# VAX (MicroVAX 3900) Simulator Usage
# 01-May-2012

**COPYRIGHT NOTICE**

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

> Original code published in 1993-2012, written by Robert M Supnik
> Copyright (c) 1993-2012, Robert M Supnik
>
> Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
>
> The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
>
> THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL ROBERT M SUPNIK BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
>
> Except as contained in this notice, the name of Robert M Supnik shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization from Robert M Supnik.

This memorandum documents the DEC VAX (MicroVAX 3900) simulator.

# 1 Simulator Files

To compile the VAX, you must define VM_VAX and USE_INT64 as part of the compilation command line. To enable extended file support (files greater than 2GB), you must define USE_ADDR64 as part of the command line as well.

| | |
|---|---|
| sim/ | scp.h |
| | sim_console.h |
| | sim_defs.h |
| | sim_ether.h |
| | sim_fio.h |
| | sim_rev.h |
| | sim_sock.h |
| | sim_tape.h |
| | sim_timer.h |
| | sim_tmxr.h |
| | scp.c |
| | sim_console.c |
| | sim_ether.c |
| | sim_fio.c |
| | sim_sock.c |
| | sim_tape.c |
| | sim_timer.c |
| | sim_tmxr.c |
| sim/vax/ | vax_defs.h |
| | vaxmod_defs.h |
| | vax_cis.c |
| | vax_cmode.c |
| | vax_cpu.c |
| | vax_cpu1.c |
| | vax_fpa.c |
| | vax_io.c |
| | vax_mmu.c |
| | vax_octa.c |
| | vax_stddev.c |
| | vax_sys.c |
| | vax_syscm.c |
| | vax_sysdev.c |
| | vax_syslist.c |
| sim/pdp11/ | pdp11_cr_dat.h |
| | pdp11_mscp.h |
| | pdp11_uqssp.h |
| | pdp11_xq.h |
| | pdp11_xq_bootrom.h |
| | pdp11_cr.c |
| | pdp11_dz.c |
| | pdp11_lp.c |
| | pdp11_rl.c |
| | pdp11_ry.c |
| | pdp11_rq.c |

pdp11_tq.c
pdp11_ts.c
pdp11_vh.c
pdp11_xq.c

Additional files are:

| | | |
|---|---|---|
| sim/vax/ | ka655x.bin | extended memory boot ROM code |

# 2 VAX Features

The VAX simulator is configured as follows:

| device name(s) | simulates |
|---|---|
| CPU | KA655"X" CPU with 16MB-512MB of memory |
| TLB | translation buffer |
| ROM | read-only memory |
| NVR | non-volatile memory |
| QBA | Qbus adapter |
| SYSD | system devices |
| TTI,TTO | console terminal |
| CLK | real-time clock |
| DZ | DZV11 4-line terminal multiplexer (up to 4) |
| VH | DHQ11 8-line terminal multiplexer (up to 4) |
| CR | CR11 card reader |
| LPT | LPV11 line printer |
| RL | RLV12/RL01(2) cartridge disk controller with four drives |
| RQ | RQDX3 MSCP controller with four drives |
| RQB | second RQDX3 MSCP controller with four drives |
| RQC | third RQDX3 MSCP controller with four drives |
| RQD | fourth RQDX3 MSCP controller with four drives |
| RY | RXV21 floppy disk controller with two drives |
| TS | TSV11/TSV05 magnetic tape controller with one drive |
| TQ | TQK50 TMSCP magnetic tape controller with four drives |
| XQ | DELQA/DEQNA Ethernet controller |
| XQB | second DELQA/DEQNA Ethernet controller |

The CR, DZ, VH, LPT, RL, RQ, RQB, RQC, RQD, RY, TS, TQ, XQ, and XQB devices can be set `DISABLED`. RQB, RQC, RQD, VH, and XQB are disabled by default.

The VAX simulator implements several unique stop conditions:

- Change mode to interrupt stack
- Illegal vector (bits<1:0> = 2 or 3)
- Unexpected exception during interrupt or exception
- Process PTE in P0 or P1 space instead of system space
- Unknown IPL
- Infinite loop (BRB/W to self at IPL 1F)

The `LOAD` command supports a simple binary format, consisting of a stream of binary bytes without origin or checksum, for loading memory, the boot ROM, or the non-volatile memory. The `DUMP` command is not implemented.

## *2.1 CPU and System Devices*

### 2.1.1 CPU

CPU options include the size of main memory and the treatment of the HALT instruction.

```
SET CPU 16M                 set memory size = 16MB
SET CPU 32M                 set memory size = 32MB
SET CPU 48M                 set memory size = 48MB
SET CPU 64M                 set memory size = 64MB
SET CPU 128M                set memory size = 128MB
SET CPU 256M                set memory size = 256MB
SET CPU 512M                set memory size = 512MB
SET CPU SIMHALT             kernel HALT returns to simulator
SET CPU CONHALT             kernel HALT returns to boot ROM console
```

The CPU implements a show command to display the I/O address map:

```
SHOW CPU IOSPACE            show I/O space address map
```

The CPU also implements a command to display a virtual to physical address translation:

```
SHOW {-kesu} CPU VIRTUAL=n  show translation for address n
                            in kernel/exec/supervisor/user mode
```

Notes on memory size:

- The real KA655 CPU only supported 16MB to 64MB of memory. The simulator implements a KA655"X", which increases supported memory to 512MB.
- The firmware (ka655x.bin) contains code to determine the size of extended memory and set up the PFN bit map accordingly. Other than setting up the PFN bit map, the firmware does not recognize extended memory and will behave as though memory size was 64MB.
- If memory size is being reduced, and the memory being truncated contains non-zero data, the simulator asks for confirmation. Data in the truncated portion of memory is lost.
- If the simulator is running VMS, the operating system may have a SYSGEN parameter set called PHYSICAL PAGES (viewable from "MCR SYSGEN SHOW PHYSICALPAGES"). PHYSICALPAGES limits the maximum number of physical pages of memory the OS will recognize. If it is set to a lower value than the new memory size of the machine, then only the first PHYSICALPAGES of memory will be recognized, otherwise the actual size of the extended memory will be realized by VMS upon each boot. Some users and/or sites may specify the PHYSICALPAGES parameter in the input file to AUTOGEN (SYS$SYSTEM:MODPARAMS.DAT). If PHYSICALPAGES is specified there, it will have to be adjusted before running AUTOGEN to recognize more memory. The default value for PHYSICALPAGES is 1048576, which describes 512MB of RAM.

