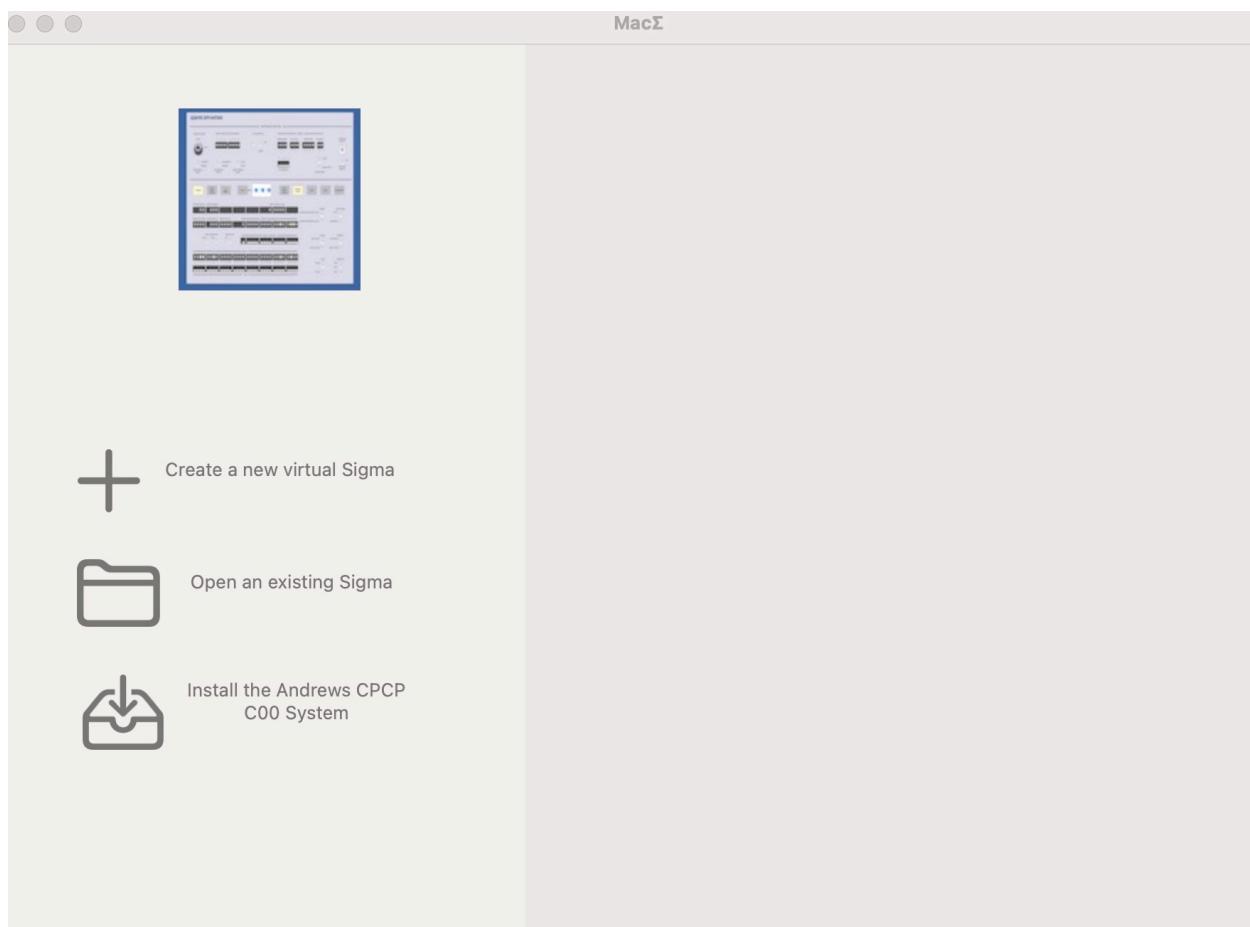


MACΣ - May 2025.

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

Machine Settings

/Volumes/SSD3/ms/Desktop/Demo2.siggy

Machine Name:

Created: 2025-05-04 08:55:45 EDT, Last Configuration Change: 2025-05-04 09:14:58 EDT

Basic Configuration

2048 Pages / 1024 KW

Integration

Optimize BDR wait loops
 Optimize Clock3 and Clock4 interrupts
 Decimal Instructions Trace
 Floating Point Instructions Trace

Startup Options

Automatic boot 2F0 Automatic Date/ Time response

Response to "DO YOU WANT DELTA?" N
Response to "DO YOU WANT HGP RECONSTRUCTION?" Enter manually
Response to "ATTEMPT BATCH QUEUE RECOVERY?" N

