octochip8

Generated by Doxygen 1.8.4

Fri Jan 17 2014 17:00:47

Contents

1 Class Index **1**

1.1 Class List 1

2 File Index **3**

2.1 File List 3

3 Class Documentation **5**

3.1 CPU Class Reference 5

3.1.1 Constructor & Destructor Documentation 6

3.1.1.1 CPU 6

3.1.1.2 ~CPU 6

3.1.2 Member Function Documentation 6

3.1.2.1 emulateCycle 6

3.1.2.2 executeOpcode 6

3.1.2.3 get_byte 7

3.1.2.4 getDrawFlag 7

3.1.2.5 getGFX 7

3.1.2.6 initalize 8

3.1.2.7 loadGame 8

3.1.2.8 setDrawFlag 8

3.1.2.9 setKeys 8

3.1.2.10 setOpcode 8

3.1.3 Member Data Documentation 8

3.1.3.1 drawFlag 8

3.1.3.2 gfx 8

3.1.3.3 I 8

3.1.3.4 key 8

3.1.3.5 memory 8

3.1.3.6 opcode 8

3.1.3.7 pc 9

3.1.3.8 running 9

3.1.3.9 SCREEN_HEIGHT 9

3.1.3.10 SCREEN_SIZE . . . 9
3.1.3.11 SCREEN_WIDTH . . . 9
3.1.3.12 sp . . . 9
3.1.3.13 stack . . . 9
3.1.3.14 V . . . 9
3.2 Graphics Class Reference . . . 9
3.2.1 Constructor & Destructor Documentation . . . 10
3.2.1.1 Graphics . . . 10
3.2.1.2 ∼Graphics . . . 10
3.2.2 Member Function Documentation . . . 10
3.2.2.1 draw . . . 10
3.2.2.2 initalize . . . 10
3.3 Input Class Reference . . . 10
3.3.1 Constructor & Destructor Documentation . . . 10
3.3.1.1 Input . . . 10
3.3.1.2 Input . . . 10
3.3.1.3 ∼Input . . . 10
3.3.2 Member Function Documentation . . . 10
3.3.2.1 initalize . . . 10

4 File Documentation **11**
4.1 src/CPU.cpp File Reference . . . 11
4.2 src/CPU.h File Reference . . . 11
4.3 src/Graphics.cpp File Reference . . . 11
4.4 src/Graphics.h File Reference . . . 11
4.5 src/Input.cpp File Reference . . . 12
4.6 src/Input.h File Reference . . . 12
4.7 src/octochip8.cpp File Reference . . . 12
4.7.1 Function Documentation . . . 12
4.7.1.1 main . . . 12
4.7.2 Variable Documentation . . . 12
4.7.2.1 cpu . . . 12
4.7.2.2 gpu . . . 12
4.7.2.3 input . . . 12

Index **13**

Chapter 1

Class Index

1.1 Class List

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

CPU 5
Graphics 9
Input 10

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

src/CPU.cpp 11
src/CPU.h 11
src/Graphics.cpp 11
src/Graphics.h 11
src/Input.cpp 12
src/Input.h 12
src/octochip8.cpp 12

Chapter 3

Class Documentation

3.1 CPU Class Reference

`#include <CPU.h>`

Public Member Functions

- CPU ()
- virtual ~CPU ()
- void initalize ()
- void loadGame (std::string filenamne)
- void emulateCycle ()
- bool getDrawFlag ()
- void setDrawFlag (bool flag)
- void setKeys ()
- vector< bool > getGFX ()

Public Attributes

- bool running

Static Public Attributes

- static const int SCREEN_SIZE = 64 * 32
- static const int SCREEN_WIDTH = 64
- static const int SCREEN_HEIGHT = 32

Private Member Functions

- void setOpcode ()
- void executeOpcode ()
- char get_byte (short number, int n)

Private Attributes

- unsigned short opcode
- vector< unsigned char > memory
- vector< unsigned char > V
- unsigned short I
- unsigned short pc
- vector< bool > gfx
- vector< unsigned short > stack
- unsigned short sp
- vector< unsigned char > key
- bool drawFlag

3.1.1 Constructor & Destructor Documentation

3.1.1.1 CPU::CPU ( )

The constructor for the class, initalise must be called after this to be used.

3.1.1.2 CPU::~CPU ( ) [virtual]

3.1.2 Member Function Documentation

3.1.2.1 void CPU::emulateCycle ( )

Emulate a cycle of the CPU

3.1.2.2 void CPU::executeOpcode ( ) [private]

Executes the opcode of the current program.

Opcode Table:

Opcode Explanation

0NNN Calls RCA 1802 program at address NNN.

00E0 Clears the screen.

00EE Returns from a subroutine.

1NNN Jumps to address NNN.

2NNN Calls subroutine at NNN.

3XNN Skips the next instruction if VX equals NN.

4XNN Skips the next instruction if VX doesn't equal NN.