Initial memory size is 16MB.

Memory can be loaded with a binary byte stream using the `LOAD` command. The `LOAD` command recognizes three switches:

```
-o              origin argument follows file name
-r              load the boot ROM
-n              load the non-volatile RAM
```

The CPU supports the BOOT command and is the only VAX device to do so. Note that the behavior of the bootstrap depends on the capabilities of the console terminator emulator. If the terminal window supports full VT100 emulation (including Multilanguage Character Set support), the bootstrap will ask the user to specify the language; otherwise, it will default to English.

These switches are recognized when examining or depositing in CPU memory:

```
-b              examine/deposit bytes
-w              examine/deposit words
-l              examine/deposit longwords
-d              data radix is decimal
-o              data radix is octal
-h              data radix is hexadecimal
-m              examine (only) VAX instructions
-p              examine/deposit PDP-11 (compatibility mode) instructions
-r              examine (only) RADIX50 encoded data
-v              interpret address as virtual, current mode
-k              interpret address as virtual, kernel mode
-e              interpret address as virtual, executive mode
-s              interpret address as virtual, supervisor mode
-u              interpret address as virtual, user mode
```

CPU registers include the visible state of the processor as well as the control registers for the interrupt system.

| name | size | comments |
|---|---|---|
| PC | 32 | program counter |
| R0 .. R14 | 32 | R0 to R14 |
| AP | 32 | alias for R12 |
| FP | 32 | alias for R13 |
| SP | 32 | alias for R14 |
| PSL | 32 | processor status longword |
| CC | 4 | condition codes, PSL<3:0> |
| KSP | 32 | kernel stack pointer |
| ESP | 32 | executive stack pointer |
| SSP | 32 | supervisor stack pointer |
| USP | 32 | user stack pointer |
| IS | 32 | interrupt stack pointer |
| SCBB | 32 | system control block base |
| PCBB | 32 | process controll block base |
| P0BR | 32 | P0 base register |
| P0LR | 22 | P0 length register |
| P1BR | 32 | P1 base register |
| P1LR | 22 | P1 length register |
| SBR | 32 | system base register |
| SLR | 22 | system length register |
| SISR | 16 | software interrupt summary register |
| ASTLVL | 4 | AST level register |
| MAPEN | 1 | memory management enable |
| PME | 1 | performance monitor enable |
| TRPIRQ | 8 | trap/interrupt pending |
| CRDERR | 1 | correctible read data error flag |
| MEMERR | 1 | memory error flag |
| PCQ[0:63] | 32 | PC prior to last PC change or interrupt; most recent PC change first |

```
WRU             8           interrupt character
```

The CPU attempts to detect when the simulator is idle. When idle, the simulator does not use any resources on the host system. Idle detection is controlled by the `SET IDLE` and `SET NOIDLE` commands:

```
SET CPU IDLE{=VMS|ULTRIX|NETBSD|FREEBSD|32V|ALL}      enable idle detection
SET CPU NOIDLE                                        disable idle detection
```

Idle detection is disabled by default. Unless ALL is specified, idle detection is operating system specific. If idle detection is enabled with an incorrect operating system setting, simulator performance could be impacted. The default operating system setting is VMS.

The CPU can maintain a history of the most recently executed instructions. This is controlled by the `SET CPU HISTORY` and `SHOW CPU HISTORY` commands:

```
SET CPU HISTORY             clear history buffer
SET CPU HISTORY=0           disable history
SET CPU HISTORY=n           enable history, length = n
SHOW CPU HISTORY            print CPU history
SHOW CPU HISTORY=n          print first n entries of CPU history
```

The maximum length for the history is 65536 entries.

### 2.1.2 Translation Buffer (TLB)

The translation buffer consists of two units, representing the system and user translation buffers, respectively. It has no registers. Each translation buffer entry consists of two 32b words, as follows:

```
word n          tag
word n+1        cached PTE
```

An invalid entry is indicated by a tag of 0xFFFFFFFF.

### 2.1.3 Qbus Adapter (QBA)

The QBA simulates the CQBIC Qbus adapter chip. It recognizes the following options:

```
SET QBA AUTOCONFIGURE       enable autoconfiguration
SET QBA NOAUTOCONFIGURE     disable autoconfiguration
```

and the following display command:

```
SHOW QBA IOSPACE            show IO space addresses
```

The QBA also implements a command to display a Qbus address to physical address translation:

```
SHOW QBA VIRTUAL=n          show translation for Qbus address n
```

Finally, the QBA implements main memory examination and modification via the Qbus map. The data width is always 16b:

```
EX QBA 0/10                 examine main memory words corresponding
                            to Qbus addresses 0-10
```

The QBA registers are:

```
name            size        comments

SCR             16          system configuration register
DSER            8           DMA system error register
MEAR            13          master error address register
SEAR            20          slave error address register
MBR             29          Qbus map base register
IPC             16          interprocessor communications register
IPL17           32          IPL 17 interrupt flags
IPL16           32          IPL 16 interrupt flags
IPL15           32          IPL 15 interrupt flags
IPL14           32          IPL 14 interrupt flags
```

### 2.1.4 Read-only memory (ROM)

The boot ROM consists of a single unit, simulating the 128KB boot ROM. It has no registers. The boot ROM is loaded with a binary byte stream using the `LOAD -r` command:

```
LOAD -r KA655X.BIN          load ROM image KA655X.BIN
```

ROM accesses a use a calibrated delay that slows ROM-based execution to about 500K instructions per second. This delay is required to make the power-up self-test routines run correctly on very fast hosts. The delay is controlled with the commands:

```
SET ROM NODELAY             ROM runs like RAM
SET ROM DELAY               ROM runs slowly
```

### 2.1.5 Non-volatile Memory (NVR)

The NVR consists of a single unit, simulating 1KB of battery-backed up memory in the SSC chip. When the simulator starts, NVR is cleared to 0, and the SSC battery-low indicator is set. Normally, NVR is saved and restored like other memory in the system. Alternately, NVR can be attached to a file. This allows its contents to be saved and restored independently of other memories, so that NVR state can be preserved across simulator runs.

