

HP 3000 Simulator User's Guide

15-Apr-2024

COPYRIGHT NOTICE

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

Original code published 1993-2012, written by Robert M Supnik
Copyright © 1993-2012, Robert M Supnik
Copyright © 2012-2024, J. David Bryan

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Except as contained in this notice, the names of the authors shall not be used in advertising or otherwise to promote the sale, use, or other dealings in this Software without prior written authorization from the authors.

1	Introduction.....	4
1.1	The HP 3000 Computer System	4
1.2	Simulator Files.....	5
2	Simulator Features.....	7
2.1	Hardware-Equivalent Actions	8
2.2	Simulator-Specific Commands.....	8
2.2.1	Numeric Display and Entry.....	9
2.2.2	Symbolic Display and Entry	9
2.2.3	Memory Addressing.....	11
2.2.3.1	EXAMINE, DEPOSIT, IEXAMINE, and IDEPOSIT	11
2.2.3.2	BREAK and NOBREAK.....	12
2.2.3.3	RUN and GO	12
2.2.4	START, LOAD, DUMP, and BOOT.....	12
2.2.4.1	Cold Load and Dump Commands for the Series III	12
2.2.4.2	Cold Load and Dump Commands for the Series 58	13
2.2.4.3	Cold Loading with the BOOT Command.....	13
2.2.5	POWER FAIL and POWER RESTORE	13
2.2.6	Device Configuration.....	13
2.2.7	Enabling and Disabling Devices	14
2.2.8	I/O Interface Assignments.....	14
2.2.9	SAVE and RESTORE.....	15
2.3	Realistic, Calibrated, and Optimized Timing	15
2.4	The Simulation Console and the System Console.....	17
2.5	Tracing Simulator Operations.....	17
3	Processor Device Simulations	19
3.1	Central Processing Unit.....	19
3.1.1	System Halt	21
3.1.2	Idling	22
3.1.3	Simulation Stops	22
3.1.4	Tracing	24
3.1.5	Registers.....	27
3.2	I/O Processor	28
3.3	Selector Channel.....	29
3.4	Multiplexer Channel	30
3.5	Channel Program Processor.....	32
4	Programmed I/O Device Simulations.....	34
4.1	30032B Asynchronous Terminal Controller	34
4.1.1	Terminal Data Interface	34
4.1.2	Terminal Control Interface.....	38
4.2	30033A Selector Channel Maintenance Board.....	40
4.3	30135A System Clock	41
4.4	30340A Intermodule Bus Adapter.....	42
5	Selector Channel I/O Device Simulations	45

5.1	30229B Disc Interface with Eight 7905/7906/7920/7925 Drives	45
5.1.1	Device Options.....	46
5.1.2	Unit Options.....	47
5.1.3	Diagnostic Support.....	47
5.1.4	BOOT Command	48
5.1.5	Tracing and Registers.....	49
6	Multiplexer Channel I/O Device Simulations	51
6.1	30209A Line Printer Controller with One 2607/13/17/18 Line Printer	51
6.1.1	Device Options.....	52
6.1.2	Unit Options.....	53
6.1.3	Vertical Format Unit	54
6.1.4	Tracing and Registers.....	56
6.2	30215A Tape Controller with Four 7970B/E Drives	58
6.2.1	Device Options.....	59
6.2.2	Unit Options.....	60
6.2.3	BOOT Command	60
6.2.4	Tracing and Registers.....	61
7	Intermodule Bus I/O Device Simulations.....	63
7.1	31262A General I/O Channel	63
7.1.1	Device Options.....	63
7.1.2	Tracing and Registers.....	65
7.2	31264A Asynchronous Data Communications Channel	67
7.2.1	Device Options.....	69
7.2.2	Unit Options.....	70
7.2.3	Tracing and Registers.....	71
7.3	CS/80 Disc and Tape Drives.....	73
7.3.1	Configuration	76
7.3.2	Device Options.....	76
7.3.3	Unit Options.....	77
7.3.4	BOOT Command	78
7.3.5	Tracing and Registers.....	78
7.4	7970E HP-IB Tape Controller with four Magnetic Tape Drives	79
7.4.1	Device Options.....	80
7.4.2	Unit Options.....	81
7.4.3	BOOT Command	82
7.4.4	Tracing and Registers.....	82

1 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* and *SIMH Users' Guide Supplement* manuals, which describe how to compile and run the simulator, as well as the general commands that may be entered at the Simulation Control Program (SCP) prompt.

[This simulator](#) is based on [SIMH](#) — the Computer History Simulator.

1.1 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 simulates two classic machines: the Series III and the Series 58. The Series III may be configured with the 30341A HP-IB Interface Module that connects HP-IB peripheral devices to the system.

The Series III represents the high end of the SIO machines, which consist of the 3000 CX, the Series I, the Series II, and the Series III. These machines use I/O interfaces based on the I/O Processor (IOP) bus.

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.

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 (although MPE V/R can accommodate bank numbers up to six bits in length).

The Series 58 is in the upper-mid range of the HP-IB machines, which consist of some 19 models from the low-end Series 30 to the high-end Series 70. The I/O interfaces for these machines connect to an Intermodule Bus (IMB). The various models differ mainly in memory, I/O capacity, and processor speed, with maximum memory sizes varying from 512K for the Series 30 to 8192K for the Series 70. The Series 58 supports 4096K.

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.

1.2 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_io.h	CPU-to-I/O controllers interface declarations
	hp3000_defs.h	System architectural declarations
	hp3000_gic.h	General I/O Channel architectural declarations
	hp3000_imb.h	Intermodule Bus architectural declarations
	hp3000_io.h	Device-to-IOP/channel interface declarations
	hp3000_mem.h	Main memory subsystem interface declarations
	hp_disclib.h	MAC/ICD disc controller simulator library declarations
	hp_tapelib.h	797x tape controller simulator library declarations
	hp3000_adcc.c	Asynchronous Data Communications Channel simulator
	hp3000_atc.c	Asynchronous Terminal Controller simulator
	hp3000_clk.c	System Clock simulator
	hp3000_cpp.c	Channel Program Processor 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_eis.c	CPU Extended Instruction Set simulator
	hp3000_cpu_fp.c	CPU floating point instruction set simulator
	hp3000_ds.c	Cartridge Disc Interface simulator
	hp3000_gic.c	General I/O Channel simulator
	hp3000_gic_dc.c	CS/80 disc drive simulator
	hp3000_gic_ma.c	Amigo magnetic tape subsystem simulator
	hp3000_imb.c	Intermodule Bus simulator
	hp3000_imba.c	Intermodule Bus Adapter 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
doc	hp3000_guide.pdf	HP 3000 Simulator User's Guide
	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](#)
- [The HP Computer Museum](#)

The reference materials used to develop the various simulated devices are listed in the comments at the start of the respective simulator source modules. |

See the *Available Software* section of the release notes file for information on the availability of operating system software that runs on the simulator.

2 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 32558A Series 58 Computer with up to 4096KW of memory
CPP	Channel Program Processor
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
IMBA	30340A Intermodule Bus Adapter
GIC	31262A General I/O Channel
ADCC	31264A Asynchronous Data Communications Channel
DC	HP-IB connector with eight CS/80 disc and tape drives
MA	HP-IB Magnetic Tape Controller with four 7970E drives

Typically, 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.

Certain devices support the creation of more than one instance; these are noted in their respective individual device descriptions below. For example, the GIC device supports the creation of up to 14 separate instances, representing the installation of 14 GIC Printed Circuit Assemblies in the computer chassis. Each GIC instance is a separate device, and all GICs may be used concurrently.

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

- MPE V/P version E.00.01.
- MPE V/R version E.01.00.
- MPE V/E version G.3P.00.

In addition, the simulator generally passes the HP 32230 Stand-Alone Diagnostics suite, the GIC, ADCC, and 7970E diagnostics of the HP 32231 Diagnostic/Utility System, and the GIC and CS/80 diagnostics of the HP 32341 Diagnostic/Utility System III; 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.
- Completion of the cold-load process.

MPE normally handles these conditions internally. However, when automating the cold-load process or running diagnostics, it may be helpful to set one or more of these stop conditions. In particular, stopping a simulator command file after cold loading allows additional system configuration to be performed before execution is begun. Also, 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.

2.1 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:

<i>Hardware Front-Panel Action</i>	<i>Equivalent Simulation Command</i>
Pressing the RUN/HALT button while halted	RUN
Pressing the RUN/HALT button while running	CTRL+E CTRL+E
Pressing the START button	START
Pressing the LOAD or LOAD and ENABLE buttons	LOAD
Pressing the DUMP or 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 on the Series III to the desired control and device number and pressing the LOAD and ENABLE buttons together to initiate the cold load sequence. On the Series 58, pressing the START button initiates a cold load from the system disc. In simulation, a cold load may be performed either explicitly or implicitly, as described in the *START*, *LOAD*, *DUMP*, and *BOOT* section below. Explicit operation is initiated by a **START**, **LOAD**, or **DUMP** command. 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.

2.2 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.

2.2.1 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 for a specific device with the **SET <device> <radix>** command, or the radix may be overridden temporarily with a command line switch, as follows:

SET <radix> Option	Command Switch	Data Interpretation
BINARY	-B	Binary values
OCTAL	-O	Octal values
DECIMAL	-D	Decimal values
HEX	-H	Hexadecimal values

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.

When examining or depositing into attached device files, the radix for addresses is decimal, and the default radix for numeric data is octal. The data default may be changed with the **SET <device> <radix>** command, or the radix may be overridden temporarily with one of the above command line switches.

2.2.2 Symbolic Display and Entry

When examining or depositing into memory, certain registers, or attached device files, 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
-L	A channel program instruction mnemonic
-M	A CPU instruction mnemonic
-T	A CPU status mnemonic

In addition, certain 8-bit devices allow pairs of successive bytes in their attached device files to be interpreted as 16-bit words. These devices normally display or enter values as bytes but can be overridden to work with words by including one of the following mode switches:

Switch	Mode Interpretation
-C	A two-character packed string
-M	A CPU instruction mnemonic
-W	A 16-bit numeric value

In the absence of a mode switch, 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 and channel 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. Adding 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 without 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.

2.2.3 Memory Addressing

The CPU simulator supports two forms of memory addresses:

- An absolute address consisting of a 4- or 6-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.

2.2.3.1 EXAMINE, DEPOSIT, IEXAMINE, and IDEPOSIT

The following CPU memory 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.

2.2.3.2 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.

2.2.3.3 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.

2.2.4 START, LOAD, DUMP, and BOOT

The START, LOAD, and BOOT commands are used to initiate the HP 3000 cold-load sequence that bootstraps an operating system or diagnostic program from a storage device. The DUMP command initiates the 3000 cold-dump sequence that writes the contents of memory to a storage device for later analysis. These commands behave differently, depending on the system type (IOP-based or IMB-based). The START command exists only in IMB-based systems.

2.2.4.1 Cold Load and Dump Commands for the Series III

The **LOAD** command implements the 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 for an SIO device, or 125 for an HP-IB device, or the command will not complete.

The **DUMP** command implements the 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, which defaults to octal. 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.