Devices

Name	IOP	Unit Address	Trace	Model	Host Path
TYA01	0	01	<input type="checkbox"/>	7012	
LPA02	0	02	<input type="checkbox"/>	7450	./lpa02
CRA03	0	03	<input type="checkbox"/>	7140	
CPA04	0	04	<input type="checkbox"/>	7165	./cpa04
MEA06	0	06	<input type="checkbox"/>	7611	
9TA80	0	80	<input type="checkbox"/>	7323	
9TA81	0	81	<input type="checkbox"/>	7323	
9TA82	0	82	<input type="checkbox"/>	7323	
9TA83	0	83	<input type="checkbox"/>	7323	
DPBF0	1	F0	<input type="checkbox"/>	7277	./dpbf0.dp
DPBF1	1	F1	<input type="checkbox"/>	7277	./dpbf1.dp
DPBF2	1	F2	<input type="checkbox"/>	7277	./dpbf2.dp
DPBF3	1	F3	<input type="checkbox"/>	7277	./dpbf3.dp
DPBF4	1	F4	<input type="checkbox"/>	7277	

+ -

Cancel Apply

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

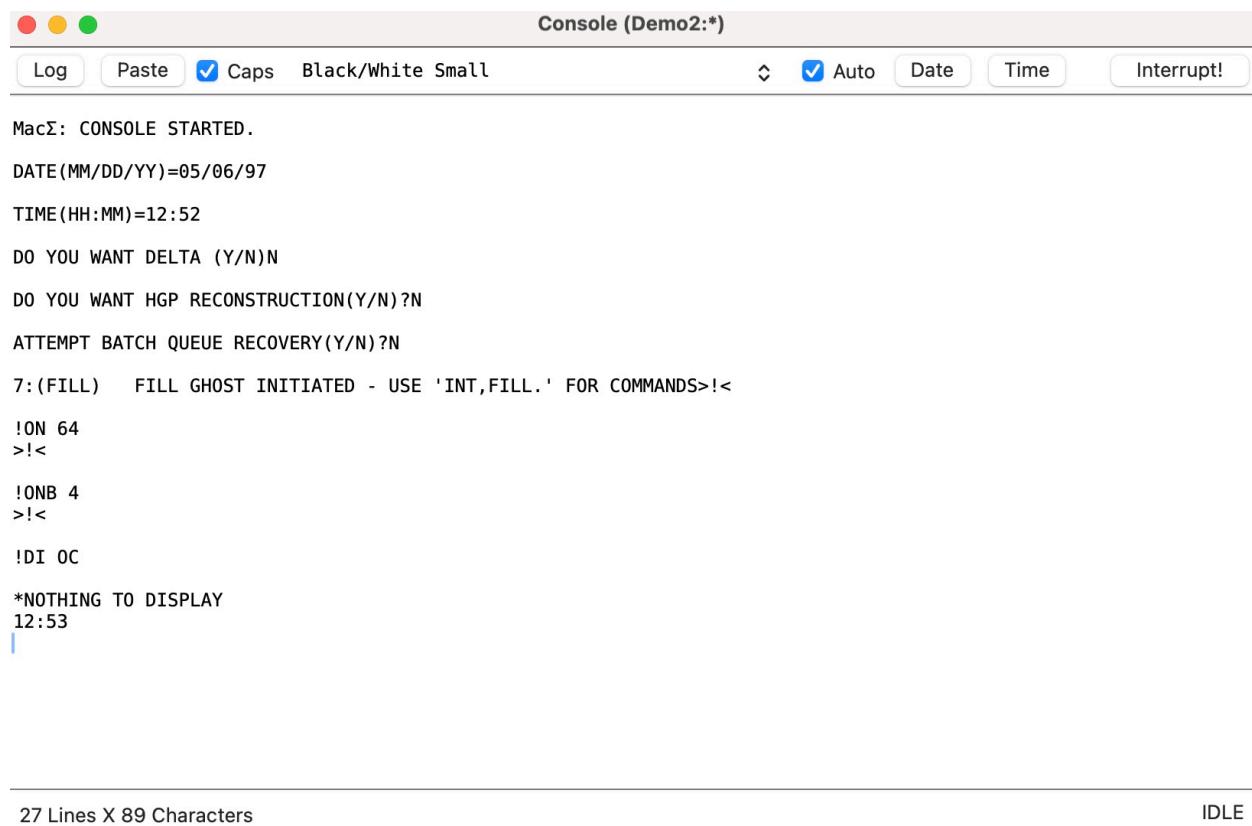
The intention is that changes be made before the system is started. The configuration window can be accessed later from the menu, however if you

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.



