Interdata 16b/32b Simulator Usage
30-Jan-2007

COPYRIGHT NOTICE

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

This memorandum documents the Interdata 16b and 32b simulators.

1 Simulator Files

sim/	scp.h	
	sim_console.h	
	sim_defs.h	
	sim_fio.h	
	sim_rev.h	
	sim_sock.h	
	sim_tape.h	
	sim_timer.h	
	sim_tmxr.h	
	scp.c	
	sim_console.c	
	sim_fio.c	
	sim_sock.c	
	sim_tape.c	
	sim_timer.c	
	sim_tmxr.c	
sim/interdata/	id_defs.h	
	id16_cpu.c	[id32_cpu.c]
	id16_dboot.c	[id32_dboot.c]
	id_dp.c	
	id_fd.c	
	id_fp.c	
	id_idc.c	
	id_io.c	
	id_lp.c	
	id_mt.c	
	id_pas.c	
	id_pt.c	
	id_tt.c	
	id_ttp.c	
	id_uvc.c	
	id16_sys.c	[id32_sys.c]

2 Interdata Features

The Interdata simulator includes simulators for a variety of 16b (I3, I4, I5, 70, 80, 7/16, 8/16, 8/16E) and 32b (7/32, 8/32) models. This is by no means a complete sampling of all the variations in the Interdata/Perkin-Elmer family. The 32b family included options for special communications instructions (7/32C, 8/32C), as well as a later extension for virtual memory (3200 series).

The Interdata simulator is configured as follows:

```
device names      simulates

CPU - 16b         Interdata 3, 4, 5, 70, 80, 7/16, or 8/16 CPU with 64KB
                   memory
                  Interdata 8/16E CPU with 256KB memory
```

```
CPU - 32b         Interdata 7/32 or 8/32 CPU with 1MB memory;
                  8/32 supports 2 or 8 register banks
SELCH             selector channel (1-4)
PT                paper tape reader/punch
TT                console terminal, Teletype interface
TTP               console terminal, PASLA interface
LFC               line frequency clock
PIC               programmable interval clock
LPT               line printer
FD                floppy disk
DP                2.5MB/10MB cartridge disk with four disk drives
DM                mass storage module (MSM)/intelligent (IDC) disk
                   controller with four disk drives
MT                magnetic tape
PAS               programmable asynchronous line controller
PASL              programmable asynchronous lines, up to 32
```

The Interdata simulator implements two unique stop conditions:

- Decode of an undefined instruction, and STOP_INST is set
- Runaway carriage control tape in the line printer

The `LOAD` command is used to load a carriage control tape for the line printer. The `DUMP` command is used to dump a contiguous portion of memory as a self-loading bootstrap paper tape. The syntax for the `DUMP` command is:

```
DUMP <filename> lowaddr-highaddr
```

The low address must be greater than or equal to X'D0'.

Devices are assigned their default device numbers, as documented in the Interdata literature. Device numbers can be changed by the command:

```
SET <device> DEVNO=num
```

Device number conflicts are not checked until simulation starts. If there is a device number conflict, simulation stops immediately with an error message.

Selector channel devices are assigned by default to selector channel 0. Selector channel assignments can be changed by the command:

```
SET <dev> SELCH=num
```

Selector channel assignments cannot introduce conflicts.

Most devices can be disabled and enabled, with the commands:

```
SET <dev> DISABLED
SET <dev> ENABLED
```

All devices are enabled by default.

