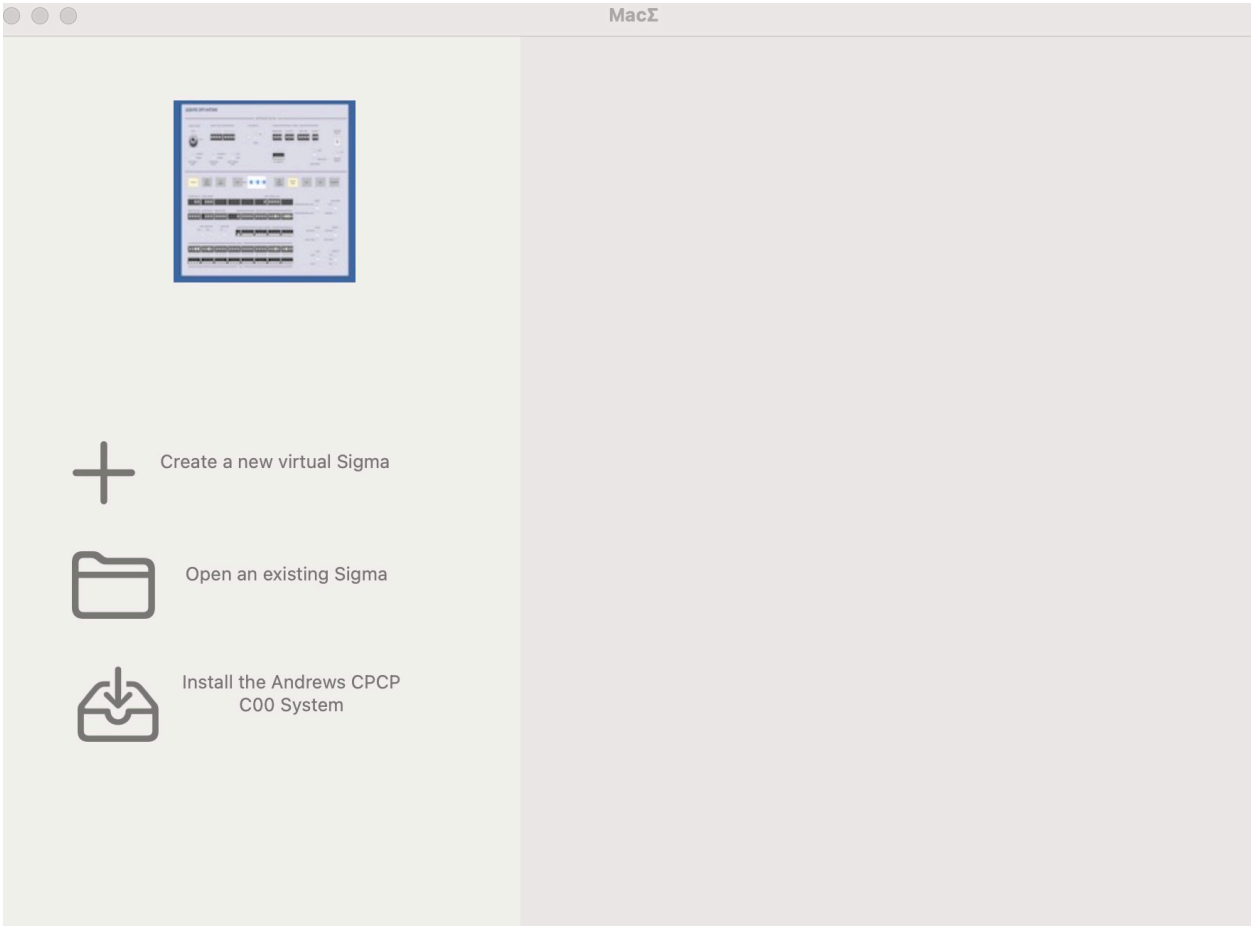
MACΣ - May 2025.

MACΣ is a MacOS emulator for an SDS Sigma 7. It provides a pretty realistic panel display, a console, and terminal windows. There is also a visual debugger. It is written entirely in Swift (except that there are a few trivial C routines).

I'm definitely not a hardware expert and my goal was just to have something that could convey the look and feel of the machine. I had not coded anything significant on the Mac before and I wanted learn Swift anyway, so implementing the panel was a fun way to start. The CPU (i.e instruction emulator) was next. I compressed the IOP/DC/UNIT stuff for devices into a simple IOP/Device model that satisfied the requirements of the Sigma IO instructions and does the actual I/O asynchronously.

A virtual Sigma (aka. "machine") is represented on disc by a MacOS package file (i.e. directory) containing the necessary settings and data to represent a



complete Sigma environment. It contains a sqlite3 database which contains tables for settings and the definition of virtual devices.

Installation

Install the Xcode source to an appropriate directory, and open the project with Xcode. You can build and run the emulator from within the Xcode IDE, however it will run considerably faster if you build a deployable app using the Product/Archive menu. Once built, use the Distribute App/Custom/Copy functions to produce an installable app.

When first started, MACΣ will create a sqlite3 database in your local library directory. This database contains tables for globally applicable (i.e. across all machines) settings, and a table of recently opened machines. It will also create a directory called MACΣ.logs for runtime log messages.

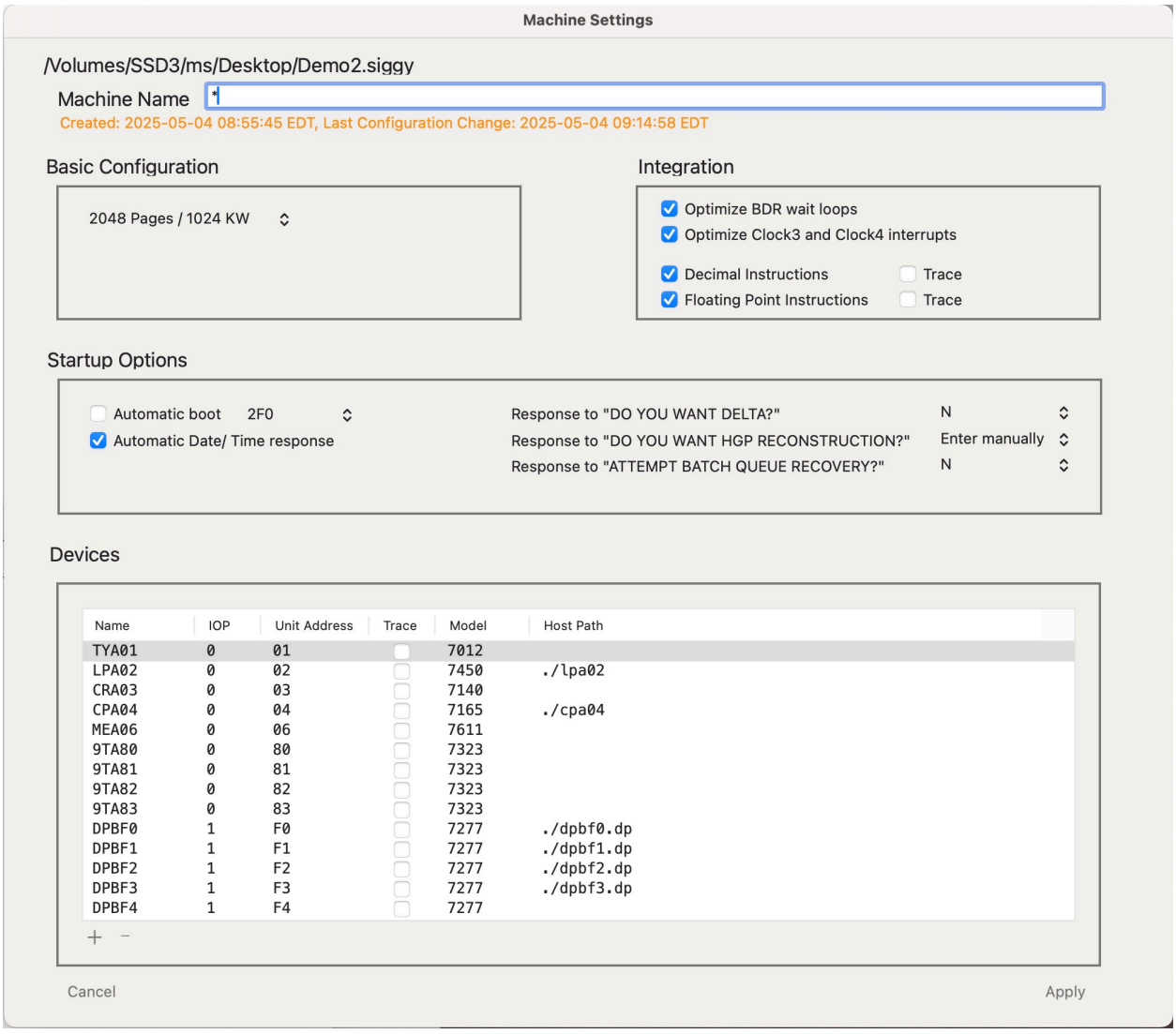
After initialization, MACΣ displays a startup window, which provides the following functions:

1. Create a new virtual Sigma
    - Asks for a location to create the Virtual Sigma directory,
    - Asks for the location of a PO tape. This will get copied into the destination directory, Initially, I used a downloaded copy of Ken Rector's F00 SIMH PO, available here: https://github.com/kenrector/sigma-cpv-kit/blob/main/f00/f00disc/f00disc.tap. After a while I settled on a large memory configuration with more tapes, disk and a card punch. The card punch can be used to transfer files from the sigma to the Mac host computer (and vice versa with the card reader - this is discussed later). You can select a zip file for the PO tape - If it contains a .mt or .tap file with the same name, it will be unzipped as it is copied to the target directory.
    - Creates a basic virtual hardware configuration including CPU, 8 IOPs, 8 discs, 4 RADs, 4 tapes, console, printer, card reader, card punch and a COC.
    - Allows the hardware configuration to be inspected and changed before booting.
    - Powers on the CPU and primes it for a tape boot.

2. Open an existing Sigma
    - Opens a browser window to locate and open the virtual sigma of interest,
    - Powers on the selected sigma and primes it for a boot from the most recently used device.

3. Install the Andrews CPCP C00 System
    - Asks for the location of the new virtual Sigma directory,
    - Creates a basic virtual hardware configuration,
    - Copies the DFBF0 and DPBF1 disk images,
    - Powers on the virtual Sigma and primes it for a disc boot.

Configuration



When creating a new Sigma, the configuration window is populated with a basic configuration and displayed.

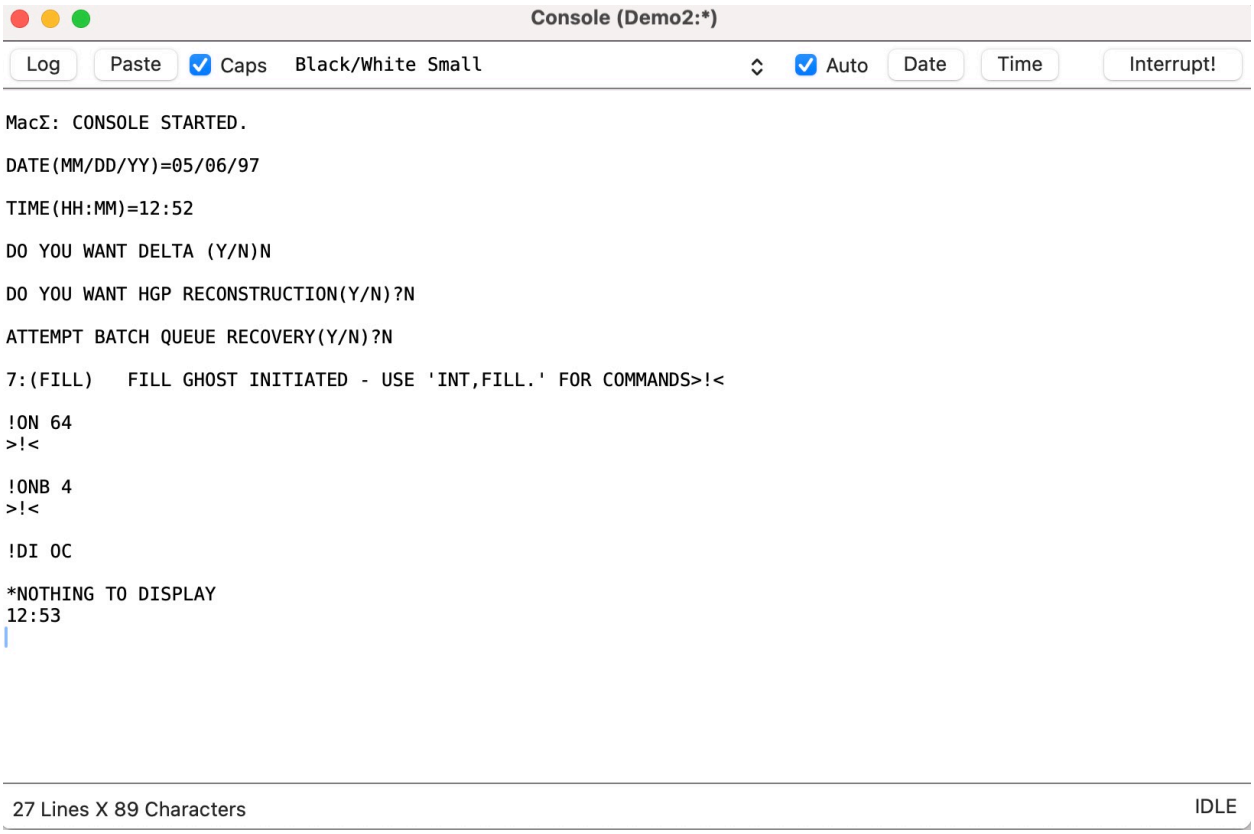
The intention is that changes be made before the system is started. The configuration window can be accessed later from the menu, however if you

modify the hardware configuration once the machine has started executing, it will be reset.

Console and Panel

The Console and Panel windows are displayed after configuration completes. These are the first windows displayed when an existing sigma is started. (An existing sigma can also be started from the desktop) by double clicking on its icon.