Successfully loading an NVR image clears the SSC battery-low indicator.

### 2.1.6 System Devices (SYSD)

The system devices are the system-specific facilities implemented in the CVAX chip, the KA655 CPU board, the CMCTL memory controller, and the SSC system support chip. Note that the simulation of these devices is incomplete and is intended strictly to allow the patched bootstrap and console code to run. The SYSD registers are:

```
name            size        comments

CADR            8           cache disable register
MSER            8           memory system error register
CONPC           32          PC at console halt
CONPSL          32          PSL at console halt
CMCSR[0:17]     32          CMCTL control and status registers
CACR            8           second-level cache control register
```

```
BDR               8          front panel jumper register
BASE              29         SSC base address register
CNF               32         SSC configuration register
BTO               32         SSC bus timeout register
TCSR0             32         SSC timer 0 control/status register
TIR0              32         SSC timer 0 interval register
TNIR0             32         SSC timer 0 next interval register
TIVEC0            9          SSC timer 0 interrupt vector register
TCSR1             32         SSC timer 1 control/status register
TIR1              32         SSC timer 1 interval register
TNIR1             32         SSC timer 1 next interval register
TIVEC1            9          SSC timer 1 interrupt vector register
ADSM0             32         SSC address match 0 address
ADSK0             32         SSC address match 0 mask
ADSM1             32         SSC address match 1 address
ADSK1             32         SSC address match 1 mask
CDGDAT[0:16383]   32         cache diagnostic data store
```

BDR<7> is the halt-enabled switch. It controls how the console firmware responds to a BOOT command, a kernel halt (if option CONHALT is set), or a console halt (BREAK typed on the console terminal). If BDR<7> is set, the console firmware responds to all these conditions by entering its interactive command mode. If BDR<7> is clear, the console firmware boots the operating system in response to these conditions.

## *2.2 I/O Device Addressing*

Qbus I/O space is not large enough to allow all possible devices to be configured simultaneously at fixed addresses. Instead, many devices have floating addresses; that is, the assigned device address depends on the presence of other devices in the configuration:

```
DZ11              all instances have floating addresses
DHQ11             all instances have floating addresses
RL11              first instance has fixed address, rest floating
RXV211            first instance has fixed address, rest floating
MSCP disk         first instance has fixed address, rest floating
TMSCP tape        first instance has fixed address, rest floating
```

To maintain addressing consistency as the configuration changes, the simulator implements DEC's standard I/O address and vector autoconfiguration algorithms for devices DZ, VH, RL, RY, RQn, and TQ. This allows the user to enable or disable devices without needing to manage I/O addresses and vectors.

In addition to autoconfiguration, most devices support the `SET <device> ADDRESS` command, which allows the I/O page address of the device to be changed, and the `SET <device> VECTOR` command, which allows the vector of the device to be changed. Explicitly setting the I/O address of a device that normally uses autoconfiguration DISABLES autoconfiguration for that device and for the entire system. As a consequence, the user may have to manually configure all other autoconfigured devices, because the autoconfiguration algorithm no longer recognizes the explicitly configured device. A device can be reset to autoconfigure with the `SET <device> AUTOCONFIGURE` command. Autoconfiguration can be restored for the entire system with the `SET QBA AUTOCONFIGURE` command.

The current I/O map can be displayed with the `SHOW QBA IOSPACE` command. Addresses that have set by autoconfiguration are marked with an asterisk (*).

All devices support the `SHOW <device> ADDRESS` and `SHOW <device> VECTOR` commands, which display the device address and vector, respectively.

## *2.3 Programmed I/O Devices*

### 2.3.1 Terminal Input (TTI)

The terminal interfaces (TTI, TTO) can be set to one of three modes, 7P, 7B or 8B:

| mode | input characters | output characters |
|---|---|---|
| 7P | high-order bit cleared | high-order bit cleared, non-printing characters suppressed |
| 7B | high-order bit cleared | high-order bit cleared |
| 8B | no changes | no changes |

The default mode is 8B.

When the console terminal is attached to a Telnet session, it recognizes BREAK. If BREAK is entered, and BDR<7> is set, control returns to the console firmware; otherwise, BREAK is treated as a normal terminal input condition.

The terminal input (TTI) polls the console keyboard for input. It implements these registers:

| name | size | comments |
|---|---|---|
| BUF | 8 | last data item processed |
| CSR | 16 | control/status register |
| INT | 1 | interrupt pending flag |
| ERR | 1 | error flag (CSR<15>) |
| DONE | 1 | device done flag (CSR<7>) |
| IE | 1 | interrupt enable flag (CSR<6>) |
| POS | 32 | number of characters input |
| TIME | 24 | input polling interval (if 0, the keyboard is polled synchronously with the TODR) |

### 2.3.2 Terminal Output (TTO)

The terminal output (TTO) writes to the simulator console window. It implements these registers:

| name | size | comments |
|---|---|---|
| BUF | 8 | last data item processed |
| CSR | 16 | control/status register |
| INT | 1 | interrupt pending flag |
| ERR | 1 | error flag (CSR<15>) |
| DONE | 1 | device done flag (CSR<7>) |
| IE | 1 | interrupt enable flag (CSR<6>) |
| POS | 32 | number of characters input |
| TIME | 24 | time from I/O initiation to interrupt |

### 2.3.3 LPV11 Line Printer (LPT)

The line printer (LPT) writes data to a disk file. The POS register specifies the number of the next data item to be written. Thus, by changing POS, the user can backspace or advance the printer.