2.1 CPU (16b)

The CPU options include memory size and CPU type:

```
SET CPU I3                  Interdata 3 (base instruction set)
SET CPU I4                  Interdata 4 (base plus single precision
                            floating point)
SET CPU 716                 Interdata 7/16 (extended instruction set)
                            (equivalent to Models 5, 70, and 80)
SET CPU 816                 Interdata 8/16 (extended plus double
                            precision floating point)
SET CPU 816E                Interdata 8/16E (extended plus double
                            precision plus expanded memory)
SET CPU 8K                  set memory size = 8KB
SET CPU 16K                 set memory size = 16KB
SET CPU 24K                 set memory size = 24KB
SET CPU 32K                 set memory size = 32KB
SET CPU 48K                 set memory size = 48KB
SET CPU 64K                 set memory size = 64KB
SET CPU 128K                set memory size = 128KB (8/16E only)
SET CPU 256K                set memory size = 256KB (8/16E only)
SET CPU CONSINT             assert console interrupt (7/16, 8/16,
                             and 8/16E only)
```

If memory size is being reduced, and the memory being truncated contains non-zero data, the simulator asks for confirmation. Data in the truncated portion of memory is lost. Initial memory size is 64KB.

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

```
-a              examine/deposit ASCII characters
-b              examine/deposit bytes
-c              examine/deposit packed ASCII characters
-f              examine/deposit fullwords
-d              data radix is decimal
-o              data radix is octal
-h              data radix is hexadecimal
-m              examine as instruction mnemonics
-v              interpret address as virtual
```

Packed characters, halfwords, fullwords, and instructions must be aligned on a halfword (16b) boundary. If an odd address is specified, the low order bit is ignored.

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

```
name            size        comments

PC              16          program counter
R0..R15         16          general registers
FR0..F14        32          single precision floating point registers
D0H..D14H       32          double precision floating point registers,
                            high order
D0L..D14L       32          double precision floating point registers,
                            low order
PSW             16          processor status word
CC              4           condition codes, PSW<12:15>
SR              16          switch register
DR              32          display register low 16 bits
DRX             8           display register extension
```

```
DRMOD           1       display mode
DRPOS           2       display pointer position
SRPOS           1       switch pointer position
IRQ[0:3]        32      interrupt requests
IEN[0:3]        32      interrupt enables
STOP_INST       1       stop on undefined instruction
STOP_WAIT       1       stop if wait state and no I/O events pending
PCQ[0:63]       16      PC prior to last branch or interrupt;
                        most recent PC change first
WRU             8       interrupt character
```

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

```
SET CPU IDLE                enable idle detection
SET CPU NOIDLE              disable idle detection
```

Idle detection is disabled by default. The CPU is considered idle if the WAIT STATE flag is set in the PSW.

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

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

The maximum length for the history is 65536 entries.

2.2 CPU (32b)

The CPU options include memory size and CPU type:

```
SET CPU 732                 Interdata 7/32, single precision floating
                            point
SET CPU DPFP                Interdata 7/32, double precision floating
                            point
SET CPU 832                 Interdata 8/32 (double precision floating
                            point, 8 general register sets)
SET CPU 2RS                 Interdata 8/32 (double precision floating
                            point, 2 general register sets)
SET CPU 64K                 set memory size = 64KB
SET CPU 128K                set memory size = 128KB
SET CPU 256K                set memory size = 256KB
SET CPU 512K                set memory size = 512KB
SET CPU 1M                  set memory size = 1024KB
SET CPU CONSINT             assert console interrupt
```

If memory size is being reduced, and the memory being truncated contains non-zero data, the simulator asks for confirmation. Data in the truncated portion of memory is lost. Initial memory size is 1024KB.

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

```
-a              examine/deposit ASCII characters
```

```
-b              examine/deposit bytes
-c              examine/deposit packed ASCII characters
-w              examine/deposit halfwords
-d              data radix is decimal
-o              data radix is octal
-h              data radix is hexadecimal
-m              examine as instruction mnemonics
-v              interpret address as virtual
```

Packed characters, halfwords, fullwords, and instructions must be aligned on a halfword (16b) boundary. If an odd address is specified, the low order bit is ignored.

