H316 Simulator Usage

Robert M Supnik

30-Apr-2020

Contents

This memorandum documents the Honeywell H316/H516 simulator.

Simulator Files

The H316 requires the following 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/h316/
  - h316_defs.h
  - h316_cpu.c
  - h316_dp.c
  - h316_fhd.c
  - h316_hi.c
  - h316_imp.c
  - h316_lp.c
  - h316_mi.c
  - h316_mt.c
  - h316_rtc.c
  - h316_stddev.c
  - h316_sys.c
  - h316_udp.c

H316/H516 Features

The Honeywell 316/516 simulator is configured as follows:

device names	simulates
CPU	H316/H516 CPU with 16/32KW memory
PTR	316/516-50 paper tape reader
PTP	316/516-52 paper tape punch
LPT	316/516 line printer
TTY	316/516-33 console terminal
MT	4100 seven track magnetic tape with four drives
CLK	316/516-12 real time clock
FHD	4400 fixed head disk
DP	4623/4653/4720 disk pack controller with eight drives
WDT	4400 fixed head disk
RTC	4400 fixed head disk
IMP	IM/TIP Specific Hardware

device names	simulates
MI1	IM/TIP Modem Interface
MI2	IM/TIP Modem Interface
MI3	IM/TIP Modem Interface
MI4	IM/TIP Modem Interface
MI5	IM/TIP Modem Interface
HI1	IMP Host Interface
HI2	IMP Host Interface
HI3	IMP Host Interface
HI4	IMP Host Interface

The H316/H516 simulator implements several unique stop conditions:

- Decode of an undefined instruction, and STOP_INST is set
- Reference to an undefined I/O device, and STOP_DEV is set
- More than INDMAX indirect references are detected during memory reference address decoding
- DMA/DMC direction does not agree with I/O device operation
- A write operation is initiated on a write locked magnetic tape unit (hangs the real system)
- A disk write overruns the specified record size (destroys the rest of the track on the real system)
- A disk track has an illegal format

The LOAD and DUMP commands are not implemented.

CPU

CPU options include choice of instruction set, memory size, DMC option, and number of DMA channels.

Command	Function
SET CPU HSA	high speed arithmetic instructions
SET CPU NOHSA	no high speed arithmetic instructions
SET CPU 4K	set memory size = 4K
SET CPU 8K	set memory size = 8K
SET CPU 12K	set memory size = 12K
SET CPU 16K	set memory size = 16K
SET CPU 24K	set memory size = 24K
SET CPU 32K	set memory size = 32K
SET CPU DMC	enable DMC option
SET CPU NODMC	disable DMC option

Command	Function
SET CPU DMA=n	set number of DMA channels to n (0-4)

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 32K. By default, the HSA and DMC options are enabled, and four DMA channels are configured.

Extended Interrupts

The H316 came with one interrupt vector and 16 individually maskable interrupt sources as standard but could optionally be extended to support up to 48 additional individually maskable interrupt sources, each with its own unique vector. Extended interrupts are enabled with the command

SET CPU EXTINT=16

and the command

SET CPU EXTINT=0

restores the original default H316 single interrupt behavior. Note that the IMP and TIP custom hardware required one additional bank of 16 interrupts.

Only 16 extended interrupts (out of a possible 48) are currently implemented.

DMA channels

The CPU includes special show commands to display the state of the DMA channels:

SHOW CPU DMAn show DMA channel n

Break on Write

The H316 emulation supports "break on memory write" breakpoints in addition to the standard "break on execution" type. For example, the command

BREAK –W 2000

will cause a break to occur any time memory location 2000_8 is written. All the usual simh break point options, including address ranges and commands to be executed automatically upon breaking, work here too.

Break on write has two restrictions – first, setting a write break on a DMC channel pointer location, 20_8 thru 57_8, will break only on explicit programmed writes to that address. Implicit DMC operations will not cause a break. Second, remember that when *any* break occurs simh prints the PC and the instruction *after* the one which actually caused the break. It will always be the previous instruction which actually modified the breakpoint location.

