Sperry*Univac 90/30 Emulator Reference

Introduction

This emulator was developed using the documentation that was publicly available on the Internet and with manuals from Charlie Gibbs' extensive archive of OS/3 material. The copy of OS/3 that the emulator runs is Release 4.2 from 1978 which was found on the Bitsavers web site.

At the time of writing the emulator includes:

- The emulator itself which comprises a virtual maintenance panel and a U100 console.
- An 8416/8418 utility program which allows you to explore virtual disk images.
- A dump/restore program which was used to convert the OS/3 tape image into a virtual disk.
- A 90/30 dis-assembler which was used to dis-assembler parts of the operating system which were used as an aid to debugging the emulator.
- The test data for the Verify System Build (VSB) jobs that was extracted from YSRC.
- A collection of CPU tests that were used to debug the instruction set emulation.
- A small collection of 90/30 and OS/3 manuals that were instrumental to the development of the emulator. There are more manuals available on Bitsavers.

Configuring the Emulator

Most of the emulator's personality is hard coded into the emulator itself. However, it is possible to configure the virtual disks that will available to the operating system. This is accomplished using simple XML files. There are two configuration files supplied with the emulator: REL042.cfg and VSB.cfg. These start the emulator with either the OS/3 4.2 release pack or the VSB SYSRES and data packs mounted. These are sufficient to run the emulator and use OS/3 to run the Verify System Build (VSB) jobs or your own jobs as desired.

To run the emulator using a specific configuration is necessary to supply the -c option on the command line, as show below:

```
U9030 -c REL042.cfg
```

In the absence of the -c option the emulator will start and you will be able to run CPU tests but you will not be able to boot the operating system since no disks will be mounted.

