MPX Thunder Krakens

R1

Generated by Doxygen 1.8.11

Contents

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:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 function_name Struct Reference

A structure to represent each function.

Data Fields

- char ∗ nameStr

    fuction'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

fuction'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

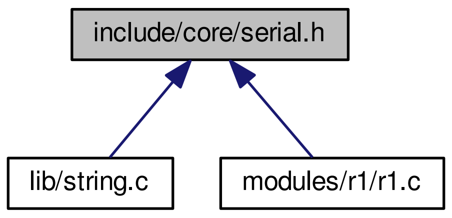
File Documentation

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 keyborad.

Parameters

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

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

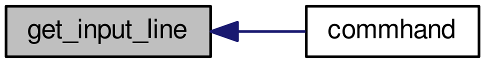
5.2.2.5 #define WithEcho 1

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*)

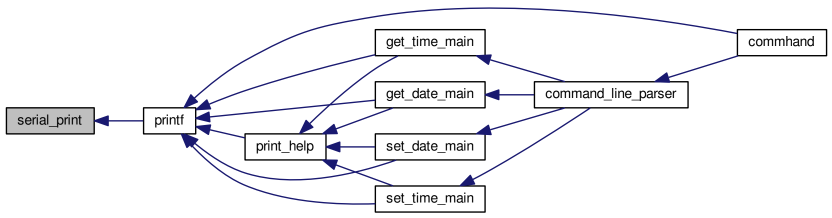
Here is the caller graph for this function:



5.2.3.2 int init_serial (int *device*)

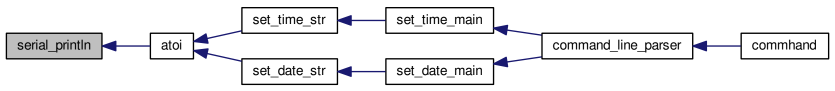
5.2.3.3 int serial_print (const char $*$ *msg*)

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*)

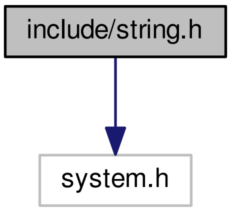
5.2.3.6 int set_serial_out (int *device*)

5.3 include/string.h File Reference

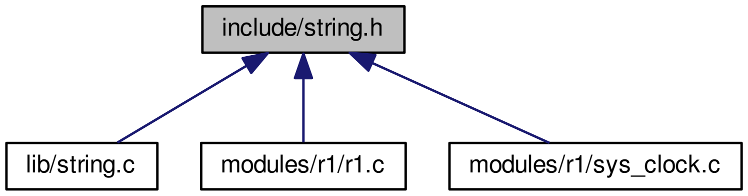
Many usefull functions that used for handling string.

`#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

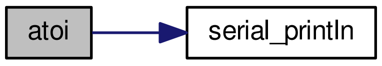
Version

R1

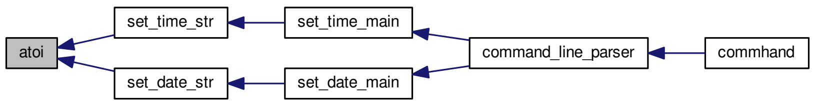
5.3.2 Function Documentation

5.3.2.1 int atoi (const char ∗ *s*)

Here is the call graph for this function:



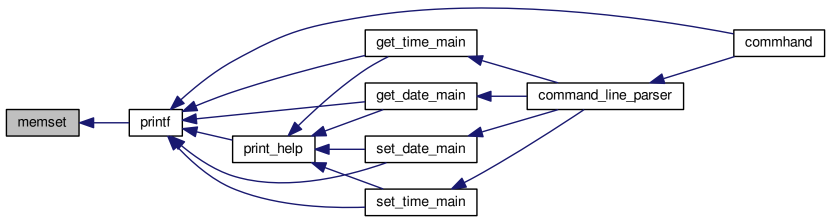
Here is the caller graph for this function:



5.3.2.2 int isspace (const char ∗ *c*)

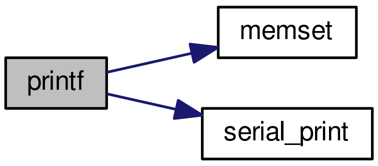
5.3.2.3 void∗ memset (void ∗ *s,* int *c,* size_t *n*)

Here is the caller graph for this function:

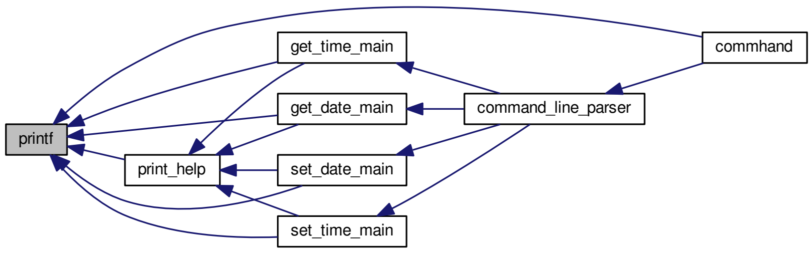


5.3.2.4 int printf (const char ∗ *format,* ...)

Here is the call graph for this function:



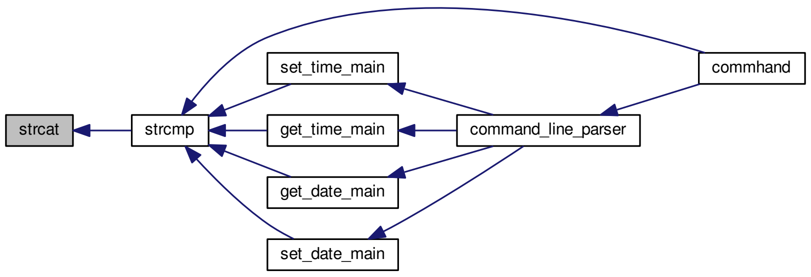
Here is the caller graph for this function:



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

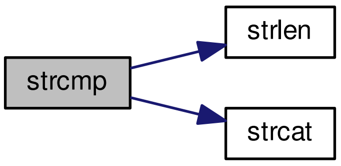
5.3.2.6 char* strcat (char * *s1*, const char * *s2*)

Here is the caller graph for this function:

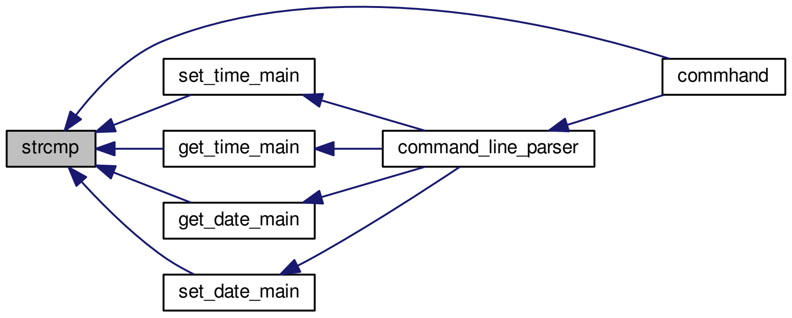


5.3.2.7 int strcmp (const char * *s1*, const char * *s2*)

Here is the call graph for this function:

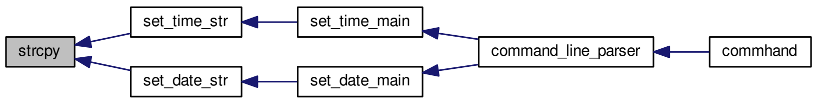


Here is the caller graph for this function:



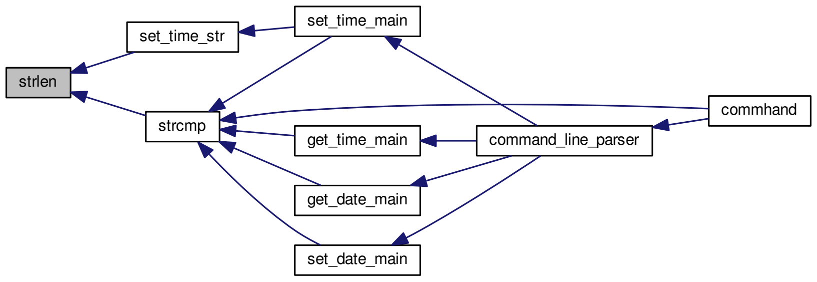
5.3.2.8 char∗ strcpy (char ∗ *s1*, const char ∗ *s2*)

Here is the caller graph for this function:



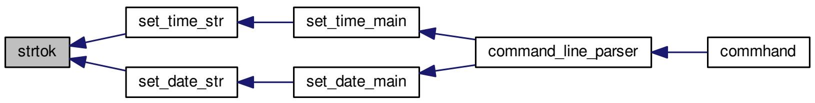
5.3.2.9 int strlen (const char ∗ *s*)

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:

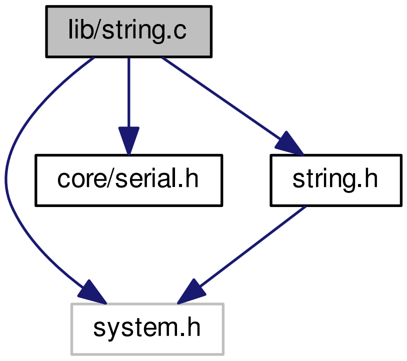


5.4 lib/string.c File Reference

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	*Destination string*
s2	*Source string*

Returns

pointer to the destination String

- char ∗ strcpy (char ∗s1, const char ∗s2)

atoi.

Convert an ASCII string to an integer.

Parameters

s	*String.*

Returns

The converted integer.

- int atoi (const char ∗s)

strcmp.

String comparison.

Parameters

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

Returns

whether they are the same or not.

- int strcmp (const char ∗s1, const char ∗s2)

ParsePadding.

Parse the number for padding.

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

Parameters

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

Returns

bIsValid Returns the validity.

AddPad.

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

Parameters

str	*In string.*
count	*Number of whitespace.*

Returns

VOID

NibbleToChar

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

Parameters

value	*The value of the nibble*

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	*Output string.*
Value	*The value of bytes.*

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

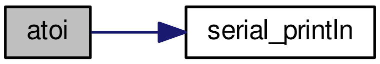
Version

R1

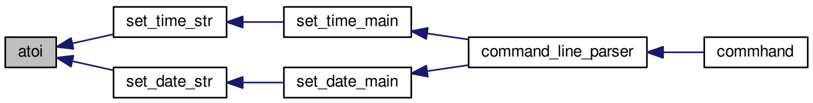
5.4.2 Function Documentation

5.4.2.1 int atoi (const char ∗ *s*)

Here is the call graph for this function:



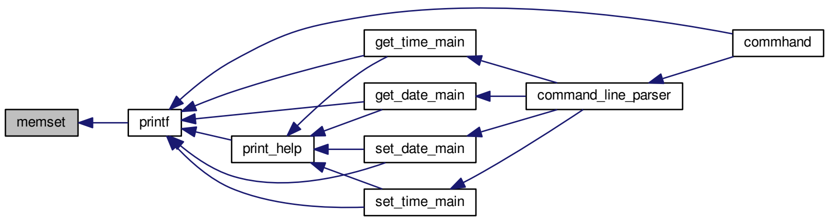
Here is the caller graph for this function:



5.4.2.2 int isspace (const char ∗ *c*)

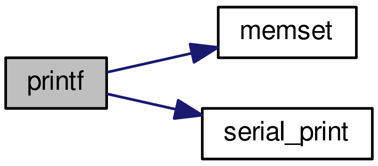
5.4.2.3 void∗ memset (void ∗ *s*, int *c*, size_t *n*)

Here is the caller graph for this function:

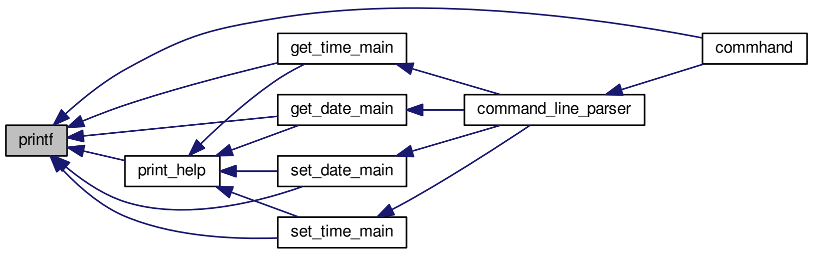


5.4.2.4 int printf (const char ∗ *format*, ...)

