

C8E

0.1

Generated by Doxygen 1.7.2

Thu Dec 23 2010 20:48:40

Contents

1	File Index	1
1.1	File List	1
2	File Documentation	3
2.1	src/C8E.c File Reference	3
2.1.1	Function Documentation	3
2.1.1.1	main	3
2.1.1.2	printUsage	3
2.2	src/CartridgeReader.c File Reference	4
2.2.1	Function Documentation	4
2.2.1.1	readCartridge	4
2.3	src/CartridgeReader.h File Reference	4
2.3.1	Detailed Description	4
2.3.2	Function Documentation	5
2.3.2.1	readCartridge	5
2.4	src/CPU.c File Reference	5
2.4.1	Detailed Description	5
2.4.2	Function Documentation	6
2.4.2.1	cleanupCPU	6
2.4.2.2	handleOpCode	6
2.4.2.3	setupCPU	6
2.4.2.4	tick	6
2.5	src/CPU.h File Reference	6
2.5.1	Detailed Description	7
2.5.2	Define Documentation	7
2.5.2.1	MAX_STACK_SIZE	7
2.5.3	Function Documentation	7
2.5.3.1	cleanupCPU	7
2.5.3.2	setupCPU	7
2.5.3.3	tick	7
2.6	src/Display.c File Reference	7
2.6.1	Define Documentation	8
2.6.1.1	DISPLAY_IDLE_TIME	8
2.6.1.2	SCREEN_HEIGHT	8
2.6.1.3	SCREEN_WIDTH	8
2.6.2	Function Documentation	8
2.6.2.1	cleanupDisplay	8
2.6.2.2	clearScreen	8
2.6.2.3	render	9

2.6.2.4	setupDisplay	9
2.7	src/Display.h File Reference	9
2.7.1	Detailed Description	9
2.7.2	Function Documentation	10
2.7.2.1	cleanupDisplay	10
2.7.2.2	clearScreen	10
2.7.2.3	drawSprite	10
2.7.2.4	render	10
2.7.2.5	setupDisplay	10
2.8	src/Logs.c File Reference	10
2.8.1	Function Documentation	11
2.8.1.1	addEntry	11
2.8.1.2	closeLogs	11
2.8.1.3	setupLogs	11
2.9	src/Logs.h File Reference	11
2.9.1	Define Documentation	12
2.9.1.1	DEFAULT_DEBUG_LEVEL	12
2.9.1.2	DEFAULT_OUTPUT_FILENAME	12
2.9.2	Enumeration Type Documentation	13
2.9.2.1	DEBUG_LEVELS	13
2.9.3	Function Documentation	13
2.9.3.1	addEntry	13
2.9.3.2	closeLogs	13
2.9.3.3	setupLogs	13
2.10	src/Memory.c File Reference	14
2.10.1	Detailed Description	14
2.10.2	Function Documentation	14
2.10.2.1	cleanupMemory	14
2.10.2.2	read	15
2.10.2.3	setupMemory	15
2.10.2.4	write	15
2.11	src/Memory.h File Reference	15
2.11.1	Define Documentation	16
2.11.1.1	DATA_SPACE_START	16
2.11.1.2	DATA_SPACE_STOP	16
2.11.1.3	MAX_REGISTERS	17
2.11.1.4	RESERVED_MEMORY_START	17
2.11.1.5	RESERVED_MEMORY_STOP	17
2.11.2	Function Documentation	17
2.11.2.1	cleanupMemory	17
2.11.2.2	read	17
2.11.2.3	setupMemory	17
2.11.2.4	write	18

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

src/ C8E.c	3
src/ CartridgeReader.c	4
src/ CartridgeReader.h (Define all functions, variables and defines for cartridge management)	4
src/ CPU.c	5
src/ CPU.h	6
src/ Display.c	7
src/ Display.h (Define all functions, variables and defines for display management)	9
src/ Logs.c	10
src/ Logs.h	11
src/ Memory.c (Define all functions, variables and defines for memory management)	14
src/ Memory.h	15

Chapter 2

File Documentation

2.1 src/C8E.c File Reference

```
#include <GLUT/glut.h>
#include <stdlib.h>
#include "Logs.h"
#include "Memory.h"
#include "CartridgeReader.h"
#include "Display.h"
```

Functions

- void [printUsage](#) ()
- int [main](#) (int argc, char **argv)

2.1.1 Function Documentation

2.1.1.1 int main (int *argc*, char ** *argv*)

Definition at line 23 of file C8E.c.