Sample Configuration File

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <iodelay>0</iodelay>
    <iotrace>y</iotrace>
    <svctrace>y</svctrace>
    <ida>
        <disk type='8418' addr='0' file='..\..\Disks\VSBRES.8418' />
        <disk type='8418' addr='1' file='..\..\Disks\SDIVSB.8418' />
    </ida>
</configuration>
```

Parameters:

- **iodelay** sets the delay interval, in milliseconds, for all I/O functions. You can set this to any value greater than or equal to zero. The higher the number, the slower the system will run. If omitted this parameter defaults to 1. If set to zero, the I/O thread will yield control of the processor without forcing a wait. This is enough to make OS/3 work on most systems. The configuration files installed with the emulator have this parameter set to zero.

  This parameter is needed to slow down the I/O on some systems. On these systems OS/3 will become unstable and may not even boot if disc interrupts are received too soon after the SIO instruction.

- **iotrace** enables or disables I/O tracing. A value of y enables tracing, anything else disables tracing. **Note**: Unless you a trying the debug OS/3 or the emulator you should probably disable I/O tracing since the file can get quite large very quickly.

- **svctrace** enables or disables SVC instruction tracing. A value of y enables tracing, anything else disables tracing. **Note**: The SVC tracing code has OS/3 specific knowledge to allow the contents of parameter packets to be dumped. If you are running something other than OS/3 you should disable SVC tracing. Also, unless you are trying to debug OS/3 or the emulator you should probably disable SVC tracing since the file can get quite large very quickly.

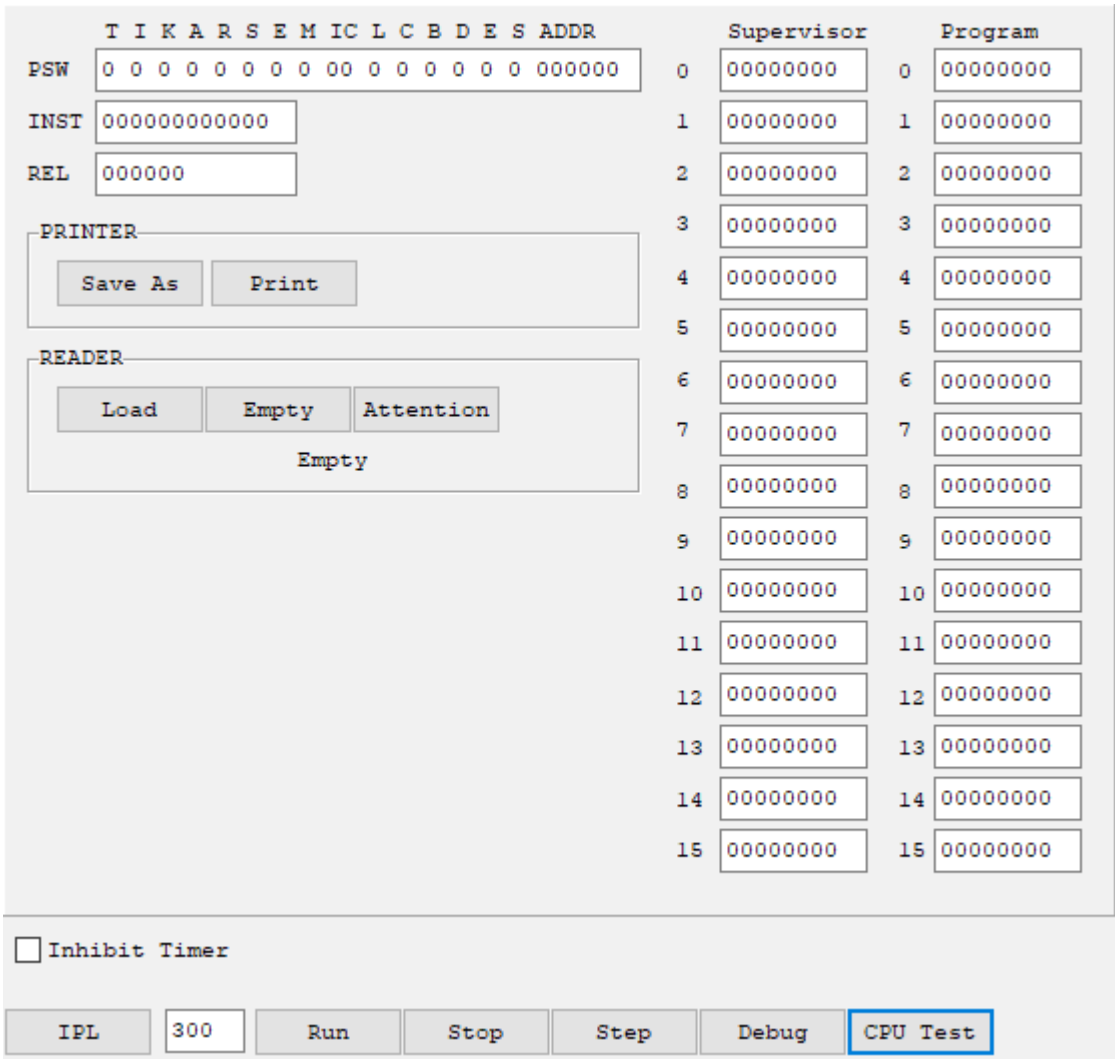
- **ida** is a container for the Integrated Disk Adapter disk configuration.

- **disk** defines a virtual disk.
  - **type** may be either 8416 or 8418
  - **addr** specifies the device address. May be 0 through 7.
  - **file** specifies the path to the virtual disk. If the files does not exist an empty disk will be created. This will need to be initialized by the operating system using the DSKPRP utility before it can be used.

The Emulator

To start the emulator simply click one of the 9030 Emulator short cuts in the start menu. When the emulator starts two windows will open, the console and the virtual maintenance panel.

90/30 Virtual Maintenance Panel