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

```
name            size        comments

PC              20          program counter
R0..R15         32          active general register set
GREG[32]        32          general register sets, 16 x 2
FR0..FR14       32          single precision floating point registers
D0H..D14H       32          double precision floating point registers,
                             high order
D0L..D14L       32          double precision floating point registers,
                             low order
PSW             16          processor status word
CC              4           condition codes, PSW<12:15>
SR              16          switch register
DR              32          display register low 16 bits
DRX             8           display register extension (x/16 only)
DRMOD           1           display mode
DRPOS           2           display pointer position
SRPOS           1           switch pointer position
MACREG[0:15]    32          memory access controller segment registers
MACSTA          5           memory access controller interrupt status
IRQ[0:3]        32          interrupt requests
IEN[0:3]        32          interrupt enables
STOP_INST       1           stop on undefined instruction
STOP_WAIT       1           stop if wait state and no I/O events pending
PCQ[0:63]       20          PC prior to last branch or interrupt;
                            most recent PC change first
WRU             8           interrupt character
```

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

```
SET CPU IDLE                enable idle detection
SET CPU NOIDLE              disable idle detection
```

Idle detection is disabled by default. The CPU is considered idle if the WAIT STATE flag is set in the PSW.

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

```
SET CPU HISTORY             clear history buffer
SET CPU HISTORY=0           disable history
```

```
SET CPU HISTORY=n             enable history, length = n
SHOW CPU HISTORY              print CPU history
SHOW CPU HISTORY=n            print first n entries of CPU history
```

The maximum length for the history is 65536 entries.

2.3 Selector Channel (SELCH0..SELCH3)

An Interdata system can have 1 to 4 selector channels (SELCH0, SELCH1, SELCH2, SELCH3). The default number of channels is 2. The number of channels can be changed with the command:

```
SET SELCH CHANNELS=num
```

All the state for a selector channel can be displayed with the command:

```
SHOW SELCH num
```

The selector channels implement these registers:

name	size	comments
SA[0:3]	20	start address, channels 0 to 3
EA[0:3]	20	end address, channels 0 to 3
CMD[0:3]	8	command, channels 0 to 3
DEV[0:3]	8	active device, channels 0 to 3
RDP[0:3]	2	read byte pointer, channels 0 to 3
WDC[0:3]	3	write data counter, channels 0 to 3
IREQ	4	interrupt requests; right to left, channels 0 to 3
IENB	4	interrupt enables; right to left, channels 0 to 3

2.4 Programmed I/O Devices

2.4.1 Paper Tape Reader/Punch (PT)

The paper tape reader and punch (PT units 0 and 1) read data from or write data to disk files. The RPOS and PPOS registers specify the number of the next data item to be read and written, respectively. Thus, by changing RPOS or PPOS, the user can backspace or advance these devices.

The paper tape reader supports the `BOOT` command. `BOOT PTR` copies the so-called '50 loader' into memory and starts it running.

The paper tape controller implements these registers:

name	size	comments
RBUF	8	reader buffer
RPOS	32	reader position in the input file
RTIME	24	time from reader start to interrupt
RSTOP_IOE	1	reader stop on I/O error
PBUF	8	punch buffer
PPOS	32	punch position in the output file
PTIME	24	time from punch start to interrupt

```
PSTOP_IOE         1           punch stop on I/O error
IREQ              1           paper tape interrupt request
IENB              1           paper tape interrupt enable
IARM              1           paper tape interrupt armed
RD                1           paper tape read/write mode
RUN               1           paper tape running
SLEW              1           paper tape reader slew mode
EOF               1           paper tape reader end of file
```

Error handling is as follows:

```
type        error          STOP_IOE  processed as

in,out      not attached      1      report error and stop
                              0      out of tape

in          end of file       1      report error and stop
                              0      out of tape

in,out      OS I/O error      x      report error and stop
```

2.4.2 Console, Teletype Interface (TT)

The Teletype keyboard (TT0) reads from the console keyboard; the Teletype printer (TT1) writes to the simulator console window. The Teletype units (TT0, TT1) can be set to one of four modes, KSR, 7P, 7B, or 8B:

```
mode  input characters        output characters

KSR   lower case converted    lower case converted to upper case,
       to upper case,         high-order bit cleared,
      high-order bit set      non-printing characters suppressed
7P    high-order bit cleared  high-order bit cleared,
                              non-printing characters suppressed
7B    high-order bit cleared  high-order bit cleared
8B    no changes              no changes
```

Changing the mode of either unit changes both. The default mode is KSR.

