

VAX-11/780 Simulator Usage

01-Jul-2007

COPYRIGHT NOTICE

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

Original code published in 1993-2007, written by Robert M Supnik
Copyright (c) 1993-2007, 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.

| | | |
|-------|---|----|
| 1 | Simulator Files | 3 |
| 2 | VAX780 Features..... | 4 |
| 2.1 | CPU and System Devices | 5 |
| 2.1.1 | CPU | 5 |
| 2.1.2 | Translation Buffer (TLB) | 7 |
| 2.1.3 | SBI Controller (SBI) | 7 |
| 2.1.4 | Memory Controllers (MCTL0, MCTL1)..... | 7 |
| 2.1.5 | Time-Of-Day Clock (TODR)..... | 8 |
| 2.1.6 | Interval Timer (TMR)..... | 8 |
| 2.1.7 | Unibus Adapter (UBA) | 8 |
| 2.1.8 | Massbus Adapters (MBA0, MBA1) | 9 |
| 2.2 | I/O Device Addressing | 9 |
| 2.3 | Programmed I/O Devices | 10 |
| 2.3.1 | Console Input (TTI)..... | 10 |
| 2.3.2 | Console Output (TTO) | 11 |
| 2.3.3 | RX01 Console Floppy Disk (RX) | 11 |
| 2.3.4 | Line Printer (LPT) | 12 |
| 2.4 | Disks..... | 12 |
| 2.4.1 | RP04/05/06/07, RM02/03/05/80 Disk Pack Drives (RP)..... | 12 |
| 2.4.2 | RL11/RL01/RL02 Cartridge Disk (RL) | 13 |
| 2.4.3 | RK611/RK06/RK07 Cartridge Disk (HK)..... | 14 |
| 2.4.4 | UDA50 MSCP Disk Controllers (RQ, RQB, RQC, RQD) | 15 |
| 2.5 | Tapes..... | 17 |
| 2.5.1 | TM03/TE16/TU45/TU77 Magnetic Tapes (TU)..... | 17 |
| 2.5.2 | TS11 Magnetic Tape (TS) | 17 |
| 2.5.3 | TUK50 TMSCP Disk Controller (TQ) | 18 |
| 2.6 | Communications Devices | 20 |
| 2.6.1 | DZ11 Terminal Multiplexer (DZ) | 20 |
| 2.7 | CR11 Card Reader (CR) | 21 |
| 3 | Symbolic Display and Input..... | 23 |

This memorandum documents the DEC VAX-11/780 simulator.

1 Simulator Files

To compile the VAX-11/780, you must define VM_VAX, VAX780, 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
              vax780_defs.h
              vax_cis.c
              vax_cmode.c
              vax_cpu.c
              vax_cpu1.c
              vax_fpa.c
              vax_mmu.c
              vax_octa.c
              vax_sys.c
              vax_syscm.c
              vax780_mba.c
              vax780_mem.c
              vax780_sbi.c
              vax780_stddev.c
              vax780_syslist.c
              vax780_uba.c

