ARPANET IMP and TIP Extensions for the H316 simulator

Robert M Supnik Robert Armstrong, bob@jfcl.com

02-Dec-2013

Copyright © 1993-2012, Robert M Supnik
Copyright © 2013, Robert Armstrong, bob@jfcl.com

COPYRIGHT NOTICE and LICENSE are at the end of this document.

Contents

  - Host Interface I/O Instructions
  - Other I/O Instructions
- COPYRIGHT NOTICE and LICENSE

Introduction

This memorandum documents the extensions made to the simh Honeywell H316 simulator to allow it to run the ARPANET Interface Message Processor (aka IMP) and Terminal Interface Processor (TIP) software.

A single IMP or TIP instance isn't very useful, however these extensions allow multiple simh instances, each running a copy of the IMP or TIP code, to communicate using virtual modem connections implemented with physical serial ports on the host computer.

Alternatively, virtual modem connections may be tunneled over TCP/IP to a remote simh IMP/TIP instance anywhere in the world.

Moreover, each simh IMP instance can then be connected via virtual host interface cards to other local PDP-10 or PDP-11 simh instances running ARPANET host software.

Simulator Files

These additional IMP/TIP specific files are added to simh:

Subdirectory	***File***	***Contains***
h316	h316_imp.h	IMP/TIP and ARPAnet specific definitions
	h316_imp.c	the IMP pseudo device
	h316_rtc.c	real time clock and watch dog timer
	h316_hi.c	host interfaces
	h316_mi.c	modem interfaces
	h316_udp.c	UDP support for MI and HI modules

Additional Features and Devices

The IMP/TIP adds the following devices to the H316 configuration:

device name	simulates
MI	modem interface (up to 5)
HI	host interface (up to 4)
WDT	watchdog timer
RTC	real time clock (replaces H316 CLK device)

device name	simulates
IMP	TASK, IMPN and MLC functions (addresses 41₈ and 42₈)

IMP Pseudo Device

The IMP pseudo device implements two sets of miscellaneous I/O instructions in the original IMP/TIP hardware. This device responds both to IO address 41_8, which implements the TASK switching interrupt and the IMP identification number, and to IO address 42_8, which implements the MLC test. The IMP device may be enabled with the standard command

```
SET IMP ENABLED
```

The IMP address, which is returned by the RDIMPN (INA 1041) instruction, may be set with the command

```
SET IMP NUM=n                set IMP station address to n
```

This value was apparently hardwired into each original IMP/TIP station. The IMP device also implements the AMIMLC (SKS 0042) instruction, which skips if the current hardware is a multi-line controller (aka a TIP). Since TIP emulation is not currently supported, this instruction is presently hard wired as a NOP (i.e. it never skips).

Registers

The IMP device implements the following registers:

name	size	comments
MLC	1	always zero (TIP flag)
IEN	1	task interrupt enabled
IRQ	1	task interrupt pending

These registers can be viewed with the command

```
EXAMINE IMP STATE
```

Debugging flags

The IMP device implements these debugging flags:

```
SET IMP DEBUG=WARN           print warnings for unusual conditions
SET IMP DEBUG=IO             trace all IMP device I/O instructions
```

Remember that you must enable debugging output first before these settings will be effective; refer to the *SIMH User's Guide*, "Controlling Debugging," for more information.

Real Time Clock

The IMP/TIP hardware contained a custom real time clock which was unique to that implementation, and further, the IMP/TIP did not have the standard H316 power line clock or real time clock. The IMP/TIP clock is in some ways similar to the H316 real time clock option, *but it is not the same.* Normally the CLK device would be disabled when emulating the IMP, however there is no actual conflict between the CLK and the RTC devices and both can be enabled simultaneously.

The RTC device supports the following SET commands:

```
SET RTC ENABLED            enable RTC emulation
SET RTC INTERVAL=i         set the RTC tick interval to i microseconds
SET RTC QUANTUM=q          set the RTC resolution to q ticks
```

The INTERVAL parameter sets the interval between RTC clock ticks, in microseconds and the QUANTUM parameter sets how often, as a number of clock ticks, simh updates the RTC count. To give an example, if INTERVAL was set to 20 and QUANTUM was 1, simh would attempt to increment the RTC count by one at a rate of 50,000 times per second. Modern PCs are very fast and it would probably be possible for simh to do this, but the amount of overhead would be huge and it would be impractical.

Instead, for example, if INTERVAL was still set to 20 but QUANTUM was now set to 50, simh would add 50 to the RTC count at a rate of 1000 times per second. This represents much less overhead and yet gives the same net counting rate as before. However, now the clock count jumps in increments of 50 rather than 1. Most software wouldn't notice, but it's always possible this could cause problems.