The Teletype has a BREAK key, which is not present on today's keyboards. To simulate pressing the break key, stop the simulator and use the command:

```
SET TT BREAK
```

Break status will be asserted, and will remain asserted for the interval specified by KTIME.

The Teletype interface implements these registers:

```
name              size        comments

KBUF              8           input buffer
KPOS              32          number of characters input
KTIME             24          input polling interval (if 0, the keyboard
                              is polled synchronously with the line clock)
TBUF              8           output buffer
TPOS              32          number of characters output
```

```
TTIME             24          time from output start to interrupt
IREQ              1           interrupt request
IENB              1           interrupt enable
IARM              1           interrupt armed
RD                1           read/write mode
FDPX              1           half-duplex
CHP               1           input character pending
```

2.4.3 Console, PASLA Interface (TTP)

Later Interdata system connect the system console via the first PASLA interface rather than the Teletype interface. The PASLA console can be simulated with a Telnet session on the first PAS line. Alternately, the PASLA console can be attached to the simulator console window, using the TTP device in place of TT.

To switch the simulator console window to TTP, use the command:

```
SET TTP ENABLED or
SET TT DISABLED
```

Device TT is automatically disabled and device TTP is enabled. To switch the simulator console window back to TT, use the command:

```
SET TT ENABLED or
SET TTP DISABLED
```

Device TTP is automatically disabled and device TT is enabled. If TTP is enabled at its default device settings, the base address for the PAS multiplexer must be changed:

```
SET PAS DEVNO=12
```

Otherwise, a device number conflict occurs.

The PASLA keyboard (TTP0) reads from the console keyboard; the PALSA printer (TTP1) writes to the simulator console window. The PASLA units (TTP0, TTP1) can be set to one of four modes, UC, 7P, 7B, or 8B:

```
mode  input characters        output characters

UC    lower case converted    lower case converted to upper case,
       to upper case,         high-order bit cleared,
      high-order bit cleared  non-printing characters suppressed
7P    high-order bit cleared  high-order bit cleared,
                              non-printing characters suppressed
7B    high-order bit cleared  high-order bit cleared
8B    no changes              no changes
```

Changing the mode of either unit changes both. The default mode is 7B.

To simulate pressing the break key, stop the simulator and use the command:

```
SET TTP BREAK
```

Break status will be asserted, and will remain asserted for the interval specified by KTIME.

The PASLA console interface implements these registers:

```
name            size        comments

CMD             16          command register
STA             8           status register
KBUF            8           input buffer
KPOS            32          number of characters input
KTIME           24          input polling interval (if 0, the keyboard
                            is polled synchronously with the line clock)
KIREQ           1           input interrupt request
KIENB           1           input interrupt enabled
KARM            1           input interrupt armed
CHP             1           input character pending
TBUF            8           output buffer
TPOS            32          number of characters output
TTIME           24          time from output start to interrupt
TIREQ           1           output interrupt request
TIENB           1           output interrupt enable
TIARM           1           output interrupt armed
```

2.4.4 Line Printer (LPT)

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

In addition, the line printer can be programmed with a carriage control tape. The `LOAD` command loads a new carriage control tape:

```
LOAD <file>                 load carriage control tape file
```

The format of a carriage control tape consists of multiple lines. Each line contains an optional repeat count, enclosed in parentheses, optionally followed by a series of column numbers separated by commas. Column numbers must be between 0 and 7; column seven is by convention top of form. The following are all legal carriage control specifications:

```
<blank line>                no punch
(5)                         5 lines with no punches
1,5,7                       columns 1, 5, 7 punched
(10)2                       10 lines with column 2 punched
0                           column 0 punched
```

The default form is 1 line long, with all columns punched.

