SIMH FAQ
13-May-2019

COPYRIGHT NOTICE

The following copyright notice applies to the SIMH source, binary, and documentation:

1 General Questions

1.1 What is SIMH?

SIMH is the Computer History Simulation system. It consists of simulators for approximately 20 different computers, all written around a common user interface package and set of supporting libraries. SIMH can be used to simulate any computer system for which sufficient detail is available, but the focus to date has been on simulating computer systems of historic interest.

1.2 Why was SIMH written?

Significant portions of the computing past are being irretrievably lost, as old systems are scrapped, documentation and software is thrown out, media become obsolete or unreadable, and inventors and pioneers die. SIMH was written as a vehicle to allow the computing past to be made accessible to a wider audience, for recreational and educational purposes. SIMH preserves historic computers as portable software that can be run on any modern system. SIMH also preserves representative software packages for these systems. With SIMH, anyone with a desktop computer can call up and run significant samples from the computing past, at any time.

1.3 What is the history of SIMH?

The SIMH project started in 1993, at the suggestion of Larry Stewart of DEC. Its immediate purpose was to preserve the fading hardware and software record of early minicomputers. Since then, the project has been expanded to include other important systems, spanning the history of computing from the late 50's to the late 80's.

SIMH's core design is based on an earlier simulation system called MIMIC. MIMIC was written in the late 1960's at Applied Data Research, by Mike McCarthy, Len Feshkens, and Bob Supnik. MIMIC was a mini-computer simulator that ran on the PDP-10. Its purpose was to facilitate the development and debugging of real-time embedded systems by using the PDP-10 timesharing environment for program development, instead of the limited facilities of the native minicomputer environments. Ironically, given SIMH's mission to preserve the computing record, all machine-readable copies of MIMIC have been lost.

1.4 Who writes and maintains SIMH?

Many people have contributed, and continue to contribute, to SIMH. The full list of contributors can be found on the SIMH web site. Bob Supnik coordinates SIMH development.

1.5 How is SIMH licensed?

SIMH is licensed under a modified X-Windows license. This license allows more or less unrestricted use of the sources and binaries. The license is included with the documentation and is also included in every source module. The software packages are available under various terms and conditions; see the documentation included with each software package.

1.6 How is SIMH distributed?

SIMH is distributed in source form from GitHub in the form of a Zip archive (https://github.com/simh/simh/archive/master.zip) or by directly accessing the GitHub repository

(https://github.com/simh/simh). For Windows users, pre-compiled binaries are also available (https://github.com/simh/Win32-Development-Binaries).

1.7 Which computer systems does SIMH simulate?

SIMH simulates the following computer systems:

Manufacturer	Model
Digital Equipment Corporation	PDP-1, PDP-4, PDP-7, PDP-8, PDP-9, PDP-10, PDP-11, PDP-15, VAX-11/780, MicroVAX 3900, VAX-11/730, VAX-11/750, VAX-8600, MicroVAX I, MicroVAX II, rtVAX 1000 (Industrial VAX 620)
Data General Corporation	Nova, Eclipse
IBM Corporation	1130, 1401, 1620, System 3, 7094 701, 704, 7010/1410, 7070/7074, 7080/702/705/7053, 7090/7094/709/704
GRI Corporation	GRI-909
Honeywell Corporation	H316/516
Hewlett Packard Corporation	HP2116, HP2100, HP21MX, HP3000
Interdata Corporation	16b systems, 7/32, 8/32
Scientific Data Systems	SDS-940, Sigma 32b
MITS	Altair 8080, Altair Z80
Royal-Mcbee	LGP-30, LGP-21
Lincoln Labs	TX-0
Manchester University	SSEM (Small Scale Experimental Machine)
Control Data Corporation	CDC1700
Burroughs	B5500

The documentation contains more details on supported models and peripherals.