It's important to note that the QUANTUM does not affect the overall clock frequency – as long as the INTERVAL is set to 20 the clock would count at an effective rate of 50 kHz regardless of the QUANTUM value.

Registers

The RTC device implements the following registers:

name	size	comments
ENA	1	RTC is enabled
COUNT	16	current count
IEN	1	RTC interrupt enabled
IRQ	1	RTC interrupt pending
TPS	1	effective ticks per second
WAIT	24	simulator time until the next tick

These registers can be viewed with the command

```
EXAMINE RTC STATE
```

Debugging flags

The RTC device implements these debugging flags:

```
SET RTC DEBUG=WARN          print warnings for unusual conditions
SET RTC DEBUG=IO            trace all RTC device I/O instructions
```

Please refer to the *SIMH User's Guide*, "Controlling Debugging," for more information and remember that you must enable debugging output first before these settings will be effective.

Watch Dog Timer

The IMP/TIP hardware also had a custom watch dog timer implementation which would force a non-maskable interrupt if was ever allowed to expire. The WDT device in simh implements this feature. The IMP/TIP watch dog timer also implemented a couple of other unrelated features – these include the status display panel, which you can see featured prominently in some IMP photos, and a flag to indicate whether the CPU was a H316 or DDP-516.

The WDT device supports the following SET commands

```
SET WDT ENABLED             enable WDT emulation
SET WDT DELAY=n             set the WDT delay to n milliseconds
```

The DELAY parameter sets the WDT timeout, in milliseconds. This is a 16 bit unsigned value, so the maximum WDT delay is just over a minute. Note that there are no H316 instructions that disable the WDT – only one to reset it – so if *the WDT device is enabled the WDT will run anytime simulation is active*. If the code fails to reset the WDT in a timely fashion, the WDT interrupt will occur regardless of the enabled or disabled state of the interrupt system.

Setting the WDT delay to zero prevents the WDT from ever generating a timeout. This special feature allows the WDT device to be enabled, so that code which executes WDT, status light or AMI512 instructions can be executed and debugged, but without inadvertently triggering a WDT timeout.

Registers

The WDT device implements the following registers

name	size	comments
COUNT	16	current countdown
TMO	1	WDT timed out
LIGHTS	16	last "set status lights"
WAIT	24	calcuated time until the next tick

These registers can be viewed with the command

```
EXAMINE WDT STATE
```

Debugging flags

The WDT device implements these debugging flags:

```
SET WDT DEBUG=WARN          print warnings for unusual conditions
SET WDT DEBUG=IO            trace all WDT device I/O instructions
SET WDT DEBUG=LIGHTS        trace IMP status light changes
```

Remember that you must enable debugging output first before these settings will be effective; refer to the *SIMH User's Guide*, "Controlling Debugging," for more information.

Communications Devices

Modem Interface

IMPs and TIPs communicated with each other via leased telephone lines and synchronous modem links. These modem links were point to point – each IMP/TIP modem communicated with exactly one other modem attached to exactly one other IMP/TIP. SIMH simulates these modem interfaces and allows them to be attached to either a real, physical, serial port on the host computer or to a virtual UDP/IP port. Either can then be connected to another simh instance running the IMP/TIP software, or even a real IMP or TIP if someone gets one working again, and a network can be assembled.

Simh implements five modem interfaces, MI1 thru MI5. Initially MI1 thru 3 are enabled and MI4 and 5 are not, however this can be changed with the commands:

```
SET MIn ENABLED             enable modem line n
SET MIn DISABLED            disable modem line n
```

A limitation of the original IMP hardware is that the DMC channels used by modem lines 4 and 5 conflict with those used by host interfaces 3 and 4. Thus it not possible to enable all modem lines and all host interfaces at the same time (see 3.2.1.2).

Modem interfaces implement one parameter which may be explicitly set:

```
SET MIn BPS=56000
```

This parameter sets the simulated line speed, in bits per second, for UDP/IP virtual modem connections.

Modem interfaces also implement an interface (local) and a line (remote) loopback feature. This cause the modem to receive its own transmitted messages, and are analogous to features in the IMP modem hardware. Loopback may be enabled or disabled with these commands

```
SET MIn LOOPINTERFACE       enable interface loopback on line n
SET MIn NOLOOPINTERFACE     disable interface loopback
SET MIn LOOPLINE            enable line loopback on line n
SET MIn NOLOOPLINE          disable line loopback
```

Registers Modem interfaces support the following registers:

name	size	comments
RXPOLL	32	receiver polling interval
RXPEND	1	receiver waiting for input
RXERR	1	receiver error flag
RXIEN	1	receiver interrupt enable
RXIRQ	1	receiver interrupt request
RXTOT	32	count of total messages received
TXDLY	32	calculated delay before transmitter done
TXIEN	1	transmitter interrupt enable
TXIRQ	1	transmitter interrupt request
TXTOT	32	count of total messages transmitted
LINK	32	link number for h316_udp module
BPS	32	simulated line speed for UDP connections
ILOOP	1	interface (local) loopback enabled
LLOOP	1	line (remote) loopback enabled