CPU State

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

name	size	comments
P	15	program counter
A	16	A register
B	16	B register
X	16	index register
SC	16	shift count
C	1	carry flag
EXT	1	extend flag
PME	1	previous mode extend flag
EXT_OFF	1	extend off pending flag
DP	1	double precision flag
SS1..4	1	sense switches 1 to 4
ION	1	interrupts enabled
INODEF	1	interrupts not deferred
INTREQ	16	interrupt requests
EXTINT	16	extended interrupt requests
EXTENB	16	extended interrupt enables
DEVRDY	16	device ready flags (read only)
DEVENB	16	device interrupt enable flags (read only)
CHREQ	20	DMA/DMC channel requests
DMAAD[0:3]	16	DMA channel current address, channels 1 to 4
DMAWC[0:3]	15	DMA channel word count, channels 1 to 4
DMAEOR[0:3]	1	DMA end of range flag, channels 1 to 4
STOP_INST	1	stop on undefined instruction
STOP_DEV	1	stop on undefined device
INDMAX	8	indirect address limit
PCQ[0:63]	15	PC prior to last JMP, JSB, or interrupt;most recent PC change first
WRU	8	interrupt character

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 65,536 entries.

Programmed I/O Devices

316/516-50 Paper Tape Reader (PTR)

The paper tape reader (PTR) reads data from a disk file. The POS register specifies the number of the next data item to be read. Thus, by changing POS, the user can backspace or advance the reader.

The paper tape reader can bet set to operate in binary, ASCII, or Unix ASCII mode:

```
SET PTR BINARY binary mode
SET PTR ASCII ASCII mode
SET PTR UASCII Unix ASCII mode
```

The mode can also be set by a switch setting in the ATTACH command:

```
ATT --B PTR <file> binary mode
ATT --A PTR <file> ASCII mode
ATT --U PTR <file> Unix ASCII mode
```

In ASCII or Unix ASCII mode, all non-zero characters have the high order bit forced on. In Unix ASCII mode, newline is converted to CR, and LF is inserted as the following character.

The paper tape reader supports the BOOT command. BOOT PTR copies the absolute binary loader into memory and starts it running.

The paper tape reader implements these registers:

name	size	comments
BUF	8	last data item processed
INTREQ	1	device interrupt request
READY	1	device ready
ENABLE	1	device interrupts enabled
POS	32	position in the input file
TIME	24	time from I/O initiation to interrupt
STOP_IOE	1	stop on I/O error

Error handling is as follows:

error	STOP_IOE	processed as
not attached	1	report error and stop
0	out of tape	
end of file	1	report error and stop
0	out of tape	

error	STOP_IOE	processed as
OS I/O error	x	report error and stop

316/516-52 Paper Tape Punch (PTP)

The paper tape punch (PTP) 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 punch. The default position after ATTACH is to position at the end of an existing file. A new file can be created if you attach with the -N switch.

The paper tape punch can bet set to operate in binary, ASCII, or Unix ASCII mode:

```
SET PTP BINARY          binary mode
SET PTP ASCII           ASCII mode
SET PTP UASCII          Unix ASCII mode
```

The mode can also be set by a switch setting in the ATTACH command:

```
ATT --B PTP <file>    binary mode
ATT --A PTP <file>    ASCII mode
ATT --U PTP <file>    Unix ASCII mode
```

In ASCII or Unix ASCII mode, all characters are masked to 7b before being written to the output file. In Unix ASCII mode, LF is converted to newline, and CR is discarded.