sim/pdp11/    pdp11_cr_dat.h
              pdp11_mscp.h
              pdp11_uqssp.h
              pdp11_xu.h
              pdp11_cr.c
              pdp11_dz.c
              pdp11_hk.c
              pdp11_lp.c
              pdp11_rl.c
```

pdp11_rp.c
pdp11_rq.c
pdp11_ry.c
pdp11_tq.c
pdp11_ts.c
pdp11_tu.c
pdp11_xu.c

Additional files are:

sim/vax/ vmb.exe standard boot code

2 VAX780 Features

The VAX780 simulator is configured as follows:

| device name(s) | simulates |
|----------------|--|
| CPU | VAX-11/780 CPU |
| TLB | translation buffer |
| SBI | system bus controller |
| MCTL0, MTCL1 | memory controllers, MS780C with 4MB memory each, or MS780E with 8MB-64MB each |
| UBA | DW780 Unibus adapter |
| MBA0, MBA1 | RH780 Massbus adapters |
| TODR | time-of-day clock |
| TMR | interval timer |
| TTI, TTO | console terminal |
| RX | console RX01 floppy disk |
| DZ | DZ11 8-line terminal multiplexer (up to 4) |
| CR | CR11 card reader |
| LPT | LP11 line printer |
| RP | RP04/05/06/07, RM02/03/05/80 Massbus disks, up to eight |
| HK | RK611/RK06(7) cartridge disk controller with eight drives |
| RL | RL11/RL01(2) cartridge disk controller with four drives |
| RQ | UDA50 MSCP controller with four drives |
| RQB | second UDA50 MSCP controller with four drives |
| RQC | third UDA50 MSCP controller with four drives |
| RQD | fourth UDA50 MSCP controller with four drives |
| RY | RX211 floppy disk controller with two drives |
| TS | TS11 magnetic tape controller with one drive |
| TQ | TUK50 TMSCP magnetic tape controller with four drives |
| TU | TM03 tape formatter with eight TE16/TU45/TU77 drives |
| XU | DEUNA/DELUA Ethernet controller |
| XUB | second DEUNA/DELUA Ethernet controller |

The DZ, LPT, RP, RL, RQ, RQB, RQC, RQD, RY, TS, TQ, TU, XU, and XUB devices can be set DISABLED. RQB, RQC, RQD, VH, XU, and XUB are disabled by default.

The VAX780 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 `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 8M           set memory size = 8MB
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

```

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 first version of the VAX-11/780 used MS780C controllers, which supported 1-4MB of memory per controller. This is the only memory controller recognized by VMS V1. MS780E controllers supported 4MB-64MB per controller.
- The controller type is set automatically based on memory size.

Initial memory size is 8MB.

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 ROM in memory controller 0
-s           load ROM in memory controller 1

```

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..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}    enable idle detection
SET CPU NOIDLE                                  disable idle detection

```

Idle detection is disabled by default. Idle detection is operating system specific. If idle detection is enabled with an incorrect operating system setting, simulator performance will be severely diminished. 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 SBI Controller (SBI)

The SBI is the VAX-11/780 system bus. The simulated SBI implements these registers:

| name | size | comments |
|--------|------|---------------------------------|
| NREQ14 | 16 | Nexus IPL14 interrupt requests |
| NREQ15 | 16 | Nexus IPL15 interrupt requests |
| NREQ16 | 16 | Nexus IPL16 interrupt requests |
| NREQ17 | 16 | Nexus IPL17 interrupt requests |
| WCSA | 16 | writeable control store address |
| WCSD | 32 | writeable control store data |
| MBRK | 13 | microbreak register |
| SBIFS | 32 | SBI fault status |
| SBISC | 32 | SBI silo compare |
| SBIMT | 32 | SBI maintenance register |
| SBIER | 32 | SBI error status |
| SBITMO | 32 | SBI timeout address |

2.1.4 Memory Controllers (MCTL0, MCTL1)

The memory controllers implement the registers for the MS780C (8MB memory) or MS780E (16MB or greater memory). Each controller implements these registers:

| name | size | comments |
|-------------|------|----------------------------------|
| CRA | 32 | control register A |
| CRB | 32 | control register B |
| CRC | 32 | control register C |
| CRD | 32 | control register D (MS780E only) |
| ROM[0:1023] | 32 | bootstrap ROM |

ROM can be loaded from a file with the commands

```

LOAD -R <file>          load MCTL0 ROM
LOAD -S <file>          load MCTL1 ROM

```

2.1.5 Time-Of-Day Clock (TODR)

The TODR tracks time since an arbitrary start in 1 microsecond intervals. It has these registers:

| name | size | comments |
|------|------|----------------------|
| TODR | 32 | time-of-day register |
| TIME | 24 | delay between ticks |

The TODR register autocalibrates against real-world time.

2.1.6 Interval Timer (TMR)

The interval timer implements the VAX architectural timer, with 1 microsecond intervals. It has these registers:

| name | size | comments |
|------|------|-----------------------------------|
| ICCS | 32 | interval timer control and status |
| ICR | 32 | interval count register |
| NICR | 32 | next interval count register |
| INT | 1 | interrupt request |

For standard VMS intervals (10 milliseconds), the interval timer autocalibrates against real-world time.