The console window is straightforward. The Date and Time buttons can be used to produce date and time responses to the boot time prompts. The date is selected to be the current month and day, but a year between 1970 and 1997 that has the correct day of week and leap year characteristics as the current year.



The log button prompts for a Mac host file for logging console input and output.

The paste button causes the paste window to be displayed. This can be used to save / edit / recall prepared command lines. The paste buffers are saved in the virtual machine database, so they persist from one session to the next.

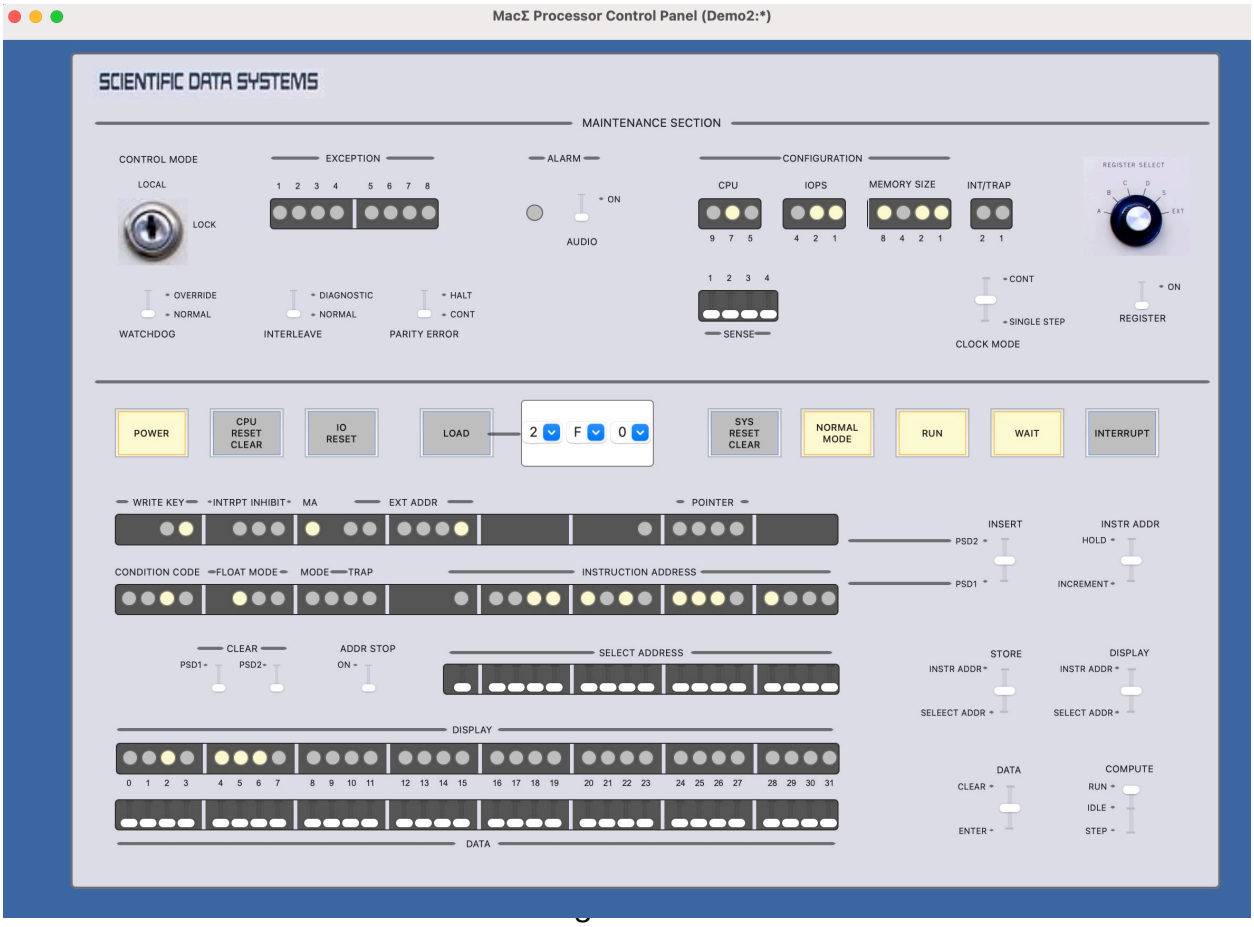
There is a selector for font size and foreground / background colours.

The auto button can be used to suppress (by unchecking it) the configured automatic responses to date/time/HGP recon/ etc prompts.

The panel window consists of various buttons, switches and lights. Some non-functional visual elements such as the key lock and the register select box (which should be a dial) are there to make things look right. In the upper maintenance section, the INT/TRAP lights work, but others have been repurposed (e.g. the memory fault lights show the recent trap or interrupt location), or don't work at all (ALARM). There is no reason why the alarm could not be implemented at some point. Most switches (other than the sense switches and the compute switch are not functional.

The lights in the lower section are updated about 5 times a second and produce an effect that is close to the way a real panel looked when the machine is running.

The Normal Mode button is red when the CPU is not running.



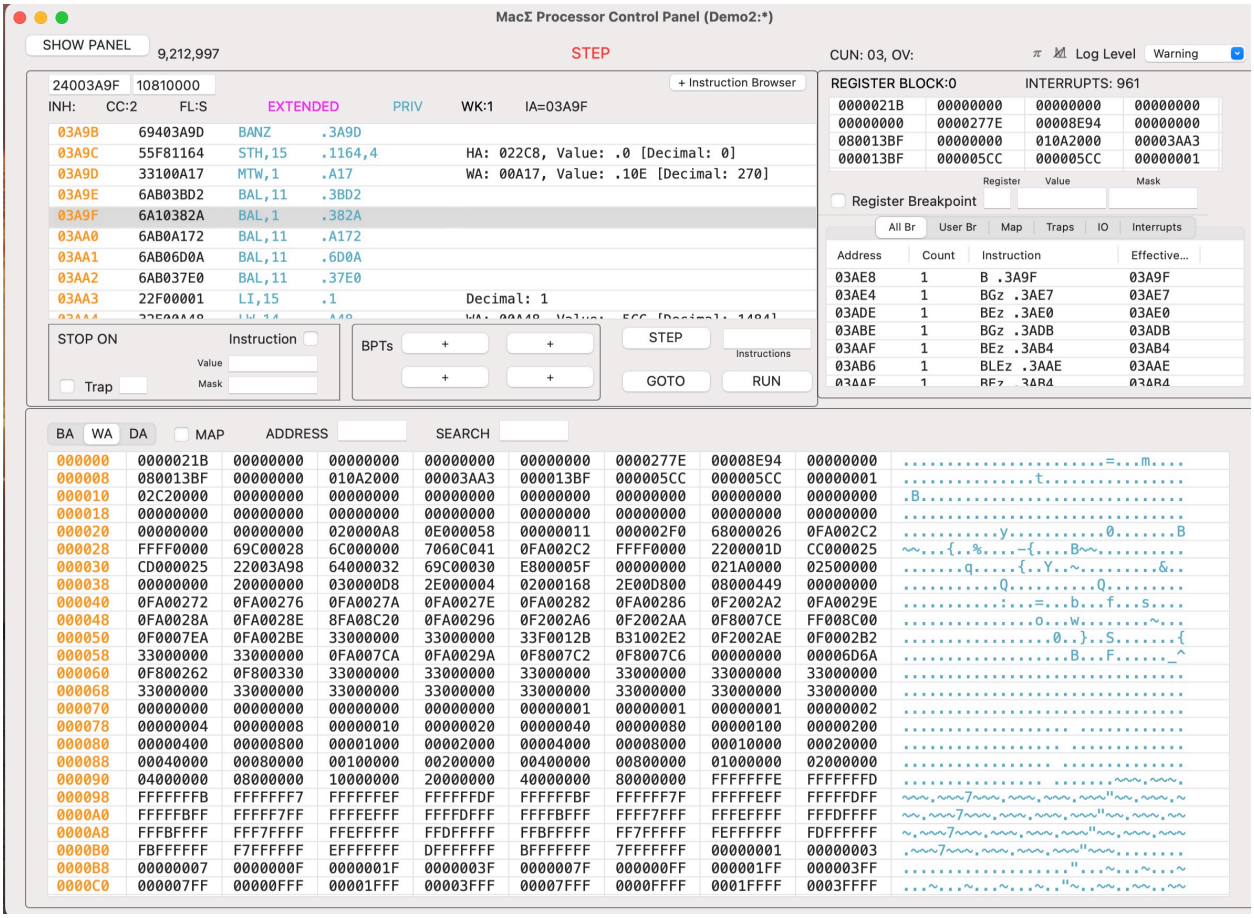
To start the machine, select the correct boot address and either push the Compute switch to run, or click the Run button.

