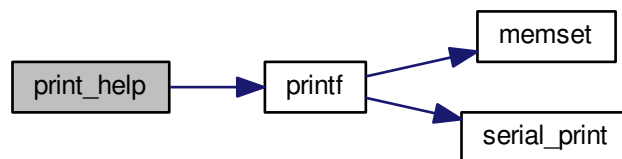


5.6.4.2 int commhand ()

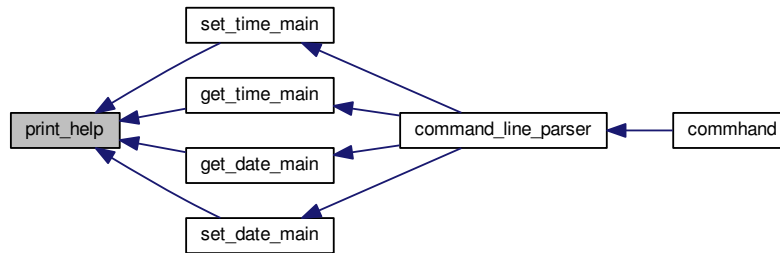
Here is the call graph for this function:

5.6.4.3 void print_help (const int *function_index*)

Here is the call graph for this function:



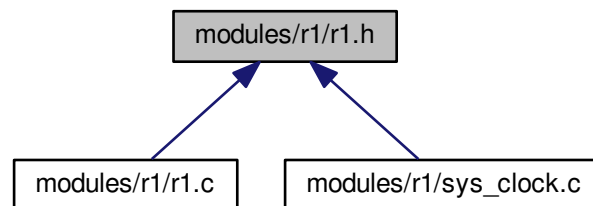
Here is the caller graph for this function:



5.7 modules/r1/r1.h File Reference

The commandhandler and functions associations for Module R1.

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



Macros

- #define [HELP](#) 0
- #define [VERSION](#) 1
- #define [GETTIME](#) 2
- #define [SETTIME](#) 3
- #define [GETDATE](#) 4
- #define [SETDATE](#) 5
- #define [SHUTDOWN](#) 6
- #define [NUM_OF_FUNCTIONS](#) 7

Functions

commhand

Accepts and handles commands from the user.

Returns

0

- int [commhand](#) ()

command_line_parser

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

Parameters

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

Returns

void

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

print_help

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

Parameters

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

Returns

void

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

5.7.1 Detailed Description

The commandhandler and functions associations for Module R1.

Author

Thunder Krakens

Date

February 2nd, 2016

Version

R1

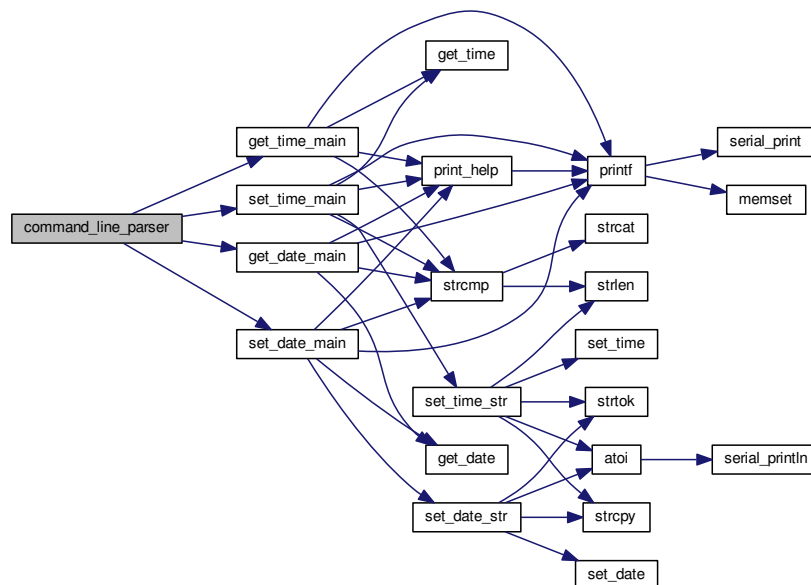
5.7.2 Macro Definition Documentation

5.7.2.1 `#define GETDATE 4`5.7.2.2 `#define GETTIME 2`5.7.2.3 `#define HELP 0`5.7.2.4 `#define NUM_OF_FUNCTIONS 7`5.7.2.5 `#define SETDATE 5`5.7.2.6 `#define SETTIME 3`5.7.2.7 `#define SHUTDOWN 6`5.7.2.8 `#define VERSION 1`

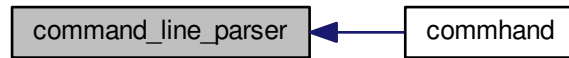
5.7.3 Function Documentation

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

Here is the call graph for this function:

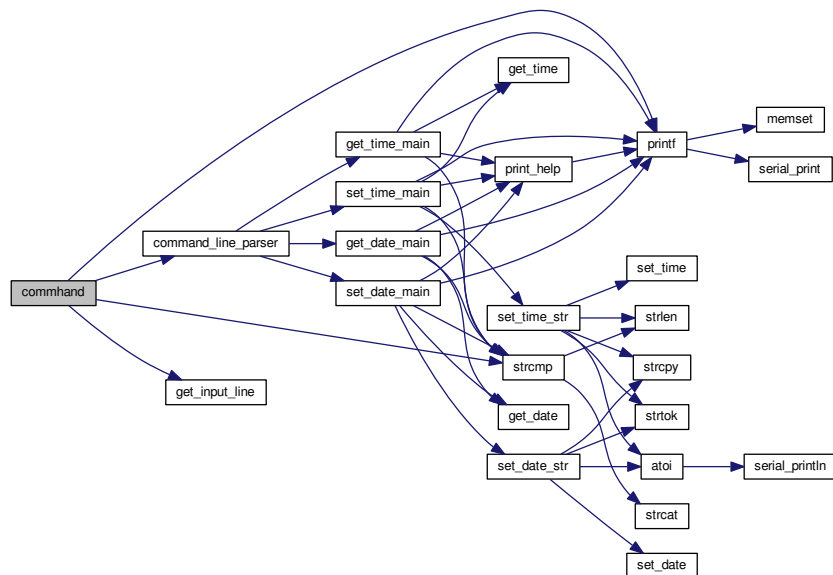


Here is the caller graph for this function:



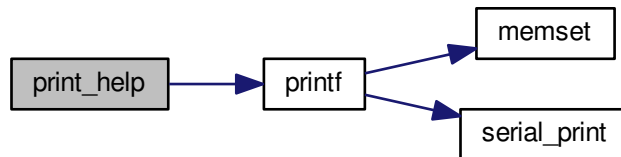
5.7.3.2 int commhand ()

Here is the call graph for this function:

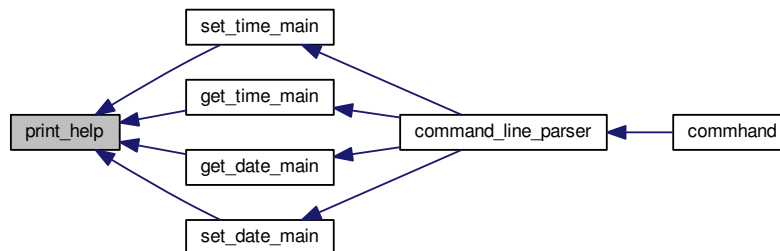


5.7.3.3 void print_help (const int *function_index*)

Here is the call graph for this function:



Here is the caller graph for this function:



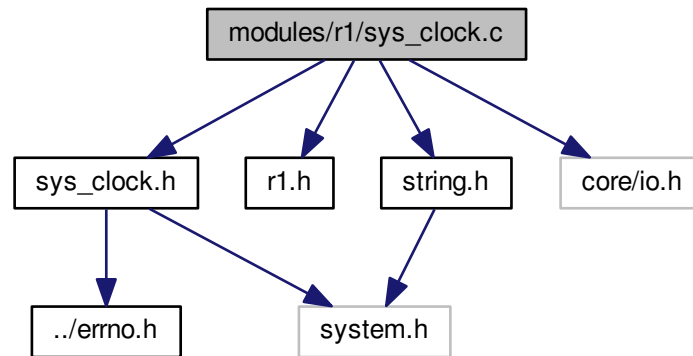
5.8 modules/r1/sys_clock.c File Reference

