HP 3000 Simulator User's Guide

Robert M Supnik J. David Bryan

5-Jan-2018

Contents

Introduction

This manual documents the features and operation of the HP 3000 simulator. It is intended for use in conjunction with the *SIMH Users' Guide* manual, which

describes how to compile and run the simulator, as well as the general commands that may be entered at the Simulation Control Program (SCP) prompt.

The HP 3000 Computer System

Hewlett-Packard sold the HP 3000 family of general-purpose business computers from 1972 through 2001. There are two major divisions within this family: the *classic* 16-bit, stack-oriented CISC machines, and the *Precision Architecture* 32-bit, register-oriented RISC machines that succeeded them. All machines run versions of MPE, the *Multiprogramming Executive* operating system.

Within the classic division, there are two additional subdivisions, based on the method used for peripheral connections: the original *SIO* machines, and the later *HP-IB* machines. The I/O interfacing hardware differs between the two types of machines, as do the privileged I/O machine instructions. The user instruction sets are identical, as are the register sets visible to the programmer. The I/O drivers are different to account for the hardware differences, and therefore they run slightly different versions of MPE.

This implementation is a simulator for the classic SIO machines. This group consists of the 3000 CX, the Series I, Series II, and the HP 3000 Series III that is simulated here. The CX and the Series I, which is a repackaged CX, are essentially subsets of the Series II/III — a smaller instruction set, limited memory size, and lower-precision floating-point instructions. Simulation of these machines may be added in the future. Future simulation of the HP-IB machines (the Series 30 through 70) is desirable, as the latest MPE versions run only on these machines, but documentation on the internals of the HP-IB hardware controllers is nonexistent.

The CX and Series I support 64K 16-bit words of memory. The Series II supports up to 256K, divided into four banks of 64K words each, and the Series III extends this to 1024K words using 16 banks. Memory is divided into variable-length code and data segments, with the latter containing a program's global data area and stack.

Memory protection is accomplished by checking program, data, and stack accesses against segment base and limit registers, which can be set only by MPE. Bounds violations cause automatic hardware traps to handler routines within MPE. Some violations may be permitted; for example, a Stack Overflow trap may cause MPE to allocate a larger stack and then restart the interrupted instruction. Memory references are position-independent, so moving segments to accommodate expansion requires only resetting of the segment registers to point at the new locations. Code segments are fully reentrant and shareable, and both code and data are virtual, as the hardware supports absent code and data segment traps.

The classic 3000s are stack machines. Most of the instructions operate on the value on the top of the stack (TOS) or on the TOS and the next-to-the-top of

the stack (NOS). To improve execution speed, the 3000 has a set of hardware registers that are accessed as the first four locations at the top of the stack, while the remainder of the stack locations reside in main memory. A hardware register renamer provides fast stack pushes and pops without physically copying values between registers.

Simulator Files

The simulator sources are divided into a set of files for the Simulator Control Program and its support libraries, and a set of files for the HP 3000 device simulations; the latter reside in a subdirectory of the directory that contains the SCP files. The former set is common to all SIMH simulators, whereas the latter set is specific to the virtual machine being simulated. The files that make up this simulator are:

Subdirectory	***File***	***Contains***
HP3000	hp3000_cpu.h	CPU architectural declarations
	hp3000_cpu_fp.h	Floating-point interface declarations
	hp3000_cpu_ims.h	CPU-to-IOP/channel interface declarations
	hp3000_defs.h	System architectural declarations
	hp3000_io.h	Device-to-IOP/channel interface declarations
	hp_disclib.h	MAC/ICD disc controller simulator library declarations
	hp_tapelib.h	797x tape controller simulator library declarations
	hp3000_atc.c	Asynchronous Terminal Controller simulator
	hp3000_clk.c	System Clock simulator
	hp3000_cpu.c	CPU simulator
	hp3000_cpu_base.c	CPU base instruction set simulator
	hp3000_cpu_cis.c	CPU COBOL II Extended Instruction Set simulator
	hp3000_cpu_fp.c	CPU floating point instruction set simulator
	hp3000_ds.c	Cartridge Disc Interface simulator
	hp3000_iop.c	I/O processor simulator
	hp3000_lp.c	Line Printer Interface simulator
	hp3000_mem.c	Main memory subsystem simulator
	hp3000_mpx.c	Multiplexer Channel simulator
	hp3000_ms.c	Magnetic Tape Controller Interface simulator
	hp3000_scmb.c	Selector Channel Maintenance Board simulator
	hp3000_sel.c	Selector Channel simulator
	hp3000_sys.c	SCP interface
	hp_disclib.c	MAC/ICD disc controller simulator library
	hp_tapelib.c	797x tape controller simulator library
	hp3000_diag.txt	SIMH/HP 3000 diagnostics performance report
	hp3000_release.txt	HP 3000 simulator release notes

PDF files of the original HP 3000 hardware and software manuals are available from these repositories:

- Bitsavers — http://www.bitsavers.org/
- The HP Computer Museum — http://www.hpmuseum.net/

A software kit containing an HP 7920 disc image with MPE V/R preinstalled is available from:

- Bitsavers — http://www.bitsavers.org/bits/HP/HP_3000/

Simulator Features

The HP 3000 simulator contains the following device simulations:

Device Name	***Simulates***
CPU	30003B Series III Computer with up to 1024 KW of memory
IOP	30003B I/O Processor
SEL	30030C Selector Channel
ATCD	30032B Asynchronous Terminal Controller data interface
ATCC	30032B Asynchronous Terminal Controller control interface
SCMB1, SCMB2	30033A Selector Channel Maintenance Boards
MPX	30036B Multiplexer Channel
CLK	30135A System Clock
LP	30209A Line Printer Controller with one 2607/13/17/18 line printer
MS	30215A Magnetic Tape Controller with four 7970B/E drives
DS	30229B Disc Interface with eight 7905/7906/7920/7925 drives

One instance of each listed device may be installed in the simulated computer chassis. Some devices support multiple connected units. As an example, the DS device simulates a single 30229B Disc Interface that connects up to eight drives. However, a second 30229B connecting an additional eight drives is not supported.

The simulator has been tested with and supports the following operating systems:

- MPE V/R version E.01.00.

In addition, the simulator generally passes the HP 32230 Stand-Alone Diagnostics suite; see *hp3000_diag.txt* for details.

The simulator may be configured to stop for any of these conditions:

- Attempted execution of an instruction that enters an infinite loop (e.g., BR P+0).
- Attempted execution of a PAUS instruction.
- Attempted execution of an undefined instruction.
- Attempted execution of an unimplemented instruction.

MPE normally handles these conditions internally. However, when running diagnostics, it may be helpful to set one or more of these stop conditions. In particular, some diagnostics execute a PAUS instruction to wait for operator action, rather than external interrupts, or branch in an infinite loop to indicate an error condition. Unless the corresponding simulator stops are active, no indications of these conditions will be observed.

The simulator also provides extensive facilities for tracing CPU and I/O device operations.

Hardware-Equivalent Actions

The current implementation does not provide simulations of the CPU or peripheral device front panels. Instead, commands entered through the simulation console are used to perform hardware actions. The simulation commands that substitute for CPU front-panel actions are:

Hardware Front-Panel Action	***Equivalent Simulation Command***
Pressing the RUN/HALT button while halted	RUN
Pressing the RUN/HALT button while running	CTRL+E
Pressing the LOAD and ENABLE buttons	LOAD
Pressing the DUMP and ENABLE buttons	DUMP
Toggling the CPU RESET switch	RESET
Setting the PF/ARS switch to ENABLE	SET CPU ARS
Setting the PF/ARS switch to DISABLE	SET CPU NOARS
Displaying the Current Instruction Register	EXAMINE CIR
Displaying the System Switch Register	EXAMINE SWCH
Setting the System Switch Register	DEPOSIT SWCH <value>

Mounting media on a peripheral device is simulated by the ***ATTACH*** command. For example, entering the ***ATTACH LP <printer-image-filename>*** command is equivalent to loading paper into an HP 2617A Line Printer. Inserting a disc pack into an HP 7905A disc drive set for unit 2 is simulated by the ***ATTACH DS2 <disc-image-filename>*** command.

Each of these commands is explained in more detail below.

In hardware, loading programs into memory from a device is accomplished by setting the System Switch Register to the desired control and device number and pressing the LOAD and ENABLE buttons together to initiate the cold load sequence. In simulation, a cold load may be performed either explicitly or implicitly. Explicit operation is described in the LOAD, DUMP, and BOOT section below. As a convenience, ***BOOT <device>*** commands may be used to implicitly cold load their respective devices and are described in the individual device descriptions below.

Simulator-Specific Commands

In general, all of the commands documented in the *SIMH Users' Guide* manual are available for use with the HP 3000 simulator. Commands whose execution or parameters are implementation-defined are specified below.

Numeric Display and Entry

When examining or depositing into memory, the radix for addresses is octal, and the default radix for numeric data is octal. The data default may be changed with the ***SET CPU { BIN | OCT | DEC | HEX }*** command, or the radix may be overridden temporarily with a command line switch, as follows:

Switch	***Interpretation***
–2	A binary value
–8 or **–O**	An octal value
–10 or **–D**	A decimal value
–16 or **–H**	A hexadecimal value

When examining or depositing into device registers, the default radix for the specified register is used unless overridden with one of the above command line switches. Defaults are listed in the register table associated with each device.

Symbolic Display and Entry

When examining or depositing into memory or certain registers, command line switches specifying the symbolic mode and format may be used to override the default numeric mode, as follows:

Switch	***Mode Interpretation***
–A	A single character in the right-hand byte
–C	A two-character packed string
–E	An EDIT subprogram operation mnemonic
–ER	An EDIT subprogram operation mnemonic starting with the right-hand byte
–I	An I/O program instruction mnemonic
–M	A CPU instruction mnemonic
–T	A CPU status mnemonic

In the absence of a mode switch or a specified symbolic default, entering values with a leading ' (apostrophe) implies ***–A***, and a leading " (quotation mark) implies ***–C***. The specific registers supporting symbolic mode are indicated in their respective device sections below.

If the **−*C*** switch is specified, the value is displayed as two characters separated by a comma. Alphanumeric, punctuation, and symbol characters are displayed within apostrophes, control characters are displayed as ASCII name abbreviations, and characters above 128 decimal are displayed in escaped numeric form with a leading backslash followed by an octal number. Depositing with **−*C*** accepts two displayable characters. If a single character is supplied, the low byte of the resulting value will be zero; follow the character with a space to pad the low byte with a blank.

If the **−*M*** switch is specified, the value is displayed as a CPU machine instruction mnemonic if it is defined. If it is not, it is displayed as a numeric value in the CPU's data radix. Any numeric operands present are displayed in a default radix unless overridden by the addition of one of these mutually exclusive format switches:

Switch	***Format Interpretation***
−A	A single character in the right-hand byte
−B	A binary value
−O	An octal value
−D	A decimal value
−H	A hexadecimal value

Numeric operands are displayed in a radix suitable to the type of the value. For CPU instruction operands:

- Register-relative displacements, S-register decrements, and K fields are displayed in the CPU's address radix, which is octal.
- Shift counts, bit positions, and starting bits and counts are displayed in decimal unless overridden by a switch on the command line.
- CIR values for the PAUS and HALT instructions are displayed in octal unless overridden by a switch on the command line.
- Immediate values are displayed in the CPU's data radix, which defaults to octal but may be set to a different radix or overridden by a switch on the command line.

For I/O program instruction operands:

- Address values are displayed in the CPU's address radix, which is octal.
- Counts are displayed in decimal unless overridden by a switch on the command line.
- Control and status values are displayed in the CPU's data radix, which defaults to octal but may be set to a different radix or overridden by a switch on the command line.

For CPU status values:

- The current code segment number is displayed in the CPU's data radix, which defaults to octal but may be set to a different radix or overridden by a switch on the command line.

For EDIT subprogram operands:

- Branch displacement fields are displayed in the CPU's address radix, which is octal.
- Character counts, loop counts, and source and target adjustments are displayed in decimal unless overridden by a switch on the command line.
- Invalid operation codes are displayed in the CPU's data radix, which defaults to octal but may be set to a different radix or overridden by a switch on the command line.

Each EDIT subprogram operation occupies from 1 to 257 bytes in memory. If a multi-byte operation begins within the examined memory range, the additional bytes required to complete the operation are displayed. Including the **−*R*** switch will begin decoding with the right-hand (low-order) byte of the first memory word in the range rather than with the left-hand byte. Invalid extended operation codes are displayed as a comma-separated pair of half-byte values (e.g., "17,12").

If the indicated display radix is not the same as the current CPU data radix, a leading character ***@****, %, #, or* ***!****, is printed for binary, octal, decimal, or hexadecimal numbers, respectively.

When the simulator examines the bit patterns of instructions in memory or registers to display in mnemonic form, each will fall into one of four categories:

1. Defined (canonical) instruction encodings, where all bits are defined or all reserved bits are zero (e.g., LOAD).
2. Undefined (non-canonical) instruction encodings, where reserved fields are "don't care" bits (e.g., MOVE).
3. Undefined (non-canonical) instruction encodings, where reserved fields are decoded (e.g., IXIT).
4. Unimplemented instruction encodings (e.g., stack opcode 072, or EADD ithout the EIS firmware option installed).

The names of the canonical instructions in category 1 are displayed in uppercase, as are the names of the non-canonical instructions in category 2. The non-canonical instruction names in category 3 are displayed in lowercase. This is to indicate to the user that the instructions that will be executed may not be the ones expected from the decoding. Instruction names in category 4 that correspond to supported firmware options are displayed in uppercase, regardless of whether or not the option is enabled. Category 4 encodings that do not correspond to instructions are displayed in octal.

Memory Addressing

The CPU simulator supports two forms of memory addresses:

- An absolute address consisting of a 4-bit bank number and a 16-bit offset within the bank, separated by a period (e.g., ***17.177777***).
- A relative address consisting of a 16-bit offset within a bank specified by a bank register (e.g., ***177777***).

Command line switches modify the interpretation of relative addresses as follows:

Switch	***Interpretation***
–P	The implied bank number is obtained from PBANK
–S	The implied bank number is obtained from SBANK

If no switch is specified, the implied bank number is obtained from DBANK.