2.1.7 Unibus Adapter (UBA)

The Unibus adapter (UBA) simulates the DW780. It recognizes these options:

```

SET UBA AUTOCONFIGURE    enable autoconfiguration
SET UBA NOAUTOCONFIGURE  disable autoconfiguration

```

and this SHOW command:

```

SHOW UBA IOSPACE          display IO address space assignments

```

The UBA also implements a command to display a Unibus address to physical address translation:

```

SHOW UBA VIRTUAL=n       show translation for Unibus address n

```

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

```

EX UBA 0/10              examine main memory words corresponding
                          to Unibus addresses 0-10

```

The UBA has these registers:

| name | size | comments |
|------|------|----------|
|------|------|----------|

| | | |
|------------|----|---|
| IPL14 | 32 | Unibus IPL14 interrupt requests |
| IPL15 | 32 | Unibus IPL15 interrupt requests |
| IPL16 | 32 | Unibus IPL16 interrupt requests |
| IPL17 | 32 | Unibus IPL17 interrupt requests |
| CNFR | 32 | configuration register |
| CR | 32 | control register |
| SR | 32 | status register |
| DR | 32 | diagnostic register |
| INT | 1 | internal UBA interrupt request |
| NEXINT | 1 | UBA Nexus interrupt request |
| AIIP | 1 | adapter initialization in progress flag |
| UIIP | 1 | Unibus initialization in progress flag |
| FMER | 32 | failing memory address |
| FUBAR | 32 | failing UBA map register |
| BRSVR0 | 32 | spare register 0 |
| BRSVR1 | 32 | spare register 1 |
| BRSVR2 | 32 | spare register 2 |
| BRSVR3 | 32 | spare register 3 |
| BRRVR4 | 32 | vector register, IPL 14 |
| BRRVR5 | 32 | vector register, IPL 15 |
| BRRVR6 | 32 | vector register, IPL 16 |
| BRRVR7 | 32 | vector register, IPL 17 |
| DPR[0:15] | 32 | data path registers 0..15 |
| MAP[0:495] | 32 | map registers 0..495 |
| AITIME | 24 | adapter initialization time |
| UITIME | 24 | Unibus initialization time |

2.1.8 Massbus Adapters (MBA0, MBA1)

The Massbus adapters (MBA0, MBA1) simulate RH780's. MBA0 is assigned to the RP disk drives, MBA1 to the TU tape drives. Each MBA has these registers:

| name | size | comments |
|------------|------|-----------------------------|
| CNFR | 32 | configuration register |
| CR | 32 | control register |
| SR | 32 | status register |
| VA | 17 | virtual address register |
| BC | 32 | byte count register |
| DR | 32 | diagnostic register |
| SMR | 32 | selected map register |
| MAP[0:255] | 32 | map registers |
| NEXINT | 1 | MBA Nexus interrupt request |

2.2 I/O Device Addressing

Unibus 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 |
| RL11 | first instance has fixed address, rest floating |
| RX11/RX211 | first instance has fixed address, rest floating |
| DEUNA/DELUA | 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, RL, RY, XU, RQ, and TQ. This allows the user to enable or disable devices without needing to manage I/O addresses and vectors.

Autoconfiguration cannot solve address conflicts between devices with overlapping fixed addresses. For example, with default I/O page addressing, the PDP-11 can support either a TUK50 or a TS11, but not both, since they use the same I/O addresses.

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 CPU AUTOCONFIGURE` command.

The current I/O map can be displayed with the `SHOW CPU 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 Console 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 Console 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 RX01 Console Floppy Disk (RX)

RX01 options include the ability to set units write enabled or write locked:

| | |
|----------------------|--------------------------|
| SET RXn LOCKED | set unit n write locked |
| SET RXn WRITEENABLED | set unit n write enabled |

The RX01 implements a special command, FLOAD, for loading VAX executables from an RT11-formatted console floppy disk image:

```
FLOAD <file_name> {<origin>}
```

FLOAD searches the floppy disk image attached to the RX01 for the named file and then loads it into VAX-11/780 memory starting at the origin. If no origin is specified, the default origin is 200 (hex).