2.2.4.2 Cold Load and Dump Commands for the Series 58

The system control panel of the Series 58 contains START, LOAD, and DUMP buttons. Each button is associated with a pair of thumbwheel switches that are used to set the default channel and device numbers for the action. In simulation, these buttons are implemented as commands. All three commands initiate the cold-load process — the **START** and **LOAD** commands from a disc or tape device, respectively, containing an operating system, and the **DUMP** command from a disc or tape device containing the Software Dump Facility program. The syntax of the commands is:

```
START {<channel/device>}
LOAD  {<channel/device>}
DUMP  {<channel/device>}
```

If the optional channel/device value is supplied, it is deposited into the SWCH register before initiating the cold load sequence. Otherwise, the channel and device values from the corresponding thumbwheel switches on the system control panel are deposited into the SWCH register. The combined channel and device numeric values are entered in the current CPU data radix, which defaults to octal, with the value equal to the channel number times eight plus the device number. A successful cold load causes an immediate entry into the cold-load trap handler unless the COLDLOAD simulation stop is enabled. Setting thumbwheel switches and simulation stops are detailed in the *Central Processing Unit* section below.

2.2.4.3 Cold Loading with the BOOT Command

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.

2.2.5 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.

2.2.6 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.

2.2.7 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:

<i>Command</i>	<i>Action</i>
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:

<i>Command</i>	<i>Action</i>
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.

2.2.8 I/O Interface Assignments

HP 3000 Series III 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:

<i>Option</i>	<i>Valid Values</i>	<i>Action</i>
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:

<i>Device Name</i>	<i>Device Number</i>	<i>Interrupt Priority</i>	<i>Interrupt Mask</i>	<i>Service Request Number</i>
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
IMBA	125	12	—	—

The SCMB device numbers and interrupt priorities may be set to any unused values. The device numbers of the MPX and IMBA devices cannot be changed.

HP 3000 Series 58 hardware I/O interfaces have configuration switches that set the IMB channel numbers to which they respond. IMB I/O devices include a **CHANNEL** option to set the channel address switch. The default settings are:

<i>Device Name</i>	<i>Channel Number</i>
GIC	11
DC	11
MA	9
ADCC	1
IMBA	1

The IMBA channel number cannot be changed.

2.2.9 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.

2.3 Realistic, Calibrated, and Optimized Timing

Operating systems and diagnostics that interact with I/O devices often have embedded timing dependencies. In practice, such software may contain assumptions regarding the time taken by certain operations. 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. In hardware, these latencies are functions of physical movements — disc seeks, magnetic tape reads, printer paper advances, etc. A simulated device must model these physical latencies as time delays between the initiation and completion of various activities. For best performance,

these delays want to be as short as possible. However, they cannot be arbitrarily short without violating the timing requirements of system software.

Moreover, diagnostics often measure the time taken by physical operations to determine if a device is operating properly. In such cases, delays optimized to be "as short as possible" will cause the diagnostic to fail. For these applications to succeed, the simulated delays must approximate the realistic hardware delays, in terms of the number of CPU instructions executed between initiation and completion.

A third consideration is the delay to use for a time-of-day clock. Typically, such a device generates interrupts at some preset rate. Were a simulated clock to schedule these interrupts "as fast as possible," the observed time of day would run far ahead of a wall clock. Even scheduling the interrupts to occur at a realistic rate, as measured in the number of CPU instructions executed between ticks, would still increment the simulated clock at a very fast rate, given that the simulator is two orders of magnitude faster than a real HP 3000. For this device, the simulated clock must tick at the same rate as a physical wall clock, so that the time of day advances as expected.

To accommodate these conflicting requirements, user control of simulated device timing is provided.

Devices simulate their I/O operation delays 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 realistic-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 two orders of magnitude faster than the original 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 this "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. Consequently, device operations will complete far more quickly than they would do in hardware. As noted above, though, there are limits to how fast I/O operations may occur without causing software malfunctions. The default optimized-timing settings have been empirically determined to work with the supported operating systems, 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.

2.4 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 an execution command such as **RUN** 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 port 0 of the first terminal multiplexer for the system console.

For convenience and by default, the system console is initially 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 the multiplexers 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

2.5 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 CONSOLE 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.

3 Processor Device Simulations

The HP 3000 Series III computer consists of the following subsystems:

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

The HP 3000 Series 58 computer consists of the following subsystems:

- 32558A Central Processing Unit
- Channel Program Processor

3.1 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:

<i>Option</i>	<i>Action</i>
III	Configure the CPU as a Series III; default
S58	Configure the CPU as a Series 58
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; Series III default
2048K	Set the memory size to 2048K words; Series 58 default
4096K	Set the memory size to 4096K words
EIS	Enable the Extended Instruction Set firmware; default
NOEIS	Disable the Extended Instruction Set firmware
CIS	Enable the COBOL II Extended Instruction Set firmware; Series 58 default
NOCIS	Disable the COBOL II Extended Instruction Set firmware; Series III default
ARS	Enable auto-restart after a power failure; default
NOARS	Disable auto-restart after a power failure
CALTIME	Use calibrated process clock timing; default
REALTIME	Use realistic process clock timing
IDLE	Enable idle detection
NOIDLE	Disable idle detection; default
START=<chan>[;<dev>]	Set the channel and device numbers of the start device; default is 11, 0
LOAD=<chan>[;<dev>]	Set the channel and device numbers of the load device; default is 9, 0

<i>Option</i>	<i>Action</i>
DUMP=<chan>[;<dev>]	Set the channel and device numbers of the dump device; default is 11, 0
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

Selecting the CPU model enables the set of I/O devices that are compatible with the model and disables all incompatible devices. For example, the ATC device is enabled when the CPU is a Series III and disabled when the CPU is a Series 58. The current CPU model also controls the applicability of certain configuration commands, as noted below.

The initial configuration is a Series III system with 1024K words of memory. Available memory sizes are specific to the CPU model selected. 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 EIS** command simulates the installation of the HP 30012A Extended Instruction Set firmware on the Series III ROM PCA and removal of the W1 and W8 jumpers from the CIR PCA to enable decoding of the instructions. The set includes four-word extended-precision floating-point instructions and decimal arithmetic instructions. The EIS is standard on HP 3000 Series II and III machines, so this option is enabled by default. If the set is disabled with the **SET CPU NOEIS** command, MPE will emulate the instructions in software.

The **SET CPU CIS** command simulates the installation of the HP 32234A COBOL II Extended Instruction Set firmware on the Series III 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 EIS and CIS firmware is standard on the Series 58. Commands that attempt to change these firmware settings on this model are rejected with **Command not allowed** errors.

The position of the PF/ARS switch on the back of the Series III 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.

Automatic power fail recovery is standard on the Series 58, so entering the **SET CPU ARS** and **SET CPU NOARS** commands will be rejected with **Command not allowed** errors.

Calibrated timing adjusts the process clock to match actual elapsed time. Realistic timing bases the process-clock interval on machine instructions executed. For the Series 58, the process clock and the system clock that maintains the time of day for MPE are equivalent.

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 **START**, **LOAD**, and **DUMP** options configure the default channel and device numbers on the corresponding thumbwheel switches of the Series 58 system control panel. The **START** and **LOAD** settings reference the system disc and tape devices, respectively, that contain an operating system, and the **DUMP** setting references a disc or

tape device that contains the Software Dump Facility program. For each setting, if the device number is omitted, it defaults to the device at bus address 0.

The **DUMPDEV** and **DUMPCTL** options configure the set of jumpers on the rear of the Series III 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.

Entering a **START**, **LOAD**, or **DUMP** option while the CPU is configured as a Series III or entering a **DUMPDEV** or **DUMPCTL** option while the CPU is configured as a Series 58 will be rejected.

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 (Series III only)
SHOW CPU PANEL	Display the system control panel settings (Series 58 only)
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, may be obtained with the **SHOW CPU SPEED** command. The multiple is referenced to the configured CPU model. The speed reported will not be representative if the simulator was executing a PAUS instruction when it was stopped.

3.1.1 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

Code	Reason for System Halt
3	Code segment 1 is absent
4	Stack overflow while executing on the ICS
5	CST length violation
6	I/O device timeout while executing IXIT or cold load
7	Cold load channel program checksum error
8	Cold load channel program abort
9	Dispatcher already enabled while executing PSEB
12	Cold load channel is not the system controller
13	CST violation while executing in segment 1
17	Cold load channel cannot become the controller
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 one of the cold-load commands has been entered.

3.1.2 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 I/O Channels. If enabled, the simulator will idle only during execution of PAUS.

3.1.3 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
COLDLOAD	Stop when a cold load completion trap occurs

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.

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 may change the meaning of those instructions. The **UNIMPL** option stops the simulator for instructions in category 4.

Undefined instruction mnemonics are printed in lowercase when displaying memory or tracing execution. For example, tracing execution of the canonical SED 1 instruction encoding (030041 octal) prints **SED 1** in the trace. Tracing a non-canonical encoding (030043 octal) prints **sed 1**.

Setting the **COLDLOAD** option stops the simulator when the cold load trap is taken at the end of the loading process. The option is only needed with IMB systems, as the IOP systems automatically stop when the cold load is complete. In a simulator command file, stopping after cold loading allows additional system configuration to be performed before execution begins.

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.

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

Explicitly bypassing the **COLDLOAD** stop is not needed. When the stop occurs, the cold load trap has already been taken, and execution is stopped at the entry to the cold load handler. Resuming execution, with or without the **-B** switch, proceeds to execute the handler.

Note that the **PAUSE** bypass 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.

3.1.4 Tracing

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

<i>Option</i>	<i>Reporting Level</i>
INSTR	Machine instructions executed
DATA	Memory data accesses
FETCH	Memory instruction fetches
REG	Register values
OPND	Memory operand values
EXEC	Matching instruction execution states
IRQ	Interrupt requests and traps
PSERV	Process clock service events (periodic)
IMBUS	Intermodule Bus signals and data words read and written

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 of the above 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 **IRQ** option traces interrupt and trap requests, including dispatcher interrupts and interrupt returns. These items are also traced with the **INSTR** option, but the **IRQ** option is useful if full instruction tracing is unnecessary, for example when tracing I/O device operation.

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 **IMBUS** option traces the Intermodule Bus signals and data exchanged with the Series 58 memory controller. This option has no effect if the CPU is configured as a Series III.

If one or more of the CPU trace options are in effect at the time that a simulation stop occurs, an instruction trace showing the stop reason will be included.