The line printer implements these registers:

```
name            size        comments

BUF             7           last data item processed
BPTR            8           line buffer pointer
LBUF[0:131]     7           line buffer
VFUP            8           vertical forms unit pointer
VFUL            8           vertical forms unit length
VFUT[0:131]     8           vertical forms unit table
IREQ            1           line printer interrupt request
IENB            1           line printer interrupt enable
IARM            1           line printer interrupt armed
POS             32          position in the output file
```

```
CTIME             24          character processing time
STIME             24          spacing operation time
STOP_IOE          1           stop on I/O error
```

Error handling is as follows:

```
error         STOP_IOE        processed as

not attached      1           report error and stop
                  0           out of paper

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

2.4.5 Line Frequency Clock (LFC)

The line frequency clock (LFC) frequency can be adjusted as follows:

```
SET LFC 60HZ                  set frequency to 60Hz
SET LFC 50HZ                  set frequency to 50Hz
```

The default is 60Hz.

The line frequency clock implements these registers:

```
name              size        comments

IREQ              1           clock interrupt request
IENB              1           clock interrupt enable
IARM              1           clock interrupt armed
TIME              24          clock frequency
```

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

2.4.6 Programmable Interval Clock (PIC)

The programmable interval clock (PIC) implements these registers:

```
name              size        comments

BUF               16          output buffer
RIC               16          reset interval and rate
CIC               12          current interval
DECR              10          current decrement value
RDP               1           read byte select
OVF               1           interval overflow flag
IREQ              1           clock interrupt request
IENB              1           clock interrupt enable
IARM              1           clock interrupt armed
```

If the interval requested is an exact multiple of 1 millisecond, the programmable clock auto-calibrates; if not, it counts instructions.

2.4.7 Floppy Disk Controller (FD)

Floppy disk options include the ability to make units write enabled or write locked.

```
SET FDn LOCKED                set unit n write locked
SET FDn WRITEENABLED          set unit n write enabled
```

Units can also be set ENABLED or DISABLED.

The floppy disk supports the BOOT command. BOOT FDn copies an autoload sequence into memory and starts it running.

The floppy disk controller implements these registers:

```
name              size        comments

CMD               8           command
STA               8           status
BUF               8           buffer
LRN               16          logical record number
ESTA[0:5]         8           extended status bytes
DBUF[0:127]       8           transfer buffer
DBPTR             8           transfer buffer pointer
IREQ              1           interrupt request
IENB              1           interrupt enabled
IARM              1           interrupt armed
CTIME             24          command response time
STIME             24          seek time, per cylinder
XTIME             24          transfer time, per byte
STOP_IOE          1           stop on I/O error
```

Error handling is as follows:

```
error          STOP_IOE       processed as

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

Floppy disk data is buffered in memory; therefore, end of file and OS I/O errors cannot occur.

2.4.8 Programmable Asynchronous Line Adapters (PAS, PASL)

The Programmable Asynchronous Line Adapters (PAS and PASL) represent, indistinguishably, individual PASLA interfaces, 2 line asynchronous multiplexers, and 8 line asynchronous multiplexers, with a maximum of 32 lines. All the lines are modeled as a terminal multiplexer, with PAS as the multiplexer controller, and PASL as the individual lines. The PASLAs perform input and output through Telnet sessions connected to a user-specified port. The ATTACH command specifies the port to be used:

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

where port is a decimal number between 1 and 65535 that is not being used for other TCP/IP activities.

Each line (each unit of PASL) can be set to one of four modes, UC, 7P, 7B, or 8B:

```
mode  input characters        output characters
```

```
UC    lower case converted    lower case converted to upper case,
       to upper case,         high-order bit cleared,
      high-order bit cleared  non-printing characters suppressed
7P    high-order bit cleared  high-order bit cleared,
                              non-printing characters suppressed
7B    high-order bit cleared  high-order bit cleared
8B    no changes              no changes
```

Each line (each unit of PASL) can also be set for modem control with the command `SET PASLn DATASET`. The defaults are 7b mode and DATASET disabled. Finally, each line supports output logging. The `SET PASLn LOG` command enables logging on a line:

```
SET PASLn LOG=filename        log output of line n to filename
```

The `SET PASLn NOLOG` command disables logging and closes the open log file, if any.