Here is the call graph for this function:



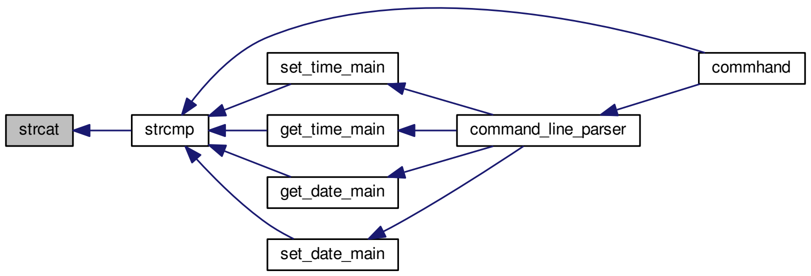
Here is the caller graph for this function:



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

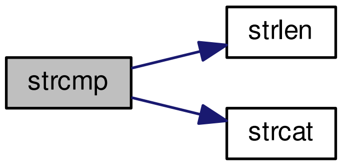
5.4.2.6 char* strcat (char * *s1*, const char * *s2*)

Here is the caller graph for this function:

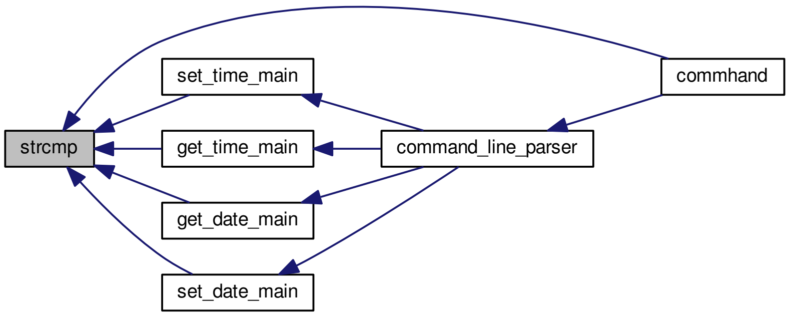


5.4.2.7 int strcmp (const char * *s1*, const char * *s2*)

Here is the call graph for this function:

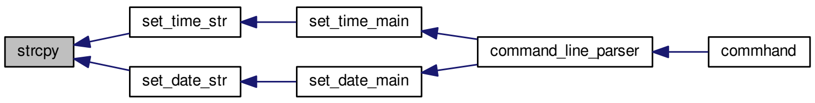


Here is the caller graph for this function:



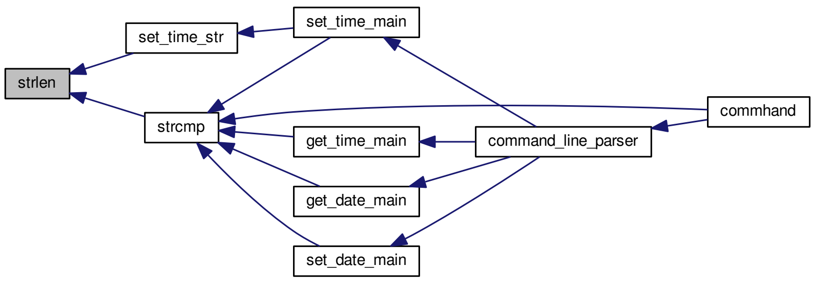
5.4.2.8 char∗ strcpy (char ∗ *s1*, const char ∗ *s2*)

Here is the caller graph for this function:



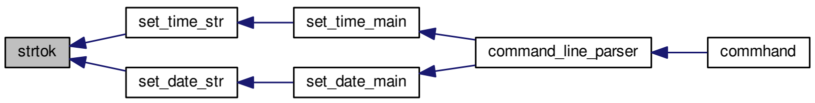
5.4.2.9 int strlen (const char ∗ *s*)

Here is the caller graph for this function:



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

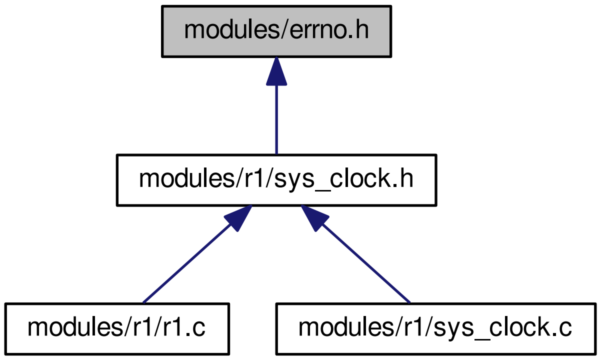
Here is the caller graph for this function:



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 datetype 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

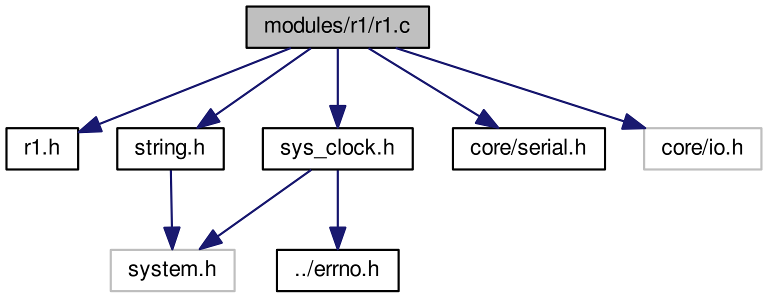
5.5.3.1 typedef unsigned int error_t

5.6 modules/r1/r1.c File Reference

The commandhander 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 fucntion.

Parameters

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

Returns

0

version

displays the version of the system currently running.

Parameters

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

Returns

0

shutdown

Closes all functions, and shuts down the system.

Parameters

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

Returns

0 for shutdown, 1 for keep running.

help_usages

shows usage message for each function.

Parameters

start_from	*the index of the beginning function.*

Returns

0

help_function

displays help text for all functions.

Parameters

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

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	*The complete input command.*
argc	*The number of tokens found.*
argv	*The array of tokens.*
MaxArgNum	*The maximum number of tokens that array can hold.*
MaxStrLen	*The maximum length of each token that string can hold.*

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	*The index number of that function.*

Returns

void

- void print_help (const int function_index)

