The

# VIC Chip-8 Emulator

**for the Commodore VIC20 personal computer with 16k memory expansion**

**Version 1.11**

*Eximietas Software*

***The RCA Cosmac VIP was one of the first computer systems to utilize the Chip-8 virtual instruction set***

# Table of contents
-----------------

## Contents

# 1. Introduction

------------------

Chip-8 is a virtual machine designed in the mid-1970s by Joseph Weisbecker for use with the COSMAC VIP and Telmac 1800 microcomputers. The purpose of the framework was to simplify the programming of games on these computers. Weisbecker designed the system to be (1) a register machine with (2) a limited instruction set and (3) with fixed width instructions. These three characteristics are in line with what today would be called a virtual RISC machine.

Weisbecker not only designed this virtual machine but was also responsible for the design of the RCA 1802 microprocessor. Although this microprocessor did not enjoy the same level of commercial success as its rivals such as the 6502, Z80 and 8080, the processor carved its own niche in the aerospace industry. A special radiation hardened version of the processor has been and is still used in space exploration and satellite projects. Weisbecker remains an under celebrated personality in the history of the microcomputer revolution.

Returning to the history of Chip-8: The programming system garnered a small enthusiastic following, however as new micro-computers entered the commercial market and new features such as color, advanced sound, disk drives and more memory was added – the Chip-8 system slowly became forgotten.

The situation changed in the early 1990s when Hewlett Packard introduced their new range of HP-48 programmable scientific calculators. These included a Chip-8 interpreter to facilitate easier game programming. This version was written by Andreas Gustaffson and is called CHIP-48.

Then an enhanced CHIP-8 instruction set was written by Eric Brynste with new capabilities such as doubling the screen resolution (from the 64x32 of the original to 128x64) and adding instructions for scrolling and so on. This variant was called the SCHIP or Superchip.

Because of the relative simplicity of the system, it was easy to port to almost any computational platform with basic display

**capabilities. Today a multitude of ports for a multitude of platforms exist.**

**More recently a new variant has become popular called the XO-CHIP (by John Earnest) which enhances the system even further.**

**The eXimietas VIC-20 Chip-8 emulator can run original Chip-8 programs as well as programs which use the altered HP-48 instructions. However it does not support the *additional* instructions that the SCHIP introduced. Therefore it is not SCHIP compatible and also not XO-CHIP compatible. In short this emulator is an essential or basic Chip-8 emulator.**

**The purpose of this software is twofold:**

1. **To allow the Commodore VIC-20 community to gain access to the library of Chip-8 software titles that has been built up over the decades.**
2. **To allow any VIC enthusiast that would write software for the Chip-8 platform, to run that software on their favorite Commodore computer.**

***The Hewlett Packard series 48 programmable calculators revived the Chip-8 instruction set***

## 2. Quick Start

----------------

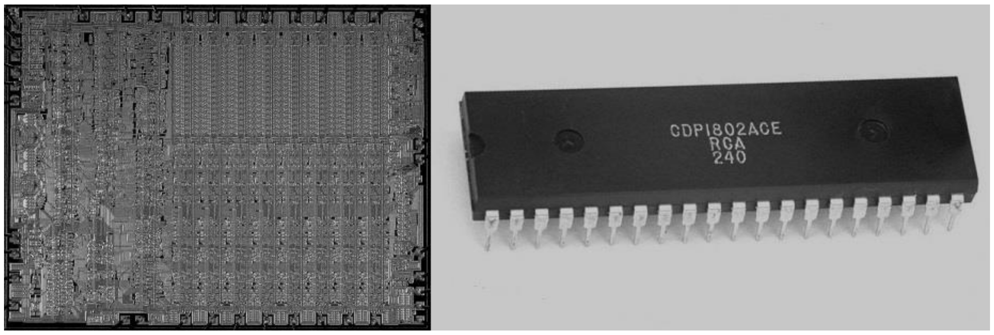
**Requirements:**
**- A Commodore Vic-20 microcomputer with 16kb expansion**
**- A Commodore 1541 disk drive or equivalent**
**- Joystick and keyboard**

**To start the emulator:**
**Insert the Chip-8 program disk into the disk drive.**
**Type: load "*",8,1 and press the return key.**
**Then type: run and then press return.**
**The Chip-8 emulator should now start.**

**Now insert a Chip-8 games disk into the disk drive.**
**Use the joystick to navigate to 'Load game'.**
**The disk will then show the game catalog.**
**Select the desired game and press the fire button.**
**The game will load and you will be returned to the main screen.**
**Select 'start game' and press fire.**

**Some older Chip-8 games will automatically start and do not have a restart function. For these games and for others that may have issues there is a restart button, the F1 key. Pressing F1 will clear all registers, counters and restart the program.**