Once PAS is attached and the simulator is running, the terminals listen for connections on the specified port. They assume that the incoming connections are Telnet connections. The connections remain open until disconnected either by the Telnet client, a `SET PAS DISCONNECT` command, or a `DETACH PAS` command.

Other special PASLA commands:

```
SHOW PAS CONNECTIONS          show current connections
SHOW PAS STATISTICS           show statistics for active connections
SET PASLn DISCONNECT          disconnects the specified line.
```

The controller (PAS) implements these registers:

```
name              size  comments

STA[0:31]         8           status, lines 0 to 31
CMD[0:31]         16          command, lines 0 to 31
RBUF[0:31]        8           receive buffer, lines 0 to 31
XBUF[0:31]        8           transmit buffer, lines 0 to 31
RIREQ             32          receive interrupt requests;
                              right to left, lines 0 to 31
RIENB             32          receive interrupt enables
RARM[0:31]        1           receive interrupt armed
XIREQ             32          transmit interrupt requests;
                              right to left, lines 0 to 31
XIENB             32          transmit interrupt enables
XARM[0:31]        1           transmit interrupt armed
RCHP[0:31]        1           receiver character present, lines 0 to 31
```

The lines (PASL) implements these registers:

```
name              size        comments

TIME[0:31]        24          transmit time, lines 0 to 31
```

The additional terminals do not support save and restore. All open connections are lost when the simulator shuts down or PAS is detached.

2.5 Cartridge Disk Controller (DP)

Cartridge disk options include the ability to make units write enabled or write locked, and to select the type of drive:

```
SET DPn LOCKED                set unit n write locked
SET DPn WRITEENABLED          set unit n write enabled
SET DPn 2315                  set unit n to 2315 (2.5MB)
SET DPn 5440                  set unit n to 5440 (10MB)
```

Units can also be set `ENABLED` or `DISABLED`.

The cartridge disk supports the `BOOT` command. To boot OS16/32, the hex form of the operating system file's extension must be placed in locations 7E:7F. The disk bootstrap looks for a valid OS16/32 volume descriptor in block 0, and uses that to locate the volume directory. It then searches the directory for a filename of the form OS16xxxx.hhh or OS32xxxx.hhh, where the xxxx is ignored and hhh is the ASCII form of the extension from locations 7E:7F. The 32b bootstrap can also boot Wollongong UNIX; locations 7E:7F must be 0. The bootstrap normally boots from the first (removable) platter in a 5440; to boot from the second (fixed) platter, use `BOOT -F`.

All drives have 256 8b bytes per sector. The other disk parameters are:

```
drive cylinders surfaces sectors

2315     203        2      24
5440     408        4      12
```

The cartridge disk controller implements these registers:

```
name              size        comments

CMD               3           current command
STA               8           controller status
BUF               8           controller buffer
HDSC              8           current head/sector select
CYL               8           current cylinder select
DBUF[0:255]       8           transfer buffer
DBPTR             16          transfer buffer point
DBLNT             16          transfer buffer length
FIRST             1           first DMA service flag
IREQ              5           interrupt requests; right-to-left,
                              controller, drives 0 to 3
IENB              5           interrupt enables
IARM[0:3]         1           interrupts armed, drives 0 to 3
STIME             24          seek latency, per cylinder
RTIME             24          rotational latency, per sector
WTIME             24          inter-word latency
```

Error handling is as follows:

```
error             processed as

not attached      disk not ready

end of file       assume rest of disk is zero
```

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

2.6 Mass Storage Module/Intelligent Disk Controller (DM)

MSM/IDC disk controller options include the ability to make units write enabled or write locked, and to select the type of drive:

```
SET DMn LOCKED                set unit n write locked
SET DMn WRITEENABLED          set unit n write enabled
SET DMn MSM80                 set unit n to storage module, 80MB
                              (67MB formatted)
SET DMn MSM300                set unit n to storage module, 300MB
                              (262MB formatted)
SET DMn MCCD16                set unit n to medium capacity, 16MB
                              (13.5MB formatted)
SET DMn MCCD48                set unit n to medium capacity, 48MB
                              (40.5MB formatted)
SET DMn MCCD80                set unit n to medium capacity, 80MB
                              (67MB formatted)
SET DMn MSM330F               set unit n to storage module, 330MB
                              (300MB formatted)
```

Units can also be set `ENABLED` or `DISABLED`.