The trace formats are interpreted as follows:

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

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

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

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

>>CPU instr: 00.056235 101005 simulation stop: Unimplemented instruction

Stop reason
Octal status register value (STA)
Octal address (P)
Octal bank (PBANK)

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

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

>>CPU fetch: 00.010342 020320 instruction fetch

Memory access classification
Octal data (instruction opcode)
Octal address (P + 1)
Octal bank (PBANK)

>>CPU reg: 00.006500 000002 A 123003, B 001340, X 000000, M i t r o c CCL

Register values (0-4 TOS registers, X, STA)
Octal stack register count (SR)
Octal stack memory address (SM)
Octal bank (SBANK)

>>CPU reg: 00.000000 000001 PB 010000, PL 025227, DL 001770,
DB 002000, Q 006510, Z 007000

Base register values
Current code segment number (from STA)
Zero
Octal bank (DBANK)

>>CPU opnd: 00.135771 000000 DFLC '+','!'
>>CPU opnd: 00.045071 000252 target fraction 3 length 6,"002222"

Instruction-specific operand value
Instruction-specific data value
Octal address (effective address)
Octal bank (PBANK, DBANK, or SBANK)

>>CPU pserv: Process clock delay 3970 service entered on the ICS

Service entry and stack status

>>CPU imbus: Memory controller received opcode Write address 00015470 data 041407
>>CPU imbus: Memory controller returned data 000000 with signals ADN | DDN | PRO

Bus opcode, address, data, and signals

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 1.100; SET CPU DEBUG; GO
BREAK 1.200; SET CPU NODEBUG; GO
```

These commands will enable tracing when the program counter reaches location 100 in memory bank 1 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.

3.1.5 Registers

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

<i>Name</i>	<i>Size</i>	<i>Radix</i>	<i>Symbolic</i>	<i>Read-Only</i>	<i>Description</i>
CIR	16	—	✓	✓	Current Instruction Register
NIR	16	—	✓	✓	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
SR	3	8			Stack Register Counter
Z	16	8			Stack Limit Register
SBANK	4	8			Stack Segment Bank Register
RA	16	8	✓		Top of Stack Register
RB	16	8	✓		Top of Stack – 1 Register
RC	16	8	✓		Top of Stack – 2 Register
RD	16	8	✓		Top of Stack – 3 Register
X	16	8	✓		Index Register
STA	16	—	✓		Status Register
SWCH	16	8	✓		Switch Register
CPX1	16	8			Run-Mode Interrupts Register
CPX2	16	8			Halt-Mode Interrupts Register
PCLK	16	8			Process Clock Register
CR	16	8			Timer Count Register
SIR	16	8			Status and Interrupt 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. The CR and SIR registers are present in the Series 58 and are significant only when that CPU model is selected.

Three 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 Series III module number during an unsolicited module interrupt. SP0 is a scratch-pad register that contains the numeric halt code when a system halt occurs. CNTR and MOD are provided for the CPU diagnostic and have no other use under simulation. SP0 is provided for the cold dump process to record the reason for a system halt.

3.2 I/O Processor

The HP 30003B I/O Processor works in conjunction with the Series III 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:

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

<i>Option</i>	<i>Reporting Level</i>
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
```

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

The I/O Processor state contains these registers:

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

3.3 Selector Channel

The HP 30030C Selector Channel provides high-speed data transfer between a device and the Series III 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
```

Memory access classification

Octal data (memory contents)

Octal address

Octal bank number

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 allocation
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	✓	Input buffer
OUTBUF	16	8	✓	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.

3.4 Multiplexer Channel

The HP 30036B Multiplexer Channel provides high-speed data transfer between from one to sixteen devices and the Series III 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:

<i>Option</i>	<i>Action</i>
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:

<i>Option</i>	<i>Reporting Level</i>
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:

<i>Name</i>	<i>Size</i>	<i>Radix</i>	<i>Description</i>
IDLE	1	2	Channel is inactive
COUNT	32	10	Count of active transfers
EXCESS	32	10	Channel cycles used in excess of allocation
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

3.5 Channel Program Processor

The Channel Program Processor is part of the standard microcode of the IMB-based systems, and a modified CPP is also used with the HP 30341A HP-IB Interface Module, where it resides on the processor card. It assumes the function of the multiplexer and selector channels of the IOP-based systems, except that instead of having dedicated hardware to execute channel programs, dedicated firmware is used instead. The CPP gains control between machine instructions when the IMB CSRQ or IRQ signal is asserted. The CPP also executes a subset of the I/O instructions of the HP-IB machines on behalf of the HP 30340A Intermodule Bus Adapter. When the CPP exits, either at the end of a channel program or when the program is waiting for a device event, such as a parallel poll response, CPU instruction execution resumes.

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

<i>Option</i>	<i>Action</i>
FILTER=<list>	Suppress tracing for channel numbers in the list
NOFILTER	Enable tracing for all channel numbers; default
DEBUG=<option>	Enable tracing
NODEBUG	Disable tracing; default

Enabling CPP 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 channels are of interest, others may be filtered out of the results by specifying their channel numbers in a **SET CPP FILTER** command. This command takes a list of individual channel numbers separated by semicolons and ranges of the form **low-high**. For example, **SET CPP FILTER=3;7-9** would exclude lines pertaining to channel 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:

<i>Option</i>	<i>Reporting Level</i>
CMD	Channel program command and instruction executions
OPND	SIOP channel program instructions
SERV	Channel processor service events
DATA	Data read from and written to memory

The **CMD** option traces the commands and channel instructions executed by the channel processor, as well as those I/O instructions executed at the request of the IMBA. The **OPND** option traces the channel program initiated by an SIOP instruction. This option attempts to list the entire program through the final Interrupt/Halt instruction reachable from the initial location. It accounts for intermediate conditional and unconditional jumps but may misinterpret memory contents if the locations skipped contain data rather than instructions. The **SERV** option traces channel processor event service scheduling and entries. The **DATA** option traces memory accesses performed by the channel processor.

Examples of the trace formats follow:

```
>>CPP cmd: Channel processor servicing channel 1
>>CPP cmd: Channel processor executing IOCL for the IMBA
>>CPP cmd: Executing channel 11 Identify | response 0212H

>>CPP opnd: 00.140576 001416 Read secondary 0EH count 20 address 00124621
>>CPP opnd: 00.140603 001000 Wait | response 0036
>>CPP opnd: 00.140605 002400 Device Specified Jump 140610
>>CPP opnd: 00.140610 000601 Interrupt/Halt 0001 | CPVA 1
```

```
>>CPP serv: Channel processor running
>>CPP serv: Completion delay 2 service scheduled
>>CPP serv: Channel processor idled

>>CPP data: 00.001423 003000 bank 0 read
>>CPP data: 00.001424 000000 bank 0 read
>>CPP data: 00.000770 000007 absolute read
>>CPP data: 00.000774 100000 absolute write
```

Channel program instruction opcodes and operand values are decoded and presented in symbolic format for easier interpretation. Memory operands are printed in the CPU's address radix, which is octal. General data operands are printed in the CPU's data radix, which defaults to octal but may be set to a different radix. Command secondaries and Amigo IDs are printed in hexadecimal and suffixed with the letter H. Data accesses for values that must reside in bank 0, such as channel program instructions, are classified separately from values that may reside in any bank, such as DRT entries.

The Channel Program Processor state contains these registers:

Name	Size	Radix	Description
ENTDLY	24	10	Entry delay until CSRQ recognition
IRQEN	1	2	Interrupt requests are enabled
IMBIRQ	8	8	Pending interrupt device number
LIMIT	16	2	Set of channels that have reached their execution limit
RESTART [0:15]	8	2	Set of channel devices that are to be restarted
FILTER	16	2	Bitmap of channels to include in traces

4 Programmed I/O Device Simulations

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

- 30032B Asynchronous Terminal Controller
- 30033A Selector Channel Maintenance Board
- 30135A System Clock
- 30340A Intermodule Bus Adapter

4.1 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.

4.1.1 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 listening 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 an individual TDI channel:

```
ATTACH ATCD<chan> <port-name>{;<rate>-<size><parity><stopbits>}
```

...where **chan** is the TDI channel number from 1-15, and **port-name** is the host name of the serial port to use (e.g., /dev/ttyS0 or COM1).

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).

If the port configuration string is omitted, the default configuration specified by the host system for that port is used.

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
CONNECT	Wait for and establish a channel connection
FASTTIME	Use optimized timing; default
REALTIME	Use realistic timing
TERMINAL	Connect using Telnet or serial ports; default
DIAGNOSTIC	Connect using diagnostic test cables
LINEORDER=<c1>[;<c2>...]	Set the channel connection order
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

Connections to the listening port or a serial port are normally established automatically while the simulation is executing. Connections attempted while the simulation is stopped are deferred until simulated execution is resumed. In cases where connections must be established before simulation resumes, such as when the executing program immediately writes to or reads from the channel, the **SET ATCD CONNECT** command may be used. If a connection is pending, it will be established immediately. Otherwise, the simulator will wait for a connection. To abort the wait manually and return to the SCP prompt, enter the SCP interrupt character, which defaults to CTRL+E.

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.

The **LINEORDER** option specifies the order in which new connections are assigned to multiplexer channels. The arguments may be single channel numbers or ranges of channel numbers of the form **m-n**, with multiple arguments separated by semicolons, except that channel 0 may not be included, as it is reserved for the system console. Telnet connections to the listening port will be assigned to multiplexer channels in the sequence specified. Omitted channels will not receive connections, unless the **ALL** keyword is supplied as the last argument. In the absence of a **SET ATCD LINEORDER** command, connections will be assigned by default in ascending channel order. The default order may be reestablished by specifying the command **SET ATCD LINEORDER=ALL**.

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

<i>Option</i>	<i>Action</i>
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. Adding the **-N** (new file) switch clears the contents of the log file if it is present. Without the **-N** switch, channel output will be appended to any preexisting log file content. 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 **SET ATCDn DISCONNECT** command will drop and then raise the Data Terminal Ready line; to disconnect the serial port, issue a **DETACH ATCDn** 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  csr: Channel 2 connection established
>>ATCD  csr: 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:

<i>Name</i>	<i>Size</i>	<i>Radix</i>	<i>Symbolic</i>	<i>Description</i>
CNTL	16	8		Control register
STAT	16	8		Status register
READ	16	8	✓	Read register
WRITE	16	8	✓	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 0-20
RBUFR [0:20]	16	8	✓	Receive channel buffers, channels 0-20
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	✓	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.

4.1.2 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:

<i>Option</i>	<i>Action</i>
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:

<i>Command</i>	<i>Action</i>
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:

<i>Option</i>	<i>Reporting Level</i>
CSRW	Interface control, status, read, and write actions
PSERV	Poll unit service events (periodic)
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:

<i>Name</i>	<i>Size</i>	<i>Radix</i>	<i>Description</i>
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 0-15
ES1 [0:15]	1	2	Enable status line 1 interrupt, channels 0-15

<i>Name</i>	<i>Size</i>	<i>Radix</i>	<i>Description</i>
MS2 [0:15]	1	2	Status line 2 match, channels 0-15
MS1 [0:15]	1	2	Status line 1 match, channels 0-15

4.2 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:

<i>Option</i>	<i>Action</i>
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:

<i>Option</i>	<i>Reporting Level</i>
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  csr: 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 present
TRMCNT	1	2	Terminal count condition is present
MISCOMP	1	2	Miscompare condition is present
DEVEND	1	2	Device end condition is present
STOP	1	2	Transfer has been stopped

4.3 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 presetable 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:

<i>Option</i>	<i>Reporting Level</i>
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  csrwl: Control is load rate | select limit | 1 millisecond rate
>>CLK  csrwl: Status is DIO OK | LR = CR | limit selected | 1 millisecond rate
>>CLK  pservl: Service entered with counter 0 increment 1 limit 1
>>CLK  pservl: Rate 1 millisecond delay 389 service rescheduled
>>CLK  iobusl: Received data 000000 with signals DSETINT
>>CLK  iobusl: Returned data 000000 with signals INTREQ
```

The System Clock state contains these registers:

<i>Name</i>	<i>Size</i>	<i>Radix</i>	<i>Description</i>
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