Console (Demo2:*)

Log Paste Caps Black/White Small

MacΣ: CONSOLE STARTED.
DATE(MM/DD/YY)=05/06/97
TIME(HH:MM)=12:52
DO YOU WANT DELTA (Y/N)N
DO YOU WANT HGP RECONSTRUCTION(Y/N)?N
ATTEMPT BATCH QUEUE RECOVERY(Y/N)?N
7:(FILL) FILL GHOST INITIATED - USE 'INT,FILL.' FOR COMMANDS>!<
!ON 64
>!<
!ONB 4
>!<
!DI OC
*NOTHING TO DISPLAY
12:53

27 Lines X 89 Characters IDLE

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.

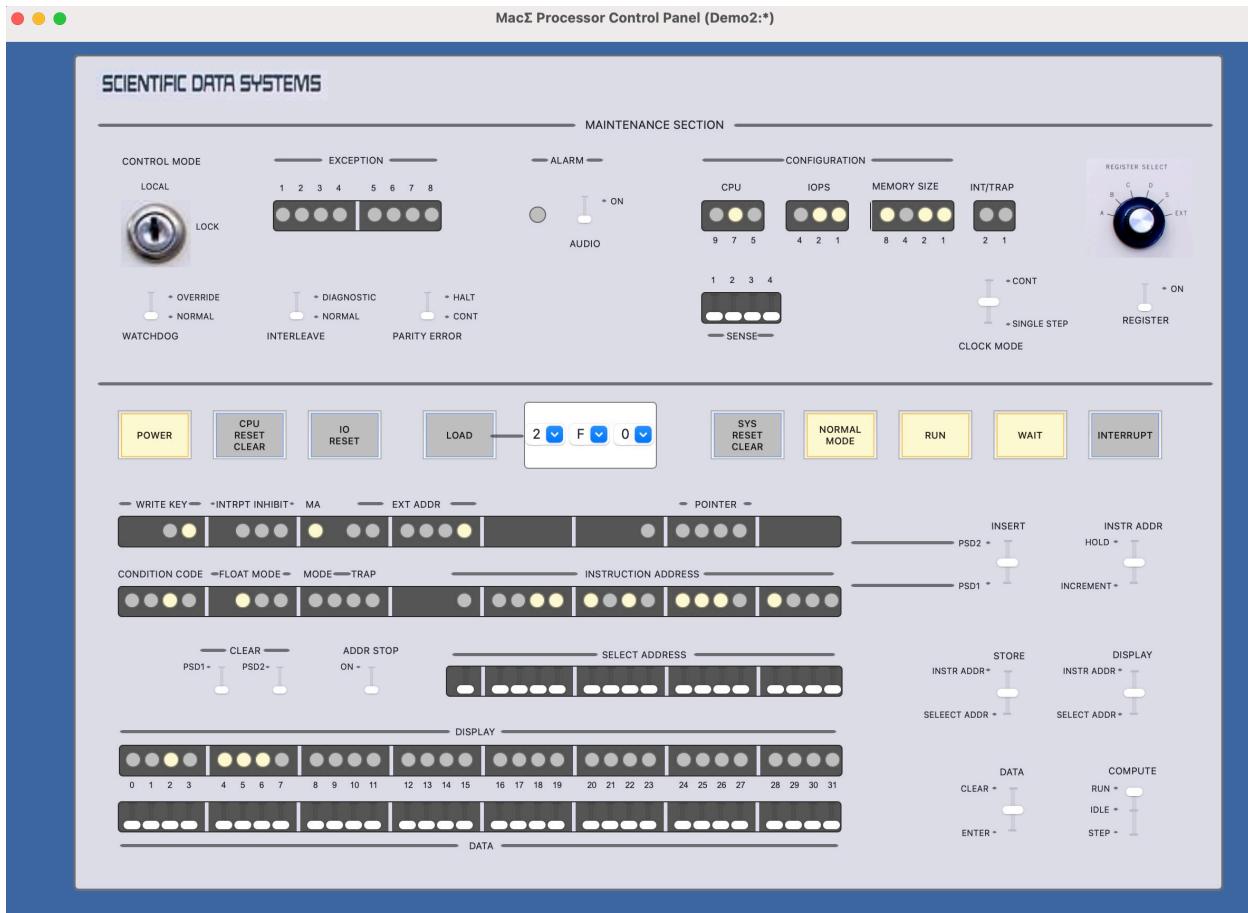
There is a selector for font size and foreground / background colours.