The RX01 implements these registers:

| name | size | comments |
|-------------|------|-------------------------------|
| FNC | 8 | function select |
| ES | 8 | error status |
| ECODE | 8 | error code |
| TA | 8 | track address |
| SA | 8 | sector address |
| STATE | 4 | protocol state |
| BPTR | 7 | data buffer pointer |
| CTIME | 24 | command initiation delay |
| STIME | 24 | seek time delay, per track |
| XTIME | 24 | transfer time delay, per byte |
| STOP_IOE | 1 | stop on I/O error |
| DBUF[0:127] | 8 | data buffer |

Error handling is as follows:

| | | |
|--------------|----------|-----------------------|
| error | STOP_IOE | processed as |
| not attached | 1 | report error and stop |

```
0 disk not ready
```

RX01 data files are buffered in memory; therefore, end of file and OS I/O errors cannot occur.

2.3.4 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 |

2.4 Disks

All VAX-11/780 disks, and the TUK50 MSCP tape, support a special form of the boot command, with the following syntax:

```
BOOT <unit>{/R5:<value>}
```

For example,

```
BOOT RP0/R5:1
```

The optional switch, /R5, specifies that R5 is to be loaded with the specified value prior to booting. If the switch is omitted, R5 is loaded with 0.

2.4.1 RP04/05/06/07, RM02/03/05/80 Disk Pack Drives (RP)

The RP controller implements the Massbus family of large disk drives. RP options include the ability to set units write enabled or write locked, to set the drive type to one of six disk types, or autosize, and to write a DEC standard 044 compliant bad block table on the last track:

```
SET RPn LOCKED set unit n write locked
```

| | |
|----------------------|---------------------------------------|
| SET RPN WRITEENABLED | set unit n write enabled |
| SET RPN RM03 | set type to RM03 |
| SET RPN RM05 | set type to RM05 |
| SET RPN RM80 | set type to RM80 |
| SET RPN RP04 | set type to RP04 |
| SET RPN RP06 | set type to RP06 |
| SET RPN RP07 | set type to RP07 |
| SET RPN AUTOSIZE | set type based on file size at attach |
| SET RPN 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 be set `ENABLED` or `DISABLED`. The RP controller supports the `BOOT` command.

The RP controller implements the registers listed below. Registers suffixed with [0:7] are replicated per drive.

| name | size | comments |
|----------|------|----------------------------------|
| CS1[0:7] | 16 | current operation |
| DA[0:7] | 16 | desired surface, sector |
| DS[0:7] | 16 | drive status |
| ER1[0:7] | 16 | drive errors |
| OF[0:7] | 16 | offset |
| DC[0:7] | 16 | desired cylinder |
| ER2[0:7] | 16 | error status 2 |
| ER3[0:7] | 16 | error status 3 |
| EC1[0:7] | 16 | ECC syndrome 1 |
| EC2[0:7] | 16 | ECC syndrome 2 |
| MR[0:7] | 16 | maintenance register |
| MR2[0:7] | 16 | maintenance register 2 (RM only) |
| HR[0:7] | 16 | holding register (RM only) |
| 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 RL11/RL01/RL02 Cartridge Disk (RL)

RL11 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 be set `ENABLED` or `DISABLED`. The RL11 supports the `BOOT` command.

The RL11 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..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 |