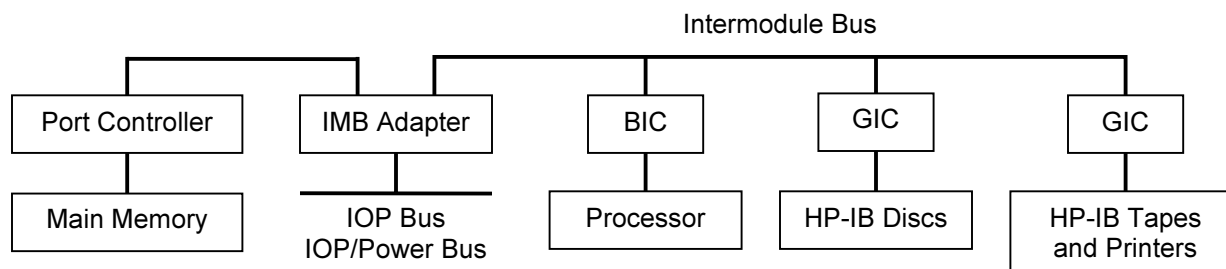
4.4 30340A Intermodule Bus Adapter

The HP 30340A Intermodule Bus Adapter is a component of the 30341A HP-IB Interface Module (nicknamed the "Starfish") that connects HP-IB peripheral devices to the Series III. HP-supported devices are the HP 7933 CS/80 disc drive, the HP 7976 ½-inch magnetic tape drive, and the HP 2680 laser page printer. A small, rack-mounted chassis houses the module and contains these Printed Circuit Assemblies:

- 30340-60002 Intermodule Bus Adapter (IMBA)
- 30070-60012 Processor
- 31000-60053 Bus Interface Controller (BIC)
- 31262-60001 General I/O Channel (GIC)
- 30340-60001 Backplane (6 position)

The IMBA connects via flat ribbon cables to the Series III IOP bus, the IOP/Power bus, and to an additional 30030-60020 Port Controller installed in the Series III chassis. The Processor is a Series 33 processor card containing modified firmware that implements the Channel Program Processor and Series 33 I/O instruction set. The BIC and GIC are standard Series 33 cards.

Graphically, the module appears as follows:



In simulation, the following I/O devices are used in a Series III system to implement the HP-IB Interface Module

- 30340A Intermodule Bus Adapter
- Channel Program Processor
- 31262A General I/O Channel
- CS/80 Disc Drives
- Amigo Tape Drives

The HP 30340A Intermodule Bus Adapter is installed in the Starfish chassis and serves as the gateway between the Series III and the Starfish. It serves three functions.

First, it enables the Series III to communicate with the Starfish processor that runs HP-IB channel programs. It is attached to the Series III's IOP Bus and IOP Power Bus, and to the Starfish's Intermodule Bus (IMB). Issuing an SIO order to the IMBA asserts the Channel Service Request (CSRQ) line on the IMB, causing the Starfish processor to run its Channel Program Processor (CPP) microcode. The CPP retrieves the memory address of the channel program and other parameters from a "mailbox" area in the Series III main memory and initiates program execution. Channel program completion status is returned to the Series III via the mailbox. The IMBA is permanently assigned to IMB channel 1.

Second, it permits the DMA controller on the Starfish's General I/O Channel (GIC) to read and write data in the Series III main memory. The IMBA interprets IMB Memory Read and Memory Write cycles and passes them through to a dedicated HP 30030 Port Controller card in the Series III. It reflects main memory parity errors back to the GIC via the IMB.

Third, it allows the Starfish processor to assert the Series III Interrupt Request (INTREQ) line to cause an external interrupt. Because the Starfish supports multiple devices and therefore multiple interrupt sources, the IMBA has a DEVNO response register that permits it to "masquerade" as different devices during an interrupt poll cycle. The Starfish processor can configure the IMBA to supply any desired device number, rather than its own permanently assigned device number of 125.

The IMBA device supports configuration of the interrupt priority and of tracing using these device options:

<i>Option</i>	<i>Action</i>
INTPRI=<n>	Set the interrupt priority; default is 12
DEBUG=<option>	Enable tracing
NODEBUG	Disable tracing; default
ENABLED	Enable the HP-IB Module devices
DISABLED	Disable the HP-IB Module devices; default

Unlike most Series III interfaces, the IMBA is not controlled by an I/O driver. Instead, MPE automatically detects the presence of the IMBA at system startup, regardless of whether or not HP-IB peripherals are configured into the system. Therefore, the IMBA is disabled by default and must be enabled explicitly when HP-IB devices are to be used. The HP-IB Interface Module operates as a unit, so enabling or disabling the IMBA enables or disables the entire module, affecting the CPP, GIC, DC, and MA devices.

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

<i>Option</i>	<i>Reporting Level</i>
CSRW	Interface control, status, read, and write actions
DATA	Port controller data accesses
IOBUS	I/O Processor bus signals and data words received and returned
IMBUS	Intermodule Bus signals and data words received and returned

The **CSRW** option traces control, status, read, and write commands sent to the interface. The **DATA** option traces memory accesses performed by the port controller on behalf of the GIC. The **IOBUS** option traces the I/O Processor backplane signals and data received and returned via the interface. The **IMBUS** option traces the Intermodule Bus signals and data received and returned via the interface.

Examples of the trace formats follow:

```
>>IMBA  csrw: Status is SIO OK | DIO OK | Power OK
>>IMBA  csrw: Channel program started
>>IMBA  data: 00.131220 004400      dma read
>>IMBA  data: 00.124621 000017      dma write
>>IMBA  iobus: Received data 000000 with signals DSTATSTB
>>IMBA  iobus: Returned data 160000 with signals (none)
>>IMBA  imbus: Channel 1 asserts CSRQ1
>>IMBA  imbus: Channel 1 received opcode I/O Write command SMSK data 000020
>>IMBA  imbus: Channel 1 returned data 040000 with signals ADN | DDN | PRO
```

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

The IMB Adapter state contains these registers:

<i>Name</i>	<i>Size</i>	<i>Radix</i>	<i>Description</i>
CSRQ	1	2	Channel service request flip-flop
DEVNO	8	8	Device number for interrupt acknowledgement

5 Selector Channel I/O Device Simulations

The Selector Channel controls these Series III I/O device interfaces:

- 30229B Disc Interface

5.1 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 | -N} DSn <image-filename>
```

Adding the **-R** (read-only) switch is equivalent to setting the drive's Disc Protect or Read Only switch to the *On* position. Adding the **-N** (new file) switch creates a full-size image file, equivalent to formatting the new disc before use.

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>
```

5.1.1 Device Options

Device options that may be specified are:

<i>Option</i>	<i>Action</i>
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:

<i>Command</i>	<i>Action</i>
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.

5.1.2 Unit Options

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

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

The **UNLOAD** 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:

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

5.1.3 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 Mismatch, Head-Sector Mismatch, 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.

5.1.4 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	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	DS device number							

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

5.1.5 Tracing and Registers

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

<i>Option</i>	<i>Reporting Level</i>
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 event entry
>>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:

<i>Name</i>	<i>Size</i>	<i>Radix</i>	<i>Read-Only</i>	<i>Description</i>
SIOSY	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

Name	Size	Radix	Read-Only	Description
WAIT	1	2		Data wait is active
CLEAR	1	2		A hardware clear is active
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	✓	Controller operation code
CSTATS	5	8	✓	Controller status
CSTATE	2	10	✓	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	✓	Current operation code, drives 0-7 and controller
USTATUS [0:8]	32	2		Unit status, drives 0-7 and controller
USTATE [0:8]	4	10	✓	Current command state, drives 0-7 and controller
UPOS [0:8]	32	10	✓	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.

6 Multiplexer Channel I/O Device Simulations

The Multiplexer Channel controls these Series III I/O device interfaces:

- 30209A Line Printer Controller
- 30215A Tape Controller

6.1 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.

The printer unit may also be attached to a FIFO (pipe) file to direct printer output to a receiving program. The receiving end of the file must be connected before the **ATTACH** is given to avoid erroneous console operation. The **-N** switch is not relevant when a FIFO file is attached.

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>
```

6.1.1 Device Options

Device options that may be specified are:

<i>Option</i>	<i>Action</i>
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:

<i>Command</i>	<i>Action</i>
SHOW LP	Display the device and unit configuration
SHOW LP MODES	Display the timing and connection modes

6.1.2 Unit Options

Unit options that may be specified are:

<i>Option</i>	<i>Action</i>
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.

6.1.3 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.

6.1.4 Tracing and Registers

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

<i>Option</i>	<i>Reporting Level</i>
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:

<i>Name</i>	<i>Size</i>	<i>Radix</i>	<i>Symbolic</i>	<i>Read-Only</i>	<i>Description</i>
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
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	✓		Read word
WRITE	16	8	✓		Write word
J2WX	10	2			Jumper J2W10-J2W1 configuration
DATOUT	16	8	✓		Data out word
DATIN	16	8	✓		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	✓		Print buffer
OVPCHR	8	8	✓		Overprint character
FORMLN	8	10		✓	Form length in lines
VFU [0:144]	12	2		✓	Vertical format unit channels
PUNCHR	8	8	✓		Punched channel character
UNPCHR	8	8	✓		Unpunched channel character
BTIME	24	10			Fast printer buffer load delay time
PTIME	24	10			Fast printing delay time
STIME	24	10			Fast paper slew per-line delay time
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.

6.2 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>
```

6.2.1 Device Options

Device options that may be specified are:

<i>Option</i>	<i>Action</i>
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:

<i>Command</i>	<i>Action</i>
SHOW MS	Display the device and unit configuration
SHOW MS TIMING	Display the timing mode

6.2.2 Unit Options

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

<i>Option</i>	<i>Action</i>
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:

<i>Command</i>	<i>Action</i>
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

6.2.3 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	9	10	11	12	13	14	15
0	0	0	0	0	1	1	0	MS device number							

The control byte defaults to the Read Record tape command.

6.2.4 Tracing and Registers

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

<i>Option</i>	<i>Reporting Level</i>
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: Tape unit 0 loaded and set online
>>MS    cmd: Tape unit 0 Write Record of 20-byte record succeeded
>>MS    inco: Tape unit 0 Write Record started at position 6120
>>MS    inco: Tape unit 0 Write Record completed at position 6148
>>MS    csrwr: Control is 000005 (Write Gap)
>>MS    csrwr: Status is ready | load point | 1600 bpi | no error | unit 0
>>MS    state: Tape unit 0 Write Gap start phase delay 3794 service scheduled
>>MS    serv: Tape unit 0 service entered
>>MS    xfer: Tape unit 0 Write Record word 1 is 052525
>>MS    iobus: Tape controller idle state returned data 000000 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	✓	Controller execution state
STATUS	16	8	✓	Controller status
USEL	4	10	✓	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
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 gap
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	✓	Current operation code, drives 0-3
USTATE [0:3]	4	10	✓	Current command state, drives 0-3
UPOS [0:3]	32	10	✓	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.

7 Intermodule Bus I/O Device Simulations

The Channel Program Processor controls these Intermodule Bus device interfaces:

- 31262A General I/O Channel (GIC)
- 31264A Asynchronous Data Communications Channel (ADCC)

The simulator provides the following HP-IB devices that may be attached to the GIC:

- CS/80 Disc and Tape Drives
- 7970E ½-inch Magnetic Tape Drives

7.1 31262A General I/O Channel

The HP 31262A General I/O Channel (GIC) connects HP-IB disc and tape units to the Intermodule Bus. It serves as the gateway between the CPU and the Hewlett-Packard Interface Bus.