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

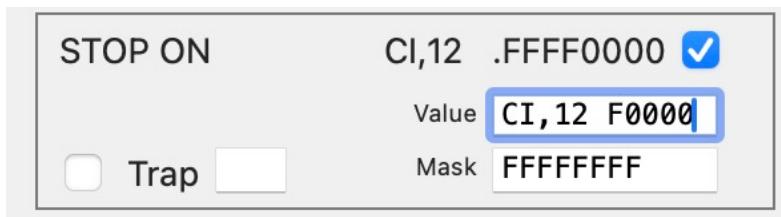
Press the Stop button in the debugger to pause the machine. Some controls in the debugger are not operative when the machine is running.



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

FOO

Sync

026E4	680026B6	B	.26B6	
026E5	72080C30	LB,0	.C30,4	BA: 030C0, Value: .00 [Decimal: 0]
026E6	693026EA	BNEz	.26EA	
026E7	680026D8	B	.26D8	
026E8	75280C4F	STB,2	.C4F,4	
026E9	33100878	MTW,1	.878	WA: 00878, Value: .0000060B [Decimal: 1547]
026EA	70200006	LCF,2	.6	CC .00, F -
026EB	E940000B	BANZ	*B	
026EC	60000027	WD,0	.27	
026ED	E80000B	B	*B	
026EE	21F00100	CT,15	.100	Decimal: 256
026EF	694026FE	BANZ	.26FE	
026F0	2230000C	LT,3	.C	Decimal: 12
026F1	60000027	WD,0	.27	
026F2	71460E72	CB,4	.E72,3	BA: 039D5, Value: .00 [Decimal: 0]
026F3	683026FC	BEz	.26FC	
026F4	643026F2	BDR,3	.26F2	
026F5	22700007	LT,7	.7	Decimal: 7
026F6	714E0351	CB,4	.351,7	BA: 00D47, Value: .00 [Decimal: 0]
026F7	693026FA	BNEz	.26FA	
026F8	702E035B	LCF,2	.35B,7	CC .05, F -
026F9	68802712	BCR,8	.2712	
026FA	207FFFFF	AI,7	.FFFFFFF	Decimal: -1
026FB	681026F6	BGEz	.26F6	
026FC	60000037	WD,0	.37	
026FD	72380C4F	LB,3	.C4F,4	BA: 0313C, Value: .1E [Decimal: 30]
026FE	31C0286B	CW,12	.286B	WA: 0286B, Value: .00300000 [Decimal: 3145728]
026FF	68402706	BAZ	.2706	
02700	321889CD	LW,1	.9CD,4	WA: 009CD, Value: .05000000 [Decimal: 83886080]
02701	72100001	LB,1	.1	BA: 00004, Value: .00 [Decimal: 0]
02702	51E228AA	CH,14	.28A,A1	HA: 05154, Value: .0200 [Decimal: 512]
02703	68402706	BAZ	.2706	
02704	6A002780	BAL,0	.2780	
02705	49600095	OR,6	.95	
02706	51E80B3A	CH,14	.B3A,4	HA: 01674, Value: .0000 [Decimal: 0]
02707	694026EA	BANZ	.26EA	
02708	50E80B3A	AH,14	.B3A,4	HA: 01674, Value: .0000 [Decimal: 0]
02709	55E80B3A	STH,14	.B3A,4	HA: 01674, Value: .0000 [Decimal: 0]
0270A	52F80A46	LH,15	.A46,4	HA: 0148C, Value: .0000 [Decimal: 0]
0270B	656026B7	BIR,6	.26B7	
0270C	680026EA	B	.26EA	
0270D	49600095	OR,6	.95	
0270E	49600076	OR,6	.76	
0270F	52E0007E	LH,14	.7E,6	HA: 00100, Value: .0000 [Decimal: 0]
02710	691026EE	BLz	.26EE	
02711	680026FE	B	.26FE	
02712	09B008C4E	PSW,11	.8C4E	SPD(WA): 08C4E, TOP: 08C4F, USED: 0000 (0), AVAIL:...
02713	6AB026FC	BAL,11	.26FC	
02714	32200007	LW,2	.7	WA: 00007, Value: .00000003 [Decimal: 3]
02715	0F000256	XPSD,0	.256	PSD(WA): 00256, Value: 0000000000000000
02716	6AB00ADE3	BAL,11	.ADE3	
02717	680028FF	B	.28FF	
02718	49400095	OR,4	.95	
02719	680026B5	B	.26B5	
0271A	21F04000	CI,15	.4000	Decimal: 16384
0271B	69402722	BANZ	.2722	

