

Chip8

0.3

Generated by Doxygen 1.8.17



# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">Chip8</a>	CHIP-8 Emulator . . . . .	??
<a href="#">Platform</a>	Handles the window, rendering, and input for the CHIP-8 emulator using SDL . . . . .	??



## Chapter 2

# File Index

### 2.1 File List

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

/home/zk/Git_Repo/CHIP-8_emulator/include/ <a href="#">Chip8.h</a>	
CHIP-8 Emulator Class . . . . .	??
/home/zk/Git_Repo/CHIP-8_emulator/include/ <a href="#">Chip8_common.h</a>	
Constant values, useful for building the emulator . . . . .	??
/home/zk/Git_Repo/CHIP-8_emulator/include/ <a href="#">Platform.h</a>	
<a href="#">Platform</a> and input output handling Class . . . . .	??



## Chapter 3

# Class Documentation

### 3.1 Chip8 Class Reference

CHIP-8 Emulator.

```
#include <Chip8.h>
```

#### Public Member Functions

- [Chip8](#) ()  
*Constructor: Initializes the CHIP-8 emulator.*
- [~Chip8](#) ()=default  
*Default destructor.*
- [std::array< uint8\\_t, NUM\\_KEYS > get\\_keypad](#) () const  
*Get the current state of the keypad.*
- [std::array< uint32\\_t, VIDEO\\_HEIGHT \\*VIDEO\\_WIDTH > get\\_video](#) () const  
*Get the current state of the video display.*
- [void LoadROM](#) (const std::string &filename)  
*Loads a ROM file into memory.*
- [void OP\\_00E0](#) ()  
*Clears the display.*
- [void OP\\_00EE](#) ()  
*Returns from a subroutine.*
- [void OP\\_1nnn](#) ()  
*Jumps to address NNN.*
- [void OP\\_2nnn](#) ()  
*Calls subroutine at address NNN.*
- [void OP\\_3xkk](#) ()  
*Skips next instruction if Vx == kk.*
- [void OP\\_4xkk](#) ()  
*Skips next instruction if Vx != kk.*
- [void OP\\_5xy0](#) ()  
*Skips next instruction if Vx == Vy.*
- [void OP\\_6xkk](#) ()  
*Sets Vx = kk.*

- void [OP\\_7xkk](#) ()  
*Sets  $Vx = Vx + kk$ .*
- void [OP\\_8xy0](#) ()  
*Sets  $Vx = Vy$ .*
- void [OP\\_8xy1](#) ()  
*Performs bitwise OR:  $Vx = Vx \mid Vy$ .*
- void [OP\\_8xy2](#) ()  
*Performs bitwise AND:  $Vx = Vx \& Vy$ .*
- void [OP\\_8xy3](#) ()  
*Performs bitwise XOR:  $Vx = Vx \wedge Vy$ .*
- void [OP\\_8xy4](#) ()  
*Adds  $Vx$  and  $Vy$ , storing result in  $Vx$  and setting  $VF$  if overflow occurs.*
- void [OP\\_8xy5](#) ()  
*Subtracts  $Vy$  from  $Vx$ , storing result in  $Vx$  and setting  $VF$  to NOT borrow.*
- void [OP\\_8xy6](#) ()  
*Shifts  $Vx$  right by 1, storing the least significant bit in  $VF$ .*
- void [OP\\_8xy7](#) ()  
*Sets  $Vx = Vy - Vx$ , setting  $VF$  to NOT borrow.*
- void [OP\\_8xyE](#) ()  
*Shifts  $Vx$  left by 1, storing the most significant bit in  $VF$ .*
- void [OP\\_9xy0](#) ()  
*Skips next instruction if  $Vx \neq Vy$ .*
- void [OP\\_Annn](#) ()  
*Sets  $I = NNN$ .*
- void [OP\\_Bnnn](#) ()  
*Jumps to location  $NNN + V0$ .*
- void [OP\\_Cxkk](#) ()  
*Sets  $Vx = \text{random byte} \& kk$ .*
- void [OP\\_Dxyn](#) ()  
*Draws a sprite at coordinate  $(Vx, Vy)$ .*
- void [OP\\_Ex9E](#) ()  
*Skips next instruction if key  $Vx$  is pressed.*
- void [OP\\_ExA1](#) ()  
*Skips next instruction if key  $Vx$  is not pressed.*
- void [OP\\_Fx07](#) ()  
*Sets  $Vx$  to the value of the delay timer.*
- void [OP\\_Fx0A](#) ()  
*Waits for a key press and stores the value in  $Vx$ .*
- void [OP\\_Fx0A\\_optimized](#) ()  
*Optimized version of [OP\\_Fx0A](#).*
- void [OP\\_Fx15](#) ()  
*Sets the delay timer to  $Vx$ .*
- void [OP\\_Fx18](#) ()  
*Sets the sound timer to  $Vx$ .*
- void [OP\\_Fx1E](#) ()  
*Sets  $I = I + Vx$ .*
- void [OP\\_Fx29](#) ()  
*Sets  $I$  to the location of the sprite for digit  $Vx$ .*
- void [OP\\_Fx33](#) ()  
*Stores the BCD representation of  $Vx$  at memory locations  $I$ ,  $I+1$ , and  $I+2$ .*
- void [OP\\_Fx55](#) ()