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

```

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


Include dependency graph for sys_clock.c:



Macros

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

Functions

set_time_main.

Sets the time for the system.

Parameters

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

Returns

0

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

get_time_main.

Retrieves system's current time.

Parameters

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

Returns

0

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

is_digit

determines if a character represents a digit.

Parameters

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

Returns

1 if it is digit, otherwise returns 0.

set_time_str.

Sets the time for the system by string.

Parameters

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

Returns

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

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

get_time.

Retrieves system's current time and date.

Parameters

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

Returns

VOID

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

set_time.

Sets the time for the system by date_time struct.

Parameters

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

Returns

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

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

get_date.

Retrieves system's current date.

Parameters

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

Returns

VOID

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

is_date_value_valid.

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

Parameters

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

Returns

VOID

set_date.

Sets the date of the system.

Parameters

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

Returns

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

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

get_date_main.

Retrieves system's current date.

Parameters

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

Returns

0

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

set_date_str.

Sets the date for the system by string.

Parameters

str	The string type of current date.
-----	----------------------------------

Returns

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

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

set_date_main.

Sets system's date.

Parameters

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

Returns

0

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

5.8.1 Detailed Description

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

Author

Thunder Krakens

Date

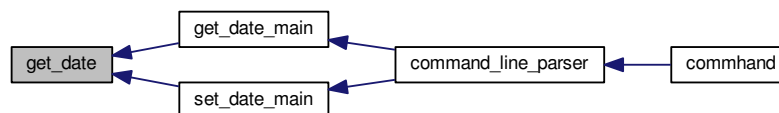
February 2nd, 2016

Version

R1

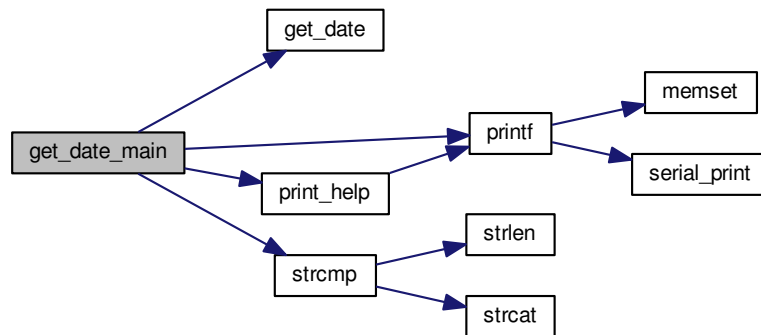
5.8.2 Macro Definition Documentation**5.8.2.1** `#define RTC_INDEX_DAY_MONTH 0x07`**5.8.2.2** `#define RTC_INDEX_DAY_WEEK 0x06`**5.8.2.3** `#define RTC_INDEX_HOUR 0x04`**5.8.2.4** `#define RTC_INDEX_HOUR_ALARM 0x05`**5.8.2.5** `#define RTC_INDEX_MINUTE 0x02`**5.8.2.6** `#define RTC_INDEX_MINUTE_ALARM 0x03`**5.8.2.7** `#define RTC_INDEX_MONTH 0x08`**5.8.2.8** `#define RTC_INDEX_SECOND 0x00`**5.8.2.9** `#define RTC_INDEX_SECOND_ALARM 0x01`**5.8.2.10** `#define RTC_INDEX_YEAR 0x09`**5.8.3 Function Documentation****5.8.3.1** `void get_date (date_time * dateTimeValues)`

Here is the caller graph for this function:

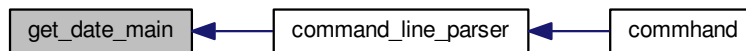


5.8.3.2 `int get_date_main (int argc, char ** argv)`

Here is the call graph for this function:

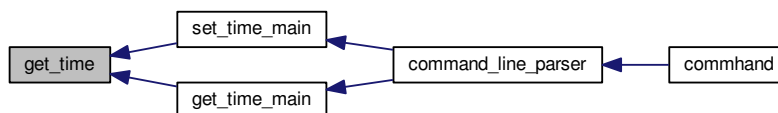


Here is the caller graph for this function:



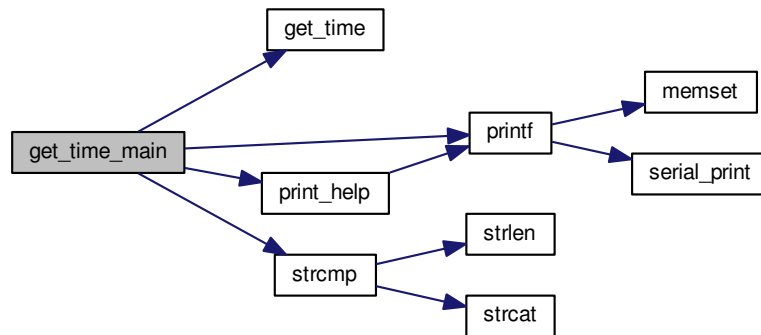
5.8.3.3 `void get_time (date_time * dateTimeValues)`

Here is the caller graph for this function:

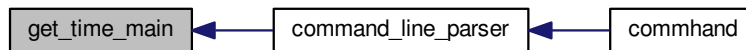


5.8.3.4 int get_time_main (int argc, char ** argv)

Here is the call graph for this function:

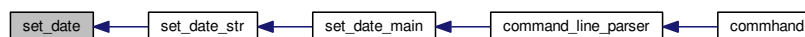


Here is the caller graph for this function:



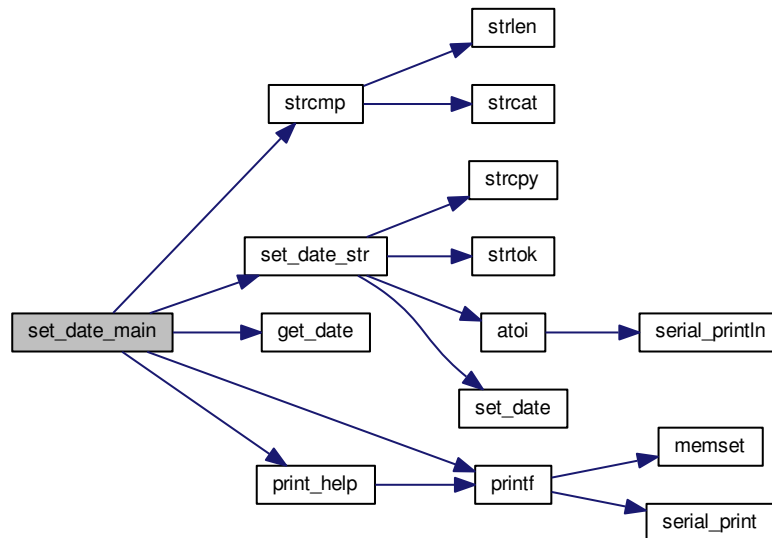
5.8.3.5 error_t set_date (const date_time * dateTimeValues)

Here is the caller graph for this function:

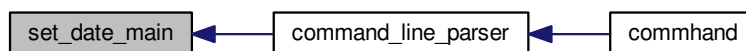


5.8.3.6 int set_date_main (int argc, char ** argv)

Here is the call graph for this function:

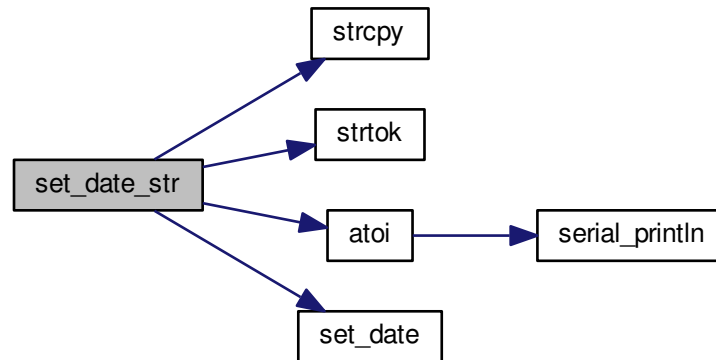


Here is the caller graph for this function:



5.8.3.7 int set_date_str (const char * str)

Here is the call graph for this function:

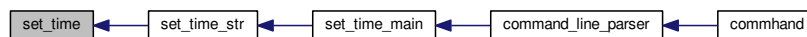


Here is the caller graph for this function:



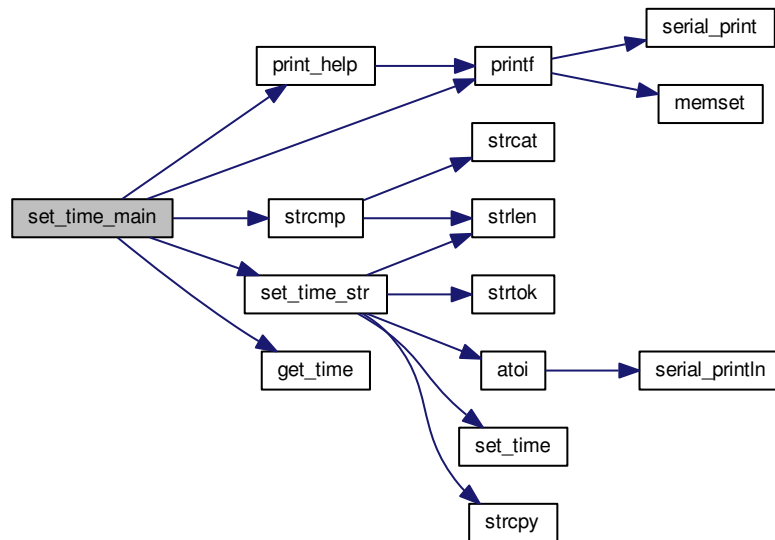
5.8.3.8 error_t set_time (const date_time * dateTimeValues)

Here is the caller graph for this function:

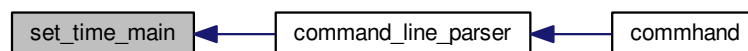


5.8.3.9 int set_time_main (int argc, char ** argv)

Here is the call graph for this function:

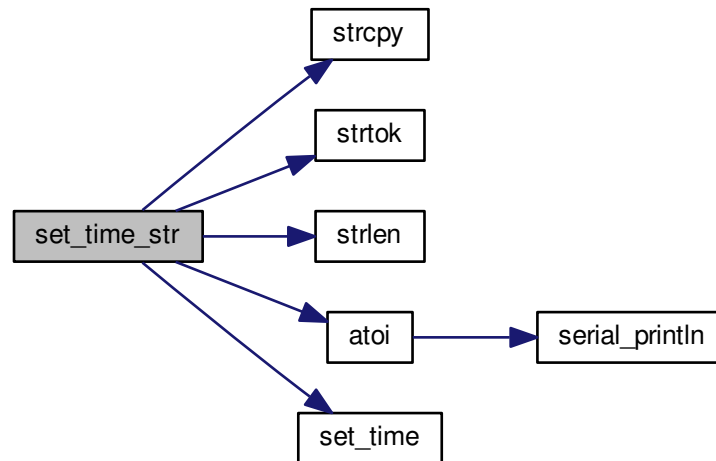


Here is the caller graph for this function:



5.8.3.10 error_t set_time_str (const char * timeStr)

Here is the call graph for this function:



Here is the caller graph for this function:

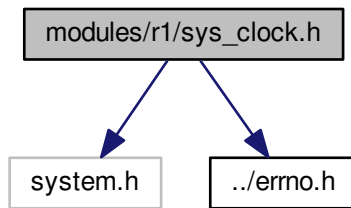


5.9 modules/r1/sys_clock.h File Reference

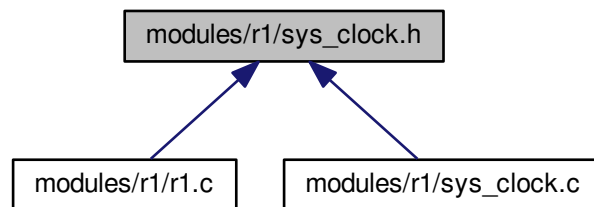
The main file that manipulates and controls the system's clock.

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

Include dependency graph for sys_clock.h:



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



Functions

set_time_main.

Sets the time for the system.

Parameters

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

Returns

0

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

get_time_main.

Retrieves system's current time.

Parameters

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

Returns

0

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

set_time_str.

Sets the time for the system by string.

Parameters

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

Returns

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

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

get_time.

Retrieves system's current time and date.

Parameters

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

Returns

VOID

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

set_time.

Sets the time for the system by date_time struct.

Parameters

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

Returns

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

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

set_date_main.

Sets system's date.

Parameters

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

Returns

0

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

get_date_main.

Retrieves system's current date.

Parameters

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

Returns

0

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

get_date.

Retrieves system's current date.

Parameters

dateTimeValues	The struct that holds the value of current date
----------------	---

Returns

VOID

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

set_date_str.

Sets the date for the system by string.

Parameters

str	The string type of current date.
-----	----------------------------------

Returns

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