1.8 Which host systems does SIMH run on?

Host System	Compiler	comments
OpenVMS/VAX	DEC C	no 64b support; no Ethernet support
OpenVMS/Alpha	DEC C	Ethernet support provided in pcap-vms
OpenVMS/IA64	DEC C	Ethernet support provided in pcap-vms
Windows 9x or Windows 2000 or Windows XP	Mingw/gcc or Visual C++	requires Winpcap for Ethernet support
Windows Vista Windows 7 Windows 8 Windows 10		requires Npcap for Ethernet support
Mac OS/X		requires libpcap for Ethernet support
Linux	gcc	requires libpcap for Ethernet support
Tru64 UNIX	DEC C	no Ethernet support
AIX		requires libpcap for Ethernet support
Solaris		requires libpcap for Ethernet support

```
HP/UX                                requires libpcap for Ethernet support
NetBSD            gcc                requires libpcap for Ethernet support
OpenBSD           gcc                requires libpcap for Ethernet support
FreeBSD           gcc                requires libpcap for Ethernet support

OS/2              EMX                no Ethernet support
```

1.9 What software packages are available to run on SIMH?

The list of available software packages can be found on the SIMH web site.

1.10 Where can I get more information on SIMH?

The SIMH web site is http://simh.trailing-edge.com and via discussion on the SIMH mailing list.

1.11 How do I search the simh mailing list archive for questions that might have been asked and answered before?

Google is your friend here. For example:

Perform a Google search for: rsts archive -V9 site:mailman.trailing-edge.com/pipermail/simh/

This finds all messages containint “rsts” “archive” but not “V9”.

2 Operational Questions

2.1 How do I install SIMH on Windows?

The simplest way is to download the pre-compiled binaries. Unzip these into the directory where you want to run SIMH. You can then run whichever binary that you want.

2.2 How do I install SIMH with Ethernet support on Windows?

The pre-compiled binaries will provide Ethernet support depending on whether or not the Npcap (or WinPCAP) package has been installed on the host computer. If you want to run with Ethernet support, you must download and install the Npcap AutoInstaller from

```
https://nmap.org/npcap/#download
```

This creates a network packet driver in Windows for SIMH to attach to.

To use network support, you must either be an administrator on the Windows machine (implied in Windows 9X), or you must set the windows packet driver to autostart when the system boots. The NpcapInstaller has an option (which defaults to true) which will automatically start the Network Packet Filter Driver.

2.3 How do I install SIMH on Unix/Linux/OSX?

Ethernet support is available only on Linux, OSX, NetBSD, OpenBSD, FreeBSD and Solaris.

- Unzip the archive of sources to a new directory. You must specify the -a switch to unzip for proper conversion of Windows cr-lf sequences to UNIX newline sequences.

- The makefile included in the simh source distribution is a GNU make makefile. Some systems have GNU make as the default make (i.e. Linux). If your system's make is not GNU make, then be sure to install the GNU make package first and then use the 'gmake' command instead of 'make':

```
% make all
```

- If you want Ethernet support in the PDP-11, VAX or VAX780, you should install your OS distribution's libpcap-devel package prior to building the simulator:
- More details about Ethernet installation and configuration can be found in the 0readme_ethernet.txt file in the simh sources.

2.4 How do I install SIMH on VMS?

Download the SIMH source kit, and UNZIP it in the directory that you want SIMH to reside in. Unpack it and set the file attributes as follows:

```
$ unzip simh-package.zip
$ set default [.directory-containing scp.c]
$ set file/attri=RFM:STM makefile,*.mms,[...]*.c,[...]*.h,[...]*.txt
```

Simulators with ethernet network devices (All the VAX simulators and the PDP11) can have functioning networking when running on Alpha or IA64 OpenVMS. In order to build and run simulators with networking support, the VMS-PCAP package must be available while building your simulator. The vms-pcap.zip file can be downloaded from https://github.com/downloads/simh/simh/vms-pcap.zip. The vms-pcap.zip file should be unpacked as follows:

```
$ unzip –aa vms-pcap.zip
```

The PCAP-VMS components are presumed (by the descript.mms file) to be located in a directory at the same level as the directory containing the simh source files. For example, if these exist here:

```
[]descrip.mms
[]scp.c
etc.
```

Then the following should exist:

```
[-.PCAP-VMS]BUILD_ALL.COM
[-.PCAP-VMS.PCAP-VCI]
[-.PCAP-VMS.PCAPVCM]
etc.
```

To build simulators:

On a VAX use:

```
$ MMx
```

On a Alpha & IA64 hosts use:

```
$ MMx                          ! With Ethernet support
$ MMx/MACRO=("NONETWORK=1)     ! Without Ethernet support
```

```
UNZIP can be found on the VMS freeware CDs, or from www.info-zip.org
MMS (Module Management System) can be licensed from HP/Compaq/Digital as part of
the VMS Hobbyist program (it is a component of the DECSET product).
MMK can be found on the VMS freeware CDs, or from
http://www.kednos.com/kednos/Open_Source/MMK
DEC C can be licensed from HP/Compaq/Digital as part of the VMS Hobbyist
program.
```

Note that the PDP-10, I7094 and Eclipse emulators cannot be built and used on VAX/VMS, because the DEC C compiler for VAX/VMS does not support 64-bit integers. OpenVMS DEC C Alpha and IA64 systems has the required 64-bit capability to build and run all of the emulators. Ethernet support is only available on Alpha VMS 7.3-1 and above.

2.5 How do I transcribe a real CD for use with SIMH?

- First, you may be able to access a real CD directly from within a simulator using RAW device mode to access the device. On Linux, and many Unix variants, support direct access to the CD ROM from SIMH:

```
sim> set rq1 cdrom
sim> att rq1 /dev/cdrom_drive
```

- On Windows, the equivalent would be:

```
sim> set rq1 cdrom
sim> att rq1 //./cdrom0
```

- As for actually making an ISO image of a CD to, On UNIX, you can copy a CD to an ISO file with the dd command:

```
% dd /if=/dev/raw_cd_device /out=/path/cdimage.iso
```

- Linux, and many Unix variants, support direct access to the CD ROM from SIMH:

```
sim> set rq1 cdrom
sim> att rq1 -f simh /dev/cdrom_drive
```

- On Windows, there are quite a few products that can do this. The two most common products are detailed below. Make sure to disable any antivirus software before proceeding. Antivirus software tends to interfere with the smooth flow of data from the CD and will occasionally transform the data in strange and unexpected ways to 'protect' you. You may also need to limit the read speed. Some burnt CD-Rs do not read correctly at the highest rate of speed, depending on the accuracy of the burner; pressed CD-ROMs should not have this problem.

  1) Roxio
  A) EZ-CD Creator 5.x
  Go to the the Disc menu and select Disc Info (there will be a delay).
  Select the track shown, then click the Read Track button.
  Enter the Save file name, then OK.
  B) Easy Media Creator 7.x
  Go to Creator Classic

Select Other Tasks | Disc and Device Utility
Drill down on the device until you find the data track, then select it
Click the 'Read Track...' button
Enter the save file name, then OK.

2) Nero 5.5
Select Recorder|Save Track
Select the track, set the output filename
Click GO

2.6 How do I transcribe other archival media for use with SIMH?

You must have access to a real system that can read the media to be transcribed (e.g., a system with a working DECtape drive to read a DECtape). Most systems have utilities to copy raw data to a disk file; that file can then be transferred over the console serial line to a system with an Internet link. Utility programs are available to convert raw data streams to SIMH format.

2.7 How can I get text files in and out of SIMH?

2.7.1 Tape input

Files coming into a simulated system can easily be achieved by using a simulated tape drive. This mechanism allows local host system files to be presented to the simulated system as a set of files on an ANSI labeled tape. If the simulated system's operating system knows how to read ANSI labeled tape this is a good choice.

```
sim> att ts0 -f ANSI-VMS file.txt,file.bin,*.c
sim> att ts0 -f ANSI-RSX11 file.txt,file.bin,*.c
sim> att ts0 -f ANSI-RSTS file.txt,file.bin,*.c
sim> att ts0 -f ANSI-RT11 file.txt,file.bin,*.c

$ MOUNT MSA0: SIMH
%MOUNT-I-WRITELOCK, volume is write locked
%MOUNT-I-MOUNTED, SIMH mounted on _MSA0:
$ TYPE MSA0:file.txt
```

2.7.2 Serial input and output

Since SIMH supports the universal serial interface using TELNET, text can be transferred using one of the serial line transfer protocols (X/Y/Zmodem, Kermit) or using standard cut and paste techniques, if the host's TELNET program supports it.

To use the TELNET feature, connect to the SIMH machine using TELNET, and set the target environment into a 'receive' mode. This is usually something like running a text editor. Then tell the TELNET program to 'send', 'transfer', or 'paste' the text that you want sent into the SIMH system.

