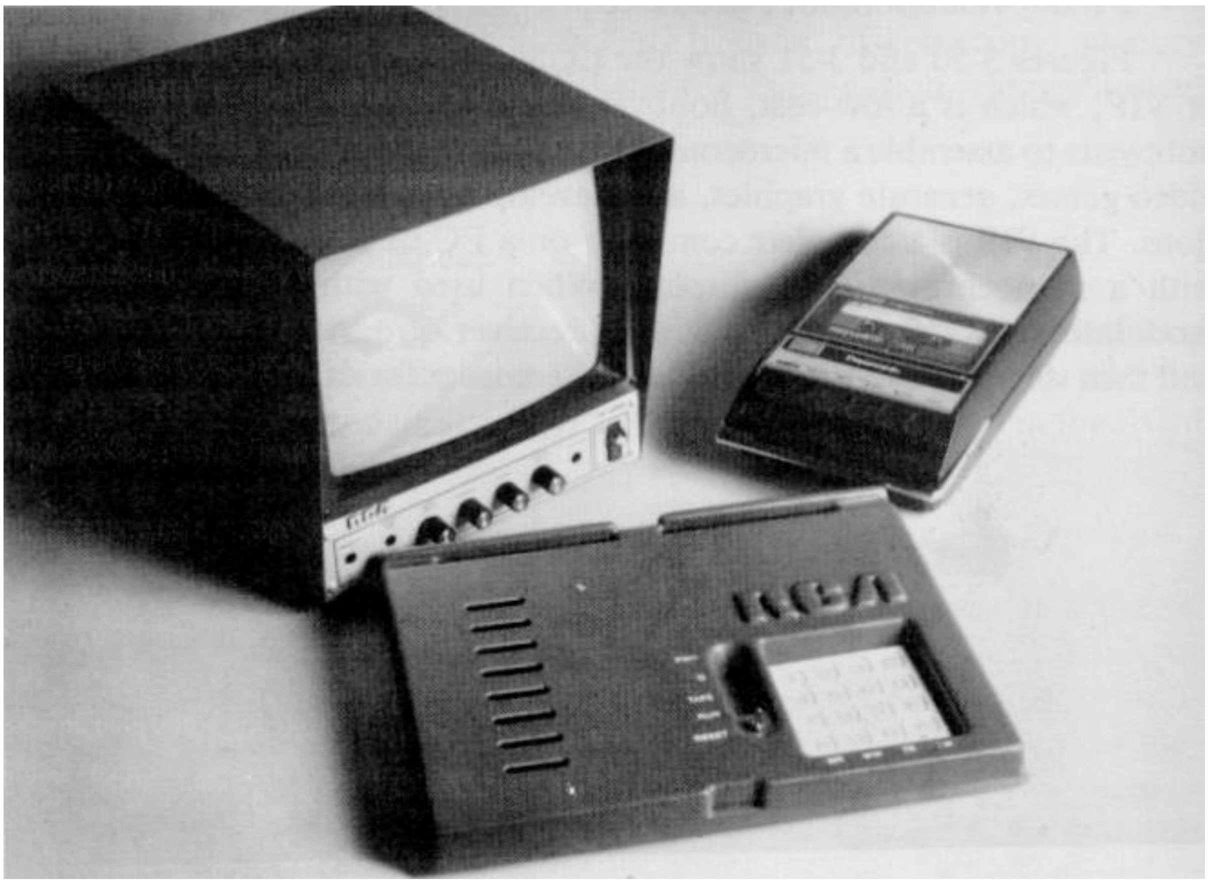
CHIP8 documentation

David WINTER (HPMANIAC)



1 - CHIP-8 features

1.1 - History

1.1.1 - The original CHIP-8

CHIP-8 is a language interpreter which was used in the late 70's and early 80's on some small commercial computers like RCA's TELMAC 1800 and COSMAC VIP, and these El-Cheapo "Make-It-Yourself" Hobbyist computers of these times like the ETI 660 and the DREAM 6800...

CHIP-8 allowed to program video games easily. The TELMAC 1800 and the COSMAC VIP were based on RCA's CDP-1802 processor. Both came with an audiocassette containing more than 12 games, dated 1977 (the complete listings of these programs, as well as these of the interpreter and the computer ROM were listed in the manuals of the COSMAC and the TELMAC). This interpreter has less than 40 instructions including arithmetic, control flow, graphics and sound.