EXAMINE, DEPOSIT, IEXAMINE, and IDEPOSIT The following address forms are valid:

```
EXAMINE <bank>.<offset>
EXAMINE <dbank-offset>
EXAMINE -P <pbank-offset>
EXAMINE -S <sbank-offset>
```

Addresses are always displayed in *<bank>.<offset>* form.

BREAK and NOBREAK The following address forms are valid:

```
BREAK
BREAK <bank>.<offset>
BREAK <pbank-offset>
```

The address defaults to the current values of PBANK and/or P. All breakpoint addresses are stored and displayed in *<bank>.<offset>* form.

RUN and GO The following address form is valid:

```
RUN <pbank-offset>
```

The offset is stored in P. If P does not lie within the PB-to-PL range, the command will be rejected.

The SIMH ***RUN*** command typically resets all devices before beginning execution. For the HP 3000, CPU and I/O resets are performed at the start of the cold load sequence; a reset after the cold load sequence completes will destroy the Interrupt Control Stack register setup performed by the microcode. Therefore, in this simulator, the ***RUN*** and ***GO*** commands are equivalent; neither resets

the devices. If an explicit CPU reset is desired, the ***RESET CPU*** command may be used.

LOAD, DUMP, and BOOT

The ***LOAD*** command implements the 3000 cold load facility. It is equivalent to pressing the LOAD and ENABLE buttons on the hardware front panel. The syntax is:

```
LOAD {<control/devno>}
```

The SWCH register controls the cold load sequence; the upper byte contains a device-specific control value, and the lower byte contains the device number. If the optional control/device number value is supplied, it is deposited into the SWCH register before initiating the cold load sequence. Otherwise, the SWCH register value must be set before entering the command. In either case, the device number must be between 3 and 63, or the command will not complete.

The ***DUMP*** command implements the 3000 cold dump facility. It is equivalent to pressing the DUMP and ENABLE buttons on the hardware front panel. The syntax is:

```
DUMP {<control/devno>}
```

The CPU ***DUMPCTL*** and ***DUMPDEV*** settings control the cold dump sequence and correspond to the settings of the jumper blocks on the rear of the CPU front panel. If the optional control/device number value is supplied, it is deposited into the SWCH register before initiating the cold dump sequence. Otherwise, the ***DUMPCTL*** and ***DUMPDEV*** values are deposited into the SWCH register. A successful cold dump sets the CIR register to the number of 64K memory banks dumped. For example, dumping a 512K-word system will display Cold dump complete, CIR: 000010 if the operation succeeds.

The control/device number values are entered in the current CPU data radix. Before entering either command, unit 0 of the associated device must be attached. A failed cold load will cause a system halt. A failed cold dump will set the CIR register to the memory bank number at which the operation failed; if the dump device is not ready or not write-enabled, the CIR register will be set to zero.

Bootable peripherals, such as the tape and disc drives, also implement the ***BOOT*** command. Entering a ***BOOT*** command for one of these devices will preset the SWCH register to the appropriate value before executing a cold load. Cold load supports booting only from unit 0 of the target device.

POWER FAIL and POWER RESTORE

The ***POWER FAIL*** command (and its alternate forms ***POWER OFF*** and ***POWER DOWN***) simulates removing AC power from the system. If the CPU was executing when the command is entered, a power-fail interrupt is generated, and MPE initiates its power-fail processing that ends with a programmed HALT.

Otherwise, power is removed, and the CPU remains halted. ***POWER FAIL*** is accepted only when power is on.

The ***POWER RESTORE*** command (and its alternate forms ***POWER ON*** and ***POWER UP***) simulates restoring AC power to the system; it is accepted only when power is off. If the CPU was executing when power failed, a power-on trap is generated, and, if auto-restart is enabled, MPE initiates its power-recovery processing. If auto-restart is disabled, the trap is set up but the simulator must be resumed with a ***CONTINUE*** command to execute the trap. If the CPU was halted when power failed, power is restored, and the CPU remains halted. Once power is off, the ***CONTINUE***, ***STEP***, ***GO***, and ***RUN*** commands will not be accepted until power is restored.

Device Configuration

Most devices support user configuration. The general forms of the configuration commands are:

```
SET {<switch> ...} <device> <option> {,<option> ...}
SET {<switch> ...} <unit> <option> {,<option> ...}
```

The options available and applicable switches are described in the individual device descriptions below.

Enabling and Disabling Devices

All devices other than the processor devices may be disabled or enabled. Disabling a device simulates removing the associated interface from the system cabinet. To disable or enable a device, use:

Command	*Action*
SET <device> DISABLED	Disable the device
SET <device> ENABLED	Enable the device

Devices that consist of multiple addressable units connected to a controller allow the units to be individually disabled or enabled. Disabling simulates disconnecting the associated unit from the controller. The commands to disable or enable a unit are:

Command	*Action*
SET <unit> DISABLED	Disable the unit
SET <unit> ENABLED	Enable the unit

Each of the above command options is replicated in the option tables of the devices to which they apply.

I/O Interface Assignments

HP 3000 hardware I/O interfaces have configuration jumpers that set device numbers, interrupt masks, and Multiplexer Channel service request priority numbers. In addition, the interrupt poll order, and therefore the interrupt priority, is established by the position of each interface along the daisy-chained interrupt cable.

Device options that may be specified to change or display interface configurations are:

Option	***Valid Values***	***Action***
DEVNO	0-127	Set the device number
INTMASK	0-15, D, E	Set the interrupt mask
INTPRI	0-31	Set the interrupt priority
SRNO	0-15	Set the service request number

For example:

```
SET MS DEVNO=6
SET MS INTMASK=E
SHOW MS DEVNO,INTMASK
```

The interrupt mask may be set to a numeric value, to ***D*** to disable the mask always, or to ***E*** to enable the mask always. Numeric mask values may be shared among devices. However, if two device, interrupt priority, or service request priority numbers conflict, the simulator will report the error when execution is attempted.

The default settings are:

Device Name	***Device Number***	***Interrupt Priority***	***Interrupt Mask***	*****Service Request Number***
CPU	–	–	–	–
IOP	–	–	–	–
SEL	–	–	–	–
MPX	127	–	–	–
ATCD	7	0	E	–
ATCC	8	8	E	–
CLK	3	1	–	–
DS	4	4	E	–
LP	14	18	E	11
MS	6	14	E	3
SCMB1	65	10	–	0
SCMB2	66	11	–	1

The SCMB device numbers and interrupt priorities may be set to any unused values.

SAVE and RESTORE

SAVE and ***RESTORE*** are supported only when the simulator executable used to restore the simulator state file is the same simulator executable used to save the file. Correctly restoring the state of the simulator depends on the layout of internal structure variables being identical to the layout of the structure variables that were saved. This is guaranteed only when using the same executable, as the layout used is implementation-defined.

Realistic, Calibrated, and Optimized Timing

Devices simulate their I/O operation delays (disc seeks, magnetic tape reads, printer paper movements, etc.) by counting specified numbers of "event ticks." For example, a disc seek completion might be scheduled to occur 500 event ticks after command initiation. Generally, one event tick is counted for each machine instruction executed, although some complex instructions may count an event tick for each internal step of the instruction (e.g., for each word moved within a block-move instruction). Device simulations provide commands that determine how the appropriate tick counts are selected for I/O operations timing:

Command	***Action***
SET <device> REALTIME	Configure the device to use realistic timing
SET <device> CALTIME	Configure the device to use calibrated timing
SET <device> FASTTIME	Configure the device to use optimized timing

A device configured to use *realistic timing* selects its tick counts to encompass the same number of machine instructions as would be executed in hardware. In real-time mode, an operation taking ten milliseconds in hardware will complete after 4000 machine instructions are executed, based on an average instruction execution time of 2.5 microseconds. In this mode, a software program will execute approximately the same number of instructions during a device operation that it would do on a real machine. Host machine speed and execution of concurrent host programs will not affect the number of simulated instructions executed for a given operation.

A device configured to use *calibrated timing* selects its tick counts to align the simulated operation periods with the corresponding time periods on the host system. In calibrated-time mode, an operation taking ten milliseconds in hardware will complete after ten milliseconds has elapsed on the host system. Because the simulator is generally one or two orders of magnitude faster than the hardware, a software program will execute far more code during a device operation than it would do on a real machine. In this mode, the amount of code

that executes will vary with the speed of the host machine and the load placed on that machine by other concurrent processes, as the simulator continually adjusts the tick counts up and down to maintain synchronization with the host time.

A device configured to use *optimized timing* selects its tick counts to minimize the operation delays. In fast-time mode, an operation taking ten milliseconds in hardware will complete after the minimum amount of time acceptable to the executing software has elapsed. In this mode, device operations complete far more quickly than they do in hardware. There are limits, however, to how fast I/O operations may occur without causing software malfunctions. In practice, system software often contains assumptions regarding the time certain operations take, so event timing may not be arbitrarily reduced. For instance, an I/O driver may "know" that a line printer takes 50 milliseconds to print a line, and therefore it can ignore interrupts safely for several milliseconds after initiating the print cycle. If the printing time is reduced below that threshold, the driver may fail to operate correctly. The default optimized-timing settings have been empirically determined to work with the supported operating systems listed above, and each device simulator allows the user to modify those settings via registers if needed.

To illustrate how the modes affect timing, consider a simulation of a Teletype terminal that operates at 10 characters per second. If the simulator runs 15 times faster than a real machine, then a user would observe that printing 100 characters takes:

- 10 seconds in CALTIME mode
  (100 characters × n event ticks per character adjusted to take exactly 100 mS each on the host system)
- 667 milliseconds in REALTIME mode
  (100 characters × 40,000 event ticks per character × 2.5 µS per tick ÷ 15 times hardware speed)
- 8.33 milliseconds in FASTTIME mode
  (100 characters × 500 event ticks per character × 2.5 µS per tick ÷ 15 times hardware speed)

If the SCP ***SET THROTTLE*** command is used to reduce the speed of the simulator, CALTIME operations will not be affected, but REALTIME and FASTTIME operations will slow proportionally. Reducing simulator speed to that of the original hardware will cause REALTIME operation times to equal CALTIME times.

Devices offer only those modes that are generally useful. For example, the CPU process clock and system clock may be configured to use REALTIME or CALTIME modes. The real-time mode will satisfy the expectations of hardware diagnostics that check the timing of operations via delay loops, whereas the calibrated mode will update the MPE time-of-day clock as expected by users of the simulated system. FASTTIME mode is not offered, as it makes no sense to

ignore the programmed time period settings and use a fixed arbitrary period instead.

Devices performing input or output typically offer a REALTIME mode for use when running the diagnostics and a FASTTIME mode for use when running operating systems. In general, software running under simulation will run faster when devices are configured for optimized-time mode, and this is the default for all peripheral devices.

The Simulation Console and the System Console

When the simulator is started, the SCP command prompt appears at the *simulation console.* For windowed host operating systems, this is typically the same window from which the simulator was started. Once a ***RUN*** command is entered at the simulation console, the standalone diagnostics and MPE operating system begin their user interactions at the *system console.* The HP 3000 reserves channel 0 of the ATC terminal multiplexer for the system console.

For convenience and by default, the system console is connected to the simulation console, so that SCP and MPE operator commands may be entered from the same window. Additional "user" terminals may be connected via Telnet or serial ports to ATC channels 1-15 as described later.

The system console may be separated from the simulation console by using the ***SET CONSOLE TELNET=<port>*** or ***SET CONSOLE SERIAL=<port>*** command. This leaves the simulation console at the initiating window and moves the system console to a Telnet or serial port, allowing the use of an HP terminal or terminal emulator. Entering the ***SET CONSOLE NOTELNET*** or ***SET CONSOLE NOSERIAL*** command will rejoin the consoles

Tracing Simulator Operations

The simulator provides options for extensive tracing of the internal operations of all devices. This is useful as an aid to hardware and software debugging as well as to gain an understanding of the internal operations of the simulated devices. Each device offers multiple trace reporting levels, from command overviews to detailed backplane signal assertions. Tracing for each device and its separate reporting levels may be enabled independently.

To obtain a trace, two SCP commands must be given. First, a *debug log* must be established with the ***SET DEBUG <target>*** command. This command is described in detail in the "Controlling Debugging" section of the *SIMH Users' Guide* manual. Typically, the target is a text file, so that the trace may be reviewed after capture. Second, tracing must be enabled for the desired devices with ***SET <device> DEBUG=<option>*** commands. These are documented below in the sections that refer to the simulated devices.

The reporting level options table given for each device is arranged in order of increasing detail. The first option listed provides the broadest overview with the least specific detail and generates the smallest number of trace lines, thereby slowing program execution the least. Subsequent options provide increasing detail at the expense of larger debug log files. Enabling all trace options with a ***SET <device> DEBUG*** command provides the fullest picture of device operation but may generate very large log files.

Some options enable tracing of periodic events, e.g., a clock tick or a device poll. Use caution when specifying these options, as the trace log may fill rapidly. Options that trace periodic behavior are noted in the option descriptions.

The formats of the trace output are specific to the devices being traced. Examples are provided in each device description section below.

Processor Device Simulations

The HP 3000 computer consists of the following subsystems:

- 30003B Central Processing Unit
- 30003B I/O Processor
- 30030C Selector Channel
- 30036B Multiplexer Channel

Central Processing Unit

The HP 3000 Central Processing Unit contains the microprogrammed machine instruction execution unit, main memory, and process clock. CPU options specify the memory size, installed firmware, and simulation configuration. The CPU is configured with commands of the form:

```
SET {-F} CPU <option>
```

Device options that may be specified are:

Option	***Action***
128K	Set the memory size to 128K words
256K	Set the memory size to 256K words
384K	Set the memory size to 384K words
512K	Set the memory size to 512K words
768K	Set the memory size to 768K words
1024K	Set the memory size to 1024K words; default for Series III
CIS	Enable the COBOL II Extended Instruction Set firmware
NOCIS	Disable the COBOL II Extended Instruction Set firmware; default
ARS	Enable auto-restart after a power failure; default
NOARS	Disable auto-restart after a power failure

Option	*Action*
CALTIME	Use calibrated process clock timing; default
REALTIME	Use realistic process clock timing
IDLE	Enable idle detection
NOIDLE	Disable idle detection; default
DUMPDEV=<n>	Set the dump device number; default is 6
DUMPCTL=<n>	Set the dump device control value; default is 004
STOP=<option>	Enable simulation stops
NOSTOP	Disable simulation stops; default
EXEC=<match>[;<mask>]	Enable execution tracing of matching instructions
NOEXEC	Disable execution tracing; default
DEBUG=<option>	Enable tracing
NODEBUG	Disable tracing; default

The initial configuration is a Series III system with 1024K words of memory. If the memory size is being reduced, and the memory being truncated contains non-zero data, the simulator asks for confirmation before proceeding. The confirmation request may be suppressed by using the ***−F*** (force) switch. Data in the truncated portion of memory is lost.