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

Branches	CALS	IO Instructions	Interrupts
Address	Count	Instruction	Effective...
02578	1	B .252F	0252F
02574	1	BGz .2577	02577
0256E	1	BEz .2570	02570
0254E	1	BGz .256B	0256B
0253F	1	BEz .2544	02544
02546	1	BLEz .253E	0253E

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

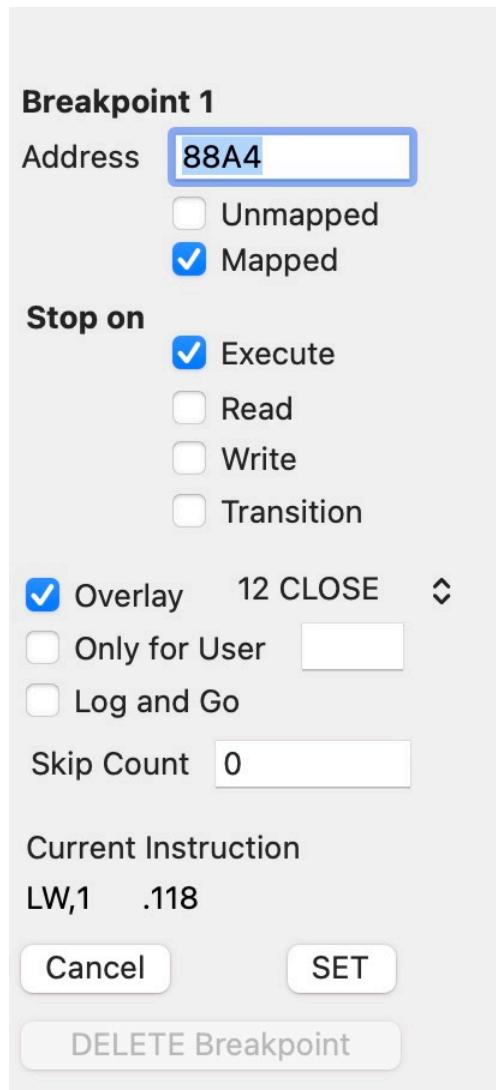
You can scroll the entire contents of

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 than 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 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.



The screenshot shows a Mac OS terminal window titled "MEA06-01". The window has standard OS X window controls (red, yellow, green) at the top left. The title bar also contains the window name. Below the title bar is a toolbar with several buttons: "Log", "Paste", "Caps" (with a checked checkbox), "Yellow/Blac" (with a dropdown arrow), "Date", "Time", and "Break". The main terminal area displays the following text:

```
CONNECTED TO MEA06, LINE 01

HONEYWELL CP-V AT YOUR SERVICE - MGSBIG7
13:08 MAY 06, '97  USER# 9      LINE# 1
LOGON PLEASE:

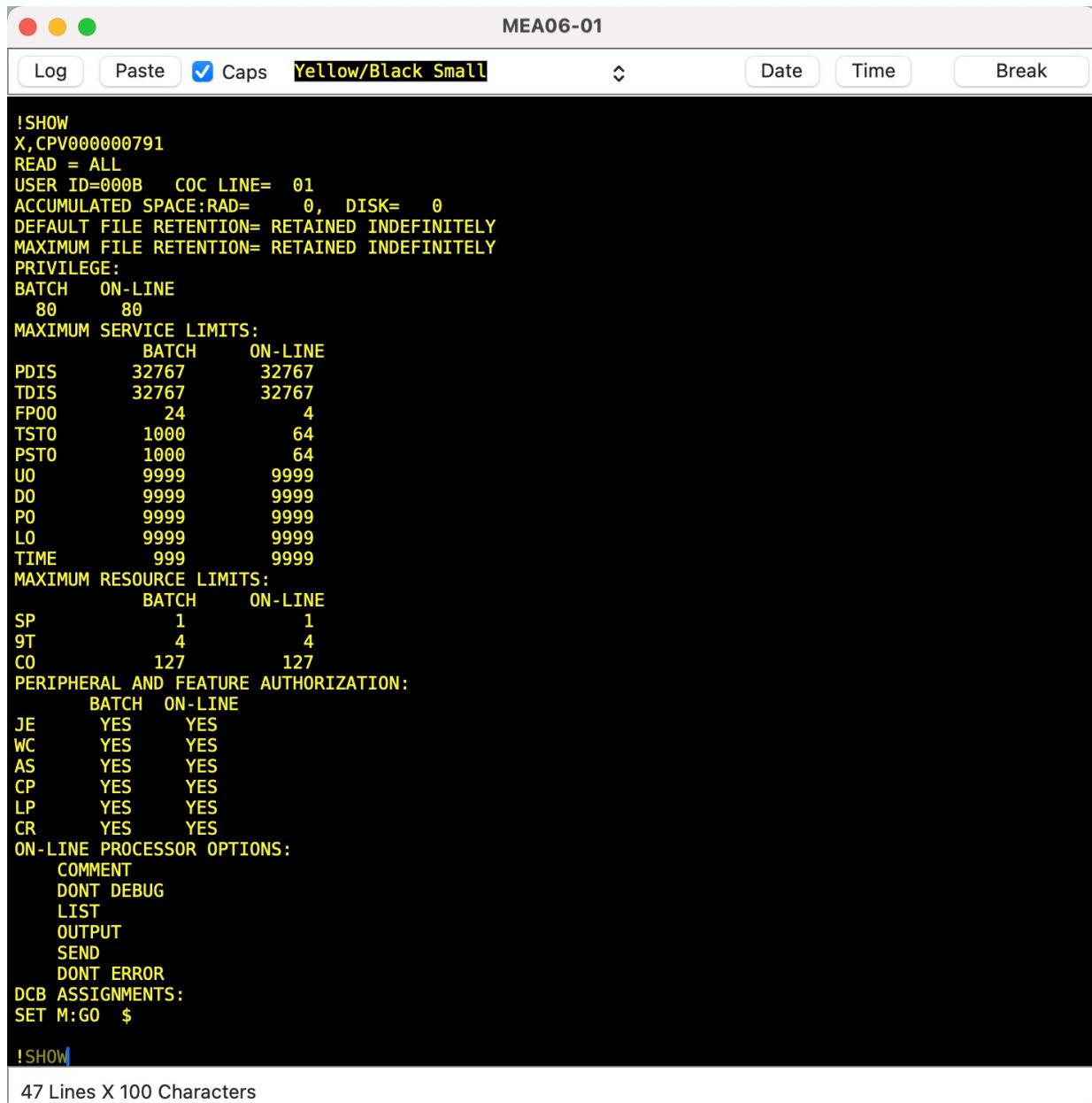
ON AT 13:08 MAY 06, '97

!|
```