The interpreter had to be very compact because of the memory limitation of these machines: the COSMAC VIP had 2Kb (although it could be expanded to 32Kb), and the TELMAC had 4Kb. CHIP-8 was only 512 bytes long.

The simplicity of this early language allowed to program these Pong, Brix, Invaders and Tank games we used to see at this early time of the videogame. A good programmer could make these games fit into less than 256 bytes.

Here is a short story about a CHIP-8 user on a DREAM-6800:

"...the DREAM and ETI 660 both appeared in Australian Electronics Magazines as construction projects. What all these computers had in common was that they were disgustingly cheap (about $100), used a hex keypad, could produce ultra stingy 64 x 32 PIXEL (The ETI 660 had 64 x 48 OR 64 x 64 with a modification) graphics for display on a TV, had about ONE kilobyte of RAM, and all ran a pseudo high-level language called CHIP-8 (which was developed by RCA for showing off the COSMAC's graphics, I think).

…

Somewhere along the way, my older brother made up a DREAM 6800. What a computer! Along with the construction articles for the DREAM & ETI 660 were heaps of CHIP-8 game listings. Some of the games were only 200 BYTES or so, so it didn't take forever to type them in. And the games were great fun. They weren't slow. And CHIP-8 was pretty much designed for making Classic style TV games anyway."

Paul HAYTER (Author of a CHIP-8 interpreter on the Amiga)

1.1.2 - CHIP-8 today

CHIP-8 was not only used in the late 70's and early 80's. It was used in the early 90's on the HP48 calculator because there was no programming tools to develop fast games on it. Most of the original CHIP-8 games work with the CHIP48 interpreter, and some new ones were programmed.

Then, a better version of CHIP-8 appeared: SUPER-CHIP. This interpreter has all the CHIP-8 features, as well as and some new ones like a 128*64 resolution.

1.2 - CHIP8 programs and memory

All the CHIP-8 programs start at address 200h (except those of the ETI-660, which start at the non-standard address of 600h). This is due to the interpreter, which used to reside in the 000h-1FFh area on the Telmac and the COSMAC VIP.

The entire memory is accessible and byte addressable. As the instructions are 16 bits long, their addresses are usually even (if some 8-bit data are inserted into the code, the instructions may become odd-addressed).

SCHIP programs run like CHIP-8 programs. The only differences between them are some new instructions in the SCHIP mode. Some of the new instructions work under both CHIP-8 and SCHIP modes. Some others work only in SCHIP mode.

1.3 - Registers

The data registers:

They are 16, all 8 bits wide, and named V0...VF. VF is used as carry (when using arithmetic instructions) and collision detector (when drawing sprites). Refer to the instructions section concerning the collisions in SCHIP mode.

The address register:

There is only one named I, 16 bits wide. As the memory is 4Kb, the interpreter uses only its 12 low bits. The remaining 4 could be set using the LOAD FONT instruction as the font was located at address 8110.

The timers:

There are two timers. One is the delay timer, and the other the sound timer. Both are 8 bits wide and down-count about 60 times per second when non- zero. The speaker will beep while the sound timer is non zero. The delay timer is generally used to make delay loops.

The stack:

It has 16 levels, allowing 16 successive subroutine calls. This may not apply to the original CHIP-8 as no documentation was found on the stack.

1.4 - Graphics

The original resolution of CHIP-8 is 64 x 32 pixels. Some modified machines could have a 64 x 48 or 64 x 64 resolution, and the TELMAC also had a second version of CHIP-8 called CHIP-82 that used a 64 x 64 resolution. As no program was found using the extended ones, CHIP8 will only use the 64 x 32 resolution.

Graphics are drawn as 8 x 1...15 sprites (they are byte coded). The origin of the screen is the upper left corner. All the coordinates are positive, start at 0, and are calculated modulo 64 for X, and 32 for Y when drawing sprites.

All drawings are done in XOR mode. When one or more pixels are erased while a sprite is drawn, the VF register is set to 01, otherwise 00.

CHIP8 has a 4 x 5 pixels hexadecimal font to draw characters. These ones are 0-9 and A-F.

The SCHIP mode is an extended CHIP-8 mode. It provides an extended graphic resolution of 128 x 64 pixels. When activated, pixels coordinates ranges are 00h-7Fh for X (0-127), 00h-3Fh for Y (0-63) and are calculated modulo 128 for X and modulo 64 for Y. It is important to note that a pixel of the 64 x 32 resolution will appear twice bigger than one of the extended resolution.