The line printer implements these registers:

```
name              size        comments

BUF               8           last data item processed
CSR               16          control/status register
INT               1           interrupt pending flag
ERR               1           error flag (CSR<15>)
DONE              1           device done flag (CSR<7>)
IE                1           interrupt enable flag (CSR<6>)
POS               32          position in the output file
TIME              24          time from I/O initiation to interrupt
STOP_IOE          1           stop on I/O error
```

Error handling is as follows:

```
error          STOP_IOE       processed as

not attached      1           report error and stop
                  0           out of paper

OS I/O error      x           report error and stop
```

### 2.3.4 Real-Time Clock (CLK)

The clock (CLK) implements these registers:

```
name              size        comments

CSR               16          control/status register
INT               1           interrupt pending flag
IE                1           interrupt enable flag (CSR<6>)
TODR              32          time-of-day register
BLOW              1           TODR battery low indicator
TIME              24          clock frequency
TPS               8           ticks per second (100)
```

The real-time clock autocalibrates; the clock interval is adjusted up or down so that the clock tracks actual elapsed time.

## *2.4 Disks*

### 2.4.1 RLV12/RL01,RL02 Cartridge Disk (RL)

RLV12 options include the ability to set units write enabled or write locked, to set the drive type to RL01, RL02, or autosize, and to write a DEC standard 044 compliant bad block table on the last track:

```
SET RLn LOCKED              set unit n write locked
SET RLn WRITEENABLED        set unit n write enabled
SET RLn RL01                set type to RL01
SET RLn RL02                set type to RL02
SET RLn AUTOSIZE            set type based on file size at ATTACH
SET RLn BADBLOCK            write bad block table on last track
```

The type options can be used only when a unit is not attached to a file. The bad block option can be used only when a unit is attached to a file. Units can also be set ENABLED or DISABLED. The RLV12 does not support the BOOT command.

The RLV12 implements these registers:

| name | size | comments |
|---|---|---|
| RLCS | 16 | control/status |
| RLDA | 16 | disk address |
| RLBA | 16 | memory address |
| RLBAE | 6 | memory address extension (RLV12) |
| RLMP,RLMP1,RLMP2 | 16 | multipurpose register queue |
| INT | 1 | interrupt pending flag |
| ERR | 1 | error flag (CSR<15>) |
| DONE | 1 | device done flag (CSR<7>) |
| IE | 1 | interrupt enable flag (CSR<6>) |
| STIME | 24 | seek time, per cylinder |
| RTIME | 24 | rotational delay |
| STOP_IOE | 1 | stop on I/O error |

Error handling is as follows:

| error | STOP_IOE | processed as |
|---|---|---|
| not attached | 1 | report error and stop |
| | 0 | disk not ready |
| end of file | x | assume rest of disk is zero |
| OS I/O error | x | report error and stop |

## 2.4.2 RQDX3 MSCP Disk Controllers (RQ, RQB, RQC, RQD)

The simulator implements four MSCP disk controllers, RQ, RQB, RQC, RQD. Initially, RQB, RQC, and RQD are disabled. Each RQ controller simulates an RQDX3 MSCP disk controller with four drives. RQ options include the ability to set units write enabled or write locked, and to set the drive type to one of many disk types:

```
SET RQn LOCKED                set unit n write locked
SET RQn WRITEENABLED          set unit n write enabled
SET RQn RX50                  set type to RX50
SET RQn RX33                  set type to RX33
SET RQn RD32                  set type to RD32
SET RQn RD51                  set type to RD51
SET RQn RD52                  set type to RD52
SET RQn RD53                  set type to RD53
SET RQn RD54                  set type to RD54
SET RQn RD31                  set type to RD31
SET RQn RA81                  set type to RA81
SET RQn RA82                  set type to RA82
set RQn RA71                  set type to RA71
SET RQn RA72                  set type to RA72
SET RQn RA90                  set type to RA90
SET RQn RA92                  set type to RA92
SET RQn RRD40                 set type to RRD40 (CD ROM)
```

```
SET RQn RAUSER{=n}            set type to RA82 with n MB's
SET -L RQn RAUSER{=n}         set type to RA82 with n LBN's
```

The type options can be used only when a unit is not attached to a file. RAUSER is a "user specified" disk; the user can specify the size of the disk in either MB (1000000 bytes) or logical block numbers (LBN's, 512 bytes each). The minimum size is 5MB; the maximum size is 2GB without extended file support, 1TB with extended file support

Units can also be set `ENABLED` or `DISABLED`. The RQ controllers do not support the `BOOT` command.

Each RQ controller implements the following special `SHOW` commands:

```
SHOW RQn TYPE                 show drive type
SHOW RQ RINGS                 show command and response rings
SHOW RQ FREEQ                 show packet free queue
SHOW RQ RESPQ                 show packet response queue
SHOW RQ UNITQ                 show unit queues
SHOW RQ ALL                   show all ring and queue state
SHOW RQn UNITQ                show unit queues for unit n
```

Each RQ controller implements these registers:

```
name              size        comments

SA                16          status/address register
S1DAT             16          step 1 init host data
CQBA              22          command queue base address
CQLNT             8           command queue length
CQIDX             8           command queue index
RQBA              22          request queue base address
RQLNT             8           request queue length
RQIDX             8           request queue index
FREE              5           head of free packet list
RESP              5           head of response packet list
PBSY              5           number of busy packets
CFLGS             16          controller flags
CSTA              4           controller state
PERR              9           port error number
CRED              5           host credits
HAT               17          host available timer
HTMO              17          host timeout value
CPKT[0:3]         5           current packet, units 0 to 3
PKTQ[0:3]         5           packet queue, units 0 to 3
UFLG[0:3]         16          unit flags, units 0 to 3
INT               1           interrupt request
ITIME             1           response time for initialization steps
                              (except for step 4)
QTIME             24          response time for 'immediate' packets
XTIME             24          response time for data transfers
PKTS[33*32]       16          packet buffers, 33W each, 32 entries
```

While VMS is not timing sensitive, most of the BSD-derived operating systems (NetBSD, OpenBSD, etc) are. The QTIME and XTIME parameters are set to values that allow these operating systems to run correctly.