This is where you will interact with the CPU and the various unit record devices like the card reader and printer. You can see the contents of all of the processor's registers. These values are updated periodically as the processor runs. There are a number of buttons that you will use to tell the emulator what to do.

Buttons

IPL

The Initial Program Load (boot) button. This button and the edit box next to it allow you to boot the operating system from any of the configured virtual disks. Simply enter the disk device address into the edit box and click the IPL button. If you are using any of the supplied configuration files the boot device will always be 300. If you don't see any errors when you click the button you will need to switch to the console window to interact with the operation system.

Stop

Once you have IPL'd you can click the Stop button to halt the processor. You will see Halted appear below the register

values.

Run

If the processor is halted clicking this button will start it running again.

Step

Once you have IPL'd, clicking this button will put the processor into single step mode. You will see Halted appear below the register values. By using single step mode you can trace the execution of instructions one at a time. The value of the various registers will be updated for each instruction. To execute the next instruction, click the Run button. To terminate single step mode, click Step again.

This is an extremely tedious way to debug a program or the emulator itself. It is far easier to used the built in debugger.

CPU Test

This button will allow you to run any of the CPU tests. The processor should not be running the operating system when you try to do this. I'm not sure what will happen if you try to run the CPU tests and the OS at the same time, but it won't be good.

Debug

Starts the integrated debugger. This will be discussed in a later chapter.

Printer Save As

Clicking this button allows you to save the current contents of the print capture file to a file of you choice. Once the save is completed the new file will be opened in Notepad.

Reader Load

Clicking this button allows you to load a file into the card reader's hopper. The file can be one of several formats: ASCII text, 16-bit Hollerith or 12-bit Hollerith. Unless you are trying to read object decks you won't need to worry about the Hollerith options. The file extension tells the emulator what format the file is in.

- ASC – unspecified ASCII text
