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

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 function_name Struct Reference

Public Attributes

- char ∗ **nameStr**
- int(∗ **function**)(int argc, char ∗∗argv)
- char ∗ **usage**
- char ∗ **help**

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

- modules/r1/r1.c

Chapter 5

File Documentation

5.1 include/core/serial.h File Reference

Serial - Header.

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)

5.1.1 Detailed Description

Serial - Header.

Author

Thunder Krakens

Date

February 2nd, 2016

Version

R1

5.2 include/string.h File Reference

Many usefull functions that used for handling string.

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

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*

Parameters

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,...)

5.2.1 Detailed Description

Many usefull functions that used for handling string.

Author

Thunder Krakens

Date

February 2nd, 2016

Version

R1

5.3 kernel/core/serial.c File Reference

Serial.

```
#include <stdint.h>
#include <string.h>
#include <core/io.h>
#include <core/serial.h>
```

Macros

- #define **NO_ERROR** 0
- #define **ESC_KEY** 27
- #define **BRACKET_KEY** 91
- #define **ENTER_KEY** 13
- #define **BACKSPACE_KEY** 127
- #define **DEL_KEY_SEQ_3** 51
- #define **DEL_KEY_SEQ_4** 126
- #define **UP_ARROW** 65
- #define **DOWN_ARROW** 66
- #define **RIGHT_ARROW** 67
- #define **LEFT_ARROW** 68

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)

MoveCursorBackchar.

Move the cursor back for specific times.

Parameters

num	*The number of times that needs to move back.*

Returns

VOID

PrintStars.

Print out the '∗' for specific times.

Parameters

num	*The number of times that needs to print.*

Returns

VOID

EchoInput.

Decides to print out the original string or stars.

Parameters

InputStr	*The string,*
bWithEcho	*Turn on the echo or not.*

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)

Variables

- int **serial_port_out** = 0
- int **serial_port_in** = 0

5.3.1 Detailed Description

Serial.

Author

Thunder Krakens

Date

February 2nd, 2016

Version

R1

5.4 lib/string.c File Reference

Many usefull functions that used for handling string.

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

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

5.5 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.

Macros

- #define **E_NOERROR** 0
- #define **E_INVPARA** 1
- #define **E_INVSTRF** 2

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

5.6 modules/r1/r1.h File Reference

The commandhander and functions associations for Module R1.

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 MaxStrLen)

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

5.7 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>
```

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)

: 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)

5.7.1 Detailed Description

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

Author

Thunder Krakens

Date

February 2nd, 2016

Version

R1

5.8 modules/r1/sys_clock.h File Reference

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

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

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)

5.8.1 Detailed Description

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

Author

Thunder Krakens

Date

February 2nd, 2016

Version

R1