The ***SET CPU CIS*** command simulates the installation of the HP 32234A COBOL II Extended Instruction Set firmware on the ROM PCA and removal of the W4 jumper from the CIR PCA to enable decoding of the instructions. The HP 32233A COBOLII/3000 compiler and programs produced by this compiler require the firmware to execute. If it is not installed, these programs will abort at run time with *ILLEGAL INSTRUCTION* errors.

The position of the PF/ARS switch on the back of the system control panel determines how the CPU responds to AC power returning after a power failure. In the *enable* position, simulated by the ***SET CPU ARS*** command, power restoration will initiate an automatic restart of the operating system. In the *disable* position, simulated by the ***SET CPU NOARS*** command, power restoration will set up the system for a restart, which must be performed manually by entering a ***CONTINUE***, ***STEP***, ***GO***, or ***RUN*** command.

Calibrated timing adjusts the process clock to match actual elapsed time. Realistic timing bases the process-clock interval on machine instructions executed.

When enabled by a ***SET CPU IDLE*** command, execution of a PAUS instruction will idle the simulator. While idle, the simulator does not use any host system processor time. Idle detection requires that the process clock and system clock be set to calibrated timing. Idling is disabled by default.

The ***DUMPDEV*** and ***DUMPCTL*** options configure the set of jumpers on the rear of the CPU front panel that preset the device number and control value for the cold dump process. These values are used when the ***DUMP*** command is entered without the optional control/device number. The device number range

is 0–127 decimal, and the control value range is 0–377 octal. The default control value corresponds to the Write Record tape command.

The ***SET CPU STOP*** command enables one or more simulation stop conditions. These are described in the Simulation Stops section below.

The ***SET CPU EXEC*** command configures instruction execution tracing. This command is described in the Tracing section below.

The CPU configuration may be displayed with the following commands:

Command	*Action*
SHOW CPU	Display the device configuration
SHOW CPU STOPS	Display the enabled simulation stops
SHOW CPU EXEC	Display the matching criteria for execution tracing
SHOW CPU DUMP	Display the cold dump device number and control value
SHOW CPU SPEED	Display the current simulation speed

When the process clock is calibrated, the current simulation speed, expressed as a multiple of the speed of a real HP 3000 Series III, may be obtained with the ***SHOW CPU SPEED*** command. The speed reported will not be representative if the simulator was executing a PAUS instruction when it was stopped.

System Halt

The MPE operating system is supported by special microcode features that perform code and data segment mapping, segment access bounds checking, privilege checking, etc. The layout of certain in-memory tables is known to both the OS and the microcode and is used to validate execution of instructions. For instance, every stack instruction is checked for a valid access within the stack segment boundaries, which are set up by the OS before program dispatch. For this reason, the 3000 cannot be operated as a "bare" machine, because these tables will not have been initialized. Similarly, the cold load process by which the OS is loaded from storage media into memory is entirely in microcode, as machine instructions cannot be executed until the required tables are loaded into memory.

This OS/microcode integration means that the microcode may detect conditions that make continued execution impossible. An example would be an absent segment fault for the segment containing the disc I/O driver. If such a condition is detected, the CPU does a *system halt*. This fatal microcode error, distinct from a regular programmed halt, causes operation to cease until the CPU is reset.

A system halt stops the simulator and reports the system halt code to the simulation console. The code indicates the reason for the system halt, as follows:

Code	*Reason for System Halt*
1	STT violation while executing in segment 1
2	Absent code segment while executing on the ICS
3	Code segment 1 is absent
4	Stack overflow while executing on the ICS
6	I/O device timeout while executing IXIT or cold load
9	Dispatcher already enabled while executing PSEB
13	CST violation while executing in segment 1
23	External interrupts disabled while executing LOCK
33	Attempt to trace segment 1

If a system halt has occurred, simulator execution may not be resumed until a ***RESET***, ***RESET CPU***, or ***LOAD*** command has been entered.

Idling

The PAUS instruction suspends instruction execution until an interrupt occurs. It is intended to defer instruction fetches from memory to allow full-bandwidth access by the Selector and Multiplexer Channels. If enabled, the simulator will idle only during execution of PAUS.

Simulation Stops

The simulator can be configured to detect certain machine instruction conditions and stop execution when one or more of them occur. Four stop options control the simulation stop conditions:

Option	*Stop Condition*
LOOP	Stop when an infinite loop is executed
PAUSE	Stop when a PAUS instruction is executed
UNDEF	Stop when an undefined instruction is executed
UNIMPL	Stop when an unimplemented instruction is executed

Setting the ***LOOP*** option stops the simulator if it attempts to execute an instruction that enters an infinite loop (e.g., BR P+0). The branch instructions TBA, TBX, BCC, BR, BCY, BNCY, BOV, and BNOV result in an infinite loop if the branch displacement is zero and the branch condition is true. The remaining branch instructions cannot result in an infinite loop, as they all modify the CPU state and so eventually will reach a point where they drop out of their loops.

Setting the ***PAUSE*** option stops the simulator if execution of a PAUS instruction is attempted. Setting the ***UNDEF*** option stops the simulator if execution of a non-canonical instruction (i.e., an instruction containing a decoded reserved

bit pattern other than that defined in the *Machine Instruction Set* manual) is attempted. Setting the ***UNIMPL*** option stops the simulator if execution of an unimplemented instruction is attempted.

After a simulation stop, execution may be resumed in one of two ways. If the cause of the stop has not been remedied and the stop option has not been disabled, resuming execution with ***CONTINUE***, ***STEP***, ***GO***, or ***RUN*** will cause the stop to occur again. Alternately, specifying the ***−B*** switch with any of the preceding commands will resume execution while bypassing the stop for the current instruction.

When the simulator examines the bit patterns of instructions to execute, each will fall into one of four categories:

1. Defined (canonical) instruction encodings, where all bits are defined or all reserved bits are zero (e.g., LOAD).
2. Undefined (non-canonical) instruction encodings, where reserved fields are "don't care" bits (e.g., MOVE).
3. Undefined (non-canonical) instruction encodings, where reserved fields are decoded (e.g., IXIT).
4. Unimplemented instruction encodings (e.g., stack opcode 072, or EADD without the EIS firmware option installed).

Instructions in categories 1 and 2 are always executed. The ***UNDEF*** option stops the simulator for instructions in category 3. The intent is to catch instructions containing reserved fields with values that change the meaning of those instructions. The ***UNIMPL*** option stops the simulator for instructions in category 4.

Bypassing a stop has the following effect:

Option	***Bypass Action***
LOOP	Resume execution of the infinite loop
PAUSE	Resume execution with the instruction following PAUS
UNDEF	Resume execution with decoding as in the hardware
UNIMPL	Resume execution with an Unimplemented Instruction trap

Note that the ***PAUSE*** action corresponds in hardware to pressing the HALT button and then the RUN button. If the stop conditions are disabled, the simulator performs the actions above, except that PAUS suspends instruction execution until a device interrupt occurs.

Tracing

When debug output logging is enabled, tracing may be configured by specifying one or more of the reporting level options:

Option	*Reporting Level*
INSTR	Machine instructions executed
DATA	Memory data accesses
FETCH	Memory instruction fetches
REG	Register values
OPND	Memory operand values
EXEC	Matching instruction execution states
PSERV	Process clock service events (periodic)

A section of an example trace is:

```
>>CPU fetch: 00.010342 020320 instruction fetch
>>CPU instr: 00.010341 000300 ZROX,NOP
>>CPU reg: 00.006500 000000 X 000000, M i t r o c CCG
>>CPU fetch: 00.010343 041100 instruction fetch
>>CPU instr: 00.010342 020320 PLDA
>>CPU data: 00.000000 001340 absolute read
>>CPU reg: 00.006500 000001 A 001340, X 000000, M i t r o c CCG
>>CPU fetch: 00.010344 037777 instruction fetch
>>CPU instr: 00.010343 041100 LOAD DB+100
>>CPU data: 00.002100 123003 data read
>>CPU reg: 00.006500 000002 A 123003, B 001340, X 000000, M i t r o c CCL
>>CPU fetch: 00.010345 023404 instruction fetch
>>CPU instr: 00.010344 037777 ANDI 377
>>CPU reg: 00.006500 000002 A 000003, B 001340, X 000000, M i t r o c CCG
>>CPU fetch: 00.010346 002043 instruction fetch
>>CPU instr: 00.010345 023404 MPYI 4
>>CPU reg: 00.006500 000002 A 000014, B 001340, X 000000, M i t r o c CCG
>>CPU fetch: 00.010347 020320 instruction fetch
```

The ***INSTR*** option traces instruction executions. Each instruction is printed before it is executed. The two opcodes of a stack instruction are printed together before the left-hand opcode is executed. If the right-hand opcode is not NOP, it is reprinted before execution, with dashes replacing the just-executed left-hand opcode.

The ***DATA*** option traces reads from and writes to memory. Each access is classified by the memory bank register that is paired with the specified offset — DMA, absolute, program, data, or stack. DMA (channel) accesses derive their bank addresses from the banks specified by Set Bank I/O program orders. Absolute accesses always use bank 0. Program, data, and stack accesses use the bank addresses in the PBANK, DBANK, and SBANK registers, respectively.

The ***FETCH*** option traces instruction fetches from memory. These accesses are separated from those traced by the ***DATA*** option because fetches usually are of little interest except when debugging the fetch/execute sequence. Because

the HP 3000 has a two-stage pipeline, fetches load the NIR (Next Instruction Register) with the instruction after the instruction about to be executed from the CIR (Current Instruction Register).

The ***REG*** option traces register values. Two sets of registers may be printed. After executing each instruction, the currently active TOS registers, the index register, and the status register are printed. After executing an instruction that may alter the base registers, the program, data, and stack segment base registers are printed.

The ***OPND*** option traces memory byte operand values. Some instructions take memory and register operands that are difficult to decode from ***DATA*** or ***REG*** traces. This option presents these operands in a higher-level format. The memory bank and address values are always those of the operands. The operand data and values printed are specific to the instruction. For example, the ALGN instruction prints its source and target operands, digit counts, and fraction counts, and the EDIT instruction displays its subprogram operations.

The ***EXEC*** option traces the execution of instructions that match user-specified criteria. When a match occurs, all CPU trace options are turned on for the duration of the execution of the matched instruction. The prior trace settings are restored when a match fails. This option allows detailed tracing of specified instructions while minimizing the log file size compared to a full instruction trace.

The ***SET CPU EXEC*** command configures the match and mask values used to qualify instructions for execution tracing. Qualification is performed by ANDing the current instruction with the specified *mask* value and then comparing the result with the specified *match* value. If the *mask* value is omitted, the *match* value must match the instruction exactly. The values are entered in the CPU's data radix, which defaults to octal but may be set to a different radix or overridden by a switch on the command line.

Setting the *mask* value allows matching a range of instructions or an instruction with a range of operand values. For example, ***SET CPU EXEC=020460;177760*** will trace execution of all COBOL-II firmware instructions, ***SET CPU EXEC=054000;077000*** will trace execution of FDIV instructions, and ***SET CPU EXEC=031000*** will trace execution of PCAL 0 instructions. To trace a stack instruction, configure the match for the opcode in the left-hand position only; this will match the opcode in either position.

The ***PSERV*** option traces process clock event service entries. Each trace reports whether the CPU was executing on the Interrupt Control Stack or the user stack when the process clock ticked. Execution on the ICS implies that the operating system is executing. As the process clock ticks every millisecond, enabling ***PSERV*** tracing can quickly produce a large number of trace lines.

The trace formats are interpreted as follows:

```
>>CPU instr: 00.010341 000300 ZROX,NOP

Items:
    Octal bank (PBANK)
    Octal address (P)
    Octal data (instruction opcode)
    Instruction mnemonic(s)

>>CPU instr: 00.001240 000006 external interrupt
>>CPU instr: 00.023736 000000 unimplemented instruction trap

Items:
    Octal bank (PBANK) at interrupt
    Octal address (P) at interrupt
    Parameter
    Interrupt classification

>>CPU data: 00.002100 123003 data read
>>CPU data: 00.000000 001340 absolute read

Items:
    Octal bank (PBANK, DBANK, or SBANK)
    Octal address (effective address)
    Octal data (memory contents)
    Memory access classification

>>CPU fetch: 00.010342 020320 instruction fetch

Items:
    Octal bank (PBANK)
    Octal address (P + 1)
    Memory access classification

>>CPU reg: 00.006500 000002 A 123003, B 001340, X 000000, M i t r o c CCL
Items:
    Octal bank (SBANK)
    Octal stack memory address (SM)
    Octal stack register count (SR)
    Register values (0-4 TOS registers, X, STA)

>>CPU reg: 00.000000 000001 PB 010000, PL 025227, DL 001770, DB 002000, Q 006510, Z 007000
Items:
    Octal bank (DBANK)
    Zero
    Current code segment number (from STA)
    Base register values
```

```
>>CPU opnd: 00.135771 000000 DFLC +,!
>>CPU opnd: 00.045071 000252 target fraction 3 length 6, "002222"
Items:
    Octal bank (PBANK, DBANK, or SBANK)
    Octal address (effective address)
    Instruction-specific data value
    Instruction-specific operand value

>>CPU pserv: Process clock service entered on the ICS

Service entry and stack status
```

For ***OPND*** traces of byte-array operands, the data values printed are the relative byte addresses. For EDIT subprogram operations, the value of the loop counter is printed.