These registers can be viewed with the command

```
EXAMINE MIn STATE
```

Debugging flags The modem device implements these debugging flags:

```
SET MIn DEBUG=WARN          print warnings for unusual conditions
SET MIn DEBUG=IO            trace all modem interface I/O instructions
SET MIn DEBUG=UDP           trace all UDP packets and connections
SET MIn DEBUG=MSG           trace all IMP messages sent or received
```

Remember that you must enable debugging output first before these settings will be effective; refer to the *SIMH User's Guide*, "Controlling Debugging," for more information.

UDP/IP Tunnels A virtual modem may also be tunneled using UDP/IP to another simh instance. UDP connections are symmetrical – each virtual modem listens for incoming packets on a port which you define, and transmit outgoing packets to another host and port which you also define. It's your responsibility to define the ports appropriately so that the input of one modem is logically connected to the output of another modem. *These connections must be one to one*. Connecting more than one modem output to the same modem input will not cause any problems for SIMH, but the IMP software will become hopelessly confused.

The general form of a virtual modem ATTACH command for UDP connections is like this

```
ATTACH MIn llll:w.x.y.z:rrrr
```

This will listen for incoming packets on port “llll”, which should be a decimal number, and will transmit outgoing packets to port “rrrr” on the host with IP address “w.x.y.z”. The actual port numbers you use are arbitrary; however some care must be used to avoid conflicts with any other network applications.

The UDP attach command also has a few alternative forms; for example

```
ATTACH MIn 4431:imp.jfcl.com:4432
```

will listen on port 4431 of the current host, do a DNS lookup to determine the IP of the host “`imp.jfcl.com`”, and then transmit to port 4432 on that host. In another example

```
ATTACH MIn 1201::1202
```

will listen to port 1201 on the current, local, host and transmit to port 1202 also on the local host. This is useful when both SIMH instances are running on the same PC. In this case, *you must use different ports for transmitting and receiving*. If they are the same, the modem will transmit to itself!

Either end of the UDP connection may be disconnected with the command

```
DETACH MIn
```

Physical Serial Tunnels The command

```
ATTACH --p MIn COMnn
```

is reserved for a possible future option to attach a physical serial port to a virtual modem. This functionality is not currently implemented.

Host Interface

NOTE: The current Host Interface implementation is presently only a skeleton – just enough code exists to allow the IMP software to run. In this version, the IMP effectively sees all attached hosts as permanently powered down. Hopefully the host interface implementation can be completed at some point, just as soon as there is a suitable host emulator for it to talk to.

The IMP used the host interface to connect the H316 to an Arpanet host mainframe. This could be a PDP-10, an SDS Sigma 7, a CDC 6600, an IBM 360, or any one of many other machines. Each IMP could support up to four host interface cards; in simh these are the devices HI1 thru HI4. Initially HI1 and 2 are enabled and HI3 and 4 are not, however this can be changed with the commands:

```
SET HIn ENABLED            enable host interface n
SET HIn DISABLED           disable host interface n
```

Registers

Host interfaces support the following registers:

name	size	comments
POLL	32	host polling interval
RXIEN	1	receiver interrupt enable
RXIRQ	1	receiver interrupt request
RXTOT	32	count of total messages received
TXIEN	1	transmitter interrupt enable
TXIRQ	1	transmitter interrupt request
TXTOT	32	count of total messages transmitted
READY	1	host ready
FULL	1	host buffer full
ERROR	1	host error
LLOOP	1	local loopback enabled

These registers can be viewed with the command

```
EXAMINE HIn STATE
```

The modem device implements these debugging flags:

```
SET HIn DEBUG=IO           trace all host interface I/O instructions
```

Remember that you must enable debugging output first before these settings will be effective; refer to the *SIMH User's Guide*, "Controlling Debugging," for more information.

UDP/IP Tunnels

In simh host interfaces are simulated using UDP connections to other simh instances that simulate the host mainframe and run the Arpanet host operating system. The ATTACH command is used to connect a host interface to this other simh instance

```
ATTACH HIn *tba*
```

The host connection may be broken with the command

```
DETACH HIn
```

Summary

Additional Devices

Table 1 lists the IMP/TIP specific devices added to simh along with their I/O address, interrupt number and vector, and DMC channel assignment. Note that all the IMP/TIP devices use the H316 extended interrupts and so interrupt 1, for example, corresponds to A bit 1 in the SMK 120 instruction. Table 2 shows the extended interrupt mask associated with these devices.