5.6.1 Detailed Description

The commandhander 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

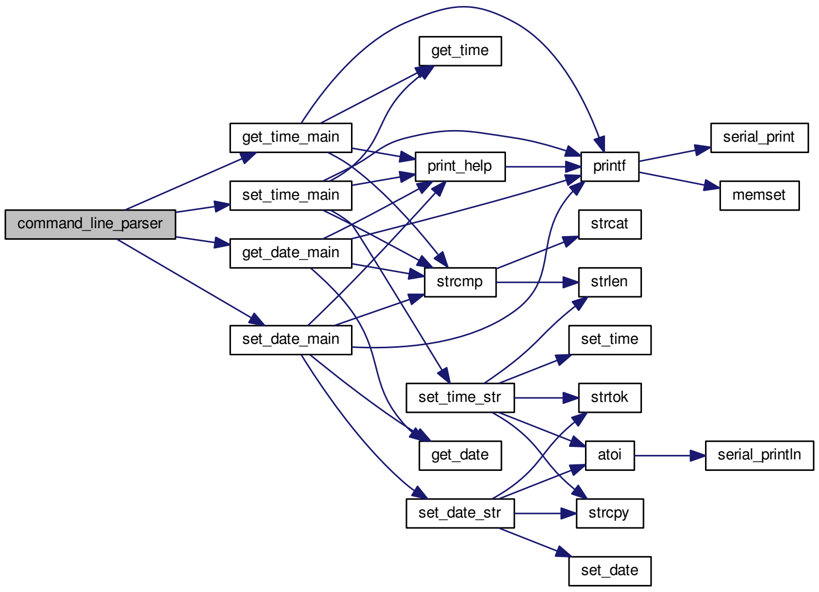
DoubleQuoteWriting

SingleQuoteWriting

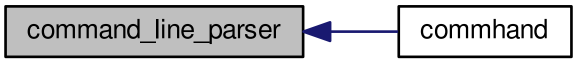
5.6.4 Function Documentation

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:

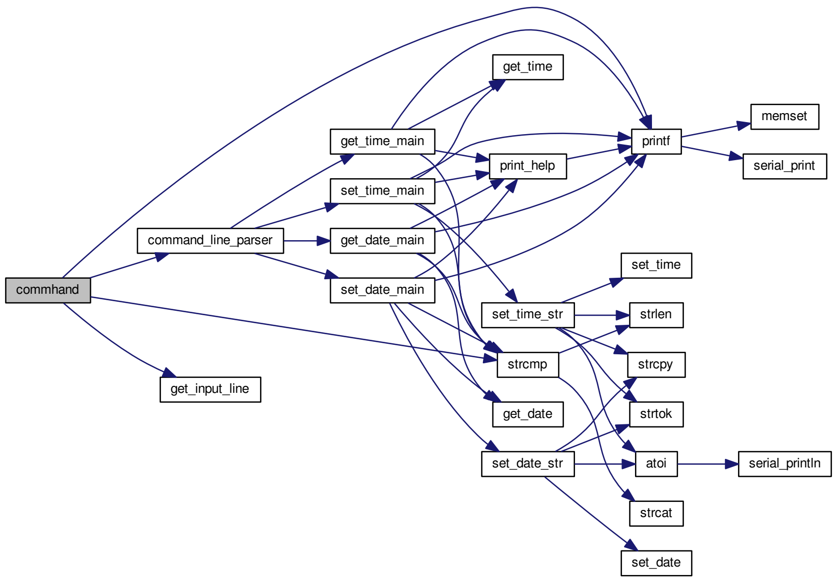


Here is the caller graph for this function:



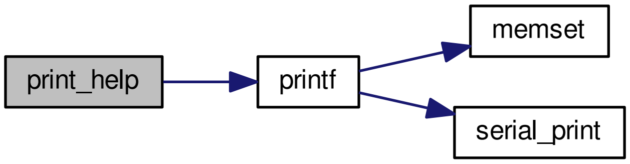
5.6.4.2 int commhand ( )

Here is the call graph for this function:

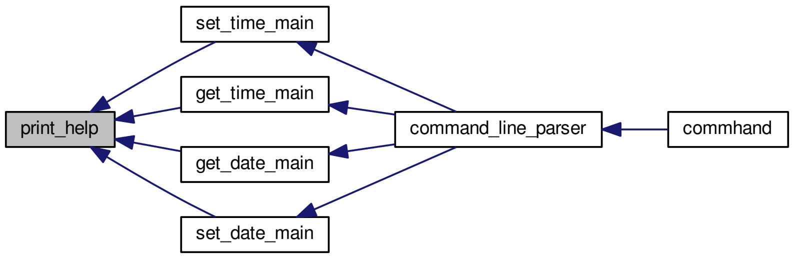


5.6.4.3 void print_help (const int *function_index*)

Here is the call graph for this function:



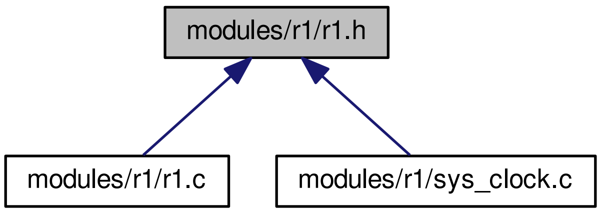
Here is the caller graph for this function:



5.7 modules/r1/r1.h File Reference

The commandhander 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	*The complete input command.*
argc	*The number of tokens found.*
argv	*The array of tokens.*
MaxArgNum	*The maximum number of tokens that array can hold.*
MaxStrLen	*The maximum length of each token that string can hold.*

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	*The index number of that function.*

Returns

void

- void print_help (const int function_index)

5.7.1 Detailed Description

The commandhander 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

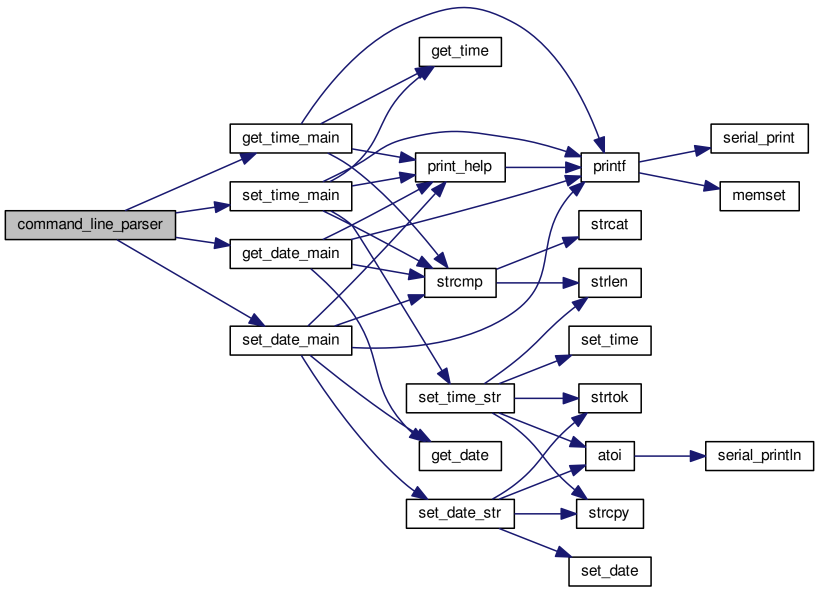
5.7.2.7 #define SHUTDOWN 6

5.7.2.8 #define VERSION 1

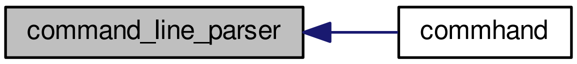
5.7.3 Function Documentation

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:

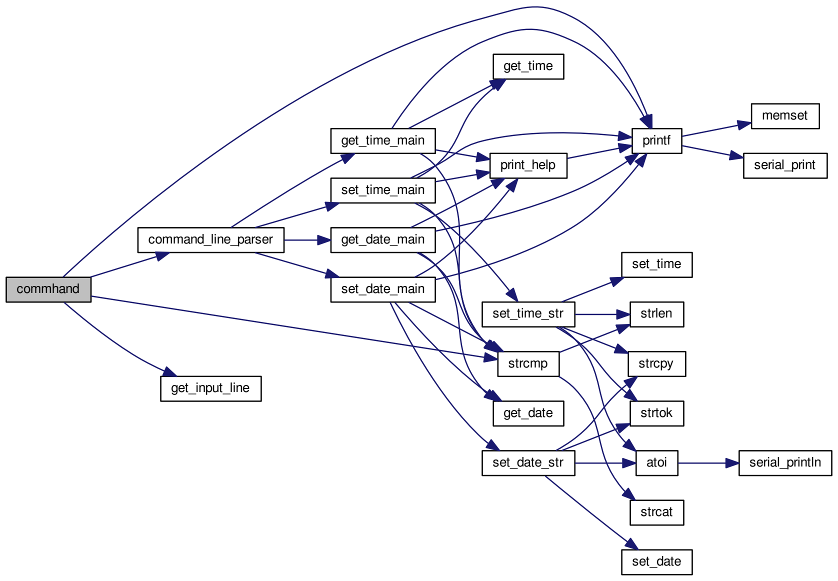


Here is the caller graph for this function:



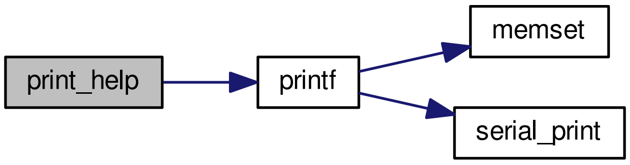
5.7.3.2 int commhand ( )

Here is the call graph for this function:

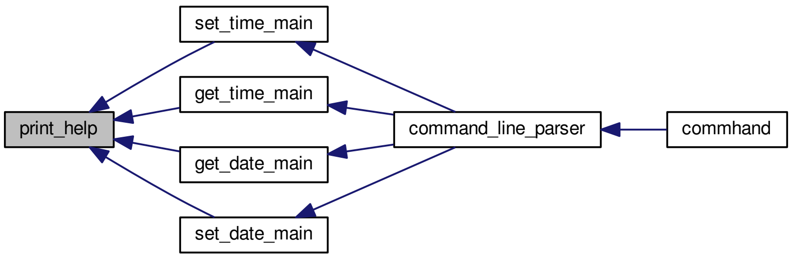


5.7.3.3 void print_help (const int *function_index*)

Here is the call graph for this function:



Here is the caller graph for this function:

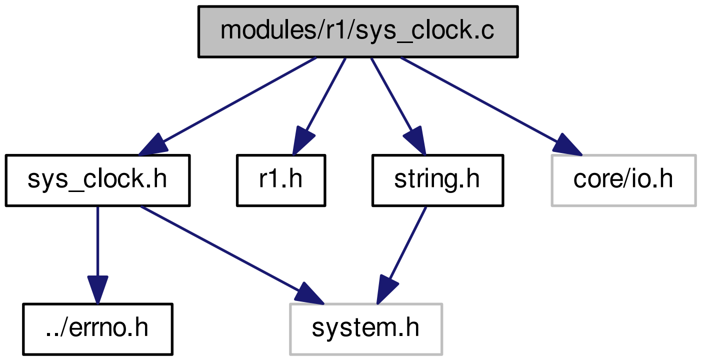


5.8 modules/r1/sys_clock.c File Reference

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

Returns

0

- int get_time_main (int argc, char ∗∗argv)

is_digit

determines if a character represents a digit.

Parameters

ch	*The character*

Returns

1 if it is digit, otherwise returns 0.

set_time_str.

Sets the time for the system by string.

Parameters

timeStr	*The string type of current Time.*

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	*The value of current time and date*

Returns

VOID

- void get_time (date_time ∗dateTimeValues)

set_time.

Sets the time for the system by date_time struct.

Parameters

dateTimeValues	*The struct that holds the time values.*

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	*The struct that holds the value of current date*

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 $\sim$ 1969, month should between 1 $\sim$ 12, while the range of the day is based on the month and year.

Parameters

year	*The value of the year*
mon	*The value of the month*
day	*The value of the day of month*

Returns

VOID

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)

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

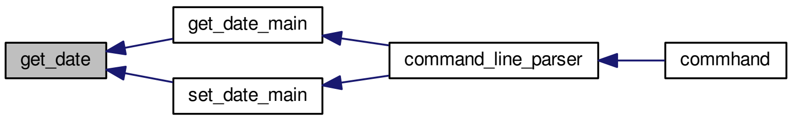
5.8.2.9 #define RTC_INDEX_SECOND_ALARM 0x01

5.8.2.10 #define RTC_INDEX_YEAR 0x09

5.8.3 Function Documentation

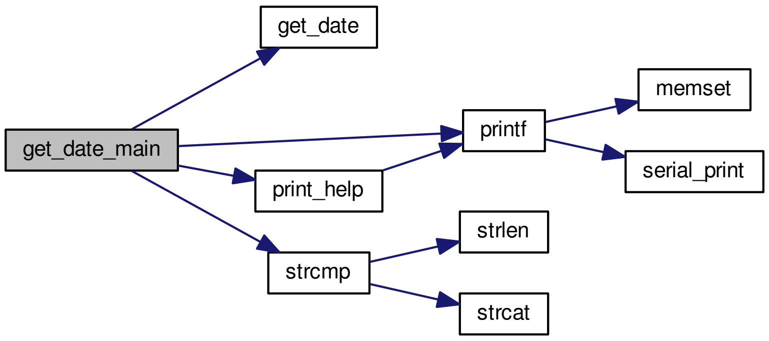
5.8.3.1 void get_date (date_time ∗ *dateTimeValues*)

Here is the caller graph for this function:

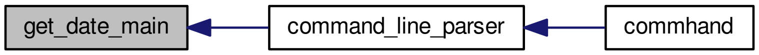


5.8.3.2 int get_date_main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:

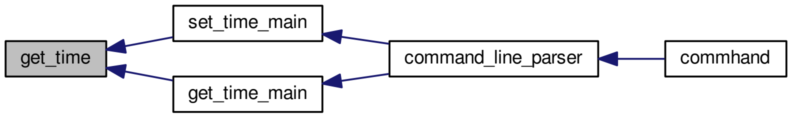


Here is the caller graph for this function:



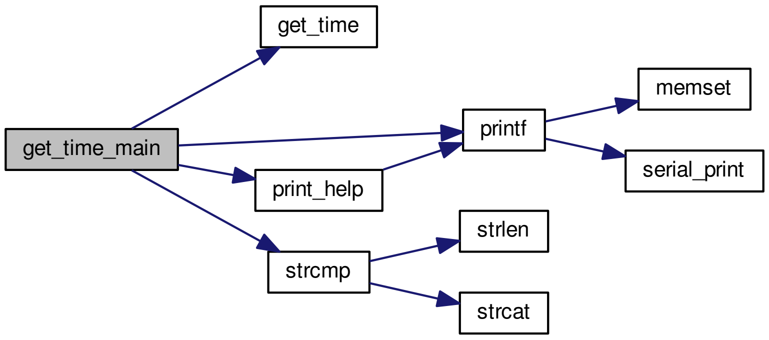
5.8.3.3 void get_time (date_time ∗ *dateTimeValues*)

Here is the caller graph for this function:

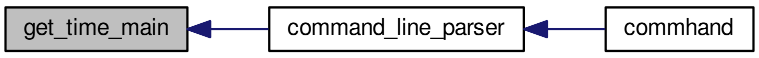


5.8.3.4 int get_time_main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:

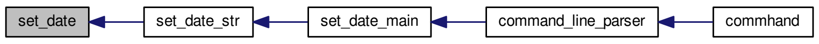


Here is the caller graph for this function:



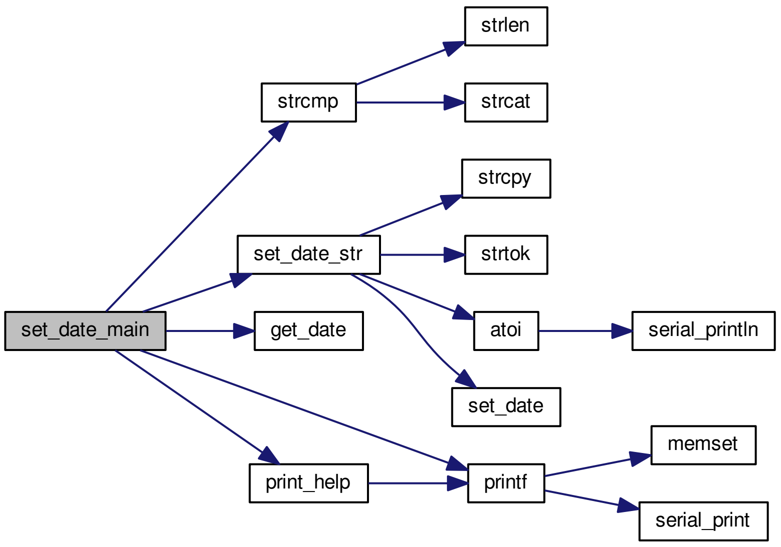
5.8.3.5 error_t set_date (const date_time ∗ *dateTimeValues*)

Here is the caller graph for this function:

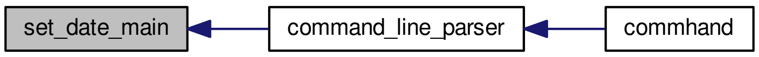


5.8.3.6 int set_date_main (int *argc,* char ∗∗ *argv*)

Here is the call graph for this function:

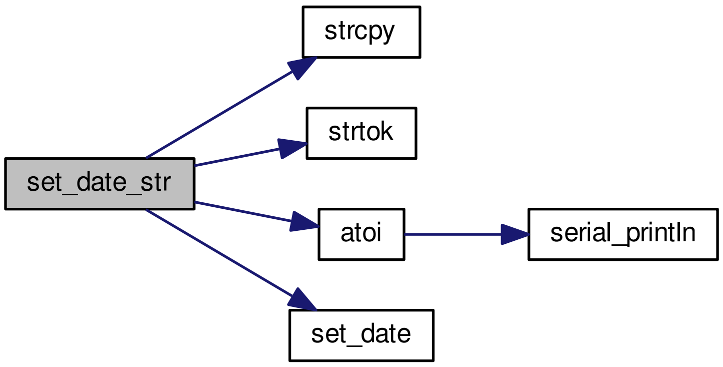


Here is the caller graph for this function:

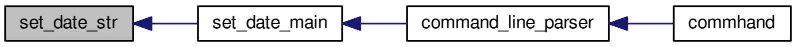


5.8.3.7 int set_date_str (const char ∗ *str*)

Here is the call graph for this function:

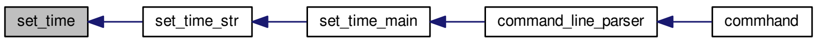


Here is the caller graph for this function:



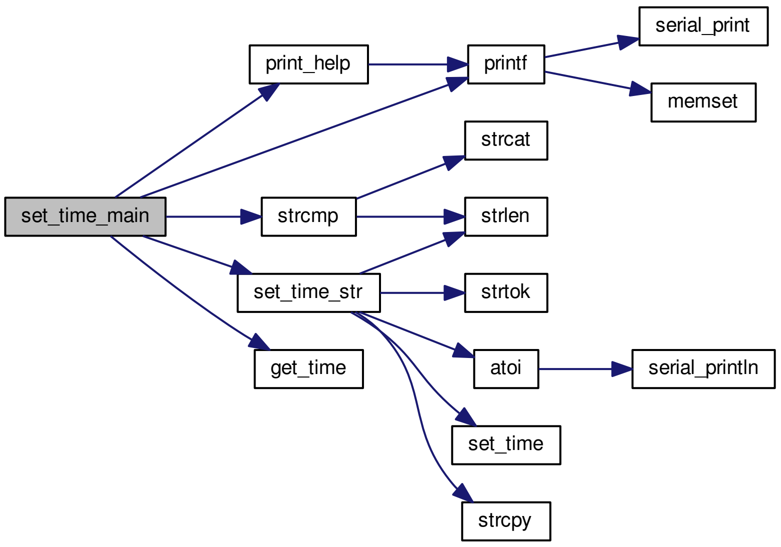
5.8.3.8 error_t set_time (const date_time ∗ *dateTimeValues*)