Each GIC provides a Processor to HP-IB (PHI) chip that acts as the bus controller to manage up to eight connected devices. Each device may be from a different protocol class (disc vs. tape vs. printer, CS/80 vs. Amigo). Although there are hardware restrictions on the type and number of devices that may be connected to a single GIC, those restrictions are not present in simulation. Five hardware configuration switches are present on the GIC PCA.

In addition to the PHI, each GIC contains a DMA sequencer, a watchdog timer, and bus request and handshake circuitry. It connects to the Intermodule Bus and has sixteen addressable registers — eight provided by the PHI, and eight provided by the DMA sequencer. The sequencer is a state machine with 27 states; the current five-bit state number is presented as bits 3-7 of Register 8. Once the target device is addressed to talk or listen and the DMA sequencer is configured with the starting address, byte count, and direction, the GIC is able to transfer the entire block to or from main memory without CPU intervention. Transfers are performed with IMB memory read and write cycles. When installed in a Starfish chassis, these are handled by the IMBA, which passes the requests through to a dedicated Series III port controller that interfaces to main memory.

7.1.1 Device Options

Device options include configuring the channel number and various switch settings. The command form is:

```
SET GIC <device-option>
```

The options that may be specified are:

<i>Option</i>	<i>Action</i>
COUNT=<n>	Set the number of GIC instances; default is 1
CHANNEL=<n>	Set switch S4 to the channel number; default is 11
HPIB	Connect the HP-IB cable to the peripherals; default
DIAGNOSTIC	Connect the HP-IB cable to a second GIC
CPU	Set switch S1 to the CPU position; default
CPP	Set switch S1 to the CPP position
A	Set switch S2 to the A position; default
B	Set switch S2 to the B position
SYS	Set switch S3 to the ON (System Controller) position; default
NOSYS	Set switch S3 to the OFF (not System Controller) position
OPERATE	Set switch S5 to the OPER position; default
TEST	Set switch S5 to the TEST position
DEBUG=<option>	Enable tracing
NODEBUG	Disable tracing; default
ENABLED	Enable the device; default for IMB systems
DISABLED	Disable the device; default for IOP systems

The **COUNT** option determines the number of separate instances of the GIC interface. While the various system configuration guides state that a small number of GICs are supported in hardware, this limitation does not exist in simulation. If additional GIC instances are desired, the total number may be specified with the **COUNT** option. For example, issuing the **SET GIC COUNT=3** command will create two additional GIC instances, while issuing **SET GIC COUNT=1** will remove all instances except the original one. When more than one GIC instance exists, the original GIC is renamed GICA, while the additional GICs are named GICB, GICC, etc. The original GIC will return to its original name if the count is reduced to 1.

The **CHANNEL** option configures the GIC to use the specified IMB channel address from 0-15. Peripheral devices that connect to the GIC must be set to the same channel number.

Setting the **DIAGNOSTIC** option simulates the installation of an HP-IB cable between two GIC cards as required by the General I/O Channel Diagnostic; the **DIAGNOSTIC** option also must be set on the other GIC to complete the connection. Setting the **HPIB** option simulates the installation of an HP-IB cable between the GIC and the peripherals.

The **CPU**, **CPP**, **A**, **B**, **SYS**, **NOSYS**, **OPERATE**, and **TEST** options configure the positions of the S1, S2, S3, and S5 hardware switches on the GIC PCA. The default settings are used for normal MPE operation. The alternate settings are used only when running the GIC diagnostic.

Device configuration may be displayed with the following commands:

<i>Command</i>	<i>Action</i>
SHOW GIC	Display the complete device configuration
SHOW GIC CHANNEL	Display the channel address
SHOW GIC COUNT	Display the number of GIC instances

The GIC device has two units that implement the watchdog timer and the NRFD holdoff timer. These units are disabled and not user-configurable.

In response to a **START**, **LOAD**, **DUMP**, **BOOT**, **RUN**, **GO**, **CONTINUE**, or **STEP** command to begin instruction execution, a check is made to ensure that GIC channel address conflicts and bus address conflicts are not present.

If the check detects an inconsistent configuration, the simulator will report the error to the simulation console, and execution will be inhibited. For example:

```
sim> SET DC CHANNEL=11
sim> SET MA CHANNEL=11
sim> SHOW DC
DC, fast timing, channel=11, 8 units
  DC0, 404MB, not attached, 7933A, bus=0, unit=0
sim> SHOW MA
MA, fast timing, channel=11, bus=0, 4 units
  MA0, not attached, unlimited capacity, SIMH format, offline, write ring
  MA1, not attached, unlimited capacity, SIMH format, offline, write ring
  MA2, not attached, unlimited capacity, SIMH format, offline, write ring
  MA3, not attached, unlimited capacity, SIMH format, offline, write ring
sim> RUN
Bus address 0 conflict (DC and MA)

Simulation stopped, P: 000000 (NOP,NOP)
sim> SET MA CHANNEL=9

sim> SET GIC COUNT=2
sim> SHOW GICA
GICA, channel=11, HP-IB cable, ...
sim> SHOW GICB
GICB, channel=11, HP-IB cable, ...
sim> RUN
Channel number 11 conflict (GICA and GICB)

Simulation stopped, P: 000000 (NOP,NOP)
sim>
```

The simulator will not begin program execution until all conflicts are resolved.

7.1.2 Tracing and Registers

When a debug log has been established, tracing may be configured by specifying one or more of the reporting level options:

<i>Option</i>	<i>Reporting Level</i>
CMD	Interface commands executed
CSRW	Interface control, status, read, and write actions
CSIRQ	Intermodule Bus channel service and interrupt requests
SERV	Interface unit service events
XFER	Data received and transmitted via HP-IB
BUF	Data read from and written to the FIFO buffer
STATE	DMA sequencer state changes executed
IMBUS	Intermodule Bus signals and data words received and returned

The **CMD** option traces the commands executed by the PHI and DMA controller. The **CSRW** option traces the control and status commands sent to the GIC. The **CSIRQ** option traces channel service and interrupt requests and responses asserted on the Intermodule Bus. The **SERV** option traces the watchdog timer event scheduling and entries. The **XFER** option traces the command and data bytes passed across the HP-IB cable. The **BUF** option traces the data bytes loaded into and unloaded from the PHI FIFO buffers. The **STATE** option traces DMA

state machine changes. The **IMBUS** option traces the Intermodule Bus signals and data received and returned via the channel.

Examples of the trace formats follow:

```
>>GIC cmd: DMA completed with status 0
>>GIC cmd: PHI becomes system controller and goes offline with reset
>>GIC cmd: PHI becomes controller-in-charge with IFC
>>GIC cmd: PHI relinquishes controller-in-charge by going online
>>GIC cmd: DMA aborted

>>GIC csrw: Register 0 (FIFO) Data 171 sent to bus
>>GIC csrw: Register 0 (FIFO) Data Tagged | 001 received from bus
>>GIC csrw: Register 7 (Bus address) control is online | bus address 0
>>GIC csrw: Register 1 (Status) status is controller | system controller
>>GIC csrw: Service requested for PHI interrupt

>>GIC csirq: Channel 11 asserts IRQ
>>GIC csirq: Channel 11 device 0 interrupted as DEVNO 88
>>GIC csirq: Channel 11 asserts CSRQ1
>>GIC csirq: Service requested for PHI interrupt

>>GIC serv: Timeout clock enabled
>>GIC serv: Channel 11 timeout delay 400778 service scheduled
>>GIC serv: Timeout service entered

>>GIC xfer: REN | IFC | SRQ | ATN | EOI | DAV | NRFD | NDAC | DIO | Mnemonic
>>GIC xfer: REN | | | | ATN | EOI | DAV | | | 01H | Poll
>>GIC xfer: REN | | | | ATN | | DAV | | | DFH | Untalk
>>GIC xfer: REN | | | | | | DAV | | | 40H | Data

>>GIC buf: Command byte 076 loaded into outbound FIFO count 1
>>GIC buf: Enable reception count 2 unloaded from outbound FIFO count 0
>>GIC buf: Tagged data byte 022 loaded into inbound FIFO count 2
>>GIC buf: Attempted load into full outbound FIFO data 000000

>>GIC state: DMA executing state 0 count 20
>>GIC state: DMA executing state 8 count 20
>>GIC state: DMA exiting in state 8 count 19

>>GIC imbus: Channel 11 received opcode I/O Write command IOCL data 000000
>>GIC imbus: Channel 14 asserts CSRQ1
>>GIC imbus: Channel 14 returned data 000000 with signals ADN | DDN | PRO
```

The GIC device state contains these registers:

Name	Size	Radix	Description
REN	1	2	Bus REN signal
IFC	1	2	Bus IFC signal
SRQ	1	2	Bus SRQ signal
ATN	1	2	Bus ATN signal
EOI	1	2	Bus EOI signal
DAV	1	2	Bus DAV signal
NRFD	1	2	Bus NRFD signal
NDAC	1	2	Bus NDAC signal
DIO	8	16	Bus data
TALK	32	2	Bus talker bitmap
LSTN	32	2	Bus listeners bitmap
INFIFO [0:7]	16	8	PHI inbound FIFO data
OUTFIFO [0:7]	16	8	PHI outbound FIFO data
PHIREG[0:7]	16	8	PHI configuration registers 0-7
GICREG[0:7]	16	8	GIC configuration registers 8-F
DMBUSY	1	2	DMA busy
DMSTATUS	2	10	DMA termination status
DMSTATE	5	10	DMA state number
DMADDRS	16	8	DMA address
DMCOUNT	16	10	DMA remaining byte count
IRQMASK	16	8	Interrupt request mask

The INFIFO and OUTFIFO registers are arranged in first-in, first-out order. Element 0 is always the first word to be removed from the FIFO memory, element 1 is the second word to be removed, etc. The IRQMASK value shows the word supplied in the last SMSK IMB command. The GIC interrupt mask flip-flop is set if the bit corresponding to the channel number is set.

7.2 31264A Asynchronous Data Communications Channel

The HP 31264A Asynchronous Data Communications Channel is an eight-port terminal multiplexer used with the HP 3000 Series 30 through 70 systems. The ADCC consists of a main channel that provides the first four ports and an extender channel that provides the second four. Together, they connect from one to eight serial terminals or modems to the HP 3000 at 14 programmable baud rates from 50 to 9600 bits per second. Character sizes are also programmable from 5 to 8 bits in length, excluding the start bit and one or two stop bits. Parity is optional and may be configured for even or odd parity generation and checking. Each port can be independently configured, including for separate send and receive rates.

MPE reserves port 0 of the ADCC on IMB channel 1 for the system console. The other ports of the first ADCC, and all ports of additional ADCCs, may be used for user terminals.

The ADCC provides the Transmitted Data (BA) and Received Data (BB) serial lines, the Request to Send (CA), Data Terminal Ready (CD), Data Rate Selector (CH), and Secondary Request to Send (SCA) control lines, and the Clear to Send (CB), Data Set Ready (CC) Ring Indicator (CE), Data Carrier Detect (CF), and Secondary Data Carrier Detect (SCF) status lines for each of eight terminals or data sets. The CE and CH signals are available only when the extender card is present.

The ADCC performs input and output through Telnet sessions connected to a user-specified listening port or through individually specified host serial ports. Concurrent Telnet and serial connections are allowed. The ADCC

also supports serial printers by specifying the log files to capture the printed output. No additional connections are needed to use this feature.

When configured by MPE as a modem port, Data Terminal Ready must be set to enable the port to accept an incoming connection. When the port connects, Data Set Ready and Data Carrier Detect will be set. 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.