2.4.3 RK611/RK06/RK07 Cartridge Disk (HK)

RK611 options include the ability to set units write enabled or write locked, to set the drive type to RK06, RK07, or autosize, and to write a DEC standard 044 compliant bad block table on the last track:

```

SET HKn LOCKED           set unit n write locked
SET HKn WRITEENABLED    set unit n write enabled
SET HKn RK06             set type to RK06
SET HKn RK07             set type to RK07
SET HKn AUTOSIZE        set type based on file size at attach
SET HKn 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 be set `ENABLED` or `DISABLED`. The RK611 supports the `BOOT` command.

The RK611 implements these registers:

| name | size | comments |
|-------|------|------------------|
| HKCS1 | 16 | control/status 1 |

| | | |
|-----------|----|---------------------------------|
| HKWC | 16 | word count |
| HKBA | 16 | bus address |
| HKDA | 16 | desired surface, sector |
| HKCS2 | 16 | control/status 2 |
| HKDS[0:7] | 16 | drive status, drives 0-7 |
| HKER[0:7] | 16 | drive errors, drives 0-7 |
| HKDB[0:2] | 16 | data buffer silo |
| HKDC | 16 | desired cylinder |
| HKOF | 8 | offset |
| HKMR | 16 | maintenance register |
| HKSPR | 16 | spare register |
| INT | 1 | interrupt pending flag |
| ERR | 1 | error flag (CSR<15>) |
| DONE | 1 | device done flag (CSR1<7>) |
| IE | 1 | interrupt enable flag (CSR1<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 |

2.4.4 UDA50 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 UDA50 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 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 be set ENABLED or DISABLED. The RQ controllers 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 0-3 |
| PKTQ[0:3] | 5 | packet queue, units 0-3 |
| UFLG[0:3] | 16 | unit flags, units 0-3 |
| 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 |

2.5 Tapes

2.5.1 TM03/TE16/TU45/TU77 Magnetic Tapes (TU)

The TU controller implements the Massbus family of 800/1600bpi magnetic tape drives. TU options include the ability to set the drive type to one of three drives (TE16, TU45, or TU77), and to set the drives write enabled or write locked.

| | |
|----------------------|-------------------------------|
| SET TUn TE16 | set unit n drive type to TE16 |
| SET TUn TU45 | set unit n drive type to TU45 |
| SET TUn TU77 | set unit n drive type to TU77 |
| SET Tun LOCKED | set unit n write locked |
| SET Tun WRITEENABLED | set unit n write enabled |

Magnetic tape units can be set to a specific reel capacity in MB, or to unlimited capacity:

| | |
|-----------------|---|
| SET TUn CAPAC=m | set unit n capacity to m MB (0 = unlimited) |
| SHOW TUn CAPAC | show unit n capacity in MB |

Units can be set ENABLED or DISABLED. The TU controller does not support the BOOT command.

The TU controller implements the following registers:

| name | size | comments |
|----------|------|--------------------------|
| CS1 | 6 | current operation |
| FC | 16 | frame count |
| FS | 16 | formatter status |
| ER | 16 | formatter errors |
| CC | 16 | check character |
| MR | 16 | maintenance register |
| TC | 16 | tape control register |
| TIME | 24 | operation execution time |
| STOP_IOE | 1 | stop of I/O error |

Error handling is as follows:

| | |
|--------------|-----------------------------------|
| error | processed as |
| not attached | tape not ready; if STOP_IOE, stop |
| end of file | bad tape |
| OS I/O error | parity error; if STOP_IOE, stop |

2.5.2 TS11 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 TS11 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 |

2.5.3 TUK50 TMSCP Disk Controller (TQ)

The TQ controller simulates the TUK50 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 TUK50 supports 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 0-3 |
| PKTQ[0:3] | 5 | packet queue, units 0-3 |
| UFLG[0:3] | 16 | unit flags, units 0-3 |
| POS[0:3] | 32 | tape position, units 0-3 |
| OBJP[0:3] | 32 | object position, units 0-3 |
| 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 DZ11 Terminal Multiplexer (DZ)

The DZ11 is an 8-line terminal multiplexer. Up to 4 DZ11's (32 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 8, with a maximum of 32.

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 DZ11 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 DZ11's modem controls; the optional switch -a turns on active disconnects (disconnect session if computer clears Data Terminal Ready). Without modem control, the DZ 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 DZ11 implements these registers:

| name | size | comments |
|------------|------|--|
| CSR[0:3] | 16 | control/status register, boards 0..3 |
| RBUF[0:3] | 16 | receive buffer, boards 0..3 |
| LPR[0:3] | 16 | line parameter register, boards 0..3 |
| TCR[0:3] | 16 | transmission control register, boards 0..3 |
| MSR[0:3] | 16 | modem status register, boards 0..3 |
| TDR[0:3] | 16 | transmit data register, boards 0..3 |
| SAENB[0:3] | 1 | silos alarm enabled, boards 0..3 |
| RXINT | 4 | receive interrupts, boards 3..0 |
| TXINT | 4 | transmit interrupts, boards 3..0 |
| MDMTCL | 1 | modem control enabled |
| AUTODS | 1 | autodisconnect enabled |

The DZ11 does not support save and restore. All open connections are lost when the simulator shuts down or the DZ is detached.

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.