- Stores registers V0 through Vx in memory starting at I.*
  - void [OP\\_Fx65](#) ()
- Reads registers V0 through Vx from memory starting at I.*
  - void [Table0](#) ()
- Handles OP\_00E\* opcode.*
  - void [Table8](#) ()
- Handles OP\_8xy\* opcode.*
  - void [TableE](#) ()
- Handles OP\_Ex\*\* opcode.*
  - void [TableF](#) ()
- Handles OP\_Fx\*\* opcode.*
  - void [OP\\_NULL](#) ()
- Handles unimplemented opcodes (NOP).*
  - void [Cycle](#) ()
- Executes one cycle of the CHIP-8 CPU.*

## Friends

- class [TestChip8](#)  
*Allows TestChip8 to access private members for unit testing.*

### 3.1.1 Detailed Description

CHIP-8 Emulator.

This class emulates a CHIP-8 CPU, providing methods to load ROMs, execute instructions, and interact with the display and keypad.

### 3.1.2 Member Function Documentation

#### 3.1.2.1 `get_keypad()`

```
std::array<uint8_t, NUM_KEYS> Chip8::get_keypad ( ) const [inline]
```

Get the current state of the keypad.

#### Returns

An array representing the keypad state (pressed keys).

### 3.1.2.2 get\_video()

```
std::array<uint32_t, VIDEO_HEIGHT * VIDEO_WIDTH> Chip8::get_video ( ) const [inline]
```

Get the current state of the video display.

#### Returns

An array representing the monochrome 64x32 display.

### 3.1.2.3 LoadROM()

```
void Chip8::LoadROM (
    const std::string & filename )
```

Loads a ROM file into memory.

#### Parameters

<i>filename</i>	Path to the ROM file.
-----------------	-----------------------

### 3.1.2.4 OP\_Dxyn()

```
void Chip8::OP_Dxyn ( )
```

Draws a sprite at coordinate (Vx, Vy).

The sprite is *n* bytes in height and starts at memory location *l*. VF is set to 1 if any pixels are erased due to collision.

### 3.1.2.5 OP\_Fx0A()

```
void Chip8::OP_Fx0A ( )
```

Waits for a key press and stores the value in Vx.

Execution pauses until a key is pressed.

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

- [/home/zk/Git\\_Repo/CHIP-8\\_emulator/include/Chip8.h](#)
- [/home/zk/Git\\_Repo/CHIP-8\\_emulator/src/chip8/Chip8.cpp](#)
- [/home/zk/Git\\_Repo/CHIP-8\\_emulator/src/chip8/Opcodes.cpp](#)

## 3.2 Platform Class Reference

Handles the window, rendering, and input for the CHIP-8 emulator using SDL.

```
#include <Platform.h>
```

### Public Member Functions

- [Platform](#) (const char \*title, int windowWidth, int windowHeight, int textureWidth, int textureHeight)  
*Constructs a [Platform](#) object.*
- [~Platform](#) ()  
*Destroys the [Platform](#) object and cleans up SDL resources.*
- void [Update](#) (const void \*buffer, int pitch)  
*Updates the display with new pixel data.*
- bool [ProcessInput](#) (uint8\_t \*keys)  
*Processes user input and updates the CHIP-8 keypad state.*

### Friends

- class [TestPlatform](#)

### 3.2.1 Detailed Description

Handles the window, rendering, and input for the CHIP-8 emulator using SDL.

### 3.2.2 Constructor & Destructor Documentation

#### 3.2.2.1 Platform()

```
Platform::Platform (  
    const char * title,  
    int windowWidth,  
    int windowHeight,  
    int textureWidth,  
    int textureHeight )
```

Constructs a [Platform](#) object.