DEVICE	**Address8**	**INTERRUPT10**	**Vector8**	**DMC10**	**DESCRIPTION**
WDT	26		000062		*Watch Dog Timer*
RTC	40	15	000102		*Real Time Clock*
IMP	41	16	000103		*Task Switch*
IMP	42				*MLC support*
HI4[1]	50	9	000074	10	*Host Interface #4 (RX)*
		4	000067	5	*Host Interface #4 (TX)*
HI3[2]	51	10	000075	16	*Host Interface #3 (RX)*
		5	000070	15	*Host Interface #3 (TX)*
HI2	60	14	000101	14	*Host Interface #2 (RX)*
		12	000077	12	*Host Interface #2 (TX)*
HI1	70	13	000100	13	*Host Interface #1 (RX)*
		11	000076	11	*Host Interface #1 (TX)*
MI1	71	1	000064	1	*Modem Interface #1 (RX)*
		6	000071	6	*Modem Interface #1 (TX)*
MI2	72	2	000065	2	*Modem Interface #2 (RX)*
		7	000072	7	*Modem Interface #2 (TX)*
MI3	73	3	000066	3	*Modem Interface #3 (RX)*
		8	000073	8	*Modem Interface #3 (TX)*
MI4[3]	74	4	000067	4	*Modem Interface #4 (RX)*
		9	000074	9	*Modem Interface #4 (TX)*
MI5[4]	75	5	000070	5	*Modem Interface #5 (RX)*
		10	000075	10	*Modem Interface #5 (TX)*

Table 1 - IMP/TIP Devices

1	2	3	4	5	6	7	8	9	10	11	12
M1RX	M2RX	M3RX	M4RX H4TX	M5RX H3TX	M1TX	M2TX	M3TX	M4TX H4RX	M5TX H3RX	H1TX	H2TX

[1] The modem 4 interrupt and DMC conflict with host 4 – only one of the two may be active.
[2] The modem 5 interrupt and DMC conflict with host 3 – only one of the two may be active.
[3] Missing reference
[4] Missing reference

Table 2 - Extended Interrupt Mask

I/O Instructions

Modem Interface I/O Instructions

Table 3 summarizes the modem interface I/O instructions. In this table, "n" represents the modem number, (1 thru 5), and "dd" is the device I/O address assigned to that modem (718 thru 758).

Mnemonic	**Opcode**	**Operation**
MnOUT	0300dd	*start modem output*
MnUNXP	0301dd	*un-cross patch modem*
MnLXP	0302dd	*enable line cross patch*
MnIXP	0303dd	*enable interface cross patch*
MnIN	0304dd	*start modem input*
MnERR	0704dd	*skip on modem error*

Table 3 – Modem Instructions

Host Interface I/O Instructions

Table 4 summarizes the host interface I/O instructions. In this table, "n" represents the host number, (1 thru 4), and "dd" is the device I/O address assigned to that host (508, 518, 608, or 708).

Mnemonic	**Opcode**	**Operation**
HnROUT	0300dd	*start regular output to host*
HnIN	0301dd	*start host input*
HnFOUT	0302dd	*start host final output*
HnXP	0303dd	*cross patch host*
HnUNXP	0304dd	*un-cross patch host*
HnENAB	0305dd	*enable host interface*
HnERR	0700dd	*skip on host error*
HnRDY	0701dd	*skip on host ready*
HnEOM	0702dd	*skip on end of host message*
HnFULL	0705dd	*skip on host buffer full*

Table 4 - Host Instructions

Other I/O Instructions

Table 5 summarizes the additional miscellaneous I/O instructions.

Mnemonic	Opcode	Device	Operation
SMK 120	170120	CPU	*set extended interrupt mask*
	030026	WDT	*reset watch dog timer*
	170026	WDT	*set status lights*
AMI512	070026	WDT	*skip if this machine is a DDP-516*[5]
CLKON	030040	RTC	*enable RTC*
CLKOFF	031040	RTC	*disable RTC*
RDCLOK	131040	RTC	*read RTC count and always skip*
TASK	030041	IMP	*cause task switch interrupt*
RDIMPN	131041	IMP	*read IMP number and skip*
AMIMLC	070042	IMP	*skip if this machine is a multi-line controller*[6]

Table 5 - Other Instructions

COPYRIGHT NOTICE and LICENSE

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

Original code published in 1993-2013, written by Robert M Supnik

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

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

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

Except as contained in this notice, the names of the authors shall not be used in advertising or otherwise to promote the sale, use or other dealings in this

[5] On simh this instruction is a NOP and never skips (simh simulates an H316, not the DDP-516).

[6] "Multi-line controller" was the official name for the TIP. MLC support is not implemented and this instruction is currently a NOP and never skips.

Software without prior written authorization from each author.