Enabling CPU tracing can produce a very large number of lines very quickly, so care should be used to enable tracing only around the area of interest. Breakpoint actions may be used to implement this; for example:

```
BREAK 100; SET CPU DEBUG; GO
BREAK 200; SET CPU NODEBUG; GO
```

These commands will enable tracing when the program counter reaches location 100 and disable tracing when it reaches location 200, thereby producing a trace of instructions executed between locations 100 and 200. Alternately, if the execution of specific instructions is of interest, the ***EXEC*** trace option may be used to reduce the debug log file size.

Registers

The CPU state contains the registers visible to the programmer and the interrupt system control registers:

Name	*Size*	*Radix*	*Symbolic*	*Read-Only*	*Description*
CIR	16	–	Y	Y	Current Instruction Register
NIR	16	–	Y	Y	Next Instruction Register
PB	16	8			Program Base Register
P	16	8			Program Counter
PL	16	8			Program Limit Register
PBANK	4	8			Program Segment Bank Register
DL	16	8			Data Limit Register
DB	16	8			Data Base Register
DBANK	4	8			Data Segment Bank Register
Q	16	8			Stack Marker Register
SM	16	8			Stack Memory Register

Name	*Size*	*Radix*	*Symbolic*	*Read-Only*	*Description*
SR	3	8			Stack Register Counter
Z	16	8			Stack Limit Register
SBANK	4	8			Stack Segment Bank Registe
RA	16	8	Y		Top of Stack Register
RB	16	8	Y		Top of Stack – 1 Register
RC	16	8	Y		Top of Stack – 2 Register
RD	16	8	Y		Top of Stack – 3 Register
X	16	8	Y		Index Register
STA	16	–	Y		Status Register
SWCH	16	8	Y		Switch Register
CPX1	16	8			Run-Mode Interrupts Register
CPX2	16	8			Halt-Mode Interrupts Register
PCLK	16	8			Process Clock Register

The CIR and NIR registers default to CPU instruction mnemonic format, and the STA register defaults to CPU status mnemonic format for display and entry but may be overridden with a numeric-format switch, if desired. The RA, RB, RC, RD, X, and SWCH registers may be examined or deposited using any of the modes described in the Symbolic Display and Entry section above.

Two additional, hidden, read-only registers may be displayed if mentioned explicitly. CNTR represents the hardware counter used by the microcode. When a PAUS or HALT instruction is executed, the microcode stores the value of the SR register in the CNTR register before flushing the TOS registers to memory (so SR is always zero after executing either instruction). The MOD register holds the module number during an unsolicited module interrupt. CNTR and MOD are provided for the CPU diagnostic and have no other use under simulation.

I/O Processor

The HP 30003B I/O Processor works in conjunction with the CPU and Multiplexer Channel to service the device interfaces. All I/O interfaces are connected to the IOP bus, which transfers direct I/O orders to the interfaces and handles memory reads and writes between the interfaces and the CPU stack. In addition, it provides the memory interface for Multiplexer Channel transfers, as well as fetching I/O program orders from main memory for the channel.

The IOP device has no configuration options. It does provide tracing, though, with selectable filtering by device number, using these device options:

Option	*Action*
FILTER=<list>	Suppress tracing for device numbers in the list
NOFILTER	Enable tracing for all device numbers; default
DEBUG=<option>	Enable tracing

Option	*Action*
NODEBUG	Disable tracing; default

Enabling IOP tracing can produce a very large number of trace lines very quickly, so care should be used to enable tracing only around the area of interest. If only certain devices are of interest, others may be filtered out of the results by specifying their device numbers in a ***SET IOP FILTER*** command. This command takes a list of individual device numbers separated by semicolons and ranges of the form ***low-high***. For example, ***SET IOP FILTER=3;7-9*** would exclude lines pertaining to device numbers 3, 7, 8, and 9 from the trace report.

When debug output logging is enabled, tracing may be configured by specifying one or more of the reporting level options:

Option	*Reporting Level*
DIO	Direct I/O orders issued
IRQ	Interrupt requests received
DATA	Multiplexer Channel memory data accesses

The ***DIO*** option traces direct I/O orders that are sent to devices. The ***IRQ*** option traces interrupt requests received and granted. The ***DATA*** option traces memory accesses performed on behalf of the Multiplexer Channel. Each access is classified as either an absolute access in memory bank 0 or a DMA access in the memory bank specified by a Set Bank I/O program order.

The trace formats are interpreted as follows:

```
>>IOP dio: Test I/O order sent to device number 3

  I/O order and device number

>>IOP irq: Device number 6 acknowledged interrupt request at priority 14

  Device number and interrupt priority number

>>IOP data: 00.001400 040000 dma write
>>IOP data: 00.000030 001434 absolute read

  Octal bank number
  Octal address
  Octal data (memory contents)
  Memory access classification
```

The I/O Processor state contains these registers:

Name	*Size*	*Radix*	*Read-Only*	*Description*
IOA	8	8	Y	I/O Address Register

Selector Channel

The HP 30030C Selector Channel provides high-speed data transfer between a device and main memory. While several interfaces may be connected to the selector channel bus, only one transfer is active at a time, and the channel remains dedicated to that interface until the transfer is complete. The channel contains its own memory port controller, so transfers to and from memory bypass the I/O Processor.

Once started by an SIO instruction, the channel executes I/O programs independently of the CPU. Program words are read, and device status is written back, directly via the port controller.

The SEL device has no configuration options. It does provide tracing, though, using these device options:

Option	*Action*
DEBUG=<option>	Enable tracing
NODEBUG	Disable tracing; default

When debug output logging is enabled, tracing may be configured by specifying one or more of the reporting level options:

Option	*Reporting Level*
CSRW	Channel command initiations and completions
PIO	Programmed I/O orders executed
STATE	Channel state changes executed
SR	Service requests received
DATA	Channel memory data accesses

The ***CSRW*** option traces the beginning and ending of channel programs. The ***PIO*** option traces programmed I/O orders that are sent to devices. The ***STATE*** option traces the entries into each of the internal channel execution states. The ***SR*** option traces service requests received from the device. The ***DATA*** option traces memory accesses performed by the port controller. Each access is classified as either an absolute access in memory bank 0 or a DMA access in the memory bank specified by a Set Bank I/O program order.

The trace formats are interpreted as follows:

```
>>SEL csrw: Device number 4 asserted REQ for channel initialization
>>SEL pio: Channel loaded IOCW 040000 (Control) from address 102757
>>SEL state: Channel entered the Fetch sequence with 14 clock cycles remaining
>>SEL sr: Device number 4 asserted CHANSR

  Operational message

>>SEL data: 00.041746 000000 dma write
>>SEL data: 00.000020 102757 absolute read

  Octal bank number
  Octal address
  Octal data (memory contents)
  Memory access classification
```

The Selector Channel state contains these registers:

Name	***Size***	***Radix***	***Symbolic***	***Description***
IDLE	1	2		Channel is inactive
SREQ	1	2		Channel is requesting service
DEVNO	8	10		Device number of the active interface
EXCESS	32	10		Channel cycles used in excess of allocati
SEQ	3	10		Current sequencer state
ORDER	4	8		Current SIO order
ROLOVR	1	2		Word count has rolled over
PFCNTL	1	2		Control word should be prefetched
PFADDR	1	2		Address word should be prefetched
BANK	4	8		Memory bank
WCOUNT	12	10		Word count
PCNTR	16	8		I/O Program Counter
CNTL	16	8		I/O Control Word
CNBUF	16	8		I/O Control Word buffer
ADDR	16	8		I/O Address Word
ADBUF	16	8		I/O Address Word buffer
INBUF	16	8	Y	Input buffer
OUTBUF	16	8	Y	Output buffer

The INBUF and OUTBUF registers may be examined or deposited using any of the modes described in the Symbolic Display and Entry section above.

Multiplexer Channel

The HP 30036B Multiplexer Channel provides high-speed data transfer between from one to sixteen devices and main memory. Concurrent transfers for multiple devices are multiplexed on a per-word basis, dependent on the service request

priorities assigned to the participating interfaces. Interfaces must have additional hardware to be channel-capable, as the channel uses separate control and data signals from those used for direct I/O. In addition, the multiplexer and selector channels differ somewhat in their use of the signals, so interfaces are generally designed for use with one or the other (the Selector Channel Maintenance Board is a notable exception that uses jumpers to indicate which channel to use).

Once started by an SIO instruction, the channel executes I/O programs independently of the CPU. Program words are read, and device status is written back, by calls to the I/O Processor.

The MPX device supports configuration of the device number used for the diagnostic interface and of tracing using these device options:

Option	***Action***
DEVNO=<n>	Set the device number; default is 127
DEBUG=<option>	Enable tracing
NODEBUG	Disable tracing; default

When debug output logging is enabled, tracing may be configured by specifying one or more of the reporting level options:

Option	***Reporting Level***
CSRW	Channel control, status, read, and write actions
PIO	Programmed I/O commands issued
STATE	Channel state changes executed
SR	Service requests received
IOBUS	I/O bus signals and data words received and returned

The ***CSRW*** option traces the beginning and ending of channel programs, as well as control, status, read, and write commands sent to the diagnostic interface. The ***PIO*** option traces programmed I/O orders that are sent to devices. The ***STATE*** option traces the entries into each of the internal channel execution states. The ***SR*** option traces service requests received from the devices. The ***IOBUS*** option traces the I/O backplane signals and data received and returned via the diagnostic interface. As the Multiplexer Channel uses the I/O Processor as its interface to main memory, channel memory access tracing is enabled by the IOP ***DATA*** trace option.

Examples of the trace formats follow:

```
>>MPX  csrw: Device number 65 asserted REQ for channel initialization
>>MPX  csrw: Control is address RAM | load registers | RAM address 15
>>MPX   pio: Channel SR 0 loaded IOCW 050000 (Sense) from address 021207
>>MPX state: Channel SR 3 entered State C
```

```
>>MPX    sr: Device number 65 asserted SR0
>>MPX iobus: Received data 000200 with signals DCONTSTB
```

The control, status, read, and write values, and the I/O bus signals are decoded and presented in symbolic format for easier interpretation.

The Multiplexer Channel state contains these registers:

Name	***Size***	***Radix***	***Description***
IDLE	1	2	Channel is inactive
COUNT	32	10	Count of active transfers
EXCESS	32	10	Channel cycles used in excess of allocatio
CNTL	16	8	Control word
STAT	16	8	Status word
ROLOVR	1	2	Word Count Rollover flip-flop
DEVEND	1	2	Device End flip-flop
STATR [**0:15**]	4	2	State RAM, SR 0-15
AUX [**0:15**]	6	8	Auxiliary RAM, SR 0-15
ORDER [**0:15**]	4	8	I/O Order RAM, SR 0-15
CNTR [**0:15**]	16	8	Counter RAM, SR 0-15
ADDR [**0:15**]	16	8	Address RAM, SR 0-15

Programmed I/O Device Simulations

The CPU controls these I/O device interfaces with direct I/O instructions:

- 30032B Asynchronous Terminal Controller
- 30033A Selector Channel Maintenance Board
- 30135A System Clock

30032B Asynchronous Terminal Controller

The HP 30032B Asynchronous Terminal Controller is a 16-channel terminal multiplexer used with the HP 3000 CX through Series III systems. The ATC connects from 1 to 16 serial terminals or modems to the HP 3000 at programmable baud rates from 75 to 2400 bits per second. Character sizes are also programmable from 5 to 12 bits in length, including the start and stop bits. Each channel can be independently configured, including for separate send and receive rates. The ATC is not buffered, so the CPU has to retrieve each character from a given channel before the next character arrives. To avoid saturating the CPU with interrupt requests, the ATC maintains an internal "mini-interrupt" system that queues requests and holds additional interrupts off until the CPU acknowledges the current request.

The HP 3000CX and Series I use a dedicated serial interface for the system console, while user terminals are connected to the ATC. For the Series II and

III, the separate card is eliminated, and channel 0 of the ATC is reserved for the console.

The ATC consists of two devices:

- ATCD — the Terminal Data Interface (TDI)
- ATCC — the Terminal Control Interface (TCI)

The Terminal Data Interface provides the serial data line connections for terminals and data sets. Five additional receive-only auxiliary channels may be connected as a group under software control to one of the sixteen main lines to detect the incoming baud rate. The Terminal Control Interface provides serial control and status lines for Bell 103 data sets.

Terminal Data Interface

The TDI provides the Transmitted Data (BA) and Received Data (BB) lines for up to sixteen terminals. It performs input and output through Telnet sessions connected to a user-specified port or through individually specified host serial ports. The TDI supports concurrent Telnet and serial connections. The ***ATTACH*** command specifies the local port to be used for Telnet connections:

```
ATTACH ATCD <port>
```

. . . where ***port*** is a decimal number between 1 and 65535 that is not being used for other TCP/IP activities. When the TDI is attached and the simulator is running, the multiplexer listens for connections on the specified port and assigns them to channels in ascending numeric order.

The ***ATTACH*** command is also used to specify the host serial port for individual TDI channels:

```
ATTACH {--V} ATCD
LINE=<chan>,CONNECT=<name>{;<rate>-<size><parity><stopbits>}
```

. . . where ***chan*** is the TDI channel number from 1-15, and ***name*** is the host name of the serial port to use (e.g., *ser0* or *COM1*). If the $-\boldsymbol{V}$ (verbose) option is included, a connection confirmation message will be output to the port.

An optional serial port configuration string may be supplied after the host name. The required values are:

- ***rate*** is the baud rate in bits per second.
- ***size*** is the character size in bits including the parity bit, if designated.
- ***parity*** designates the parity to use: *N* (no), *E* (even), *O* (odd), *M* (mark), or *S* (space).
- ***stopbits*** is the number of stop bits (*1*, *1.5*, or *2*).

Serial connections default to 9600 baud, 8-bit characters, no parity, and one stop bit.