**The F7 key will exit the program and return to the menu. It also clears the Chip-8 memory upon exit.**



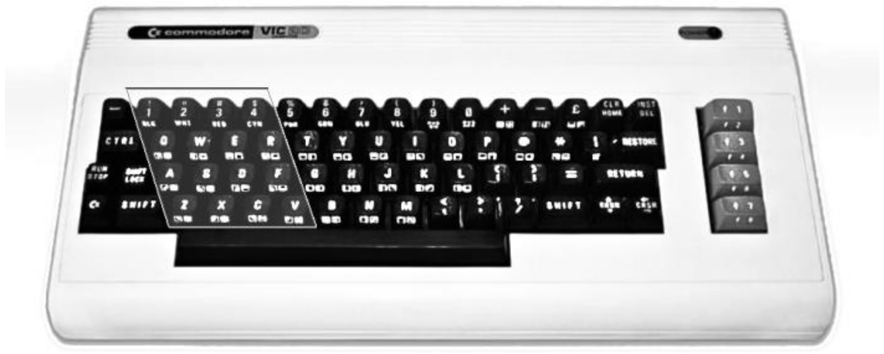
***The RCA 1802 die and the finished CPU***

## 3. Key mapping

----------------

**The COSMAC VIP microcomputer has a keypad with 16 keys which represent the hexadecimal number system, in other words the numbers 0 to 9 and the letters A to F (representing 10 to 15). This allows a programmer to enter machine language directly into the machine. This keypad is also used as the input device for games. Different programs use different keys based upon a programmer's preference.**

**The eXimietas VIC-20 Chip-8 emulator maps these keys to the following Commodore keys by default:**



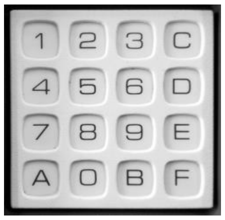
**1,2,3,4,Q,W,E,R,A,S,D,F,Z,X,C,V map to the COSMAC keys 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F.**

**These keys are reconfigurable. The following Commodore keys can be used for mapping: 0-9, A-Z and the space bar.**

**The COSMAC keys can also be mapped to the joystick. For each game one can either use the keyboard or the joystick, but one cannot use both at the same time.**

**Most of the games included with this emulator have been mapped to use the joystick.**

**On the main emulator menu use the "view/assign keys" option to reassign values to either the joystick or the keyboard.**



## 4. Adding software titles

-----------------------------

**To add a new software title to the collection one cannot only copy the Chip-8 file to a floppy disk and immediately start to use it on the emulator. Three things need to be done:**

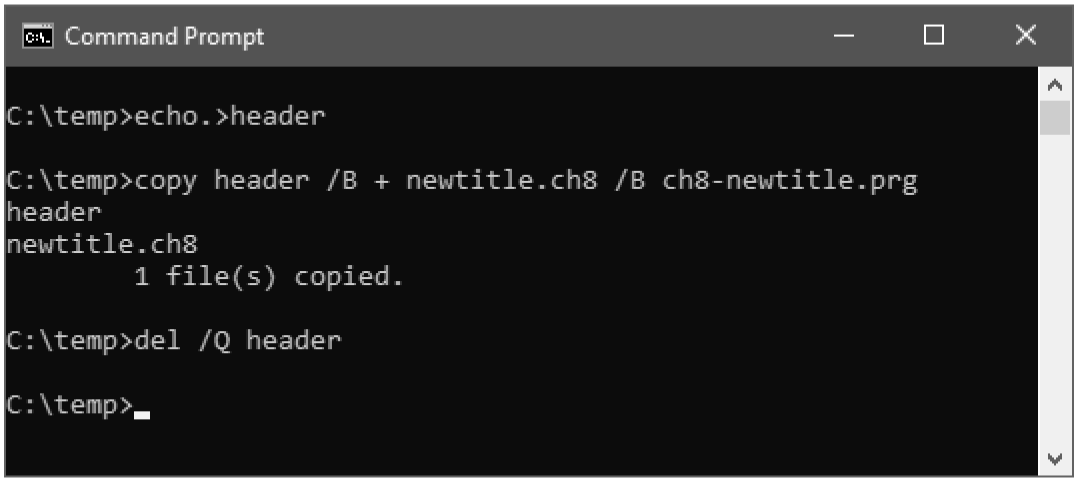
1. **Add a two byte header to the file**
2. **Rename the file appropriately**
3. **Apply appropriate emulator settings to the title**

### Adding a two byte file header

-----------------------------

**Adding a two byte header (the bytes can have any value) is most easily done before the program file is copied to the floppy. One can either use OS command line tools or HEX file editor software.**