- ASM – assembler source code
- RPG – RPG source code
- H16 – 16 bit Hollerith code
- H80 – 12 bit Hollerith code
- JCL – Job Control

There will probably be new extensions added in the future to denote things like Cobol source, etc.

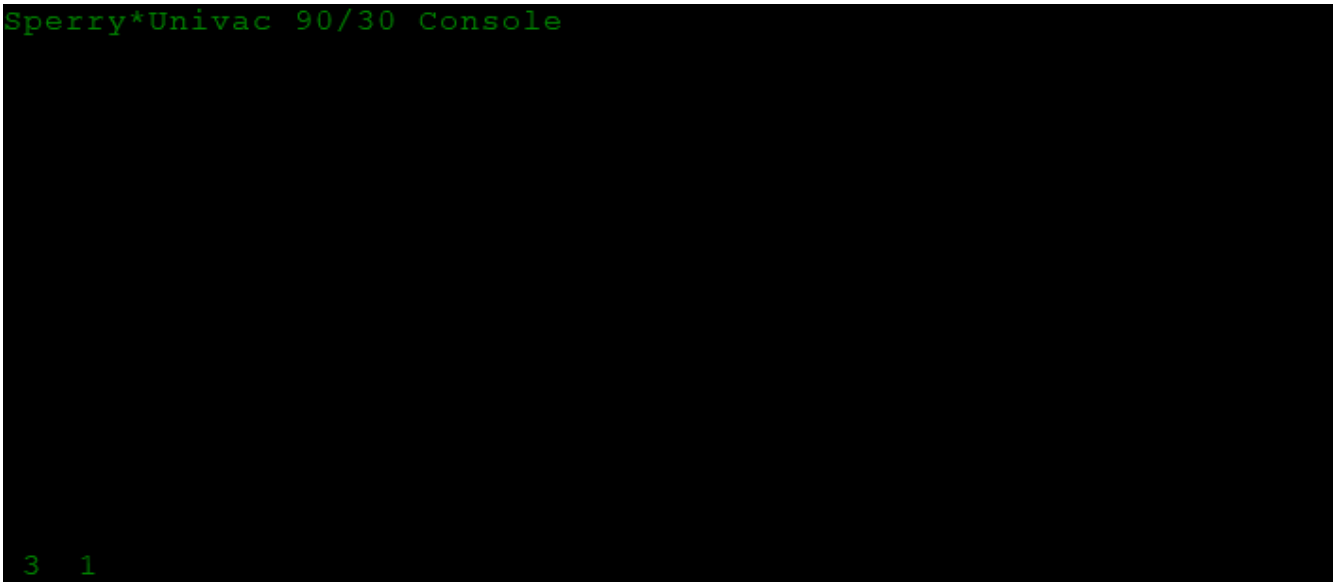
Reader Empty

Clicking this button removes card files from the reader's hopper. Useful if you use the Load button to add the wrong file or a job aborts leaving a partial file in the hopper.

Reader Attention

Clicking this button will send an attention interrupt from the reader to the processor. This will cause OS/3 to attempt to read the cards in the hopper and run them as a job or add them to the input spool file depending on the contents of the first card in the hopper.

90/30 Console



This is where you will interact with the operating system. The console provides a very limited U100 emulation. Just enough, in fact, to satisfy OS/3.

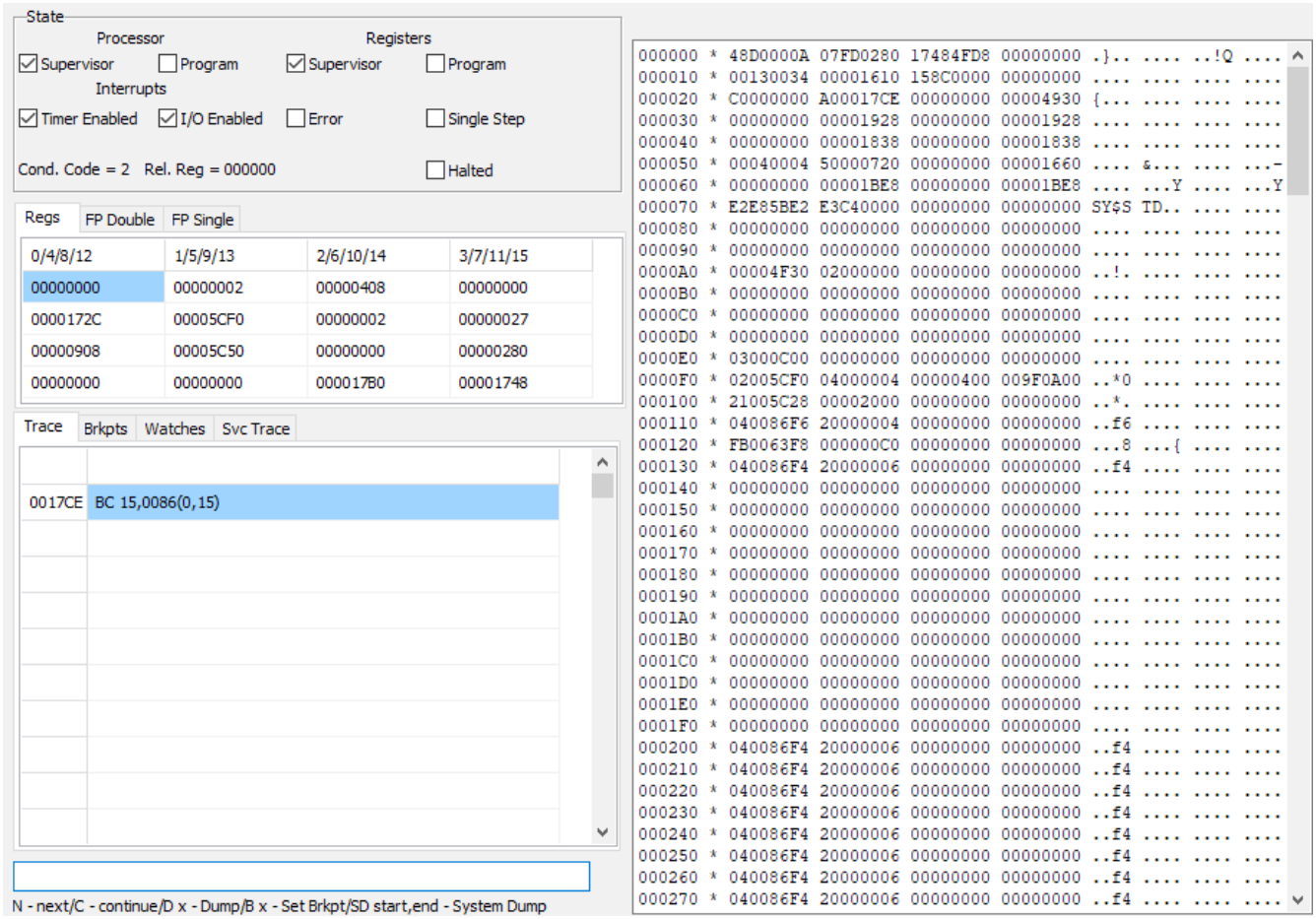
Very few of the Uniscope special function keys are implemented. These are:

- Msg Wait – press F11 to send Msg Wait to the processor. This is the key you will use to signify that you want to enter a console command.
- Transmit – Press F12 to transmit the current contents of the console window to the processor. This key does nothing if you see WAIT in the lower right corner of the window.
- Home – places the cursor in the first unprotected position.
- Up – moves the cursor to an unprotected position on a line above the current cursor location.
- Down – moves the cursor to an unprotected position on a line below the current cursor location.
- Left – moves the cursor to the first unprotected position to the left of the cursor.
- Right – moves the cursor to the first unprotected position to the right of the cursor.
- Backspace – moves the cursor to the first unprotected position to the left of the cursor and overwrites that position with a space.

Debugger

90/30 Debugger

Clicking the Debug button on the virtual maintenance panel will cause this window to appear.



This windows shows you pretty much everything you need to know about the current status of the emulator. The various sections of the window show that status of different aspects of the emulator:

State:

This section shows the current contents of the Program Status Word (PSW). The check boxes are read-only and show which bits are on or off. You can also see the current setting of the condition code and relocation register.