Error handling is as follows:

```
error             processed as

not attached      disk not ready

end of file       assume rest of disk is zero

OS I/O error      report error and stop
```

### 2.4.3 RXV21/RX02 Floppy Disk (RY)

RXV21 options include the ability to set units write enabled or write locked, single or double density, or autosized:

```
SET RYn LOCKED                set unit n write locked
SET RYn WRITEENABLED          set unit n write enabled
SET RYn SINGLE                set unit n single density
SET RYn DOUBLE                set unit n double density (default)
SET RYn AUTOSIZE              set unit n autosized
```

The RXV21 does not support the `BOOT` command.

The RXV21 implements these registers:

```
name              size        comments

RYCS              16          status
RYBA              16          buffer address
RYWC              8           word count
RYDB              16          data buffer
RYES              12          error status
RYERR             8           error code
RYTA              8           current track
RYSA              8           current sector
STAPTR            4           controller state
INT               1           interrupt pending flag
ERR               1           error flag (CSR<15>)
TR                1           transfer ready flag (CSR<7>)
IE                1           interrupt enable flag (CSR<6>)
DONE              1           device done flag (CSR<5>)
CTIME             24          command completion time
STIME             24          seek time, per track
XTIME             24          transfer ready delay
STOP_IOE          1           stop on I/O error
SBUF[0:255]       8           sector buffer array
```

Error handling is as follows:

```
error          STOP_IOE       processed as

not attached      1           report error and stop
                  0           disk not ready
```

RX02 data files are buffered in memory; therefore, end of file and OS I/O errors cannot occur.

## 2.5 Tapes

### 2.5.1 TSV11/TSV05 Magnetic Tape (TS)

TS options include the ability to make the unit write enabled or write locked.

```
SET TS LOCKED                   set unit write locked
SET TS WRITEENABLED             set unit write enabled
```

The TS drive can be set to a specific reel capacity in MB, or to unlimited capacity:

```
SET TS0 CAPAC=m                 set capacity to m MB (0 = unlimited)
SHOW TS0 CAPAC                  show capacity in MB
```

The TSV11 does not support the `BOOT` command.

The TS controller implements these registers:

```
name                size        comments

TSSR                16          status register
TSBA                16          bus address register
TSDBX               16          data buffer extension register
CHDR                16          command packet header
CADL                16          command packet low address or count
CADH                16          command packet high address
CLNT                16          command packet length
MHDR                16          message packet header
MRFC                16          message packet residual frame count
MXS0                16          message packet extended status 0
MXS1                16          message packet extended status 1
MXS2                16          message packet extended status 2
MXS3                16          message packet extended status 3
MXS4                16          message packet extended status 4
WADL                16          write char packet low address
WADH                16          write char packet high address
WLNT                16          write char packet length
WOPT                16          write char packet options
WXOPT               16          write char packet extended options
ATTN                1           attention message pending
BOOT                1           boot request pending
OWNC                1           if set, tape owns command buffer
OWNM                1           if set, tape owns message buffer
TIME                24          delay
POS                 32          position
```

Error handling is as follows:

```
error               processed as

not attached        tape not ready

end of file         bad tape

OS I/O error        fatal tape error
```

### 2.5.2 TQK50 TMSCP Disk Controller (TQ)

The TQ controller simulates the TQK50 TMSCP disk controller. TQ options include the ability to set units write enabled or write locked, and to specify the controller type and tape length:

```
SET TQn LOCKED                set unit n write locked
SET TQn WRITEENABLED          set unit n write enabled
SET TQ TK50                   set controller type to TK50
SET TQ TK70                   set controller type to TK70
SET TQ TU81                   set controller type to TU81
SET TQ TKUSER{=n}             set controller type to TK50 with
                              tape capacity of n MB
```

User-specified capacity must be between 50 and 2000 MB. The TQK50 does not support the `BOOT` command.

Regardless of the controller type, individual units can be set to a specific reel capacity in MB, or to unlimited capacity:

```
SET TQn CAPAC=m               set unit n capacity to m MB (0 = unlimited)
SHOW TQn CAPAC                show unit n capacity in MB
```

The TQ controller implements the following special `SHOW` commands:

```
SHOW TQ TYPE                  show controller type
SHOW TQ RINGS                 show command and response rings
SHOW TQ FREEQ                 show packet free queue
SHOW TQ RESPQ                 show packet response queue
SHOW TQ UNITQ                 show unit queues
SHOW TQ ALL                   show all ring and queue state
SHOW TQn UNITQ                show unit queues for unit n
```

The TQ controller implements these registers:

```
name              size        comments

SA                16          status/address register
S1DAT             16          step 1 init host data
CQBA              22          command queue base address
CQLNT             8           command queue length
CQIDX             8           command queue index
RQBA              22          request queue base address
RQLNT             8           request queue length
RQIDX             8           request queue index
FREE              5           head of free packet list
RESP              5           head of response packet list
PBSY              5           number of busy packets
CFLGS             16          controller flags
CSTA              4           controller state
PERR              9           port error number
CRED              5           host credits
HAT               17          host available timer
HTMO              17          host timeout value
CPKT[0:3]         5           current packet, units 0 to 3
PKTQ[0:3]         5           packet queue, units 0 to 3
UFLG[0:3]         16          unit flags, units 0 to 3
```

```
POS[0:3]          32          tape position, units 0 to 3
OBJP[0:3]         32          object position, units 0 to 3
INT               1           interrupt request
ITIME             1           response time for initialization steps
                              (except for step 4)
QTIME             24          response time for 'immediate' packets
XTIME             24          response time for data transfers
PKTS[33*32]       16          packet buffers, 33W each, 32 entries
```

Error handling is as follows:

```
error             processed as

not attached      tape not ready

end of file       end of medium

OS I/O error      fatal tape error
```

## *2.6 Communications Devices*

### 2.6.1 DZV11 Terminal Multiplexer (DZ)

The DZV11 is an 4-line terminal multiplexor. Up to 4 DZ11's (16 lines) are supported. The number of lines can be changed with the command

```
SET DZ LINES=n                set line count to n
```

The line count must be a multiple of 4, with a maximum of 16.