5XY0 Skips the next instruction if VX equals VY.

6XNN Sets VX to NN.

7XNN Adds NN to VX.

8XY0 Sets VX to the value of VY.

8XY1 Sets VX to VX or VY.

8XY2 Sets VX to VX and VY.

8XY3 Sets VX to VX xor VY.

8XY4 Adds VY to VX. VF is set to 1 when there's a carry, and to 0 when there isn't.

8XY5 VY is subtracted from VX. VF is set to 0 when there's a borrow, and 1 when there isn't.

8XY6 Shifts VX right by one. VF is set to the value of the least significant bit of VX before the shift.[2]

8XY7 Sets VX to VY minus VX. VF is set to 0 when there's a borrow, and 1 when there isn't.

8XYE Shifts VX left by one. VF is set to the value of the most significant bit of VX before the shift.[2]

9XY0 Skips the next instruction if VX doesn't equal VY.

ANNN Sets I to the address NNN.

BNNN Jumps to the address NNN plus V0.

CXNN Sets VX to a random number and NN.

DXYN Draws a sprite at coordinate (VX, VY) that has a width of 8 pixels and a height of N pixels. Each row of 8 pixels is read as bit-coded (with the most significant bit of each byte displayed on the left) starting from memory location I; I value doesn't change after the execution of this instruction. As described above, VF is set to 1 if any screen pixels are flipped from set to unset when the sprite is drawn, and to 0 if that doesn't happen.

EX9E Skips the next instruction if the key stored in VX is pressed.

EXA1 Skips the next instruction if the key stored in VX isn't pressed.

FX07 Sets VX to the value of the delay timer.

FX0A A key press is awaited, and then stored in VX.

FX15 Sets the delay timer to VX.

FX18 Sets the sound timer to VX.

FX1E Adds VX to I.[3]

FX29 Sets I to the location of the sprite for the character in VX. Characters 0-F (in hexadecimal) are represented by a 4x5 font.

FX33 Stores the Binary-coded decimal representation of VX, with the most significant of three digits at the address in I, the middle digit at I plus 1, and the least significant digit at I plus 2. (In other words, take the decimal representation of VX, place the hundreds digit in memory at location in I, the tens digit at location I+1, and the ones digit at location I+2.)

FX55 Stores V0 to VX in memory starting at address I.[4]

FX65 Fills V0 to VX with values from memory starting at address I.[4]

3.1.2.3 char CPU::get_byte (short *number,* int *n*) [private]

Gets the nth byte of a short. 0 is the least signifigant byte and 1 is the most signifigant byte.

Parameters

number	The short which to split The byte to return, from least to most signifigant.

3.1.2.4 bool CPU::getDrawFlag ( )

Gets the current draw flag determining wether or not to draw during this cpu cycle.

Returns

A bool of the current draw flag. True = Draw screen. False = Don't draw screen.

3.1.2.5 vector< bool > CPU::getGFX ( )

Gets the vector object of pixels for the current screen.

3.1.2.6 void CPU::initalize ( )

Called after construction, sets up all registers and memory.

3.1.2.7 void CPU::loadGame (std::string *filenamne*)

Load a game into the emulator.

Parameters

filenamne	The file to load. Type will probably change later.

3.1.2.8 void CPU::setDrawFlag (bool *flag*)

Sets the draw flag. Should be called at end of every cpu loop.

Parameters

flag	The boolean value to set the draw flag.

3.1.2.9 void CPU::setKeys ( )

Sets the keys for the current screen.

3.1.2.10 void CPU::setOpcode ( ) [private]

Sets the opcode of the current program to the instruction at PC.

3.1.3 Member Data Documentation

3.1.3.1 bool CPU::drawFlag [private]

< The current state of the key

3.1.3.2 vector<bool> CPU::gfx [private]

A vector representing the current screen

3.1.3.3 unsigned short CPU::I [private]

The index register, counts down from value to 0 when in use.

3.1.3.4 vector<unsigned char> CPU::key [private]

3.1.3.5 vector<unsigned char> CPU::memory [private]

The virtual memory - 8k memory

3.1.3.6 unsigned short CPU::opcode [private]

The current operation code.

3.1.3.7 unsigned short CPU::pc [private]

The program counter, counts down from value to 0 when in use.

3.1.3.8 bool CPU::running

The height of the screen in pixels. A boolean for the running state of the CPU

3.1.3.9 const int CPU::SCREEN_HEIGHT = 32 [static]

The width of the screen in pixels.

3.1.3.10 const int CPU::SCREEN_SIZE = 64 * 32 [static]

The amount of pixels for the screen

3.1.3.11 const int CPU::SCREEN_WIDTH = 64 [static]

3.1.3.12 unsigned short CPU::sp [private]

The pointer to the current level in the stack

3.1.3.13 vector<unsigned short> CPU::stack [private]

The stack. Has 16 levels. ha pancakes

3.1.3.14 vector<unsigned char> CPU::V [private]

The CPU registers. CPU registers: The Chip 8 has 15 8-bit general purpose registers named V0,V1 up to VE. The 16th register is used for the 'carry flag'.