The paper tape punch implements these registers:

name	size	comments
BUF	8	last data item processed
INTREQ	1	device interrupt request
READY	1	device ready
ENABLE	1	device interrupts enabled
POWER	1	device powered up
POS	32	position in the output file
TIME	24	time from I/O initiation to interrupt
PWRTIME	24	time from I/O request to power up
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 tape	

error	STOP_IOE	processed as
OS I/O error	x	report error and stop

316/516-33 Console Teletype (TTY)

The console Teletype (TTY) consists of four separate units:

```
TTY0                    keyboard
TTY1                    printer
TTY2                    paper tape reader
TTY3                    paper tape punch
```

The keyboard and printer (TTY0, TTY1) can be set to one of four modes, KSR, 7P, 7B, or 8B:

mode	input characters	output characters
KSR	lower case converted to upper case, high-order bit set	lower case converted to upper case, 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

The default mode is KSR. The Teletype keyboard reads from the console keyboard, and the printer writes to the simulator console window.

The paper tape reader (TTY2) can bet set to operate in binary, ASCII, or Unix ASCII mode:

```
SET TTY2 BINARY         binary mode
SET TTY2 ASCII          ASCII mode
SET TTY2 UASCII         Unix ASCII mode
```

The mode can also be set by a switch setting in the ATTACH command:

```
ATT --B TTY2 <file>   binary mode
ATT --A TTY2 <file>   ASCII mode
ATT --U TTY2 <file>   Unix ASCII mode
```

In ASCII or Unix ASCII mode, all non-zero characters have the high order bit forced on. In Unix ASCII mode, newline is converted to CR, and LF is inserted as the following character.

The paper tape reader is started by program output of XON or by the command SET TTY2 START. The paper tape reader is stopped by reader input of XOFF or by the command SET TTY2 STOP.

The Teletype paper tape punch (TTY3) can bet set to operate in binary, ASCII, or Unix ASCII mode:

```
SET TTY3 BINARY         binary mode
SET TTY3 ASCII          ASCII mode
SET TTY3 UASCII         Unix ASCII mode
```

The mode can also be set by a switch setting in the ATTACH command:

```
ATT --B TTY3 <file>       binary mode
ATT --A TTY3 <file>       ASCII mode
ATT --U TTY3 <file>       Unix ASCII mode
```

In ASCII or Unix ASCII mode, all characters are masked to 7b before being written to the output file. In Unix ASCII mode, LF is converted to newline, and CR is discarded.

The Teletype paper tape punch is started by program output of TAPE or by the command SET TTY3 START. The punch is stopped by program output of XOFF or by the command SET TTY3 STOP.

The TTY implements these registers:

name	size	comments
BUF	8	last data item processed
IN2ND	9	holding buffer, input busy wait; the high order bit indicates character present
MODE	1	read/write mode
READY	1	device ready flag
BUSY	1	device busy flag
INT	1	device interrupt request
ENABLE	1	device interrupt enabled
KPOS	32	number of keyboard characters input
KTIME	24	keyboard polling interval
KBTIME	24	keyboard busy wait after receive
TPOS	32	number of printer characters output
TTIME	24	time from I/O initiation to interrupt
RPOS	32	current reader character position
PPOS	32	current punch character position

316/516-12 Real Time Clock (CLK)

The real time clock (CLK) frequency can be adjusted as follows:

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

The default is 60Hz.

The clock implements these registers:

name	size	comments
INTREQ	1	device interrupt request
READY	1	device ready
ENABLE	1	device interrupts enabled
TIME	24	clock interval

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

Note that previous releases of simh did not allow the CLK device to be disabled. However, this device was optional and it was possible to configure an H316 system without one (although this apparently rarely happened). Current simh releases will allow the CLK device to be disabled. When the CLK device is disabled it does not respond to the clock specific IO instructions; it does not increment location 61_8, and it does not generate interrupts. The SMK and OTK instructions are unaffected.

316/516 Line Printer (LPT)

The line printer (LPT) writes data to a disk file. The POS register specifies the number of the next data item to be written. Thus, by changing POS, the user can backspace or advance the printer. The default position after ATTACH is to position at the end of an existing file. A new file can be created if you attach with the -N switch.