The DZ11 supports three character processing modes, 7P, 7B, and 8B:

```
mode  input characters        output characters

7P    high-order bit cleared  high-order bit cleared,
                              non-printing characters suppressed
7B    high-order bit cleared  high-order bit cleared
8B    no changes              no changes
```

The default is 8B.

The DZV11 supports logging on a per-line basis. The command

```
SET DZ LOG=line=filename
```

enables logging for the specified line to the indicated file. The command

```
SET DZ NOLOG=line
```

disables logging for the specified line and closes any open log file. Finally, the command

```
SHOW DZ LOG
```

displays logging information for all DZ lines.

The terminal lines perform input and output through Telnet sessions connected to a user-specified port. The `ATTACH` command specifies the port to be used:

```
ATTACH {-am} DZ <port>        set up listening port
```

where port is a decimal number between 1 and 65535 that is not being used for other TCP/IP activities. The optional switch -m turns on the DZV11's modem controls; the optional switch -a turns on active disconnects (disconnect session if computer clears Data Terminal Ready). Without modem control, the DZV11 behaves as though terminals were directly connected; disconnecting the Telnet session does not cause any operating system-visible change in line status.

Once the DZ is attached and the simulator is running, the DZ will listen for connections on the specified port. It assumes that the incoming connections are Telnet connections. The connection remains open until disconnected by the simulated program, the Telnet client, a `SET DZ DISCONNECT` command, or a `DETACH DZ` command.

Other special DZ commands:

```
SHOW DZ CONNECTIONS                show current connections
SHOW DZ STATISTICS                 show statistics for active connections
SET DZ DISCONNECT=linenumber       disconnects the specified line.
```

The DZV11 implements these registers:

```
name              size        comments

CSR[0:3]          16          control/status register, boards 0 to 3
RBUF[0:3]         16          receive buffer, boards 0 to 3
LPR[0:3]          16          line parameter register, boards 0 to 3
TCR[0:3]          16          transmission control register, boards 0 to 3
MSR[0:3]          16          modem status register, boards 0 to 3
TDR[0:3]          16          transmit data register, boards 0 to 3
SAENB[0:3]        1           silo alarm enabled, boards 0 to 3
RXINT             4           receive interrupts, boards 3 to 0
TXINT             4           transmit interrupts, boards 3 to 0
MDMTCL            1           modem control enabled
AUTODS            1           autodisconnect enabled
```

The DZV11 does not support save and restore. All open connections are lost when the simulator shuts down or the DZ is detached.

## 2.6.2 DHQ11 Terminal Multiplexer (VH)

The DHQ11 is an 8-line terminal multiplexer for Qbus systems. Up to 4 DHQ11's are supported.

The DHQ11 is a programmable asynchronous terminal multiplexer. It has two programming modes: DHV11 and DHU11. The register sets are compatible with these devices. For transmission, the DHQ11 can be used in either DMA or programmed I/O mode. For reception, there is a 256-entry FIFO for received characters, dataset status changes, and diagnostic information, and a programmable input interrupt timer (in DHU mode). The device supports 16-, 18-, and 22-bit addressing. The DHQ11 can be programmed to filter and/or handle XON/XOFF characters independently of the processor. The DHQ11 supports programmable bit width (between 5 and 8) for the input and output of characters.

The DHQ11 has a rocker switch for determining the programming mode. By default, the DHV11 mode is selected, though DHU11 mode is recommended for applications that can support it. The VH controller may be adjusted on a per controller basis as follows:

```
SET VHn DHU                   use the DHU programming mode and registers
SET VHn DHV                   use the DHV programming mode and registers
```

DMA output is supported. In a real DHQ11, DMA is not initiated immediately upon receipt of TX.DMA.START but is dependent upon some internal processes. The VH controller mimics this behavior by default. It may be desirable to alter this and start immediately, though this may not be compatible with all operating systems and diagnostics. You can change the behavior of the VH controller as follows:

```
SET VHn NORMAL                use normal DMA procedures
SET VHn FASTDMA               set DMA to initiate immediately
```

The terminal lines perform input and output through Telnet sessions connected to a user-specified port. The `ATTACH` command specifies the port to be used:

```
ATTACH VH <port>              set up listening port
```

where port is a decimal number between 1 and 65535 that is not being used for other TCP/IP activities. This port is the point of entry for all lines on all VH controllers.

The number of lines can be changed with the command

```
SET VH LINES=n                set line count to n
```

The line count must be a multiple of 8, with a maximum of 32.

Modem and auto-disconnect support may be set on an individual controller basis. The SET MODEM command directs the controller to report modem status changes to the computer. The SET HANGUP command turns on active disconnects (disconnect session if computer clears Data Terminal Ready).

```
SET VHn [NO]MODEM             disable/enable modem control
SET VHn [NO]HANGUP            disable/enable disconnect on DTR drop
```

Once the VH is attached and the simulator is running, the VH will listen for connections on the specified port. It assumes that the incoming connections are Telnet connections. The connection remains open until disconnected by the simulated program, the Telnet client, a `SET VH DISCONNECT` command, or a `DETACH VH` command.

Other special VH commands:

```
SHOW VH CONNECTIONS           show current connections
SHOW VH STATISTICS            show statistics for active connections
SET VH DISCONNECT=linenumber  disconnects the specified line.
```

The DHQ11 implements these registers, though not all can be examined from SCP:

| name | size | comments |
|---|---|---|
| CSR[0:3] | 16 | control/status register, boards 0 to 3 |
| RBUF[0:3] | 16 | receive buffer, boards 0 to 3 |
| LPR[0:3] | 16 | line parameter register, boards 0 to 3 |
| RXINT | 4 | receive interrupts, boards 3..0 |
| TXINT | 4 | transmit interrupts, boards 3..0 |

[more to be described...]