To get text out of the system, have the TELNET program either log the output, or if the TELNET program supports a backscroll region you can use that. Tell the SIMH system to 'type' or 'cat' the text file, sending the output to the TELNET device, where you can edit it into a text file.

Many TELNET programs also support transferring large files via X/Y/ZModem or Kermit, which you can use as long as the SIMH system has the appropriate matching program.

C-Kermit from Columbia University (http://www.columbia.edu/kermit) is probably the most universal way to transfer files in and out of SIMH systems.

2.7.3 Paper Tape input and output

Some systems have paper tape devices which can be attached to a file on the host system and be read within the simulated system. Systems with a paper tape reader also have a paper tape punch which can be used for output.

2.7.4 Ethernet input and output

If the SIMH system supports Ethernet connectivity (PDP-11, VAX), you can also use the various network copy programs (FTP, DECNET) to transfer files.

2.7.5 Printer output

Finally, you can "print" text files to the simulated line printer. Printer output is automatically formatted as an ASCII text file.

2.8 How can I get binary files in and out of SIMH?

2.8.1 Serial input and output

Since SIMH supports the universal serial interface using TELNET, binary files can be transferred using one of the serial line transfer protocols (X/Y/ZModem, Kermit) or by converting the binary to a text-encoded file (HEXify, UUENCODE, VMShare, etc.) and transferred in text mode (see section 2.7).

Many TELNET programs also support transferring large files via X/Y/ZModem or Kermit, which you can use as long as the SIMH system has the appropriate matching program.

C-Kermit from Columbia University (http://www.columbia.edu/kermit) is probably the most universal way to transfer files in and out of SIMH systems.

2.8.2 Ethernet input and output

If the SIMH system supports Ethernet connectivity (PDP-11, VAX), you can also use the various network copy programs (FTP, DECNET) to transfer files.

2.9 Can I connect real devices on the host computer to SIMH?

Currently Ethernet devices, Serial Ports and physical disks and/or CDROMs are the devices which can be accessed directly from simh simulators.

2.10 My Windows host can't communicate with the PDP-11 or VAX over Ethernet; why?

Current versions of the simulators will directly allow you to communicate with both other systems on your LAN and your host computer by attaching the simulator's XQ device to the primary network interface on your host.

2.11 My Linux, OSX or other Unix host can't communicate with the PDP-11 or VAX over Ethernet; why?

The network stacks on these systems don't naturally receive packets which are transmitted with the pcap_sendpacket API. This issue can be accommodated in one of four ways:

1) Use NAT mode for your network connection. This mode only allows TCP/IP traffic to flow between the systems (No DECnet, LAT or other proprietary LAN protocols), which will be fine for single simulator situations.
2) add a second Ethernet controller, attach both controllers to the same switch or hub, and attach SIMH to the second controller. Don't configure any host network protocols on the second Ethernet controller. The host and SIMH will now be able to communicate across the physical network connection
3) If the host has internal (kernel level) network bridging support, then the host's network configuration can be setup to allow direct communication between the host and the simulated system. The simh networking layer can accommodate tun/tap and/or vde networking to achieve this. Details of how this is done and which hosts it can work on can be found in the 0readme_ethernet.txt file in the simh zip file.
4) Enable 2 XQ (or XU) devices in the simulator and use one in NAT mode to talk to the host system and connect the other to LAN for simulator to simulator communications and other LAN protocols.

2.12 How can I use my wireless Ethernet card with SIMH?

The best approach here is to use NAT mode on your network connection. This will work fine for simulators using TCP/IP to talk to either their host system and/or to reach the Internet. Meanwhile, as for directly using the wireless network card the following are some of the considerations:

Wireless Ethernet is something of a misnomer - it "works like" Ethernet. Wireless cards behave differently than real Ethernet cards in promiscuous mode. Some wireless cards can't operate in promiscuous mode but can sometimes be successfully used with existing SIMH code. Sometimes this will also depend on functionality provided by the wireless router you may be connected to. Many wireless routers will not be well behaved when you attempt this.