The **ATTACH** command specifies the local network port to be used for Telnet connections:

```
ATTACH ADCC <netport>
```

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

When multiple ADCC instances exist, each may be attached to its own unique network listening port. With this configuration, each Telnet connection would specify the ADCC instance to use by its network port choice. Alternatively, ADCC instances may be attached to *port groups* that use the same network listening port. Attaching a second instance to the same port number as the first will make the two instances part of the same port group. Telnet connections to the common listening port will assign the connections to the available terminal ports provided by the first instance and, when those are all occupied, to the ports provided by the second instance. Only when all terminal ports of all ADCC instances belonging to the same port group are in use will a connection attempt be refused. Multiple concurrent port groups are also possible by assigning different network ports to each group of instances that belong to a given port group. A mix of individual and grouped ADCC instances is also possible.

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

```
ATTACH ADCC<n> <serport-name>{;<rate>-<size><parity><stopbits>}
```

...where **n** is the ADCC port number from 0-3 or 0-7, depending on whether the extender is present, and **serport-name** is the host name of the serial port to use (e.g., */dev/ttyS0* or *COM1*).

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*).

If the port configuration string is omitted, the default configuration specified by the host system for that port is used.

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

```
SET ADCC <device-option>  
SET ADCCn <unit-option>
```

7.2.1 Device Options

Device options that may be specified are:

<i>Option</i>	<i>Action</i>
COUNT=<n>	Set the number of ADCC instances; default is 1
CHANNEL=<n>	Set switch S1 to the channel number; default is 1
TERMINAL	Connect using Telnet or serial ports; default
DIAGNOSTIC	Connect using the diagnostic test connector
CPU	Set switch S2 to the CPU position; default
CPP	Set switch S2 to the CPP position
EXTENDER	Install the main and extender ADCC cards; default
NOEXTENDER	Install the main ADCC card only
CONNECT	Wait for and establish a port connection
FASTTIME	Use optimized timing; default
REALTIME	Use realistic timing
LINEORDER=<c1>[;<c2>...]	Set the port connection order
DEBUG=<option>	Enable tracing
NODEBUG	Disable tracing; default
ENABLED	Enable the device; default for IMB systems
DISABLED	Disable the device; default for IOP systems

The **COUNT** option determines the number of separate instances of the ADCC interface. If additional ADCC instances are desired, the total number may be specified with the **COUNT** option. For example, issuing the **SET ADCC COUNT=3** command will create two additional ADCC instances, while issuing **SET ADCC COUNT=1** will remove all instances except the original one. When more than one ADCC instance exists, the original ADCC is renamed ADCCA, while the additional ADCCs are named ADCCB, ADCCC, etc. The original ADCC will return to its original name if the count is reduced to 1.

The **CHANNEL** option configures the ADCC to use the specified IMB channel address from 1-15. One ADCC must be configured for channel 1 to support the MPE system console. Additional ADCCs may specify any unused address.

Setting the **DIAGNOSTIC** option simulates the installation of the HP 5060-5563 ADCC Test Connector as required by the Asynchronous Data Communications Channel Diagnostic. For each port, this connects the Send Data line to the Receive Data line, as well as the following sets of control and status lines: CA to CB and CF, CD to CC, SCA to SCF, and CH to CE. In addition, all sessions are disconnected, and the multiplexer is detached from the Telnet listening port. While in diagnostic mode, the **ATTACH ADCC** command is not allowed. Setting the **TERMINAL** option simulates the installation of the ADCC Cable Hood between the ADCC and the terminal junction panel., allowing the multiplexer to be attached to accept incoming connections again.

The **CPU** and **CPP** options configure the position of the S2 hardware switches on the ADCC PCAs. The default setting is used for normal MPE operation. The alternate setting is used only when running the ADCC diagnostic.

The **EXTENDER** option installs the ADCC extender and the crossover connector between the main and extender PCAs to provide eight terminal ports and support for the CE and CH modem signals. The **NOEXTENDER** option removes the extender, providing only the four ports of the main PCA.

Connections to the listening port or a serial port are normally established automatically while the simulation is executing. Connections attempted while the simulation is stopped are deferred until simulated execution is resumed. In cases where connections must be established before simulation resumes, such as when an executing program immediately writes to or reads from the terminal port, the **SET ADCC CONNECT** command may be used. If a connection is pending, it will be established immediately. Otherwise, the simulator will wait for a connection.

To abort the wait manually and return to the SCP prompt, enter the SCP interrupt character, which defaults to CTRL+E.

The ADCC supports programmable data transfer rates from 50 to 9600 baud. When the **REALTIME** option 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. Selecting the **FASTTIME** option 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 operation. Resetting the device with the **RESET -P** (power-on reset) command restores the original optimized time.

The **LINEORDER** option specifies the order in which new connections are assigned to multiplexer ports. The arguments may be single port numbers or ranges of port numbers of the form **m-n**, with multiple arguments separated by semicolons, except that port 0 may not be included for the ADCC configured as IMB channel 1, as it is reserved for the system console. Telnet connections to the listening port will be assigned to multiplexer ports in the sequence specified. Omitted ports will not receive connections, unless the **ALL** keyword is supplied as the last argument. In the absence of a **SET ADCC LINEORDER** command, connections will be assigned by default in ascending port order. The default order may be reestablished by specifying the command **SET ADCC LINEORDER=ALL**.

7.2.2 Unit Options

Unit options that may be specified for individual ADCC ports are:

<i>Option</i>	<i>Action</i>
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 port 0 of the first ADCC
NOCAPSLOCK	Input characters are unchanged; default
UC	Upshift lowercase output characters to uppercase
7B	Output with high-order bit cleared; default
7P	Output with high-order bit cleared, non-printing suppressed; default for port 0
8B	Output characters are unchanged
LOG=<filename>	Enable output logging
NOLOG	Disable output logging; default
DISCONNECT	Disconnect the port

Ports 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 I/O driver. 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 port 0 of the ADCC on IMB channel 1 (the system console) and **NOCAPSLOCK** for all other ports.

Each port may be set to one of four output modes: **UC**, **7B**, **7P**, or **8B**. The default mode is **7P** for channel 1 port 0 and **7B** for all other ports. 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 MPE terminal types 4 and 6.

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 port must be set to **8B**, **REMOTEACK**, and **NOCAPSLOCK** modes.

Each port supports independent I/O logging to a file. Adding the **-N** (new file) switch clears the contents of the log file if it is present. Without the **-N** switch, port output will be appended to any preexisting log file content. Disabling logging also closes the log file.

A port that has been configured in MPE as a serial printer using driver HIOASLP2 must have its log file open before printing is attempted. If the file is closed, the output will be silently discarded. Placing the required **SET ADCCn LOG=<filename>** command in the simulator startup file is recommended.

A port may be manually disconnected from its associated Telnet session with the **SET ADCCn DISCONNECT** command. Otherwise, the connection will remain open until disconnected either by the Telnet client or a **DETACH ADCC** command. For a serial connection, the **SET ADCCn DISCONNECT** command will drop and then raise the Data Terminal Ready line; to disconnect the serial port, issue a **DETACH ADCCn** command.

ADCC device configuration may be displayed with the following commands:

Command	Action
SHOW ADCC	Display the device and unit configuration
SHOW ADCC CHANNEL	Display the IMB channel assignment
SHOW ADCC CONNECTIONS	Display the port connections
SHOW ADCC STATISTICS	Display the port I/O statistics
SHOW ADCC LINEORDER	Display the port connection order

ADCC unit configuration may be displayed with the following commands:

Command	Action
SHOW ADCC<n>	Display the selected port configuration
SHOW ADCC<n> LOG	Display the selected port logging status

7.2.3 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
CSRW	Interface control, status, read, and write actions
CSIRQ	Intermodule Bus channel service and interrupt requests
SERV	Port unit service events
PSERV	Poll unit service events (periodic)
XFER	Data receptions and transmissions
IMBUS	Intermodule Bus signals and data words received and returned

The **CSRW** option traces control, status, read, and write commands sent to the interface. The **CSIRQ** option traces channel service and interrupt requests and responses asserted on the Intermodule Bus. The **SERV** option traces port unit event service scheduling and entries. The port 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 and serial ports for connections and input characters to be received by the active ADCC ports. The **XFER** option traces the characters received and transmitted by the active ports. The **IMBUS** option traces the Intermodule Bus signals and data received and returned via the interface.

Examples of the trace formats follow:

```
>>ADCC  csrw: Port 7 connection established
>>ADCC  csrw: Register 0 (FIFO) command is Talk 7
>>ADCC  csrw: Register 2 (IRQ conditions) status is IRQ | available | empty
>>ADCC  csrw: Register 0 (FIFO) command is Secondary 06H
>>ADCC  csrw: Register 0 (FIFO) data is 0BH
>>ADCC  csrw: Port 7 modifier 6 transmitter is 1200 baud
>>ADCC  csrw: Port 7 modifier 0 UART input character is 'U'
>>ADCC  csrw: Register 0 (FIFO) special data 125 received
>>ADCC  csrw: Port 7 receive data overrun

>>ADCC  csirq: Port 2 asserts CSRQ for DSJ 2 condition modem change
>>ADCC  csirq: Channel 2 device 2 interrupted as DEVNO 18
>>ADCC  csirq: OBII status is channel 2 | device 2
>>ADCC  csirq: Port 6 asserts CSRQ for DSJ 0 condition RR occupied
>>ADCC  csirq: OBSI status is device request | channel 2 | device 6

>>ADCC  serv: Port 7 delay 1621 service scheduled
>>ADCC  serv: Port 7 service entered

>>ADCC  pserv: Poll delay 3891 service entered
>>ADCC  pserv: Poll service stopped

>>ADCC  xfer: Port 3 connection raised DSR
>>ADCC  xfer: Port 3 break detected
>>ADCC  xfer: Port 3 character CR received by the UART
>>ADCC  xfer: Port 3 character CR sent
>>ADCC  xfer: Port 3 character LF sent
>>ADCC  xfer: Port 3 character NUL discarded by output filter
>>ADCC  xfer: Port 3 disconnection dropped DSR

>>ADCC  imbus: Channel 2 received opcode I/O Write command WIOC register C
              data 000012 with signals ADO | DDO | PRI
>>ADCC  imbus: Channel 2 returned data 000000 with signals ADN | DDN | IRQ | PRO
>>ADCC  imbus: Channel 2 asserts IRQ
>>ADCC  imbus: Channel 2 received opcode I/O Read command IPOLL register 0
              data 000000 with signals ADO | DDO | PRI
>>ADCC  imbus: Channel 2 returned data 020000 with signals ADN | DDN | IRQ | PRO
```


The ADCC state contains these registers:

<i>Name</i>	<i>Size</i>	<i>Radix</i>	<i>Symbolic</i>	<i>Description</i>
ADDRGRP	3	10		HP-IB addressing group
PRIADDR	5	10		HP-IB primary address
SECADDR	5	10		HP-IB secondary address
IRQMASK	16	8		Interrupt mask channel bit
IRQ	8	2		Interrupt request port bits
CSRQ	16	2		Channel service request port bits
CPRUN	8	2		Channel program running bits
FTIME	24	10		Fast transmit/receive time
THR [0:7]	8	8	✓	Transmitter holding register, ports 0-7
TR [0:7]	8	8	✓	Transmitter register, ports 0-7
RHR [0:7]	8	8	✓	Receiver holding register, ports 0-7
RR [0:7]	8	8	✓	Receiver register, ports 0-7
DSJ [0:7]	2	10		Device Specified Jump response, ports 0-7
MCONTROL [0:7]	5	2		Modem control lines, ports 0-7
MSTATUS [0:7]	5	2		Modem status lines, ports 0-7
SVMASK [0:7]	3	2		Service conditions mask, ports 0-7
STMASK [0:7]	5	2		Modem status mask, ports 0-7
STREF [0:7]	5	2		Modem status reference, ports 0-7

The ADDRGRP register value is 1 for a Listen Address Group command and 2 for a Talk Address Group command. The PRIADDR value selects the port number to address, and the SECADDR value selects the operation to perform.