The DHQ11 does not support save and restore. All open connections are lost when the simulator shuts down or the VH is detached.

### 2.6.3 DELQA-T/DELQA/DEQNA Qbus Ethernet Controllers (XQ, XQB)

The simulator implements two DELQA-T/DELQA/DEQNA Qbus Ethernet controllers (XQ, XQB). Initially, XQ is enabled, and XQB is disabled. Options allow control of the MAC address, the controller mode, and the sanity timer.

```
SET  XQ MAC=<mac-address>     ex. 08-00-2B-AA-BB-CC
SHOW XQ MAC
```

These commands are used to change or display the MAC address. <mac-address> is a valid Ethernet MAC, delimited by dashes or periods. The controller defaults to 08-00-2B-AA-BB-CC, which should be sufficient if there is only one SIMH controller on your LAN. Two cards with the same MAC address will see each other's packets, resulting in a serious mess.

```
SET  XQ TYPE={DEQNA|[DELQA]|DELQA-T}
SHOW XQ TYPE
```

These commands are used to change or display the controller mode. DELQA mode is better and faster but may not be usable by older or non-DEC OS's. Also, be aware that DEQNA mode is not supported by many modern OS's. The DEQNA-LOCK mode of the DELQA card is emulated by setting the the controller to DEQNA -- there is no need for a separate mode. DEQNA-LOCK mode behaves exactly like a DEQNA, except for the operation of the VAR and MOP processing.

```
SET  XQ SANITY={ON|[OFF]}
SHOW XQ SANITY
```

These commands change or display the INITIALIZATION sanity timer (DEQNA jumper W3/DELQA switch S4). The INITIALIZATION sanity timer has a default timeout of 4 minutes, and cannot be turned off, just reset. The normal sanity timer can be set by operating system software regardless of the state of this switch. Note that only the DEQNA (or the DELQA in DEQNA-LOCK mode (=DEQNA)) supports the sanity timer -- it is ignored by a DELQA in Normal mode, which uses switch S4 for a different purpose.

```
SET  XQ POLL={DEFAULT|4..2500}
SHOW XQ POLL
```

These commands change or display the service polling timer. The polling timer is calibrated to run the service thread 200 times per second. This value can be changed to accommodate particular system requirements for more (or less) frequent polling.

```
SHOW XQ STATS
```

This command will display the accumulated statistics for the simulated Ethernet controller.

To access the network, the simulated Ethernet controller must be attached
to a real Ethernet interface:

```
ATTACH XQ0 {ethX|<device_name>}     ex. eth0 or /dev/era0
SHOW XQ ETH
```

where X in 'ethX' is the number of the Ethernet controller to attach, or the real device name. The X number is system-dependant. If you only have one Ethernet controller, the number will probably be 0. To find out what your system thinks the Ethernet numbers are, use the SHOW XQ ETH command. The device list can be quite cryptic, depending on the host system, but is probably better than guessing. If you do not attach the device, the controller will behave as though the Ethernet cable were unplugged.

XQ and XQB have the following registers:

```
name            size        comments

SA0             16          station address word 0
SA1             16          station address word 1
SA2             16          station address word 2
SA3             16          station address word 3
SA4             16          station address word 4
SA5             16          station address word 5
RBDL            32          receive buffer descriptor list
XBDL            32          trans(X)mit buffer descriptor list
CSR             16          control status register
VAR             16          vector address register
INT             1           interrupt request flag
```

One final note: because of its asynchronous nature, the XQ controller is not limited to the ~1.5Mbit/sec of the real DEQNA/DELQA controllers, nor the 10Mbit/sec of a standard Ethernet. Attach it to a Fast Ethernet (100 Mbit/sec) card, and "Feel the Power!" :-)

## *2.7 CR11 Card Reader (CR)*

The card reader (CR) implements a single controller (the CR11) and card reader (e.g., Documation M200, GDI Model 100) by reading a file and presenting lines or cards to the simulator. Card decks may be represented by plain text ASCII files, card image files, or column binary files. The CR11 controller is also compatible with the CM11-F, CME11, and CMS11.

Card image files are a file format designed by Douglas W. Jones at the University of Iowa to support the interchange of card deck data. These files have a much richer information carrying capacity than plain ASCII files. Card Image files can contain such interchange information as card-stock color, corner cuts, special artwork, as well as the binary punch data representing all 12 columns. Complete details on the format, as well as sample code, are available at Prof. Jones's site: http://www.cs.uiowa.edu/~jones/cards/.

Examples of the CR11 include the M8290 and M8291 (CMS11). All card readers use a common vector at 0230 and CSR at 177160. Even though the CR11 is normally configured as a BR6 device, it is configured for BR4 in this simulation.

The card reader supports ASCII, card image, and column binary format card “decks.” When reading plain ASCII files, lines longer than 80 characters are silently truncated. Card image support is included for 80 column Hollerith, 82 column Hollerith (silently ignoring columns 0 and 81), and 40 column Hollerith (mark-sense) cards. Column binary supports 80 column card images only. All files are attached read-only (as if the -R switch were given).

```
ATTACH -A CR <file>                 file is ASCII text
ATTACH -B CR <file>                 file is column binary
ATTACH -I CR <file>                 file is card image format
```

If no flags are given, the file extension is evaluated. If the filename ends in .TXT, the file is treated as ASCII text. If the filename ends in .CBN, the file is treated as column binary. Otherwise, the CR driver looks for a

card image header. If a correct header is found the file is treated as card image format, otherwise it is treated as ASCII text.

The correct character translation MUST be set if a plain text file is to be used for card deck input. The correct translation SHOULD be set to allow correct ASCII debugging of a card image or column binary input deck. Depending upon the operating system in use, how it was generated, and how the card data will be read and used, the translation must be set correctly so that the proper character set is used by the driver. Use the following command to explicitly set the correct translation:

```
SET TRANSLATION={DEFAULT|026|026FTN|029|EBCDIC}
```

This command should be given after a deck is attached to the simulator. The mappings above are completely described at http://www.cs.uiowa.edu/~jones/cards/codes.html. Note that DEC typically used 029 or 026FTN mappings.