The line printer can be connected to the IO bus, a DMC channel, or a DMA channel:

```
SET LPT IOBUS           connect to IO bus
SET LPT DMC=n           connect to DMC channel n (1-16)
SET LPT DMA=n           connect to DMA channel n (1 to 4)
```

By default, the line printer is connected to the IO bus.

The line printer implements these registers:

name	size	comments
WDPOS	6	word position in current scan
DRPOS	6	drum position
CRPOS	1	carriage position
PRDN	1	print done flag
RDY	1	ready flag
EOR	1	(DMA/DMC) end of range flag
DMA	1	transfer using DMA/DMC

name	size	comments
INTREQ	1	device interrupt request
ENABLE	1	device interrupt enable
SVCST	2	service state
SVCCH	2	service channel
BUF	8	buffer
POS	32	position in the output file
XTIME	24	delay between transfers
ETIME	24	delay at end of scan
PTIME	24	delay for shuttle/line advance
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

4400 Fixed Head Disk (FHD)

Fixed head disk options include the ability to set the number of surfaces to a fixed value between 1 and 16, or to autosize the number of surfaces from the attached file:

```
SET FHD 1S              one surface (98K)
SET FHD 2S              two platters (196K)
:
SET FHD 16S             sixteen surfaces (1568K)
SET FHD AUTOSIZE        autosized on ATTACH
```

The default is one surface.

The fixed head disk can be connected to the IO bus, a DMC channel, or a DMA channel:

```
SET FHD IOBUS           connect to IO bus
SET FHD DMC=n           connect to DMC channel n (1-16)
SET FHD DMA=n           connect to DMA channel n (1 to 4)
```

By default, the fixed head disk is connected to the IO bus.

The fixed head disk implements these registers:

name	size	comments
CW1	16	control word 1 (read write, surface, track)
CW2	16	control word 2 (character address)
BUF	16	data buffer
BUSY	1	controller busy flag
RDY	1	transfer ready flag
DTE	1	data transfer error flag
ACE	1	access error flag
EOR	1	(DMA/DMC) end of range
DMA	1	transfer using DMA/DMC
CSUM	1	transfer parity checksum
INTREQ	1	device interrupt request
ENABLE	1	device interrupt enable
TIME	24	delay between words
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	

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

4100 7-track Magnetic Tape (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 controller can be connected to the IO bus, a DMC channel, or a DMA channel:

```
SET MT IOBUS            connect to IO bus
SET MT DMC=n            connect to DMC channel n (1-16)
```

```
SET MT DMA=n            connect to DMA channel n (1 to 4)
```

By default, the magnetic tape controller is connected to the IO bus.

The magnetic tape controller implements these registers:

name	size	comments
BUF	16	data buffer
USEL	2	unit select
BUSY	1	controller busy flag
RDY	1	transfer ready flag
ERR	1	error flag
EOF	1	end of file flag
EOR	1	(DMA/DMC) end of range
DMA	1	transfer using DMA/DMC
MDIRQ	1	motion done interrupt request
INTREQ	1	device interrupt request
ENABLE	1	device interrupt enable
DBUF[0:65535]	8	transfer buffer
BPTR	17	transfer buffer pointer
BMAX	17	transfer size (reads)
CTIME	24	start/stop time
XTIME	24	delay between words
POS[0:3]	32	position, units 0 to 3
STOP_IOE	1	stop on I/O error

Error handling is as follows:

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

4623/4651/4720 Disk Packs (DP)

The disk controller can be configured as a 4623, supporting 10-surface disk packs; a 4651, supporting 2-surface disk packs; or a 4720, supporting 20-surface disk packs:

```
SET DP 4623             controller is 4623
SET DP 4651             controller is 4651
SET DP 4720             controller is 4720
```

The default is 4651. All disk packs on the controller must be of the same type.