TDI configuration options are available for the device and for the individual units. The command forms are:

```
SET ATCD <device-option>
SET ATCDn <unit-option>
```

Device options that may be specified are:

Option	***Action***
FASTTIME	Use optimized timing; default
REALTIME	Use realistic timing
TERMINAL	Connect using Telnet or serial ports; default
DIAGNOSTIC	Connect using diagnostic test cables
DEVNO=<n>	Set the device number; default is 7
INTMASK=<n>	Set the interrupt mask; default is E
INTPRI=<n>	Set the interrupt priority; default is 0
DEBUG=<option>	Enable tracing
NODEBUG	Disable tracing; default
ENABLED	Enable the device; default
DISABLED	Disable the device

The TDI supports programmable data transfer rates from 75 to 2400 baud. When realistic timing is enabled, the simulation accurately models the transmission and reception baud rates (in machine instructions). For example, a port set for 1200 baud will take twice as long to output a given listing as a port set for 2400 baud. Optimized timing reduces the timing to the minimum necessary to operate correctly; this is much faster than the real terminal multiplexer would operate.

The delay time used by the simulation in ***FASTTIME*** mode may be set via the register interface. This value may be adjusted as necessary to work around any HP 3000 software problems that are triggered by the abnormally rapid TDI operation. Resetting the device with the ***RESET –P*** (power-on reset) command restores the original optimized time.

Enabling the diagnostic mode simulates the installation of eight HP 30062-60003 diagnostic test (loopback) cables between channels 0-1, 2-3, etc., as required by the multiplexer diagnostic. Each cable connects the Send Data line of one channel to the Receive Data line of the other channel, and vice versa. In addition, all sessions are disconnected, and the multiplexer is detached from the Telnet listening port. While in diagnostic mode, the ***ATTACH ATCD*** command is not allowed. Enabling terminal mode allows the multiplexer to be attached to accept incoming connections again.

Unit options that may be specified for individual TDI channels are:

Option	*Action*
LOCALACK	Discard ENQ and reply with ACK internally; default
REMOTEACK	Transmit ENQ and receive ACK from the remote device
CAPSLOCK	Upshift lowercase input characters to uppercase; default for channel 0
NOCAPSLOCK	Input characters are unchanged; default for channels 1-15
UC	Upshift lowercase output characters to uppercase
7B	Output with high-order bit cleared; default for channels 1-15
7P	Output with high-order bit cleared, non-printing suppressed; default for channel 0
8B	Output characters without changing
LOG=<filename>	Enable output logging
NOLOG	Disable output logging; default
DISCONNECT	Disconnect the channel

Channels that are configured for MPE terminal type 10 expect HP terminals or terminal emulators to be connected and will handshake transfers by sending an ENQ and expecting an ACK in reply. A device that does not provide ENQ/ACK handshaking will hang during output. However, if ***LOCALACK*** is specified, the handshake will take place locally within the simulator. The ENQ will be discarded, and a locally generated ACK will be returned to the caller. In addition to enabling the use of non-HP terminals, this option significantly improves the performance of common HP terminal emulators. Specifying ***REMOTEACK*** will pass ENQ to the terminal for handling; this is necessary to avoid output overruns if a real HP terminal is connected to a serial port.

Some HP 3000 programs, e.g., SLEUTH, require command input in upper case, although mixed-case output is supported. As an aid to avoid toggling the host keyboard in and out of CAPS LOCK mode, the multiplexer provides this function locally. The default modes are ***CAPSLOCK*** for channel 0 (the system console) and ***NOCAPSLOCK*** for the other channels.

Each channel may be set to one of four output modes (***UC***, ***7B***, ***7P***, or ***8B***). The default mode is ***7P*** for channel 0 and ***7B*** for the other channels. If the system console is set to MPE terminal type 10, the mode must be changed to ***7B*** or ***8B*** to allow ESC, DC1, and ENQ characters to pass through to the terminal. Alternatively, the ***SET CONSOLE PCHAR*** command may be used to redefine the set of printable characters. Modes ***UC*** and ***7P*** are compatible with terminal types 0-9.

File transfer using the Reflection terminal emulator requires an 8-bit data path. To achieve this, the session must use MPE terminal type 12 (this may be configured either during a system reload or by specifying the ***TERM=12*** parameter when logging on with ***:HELLO***), and the channel must be set to ***8B***, ***REMOTEACK***, and ***NOCAPSLOCK*** modes.

Each channel supports independent I/O logging to a file. Disabling logging also closes the log file.

A channel may be manually disconnected from its associated Telnet session with the ***SET ATCDn DISCONNECT*** command. Otherwise, the connection will remain open until disconnected either by the Telnet client or a ***DETACH ATCD*** command, unless the channel is controlled by the TCI. For a serial connection, the command will drop and then raise the Data Terminal Ready line; to disconnect the serial port, include the ***−C*** switch in the command.

TDI device configuration may be displayed with the following commands:

Command	*Action*
SHOW ATCD	Display the device and unit configuration
SHOW ATCD MODES	Display the connection modes
SHOW ATCD CONNECTIONS	Display the channel connections
SHOW ATCD STATISTICS	Display the channel I/O statistics
SHOW ATCD DEVNO	Display the device number assignment
SHOW ATCD INTMASK	Display the interrupt mask setting
SHOW ATCD INTPRI	Display the interrupt priority assignment

TDI unit configuration may be displayed with the following commands:

Command	*Action*
SHOW ATCD<n>	Display the selected channel configuration
SHOW ATCD<n> LOG	Display the selected channel logging status

In addition to the current configuration settings, a channel controlled by the TCI will be marked *data set* in the listing; a channel not controlled will be marked *direct*.

When debug output logging is enabled, tracing may be configured by specifying one or more of the reporting level options:

Option	*Reporting Level*
CSRW	Interface control, status, read, and write actions
SERV	Line unit service events
PSERV	Poll unit service events (periodic)
XFER	Data receptions and transmissions
IOBUS	I/O bus signals and data words received and returned

The ***CSRW*** option traces control, status, read, and write commands sent to the interface. The ***SERV*** option traces line unit event service scheduling and

entries. The line service is entered to transmit or receive each character at the configured baud rate. The ***PSERV*** option traces poll unit event service entries. The poll service is entered once every 10 milliseconds to poll the Telnet port for connections and input characters to be received by the active channels. The ***XFER*** option traces the characters received and transmitted by the active channels. The ***IOBUS*** option traces the I/O backplane signals and data received and returned via the interface.

Examples of the trace formats follow:

```
>>ATCD  cmds: Status is DIO OK | complete | send
>>ATCD  serv: Channel 0 delay 1708 service scheduled
>>ATCD pserv: Poll service entered
>>ATCD  xfer: Channel 0 character 'G' sent
>>ATCD iobus: Received data 000000 with signals DREADSTB
```

The Terminal Data Interface state contains these registers:

Name	***Size***	***Radix***	***Symbolic***	***Action***
CNTL	16	8		Control register
STAT	16	8		Status register
READ	16	8	Y	Read register
WRITE	16	8	Y	Write register
FLAG	1	2		Data flag flip-flop
MASK	1	2		Interrupt mask flip-flop
FTIME	24	10		Fast receive/send time
RSTAT [**0:20**]	16	8		Receive channel status, channels 0-20
RPARM [**0:20**]	16	8		Receive channel parameters, channels 0-
RBUFR [**0:20**]	16	8	Y	Receive channel buffers, channels 0-2
SSTAT [**0:15**]	16	8		Send channel status, channels 0-15
SPARM [**0:15**]	16	8		Send channel parameters, channels 0-15
SBUFR [**0:15**]	16	8	Y	Send channel buffers, channels 0-15

The READ, WRITE, RBUFR, and SBUFR registers may be examined or deposited using any of the modes described in the Symbolic Display and Entry section above.

Terminal Control Interface

The TCI provides the Request to Send (CA) and Data Terminal Ready (CD) control lines, and the Data Set Ready (CC) and Carrier Detect (CF) status lines for each of sixteen terminals or data sets. The modem controls model the Bell 103A data set without ring detection.

Data Terminal Ready must be set to enable a channel to accept an incoming connection. When a channel connects, Data Set Ready and Carrier Detect will

be set. Configuring the terminal subtype to 1 in MPE will enable the TCI to establish these settings. In addition, aborting a session will drop Data Terminal Ready after the user is logged out, which will drop the Telnet connection. If the user drops the Telnet connection manually, the session will be logged out. These actions will not occur for channels configured for terminal subtype 0 in MPE, which does not use the TCI.

ATCC device options that may be specified are:

Option	***Action***
TERMINAL	Connect using Telnet or serial ports; default
DIAGNOSTIC	Connect using diagnostic test cables
DEVNO=<n>	Set the device number; default is 8
INTMASK=<n>	Set the interrupt mask; default is E
INTPRI=<n>	Set the interrupt priority; default is 8
DEBUG=<option>	Enable tracing
NODEBUG	Disable tracing; default
ENABLED	Enable the device; default
DISABLED	Disable the device

Enabling the diagnostic mode simulates the installation of eight HP 30062-60003 diagnostic test (loopback) cables between channels 0-1, 2-3, etc., as required by the multiplexer diagnostic. Each cable connects the Data Terminal Ready and Request to Send lines, respectively, of one channel to the Data Set Ready and Carrier Detect lines of the other channel, and vice versa. Enabling terminal mode allows the multiplexer to respond to incoming connections again.

There are no TCI units or unit commands.

TCI device configuration may be displayed with the following commands:

Command	***Action***
SHOW ATCC	Display the device configuration
SHOW ATCC MODE	Display the connection mode
SHOW ATCC DEVNO	Display the device number assignment
SHOW ATCC INTMASK	Display the interrupt mask setting
SHOW ATCC INTPRI	Display the interrupt priority assignment

When debug output logging is enabled, tracing may be configured by specifying one or more of the reporting level options:

Option	***Reporting Level***
CSRW	Interface control, status, read, and write actions
PSERV	Poll unit service events (periodic)

Option	*Reporting Level*
XFER	Control and status line changes
IOBUS	I/O bus signals and data words received and returned

The ***CSRW*** option traces control, status, read, and write commands sent to the interface. The ***PSERV*** option traces poll unit event service entries. The poll service is entered once every 10 milliseconds to poll the channels for changes in the control or status lines. The ***XFER*** option traces the changes in control and status lines by the active channels. The ***IOBUS*** option traces the I/O backplane signals and data received and returned via the interface.

Examples of the trace formats follow:

```
>>ATCC csrw: Channel 2 control is C2 | C1 | ~S2 | ~S1
>>ATCC pserv: Poll service entered
>>ATCC xfer: Channel 2 line status is RTS | DTR
>>ATCC iobus: Received data 031374 with signals DCONTSTB
```

The Terminal Control Interface state contains these registers:

Name	*Size*	*Radix*	*Description*
CNTL	16	8	Control register
STAT	16	8	Status register
CNTR	16	10	Channel counter
SCAN	1	2	Scan flip-flop
MASK	1	2	Interrupt mask flip-flop
C2 [0:15]	1	2	Control line 2, channels 0-15
C1 [0:15]	1	2	Control line 1, channels 0-15
S2 [0:15]	1	2	Status line 2, channels 0-15
S1 [0:15]	1	2	Status line 1, channels 0-15
ES2 [0:15]	1	2	Enable status line 2 interrupt, channels 0-1
ES1 [0:15]	1	2	Enable status line 1 interrupt, channels 0-1
MS2 [0:15]	1	2	Status line 2 match, channels 0-15
MS1 [0:15]	1	2	Status line 1 match, channels 0-15

30033A Selector Channel Maintenance Board

The HP 30033A Selector Channel Maintenance Board provides the circuitry necessary to test the I/O bus signals driven and received by the Selector and Multiplexer Channels. Used with the Stand-Alone Selector Channel and Multiplexer Channel diagnostics, the SCMB is used to verify that the correct bus signals are driven in response to each of the programmed I/O orders, and that the channel responds correctly to the signals returned to it. The SCMB functions as a programmable interface that can log incoming signals and drive the outgoing

signals, as well as simulate a number of interface hardware faults. Two SCMBs are provided; they are named *SCMB1* and *SCMB2*.

Both SCMBs are normally disabled, as they are used only with the standalone diagnostics. The CPU and Selector Channel diagnostics both use one SCMB. If SCMB1 is used, it may be referred to as *SCMB* if desired. The Multiplexer Channel diagnostic uses both SCMBs.

Device options that may be specified for the two SCMBs are:

Option	***Action***
SC	Connect the SCMB to the Selector Channel bus
MX	Connect the SCMB to the Multiplexer Channel bus; default
DEVNO=<n>	Set the device number; defaults are 65 and 66
INTPRI=<n>	Set the interrupt priority; defaults are 10 and 11
SRNO=<n>	Set the service request number; defaults are 0 and 1
DEBUG=<option>	Enable tracing
NODEBUG	Disable tracing; default
ENABLED	Enable device
DISABLED	Disable device; default

The ***SC*** option installs the SCMB on the Selector Channel bus and configures operation for the Selector Channel diagnostic. The ***MX*** option installs the SCMB on the Multiplexer Channel bus and configures operation for the Multiplexer Channel and CPU diagnostics. The ***SRNO*** value may be changed only when the SCMB is connected to the Multiplexer Channel bus.

When debug output logging is enabled, tracing may be configured by specifying one or more of the reporting level options:

Option	***Reporting Level***
CSRW	Interface control, status, read, and write actions
XFER	Data read and write transfers
SERV	SR delay service events
IOBUS	I/O bus signals and data words received and returned

The ***CSRW*** option traces control, status, read, and write commands sent to the interface. The ***XFER*** option traces data words transferred to and from the interface by the channel. The ***SERV*** option traces event service scheduling and entries used to schedule delays in the service requests to the channel. The ***IOBUS*** option traces the I/O backplane signals and data received and returned via the interface.