2.1.1.2 void printUsage ()

Definition at line 14 of file C8E.c.

2.2 src/CartridgeReader.c File Reference

```
#include "CartridgeReader.h"
#include <stdio.h>
```

Functions

- int [readCartridge](#) (const char *const filename, unsigned char *data)
Provide a pointer to the cartridge data.

2.2.1 Function Documentation

2.2.1.1 int readCartridge (const char *const filename, unsigned char * data)

Provide a pointer to the cartridge data.

Parameters

in	filename	Point the file to load into memory
out	data	Buffer that eventually receive the cartridge data. It must be initialized and big enough.

Returns

The number of bytes read if the file exists, -1 otherwise.

Definition at line 30 of file CartridgeReader.c.

2.3 src/CartridgeReader.h File Reference

Define all functions, variables and defines for cartridge management.

Functions

- int [readCartridge](#) (const char *const filename, unsigned char *data)
Provide a pointer to the cartridge data.

2.3.1 Detailed Description

Define all functions, variables and defines for cartridge management.

Version

0.1

Date

December 12, 2010

Author

Maxime Gaudin

Definition in file [CartridgeReader.h](#).

2.3.2 Function Documentation**2.3.2.1 int readCartridge (const char *const *filename*, unsigned char * *data*)**

Provide a pointer to the cardridge data.

Parameters

in	<i>filename</i>	Point the file to load into memory
out	<i>data</i>	Buffer that eventually receive the cardridge data. It must be initialized and big enough.

Returns

The number of bytes read if the file exists, -1 otherwise.

Definition at line 30 of file CartridgeReader.c.

2.4 src/CPU.c File Reference

```
#include "CPU.h"
#include "Logs.h"
#include "Display.h"
#include "Memory.h"
```

Functions

- int [setupCPU](#) ()
- void [cleanupCPU](#) ()
- void [tick](#) ()
- void [handleOpCode](#) ()

2.4.1 Detailed Description**Version**

0.1

Date

December 13, 2010

Author

Maxime Gaudin

Definition in file [CPU.c](#).

2.4.2 Function Documentation

2.4.2.1 void cleanupCPU ()

Definition at line 71 of file CPU.c.

2.4.2.2 void handleOpCode ()

Definition at line 353 of file CPU.c.

2.4.2.3 int setupCPU ()

Definition at line 54 of file CPU.c.

2.4.2.4 void tick ()

Definition at line 348 of file CPU.c.

2.5 src/CPU.h File Reference

Defines

- #define [MAX_STACK_SIZE](#) 0xF

Define the maximum stack size, i.e. the maximum amount of subroutine calls.

Functions

- int [setupCPU](#) ()
- void [cleanupCPU](#) ()
- void [tick](#) ()

2.5.1 Detailed Description

Version

0.1

Date

December 12, 2010

Author

Maxime Gaudin

Definition in file [CPU.h](#).

2.5.2 Define Documentation

2.5.2.1 #define MAX_STACK_SIZE 0xF

Define the maximum stack size, i.e. the maximum amount of subroutine calls.

Definition at line 31 of file CPU.h.

2.5.3 Function Documentation

2.5.3.1 void cleanupCPU ()

Definition at line 71 of file CPU.c.

2.5.3.2 int setupCPU ()

Definition at line 54 of file CPU.c.

2.5.3.3 void tick ()

Definition at line 348 of file CPU.c.

2.6 src/Display.c File Reference

```
#include "Display.h"
#include <string.h>
#include <stdlib.h>
#include <GLUT/glut.h>
#include "Logs.h"
```

Defines

- `#define SCREEN_WIDTH 64`
- `#define SCREEN_HEIGHT 32`
- `#define DISPLAY_IDLE_TIME 16`

Functions

- `int setupDisplay (int argc, char **argv)`
Setup all display related memory buffer and glut framework.
- `int cleanupDisplay ()`
- `void render (int)`
- `int clearScreen ()`
Clear screen.