**The following example shows how a two byte header can be added to a Chip-8 file from the Microsoft Windows command line. The example is non-destructive and creates a new file while leaving the original file intact:**



```
C:\temp>echo.>header

C:\temp>copy header /B + newtitle.ch8 /B ch8-newtitle.prg
header
newtitle.ch8
        1 file(s) copied.

C:\temp>del /Q header

C:\temp>
```

**Other modern operating systems also have command line tools that can accomplish a similar task to the example shown above. (See your OS documentation)**

**Correct Chip-8 filename format for use with this emulator**
**----------------------------------------------------------**

**For the emulator to be able to load a new Chip-8 file the filename format needs to be correct. The format has the following specification:**

**A four byte character header with the format: ch8-**

**This indicates to the emulator that this is a Chip-8 file. Then the following up to 12 characters of the filename should only contain: lower case characters, numeric characters, spaces or dashes.**

**One can use upper case and non-standard characters but if the emulator does not have the character in its character set, it will display a placeholder character instead.**

**Emulator settings for each game/program**
**----------------------------------------**

**Each loaded game can be customized from the emulator's main menu.**

**'view/assign keys' allows each game to have its own key/joystick mappings.**

**'change settings' allows each game to have its own cosmetic look, feel and sound.**

**'change settings' -> 'chip-8 adv cpu' allows each game to use the instruction set interpretation they were intended for. There are five main instruction variances and the defaults will generally work with modern Chip-8 games.**

**'change settings' -> 'chip-8 adv cpu' -> 'cpu delay'**
**Some games are just unplayable they are so fast. This is usually due to the fact that a game was programmed on a Chip-8 platform which has a lower execution speed. To compensate for this each instruction can be delayed by an arbitrary delay amount. The value ranges from 0 to 7 and testing is the best way to achieve the desired play speed.**

## **Saving game settings for each game**

**----------------------------------**

**On the main emulator menu select 'save settings' to save the current loaded game configuration to disk. It saves the key mapping, settings and advanced CPU settings to a file linked to the game. The settings file has the prefix 'cfg-' and then the game name (with the ch8- prefix now stripped off). When a new game is saved for the first time the configuration file is created and when one selects save settings afterwards the config file gets overwritten. This allows for configurations to be changed according to personal preference.**

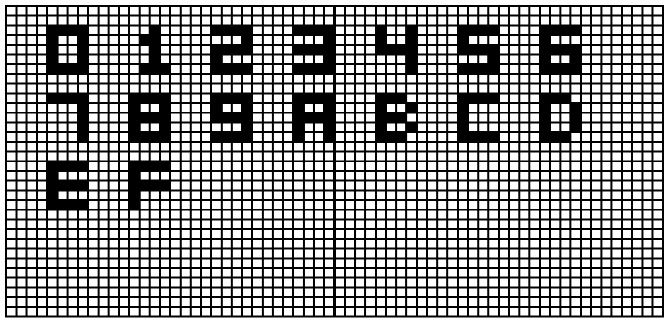
**oooOOOooo**

***Jean, Joyce and Joseph Weisbecker***

***Joyce has been credited with being the first female commercial computer game programmer***

## Appendix A: Specifications

| Original CHIP-8 specification |
|---|
| Memory: Direct access to 4 kilobytes of RAM<br>Display: 64 x 32 pixel monochromatic bitmap |
| 16 x General-purpose registers - 08 bits wide - numbered 0 through F (V0 - VF) |
| VF is the flag register - instructions set VF's value to 1 or 0 based on some rule, for example using it as a carry flag. The register is also general purpose. |
| Index register (I) - 16 bits wide - pointer to a location in memory |
| Delay timer - 08 bits wide - decremented at a rate of 50 or 60 Hz (depending on the base clock) until it reaches 0 |
| Sound timer - 08 bits wide - functions like the delay timer, also emits a beeping sound as long as the timer is not 0 |
| Program counter (PC) - 16 bits wide - points to the current instruction in memory |
| Stack - 16 entries deep and 16-bits wide - used to call subroutines and to return from them |



*64 x 32 bitmap display demonstrating the built in font (See memory map)*

| Contemporary CHIP-8 memory map | |
|---|---|
| $0000<br>Decimal: 0000<br>Start of usable memory. It has become customary to leave the first 80 bytes alone. This may be due to early programs using this area as scratch memory. | |
| $0050<br>Decimal: 0080<br>Contemporary Chip-8 designs place the system font at location $0050. The font comprises of 16 characters of 5 bytes each (80 bytes in total). Each character is 4 x 5 pixels and only the high nibble for each byte is used. | 512 bytes reserved system memory |
| $00A0<br>Decimal: 0160<br>Unused space. Traditionally the first 512 bytes were used for the Chip-8 interpreter. On this VIC emulator this area is being used by system routines. | |
| $0200 *- $1000<br>Decimal: 0512 – 4096<br><br>Chip-8 application memory | 3.5kbyte usable memory |