For the initial tape, boot respond SFTP to the prompt for options. Use the date and time buttons on the console window to respond to the date and time prompts.

Press the Interrupt button to switch the from the panel display to the debugger. You can also use the View menu.

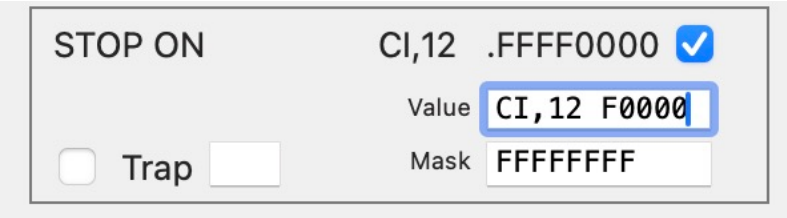
Debug Window

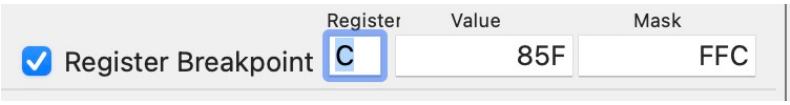
Press the Stop button in the debugger to pause the machine. Some controls in the debugger are not operative when the machine is running.



There is a lot to the debugger window. Starting at the top left, the Show-Panel button returns to the panel display. To its right is the instruction counter. The current execution status (in this case instruction step mode) is shown in red. On the right top side is the log level selector (various settings produce less or more log output). These log messages can be seen in the XCODE IDE and they are written to disk in the local library MacΣ.logs directory.

Below the status bar is an instruction display with the PSD in the top left. The current instruction is highlighted. The previous and next 32(or so) instructions can be scrolled. There are four breakpoints that can be set with the buttons in the center below the instruction window. Each breakpoint can stop on any combination of execute, read, write, or "transition" (execution transitions from below to above the specified address). To the left of the breakpoint section, there are selectable STOPs such as on a specific trap, a specific instruction opcode, and, in the register section, there is a single register breakpoint to stop when a specific register contains a specific value.

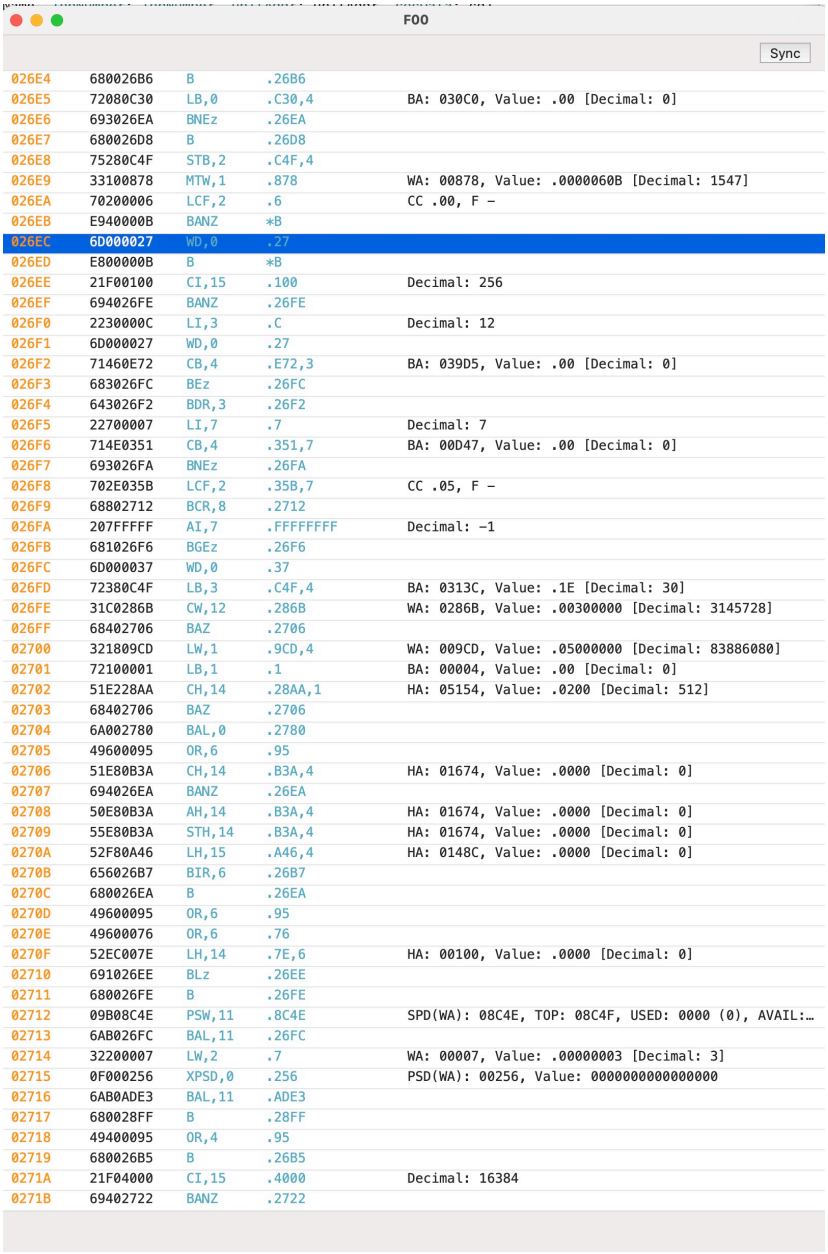




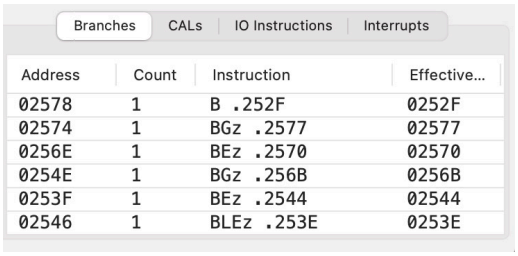
Above and to the right of the instruction display area the current user(CUN) and monitor overlay(OV) are displayed. The addresses of the S:CUN location and the P:NAME location are derived at the time the first mapped instruction is executed.