30 Lines X 82 Characters

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.



The screenshot shows a terminal window titled "MEA06-01". The window has a title bar with three colored circles (red, yellow, green) and several status indicators: Log, Paste, Caps (with a checked checkbox), Yellow/Black Small, Date, Time, and Break. The main pane displays a series of system configuration parameters and limits. The text is color-coded in yellow and white on a black background. Key sections include:

- SHOW**
- X,CPV000000791
- READ = ALL
- USER ID=000B COC LINE= 01
- ACCUMULATED SPACE:RAD= 0, DISK= 0
- DEFAULT FILE RETENTION= RETAINED INDEFINITELY
- MAXIMUM FILE RETENTION= RETAINED INDEFINITELY
- PRIVILEGE:
- BATCH ON-LINE
80 80
- MAXIMUM SERVICE LIMITS:

	BATCH	ON-LINE
PDIS	32767	32767
TDIS	32767	32767
FP00	24	4
TST0	1000	64
PST0	1000	64
U0	9999	9999
D0	9999	9999
P0	9999	9999
L0	9999	9999
TIME	999	9999
- MAXIMUM RESOURCE LIMITS:

	BATCH	ON-LINE
SP	1	1
9T	4	4
CO	127	127
- PERIPHERAL AND FEATURE AUTHORIZATION:

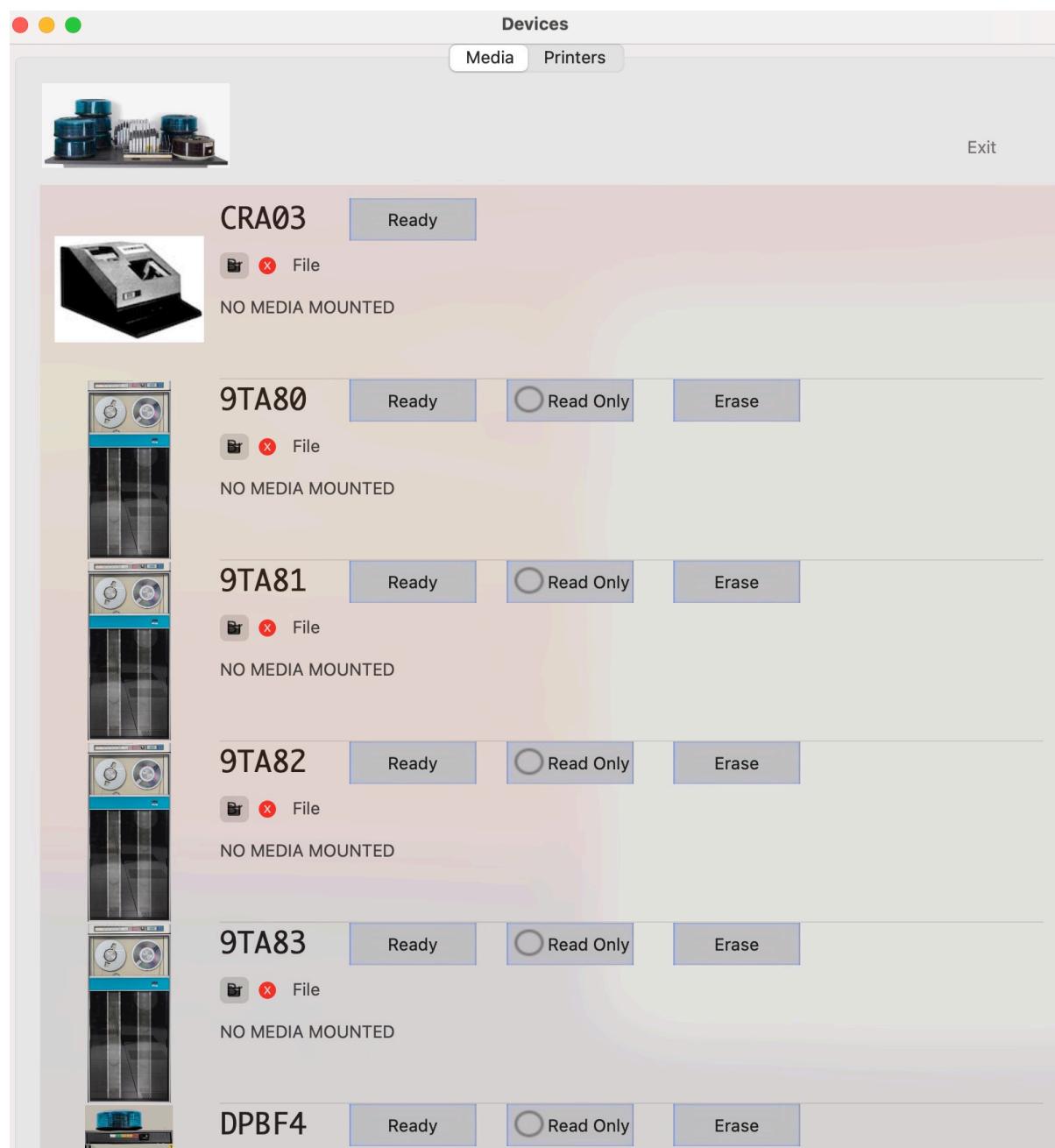
	BATCH	ON-LINE
JE	YES	YES
WC	YES	YES
AS	YES	YES
CP	YES	YES
LP	YES	YES
CR	YES	YES
- ON-LINE PROCESSOR OPTIONS:
 - COMMENT
 - DONT DEBUG
 - LIST
 - OUTPUT
 - SEND
 - DONT ERROR
- DCB ASSIGNMENTS:
SET M:GO \$
- SHOW**

At the bottom of the terminal window, it says "47 Lines X 100 Characters".