Here is the caller graph for this function:

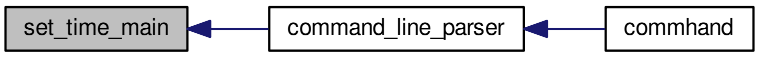


5.8.3.9 int set_time_main (int *argc*, char ** *argv*)

Here is the call graph for this function:

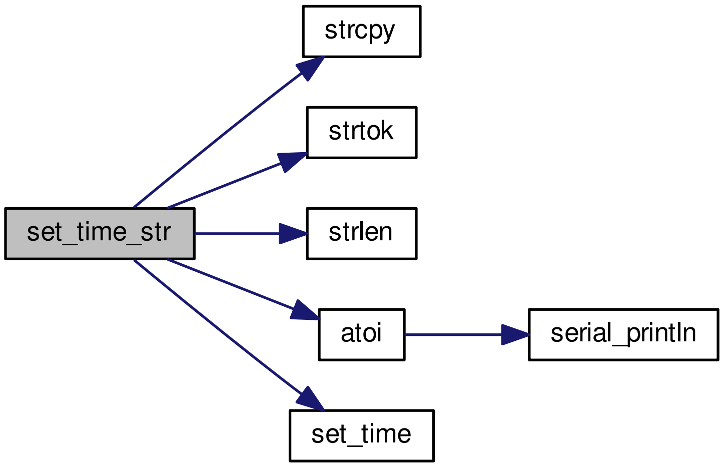


Here is the caller graph for this function:

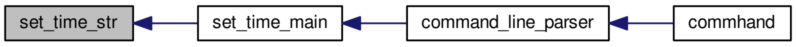


5.8.3.10 error_t set_time_str (const char ∗ *timeStr*)

Here is the call graph for this function:



Here is the caller graph for this function:

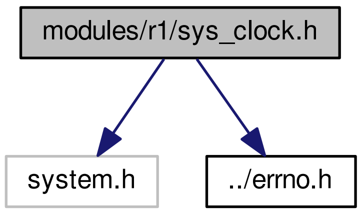


5.9 modules/r1/sys_clock.h File Reference

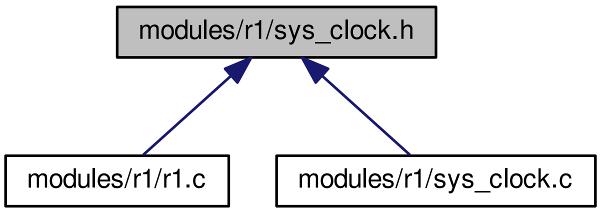
The main file that manipulates and controls the system's clock.

```
#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	*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	*The number of tokens found.*
argv	*The array of tokens.*

Returns

0

- int get_time_main (int argc, char ∗∗argv)

set_time_str.

Sets the time for the system by string.

Parameters

timeStr	*The string type of current Time.*

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	*The value of current time and date*

Returns

VOID

- void get_time (date_time ∗dateTimeValues)

set_time.

Sets the time for the system by date_time struct.

Parameters

dateTimeValues	*The struct that holds the time values.*

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

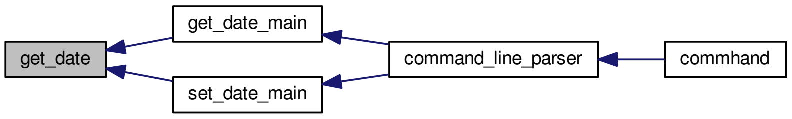
Version

R1

5.9.2 Function Documentation

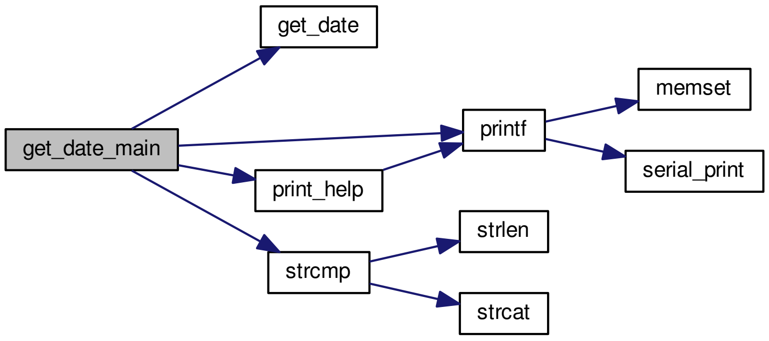
5.9.2.1 void get_date (date_time ∗ *dateTimeValues*)

Here is the caller graph for this function:

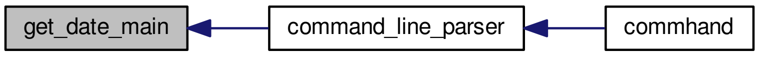


5.9.2.2 int get_date_main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:

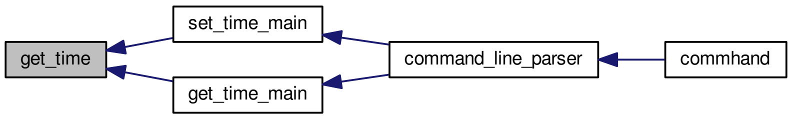


Here is the caller graph for this function:



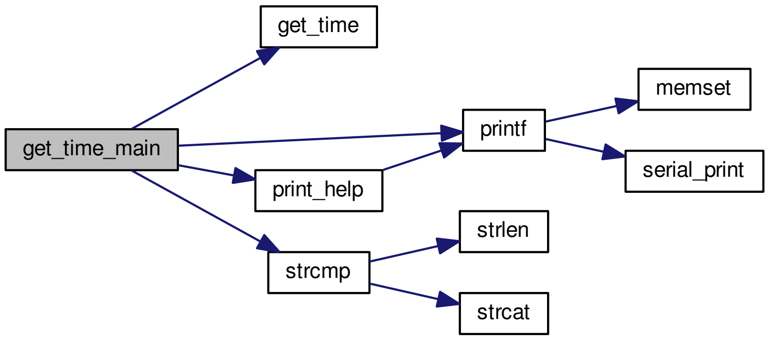
5.9.2.3 void get_time (date_time * *dateTimeValues*)

Here is the caller graph for this function:

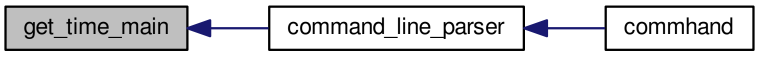


5.9.2.4 int get_time_main (int *argc*, char ** *argv*)

Here is the call graph for this function:

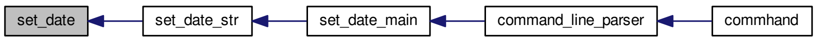


Here is the caller graph for this function:



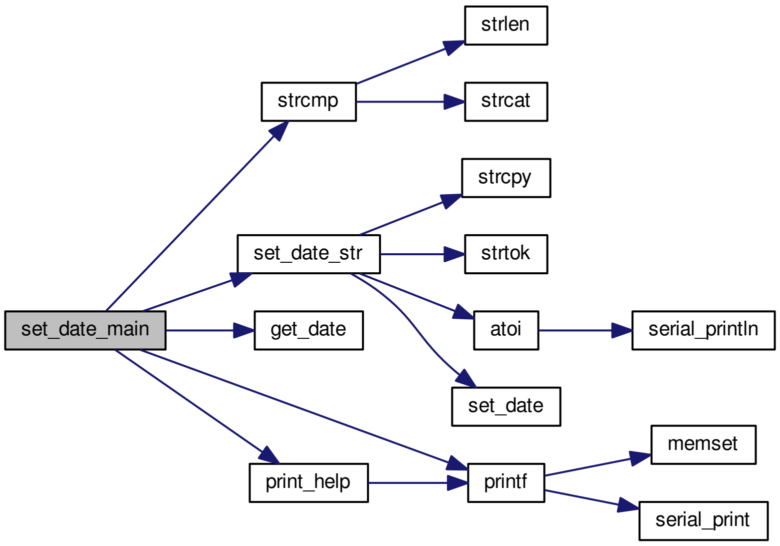
5.9.2.5 error_t set_date (const date_time ∗ *dateTimeValues*)

Here is the caller graph for this function:

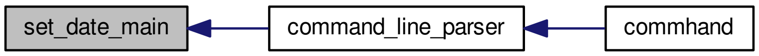


5.9.2.6 int set_date_main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:

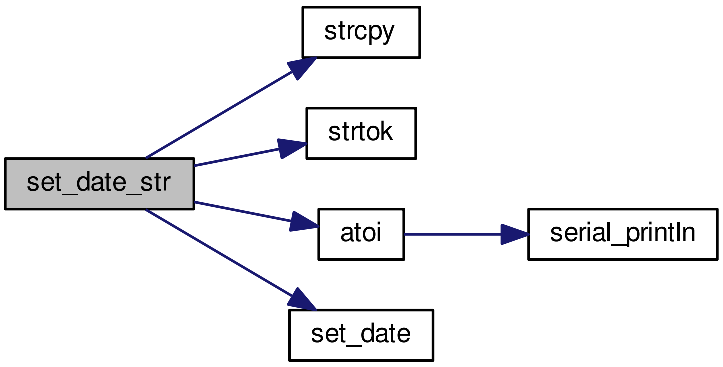


Here is the caller graph for this function:

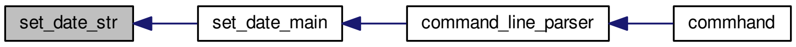


5.9.2.7 int set_date_str (const char ∗ *str*)

Here is the call graph for this function:

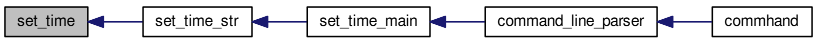


Here is the caller graph for this function:



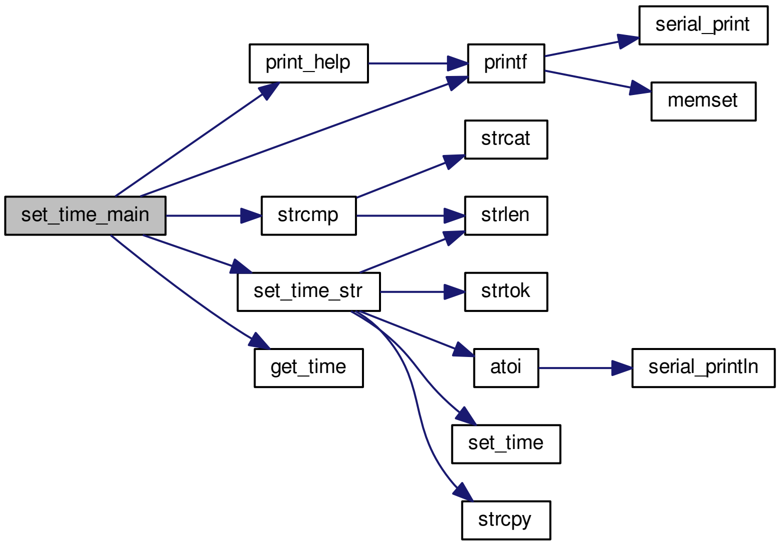
5.9.2.8 error_t set_time (const date_time ∗ *dateTimeValues*)

Here is the caller graph for this function:

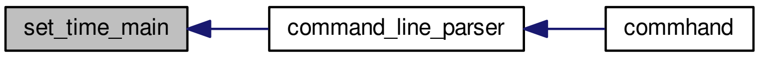


5.9.2.9 int set_time_main (int *argc,* char ∗∗ *argv*)

Here is the call graph for this function:

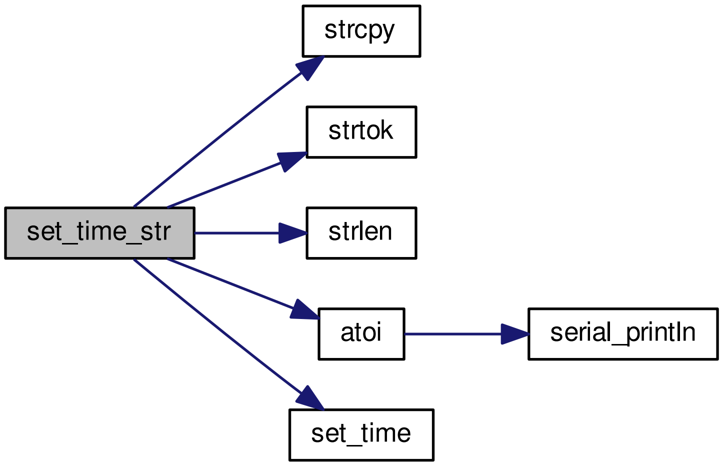


Here is the caller graph for this function:

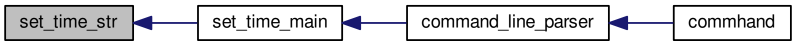


5.9.2.10 error_t set_time_str (const char * *timeStr*)

Here is the call graph for this function:



Here is the caller graph for this function:



Index