Regs/FP Double/FP Single

This section shows you the contents of the currently selected register set (Program or Supervisor) and the floating point registers. You can view the floating point registers as either double or single precision by clicking the appropriate tab.

Trace/Brkpts/Watches/Svc Trace

This section shows the execution trace (the last 100 instructions executed), the current breakpoints, the current watches and the last 100 SVC instructions executed.

Memory

The section to the right shows the contents of memory. It will always start with the last address specified to the (D)isplay

command. It will display the 1024 bytes following the start address.

Command

The edit box at the bottom of the window is the only part of the window that is not read-only. You use this box to enter debugger commands.

The commands are:

- (N)ext – execute the next instruction.
- (C)ontinue – resume execution at the next instruction and continue until a breakpoint or a watch causes it to stop.
- (D)isplay – display memory starting a the given address. You can give an optional second parameter denoting the base register. If given, the address will be relative to the base register. The address is relative to the current relocation register. e.g. D 1F0. **Note**: All addresses, register numbers and data values are given in hexadecimal.
- (B)reakpoint – toggles the breakpoint at the given address. e.g. B 1F0. If there is not currently a breakpoint for address 1F0 a new one will be created. If a breakpoint already exists it will be deleted.
- (W)atch – toggles the watch at the given address. If a watch already exists for the address it will be deleted. When creating a new watch you must give a second parameter designating what it is you want to wait for. A value of zero will cause the watch to fire whenever the value of the addressed byte changes. A non-zero value will be used as a mask. The watch will fire whenever any of the bits given in the mask are set in the addressed byte. e.g. W 1F0,80 will create a watch that will fire when the first bit of the byte at 1F0 is set to one.
- (M)odify – modifies a single byte of memory. The address is relative to the current relocation register. e.g. M 1F0,80 will set the byte at 1F0 to value 80.
- (S)system(D)ump – dumps the given address range to a disc file. e.g. SD 1F0,200 will dump 16 bytes of memory to disc. You will be asked for the file name.