2.6.1 Define Documentation

2.6.1.1 `#define DISPLAY_IDLE_TIME 16`

Definition at line 42 of file Display.c.

2.6.1.2 `#define SCREEN_HEIGHT 32`

Definition at line 40 of file Display.c.

2.6.1.3 `#define SCREEN_WIDTH 64`

Definition at line 39 of file Display.c.

2.6.2 Function Documentation

2.6.2.1 `int cleanupDisplay ()`

Definition at line 57 of file Display.c.

2.6.2.2 `int clearScreen ()`

Clear screen.

Returns

1 if any pixel has been erase, 0 Otherwise.

Definition at line 81 of file Display.c.

2.6.2.3 void render (int)

Definition at line 63 of file Display.c.

2.6.2.4 int setupDisplay (int argc, char ** argv)

Setup all display related memory buffer and glut framework.

Definition at line 46 of file Display.c.

2.7 src/Display.h File Reference

Define all functions, variables and defines for display management.

Functions

- int [setupDisplay](#) (int argc, char **argv)
Setup all display related memory buffer and glut framework.
- int [cleanupDisplay](#) ()
- int [drawSprite](#) (unsigned char X, unsigned char Y, const char *const spriteData, unsigned char len)
Display a [len] byte sprite contained into [spriteData] at ([X], [Y]). TECHNICAL DESCRIPTION TODO.
- void [render](#) (int)
- int [clearScreen](#) ()
Clear screen.

2.7.1 Detailed Description

Define all functions, variables and defines for display management.

Version

0.1

Date

December 12, 2010

Author

Maxime Gaudin

Definition in file [Display.h](#).

2.7.2 Function Documentation

2.7.2.1 int cleanupDisplay ()

Definition at line 57 of file Display.c.

2.7.2.2 int clearScreen ()

Clear screen.

Returns

1 if any pixel has been erase, 0 Otherwise.

Definition at line 81 of file Display.c.

2.7.2.3 int drawSprite (unsigned char *X*, unsigned char *Y*, const char *const *spriteData*, unsigned char *len*)

Display a [*len*] byte sprite contained into [*spriteData*] at ([*X*], [*Y*]). TECHNICAL DESCRIPTION TODO.

Returns

1 if any pixel has been erase, 0 Otherwise.

2.7.2.4 void render (int)

Definition at line 63 of file Display.c.

2.7.2.5 int setupDisplay (int *argc*, char ** *argv*)

Setup all display related memory buffer and glut framework.

Definition at line 46 of file Display.c.

2.8 src/Logs.c File Reference

```
#include "Logs.h"
```

Functions

- int [setupLogs](#) (int redirect, unsigned char debugLevel, char *const outputFile-name)

Setup output log file and debug level to values passed in paramaters. Moreover, a file descriptor is created and initialized. if [redirect], log are also written in stdou.

- int `closeLogs` ()

Close output log file descriptor and flush file buffer.

- void `addEntry` (unsigned char level, const char *const message)

Add new entry in output log file if [level] is below or equal to debug level.

2.8.1 Function Documentation

2.8.1.1 void addEntry (unsigned char level, const char *const message)

Add new entry in output log file if [level] is below or equal to debug level.

Definition at line 56 of file Logs.c.

2.8.1.2 int closeLogs ()

Close output log file descriptor and flush file buffer.

Returns

0 if success, 0 otherwise.

Definition at line 50 of file Logs.c.

2.8.1.3 int setupLogs (int redirect, unsigned char debugLevel, char *const outputFilename)

Setup output log file and debug level to values passed in paramaters. Moreover, a file descriptor is created and initialized. if [redirect], log are also written in stdou.

Returns

0 if success, 0 otherwise.

Definition at line 36 of file Logs.c.