The THR, TR, RHR, and RR registers default to single-character display but may be examined or deposited using any of the modes described in the *Symbolic Display and Entry* section above.

The MCONTROL register value displays five binary values corresponding to the input character echo state, plus the CA (RTS), CD (DTR), CH (DRS), and SCA (SRTS) modem control lines, respectively.

The MSTATUS, STMASK, and STREF register values each display five binary values corresponding to the CB (CTS), CC (DSR), CE (RI), CF (DCD), and SCF (SDCD) modem status lines, respectively.

7.3 CS/80 Disc and Tape Drives

The HP 31262A General I/O Channel connects from one to eight Command Set 80 disc or tape units to the HP 3000. Available models include disc drives ranging in capacity from 28MB to 574 MB and cartridge tape drives having 16 MB, 67 MB, or 134 MB capacity. In hardware, up to four CS/80 drives can be connected to a single GIC; bus loading and the 1 MB/second data transfer rate impose the limitation. This restriction does not exist under simulation.

Certain CS/80 disc models were offered with integrated cartridge tape drives. These were typically configured as a unified controller occupying a single HP-IB address, with the disc and tape drives responding at that address as CS/80 unit numbers 0 and 1, respectively. However, these combination models were supported by MPE only when the tape drive had its own dedicated controller. The HP 7911, 7912, and 7914 models could be ordered with Option 001 to provide a second, independent tape controller that used a second HP-IB address, with the drive as unit 0. The HP 7942 and 7946 models could not, meaning that they could not be used under MPE.

The CS/80 simulator does not support separate controllers for the combination drives. Therefore, the HP 7911, 7912, and 7914 models are offered with Option 140, which deletes the integrated tape drive, and the HP 7942 and 7946 are replaced by the tape-less 7941 and 7945 models. The combination models can be effectively simulated by including a standalone HP 9144 cartridge tape drive with each associated tape-less disc model.

Note that under MPE V/R, only the HP 7911, 7912, 7933, and 7935 disc models are supported. These models are also supported under MPE V/P and V/E. Later models, including the HP 9144 cartridge tape drive, are supported only by specific MPE V/E releases; see the HP *Peripheral Configuration Guide* (part number 5954-9306, February 1988), available from the HP Computer Museum, for details.

Attaching a disc or tape image file to a unit simulates installing a removable HP 7935 disc pack, loading the heads on a fixed disc drive, or inserting a tape cartridge into a tape drive:

```
ATTACH {-E | -N} {-R | -X} {-S} DCn <image-filename>
```

Specifying the **-E** (exist) switch ensures that the specified file exists; if it does not, **File open error** is printed. Specifying the **-N** (new file) switch creates a new, blank, full-size image file and initializes the media. If an existing file is specified with **-N**, it is reinitialized. For tape images, initialization includes certification. Omitting both switches will open the file if it exists or create the file if it does not; a created file will have zero size and must be initialized before use with the MPE VINIT subsystem FORMAT command.

For cartridge tape units, the **-E** and **-N** switches behave as above. When creating a new file for a 9144A drive, including the **-S** (short-length) switch specifies a 16 MB, 150-foot cartridge tape image; omitting it specifies a long-length 67 MB, 600-foot cartridge. For a 9145A drive, the respective sizes are 32 MB and 134 MB. When attaching an existing file, the file size automatically determines the cartridge size; including **-S** verifies that the file being attached is a short cartridge image.

Specifying the **-R** (read-only) switch is equivalent to setting the cartridge's Write Protect switch to the *Safe* position. **-R** is only valid with an existing file; if it is specified with a nonexistent file, **File open error** is printed.

Tape cartridges must be initialized and certified before they can be used. Existing tape images are checked for initialization and certification when they are attached. A certified image must contain only data records and file marks. An initialized image must contain only erased blocks, data records, and file marks. To be valid, each record of the image file must be properly formatted. If the image fails the check, **Volume is not initialized** or **Volume is not certified** is printed; the file remains attached, but it must be initialized or certified before use.

For unrestricted use, a tape cartridge image file must contain the full number of formatted blocks defined by the cartridge size. Some conversion programs produce properly formatted images containing less than the required number of tape blocks. Such valid but incomplete files may be attached and will be marked as initialized and certified, but they are restricted to read-only access; **Volume is write protected** is printed on the console as a warning. Incomplete files appear as short or long cartridges, depending on the number of blocks contained in the file. Programs attempting to read blocks that are not physically present in the file will receive *No Data Found* errors.

The user may optionally extend such incomplete files to their full complement of tape blocks by specifying the **-X** (extend) switch. This appends and certifies the remaining blocks in the image file. Files are extended to the long cartridge size unless the **-S** switch is added and the number of blocks in the incomplete file is less than the defined short-cartridge count. After extending, file use is unrestricted. **-X** may also be used to extend a full-size short-length cartridge image to the long-length cartridge image size. **-X** is ignored if the image is not initialized and certified; it only extends valid images.

To summarize, the switch combinations perform the following actions for disc and tape images, respectively:

Switch	Disc Action
(none)	Open an existing disc file or create a new disc file
-E	Open an existing disc file
-N	Create and initialize a new disc file

Switch	Tape Action
(none)	Open an existing tape file or create a new long tape file
-S	Open an existing short tape file or create a new short tape file
-E	Open an existing tape file
-E -S	Open an existing short tape file
-N	Create and certify a new long tape file
-N -S	Create and certify a new short tape file
-R	Open and write protect an existing tape file
-R -S	Open and write protect an existing short tape file
-X	Open an existing tape file and extend it to long size if needed
-X -S	Open an existing tape file and extend it to short size if needed

The **-R**, **-S**, and **-X** switches are used only with tape images and are rejected if specified for a disc image. Specifying **-S** with an existing file that is larger than the short cartridge size is also rejected. The **-N** and **-R** switches are mutually exclusive, as are the **-X** and **-R** switches. All of these combinations result in **Invalid switch** errors.

At simulator startup, all units return *Power Fail* status. If needed, this status may be cleared manually by issuing a **RESET -C DC** command to issue an HP-IB Device Clear to all enabled units. An unattached unit also returns *Not Ready* status.

Attached media image files may be viewed or changed with an **EXAMINE** or **DEPOSIT** command that specifies a unit name and a decimal linear address. These commands display or enter values as 8-bit bytes by default. However, the **-C**, **-M**, and **-W** command-line switches may be included to interpret pairs of successive bytes as 16-bit words, as described the *Symbolic Display and Entry* section above.

Detaching the disc image file simulates unloading the heads and, for the HP 7935, removing the disc pack:

```
DETACH {-F} DCn
```

Detaching does not occur immediately. Instead, a Release Request is sent to the MPE operating system, and the simulator prints, e.g., **Bus 0 unit 0 release requested** on the simulation console. If the operating system grants the request, the file is detached, and **Bus 0 unit 0 released and detached** is printed. If the request is denied by the operating system, **Bus 0 unit 0 release denied** is printed, and the image file remains attached to the unit.

Specifying the **-F** (force) switch will detach the file immediately without notifying the operating system. This switch should be used with care, typically only when the operating system is not running. Forcing a disc image change while MPE is running will result in disc corruption.

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

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

A failed seek will report *Unit Fault* status. A failed read or write will report *Unrecoverable Data* status. The target operating system will then react to this error as though the drive had encountered a bad disc or tape block.

7.3.1 Configuration

Device and unit options include configuring the timing, bus address, drive type, and the ability to add and remove drives from the bus. The command forms are:

```
SET DC <device-option>
SET DCn <unit-option>
```

A note on terminology: both CS/80 and SIMH use the terms *device* and *unit*. In hardware, a CS/80 device corresponds to a physical disc or tape drive identified by its bus address, and a CS/80 unit corresponds to the device controller or a separate storage implementation within a device identified by its unit number. For example, the HP 7933A device contains a hard disc drive at unit 0 and the disc controller at unit 15.

A SIMH device refers to the set of connected drives, and a SIMH unit refers to one storage implementation connected to a media image file on the host platform. For this DC device, CS/80 units and SIMH units correspond one-for-one, although the assignment of CS/80 units to SIMH units is arbitrary and determined by the unique combination of bus address and CS/80 unit number. In the remainder of this section, the terms *device* and *unit* have their SIMH meanings unless the term *CS/80* is specified explicitly.

7.3.2 Device Options

Device options that may be specified are:

<i>Option</i>	<i>Action</i>
COUNT=<n>	Set the number of DC instances; default is 1
FASTTIME	Use optimized timing; default
REALTIME	Use realistic timing
CHANNEL=<n>	Set the channel to which the drives are connected; default is 11
DEBUG=<option>	Enable tracing
NODEBUG	Disable tracing; default
ENABLED	Enable the device; default for IMB systems
DISABLED	Disable the device; default for IOP systems

The **COUNT** option determines the number of separate instances of the DC drive set. If additional GIC instances are created, then each GIC must have a corresponding set of connected drives. The total number of drive sets may be specified with the **COUNT** option. For example, issuing the **SET DC COUNT=2** command will create an additional DC instance that may be connected to the second GIC by assigning the same channel number to both. Issuing **SET DC COUNT=1** will remove all instances except the original one. When more than one DC instance exists, the original DC is renamed DCA, while the additional instances are named DCB, DCC, etc. The original DC will return to its original name if the count is reduced to 1.

When realistic timing is enabled, the simulation accurately models the disc and tape 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 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 operation. Entering a *power-on* reset command, **RESET -P DC**, restores the original optimized times.

The **CHANNEL** option connects the DC drive set to the GIC with the specified IMB channel address from 2-15. The address determines the specific GIC to which the drives will respond. More than one DC instance can be connected to the same GIC, as long as the bus addresses assigned to the devices are unique.

Device configuration may be displayed with the following commands:

Command	Action
SHOW DC	Display the device and unit configuration
SHOW DC CHANNEL	Display the channel address

7.3.3 Unit Options

Unit options that may be specified for individual CS/80 drives are:

Option	Action
7911A	Set the drive type to 7911A with Option 140
7912A	Set the drive type to 7912A with Option 140
7914A	Set the drive type to 7914A with Option 140
7933A	Set the drive type to 7933A; default
7935A	Set the drive type to 7935A
7936H	Set the drive type to 7936H
7937H	Set the drive type to 7937H
7945A	Set the drive type to 7945A
7957B	Set the drive type to 7957B
7958B	Set the drive type to 7958B
7959B	Set the drive type to 7959B
9144A	Set the drive type to 9144A
9145A	Set the drive type to 9145A
BUS=<n>	Set the drive's bus address; default is the unit number
ENABLED	Enable the unit; default for unit 0
DISABLED	Disable the unit; default for units 1-7

The **7911A**, **7912A**, **7914A**, **7933A**, **7935A**, **7936H**, **7937H**, **7945A**, **7957B**, **7958B**, and **7959B** options select the disc drive model. The **9144A** and **9145A** options select standalone cartridge tape drive models. Option 140 deletes the integrated cartridge tape drive from the HP 7911A, 7912A, and 7914A.