Individual units can be write enabled or write locked:

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

Units can be also be set ENABLED or DISABLED.

The disk pack controller can be connected to a DMC channel or a DMA channel; it cannot be connected to the IO bus:

```
SET DP DMC=n            connect to DMC channel n (1-16)
SET DP DMA=n            connect to DMA channel n (1 to 4)
```

The disk pack controller supports variable track formatting. Each track can contain between 1 and 103 records, with a minimum size of 1 word and a maximum size of 1893 words. Record addresses are unconstrained. The simulator provides a command to perform a simple, fixed record size format of a new disk:

```
SET DPn FORMAT=k        format unit n with k words per record
SET -R DPn FORMAT=k     format unit n with k records per track
```

Record addresses can either be geometric (cylinder/track/sector) or simple sequential starting from 0:

```
SET DPn FORMAT=k        format with geometric record addresses
SET -S DPn FORMAT=k     format with sequential record addresses
```

Geometric address have the cylinder number in bits<1:8>, the head number in bits<9:13>, and the sector number in bits <14:16>.

A summary of the current format, and its validity, can be obtained with the command:

```
SHOW DPn FORMAT         display format of unit n
```

To accommodate the variable formatting, each track is allocated 2048 words in the data file. A record consists of a three word header, the data, and a five word trailer:

```
word 0              record length in words, not including header/trailer
word 1              record address
word 2              number of extension words used (0-4)
word 3              start of data record
word 3+n-1          end of data record
word 3+n..7+n       record trailer: up to four extension words,
                    plus checksum
```

A record can "grow" by up to four words without disrupting the track formatting; writing more than four extra words destroys the formatting of the rest of the track and causes a simulator error.

The disk pack controller implements these registers:

name	size	comments
STA	16	status
BUF	16	data buffer
FNC	4	controller function
CW1	16	command word 1
CW2	16	command word 2
CSUM	16	record checksum
BUSY	1	controller busy
RDY	1	transfer ready
EOR	1	(DMA/DMC) end of range
DEFINT	1	seek deferred interrupt pending
INTREQ	1	interrupt request
ENABLE	1	interrupt enable
TBUF[0:2047]	16	track buffer
RPTR	11	pointer to start of record in track buffer
WPTR	11	pointer to current word in record
BCTR	15	bit counter for formatting
STIME	24	seek time, per cylinder
XTIME	24	transfer time, per word
BTIME	24	controller busy time

Error handling is as follows:

error	processed as
not atached	pack off line; if STOP_IOE, stop
end of file	ignored
OS I/O error	data error; if STOP_IOE, stop

Symbolic Display and Input

The H316/H516 simulator implements symbolic display and input. Display is controlled by command line switches:

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

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

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

Instruction input uses standard H316/H516 assembler syntax. There are six instruction classes: memory reference, I/O, control, shift, skip, and operate.

Memory reference instructions have the format

```
memref{*} {C/Z} address{,1}
```

where * signifies indirect, C a current sector reference, Z a sector zero reference, and 1 indexed. The address is an octal number in the range 0 - 077777; if C or Z is specified, the address is a page offset in the range 0 - 0777. Normally, C is not needed; the simulator figures out from the address what mode to use. However, when referencing memory outside the CPU, there is no valid PC, and C must be used to specify current sector addressing.

I/O instructions have the format

```
io function,device
```

The function is an octal number in the range 0 - 17. The device is a symbolic name (e.g., TTY) or an octal number in the range 0-77.

Control and operate instructions consist of a single opcode

```
opcode
```

Shift instructions have the format

```
shift n
```

where n is an octal number in the range 0-77.

Skip instructions have the format

```
sub-op sub-op sub-op...
```

The simulator checks that the combination of sub-opcodes is legal.

COPYRIGHT NOTICE and LICENSE

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

CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Except as contained in this notice, the name of Robert M Supnik shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization from Robert M Supnik. “‘