2.9 src/Logs.h File Reference

```
#include <stdio.h>
```

Defines

- `#define DEFAULT_DEBUG_LEVEL 1`
Specifies the default debug level : Warning.
- `#define DEFAULT_OUTPUT_FILENAME "DEBUG_LOGS"`
Specifies the default output filename, i.e. the file where log will be written.

Enumerations

- `enum DEBUG_LEVELS {`
 `ERROR = 0, WARNING = 1, DRAWING = 2, DISASSEMBLY = 3,`
 `LOW_LEVEL_OPERATION = 4 }`

Functions

- `int setupLogs (int redirect, unsigned char debugLevel, char *const outputFile-name)`
Setup output log file and debug level to values passed in parameters. Moreover, a file descriptor is created and initialized. if [redirect], log are also written in stdout.
- `int closeLogs ()`
Close output log file descriptor and flush file buffer.
- `void addEntry (unsigned char level, const char *const message)`
Add new entry in output log file if [level] is below or equal to debug level.

2.9.1 Define Documentation

2.9.1.1 `#define DEFAULT_DEBUG_LEVEL 1`

Specifies the default debug level : Warning.

Definition at line 23 of file Logs.h.

2.9.1.2 `#define DEFAULT_OUTPUT_FILENAME "DEBUG_LOGS"`

Specifies the default output filename, i.e. the file where log will be written.

Definition at line 26 of file Logs.h.

2.9.2 Enumeration Type Documentation

2.9.2.1 enum DEBUG_LEVELS

Enumerator:

ERROR

WARNING

DRAWING

DISASSEMBLY

LOW_LEVEL_OPERATION

Definition at line 20 of file Logs.h.

2.9.3 Function Documentation

2.9.3.1 void addEntry (unsigned char *level*, const char *const *message*)

Add new entry in output log file if [level] is below or equal to debug level.

Definition at line 56 of file Logs.c.

2.9.3.2 int closeLogs ()

Close output log file descriptor and flush file buffer.

Returns

0 if success, 0 otherwise.

Definition at line 50 of file Logs.c.

2.9.3.3 int setupLogs (int *redirect*, unsigned char *debugLevel*, char *const *outputFilename*)

Setup output log file and debug level to values passed in paramaters. Moreover, a file descriptor is created and initialized. if [redirect], log are also written in stdou.

Returns

0 if success, 0 otherwise.

Definition at line 36 of file Logs.c.

2.10 src/Memory.c File Reference

Define all functions, variables and defines for memory management.

```
#include "Memory.h"
#include "Logs.h"
#include <stdlib.h>
#include <string.h>
```

Functions

- int [setupMemory](#) ()
Initialize memory to 0.
- void [cleanupMemory](#) ()
Cleanup all memory.
- int [write](#) (unsigned short addr, unsigned char *const data, unsigned int len)
write [len] bytes from [data] into memory at adress [addr]
- int [read](#) (short addr, unsigned short len, unsigned char *const buffer)
Read [len] bytes of data from address [addr] to buffer.

2.10.1 Detailed Description

Define all functions, variables and defines for memory management.

Version

0.1

Date

December 12, 2010

Author

Maxime Gaudin

Definition in file [Memory.c](#).

2.10.2 Function Documentation

2.10.2.1 void [cleanupMemory](#) ()

Cleanup all memory.

Definition at line 39 of file Memory.c.

2.10.2.2 int read (short *addr*, unsigned short *len*, unsigned char *const *buffer*)

Read [*len*] bytes of data from address [*addr*] to buffer.

Parameters

in	<i>addr</i>	Address where read begins
in	<i>len</i>	Number of bytes read
out	<i>buffer</i>	Pointer to the data buffer

Returns

0 if success, 1 otherwise.

Definition at line 56 of file Memory.c.

2.10.2.3 int setupMemory ()

Initialize memory to 0.

Returns

0 if success, 1 otherwise.

Definition at line 28 of file Memory.c.

2.10.2.4 int write (unsigned short *addr*, unsigned char *const *data*, unsigned int *len*)

write [*len*] bytes from [*data*] into memory at adress [*addr*]

Parameters

in	<i>addr</i>	Address where data will be written
in	<i>data</i>	Pointer to data buffer
in	<i>len</i>	Number of byte written

Returns

0 if success, 1 otherwise.