One of the caveats is that the simulated machine cannot run any software which changes the simulated MAC address, or the network connection will stop working. For example, DECNET Phase IV (or Phase V in compatibility mode) tries to change the MAC of the network card to AA-00-04-xx-xx-xx. Nor can you preset the wireless MAC address to the anticipated target DECNET address using something like SMAC to get DECNET to work - DECNET will see the MAC already preset to the required DECNET address and generate an invalid media (duplicate address) fault.

Otherwise, TCP/IP, LAT, VMS Clustering, and DECNET Phase V in non-compatibility mode work fine.

To get wireless cards to work with SIMH, set the simulated MAC to be the same as the MAC of the wireless card. An example:

```
c:\> IPCONFIG/ALL
```

```
Windows 2000 IP Configuration

  Host Name . . . . . . . . . . . . : LLOH3-EXP29189
  Primary DNS Suffix  . . . . . . . : ad.tasc.com
  Node Type . . . . . . . . . . . . : Hybrid
  IP Routing Enabled. . . . . . . . : No
  WINS Proxy Enabled. . . . . . . . : No
  DNS Suffix Search List. . . . . . : ad.tasc.com

Ethernet adapter Local Area Connection:

  Connection-specific DNS Suffix  . :
  Description . . . . . . . . . . . : D-Link DWL-650+ Wireless Cardbus
  Physical Address. . . . . . . . . : 00-80-C8-08-CE-DB   <-- MAC address
  DHCP Enabled. . . . . . . . . . . : No
  IP Address. . . . . . . . . . . . : 192.168.0.5
  Subnet Mask . . . . . . . . . . . : 255.255.255.0
  Default Gateway . . . . . . . . . :
  DNS Servers . . . . . . . . . . . :
  Primary WINS Server . . . . . . . : 132.228.188.100
  Secondary WINS Server . . . . . . : 132.228.196.98

c:\> VAX
VAX simulator V3.2-1
sim> DO VAX_CONFIG.DO               <-- setup VAX as normal
sim> SET XQ MAC=00-80-C8-08-CE-DB   <-- set XQ MAC to wireless MAC address
sim> B CPU                          <-- and continue...
```

2.13 Why doesn't simh idling work on my Unix host?

Some host systems have default clock tick sizes which are greater than what is required to produce useful simh idling behavior.Useful Idling depends on a simulators ability to sleep for intervals which are less than or equal to the simulated system's clock tick. Best idling behavior is realized when sleep intervals can be as small as 1ms. When a simulator starts, simh determines the host system's clock tick size and based on this determination, idling will be supported or not. The SHOW VERSION command will display (among other things) the .host OS's clock tick size which simh has determined. On some platforms (Windows), the host system's clock tick size can be dynamically changed by non-privileged user mode code. On Windows systems, simh sets the OS clock tick size to 1ms so that idling can be supported. Changing the OS tick size on other platforms may be achieved in some system specific way.

On Solaris, the file /etc/system contains parameters used to adjust various operating system details. Adding the following lines to this file and rebooting the system will set the OS clock tick to 1ms and simh will have idling support:

```
set hires_tick=1
set hires_hz=1000
```

3 Writing and Debugging New Code

3.1 What resources are available for writing new simulators?

The SIMH web site contains documentation on the internals of SIMH, as well as specific help for writing new peripherals for several of the popular simulators. The doc directory in the github simh source code has a file doc/simh.doc which describes the internal APIs used by simulator writers.

3.2 What debugging facilities are available in SIMH?

Most simulators provide the following debugging capabilities:

- Symbolic assembly and disassembly of memory contents.
- Numeric examination and modification of the data store of any simulated device.
- Numeric search on both memory and device data.
- Visibility to simulator internal structures, such as the event queue.
- An unlimited number of instruction breakpoints.
- Proceed counts on breakpoints.
- Automatic execution of simulator commands on a breakpoint.
- Stepped execution (from single step to 'n' steps).
- A PC change queue, usually 64 instructions deep.
- Instruction execution history recording and display.

Specific simulators may provide additional features, such as an instruction history buffer, CPU and/or device logging, and breakpoints on memory reads and writes.

3.3 When do I need to use the host debugger for debugging a simulator?

While a simulator is being debugged, its execution of instructions or debugging support code may be unreliable. During this process, the programmer may need to use the host debugger to stop in the middle of an instruction execution, or to trap an error condition. Host debugger breakpoints should be invisible to the simulator; with the exception of clock calibration, all simulator events are driven off the event queue rather than real-world events.