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

set_date.

Sets the date of the system.

Parameters

dateTimeValues	The struct that holds the value of date
----------------	---

Returns

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

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

5.9.1 Detailed Description

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

Author

Thunder Krakens

Date

February 2nd, 2016

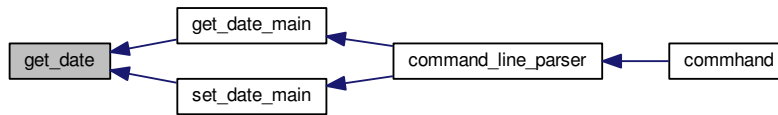
Version

R1

5.9.2 Function Documentation

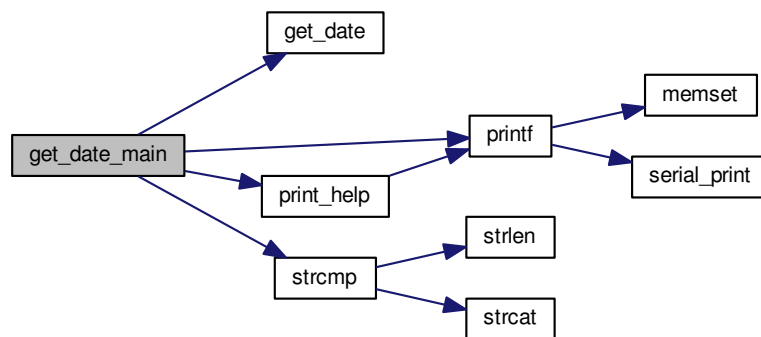
5.9.2.1 void get_date (date_time * dateTimeValues)

Here is the caller graph for this function:

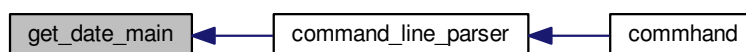


5.9.2.2 int get_date_main (int argc, char ** argv)

Here is the call graph for this function:

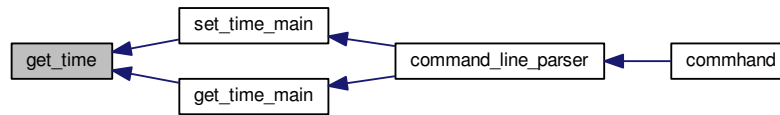


Here is the caller graph for this function:



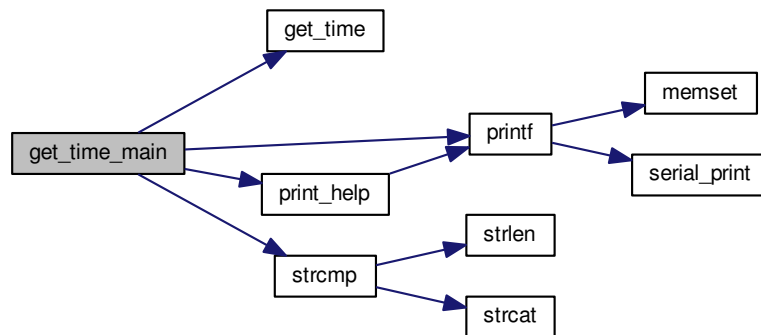
5.9.2.3 void get_time (date_time * dateTimeValues)

Here is the caller graph for this function:

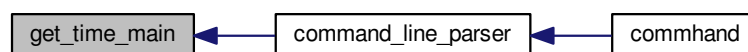


5.9.2.4 int get_time_main (int argc, char ** argv)

Here is the call graph for this function:

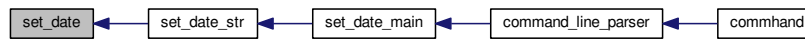


Here is the caller graph for this function:



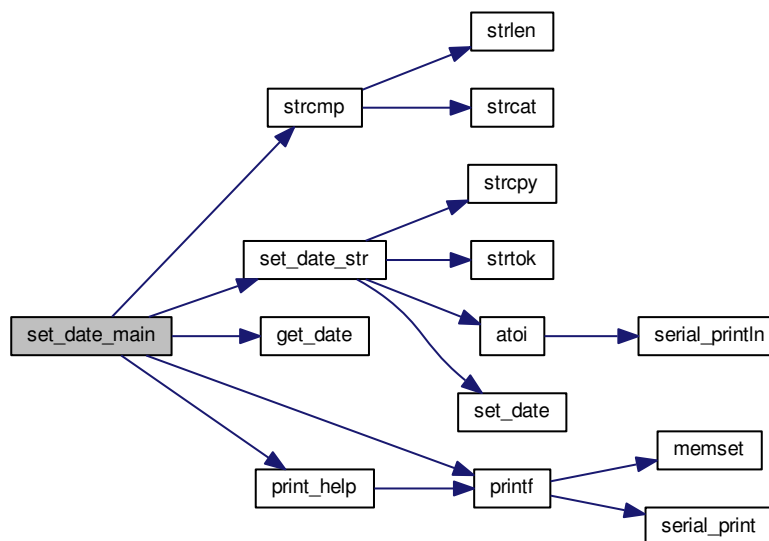
5.9.2.5 `error_t set_date (const date_time * dateTimeValues)`

Here is the caller graph for this function:

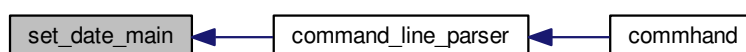


5.9.2.6 `int set_date_main (int argc, char ** argv)`

Here is the call graph for this function:

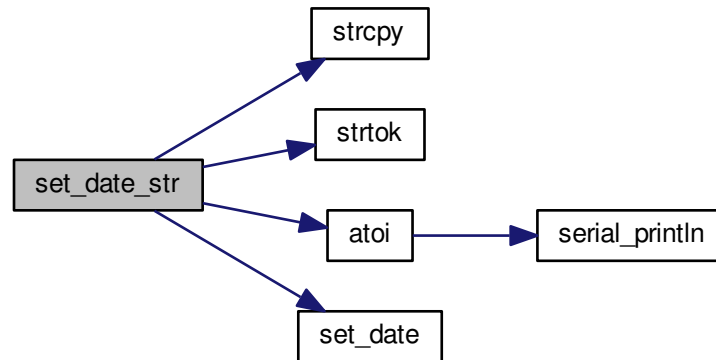


Here is the caller graph for this function:



5.9.2.7 int set_date_str (const char * str)

Here is the call graph for this function:

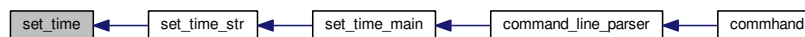


Here is the caller graph for this function:



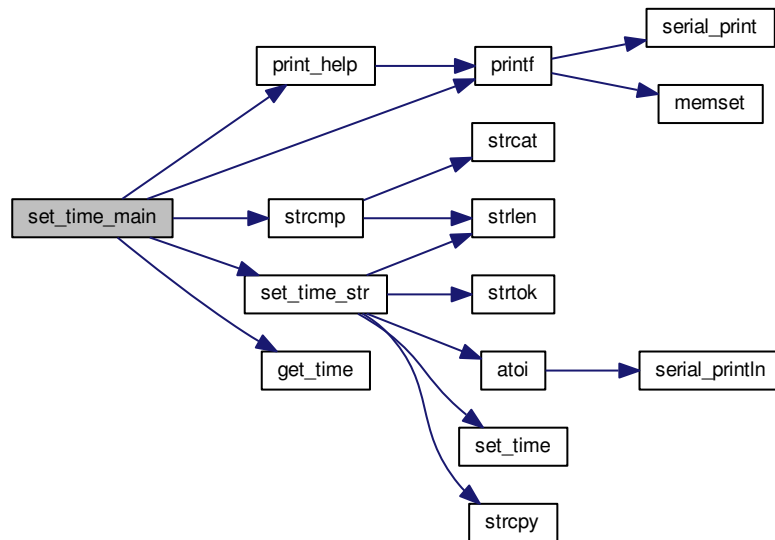
5.9.2.8 error_t set_time (const date_time * dateTimeValues)

Here is the caller graph for this function:

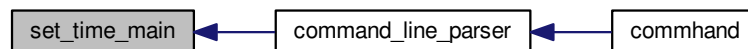


5.9.2.9 int set_time_main (int argc, char ** argv)

Here is the call graph for this function:

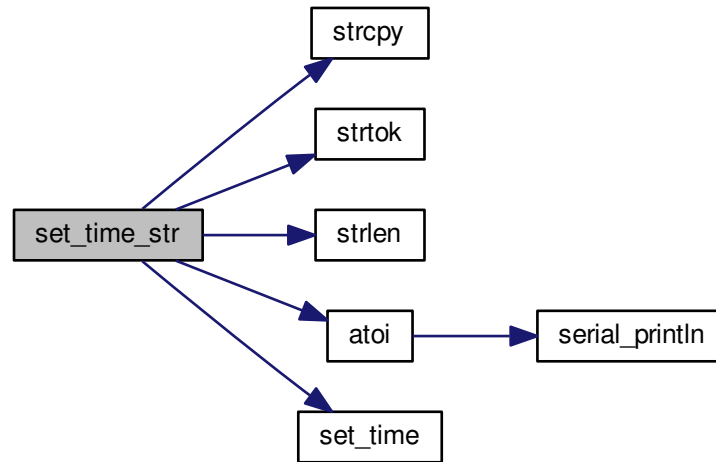


Here is the caller graph for this function:



5.9.2.10 error_t set_time_str (const char * timeStr)

Here is the call graph for this function:



Here is the caller graph for this function:



Index

atoi
 string.c, [24](#)
 string.h, [16](#)

COM1
 serial.h, [10](#)

COM2
 serial.h, [10](#)

COM3
 serial.h, [10](#)

COM4