The GOTO button is useful when stepping through code and you want to skip several instructions or skip over a subroutine. Select the instruction in the display where you want to go and click goto. You can step multiple instructions at once by putting a number on the field to the right of the step button before it is pressed.

The + Instruction Browser button creates a new floating instruction browser window that is capable of scrolling all of memory
You can start multiple instruction browser windows if there is more than one section of code that needs to be observed. The sync button causes the

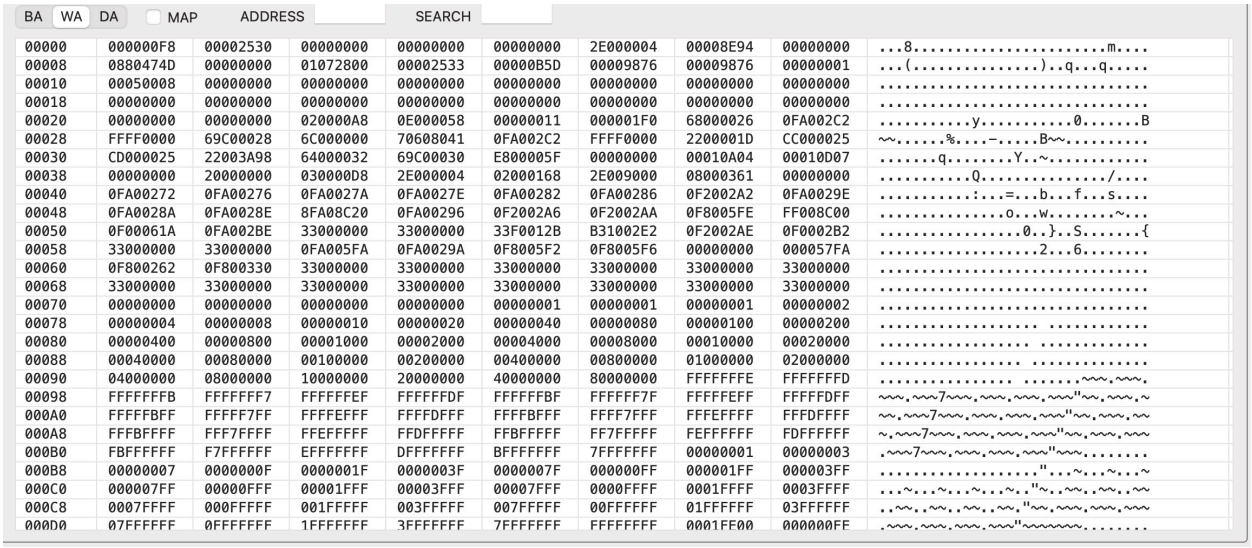


instruction browser to be synchronized with the instruction display in the main debug window.



Below the register display you can select the last 32 branches (all and user mode); Map related intructions; Traps; IO instructions, and Interrupts.

At the bottom is the memory display. You can scroll the entire contents of



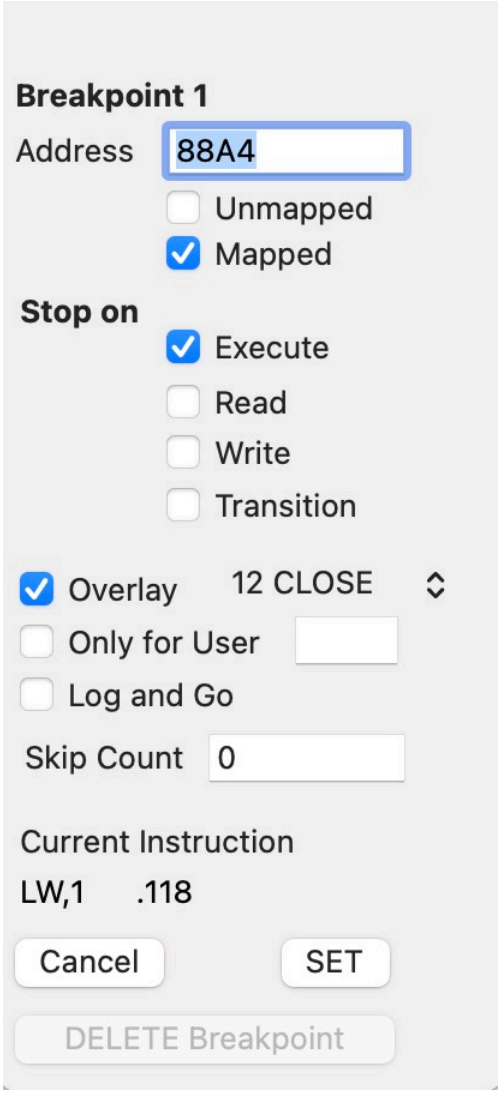
00000	000000F8	00002530	00000000	00000000	00000000	2E000004	00008E94	00000000	...8.....m....
00008	0880474D	00000000	01072800	00002533	00000B5D	00009876	00009876	00000001	...(.....)..q...q.....
00010	00050008	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
00018	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	.....
00020	00000000	00000000	020000A8	0E000058	00000011	000001F0	68000026	0FA002C2	.....y.....0.....B
00028	FFFF0000	69C00028	6C000000	70608041	0FA002C2	FFFF0000	2200001D	CC000025	~~.....%....-.....B~~.....
00030	CD000025	22003A98	64000032	69C00030	E800005F	00000000	00010A04	00010D07	.....q.....Y..~.....
00038	00000000	20000000	030000D8	2E000004	02000168	2E009000	08000361	00000000	.....Q...../....
00040	0FA00272	0FA00276	0FA0027A	0FA0027E	0FA00282	0FA00286	0F2002A2	0FA0029E	.....:...=...b...f...s....
00048	0FA0028A	0FA0028E	8FA08C20	0FA00296	0F2002A6	0F2002AA	0F8005FE	FF008C00	.....o...w.....~...
00050	0F00061A	0FA002BE	33000000	33000000	33F0012B	B31002E2	0F2002AE	0F0002B2	.....0..}..S.....{
00058	33000000	33000000	0FA005FA	0FA0029A	0F8005F2	0F8005F6	00000000	000057FA	.....2...6.....
00060	0F800262	0F800330	33000000	33000000	33000000	33000000	33000000	33000000	.....
00068	33000000	33000000	33000000	33000000	33000000	33000000	33000000	33000000	.....
00070	00000000	00000000	00000000	00000000	00000001	00000001	00000001	00000002	.....
00078	00000004	00000008	00000010	00000020	00000040	00000080	00000100	00000200	....
00080	00000400	00000800	00001000	00002000	00004000	00008000	00010000	00020000	....
00088	00040000	00080000	00100000	00200000	00400000	00800000	01000000	02000000	....
00090	04000000	08000000	10000000	20000000	40000000	80000000	FFFFFFFE	FFFFFFFD	....~~~.~~~.
00098	FFFFFFFB	FFFFFFF7	FFFFFFEF	FFFFFFDF	FFFFFFBF	FFFFFF7F	FFFFFEFF	FFFFFDFF	~~~.~~~7~~~.~~~.~~~.~~~"~~.~~~.~
000A0	FFFFFBFF	FFFFF7FF	FFFFEFFF	FFFFDFFF	FFFFBFFF	FFFF7FFF	FFFEFFFF	FFFDFFFF	~~.~~~7~~~.~~~.~~~.~~~"~~.~~~.~~
000A8	FFFBFFFF	FFF7FFFF	FFEFFFFF	FFDFFFFF	FFBFFFFF	FF7FFFFF	FEFFFFFF	FDFFFFFF	~.~~~7~~~.~~~.~~~.~~~"~~.~~~.~~~
000B0	FBFFFFFF	F7FFFFFF	EFFFFFFF	DFFFFFFF	BFFFFFFF	7FFFFFFF	00000001	00000003	.~~~7~~~.~~~.~~~.~~~"~~~.....
000B8	00000007	0000000F	0000001F	0000003F	0000007F	000000FF	000001FF	000003FF	....."...~...~...~
000C0	000007FF	00000FFF	00001FFF	00003FFF	00007FFF	0000FFFF	0001FFFF	0003FFFF	...~...~...~...~..."~..~~..~~..~~
000C8	0007FFFF	000FFFFF	001FFFFF	003FFFFF	007FFFFF	00FFFFFF	01FFFFFF	03FFFFFF	..~~..~~..~~..~~."~~.~~~.~~~.~~~
000D0	07FFFFFF	0FFFFFFF	1FFFFFFF	3FFFFFFF	7FFFFFFF	FFFFFFFF	0001FF00	000000FF	.~~~.~~~.~~~.~~~"~~~~~~~~.....