Examples of the trace formats follow:

```
>>SCMB1 csrw: Control is device end | load control IOAW | count nothing
>>SCMB1 xfer: Counter/buffer value 022222 read
>>SCMB1 serv: Delay 2 SR service scheduled
>>SCMB1 iobus: Received data 000000 with signals ACKSR | PCONTSTB | CHANSO
```

The SCMB state contains these registers:

Name	*Size*	*Radix*	*Description*
CNTL	16	8	Control register
STAT	16	8	Status register
CNTR	16	8	Counter/buffer register
SIOBSY	1	2	SIO is active
CHANSR	1	2	Channel service request is active
DEVSR	1	2	Device service request is active
INXFR	1	2	Input transfer is active
OUTXFR	1	2	Output transfer is active
JMPMET	1	2	Jump condition is met
XFRERR	1	2	Transfer error condition is present
EOT	1	2	End of transfer condition is presen
TRMCNT	1	2	Terminal count condition is present
MISCMP	1	2	Miscompare condition is present
DEVEND	1	2	Device end condition is present
STOP	1	2	Transfer has been stopped

30135A System Clock

The HP 30135A System Clock is used with Series II and III systems and provides a programmable interval clock employed as the MPE time-of-day and process-switching clock. The clock provides programmable periods of 10 microseconds to 10 seconds in decade increments. Each "tick" of the clock increments a presettable counter that may be compared to a selected limit value. The clock may request an interrupt when the values are equal, and a status indication is provided if the counter reaches the limit a second time without acknowledgement.

CLK device options that may be specified are:

Option	*Action*
CALTIME	Use calibrated timing; default
REALTIME	Use realistic timing
DEVNO=<n>	Set the device number; default is 3
INTPRI=<n>	Set the interrupt priority; default is 1
DEBUG=<option>	Enable tracing
NODEBUG	Disable tracing; default
ENABLED	Enable the device; default
DISABLED	Disable the device

Calibrated timing aligns the simulated clock periods with the clock on the host system. When calibrated, each of the programmable periods will elapse after the corresponding amount of host-system time.

When realistic timing is enabled, the simulation models the programmable periods in terms of machine instructions executed. Calibrated timing is required to enable idling. Realistic timing is necessary to pass the hardware diagnostics.

When debug output logging is enabled, tracing may be configured by specifying one or more of the reporting level options:

Option	***Reporting Level***
CSRW	Interface control, status, read, and write actions
PSERV	Clock unit service events (periodic)
IOBUS	I/O bus signals and data words received and returned

The ***CSRW*** option traces control, status, read, and write commands sent to the interface. The ***PSERV*** option traces event service scheduling and entries, which occur at a periodic rate dependent on the programmable configuration. The ***IOBUS*** option traces the I/O backplane signals and data received and returned via the interface.

Examples of the trace formats follow:

```
>>CLK  csrw: Control is load rate | select limit | 1 millisecond rate
>>CLK  csrw: Status is DIO OK | LR = CR | limit selected | 1 millisecond rate
>>CLK pserv: Service entered with counter 0 increment 1 limit 1
>>CLK pserv: Rate 1 millisecond delay 389 service rescheduled
>>CLK iobus: Received data 000000 with signals DSETINT
>>CLK iobus: Returned data 000000 with signals INTREQ
```

The System Clock state contains these registers:

Name	***Size***	***Radix***	***Description***
CNTL	16	8	Control Register
STAT	16	8	Status Register
COUNT	16	8	Count Register
LIMIT	16	8	Limit Register
RATE	3	8	Clock Rate Register
SYSIRQ	1	2	System Interrupt Request flip-flop
LIMIRQ	1	2	Count = Limit Interrupt Request flip-flop
OVFIRQ	1	2	Count = Limit Overflow Interrupt Request flip-flop

Selector Channel I/O Device Simulations

The Selector Channel controls these I/O device interfaces:

- 30229B Disc Interface

30229B Disc Interface with Eight 7905/7906/7920/7925 Drives

The HP 30129A Cartridge Disc Subsystem connects the 7905A, 7906A, 7920A, and 7925A disc drives to the HP 3000. The subsystem consists of a 30229B Cartridge Disc Interface, a 13037D Multiple-Access Disc Controller ("MAC"), and from one to eight MAC drives. The subsystem uses the Selector Channel to achieve a 937.5 KB/second transfer rate to the CPU.

The disc controller connects from one to eight HP 7905 (15 MB), 7906 (20 MB), 7920 (50 MB), or 7925 (120 MB) disc drives to interfaces installed in from one to eight CPUs. The drives use a common command set and present data to the controller synchronously at a 468.75 kiloword per second (2.133 microseconds per word) data rate.

The disc interface is used to connect the HP 3000 CPU to the 13037's device controller. While the controller supports multiple-CPU systems, the HP 3000 does not use this capability.

The simulation provides up to eight disc drives. Drive types may be intermixed; 7905s are selected by default. Attaching a disc image file to a unit simulates inserting a disc pack into a drive:

```
ATTACH {-R} DSn <image-filename>
```

Adding the ***–R*** (read-only) switch is equivalent to setting the drive's Disc Protect switch on.

If the host operating system returns an error when seeking, reading, or writing a disc image file, the simulator will report the error to the simulation console, e.g.:

```
HP 3000 simulator disc library I/O error: No space left on device
```

A simulated disc seek will fail with Status 2 Error (Drive Fault) status, and the drive's heads will unload. Reloading the heads will clear the drive fault. A simulated disc read or write will fail with Uncorrectable Data Error status. The target operating system will then react to this error as though a real drive had encountered a bad disc sector.

A drive's unit number is not set explicitly. Instead, the drive unit number is derived from the simulation unit number. For example, unit DS0 responds to disc unit select number 0. Changing the unit select switch on a mounted drive is equivalent to detaching and reattaching the disc image file to the corresponding simulation unit.

Device and unit options include configuring the timing, drive type, protection and format status, and the ability to set drives ready or not-ready. The command forms are:

```
SET DS <device-option>
SET DSn <unit-option>
```

Device Options

Device options that may be specified are:

Option	*Action*
FASTTIME	Use optimized timing; default
REALTIME	Use realistic timing
DIAGNOSTIC	Reset the diagnostic override table
DIAGNOSTIC=<params>	Add an entry to the diagnostic override table
NODIAGNOSTIC	Clear the diagnostic override table; default
DEVNO=<n>	Set the device number; default is 4
INTMASK=<n>	Set the interrupt mask; default is E
INTPRI=<n>	Set the interrupt priority; default is 4
DEBUG=<option>	Enable tracing
NODEBUG	Disable tracing; default
ENABLED	Enable the device; default
DISABLED	Disable the device

Device configuration may be displayed with the following commands:

Command	*Action*
SHOW DS	Display the device and unit configuration
SHOW DS DIAGNOSTIC	Display the diagnostic override table
SHOW DS TIMING	Display the timing mode

When realistic timing is enabled, the simulation accurately models the disc movement times (in machine instructions). For example, seeking takes longer if the positioner movement distance is farther. Optimized timing reduces the timing to the minimum necessary to operate correctly; this is much faster than a real disc drive would operate.

The delay times used by the simulation in ***FASTTIME*** mode may be set via the register interface. The values may be adjusted as necessary to work around any HP 3000 software problems that are triggered by the abnormally rapid disc operation. Resetting the device with the ***RESET –P*** (power-on reset) command restores the original optimized times.

The ***SET DS DIAGNOSTIC*** and ***SET DS NODIAGNOSTIC*** commands provide diagnostic support. See the Diagnostic Support section below for details.

Unit Options

Unit options that may be specified for individual disc drives are:

Option	***Action***
7905	Set the drive type to 7905; default
7906	Set the drive type to 7906
7920	Set the drive type to 7920
7925	Set the drive type to 7925
UNLOAD	Set the drive's Run/Stop switch to *Stop*; default when detached
LOAD	Set the drive's Run/Stop switch to *Run*; default when attached
PROTECT	Set the 7920's or 7925's Read Only switch to *On*
PROTECT=UPPER	Set the 7905's or 7906's Protect Upper Disc switch to *On*
PROTECT=LOWER	Set the 7905's or 7906's Protect Lower Disc switch to *On*
UNPROTECT	Set the 7920's or 7925's Read Only switch to *Off*
UNPROTECT=UPPER	Set the 7905's or 7906's Protect Upper Disc switch to *Off*
UNPROTECT=LOWER	Set the 7905's or 7906's Protect Lower Disc switch to *Off*
FORMAT	Set the drive's Format switch to *Enabled*
NOFORMAT	Set the drive's Format switch to *Disabled*; default
ENABLED	Enable the unit; default
DISABLED	Disable the unit

The ***UNOAD*** and ***LOAD*** options unload and load the drive's heads from the disc pack, setting the drive not-ready and ready, respectively. The former provides a convenient method of setting a drive "down" without detaching the associated disc image file.

PROTECT=UPPER and ***PROTECT=LOWER*** protect the upper and lower platters, respectively, of 7905 and 7906 drives from writing; ***PROTECT*** without a value protects both platters. For the 7920 and 7925 drives, ***PROTECT*** protects the entire drive. The ***UNPROTECT*** options remove drive protection and enable writing.

The ***FORMAT*** option enables certain controller commands, such as Initialize, that alter the sector address fields. Typically, this option is needed when reloading the operating system from tape to disc. The ***NOFORMAT*** option inhibits these commands and permits only the standard Write command, subject to the appropriate drive protection status.

Drive configuration may be displayed with the following command:

Command	*Action*
SHOW DS<n>	Display the selected drive's configuration

Diagnostic Support

The offline disc diagnostic (program D419A) tests features of the disc that are not supported in simulation, e.g., the detection and correction of sector data errors. However, the simulation does support recovery from host file system errors, which are mapped to uncorrectable data errors. Because file system errors cannot be generated intentionally, a table of diagnostic overrides is provided to return specified error status values instead of otherwise normal completion status. This allows the error recovery simulation code to be tested.

The initially empty diagnostic override table may be populated with one or more entries using this command:

```
SET DS DIAGNOSTIC=<cylinder>;<head>;<sector>;<opcode>;<SPD>;<status>
```

. . . where:

- ***cylinder*** is a decimal value from 0 to 822.
- ***head*** is a decimal value from 0 to 8.
- ***sector*** is a decimal value from 0 to 63.
- ***opcode*** is an octal value from 0 to 26.
- ***SPD*** is any combination of the letters S, P, and D.
- ***status*** is an octal value from 0 to 37.

If a command specifies the opcode value as 15 (Request Syndrome) and the status value as 17 (Correctable Data Error), then four additional values must be appended to the command line above:

```
;<displacement>;<syndrome 1>;<syndrome 2>;<syndrome 3>
```

. . . where:

- ***displacement*** is a decimal value from –5 to 135.
- ***syndrome 1-3*** are octal values from 0 to 177777.

When the table is populated, the first entry becomes the initial "current" entry. As the diagnostic program issues each disc controller command, the cylinder, head, sector, and command opcode values from the current override entry are checked against the corresponding current controller values. If the values match, then the controller status and Spare/Protected/Defective values are set from the entry rather than being generated by the command, and the next entry becomes the current entry. These values will then be returned to the program as the completion status of the current command. This process continues until

the table is exhausted or the program ends. The current entry may be reset to the first entry by specifying the ***SET DS DIAGNOSTIC*** command with no parameters. Entering the ***SET DS NODIAGNOSTIC*** command will clear the table.

If the disc controller command performs address verification, then any SPD/status value(s) provided will be used as the result of the verification. In particular, setting the P bit for a Write command will cause a Protected Track error if the drive's Format switch is set to *Disabled*, and any status value other than Normal Completion, Correctable Data Error, or Uncorrectable Data Error will cause a verification abort.

In hardware, the errors that may occur during verification are Cylinder Miscompare, Head-Sector Miscompare, Sync Timeout, Illegal Spare Access, and Defective Track; any of these errors may be simulated. In addition, an Uncorrectable Data Error will occur in hardware if the controller is unable to verify any of the 16 sectors starting at the sector preceding the target sector, but this error cannot be simulated. Instead, specifying either a Correctable Data Error or an Uncorrectable Data Error will cause an abort at the end of the first sector of a Read, Write, or Verify command.

Correctable Data Error and Uncorrectable Data Error may also be specified for the Request Syndrome command. The former requires the values to be returned for the displacement and three syndrome words; the latter does not, as the values are meaningless if the error cannot be corrected.

BOOT Command

The interface supports the ***BOOT*** command as an alternative to ***LOAD***. The ***BOOT DS0*** command is equivalent in hardware to setting the SWCH register to configure the control byte and device number, and pressing the LOAD and ENABLE buttons to begin the cold load process for unit 0. The SWCH register is set as follows:

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8 — 15***
0	0	0	0	0	0	0	0	DS device number

The control byte defaults to the Cold Load Read disc command.

Tracing and Registers

When debug output logging is enabled, tracing may be configured by specifying one or more of the reporting level options:

Option	*Reporting Level*
CMD	Controller commands executed
INCO	Controller command initiations and completions
CSRW	Interface control, status, read, and write actions
STATE	Controller command state changes executed
SERV	Disc unit service events
XFER	Data reads and writes
IOBUS	I/O bus signals and data words received and returned

The ***CMD*** option traces the commands executed by the disc controller. The ***INCO*** option traces the beginning and ending of commands, including command termination status. The ***CSRW*** option traces control, status, read, and write commands sent to the interface. The ***STATE*** option traces the entries into each of the internal controller execution states. The ***SERV*** option traces disc unit event service scheduling and entries. The ***XFER*** option traces the data words read from or written to the disc. The ***IOBUS*** option traces the signals and data received and returned via the I/O backplane and interface and via the data, flag, and function buses between the interface and controller.