The MSM/IDC controller supports the `BOOT` command. To boot OS16/32, the hex form of the operating system file's extension must be placed in locations 7E:7F. The disk bootstrap looks for a valid OS16/32 volume descriptor in block 0, and uses that to locate the volume directory. It then searches the directory for a filename of the form OS16xxxx.hhh or OS32xxxx.hhh, where the xxxx is ignored and hhh is the ASCII form of the extension from locations 7E:7F. The 32b bootstrap can also boot Wollongong UNIX; locations 7E:7F must be 0. Note that only the MSM80 and MSM300 drives can be bootstrapped; the boot code does not recognize the other drives.

All drives have 256 8b bytes per sector. The other disk parameters are:

```
Drive  cylinders surfaces sectors

MSM80      823      5      64
MSM300     823     19      64
MCCD16     823      1      64
MCCD48     823      3      64
MCCD80     823      5      64
MSM300F   1024     16      64
```

The MSM/IDC disk controller implements these registers:

```
name              size        comments

STA               8           controller status
BUF               8           controller buffer
SEC               8           current sector select
DBUF[0:767]       8           transfer buffer
DBPTR             16          transfer buffer point
DBLNT             16          transfer buffer length
FIRST             1           first DMA service flag
```

```
CWDPTR          2       controller write data byte pointer
DWDPTR          1       drive write data byte pointer
IREQ            5       interrupt requests; right-to-left,
                        controller, drives 0 to 3
IENB            5       interrupt enables
SIREQ           5       saved interrupt requests
ICARM           1       controller interrupt armed
IDARM[0:3]      1       drive interrupts armed, drives 0 to 3
STIME           24      seek latency, per cylinder
RTIME           24      rotational latency, per sector
WTIME           24      inter-word latency
```

Error handling is as follows:

```
error           processed as

not attached    disk not ready

end of file     assume rest of disk is zero

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

2.7 Magnetic Tape Controller (MT)

Magnetic tape options include the ability to make units write enabled or write locked.

```
SET MTn LOCKED              set unit n write locked
SET MTn WRITEENABLED        set unit n write enabled
```

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

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

Units can also be set `ENABLED` or `DISABLED`.

The magnetic tape supports the `BOOT` command. `BOOT MTn` copies an autoload sequence into memory and starts it running.

The magnetic tape controller implements these registers:

```
name            size        comments

CMD             8           command
STA             8           status
BUF             8           buffer
DBUF[0:65535]   8           transfer buffer
DBPTR           16          transfer buffer pointer
DBLNT           16          transfer buffer length
XFR             1           transfer in progress flag
FIRST           1           first DMA service flag
IREQ            4           interrupt requests; right to left,
                            drives 0 to 3
IENB            4           interrupt enables
IARM[0:3]       1           interrupts armed, drives 0 to 3
STOP_IOE        1           stop on I/O error
```

```
WTIME           1          word transfer time
RTIME           1          interrecord latency
UST[0:3]        8          unit status, drives 0 to 3
POS[0:3]        32         tape position, drives 0 to 3
```

Error handling is as follows:

```
error           processed as

not attached    tape not ready; if STOP_IOE, stop

end of file     set error flag

OS I/O error    set error flag; if STOP_IOE, stop
```

3 Symbolic Display and Input

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

```
-a              display byte as ASCII character
-c              display halfword as two packed ASCII characters
-m              display instruction mnemonics
```

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

```
' or -a         ASCII character
" or -c         two packed ASCII characters
alphabetic      instruction mnemonic
numeric         hexadecimal number
```

3.1 16b Instruction Input

Instruction input uses standard Interdata assembler syntax. There are seven instruction classes: short branch, extended short branch, short immediate, register, register-register, memory, and register-memory.

Short branch instructions have the format