memory here. The BA/WA/DA selectors alter the address column on the left of the display. Check the MAP box to display the memory mapped (if possible). Entering a value in the address field causes the memory display to position to that address. The search field can be used to search for single word hex values or for text strings.

Memory can be altered from this window by entering word values in hex.

Breakpoints

Click on one of the four breakpoint buttons showing a + to add a breakpoint. Various combinations are possible. A Transition breakpoint can catch a piece of code that moves from below a specific boundary address to something higher that the boundary. This is useful for catching jobs just starting.



Note that breakpoints can be specific to a user, and/or specific to a particular monitor overlay. Checking “Log and go” causes a a register dump to be written to the log file when the breakpoint is hit, and to be automatically continued.

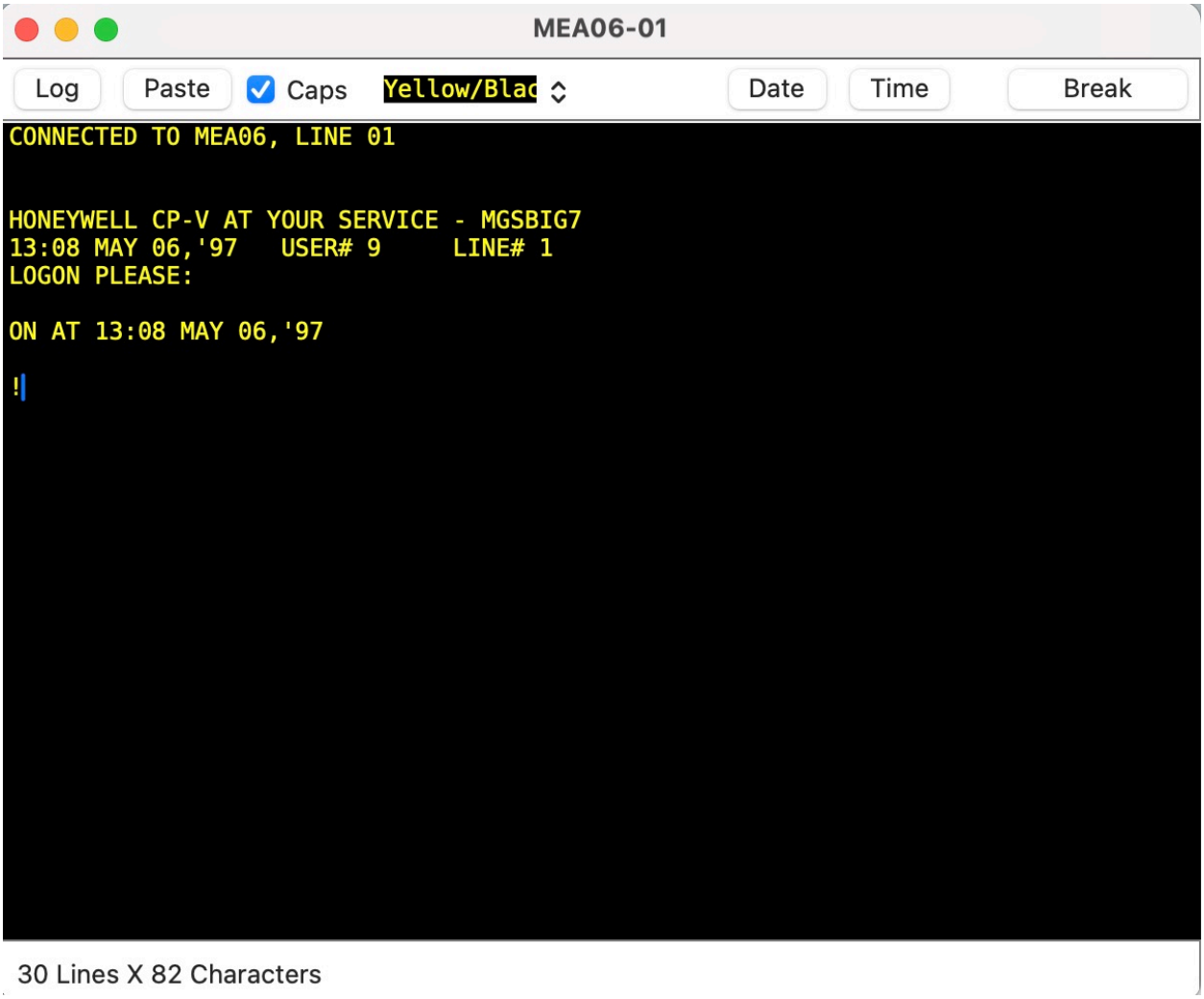
For F00, the tape boot and GENMDs take about 2.5 minutes to complete. There is a 6 minute video of the initial creation of a virtual machine and the tape boot here: https://drive.google.com/file/d/1bF8WO5OJXn5wJQTQxpfkby6RfpbXe2Ft/view?usp=share_link

After the GENMDs complete, the system can be ZAPped and subsequently booted from the RAD. A RAD boot takes on the order of 3 seconds on a Mac with M1 processor.

Once FILL starts, it should be possible to logon.