Examples of the trace formats follow:

```
>>DS   cmd: Unit 0 Seek to cylinder 410 head 2 sector 47
>>DS  inco: Unit 0 Seek command completed with Normal Completion status
>>DS  csrw: Control is 004000 (Write)
>>DS state: Unit 0 Write data phase entered from service
>>DS  serv: Unit 0 Write data phase delay 1 service scheduled
>>DS  xfer: Unit 0 Write word 3400 is 177777
>>DS iobus: Received data 005000 with signals PCONTSTB
>>DS iobus: Controller (idle) received data 005000 with flags CMRDY | EOD
```

The Disc Interface state contains these registers:

Name	*Size*	*Radix*	*Read-Only*	*Description*
SIOBSY	1	2		SIO is active
DEVSR	1	2		Device service request is active
INXFR	1	2		Input transfer is active
OUTXFR	1	2		Output transfer is active
INTMSK	1	2		Interrupt mask is enabled
JMPMET	1	2		Jump condition is met
DEVEND	1	2		Device end is active
DATOVR	1	2		Data overrun has occurred
ENDDAT	1	2		End of data has occurred
TEST	1	2		Test mode is active
WAIT	1	2		Data wait is active
CLEAR	1	2		A hardware clear is active

Name	*Size*	*Radix*	*Read-Only*	*Description*
CMRDY	1	2		A command word is ready
DTRDY	1	2		A data word is ready
EOD	1	2		The last data word has been transferred
INTOK	1	2		An interrupt is allowed
OVRUN	1	2		The current transfer has overrun
XFRNG	1	2		The current transfer is no good
BUFFER	16	8		Data buffer register
STATUS	16	8		Status register
RETRY	4	10		Retry counter
OPCODE	5	8	Y	Controller operation code
CSTATS	5	8	Y	Controller status
CSTATE	2	10	Y	Controller state
EOC	1	2		End of cylinder has been seen
VERIFY	1	2		Address verification is active
SPDU	16	8		Spare/Protected/Defective flags and unit number
FLMASK	4	8		File mask
CYL	16	10		Controller cylinder
HEAD	6	10		Controller head
SECTOR	8	10		Controller sector
COUNT	16	10		Word or sector counter
SECBUF [0:137]	16	8		Sector data buffer
INDEX	8	10		Sector buffer current index
LENGTH	8	10		Sector buffer valid length
TTIME	24	10		Fast one-track seek time
FTIME	24	10		Fast full-stroke seek time
STIME	24	10		Fast one-sector rotation time
XTIME	24	10		Fast one-word transfer time
GTIME	24	10		Fast intersector gap rotation time
OTIME	24	10		Fast controller overhead time
UCYL [0:8]	10	10		Current cylinder, drives 0-7 and controller
UOPCODE [0:8]	6	8	Y	Current operation code, drives 0-7 and contro
USTATUS [0:8]	32	2		Unit status, drives 0-7 and controller
USTATE [0:8]	4	10	Y	Current command state, drives 0-7 and controller
UPOS [0:8]	32	10	Y	Current byte position, drives 0-7 and controller
UWAIT [0:8]	32	8		Scheduled wait delay, drives 0-7 and controller

The BUFFER and SECBUF registers may be examined or deposited using any of the modes described in the Symbolic Display and Entry section above.

Multiplexer Channel I/O Device Simulations

The Multiplexer Channel controls these I/O device interfaces:

- 30209A Line Printer Controller
- 30215A Tape Controller

30209A Line Printer Controller with One 2607/13/17/18 Line Printer

The HP 30118A, 30127A, 30128A, and 30133A Line Printer Subsystems connect the 2607A, 2613A, 2618A, and 2617A printers, respectively, to the HP 3000. Each subsystem consists of a 30209A Line Printer Controller, employing a 30051A Universal Interface (Differential) and an interconnecting cable, and an HP 2607A (200 lines per minute), HP 2613 (300 lpm), HP 2617 (600 lpm), or HP 2618 (1250 lpm) line printer. These subsystems employ the Multiplexer Channel to achieve a 360 KB/second transfer rate from the CPU.

The simulation provides one printer unit. The 2617 is selected by default. Attaching a text file to the unit simulates loading paper into the printer:

```
ATTACH {-N} LP <image-filename>
```

Adding the ***–N*** (new file) switch clears the contents of the image file if present. Without the ***–N*** switch, printer output will be appended to any preexisting image file content.

Printer output written to the image file is typically buffered by the host operating system's underlying stream I/O routines. While it is running, the simulator flushes the file after each printer top-of-form request to permit convenient inspection of the image file. Stopping the simulator also flushes the file.

If the host operating system returns an error when writing to the printer image file, the simulator will report the error to the simulation console, e.g.:

```
HP 3000 simulator printer I/O error: No space left on device
```

The printer goes offline with an alarm condition, and the simulator stops. Simulation may then be resumed, either with the printer set back online if the problem is fixed, or with the printer remaining offline if the problem is uncorrectable.

Detaching the text file from the unit with the ***DETACH LP*** command simulates running out of paper. If the command is entered while there are characters in the print buffer or, for the 2607 only, the print location is not at the top of the form, **Command not completed** is displayed on the simulation console, and the file remains attached until the required conditions are true. Once simulation is resumed and the print operations complete, the printer is set offline and detached automatically.

Detaching may also be forced with the ***DETACH –F LP*** command. This simulates physically removing the paper and takes effect immediately, regardless of any printing operations in progress.

Device and unit options include configuring the printer type and timing, output format, vertical format unit (VFU), and the ability to set the printer offline or online. The command forms are:

```
SET LP <device-option>
SET LP <unit-option>
```

Device Options

Device options that may be specified are:

Option	***Action***
FASTTIME	Use optimized timing; default
REALTIME	Use realistic timing
PRINTER	Connect using the printer interface cable; default
DIAGNOSTIC	Connect using the Diagnostic Hardware Assembly
DEVNO=<n>	Set the device number; default is 14
INTMASK=<n>	Set the interrupt mask; default is E
INTPRI=<n>	Set the interrupt priority; default is 18
SRNO=<n>	Set the service request number; default is 11
DEBUG=<option>	Enable tracing
NODEBUG	Disable tracing; default
ENABLED	Enable the device; default
DISABLED	Disable the device

The printer supports realistic and optimized timing modes. Realistic timing attempts to model the print buffer load and print-and-space operation delays inherent in the physical hardware. For example, output of lines with more characters takes longer than output of lines with fewer characters, and spacing six lines takes approximately six times longer than spacing one line.

Optimized timing reduces operation delays to the minimums necessary to operate correctly; this is much faster than a real line printer would operate.

The delays used by the simulation in ***FASTTIME*** mode may be set via the register interface. The values may be adjusted as necessary to work around any HP 3000 software problems that are triggered by the unusually rapid print operations. Resetting the device with the ***RESET −P*** (power-on reset) command restores the original optimized times.

The ***DIAGNOSTIC*** option simulates the installation of the HP 30049C Diagnostic Hardware Assembly in place of the printer interface cable. This is needed to run the offline universal interface diagnostic (program D435A). The CNLED register provides the state of the *CONT 6* through *CONT 10* LEDs, and the J2WX, DATOUT, DCOUT, DFIN, and DENDIN registers provide the logic levels of the corresponding test pins on the DHA.

Setting the ***PRINTER*** option reconnects the cable between the line printer and the interface.

Device configuration may be displayed with the following commands:

Command	*Action*
SHOW LP	Display the device and unit configuration
SHOW LP MODES	Display the timing and connection modes

Unit Options

Unit options that may be specified are:

Option	*Action*
2607	Set the printer model to 2607
2613	Set the printer model to 2613
2617	Set the printer model to 2617; default
2618	Set the printer model to 2618
VFU	Install the HP standard 66-line VFU tape; default
VFU=<filename>	Install a custom VFU tape image
OFFLINE	Set the printer offline; default when detached
ONLINE	Set the printer online; default when attached
EXPAND	Write expanded output to the printer image file; default
COMPACT	Write compact output to the printer image file

The ***2607***, ***2613***, ***2617***, and ***2618*** options select the printer model. Each printer is configured with Option 001, which provides a 128-character set for the HP 2607 and 96-character sets for the HP 2613, 2617, and 2618. The 2607 and 2618 support 132-character print line lengths, while the 2613 and 2617 support 136-character lines. Exceeding the line length on the 2607 causes an automatic print-and-space operation. Exceeding the line length on the 2613, 2617, or 2618 discards the excess characters.

The ***VFU*** option configures the printer's vertical format unit, as described in the Vertical Format Unit section below.

The ***OFFLINE*** and ***ONLINE*** options place the printer offline and online, respectively. The former provides a convenient method of setting the printer "down" without detaching the associated output file.

The printer will not go offline if there are characters in the print buffer. Instead, the offline condition is held off until the line is printed and paper movement is complete. The ***SET LP OFFLINE*** command checks for data in the print buffer or a print operation in progress. If either condition is true, the action is deferred, and **Command not completed** is displayed on the simulation

console. A ***SHOW LP*** command will show that the device is still online. Once simulation is resumed and the print operation completes, the printer is set offline. No console message reports this, although a subsequent ***SHOW LP*** command will indicate the new status.

Entering a ***SET LP ONLINE*** command while an offline or detach action is deferred will cancel the action without triggering any programmed online-to-offline or offline-to-online status transition interrupt. A ***RESET LP*** command also cancels any deferred offline or detach action. Additionally, it clears the print buffer and terminates any print action in progress, so a ***SET LP OFFLINE*** or ***DETACH LP*** will succeed if issued afterward.

The ***EXPAND*** and ***COMPACT*** options control the format of lines written to the printer image file. In compact mode, a carriage-return/line-feed character pair terminates a printed line, but subsequent line spacing is performed by line-feed characters alone. A top-of-form request will emit a form-feed character instead of the number of line-feeds required to reach the top of the next form. This mode is suitable for sending the printer output to a physical printer connected to the host.

In expanded mode, paper advance is handled by emitting the correct number of carriage-return/line-feed pairs. This mode is suitable for retaining printer output as a text file.

The HP 2613, 2617, and 2618 printers are capable of overprinting. If the printer is sent a format code to suppress spacing, the characters in the buffer are printed and the buffer is emptied, but the paper is not advanced. The next print operation will print its characters over those already present on the paper. In simulation, overprinting is performed in one of two ways, depending on the print mode.

In compact mode, overprinting is simulated by emitting a carriage-return character at the end of the line. In expanded mode, overprinting is simulated by merging characters in the buffer before writing them as a single line in the printer image file. As the second and subsequent lines are output, each new character is compared with its corresponding character in the buffer. If the character in the buffer is a space, the new character replaces it. If the new character is a space or is the same as the character in the buffer, the character in the buffer is retained. Otherwise, the character in the buffer is replaced with a special character representing an overprinted combination. This character defaults to DEL (octal 177) but may be changed by altering the OVPCHR register value.

The HP 2607 printer cannot overprint. A request to suppress spacing will result in a single-line paper advance after printing.

Vertical Format Unit

The HP 2607 supports an 8-channel vertical format unit (VFU), and the HP 2613, 2617, and 2618 printers support 12-channel VFUs. A continuous punched

paper tape that controls paper spacing is installed in the VFU reader. The printer may be commanded to advance the paper until a punched hole is detected in a specified VFU channel. The length of the tape establishes the length of the forms loaded into the printer.

Initially, the standard VFU tape (part number 1535-2655 for the HP 2607 or 02613-80001 for the HP 2613, 2617, and 2618) is installed. This tape associates channels 1-8 with the following printer actions:

Channel	***Printer Advances to***
1	Top of form
2	Bottom of form
3	Next single space
4	Next double space
5	Next triple space
6	Next half-page
7	Next quarter-page
8	Next sixth-page

For the 02613-80001 tape, channel 9 is punched the same as channel 2, and channels 10-12 are uncommitted.

The simulator supports the use of custom VFU tape images. A custom tape may be installed in place of the standard tape by issuing the ***SET LP VFU=<filename>*** command. Custom tapes must reserve channel 1 for the top-of-form location, but the other channels may be defined as desired.

A custom tape image is a plain-text file that starts with a VFU definition line and then contains one channel-definition line for each line of the form. The number of channel-definition lines establishes the form length.

A semicolon appearing anywhere on a line begins a comment, and the semicolon and all following characters are ignored. Zero-length lines, including lines beginning with a semicolon, are ignored. Note that a line containing one or more blanks is not a zero-length line, so, for example, the line " *; a comment starting in column 2*" is not ignored.

The first (non-ignored) line in the file must be a VFU definition line of this exact form:

```
VFU=<punch characters>,<no-punch character>{,<title>}
```

. . . where:

- ***punch characters*** is a set of one or more characters used interchangeably to represent a punched location.
- ***no-punch character*** is a single character representing a non-punched location.

- ***title*** is an optional description that is printed by the ***SHOW LP VFU*** command; the description "Custom VFU" is used if the title is omitted.

If the ***VFU*** line is missing or not of the correct form, then **Format error** is displayed on the simulation console, and the VFU tape is not changed.

The remaining lines define the channels punched for each line of the printed form. The line format consists of a sequence of punch, no-punch, and "other" characters, in channel order. Each punch or no-punch character defines a channel state, starting with channel 1 and proceeding left-to-right until all channels for the VFU are defined; any the extra channel states on the line are ignored. If the line terminates before all channels are defined, the remaining channels are set to the no-punch state. Any "other" characters, i.e., neither punch characters nor no-punch characters, are ignored and may be used freely to delineate the tape channels.

For a standard 66-line form, the first printable line (form line 1) is paper line 4, and the last printable line (form line 60) is paper line 63. The first channel-definition line in the file specifies form line 1, i.e., the top of the form, and must have a punch in channel 1. Channel line 60 defines form line 60, i.e., the bottom of the form. Six more channel lines follow: three for the bottom margin, and three for the top margin of the next form.

An example custom VFU file using a 66-line tape definition for an 8-channel VFU is:

```
; the VFU definition
;
; the set of punch characters is "1" and "X"
; the no-punch character is "0"
; all other characters are ignored

VFU=1X,0,A binary tape image

; the channel definitions

1 0 1 1 1 1 1 1 ; top of form (line 1; must have a punch in channel 1)
0-0-X-0-0-0-0-0 ; single space (line 2)
0011 ; channels 5-8 default to no-punch (line 3)
[...]
0 1 1 0 0 0 0 0 ; bottom of form (line 60)
0 0 0 0 0 0 0 0 ; bottom form margin (line 61)
0 0 0 0 0 0 0 0 ; bottom form margin (line 62)
0 0 0 0 0 0 0 0 ; bottom form margin (line 63)
0 0 0 0 0 0 0 0 ; top of form margin (line 64)
0 0 0 0 0 0 0 0 ; top of form margin (line 65)
0 0 0 0 0 0 0 0 ; top of form margin (line 66)
```

If a custom tape has been used, the standard tape may be reinstalled by issuing

the ***SET LP VFU*** command.

Attempting to command an advance to a channel that is not punched will cause a tape fault, and the printer will go offline. Setting the printer back online will clear the fault.

The current VFU definition may be displayed with the following command:

Command	*Action*
SHOW LP VFU	Display the currently loaded VFU tape definition

This command displays the current VFU tape title and then the channel definitions for each form line. By default, a punched channel is indicated by an "O" character, and an unpunched channel is indicated by a period (".") These characters may be changed by depositing new values into the PUNCHR and UNPCHR registers, respectively.