* The PC (program counter) is set to this address at the start of program execution

## THE ORIGINAL CHIP-8 MACHINE LANGUAGE INSTRUCTION SET

| Two byte Code | Description |
|---|---|
| 00E0 | Clear screen |
| 00EE | Return from Subroutine |
| 1nnn | Jump to nnn |
| 2nnn | Call subroutine at nnn |
| 3Xnn | Skip one Instruction if VX=nn |
| 4Xnn | Skip one Instruction if VX is not equal to nn |
| 5XY0 | Skip one Instruction if VX=VY |
| 6Xnn | Assign value nn to VX |
| 7Xnn | Add nn to VX |
| 8XY0 | Copy VY to VX (VX is set to the value of VY) |
| 8XY1 | OR VX with VY |
| 8XY2 | AND VX with VY |
| 8XY3 | XOR VX with VY |
| 8XY4 | Add VY to VX. If result >FF, then VF=1. |
| 8XY5 | Sets VX to the result of VX - VY. If VX<VY, then VF=0 else VF=1 |
| 8XY6 | VY to VX, then shift VX one bit right, store shifted bit in VF |
| 8XY7 | Sets VX to the result of VY - VX. If VY<VX, then VF=0 else VF=1 |
| 8XYE | VY to VX, then shift VX one bit left, store shifted bit in VF |
| 9XY0 | Skip one instruction if VX does not equal VY |
| Annn | Set memory Index Register to nnn |
| Bnnn | Jump to location nnn+V0 |
| CXnn | Get random byte, then AND with nn then store in VX |
| DXYn | Draw n-byte sprite at VX,VY. Sprite location pointed to by I |
| EX9E | Skip one instruction if keypress=VX, do not wait |
| EXA1 | Skip one instruction if keypress does not equal VX, do not wait |
| FX07 | Store timer value in VX |
| FX0A | Wait for keypress, when pressed store value in VX and continue execution |
| FX15 | Initialize timer. 01=20 mS or 17 ms, depending whether base clock is 50 or 60hz |
| FX18 | Sound tone for timer interval multiplied by VX |
| FX1E | Add VX to index register (I) |
| FX29 | Set I (index register) to address of system font (16 sprites) indicated by VX (0-F - low nibble) |
| FX33 | Store 3 digit decimal equivalent of VX at address I, I+1, I+2 |
| FX55 | Store V0 through VX at location pointed to by I. Endstate: I=I+X+1 |
| FX65 | Load V0 through VX at location pointed to by I. Endstate: I=I+X+1 |

**Note: Instructions Bnnn, 8XY6, 8XYE, FX55 and FX65 are the instructions that have the HP-48 variations**

## MACHINE LANGUAGE INSTRUCTIONS FEATURED IN ORIGINAL CONFIGURATIONS BUT THAT ARE RARE IN MODERN CONFIGURATIONS

| Two byte Code | Description |
|---|---|
| 0000 | No Operation. In this VIC-20 version this instruction results in a jmp * |
| F000 | Jump to Monitor (CHIPOS)<br>Not implementred in this VIC-20 version |
| FX17 | Set the Pitch of the Tone Generator to VX<br>Not implemented in this VIC-20 version |
| FX70 | Send data in VX to RS485 Port<br>Not implemented in this VIC-20 version<br>It has to be noted that using these instructions for serial communications would allow for the implementation of networked Chip-8 games for the VIC-20. Something to consider in future versions of the Vic-20 software. |
| FX71 | Waits for received RS485 data. Place in VX<br>Not implemented in this VIC-20 version |
| FX72 | Set RS485 Baud rate<br>Not implemented in thisVIC-20 version |

## Appendix B: Future additions and bug reporting

-------------------------------------------------------

- **Optimization of the Chip-8 engine.**
    - **Moving the general purpose registers, Index register, Program counter and the Stack to the first 256 bytes of Commodore memory to take advantage of the instruction cycle savings.**
    - **Optimize the display routine (A lot!)**
- **A machine code monitor within the emulator**
- **If there is enough interest – A 'Chip-8 to Commodore' executable program wrapper so that programmers can turn their Chip-8 creations into VIC-20 standalone applications.**
- **A turbo disk loader for the emulator.**
- **Add the lesser implemented Chip-8 serial port instructions so that networked VIC-20 Chip-8 games can be programmed.**
- **BUG FIXES of the emulators Chip-8 instruction code: I know that they definitely must exist! So expect version updates v1.12 and so on.**

**If you have found a bug in the software you can contact me on the VIC Denial forum, my user handle is Huffelduff.**

**oooOOOooo**

**Acknowledgements**

**----------------**

**The default program that is in the Chip-8 emulator memory is an opcode test program by a programmer with the moniker Corax89.**

**The program can be found here:**
**https://github.com/corax89/chip8-test-rom**