The documentation for this class was generated from the following files:

- src/CPU.h
- src/CPU.cpp

3.2 Graphics Class Reference

`#include <Graphics.h>`

Public Member Functions

- void initalize ()
- void draw (std::vector< bool > screen)
- Graphics ()
- virtual ~Graphics ()

3.2.1 Constructor & Destructor Documentation

3.2.1.1 Graphics::Graphics ( )

3.2.1.2 Graphics::~Graphics ( ) [virtual]

3.2.2 Member Function Documentation

3.2.2.1 void Graphics::draw (std::vector< bool > *screen*)

3.2.2.2 void Graphics::initalize ( )

The documentation for this class was generated from the following files:

- src/Graphics.h
- src/Graphics.cpp

3.3 Input Class Reference

`#include <Input.h>`

Public Member Functions

- void initalize ()
- Input ()
- Input (const Input &orig)
- virtual ~Input ()

3.3.1 Constructor & Destructor Documentation

3.3.1.1 Input::Input ( )

3.3.1.2 Input::Input (const Input & *orig*)

3.3.1.3 Input::~Input ( ) [virtual]

3.3.2 Member Function Documentation

3.3.2.1 void Input::initalize ( )

The documentation for this class was generated from the following files:

- src/Input.h
- src/Input.cpp

Chapter 4

File Documentation

4.1 src/CPU.cpp File Reference

```
#include <algorithm>
#include <iostream>
#include "CPU.h"
```

4.2 src/CPU.h File Reference

```
#include <string>
#include <vector>
```

Classes

- class CPU

4.3 src/Graphics.cpp File Reference

```
#include "Graphics.h"
```

4.4 src/Graphics.h File Reference

```
#include "CPU.h"
#include <vector>
```

Classes

- class Graphics

4.5 src/Input.cpp File Reference

```
#include "Input.h"
```

4.6 src/Input.h File Reference

Classes

- class Input

4.7 src/octochip8.cpp File Reference

```
#include "CPU.h"
#include "Graphics.h"
#include "Input.h"
#include <iostream>
#include <string>
```

Functions

- int main (void)

Variables

- CPU cpu
- Graphics gpu
- Input input

4.7.1 Function Documentation

4.7.1.1 int main (void)

The input object to be used The main function to start it all off.

Returns

Exit code. 0 is normal.

4.7.2 Variable Documentation

4.7.2.1 CPU cpu

The CPU object to be used

4.7.2.2 Graphics gpu

4.7.2.3 Input input

The GPU object to be used

Index

~CPU
    CPU, 6
~Graphics
    Graphics, 10
~Input
    Input, 10

CPU, 5
    ~CPU, 6
    CPU, 6
    CPU, 6
    drawFlag, 8
    emulateCycle, 6
    executeOpcode, 6
    get_byte, 7
    getDrawFlag, 7
    getGFX, 7
    gfx, 8
    I, 8
    initalize, 7
    key, 8
    loadGame, 8
    memory, 8
    opcode, 8
    pc, 8
    running, 9
    SCREEN_HEIGHT, 9
    SCREEN_SIZE, 9
    SCREEN_WIDTH, 9
    setDrawFlag, 8
    setKeys, 8
    setOpcode, 8
    sp, 9
    stack, 9
    V, 9
cpu
    octochip8.cpp, 12

draw
    Graphics, 10
drawFlag
    CPU, 8

emulateCycle
    CPU, 6
executeOpcode
    CPU, 6

get_byte
    CPU, 7
getDrawFlag
    CPU, 7
getGFX
    CPU, 7
gfx
    CPU, 8
gpu
    octochip8.cpp, 12
Graphics, 9
    ~Graphics, 10
    draw, 10
    Graphics, 10
    initalize, 10

I
    CPU, 8
initalize
    CPU, 7
    Graphics, 10
    Input, 10
Input, 10
    ~Input, 10
    initalize, 10
    Input, 10
input
    octochip8.cpp, 12

key
    CPU, 8

loadGame
    CPU, 8

main
    octochip8.cpp, 12
memory
    CPU, 8

octochip8.cpp
    cpu, 12
    gpu, 12
    input, 12
    main, 12
opcode
    CPU, 8

pc
    CPU, 8

running
    CPU, 9

SCREEN_HEIGHT
    CPU, 9
SCREEN_SIZE
    CPU, 9
SCREEN_WIDTH
    CPU, 9
setDrawFlag
    CPU, 8
setKeys
    CPU, 8
setOpcode
    CPU, 8
sp
    CPU, 9
src/CPU.cpp, 11
src/CPU.h, 11
src/Graphics.cpp, 11
src/Graphics.h, 11
src/Input.cpp, 12
src/Input.h, 12
src/octochip8.cpp, 12
stack
    CPU, 9

V
    CPU, 9