```
sbop mask,address
```

where the mask is a hex (decimal) number between 0 and F (15), and the address is within +32 (forward branch) or -32 (backward branch) of the current location.

Extended short branch instructions have the format

```
sbxop address
```

where the address is within +32 or -32 of the current location. For extended short branches, the simulator chooses the forward or backward direction automatically.

Short immediate instructions have the format

```
siop regnum,immed
```

where the register number is a hex (decimal) number, optionally preceded by R, between 0 and F (15), and the immediate is a hex digit between 0 and F.

Register instructions have the format

```
rop regnum
```

where the register number is a hex (decimal) number, optionally preceded by R, between 0 and F (15).

Register-register instructions have the format

```
rrop regnum,regnum
```

where the register numbers are hex (decimal) numbers, optionally preceded by R, between 0 and F (15).

Memory instructions have the format

```
mop address{(index)}
```

where address is a hex number between 0 and 0xFFFF, and the index register is a hex (decimal) number, optionally preceded by R, between 1 and F (15).

Register-memory instructions have the format

```
rmop regnum,address{(index)}
```

where the register number is a hex (decimal) number, optionally preceded by R, between 0 and F (15), the address is a hex number between 0 and 0xFFFF, and the index register is a hex (decimal) number, optionally preceded by R, between 1 and F (15).

3.2 32b Instruction Input

Instruction input uses standard Interdata assembler syntax. There are nine instruction classes: short branch, extended short branch, short immediate, 16b immediate, 32b immediate, register, register-register, memory, and register-memory. Addresses, where required, can be specified as either absolute numbers or relative to the current location (.+n or .-n).

Short branch instructions have the format

```
sbop mask,address
```

where the mask is a hex (decimal) number between 0 and F (15), and the address is within +32 (forward branch) or -32 (backward branch) of the current location.

Extended short branch instructions have the format

```
sbxop address
```

where the address is within +32 or -32 of the current location. For extended short branches, the simulator chooses the forward or backward direction automatically.

Short immediate instructions have the format

```
siop regnum,immed
```

where the register number is a hex (decimal) number, optionally preceded by R, between 0 and F (15), and the immediate is a hex digit between 0 and F.

16b immediate instructions have the format

```
i16op regnum,immed16{(index)}
```

where the register number is a hex (decimal) number, optionally preceded by R, between 0 and F (15), the immediate is a hex number between 0 and 0xFFFF, and the index register is a hex (decimal) number, optionally preceded by R, between 1 and F (15).

32b immediate instructions have the format

```
i32op regnum,immed32{(index)}
```

where the register number is a hex (decimal) number, optionally preceded by R, between 0 and F (15), the immediate is a hex number between 0 and 0xFFFFFFFF, and the index register is a hex (decimal) number, optionally preceded by R, between 1 and F (15).

Register instructions have the format

```
rop regnum
```

where the register number is a hex (decimal) number, optionally preceded by R, between 0 and F (15).

Register-register instructions have the format

```
rrop regnum,regnum
```

where the register numbers are hex (decimal) numbers, optionally preceded by R, between 0 and F (15).

Memory instructions have the format

```
mop address{(index)} or
mop address{(index1,index2)}
```

where address is a hex number between 0 and 0xFFFF, and the index registers are hex (decimal) numbers, optionally preceded by R, between 1 and F (15).

Register-memory instructions have the format

```
rmop regnum,address{(index)} or
rmop regnum,address{(index1,index2)}
```

where the register number is a hex (decimal) number, optionally preceded by R, between 0 and F (15), the address is a hex number between 0 and 0xFFFF, and the index registers are hex (decimal) numbers, optionally preceded by R, between 1 and F (15).

For memory operands, the simulator automatically chooses the format (RX1, RX2, RX3) that consumes the fewest bytes. If both RX1 and RX2 are feasible, the simulator chooses RX1.