DEC operating systems used a variety of methods to determine the end of a deck (recognizing that 'hopper empty' does not necessarily mean the end of a deck. Below is a summary of the various operating system conventions for signaling end of deck:

```
RT-11:      12-11-0-1-6-7-8-9 punch in column 1

RSTS/E:     12-11-0-1 or 12-11-0-1-6-7-8-9 punch in column 1

RSX:        12-11-0-1-6-7-8-9 punch

VMS:        12-11-0-1-6-7-8-9 punch in first 8 columns

TOPS:       12-11-0-1 or 12-11-0-1-6-7-8-9 punch in column 1
```

Using the AUTOEOF setting, the card reader can be set to automatically generate an EOF card consisting of the 12-11-0-1-6-7-8-9 punch in columns 1-8. When set to CD11 mode, this switch also enables automatic setting of the EOF bit in the controller after the EOF card has been processed. [The CR11 does not have a similar capability.] By default AUTOEOF is enabled.

```
SET CR AUTOEOF
SET CR NOAUTOEOF
```

The default card reader rate for the CR11 is 285 cpm. The reader rate can be set to its default value or to anywhere in the range 200..1200 cpm. This rate may be changed while the unit is attached.

```
SET CR RATE={DEFAULT|200..1200}
```

It is standard operating procedure for operators to load a card deck and press the momentary action RESET button to clear any error conditions and alert the processor that a deck is available to read. Use the following command to simulate pressing the card reader RESET button,

```
SET CR RESET
```

Another common control of physical card readers is the STOP button. An operator could use this button to finish the read operation for the current card and terminate reading a deck early. Use the following command to simulate pressing the card reader STOP button.

```
SET CR STOP
```

The simulator does not support the `BOOT` command. The simulator does not stop on file I/O errors. Instead the controller signals a reader check to the CPU.

The CR controller implements these registers:

| name | size | comments |
|---|---|---|
| BUF | 8 | ASCII value of last column processed |
| CRS | 16 | CR11 status register |
| CRB1 | 16 | CR11 12-bit Hollerith character |
| CRB2 | 16 | CR11 8-bit compressed character |
| CRM | 16 | CR11 maintenance register |
| CDST | 16 | CD11 control/status register |
| CDCC | 16 | CD11 column count |
| CDBA | 16 | CD11 current bus address |
| CDDB | 16 | CD11 data buffer, 2nd status |
| BLOWER | 2 | blower state value |
| INT | 1 | interrupt pending flag |
| ERR | 1 | error flag (CRS<15>) |
| IE | 1 | interrupt enable flag (CRS<6>) |
| POS | 32 | file position - do not alter |
| TIME | 24 | delay time between columns |

# 3 Symbolic Display and Input

The VAX simulator implements symbolic display and input. Display is controlled by command line switches:

| | |
|---|---|
| -a,-c | display as ASCII data |
| -m | display instruction mnemonics |
| -p | display compatibility mode mnemonics |
| -r | display RADIX50 encoding |

Input parsing is controlled by the first character typed in or by command line switches:

| | |
|---|---|
| ' or -a | ASCII characters (determined by length) |
| " or -c | ASCII string (maximum 60 characters) |
| -p | compatibility mode instruction mnemonic |
| alphabetic | instruction mnemonic |
| numeric | octal number |

VAX instruction input uses standard VAX assembler syntax. Compatibility mode instruction input uses standard PDP-11 assembler syntax.

The syntax for VAX specifiers is as follows:

| syntax | specifier | displacement | comments |
|---|---|---|---|
| #s^n, #n | 0n | - | short literal, integer only |
| [Rn] | 4n | - | indexed, second specifier follows |
| Rn | 5n | - | PC illegal |
| (Rn) | 6n | - | PC illegal |
| -(Rn) | 7n | - | PC illegal |
| (Rn)+ | 8n | - | |
| #i^n, #n | 8F | n | immediate |
| @(Rn)+ | 9n | - | |
| @#addr | 9F | addr | absolute |
| {+/-}b^d(Rn) | An | {+/-}d | byte displacement |
| b^d | AF | d - PC | byte PC relative |

```
@{+/-}b^d(Rn)     Bn          {+/-}d      byte displacement deferred
@b^d              BF          d - PC      byte PC relative deferred
{+/-}w^d(Rn)      Cn          {+/-}d      word displacement
w^d               CF          d - PC      word PC relative
@{+/-}w^d(Rn)     Dn          {+/-}d      word displacement deferred
@w^d              DF          d - PC      word PC relative deferred
{+/-}l^d(Rn)      En          {+/-}d      long displacement
l^d               EF          d - PC      long PC relative
@{+/-}l^d(Rn)     Fn          {+/-}d      long displacement deferred
@l^d              FF          d - PC      long PC relative deferred
```

If no override is given for a literal (s^ or i^) or for a displacement or PC relative address (b^, w^, or l^), the simulator chooses the mode automatically.

# Appendix - The KA655"X"

The real KA655 is limited to 64MB of memory, and the KA655 firmware is coded to this limit. However, the VAX operating systems (VMS, Ultrix, NetBSD) know very little about the hardware details. Instead, they take their memory size information from the Restart Parameter Block (RPB). If the firmware sets up an RPB for more than 64MB, the operating systems use the extra memory without requiring source changes.

If more than 64MB of memory is configured, the simulator implements an 18th CMCTL register. This read-only register gives the size of main memory in MB. The console firmware (ka655x.bin) uses this to set up the RPB. Other parts of the firmware are generally unaware of extended memory; thus, all the diagnostic commands operate only on the first 64MB of memory. However SHOW MEM will display the total amount of memory the simulator is configured with

If 64MB or less of memory is configured, the 18th CMCTL register is invisible, and the simulator operates like a real KA655.