 serial.h, [11](#)

COMPLETION
 r1.c, [33](#)

command_line_parser
 r1.c, [34](#)
 r1.h, [38](#)

CommandPaserStat
 r1.c, [33](#)

commhand
 r1.c, [34](#)
 r1.h, [39](#)

documentation/mainpage.dox, [9](#)

DoubleQuoteWriting
 r1.c, [33](#)

E_INVPARA
 errno.h, [29](#)

E_INVSTRF
 errno.h, [29](#)

E_INVUSRI
 errno.h, [29](#)

E_NOERROR
 errno.h, [29](#)

errno.h
 E_INVPARA, [29](#)
 E_INVSTRF, [29](#)
 E_INVUSRI, [29](#)
 E_NOERROR, [29](#)
 error_t, [29](#)

error_t
 errno.h, [29](#)

function
 function_name, [7](#)

function_name, [7](#)
 function, [7](#)
 help, [7](#)
 nameStr, [7](#)
 usage, [8](#)

GETDATE
 r1.h, [38](#)

GETTIME
 r1.h, [38](#)

get_date
 sys_clock.c, [45](#)
 sys_clock.h, [55](#)

get_date_main
 sys_clock.c, [45](#)
 sys_clock.h, [56](#)

get_input_line
 serial.h, [11](#)

get_time
 sys_clock.c, [46](#)
 sys_clock.h, [56](#)

get_time_main
 sys_clock.c, [46](#)
 sys_clock.h, [57](#)

HELP
 r1.h, [38](#)

help
 function_name, [7](#)

include/core/serial.h, [9](#)

include/string.h, [12](#)

init_serial
 serial.h, [11](#)

isspace
 string.c, [24](#)
 string.h, [16](#)

lib/string.c, [20](#)

MAX_ARGC
 r1.c, [33](#)

MOD_VERSION
 r1.c, [33](#)

memset
 string.c, [25](#)

- string.h, 17
- modules/errno.h, 28
- modules/r1/r1.c, 29
- modules/r1/r1.h, 36
- modules/r1/sys_clock.c, 40
- modules/r1/sys_clock.h, 51
- NUM_OF_FUNCTIONS
 - r1.h, 38
- nameStr
 - function_name, 7
- NormalWriting
 - r1.c, 33
- NotWriting
 - r1.c, 33
- print_help
 - r1.c, 35
 - r1.h, 39
- printf
 - string.c, 25
 - string.h, 17
- r1.c
 - COMPLETION, 33
 - command_line_parser, 34
 - CommandPaserStat, 33
 - commhand, 34
 - DoubleQuoteWriting, 33
 - MAX_ARGC, 33
 - MOD_VERSION, 33
 - NormalWriting, 33
 - NotWriting, 33
 - print_help, 35
 - SingleQuoteWriting, 33
 - USER_INPUT_BUFFER_SIZE, 33
- r1.h
 - command_line_parser, 38
 - commhand, 39
 - GETDATE, 38
 - GETTIME, 38
 - HELP, 38
 - NUM_OF_FUNCTIONS, 38
 - print_help, 39
 - SETDATE, 38
 - SETTIME, 38
 - SHUTDOWN, 38
 - VERSION, 38
- RTC_INDEX_DAY_MONTH
 - sys_clock.c, 45
- RTC_INDEX_DAY_WEEK
 - sys_clock.c, 45
- RTC_INDEX_HOUR_ALARM
 - sys_clock.c, 45
- RTC_INDEX_HOUR
 - sys_clock.c, 45
- RTC_INDEX_MINUTE_ALARM
 - sys_clock.c, 45
- RTC_INDEX_MINUTE
 - sys_clock.c, 45
- RTC_INDEX_MONTH
 - sys_clock.c, 45
- RTC_INDEX_SECOND_ALARM
 - sys_clock.c, 45
- RTC_INDEX_SECOND
 - sys_clock.c, 45
- RTC_INDEX_YEAR
 - sys_clock.c, 45
- SETDATE
 - r1.h, 38
- SETTIME
 - r1.h, 38
- SHUTDOWN