The SCHIP mode provides an 8 x 10 decimal character font, and a 16 x 16 sprite. Their drawing modes are the same than in CHIP-8. Both of the fonts are usable in 64 x 32 and 128 x 64 resolutions. Depending on the resolution used, the size of the characters will change because of the size of the pixels...

1.5 - Instructions

NNN is an address
KK is an 8 bit constant
X and Y are two 4 bits constants

0NNN	Call 1802 machine code program at NNN (not implemented)
00CN	Scroll down N lines (***)
00FB	Scroll 4 pixels right (***)
00FC	Scroll 4 pixels left (***)
00FD	Quit the emulator (***)
00FE	Set CHIP-8 graphic mode (***)
00FF	Set SCHIP graphic mode (***)
00E0	Erase the screen
00EE	Return from a CHIP-8 sub-routine
1NNN	Jump to NNN
2NNN	Call CHIP-8 sub-routine at NNN (16 successive calls max)
3XKK	Skip next instruction if VX == KK
4XKK	Skip next instruction if VX != KK
5XY0	Skip next instruction if VX == VY
6XKK	VX = KK
7XKK	VX = VX + KK
8XY0	VX = VY
8XY1	VX = VX OR VY
8XY2	VX = VX AND VY
8XY3	VX = VX XOR VY (*)
8XY4	VX = VX + VY, VF = carry
8XY5	VX = VX - VY, VF = not borrow (**)
8XY6	VX = VX SHR 1 (VX=VX/2), VF = carry
8XY7	VX = VY - VX, VF = not borrow (*) (**)
8XYE	VX = VX SHL 1 (VX=VX*2), VF = carry
9XY0	Skip next instruction if VX != VY
ANNN	I = NNN
BNNN	Jump to NNN + V0
CXKK	VX = Random number AND KK
DXYN	Draws a sprite at (VX,VY) starting at M(I). VF = collision. If N=0, draws the 16 x 16 sprite, else an 8 x N sprite.
EX9E	Skip next instruction if key VX pressed
EXA1	Skip next instruction if key VX not pressed
FX07	VX = Delay timer
FX0A	Waits a keypress and stores it in VX
FX15	Delay timer = VX

FX18	Sound timer = VX
FX1E	I = I + VX
FX29	I points to the 4 x 5 font sprite of hex char in VX
FX33	Store BCD representation of VX in M(I)...M(I+2)
FX55	Save V0...VX in memory starting at M(I)
FX65	Load V0...VX from memory starting at M(I)
FX75	Save V0...VX (X<8) in the HP48 flags (***)
FX85	Load V0...VX (X<8) from the HP48 flags (***)

(*): Used to be undocumented (but functional) in the original docs.
(**): When you do VX - VY, VF is set to the negation of the borrow. This means that if VX is superior or equal to VY, VF will be set to 01, as the borrow is 0. If VX is inferior to VY, VF is set to 00, as the borrow is 1.
(***): SCHIP Instruction. Can be used in CHIP8 graphic mode.

NOTES:
As the interpreter is emulated, all the 0NNN instructions cannot be implemented. Only 00E0, 00EE and the SCHIP instructions are available.

The SCHIP graphic instructions can be used in CHIP-8 graphic mode. This, a 4 pixels left or right scrolling in SCHIP graphic mode will be interpreted as a TWO PIXELS scroll in CHIP-8 mode. Remember that a segment of 4 pixels in 128 x 64 resolution has the same size than a 2 pixels one in 64 x 32 resolution.

Drawing the 16 x 16 pixels SCHIP sprite in CHIP-8 mode will display an 8 x 16 sprite (because only the first 16 bytes of this sprite will be used). The result is that a 16 x 16 SCHIP sprite will not be correctly displayed in the 64 x 32 resolution. However, this instruction allows drawing an 8 x 16 sprite, which cannot be performed with the standard CHIP-8 drawing instruction.

We saw that a pixel of the 64 x 32 resolution is twice bigger than one of the 128 x 64 one. Another consequence of this is that the vertical scrolling instruction is different in the 64 x 32 resolution. In this one, it will scroll half the lines it would have scrolled in SCHIP mode. Note that if the number of lines to scroll is ODD, the scroll will be performed with a half-pixel shift !

1.6 - Keyboard

Most of the original CHIP-8 programs used a 16 key hex keyboard, which looked like this:

1	2	3	C
4	5	6	D
7	8	9	E
A	0	B	F