#### Parameters

<i>title</i>	The title of the window.
<i>windowWidth</i>	The width of the window in pixels.
<i>windowHeight</i>	The height of the window in pixels.
<i>textureWidth</i>	The width of the CHIP-8 screen texture.
<i>textureHeight</i>	The height of the CHIP-8 screen texture.

### 3.2.3 Member Function Documentation

#### 3.2.3.1 ProcessInput()

```
bool Platform::ProcessInput (
    uint8_t * keys )
```

Processes user input and updates the CHIP-8 keypad state.

##### Parameters

<i>keys</i>	Pointer to the CHIP-8 keypad state array (16 keys).
-------------	-----------------------------------------------------

##### Returns

True if the user wants to quit, otherwise false.

#### 3.2.3.2 Update()

```
void Platform::Update (
    const void * buffer,
    int pitch )
```

Updates the display with new pixel data.

##### Parameters

<i>buffer</i>	A pointer to the pixel data buffer.
<i>pitch</i>	The number of bytes per row in the buffer.

### 3.2.4 Friends And Related Function Documentation

#### 3.2.4.1 TestPlatform

```
friend class TestPlatform [friend]
```

Allows unit tests to access private members.

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

- [/home/zk/Git\\_Repo/CHIP-8\\_emulator/include/Platform.h](#)
- [/home/zk/Git\\_Repo/CHIP-8\\_emulator/src/platform/Platform.cpp](#)

## Chapter 4

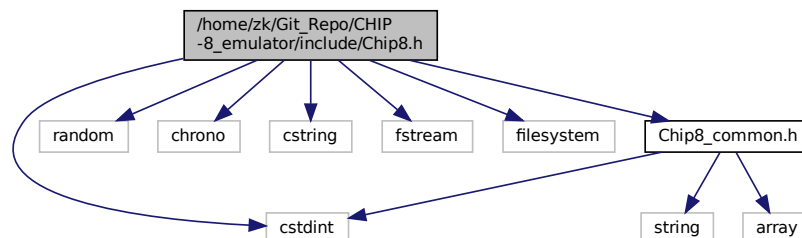
# File Documentation

### 4.1 /home/zk/Git\_Repo/CHIP-8\_emulator/include/Chip8.h File Reference

CHIP-8 Emulator Class.

```
#include <cstdint>
#include <random>
#include <chrono>
#include <cstring>
#include <fstream>
#include <filesystem>
#include "Chip8_common.h"
```

Include dependency graph for Chip8.h:



## Classes

- class [Chip8](#)  
*CHIP-8 Emulator.*

### 4.1.1 Detailed Description

CHIP-8 Emulator Class.

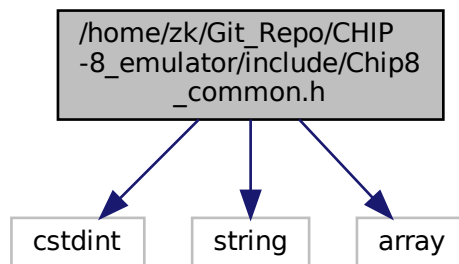
This file contains the declaration of the [Chip8](#) class, which emulates a CHIP-8 system. The class provides functionality to load ROMs, execute instructions, and manage memory, registers, and timers.

## 4.2 /home/zk/Git\_Repo/CHIP-8\_emulator/include/Chip8\_common.h File Reference

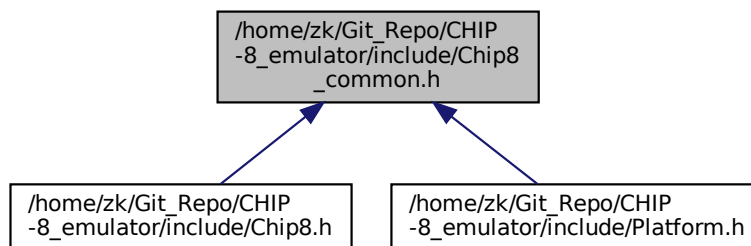
Constant values, useful for building the emulator.

```
#include <stdint>
#include <string>
#include <array>
```

Include dependency graph for Chip8\_common.h:



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



### Variables

- const int [DEFAULT\\_VIDEO\\_SCALE](#) = 10  
*Default video scaling factor for rendering.*
- const int [DEFAULT\\_CYCLE\\_DELAY](#) = 1  
*Default cycle delay for the emulator (in milliseconds).*
- const unsigned int [START\\_ADDRESS](#) = 0x200  
*Start address in memory where CHIP-8 programs are loaded.*
- const unsigned int [FONTSET\\_START\\_ADDRESS](#) = 0x50

- *Memory address where the CHIP-8 fontset starts.*
- const unsigned int `FONT_SIZE` = 5  
*Size (in bytes) of each character in the fontset.*
- const unsigned int `MEMORY_SIZE` = 4096  
*Total size of CHIP-8 memory (4KB).*
- const unsigned int `VIDEO_WIDTH` = 64  
*Width of the CHIP-8 display (in pixels).*
- const unsigned int `VIDEO_HEIGHT` = 32  
*Height of the CHIP-8 display (in pixels).*
- const unsigned int `FONTSET_SIZE` = 80  
*Total size of the CHIP-8 fontset (in bytes).*
- constexpr std::array< uint8\_t, `FONTSET_SIZE` > `fontset`  
*CHIP-8 built-in fontset.*
- const unsigned int `NUM_REGISTERS` = 16  
*Number of general-purpose registers (V0-VF).*
- const unsigned int `STACK_SIZE` = 16  
*Stack size for subroutine calls (16 levels).*
- const unsigned int `NUM_KEYS` = 16  
*Number of keys in the CHIP-8 hexadecimal keypad (0-F).*

### 4.2.1 Detailed Description

Constant values, useful for building the emulator.

This file contains the declaration of the `Chip8` global values, to build the CHIP-8 system.

### 4.2.2 Variable Documentation

#### 4.2.2.1 fontset

```
constexpr std::array<uint8_t, FONTSET_SIZE> fontset [constexpr]
```

**Initial value:**

```
= {
    0xF0, 0x90, 0x90, 0x90, 0xF0,
    0x20, 0x60, 0x20, 0x20, 0x70,
    0xF0, 0x10, 0xF0, 0x80, 0xF0,
    0xF0, 0x10, 0xF0, 0x10, 0xF0,
    0x90, 0x90, 0xF0, 0x10, 0x10,
    0xF0, 0x80, 0xF0, 0x10, 0xF0,
    0xF0, 0x80, 0xF0, 0x90, 0xF0,
    0xF0, 0x10, 0x20, 0x40, 0x40,
    0xF0, 0x90, 0xF0, 0x90, 0xF0,
    0xF0, 0x90, 0xF0, 0x10, 0xF0,
    0xF0, 0x90, 0xF0, 0x90, 0x90,
    0xE0, 0x90, 0xE0, 0x90, 0xE0,
    0xF0, 0x80, 0x80, 0x80, 0xF0,
    0xE0, 0x90, 0x90, 0x90, 0xE0,
    0xF0, 0x80, 0xF0, 0x80, 0xF0,
    0xF0, 0x80, 0xF0, 0x80, 0x80
}
```

CHIP-8 built-in fontset.

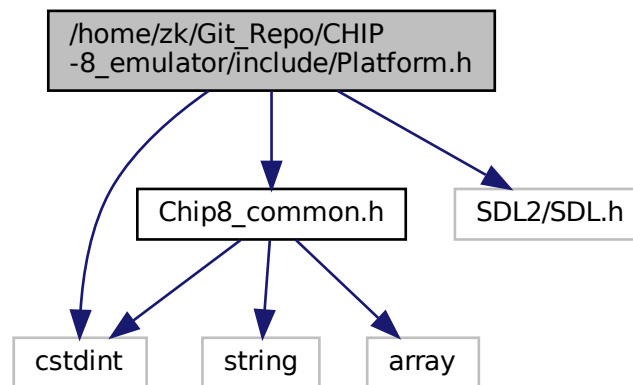
This array stores the 16 hexadecimal characters (0-F) used for rendering.

### 4.3 /home/zk/Git\_Repo/CHIP-8\_emulator/include/Platform.h File Reference

[Platform](#) and input output handling Class.

```
#include "Chip8_common.h"
#include <stdint>
#include <SDL2/SDL.h>
```

Include dependency graph for Platform.h:



#### Classes

- class [Platform](#)

*Handles the window, rendering, and input for the CHIP-8 emulator using SDL.*

#### 4.3.1 Detailed Description

[Platform](#) and input output handling Class.

This file contains the declaration of the [Platform](#) class, which emulates the keypad and the screen. The class provides functionality to update the screen and the keys input.