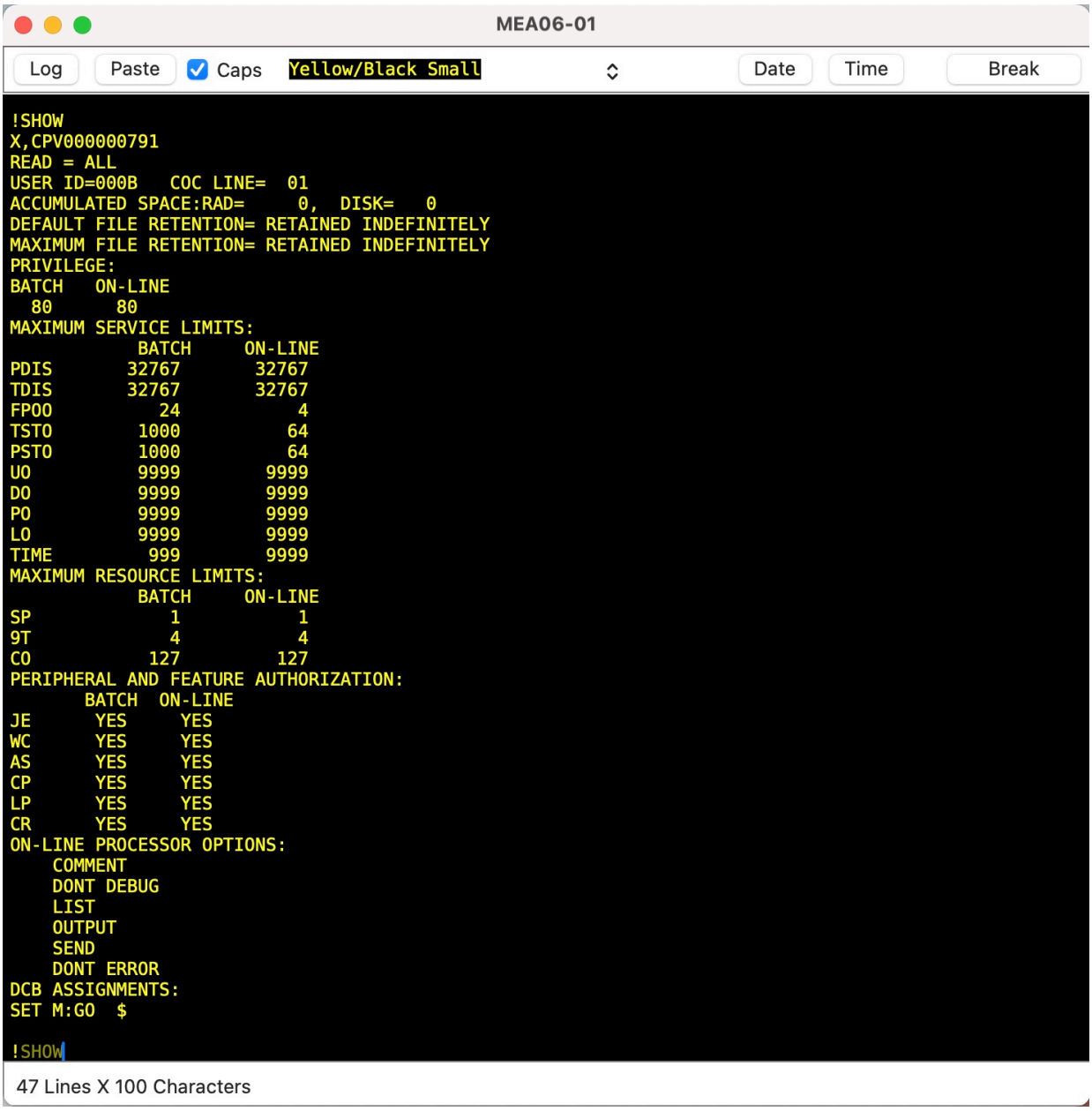
Start a new terminal from the menu: click Machine > New Terminal Window. The terminal window is identical to the console window except that the “interrupt” button is replaced by the “break” button, and that there is no “auto” checkbox.

Use the :SYS,LBE account to logon initially.



Both the terminal and the console can be scrolled back through the session history.

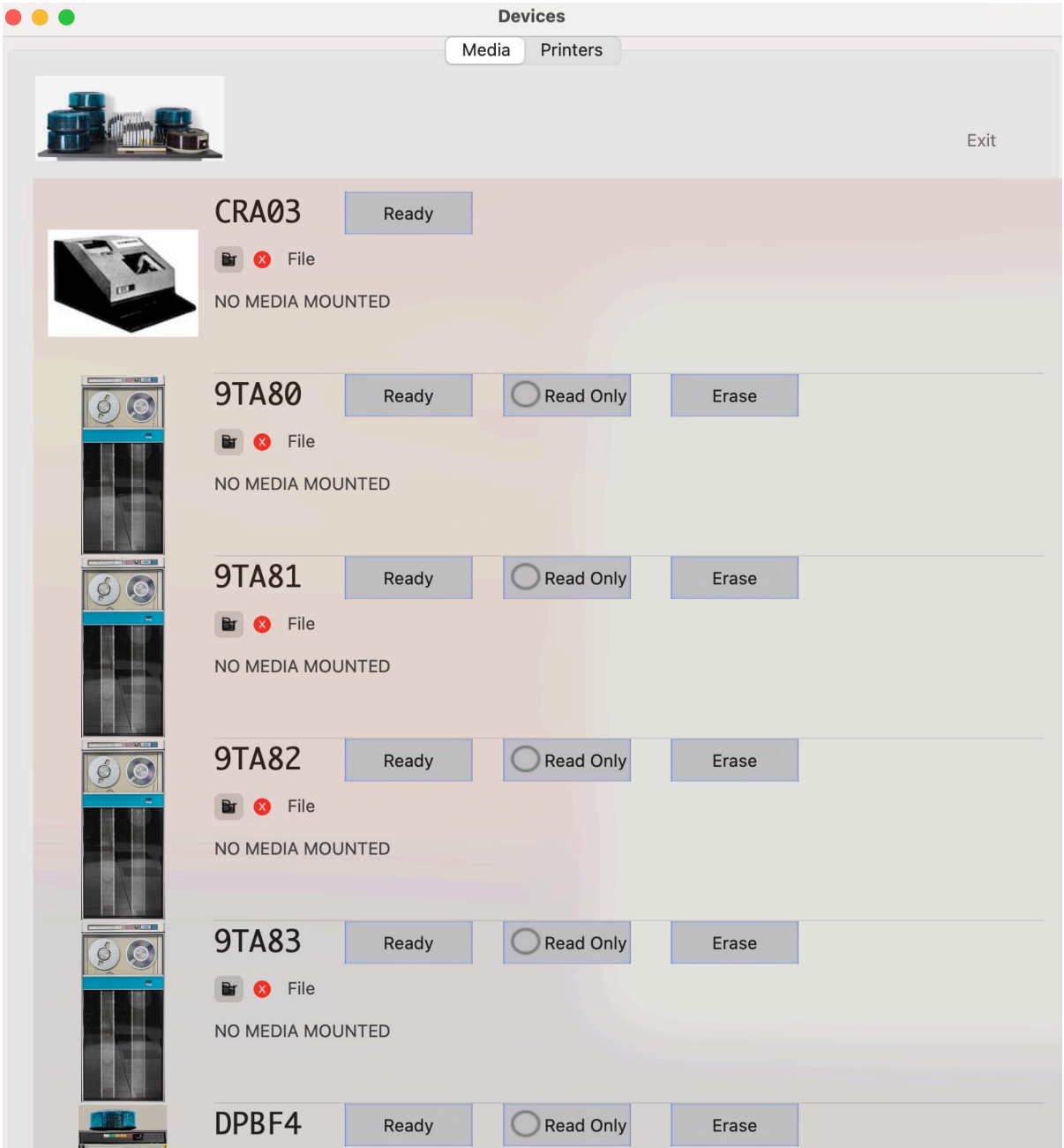
They are also capable of previous command editing: press the up arrow to show the last command line entered. Continue to press the up arrow to move backwards through the command line history. Once the command to be modified or re-issued is reached, the command can be changed by using the left arrow, right arrow to position the cursor and the tab key to toggle insertion mode. Press return to enter the edited line.