 - r1.h, 38
- serial.h
 - COM1, 10
 - COM2, 10
 - COM3, 10
 - COM4, 11
 - get_input_line, 11
 - init_serial, 11
 - serial_print, 11
 - serial_println, 11
 - set_serial_in, 12
 - set_serial_out, 12
 - WithEcho, 11
 - WithoutEcho, 11
- serial_print
 - serial.h, 11
- serial_println
 - serial.h, 11
- set_date
 - sys_clock.c, 47
 - sys_clock.h, 57
- set_date_main
 - sys_clock.c, 47
 - sys_clock.h, 58
- set_date_str
 - sys_clock.c, 48
 - sys_clock.h, 58
- set_serial_in
 - serial.h, 12
- set_serial_out
 - serial.h, 12
- set_time
 - sys_clock.c, 49
 - sys_clock.h, 59
- set_time_main
 - sys_clock.c, 49
 - sys_clock.h, 59

- sys_clock.c, [49](#)
- sys_clock.h, [59](#)
- set_time_str
 - sys_clock.c, [50](#)
 - sys_clock.h, [60](#)
- SingleQuoteWriting
 - r1.c, [33](#)
- sprintf
 - string.c, [25](#)
 - string.h, [17](#)
- strcat
 - string.c, [26](#)
 - string.h, [18](#)
- strcmp
 - string.c, [26](#)
 - string.h, [18](#)
- strcpy
 - string.c, [27](#)
 - string.h, [19](#)
- string.c
 - atoi, [24](#)
 - isspace, [24](#)
 - memset, [25](#)
 - printf, [25](#)
 - sprintf, [25](#)
 - strcat, [26](#)
 - strcmp, [26](#)
 - strcpy, [27](#)
 - strlen, [27](#)
 - strtok, [27](#)
- string.h
 - atoi, [16](#)
 - isspace, [16](#)
 - memset, [17](#)
 - printf, [17](#)
 - sprintf, [17](#)
 - strcat, [18](#)
 - strcmp, [18](#)
 - strcpy, [19](#)
 - strlen, [19](#)
 - strtok, [19](#)
- strlen
 - string.c, [27](#)
 - string.h, [19](#)
- strtok
 - string.c, [27](#)
 - string.h, [19](#)
- sys_clock.c
 - get_date, [45](#)
 - get_date_main, [45](#)
 - get_time, [46](#)
 - get_time_main, [46](#)
 - RTC_INDEX_DAY_MONTH, [45](#)
 - RTC_INDEX_DAY_WEEK, [45](#)
 - RTC_INDEX_HOUR_ALARM, [45](#)
 - RTC_INDEX_HOUR, [45](#)
 - RTC_INDEX_MINUTE_ALARM, [45](#)
 - RTC_INDEX_MINUTE, [45](#)
 - RTC_INDEX_MONTH, [45](#)
 - RTC_INDEX_SECOND_ALARM, [45](#)
 - RTC_INDEX_SECOND, [45](#)
 - RTC_INDEX_YEAR, [45](#)
 - set_date, [47](#)
 - set_date_main, [47](#)
 - set_date_str, [48](#)
 - set_time, [49](#)
 - set_time_main, [49](#)
 - set_time_str, [50](#)
- sys_clock.h
 - get_date, [55](#)
 - get_date_main, [56](#)
 - get_time, [56](#)
 - get_time_main, [57](#)
 - set_date, [57](#)
 - set_date_main, [58](#)
 - set_date_str, [58](#)
 - set_time, [59](#)
 - set_time_main, [59](#)
 - set_time_str, [60](#)
- USER_INPUT_BUFFER_SIZE
 - r1.c, [33](#)
- usage
 - function_name, [8](#)
- VERSION
 - r1.h, [38](#)
- WithEcho
 - serial.h, [11](#)
- WithoutEcho
 - serial.h, [11](#)