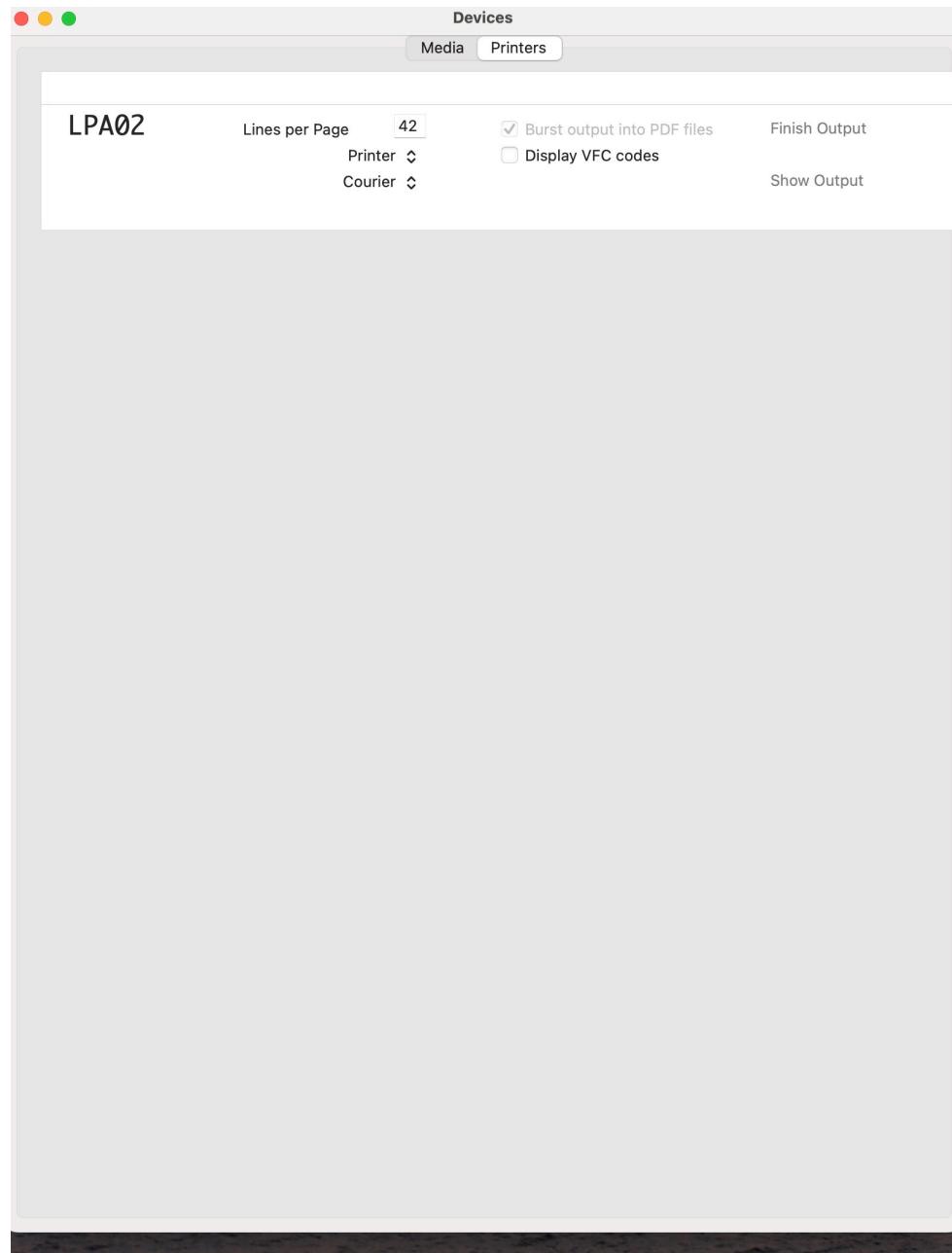
Media and Line Printer Options

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.

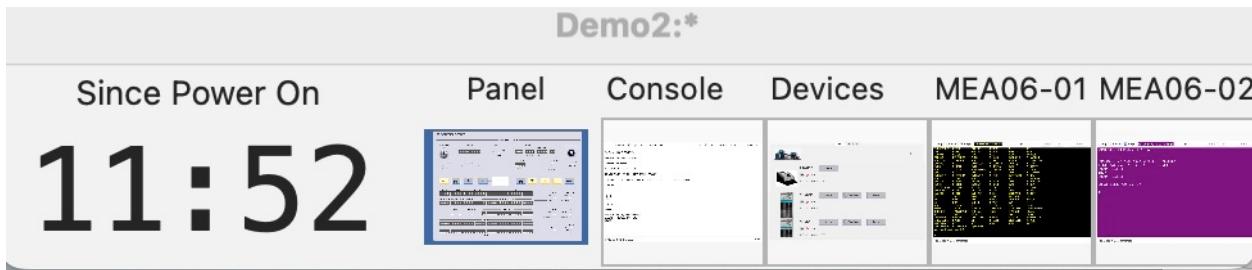


The printer tab can be used to change the lines per page, paper size and font for each printer device.



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.

Snapshots

Existing Snapshots

A snapshot is a copy of the virtual device contents and the hardware configuration. It does not include any state information about the running machine. When a snapshot is restored, it must then be booted. A snapshot can be used to clone a machine by restoring it to a different one.

Snapshots	
F00-With-SST_and_FRES 2024-02-27, 07:26:42 /Users/ms/Desktop/F00.siggy	<button>Restore</button> <button>Delete</button>
F00-With-SST 2024-02-23, 13:30:59 /Users/ms/Desktop/F00.siggy	<button>Restore</button> <button>Delete</button>
F00-Base 2024-02-02, 12:01:02 /Users/ms/Desktop/F00.siggy	<button>Restore</button> <button>Delete</button>

Create New Snapshot

Title

Done Create Snapshot