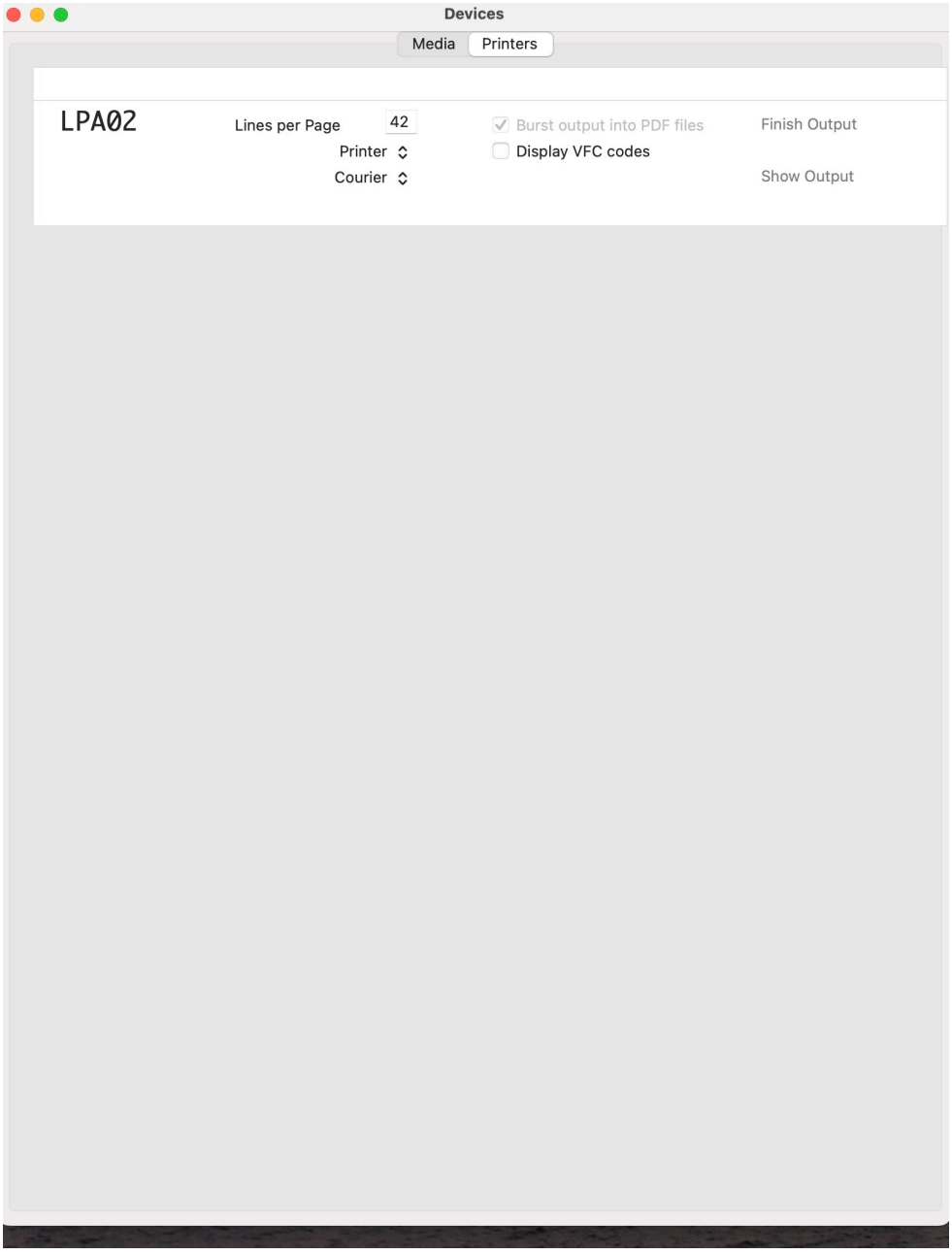
Media and Line Printer Options

This window is started by selecting Media and Printing from the Machine menu. The media tab is used to connect Mac host files with CP-V devices. It is used when a tape or disk mount is requested on the console to connect the Mac file to the appropriate device. This can be done using the browse button in the entry for the appropriate device, or by dragging and dropping the applicable file.

Similarly for the card reader, a Mac file can be attached by dragging it to the Card reader entry in this window. Perhaps more simply a Mac file that is dragged and dropped into the Panel window is also processed as a card reader input. Both tape and card handling work best with the HOTAVR and HOTCARD PASS2 options specified.

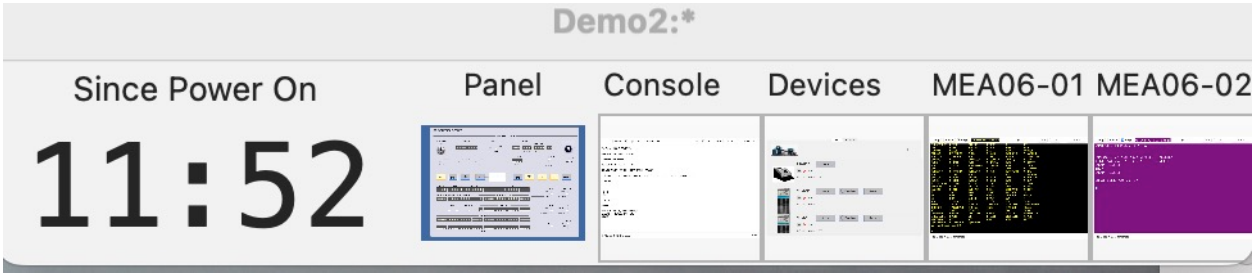


The printer tab can be used to change the lines per page, paper size and font for each printer device.



The Window Toolbar

The window toolbar is enabled and disabled from the view menu. It contains a session timer and buttons for each of the windows that belong to the virtual machine. Clicking one one of the buttons hides/shows the applicable window.



This is especially helpful on smaller displays.

Print and Punch Output

The View menu also contains options for displaying (using the Mac finder) the Print and Punch output directories. A job's printer output is typically collected and output to a pdf file. These files are grouped by logon in the virtual machine print directory.

Card punch output is also grouped by logon in the virtual machine punch directory. Each file is a txt file. Using PCL to copy an edit file to the card punch provides an easy way to convert the file to ascii and make it usable on the Mac.

Snapshots

Snapshots take a copy of the device files in the virtual machine directory. They do not include memory or other run-state information. They can be restored to a different virtual machine than the one from which they were taken.