Tracing and Registers

When debug output logging is enabled, tracing may be configured by specifying one or more of the reporting level options:

Option	*Reporting Level*
CMD	Printer commands executed
CSRW	Interface control, status, read, and write actions
SERV	Printer, pulse, and transfer timer unit service events
XFER	Data transmissions
STATE	Device handshake state changes executed
IOBUS	I/O bus signals and data words received and returned

The ***CMD*** option traces the commands executed by the printer. The ***CSRW*** option traces control, status, read, and write commands sent to the interface. The ***SERV*** option traces printer and interface unit event service entries. The ***XFER*** option traces the data words and format commands written to the printer. The ***STATE*** option traces the scheduling of and entries into each of the internal device handshake execution states. The ***IOBUS*** option traces the signals and data received and returned via the I/O backplane.

Examples of the trace formats follow:

```
>>LP   cmd: Printed 132 characters on line 8
>>LP   cmd: Printer commanded to slew to VFU channel 1 from line 9
>>LP   cmd: Printer advanced 58 lines to line 1
>>LP  csrw: Control is character | interrupt status | word xfer
>>LP  csrw: Status is DIO OK | interrupt | interrupt status | system
```

```
>>LP  serv: Transfer delay 1491 service scheduled
>>LP  serv: Device Command 1 state printer service entered
>>LP  xfer: Character 'M' sent to printer
>>LP  xfer: Format code 001 sent to printer
>>LP state: Sequencer transitioned from Idle state to Device Flag 1 state
>>LP iobus: Received data 046515 with signals ACKSR | PWRITESTB | CHANSO
>>LP iobus: Returned data 060004 with signals INTREQ | JMPMET
```

The Line Printer Controller state contains these registers:

Name	***Size***	***Radix***	***Symbolic***	***Read-Only***	***Description***
SIOBSY	1	2			SIO is active
CHANSR	1	2			Channel service request is activ
DEVSR	1	2			Device service request is active
INXFR	1	2			Input transfer is active
OUTXFR	1	2			Output transfer is active
RDXFR	1	2			Read transfer is in progress
WRXFR	1	2			Write transfer is in progress
INTMSK	1	2			Interrupt mask is active
DEVCMD	1	2			Device command
DEVFLG	1	2			Device flag
DEVEND	1	2			Device end is active
SEQSTA	8	10			Sequencer state
CNTL	16	8			Control word
ISTAT	16	8			Interrupt status
DSTAT	16	8			Device status
READ	16	8	Y		Read word
WRITE	16	8	Y		Write word
J2WX	10	2			Jumper J2W10-J2W1 configuration
DATOUT	16	8	Y		Data out word
DATIN	16	8	Y		Data in word
DCOUT	1	2			Device command out
DFIN	1	2			Device flag in
DENDIN	1	2			Device end in
DIAGCN	16	8			DHA control word
CNLED	5	2			Control bit LEDs CONT 6-CONT 10
PFWARN	1	2			Power-fail warning exists
PFAULT	1	2			Paper fault exists
TFAULT	1	2			Tape fault exists
OLPEND	1	2			Offline transition is pending
PRLINE	8	10			Current print line number
BUFIDX	8	10			Current print buffer index
PRTBUF [**0:423**]	8	8	Y		Print buffer
OVPCHR	8	8	Y		Overprint character
FORMLN	8	10		Y	Form length in lines
VFU [**0:144**]	12	2		Y	Vertical format unit channels

Name	*Size*	*Radix*	*Symbolic*	*Read-Only*	*Description*
PUNCHR	8	8	Y		Punched channel character
UNPCHR	8	8	Y		Unpunched channel character
BTIME	24	10			Fast printer buffer load delay t
PTIME	24	10			Fast printing delay time
STIME	24	10			Fast paper slew per-line delay t
POS	32	10			Printer file current position

The PRTBUF, OVPCHR, PUNCHR, and UNPCHR registers default to single-character format display and entry but may be overridden with numeric-format switches, if desired. The READ, WRITE, DATOUT, and DATIN registers default to octal display but may be displayed in single-character format by specifying the ***−A*** switch. Symbolic entry for these registers is also supported.

The PFAULT and TFAULT registers indicate a paper fault and a tape fault, respectively, when they contain the value 1. Attaching the printer will clear a paper fault. Setting the printer online will clear a tape fault.

The character used to represent an overprinted position may be changed by depositing a new value into the OVPCHR register. For example, the ***DEPOSIT LP OVPCHR @**** *command will change the character from the default DEL (octal 177) to the commercial-at sign (octal 100). Similarly, the characters used to represent punched and unpunched VFU channels in the* **SHOW LP VFU*** display may be changed by depositing new values in the PUNCHR and UNPCHR registers, respectively.

The FORMLN register holds the current VFU form length, and the VFU register array holds the content of the current VFU tape. The register defaults to binary display, with channel 1 in the most-significant bit and channel 8 or 12 in the least-significant bit, depending on the VFU width. Elements 1-n correspond to VFU form lines 1-n, where a 1 value indicates a punch, and a 0 value indicates a no-punch. Element 0 is the logical OR of elements 1-n, so a 1 value indicates that the channel is punched somewhere on the tape, and a 0 value indicates that the channel is not punched.

30215A Tape Controller with Four 7970B/E Drives

The HP 30115A Magnetic Tape Subsystem connects the 7970B/E ½-inch magnetic tape drives to the HP 3000. The subsystem consists of a 30215A two-card tape controller processor and controller interface, and from one to four HP 7970B 800-bpi NRZI or HP 7970E 1600-bpi PE drives. The two drive types may be mixed on a single controller. The subsystem uses the Multiplexer Channel to achieve a 36 KB/second (NRZI) or 72 KB/second (PE) transfer rate to the CPU.

The simulation provides up to four tape drives. 7970Es are selected by default. Attaching a tape image file to a unit simulates mounting a tape reel on a drive:

```
ATTACH {-R} {-F} MSn {<format>} <image-filename>
```

Adding the ***−R*** (read-only) switch is equivalent to mounting the tape without a write ring in place. Adding the ***−F*** switch and a format identifier declares the tape image format to be used. Supported formats are ***SIMH***, ***E11***, ***TPC***, and ***P7B***. If the ***−F*** switch and a format identifier are not supplied, then SIMH tape image format is used. Note that erase gaps embedded in the tape image file are supported only in SIMH image format mode.

If the host operating system returns an error when reading or writing a tape image file, the simulator will report the error to the simulation console, e.g.:

```
HP 3000 simulator tape library I/O error: No space left on device
```

If this or another tape library error occurs, e.g., due to an illegal or damaged tape format, the simulator stops with an appropriate error message. Resuming the simulation will fail the tape operation with Tape Error status. The target operating system will then react to this error as though a real drive had encountered a bad tape block (CRC or MTE).

A drive's unit number is not set explicitly. Instead, the drive unit number is derived from the simulation unit number. For example, unit MS0 responds to tape unit select number 0. Pressing a different unit select button on a mounted drive is equivalent to detaching and reattaching the tape image file to the corresponding simulation unit. Pressing the OFF button is equivalent to setting the drive offline.

Device and unit options include configuring the drive type and timing, tape reel size, tape image format, and the ability to set drives offline or online. The command forms are:

```
SET MS <device-option>
SET MSn <unit-option>
```

Device Options

Device options that may be specified are:

Option	***Action***
FASTTIME	Use optimized timing; default
REALTIME	Use realistic timing
DEVNO=<n>	Set the device number; default is 6
INTMASK=<n>	Set the interrupt mask; default is E
INTPRI=<n>	Set the interrupt priority; default is 14
SRNO=<n>	Set the service request number; default is 3
DEBUG=<option>	Enable tracing
NODEBUG	Disable tracing; default
ENABLED	Enable the device; default
DISABLED	Disable the device

When realistic timing is enabled, the simulation accurately models the tape movement times (in machine instructions). For example, rewinding takes longer if the tape is positioned farther from the load point. Realistic timing is necessary to pass the magnetic tape diagnostic timing tests.

Optimized timing reduces the timing to the minimum necessary to operate correctly; this is much faster than a real tape drive would operate. In addition, enabling optimized timing also omits the initial erase gap normally written after the BOT marker. This does not affect gaps written with the Write Gap command.

The delay times used by the simulation in ***FASTTIME*** mode may be set via the register interface. The values may be adjusted as necessary to work around any HP 3000 software problems that are triggered by the unusually rapid tape operation. Resetting the device with the ***RESET −P*** (power-on reset) command restores the original optimized times.

Device configuration may be displayed with the following commands:

Command	*Action*
SHOW MS	Display the device and unit configuration
SHOW MS TIMING	Display the timing mode

Unit Options

Unit options that may be specified for individual tape drives are:

Option	*Action*
7970B	Set the drive type to 7970B
7970E	Set the drive type to 7970E; default
REEL=<n>	Set the reel size in feet; default is unlimited
CAPACITY=<n>	Set the reel capacity in megabytes; default is unlimited
OFFLINE	Set the unit offline; default when detached
ONLINE	Set the unit online; default when attached
FORMAT=<fmt>	Set the tape image format; default is SIMH format
ENABLED	Enable the unit; default
DISABLED	Disable the unit

The reel size may be set to 600-, 1200-, or 2400-foot capacity. Setting the capacity or reel size to 0 specifies unlimited capacity; in this configuration, the controller never returns an end-of-tape indication.

The ***OFFLINE*** and ***ONLINE*** options place a drive offline and online, respectively. The former provides a convenient method of setting a drive "down" without detaching the associated tape image file.

The tape image format for future ***ATTACH*** commands may be set to one of the format identifiers listed previously. The unit must be detached when the format is set.

Drive configuration may be displayed with the following commands:

Command	*Action*
SHOW MS<n>	Display the selected drive's configuration
SHOW MS<n> REEL	Display the selected drive's reel size or capacity
SHOW MS<n> CAPACITY	Display the selected drive's reel size or capacity
SHOW MS<n> FORMAT	Display the selected drive's tape image format

BOOT Command

The interface supports the ***BOOT*** command as an alternative to ***LOAD***. The ***BOOT MS0*** command is equivalent in hardware to setting the SWCH register to configure the control byte and device number, and pressing the LOAD and ENABLE buttons to begin the cold load process for unit 0. The SWCH register is set as follows:

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8 — 15*
0	0	0	0	0	1	1	0	MS device

The control byte defaults to the Read Record tape command.

Tracing and Registers

When debug output logging is enabled, tracing may be configured by specifying one or more of the reporting level options:

Option	*Reporting Level*
CMD	Controller commands executed
INCO	Controller command initiations and completions
CSRW	Interface control, status, read, and write actions
STATE	Controller command state changes executed
SERV	Tape unit service events
XFER	Data reads and writes
IOBUS	I/O bus signals and data words received and returned

The ***CMD*** option traces the commands executed by the tape controller. The ***INCO*** option traces the beginning and ending of commands, including the starting and ending tape position. The ***CSRW*** option traces control, status,

read, and write commands sent to the interface. The ***STATE*** option traces the scheduling of and entries into each of the internal controller execution states. The ***SERV*** option traces tape unit event service entries. The ***XFER*** option traces the data words read from or written to the tape. The ***IOBUS*** option traces the signals and data received and returned via the I/O backplane and interface and via the data, flag, and function buses between the interface and controller.

Examples of the trace formats follow:

```
>>MS   cmd: Unit 0 Write Record of 20-byte record succeeded
>>MS  inco: Unit 0 Write Record started at position 6120
>>MS  csrw: Control is 000005 (Write Gap)
>>MS  csrw: Status is ready | load point | 1600 bpi | no error | unit 0
>>MS state: Unit 0 Write Gap start phase delay 3794 service scheduled
>>MS  serv: Unit 0 service entered
>>MS  xfer: Unit 0 Write Record word 1 is 052525
>>MS iobus: Controller (idle) returned data 000000 with functions RQSRV
>>MS iobus: Received data 125252 with signals ACKSR | PWRITESTB | CHANSO
```

The Tape Controller state contains these registers:

Name	***Size***	***Radix***	***Read-Only***	***Description***
SIOBSY	1	2		SIO is active
CHANSR	1	2		Channel service request is active
DEVSR	1	2		Device service request is active
INXFR	1	2		Input transfer is active
OUTXFR	1	2		Output transfer is active
INTMSK	1	2		Interrupt mask is active
UINTRP	1	2		Unit interrupt is active
DEVEND	1	2		A device end has occurred
XFRERR	1	2		A transfer error has occurred
BUFWRD	16	8		Buffer word
ATUNIT	16	10		Unit number requesting attention
CLASS	4	10		Current command classification
FLAGS	8	2		Interface state flags
CSTATE	4	10	Y	Controller execution state
STATUS	16	8	Y	Controller status
USEL	4	10	Y	Unit number currently selected
UATTN	4	2		Bitmap of units requesting attention
RECBUF [0:65537]	8	8		Record buffer
LIBSTA	16	10		Status from last tape support library call
LENGTH	24	10		Data buffer valid length
INDEX	24	10		Data buffer current index
GAPLEN	32	10		Length of erase gap preceding the current record
INPOS	32	10		Initial tape position
RSTART	24	10		Fast rewind start time

Name	*Size*	*Radix*	*Read-Only*	*Description*
RRATE	24	10		Fast rewind rate
RSTOP	24	10		Fast rewind stop time
BTIME	24	10		Fast start time if positioned at the BOT
ITIME	24	10		Fast start time if positioned at an inter-record g
DTIME	24	10		Fast data transfer time per byte
OTIME	24	10		Fast controller overhead time
UPROP [**0:3**]	16	8		Drive properties, drives 0-3
USTATUS [**0:3**]	16	2		Unit status, drives 0-3
UOPCODE [**0:3**]	6	10	Y	Current operation code, drives 0-3
USTATE [**0:3**]	4	10	Y	Current command state, drives 0-3
UPOS [**0:3**]	32	10	Y	Current byte position, drives 0-3

The BUFWRD and RECBUF registers may be examined or deposited using any of the modes described in the Symbolic Display and Entry section above.

COPYRIGHT NOTICE and LICENSE