Definition at line 44 of file Memory.c.

2.11 src/Memory.h File Reference

Defines

- #define [RESERVED_MEMORY_START](#) 0x0
Specifies where memory starts (0x0, what a surprise isn't it ??).

- `#define RESERVED_MEMORY_STOP 0x200`
Specifies where the memory stops.
- `#define DATA_SPACE_START 0x200`
Specifies the beginning of the data space.
- `#define DATA_SPACE_STOP 0xFF`
Specifies the end of the data space.
- `#define MAX_REGISTERS 0xF`
Specifies the maximum number of registers..

Functions

- `int setupMemory ()`
Initialize memory to 0.
- `void cleanupMemory ()`
Cleanup all memory.
- `int write (unsigned short addr, unsigned char *const data, unsigned int len)`
write [len] bytes from [data] into memory at adress [addr]
- `int read (short addr, unsigned short len, unsigned char *const buffer)`
Read [len] bytes of data from address [addr] to buffer.

2.11.1 Define Documentation

2.11.1.1 `#define DATA_SPACE_START 0x200`

Specifies the beginning of the data space.

Definition at line 36 of file Memory.h.

2.11.1.2 `#define DATA_SPACE_STOP 0xFF`

Specifies the end of the data space.

Definition at line 38 of file Memory.h.

2.11.1.3 #define MAX_REGISTERS 0xF

Specifies the maximum number of registers..

Definition at line 41 of file Memory.h.

2.11.1.4 #define RESERVED_MEMORY_START 0x0

Specifies where memory starts (0x0, what a surprise isn't it ??).

Definition at line 31 of file Memory.h.

2.11.1.5 #define RESERVED_MEMORY_STOP 0x200

Specifies where the memory stops.

Definition at line 33 of file Memory.h.

2.11.2 Function Documentation

2.11.2.1 void cleanupMemory ()

Cleanup all memory.

Definition at line 39 of file Memory.c.

2.11.2.2 int read (short *addr*, unsigned short *len*, unsigned char *const *buffer*)

Read [*len*] bytes of data from address [*addr*] to buffer.

Parameters

in	<i>addr</i>	Address where read begins
in	<i>len</i>	Number of bytes read
out	<i>buffer</i>	Pointer to the data buffer

Returns

0 if success, 1 otherwise.

Definition at line 56 of file Memory.c.

2.11.2.3 int setupMemory ()

Initialize memory to 0.

Returns

0 if success, 1 otherwise.

Definition at line 28 of file Memory.c.

2.11.2.4 `int write (unsigned short addr, unsigned char *const data, unsigned int len)`

write [len] bytes from [data] into memory at adress [addr]

Parameters

<code>in</code>	<code><i>addr</i></code>	Address where data will be written
<code>in</code>	<code><i>data</i></code>	Pointer to data buffer
<code>in</code>	<code><i>len</i></code>	Number of byte written

Returns

0 if success, 1 otherwise.

Definition at line 44 of file Memory.c.

Index

addEntry
 Logs.c, [11](#)
 Logs.h, [13](#)

C8E.c
 main, [3](#)
 printUsage, [3](#)

CartridgeReader.c
 readCartridge, [4](#)

CartridgeReader.h
 readCartridge, [5](#)

cleanupCPU
 CPU.c, [6](#)
 CPU.h, [7](#)

cleanupDisplay
 Display.c, [8](#)
 Display.h, [10](#)

cleanupMemory
 Memory.c, [14](#)
 Memory.h, [17](#)

clearScreen
 Display.c, [8](#)
 Display.h, [10](#)

closeLogs
 Logs.c, [11](#)
 Logs.h, [13](#)

CPU.c
 cleanupCPU, [6](#)
 handleOpCode, [6](#)
 setupCPU, [6](#)
 tick, [6](#)

CPU.h
 cleanupCPU, [7](#)
 MAX_STACK_SIZE, [7](#)
 setupCPU, [7](#)
 tick, [7](#)

DATA_SPACE_START
 Memory.h, [16](#)