If the programmer needs to force a simulator stop from the host debugger, most simulators provide an "address stop" global variable. Setting this variable to 1 will cause the simulator to stop after completing the current instruction.

3.4 What is the release process for SIMH?

The latest development code is available on the public source code repository at https://github.com/simh/simh/archive/master.zip. Since the latest code is always publicly available and bugs are generally fixed somewhat quickly, there hasn't been a driving need for formal releases.

Bob Supnik's original development efforts are tracked in the Supnik-Current branch of the github repository. Any changes that he makes there are merged directly into the master branch whenever he provides his current state.

Bob's original development activites released new versions of SIMH whenever a significant number of new features, or important bug fixes, has accumulated. The major version number only changes when there is a major restructuring of SIMH's internal structures. The minor version number is changed when the format of the save/restore file must be updated.

4 VAX

4.1 Where can I get software and hobbyist licenses for the VAX?

HP (formerly Compaq formerly DEC) provides licenses to OpenVMS for hobbyist use. A description of the hobbyist license program can be found on http://www.openvmshobbyist.com.

4.2 How do I install VMS?

To install VMS, you will need a distribution CD ROM. Any version after VMS 5.5-2 should run on the MicroVAX 3900 simulator.

- Transcribe the distribution CD ROM to an ISO-format CD image file. (See question 2.5 for information on how to do this.)
- Set drive RQ1 to be a CD ROM.
- Attach the CD ROM image file to simulated drive RQ1.
- Set drive RQ0 to be the type of disk you want. Be sure that the disk is large enough to hold VMS.
- Attach a blank disk image file to simulated drive RQ0.
- Boot the CPU.
- When the self-test code completes, boot the CD ROM.
- Use standalone backup to restore the CD ROM contents to the simulated disk.

```
sim> set rq0 rd54
sim> set rq1 cdrom
sim> att rq0 new_vms.dsk
sim> att rq1 cd_rom_image.iso
sim> boot cpu
:
>>> boot rq1

$ (prompt from standalone backup)
```

A writeup on the procedure can be found on the VMS hobbyist site.

4.3 How do I install NetBSD?

Directions for installing NetBSD on the NetBSD web site, at http://www.netbsd.org/Ports/vax/emulator-howto.html.

4.4 How do I install Ultrix?

Ultrix is not presently licensed for hobbyist use. If you have a valid license for Ultrix, and distribution tapes for a version that supports the MicroVAX 3900 series (V4 or later), then you should be able to install Ultrix on the simulator.

- Transcribe the distribution tapes to SIMH-format tape image files. (See question 2.6 for information on how to do this.)
- Mount the installation tape image on simulated drive TQ0.
- Set drive RQ0 to be the type of disk you want. Be sure that the disk is large enough to hold Ultrix.
- Mount a blank disk image file on simulated drive RQ0.
- Boot the CPU.
- When the self-test code completes, boot the installation tape.

- The installation tape will guide you through the installation of Ultrix.

```
sim> set rq0 rd54
sim> att rq0 new_vms.dsk
sim> att tq0 ultrix_install.tap
sim> boot cpu
   :
>>> boot mua0

(Ultrix installation dialog)
```

4.5 What's the CPU serial number for my hobbyist license PAK?

On a MicroVAX 3900, the CPU serial number is not readable and can be an arbitrary value. 12345 will work fine.

4.6 How do I import and make my hobbyist license PAKs usable?

See How can I import files to a simulated VMS environment? and How do I make imported files readable on a simulated VMS system?

4.7 How do I change the simulator from a VAXserver 3900 to a MicroVAX 3900?

To change the type between a MicroVAX 3900 and a VAXServer 3900 use the following commands:

```
sim> set cpu model=VAXServer
sim> set cpu model=MicroVAX
```

and boot the simulated VAX.

4.8 Is there an example of the simulator running VMS?

This example assumes you are trying to emulate a MicroVAX 3900 with 64MB of memory, with a single 1GB disk drive, a CDROM, and an Ethernet controller.

