VCC Debugger

# Overview

The VCC Debugger is your window into the internals of the Color Computer 3 (CoCo3). It shows you in real-time what the processor, memory, and memory management unit (MMU) is doing at any given moment. It is made up of four windows:

* **Memory Display** – this displays the 64KB memory space that is visible to the processor. Even though the CoCo3 has a minimum of 128KB of RAM, only 64KB can be seen by the processor. The MMU is responsible for mapping the extra RAM into this 64KB space.
* **Processor State** – this displays the current state of the processor. Internal registers, Condition Code register, stack/user pointers, and the program counter are updated in real time. In addition, you can halt, step, and run the processor at any time.
* **Breakpoints** – this window allows you to load an LWASM assembler listing and set breakpoints. This provides a source level debugging capability for a loaded program.
* **MMU Monitor** – this window will let you view all memory available to the machine and how it is being mapped into the 64K memory space visible to the processor.

# Debugger Menu

All debugger windows are accessed through the **Debugger** option in the main menu.

Qr code

Description automatically generated

Selecting one of these options will launch the corresponding window.

**Note:** Interrogating the machine’s internals and displaying the results in real time *may*impact the frame rate of the emulator. On reasonably fast machines this will not be an issue. However, if you are playing a graphic intensive game and want to maintain 60 FPS, keep the debugger windows closed until you need them. When the windows are closed, the emulator’s framerate are not impacted.

# Memory Display

Table

Description automatically generatedSelecting the memory display option will present the 64KB processor space. All values in the memory display are updated in real time.

Press this to close the window

You can enter a hexadecimal address here and the display will scroll to that address.

If the character can be displayed it will by shown here, otherwise it will be shown as a “.”

Any address from 0000 to FFFF will be displayed

# Processor State

Selecting the processor state option will show the internals of the CPU. All values in the display are updated in real time.

CPU Program Counter is displayed here.

Processor Registers are shown here

Graphical user interface, text, application

Description automatically generated with medium confidence

Processor Condition Codes:

**E**: All registers were stacked from last interrupt.

**F**: Fast interrupt (FIRQ) disable flag.

**H**: Half carry flag.

**I**: Interrupt (IRQ) disable flag.

**N**: Last operation was negative.

**Z**: Last operation was zero.

**V**: Last operation had an overflow.

**C**: Full carry flag.

Processor Controls:

**Halt**: Stops the processor from running. All displays will freeze with the state of the machine at the time the button is pressed.

**Run**: Starts the processor running again. The machine will resume exactly where it left off when it was halted.

**Step**: Cycles the processor by one step. It will execute the next instruction and halt again.

Press this to close the window

# Breakpoints

Selecting the breakpoints option will allow you to load an LWASM source listing and set breakpoints.

Graphical user interface

Description automatically generated

**Breakpoint Controls**

**New Breakpoint:** To set a breakpoint, click on a source line below, and then click on New Breakpoint. You will only be able to set a breakpoint on a line with an address. All addresses will appear in the second column of the listing.

When a breakpoint is set, the first column will indicate which breakpoint has been set.

**Remove Breakpoint:** To remove a breakpoint, click on the breakpoint listed *in the breakpoint control window.*The source will scroll to the breakpoint. Click on the Remove Breakpoint button and the breakpoint will be removed.

**On/Off:** You can enable or disable a breakpoint by clicking on a breakpoint listed in the breakpoint control window.Clicking on the Off button will disable the breakpoint. A disabled breakpoint will remain listed in the source, but the CPU will not halt when it reaches the instruction. You can re-enable the breakpoint by clicking the On button.

Select the LWASM Source Listing

**1st Column:** Indicates set breakpoints.

**2nd Column:** Memory Address – you can only set breakpoints on lines with an address.

**3rd Column:** Bytes in memory, this could be CPU instructions or data.

You can use this field to search the source for text. Pressing the find button repeatedly will move to the next location where the text is found.

## Source Listing

The Source Debugger can only deal with certain source listings. Currently the breakpoints window will only accept **LWASM Source Listings**. To create such as listing, you must specific this option on the LWASM command line.

Here is an example command line:

lwasm --format=decb --list=sr3.lst --output=sr3.bin sr3.asm

This produces a listing file like this:

6000 1A50 ( sr3.asm):00047 START ORCC #$50 \* Turn off interrupts

6002 7FFF40 ( sr3.asm):00048 CLR $FF40

6005 7FFFDE ( sr3.asm):00049 CLR $FFDE

6008 FC0112 ( sr3.asm):00050 LDD $112

600B FD64CC ( sr3.asm):00051 STD RNUM

600E FD64CE ( sr3.asm):00052 STD RNUM+2

6011 8655 ( sr3.asm):00053 LDA #$55

6013 9771 ( sr3.asm):00054 STA <$71

6015 8E601D ( sr3.asm):00055 LDX #STRT

6018 9F72 ( sr3.asm):00056 STX <$72

601A 171560 ( sr3.asm):00057 LBSR HNAM

When the listing is loaded, it scans the file looking for the 4-byte address (6000 in the case of the first line). When it finds a line with an address it will make note of the listing line number. When a breakpoint address is sent to the CPU and the CPU’s Program Counter (PC) matches the breakpoint address, the CPU will halt.

When the CPU halts, the breakpoint window will scan the source listing and locate the source line with the matching address. This allows you to see the source exactly as it appears in the listing including any comments.

**Note:** The source listing is used as the only means of source debugging. No disassembly of machine code is performed. This means if the program uses self-modifying code or data as instructions, the source listing window may not reflect exactly what the machine is executing. The memory window will contain the exact bytes seen by the processor, but the source listing may not. However, the breakpoint window will try its best to stay relevant to the area of the code being executed.

# MMU Monitor

Selecting the MMU Monitor option will allow you to see the current state of the Memory Management Unit (MMU). This device handles the mapping of the machine’s real memory (128KB or 512KB in most CoCo3s) to the CPU’s 64KB memory space.

On a CoCo3, the MMU has two sets of mappings of real memory to CPU memory: MAP 0 and MAP 1. Which mapping is used is determined by the **MMU Task Bit** (0 = MAP 0, 1 = MAP 1). The monitor will display which map is in effect by drawing lines from the MAP column to the CPU Memory ranges.

This shows that MAP 1 is currently not in use and invisible to the CPU.

This shows that MAP 0 is currently being mapped into the CPU Memory space.

The MMU is controlled by several registers:

**MMU Task Bit:** This selects which map set should be seen by the CPU.

**MMU Enable Bit:** This allows the CoCo3 to be CoCo1/2 compatible. Disabling the MMU will just provide a static 64KB to the CPU.

**Ram Vectors**: Enabling this register will cause CPU memory FE00-FEFF to remain constant regardless of the mapping.

**Rom Mapping:** A two-bit register that gives you several options on mapping RAM/ROM:

0 or 1 :16K Internal ROM, 16K External ROM

2 :32K Internal ROM

3 :32K External ROM

This is a display of the real memory in the machine.

Use this selection control to display a page of real memory.

Table

Description automatically generated

The MMU will map real memory into CPU memory by 8KB pages – that is 8192 bytes per page. These pages are numbered from 0 to 3F (hex) and are placed in the MMU’s 8 registers to make a mapping. The CoCo3’s MMU has two sets of 8 registers, so that two maps can always be ready to be used.