DATA_SPACE_STOP
 Memory.h, [16](#)

DEBUG_LEVELS
 Logs.h, [13](#)

DEFAULT_DEBUG_LEVEL
 Logs.h, [12](#)

DEFAULT_OUTPUT_FILENAME
 Logs.h, [12](#)

DISASSEMBLY
 Logs.h, [13](#)

Display.c
 cleanupDisplay, [8](#)
 clearScreen, [8](#)
 DISPLAY_IDLE_TIME, [8](#)
 render, [8](#)
 SCREEN_HEIGHT, [8](#)
 SCREEN_WIDTH, [8](#)
 setupDisplay, [9](#)

Display.h
 cleanupDisplay, [10](#)
 clearScreen, [10](#)
 drawSprite, [10](#)
 render, [10](#)
 setupDisplay, [10](#)

DISPLAY_IDLE_TIME
 Display.c, [8](#)

DRAWING
 Logs.h, [13](#)

drawSprite
 Display.h, [10](#)

ERROR
 Logs.h, [13](#)

handleOpCode
 CPU.c, [6](#)

Logs.c
 addEntry, [11](#)
 closeLogs, [11](#)
 setupLogs, [11](#)

Logs.h
 addEntry, [13](#)

- closeLogs, [13](#)
- DEBUG_LEVELS, [13](#)
- DEFAULT_DEBUG_LEVEL, [12](#)
- DEFAULT_OUTPUT_FILENAME, [12](#)
- DISASSEMBLY, [13](#)
- DRAWING, [13](#)
- ERROR, [13](#)
- LOW_LEVEL_OPERATION, [13](#)
- setupLogs, [13](#)
- WARNING, [13](#)
- LOW_LEVEL_OPERATION
 - Logs.h, [13](#)
- main
 - C8E.c, [3](#)
- MAX_REGISTERS
 - Memory.h, [16](#)
- MAX_STACK_SIZE
 - CPU.h, [7](#)
- Memory.c
 - cleanupMemory, [14](#)
 - read, [14](#)
 - setupMemory, [15](#)
 - write, [15](#)
- Memory.h
 - cleanupMemory, [17](#)
 - DATA_SPACE_START, [16](#)
 - DATA_SPACE_STOP, [16](#)
 - MAX_REGISTERS, [16](#)
 - read, [17](#)
 - RESERVED_MEMORY_START, [17](#)
 - RESERVED_MEMORY_STOP, [17](#)
 - setupMemory, [17](#)
 - write, [18](#)
- printUsage
 - C8E.c, [3](#)
- read
 - Memory.c, [14](#)
 - Memory.h, [17](#)
- readCartridge
 - CartridgeReader.c, [4](#)
 - CartridgeReader.h, [5](#)
- render
 - Display.c, [8](#)
 - Display.h, [10](#)
- RESERVED_MEMORY_START
 - Memory.h, [17](#)
- RESERVED_MEMORY_STOP
 - Memory.h, [17](#)
- SCREEN_HEIGHT
 - Display.c, [8](#)
- SCREEN_WIDTH
 - Display.c, [8](#)
- setupCPU
 - CPU.c, [6](#)
 - CPU.h, [7](#)
- setupDisplay
 - Display.c, [9](#)
 - Display.h, [10](#)
- setupLogs
 - Logs.c, [11](#)
 - Logs.h, [13](#)
- setupMemory
 - Memory.c, [15](#)
 - Memory.h, [17](#)
- src/C8E.c, [3](#)
- src/CartridgeReader.c, [4](#)
- src/CartridgeReader.h, [4](#)
- src/CPU.c, [5](#)
- src/CPU.h, [6](#)
- src/Display.c, [7](#)
- src/Display.h, [9](#)
- src/Logs.c, [10](#)
- src/Logs.h, [11](#)
- src/Memory.c, [14](#)
- src/Memory.h, [15](#)
- tick
 - CPU.c, [6](#)
 - CPU.h, [7](#)
- WARNING
 - Logs.h, [13](#)
- write
 - Memory.c, [15](#)
 - Memory.h, [18](#)