The host OS is Windows NT/2000/XP, and you have previously dumped the contents of the VMS Hobbyist CD to a disk file as detailed in 2.5, and have loaded npcap/WinPCAP on the system for Ethernet support. Other host OS's will look similar but will have different file name syntax.

```
c:\simh> vax                          ; run VAX emulator
sim> set cpu 64m                      ; set memory size to 64MB
sim> load -r vax\ka655x.bin           ; load the MicroVAX 3900 console ROM
sim> attach NVR vax\ka655.nvr         ; create/load a Non-Volatile RAM file
sim> set LPT disable                  ; disable devices we don't want/need
sim> set TQ disable                   ;   "
sim> set rq0 ra90                     ; set disk 0 to 1GB (RA90 size)
sim> attach rq0 vax\vaxsys.dsk        ; create/use disk file
sim> set rq1 rrd40                    ; set disk 1 as a cdrom
sim> attach -r rq1 vax\hobbyist.dsk   ; attach cdrom dump file as read-only
sim> set rq2 offline                  ; turn off disk rq2
```

```
sim> set rq3 offline                ; turn off disk rq3
sim> attach xq eth0                 ; attach to host ethernet controller
sim> b cpu                          ; start (boot) VAX console

KA655-B V5.3, VMB 2.7
 1) Dansk                           ; will not appear if the controlling
    ..                              ; keyboard doesn't support multi-
15) Svenska                         ; national characters!
 (1..15): 5
Performing normal system tests.
40..39..38..37..36..35..34..33..32..31..30..29..28..27..26..25..
24..23..22..21..20..19..18..17..16..15..14..13..12..11..10..9..
8..7..6..5..4..3..
Tests completed.
>>> show device                     ; tell console to show all devices
UQSSP Disk Controller 0 (772150)
-DUA0 (RA90)
-DUA1 (RRD40)

Ethernet Adapter 0 (774440)
-XQA0 (08-00-2B-AA-BB-CC)
>>> b dua1                          ; tell console to boot cdrom
(BOOT/R5:1 DUA1)

2..1..0
```

4.9 How can I import files to a simulated VMS environment?

4.9.1 The Easy Way

- Present the files to the VMS system using a pseudo tape drive with the ANSIFILES tape format. This approach is both easy and achieves direct access by the VMS system in a single step.

```
c:\simh> vax                        ; run VAX emulator
sim> attach ts0 -f ansi-vms Hobbyist-USE-ONLY-VA.TXT,*.exe,

within the running simulator:

$ mount MSA0: SIMH
%MOUNT-I-MOUNTED, SIMH mounted on _MSA0:
$ @MSA0:Hobbyist-USE-ONLY-VA.TXT
```

4.9.2 Alternatively

- Use a CD burner program, like Easy CD Creator or Nero, to create an ISO 9660 CD image containing the files you want to import. Note that file names are limited to DOS '8.3' conventions.
- Attach the simulated CD image to a simulated CD drive.
- Mount the simulated CD as an ISO 9660 file system under VMS.
- Copy the files you need from the simulated CD to the simulated disk.

(Thanks to Tim Stark for this suggestion.)

4.10 How do I make imported files readable on a simulated VMS system?

Files imported using the ANSITAPE paradigm are directly usable without further manipulation.

Some imported files may need to have their file attributes set appropriately in order to be easily usable in the simulated VMS system. This may be the case for text files which come from Unix or Windows systems that have LF or CRLF line endings and may have been transported to the VMS system via binary network transport OR via a CD image. DIRECTORY/FULL will display the file's attributes. A file transferred in binary mode will likely have record attributes that say: "Fixed 512 byte records".

A file's record attributes can be changed to handle text files with LF line endings with

```
$ SET FILE/ATTRIBUTE=RFM:STMLF
```

A file's record attributes can be changed to handle text files with CRLF line endings with:

```
$ SET FILE/ATTRIBUTE=RFM:STM
```

This will work with the latest version of VMS, but earlier versions didn't have the SET FILE/ATTRIBUTE command.

4.11 How can I export files from a simulated VMS environment?

- Utility ODS2 (available on the Web) can read an ODS-2 disk image and copy files from that image to the host file system.
- Text files can be printed to the simulated line printer, as described above.

5 PDP-11

5.1 When installing RSTS/E from simulated magnetic tape, the installation process hangs with no error message; why?

RSTS/E installation from magnetic tape requires that the tape be write locked.