The **BUS** option sets the HP-IB address of the drive. Bus addresses range from 0-7, and each address must be unique before program execution begins. For example, assuming that both DC0 and DC1 are enabled, a 7912A and 7933A might be configured as follows:

```
sim> SET DC0 7912A,BUS=3
sim> SET DC1 7933A,BUS=5
```

The assigned **DC{n}** unit numbers need not be successive, nor must the CS/80 bus addresses be ascending; any enabled SIMH unit can be assigned to any CS/80 device address.

All units but the first are initially disabled. A disabled unit is removed from the bus and does not communicate with the interface. Enabled drives that are unattached still communicate with the GIC and continue to receive bus transactions. It is therefore more efficient to disable all unused units. Note that a unit may be enabled within the configuration command, e.g., **SET DC1 ENABLED,7912A,BUS=3**.

Unit configuration may be displayed with the following command:

Command	Action
SHOW DC<n>	Display the selected unit's configuration

In response to a **START**, **LOAD**, **DUMP**, **BOOT**, **RUN**, **GO**, **CONTINUE**, or **STEP** command to begin instruction execution, a check is made to ensure that GIC channel address and CS/80 bus address and model number conflicts are not present. If the check detects an inconsistent configuration, the simulator will report the error to the simulation console, and execution will be inhibited. For example:

```
sim> SHOW DC
DC, fast timing, channel=11, 8 units
  DC0, 65MB, not attached, 7912A, bus=0, unit=0
  DC1, 28MB, not attached, 7911A, bus=0, unit=0
sim> RUN
DC bus device 0 has multiple assignments for unit 0

Simulation stopped, P: 000000 (NOP,NOP)
sim>
```

The simulator will not begin program execution until all conflicts are resolved.

7.3.4 BOOT Command

For IMB systems, the **BOOT** command may be used as an alternative to **START** or **LOAD** to initiate the system cold load procedure for the specified unit. The **BOOT DC<n>** command is the simulation equivalent to setting the START thumbwheel switches to the appropriate channel and device numbers and then pressing the START system control panel button 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	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0		Channel				Device	

Cold loading a system from the disc or cartridge tape is supported.

For Starfish systems, the **BOOT** command may be used an alternative to **LOAD**. The **BOOT DC<n>** command is equivalent in hardware to setting the SWCH register to configure the device numbers, 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	9	10	11	12	13	14	15
DC device number								IMBA device number							

The upper byte defaults to the DRT number of the specified disc unit, and the lower byte defaults to the DRT number of the IMB Adapter.

7.3.5 Tracing and Registers

When a debug log has been established, 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
STATE	Controller state changes executed
SERV	Controller and drive unit service events

The **CMD** option traces the commands executed by the drives. The **INCO** option traces the beginning and ending of commands, including command termination status. The **STATE** option traces the changes between the internal controller execution states. The **SERV** option traces controller and drive service scheduling and entries.

Examples of the trace formats follow:

```
>>DC cmd: Device 0 Set Unit 0 executed
>>DC cmd: Device 0 Set Length 65024 executed
>>DC cmd: Device 0 Set Address 2 executed
>>DC cmd: Device 0 Locate and Read executing

>>DC inco: Device 0 unit 0 Locate and Read started
>>DC inco: Device 0 unit 0 read block 1
>>DC inco: Device 0 unit 0 Locate and Read completed with qstat 0

>>DC state: Device 0 transitioned from Reporting Wait state to Command Wait state
>>DC state: Device 0 parallel poll response disabled
>>DC state: Device 0 transitioned from Command Wait state to Execution Wait state
>>DC state: Device 0 transitioned from Execution Wait state to Read state
>>DC state: Device 0 parallel poll response enabled
>>DC state: Device 0 transitioned from Read state to Reporting Wait state

>>DC serv: Device 0 state Execution Wait positioning delay 16 service scheduled
>>DC serv: Device 0 state Execution Send data delay 1 service scheduled
>>DC serv: Device 0 unit 0 state Reporting Wait service entered
```

The CS/80 device state contains these registers:

Name	Size	Radix	Read-Only	Description
PP	1	2	✓	Parallel poll is active
PPR	8	2	✓	Set of parallel poll responses
OPCODE [0:7]	6	10	✓	Controller opcode, devices 0-7
UNIT [0:7]	4	10	✓	Current unit selected, devices 0-7
CSTATE [0:7]	6	10	✓	Controller state, devices 0-7
CTIME	24	10		Fast controller overhead time
STIME	24	10		Fast controller per-byte status transfer time
KTIME	24	10		Fast unit one-step seek time
TTIME	24	10		Fast unit additional step traverse time
PTIME	24	10		Fast unit on-track positioning time
BTIME	24	10		Fast unit per-block traverse time
DTIME	24	10		Fast unit per-byte data transfer time
RTIME	24	10		Fast unit block residue traverse time

7.4 7970E HP-IB Tape Controller with four Magnetic Tape Drives

The HP-IB interface for the HP 7970E 1600-bpi PE ½-inch magnetic tape drive connects one to four drives to the HP 31262A General I/O Channel. This is the required connection method when using the drive with IMB-based HP 3000 computer systems. The interface installs in the chassis of the master drive and replaces the HP 30215 tape controller and parallel interface that is used with IOP-based 3000s such as the Series II and III. The master drive could be ordered with an HP-IB interface by specifying Option 436, or a parallel-interface drive could be retrofitted with an HP-IB interface by ordering HP part number 26072A.

Designed originally for connection to the HP 300 ("Amigo") computer system, the command set was expanded to include several related to HP-IB operation. They include the Amigo Identify and Amigo Clear sequences, Read and Write Loopback channel tests, and Device Specified Jump commands. The interface also includes a 128-byte

FIFO buffer between the drives and the bus to enable more efficient use of the GIC to which it is attached. The interface occupies a single bus address, with each drive having a user-selectable unit number from 0-3.

The simulation provides up to four tape drives. Attaching a tape image file to a unit simulates mounting a tape reel on a drive:

```
ATTACH {-R} {-F} MAn {<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.

Attached tape image files may be viewed or changed with an **EXAMINE** or **DEPOSIT** command that specifies a unit name and a decimal linear address. These commands display or enter values as 8-bit bytes by default. However, the **-C**, **-M**, and **-W** command-line switches may be included to interpret pairs of successive bytes as 16-bit words, as described in the *Symbolic Display and Entry* section above.

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 Multiple Track Error status. The target operating system will then react to this error as though a real drive had encountered a bad tape record.

A drive's unit number is not set explicitly. Instead, the drive unit number is derived from the simulation unit number. For example, unit MA0 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 timing, bus address, tape reel size, tape image format, and the ability to set drives offline or online. The command forms are:

```
SET MA <device-option>
SET Man <unit-option>
```

7.4.1 Device Options

Device options that may be specified are:

<i>Option</i>	<i>Action</i>
FASTTIME	Use optimized timing; default
REALTIME	Use realistic timing
CHANNEL=<n>	Set the channel to which the drives are connected; default is 9
BUS=<n>	Set the bus address to which the drives are connected; default is 0
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 MA	Display the device and unit configuration
SHOW MA CHANNEL	Display the channel assignment
SHOW MA BUS	Display the bus address assignment

7.4.2 Unit Options

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

Option	Action
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 default 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. In addition, the tape image format may be specified temporarily in each **ATTACH** command.

Drive configuration may be displayed with the following commands:

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

7.4.3 BOOT Command

For IMB systems, the **BOOT** command may be used as an alternative to **LOAD** to initiate the system cold load procedure for the tape drive. The **BOOT MA0** command is the simulation equivalent to setting the LOAD thumbwheel switches to the appropriate channel and device numbers and then pressing the LOAD system control panel button 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	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0		Channel				Device	

For Starfish systems, the **BOOT** command may be used as an alternative to **LOAD**. The **BOOT MA0** command is equivalent in hardware to setting the SWCH register to configure the device numbers, 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	9	10	11	12	13	14	15
MA device number								IMBA device number							

The upper byte defaults to the DRT number of the specified disc unit, and the lower byte defaults to the DRT number of the IMB Adapter.

The system microcode supports cold loading only from unit 0. Attempting to boot from another unit will result in a **Command not allowed** error.

7.4.4 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
STATE	Controller command state changes executed
SERV	Tape unit service events
XFER	Data reads and writes
IOBUS	Tape controller 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 **STATE** option traces the entries into each of the internal controller execution states. The **SERV** option traces tape unit event service scheduling and entries. The **XFER** option traces the data bytes read from or written to the tape. The **IOBUS** option traces the signals and data received and returned via the data, flag, and function buses between the bus interface and the tape controller.

Examples of the trace formats follow:

```
>>MA    cmd: Device 1 Tape Command Write Record executing
>>MA    cmd: Device 1 Device Specified Jump 0 executing
>>MA    cmd: Device 1 Write Data executing
>>MA    cmd: Tape unit 0 Write Record of 200-byte record succeeded

>>MA    inco: Device 1 Read Status command started
>>MA    inco: Device 1 status is load point | online
>>MA    inco: Device 1 Read Status command completed with DSJ 0
>>MA    inco: Tape unit 3 Write Record started at position 0
```

```
>>MA    inco: Tape unit 3 write gap call of 2.9-inch erase gap succeeded
>>MA    inco: Tape unit 3 write call of 100-word record succeeded
>>MA    inco: Tape unit 3 Write Record completed at position 5008

>>MA    state: Device 1 transitioned from Command Ready state to Command Send state
>>MA    state: Tape unit 3 Write Record start phase event entry
>>MA    state: Device 1 parallel poll response enabled

>>MA    serv: Tape unit 1 Read Record data phase delay 5 service scheduled
>>MA    serv: Device 1 Tape unit 1 service entered
>>MA    serv: Device 1 Bus controller state Command Send service entered

>>MA    xfer: Tape controller Write Record FIFO 90 data 141 byte 99 loaded
>>MA    xfer: Tape unit 3 Write Record FIFO 93 data 010 byte 10 unloaded

>>MA    iobus: Tape controller idle state received data 000007 flags CMXEQ | INTOK
>>MA    i0bus: Tape controller busy state returned data 000001 functions IFGTC
```

The tape drives state contains these registers:

Name	Size	Radix	Read-Only	Description
DEVS	8	2	✓	Bus address bitmap of devices present on the bus
XFER	1	2	✓	Tape data transfer across channel is active
PP	1	2	✓	Parallel poll is in progress
PPR	8	2		Bus address bitmap of parallel poll responses
ISTATE	4	10		Interface state
ICMD	4	10		Interface command
COUNT	16	10		Count of bytes read or written
DSJ	2	10		Device Specified Jump result
CSTATE	4	10	✓	Controller execution state
STATUS	16	8	✓	Controller status
USEL	4	10	✓	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
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 gap
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	✓	Current operation code, drives 0-3
USTATE [0:3]	4	10	✓	Current command state, drives 0-3
UPOS [0:3]	32	10	✓	Current byte position, drives 0-3

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