OPERATOR GUIDE TO THE PLURIBUS IMP

<<Version = 1200 Patch 7>>

This is a general guide to running the IMP for people who are not IMP programmers. Familiarity with Pluribus hardware and general systems concepts is assumed.

Some of the commands described in this document require that the site have the override capabilities enabled on their Node. The site operator must call the NOC to obtain this capability before they can use these commands.

10.1 INTRODUCTION

Because the new version of software (PSE 1200 Patch 7) does not support DDT, without NOC intervention, a diagnostic operating system was created to help diagnose IMP problems. This diagnostic must be run with the IMP off-line to the network. Before loading in the diagnostic operating system, the modem connections to the IMP must be uncoupled. The diagnostic software is not compatable with the operational version of IMPSYS. If the modem connections are not uncoupled, and the diagnostic system is loaded, the IMP will not come up.

The commands for the diagnostic software are the same as those for the operational version of IMPSYS. These can be found in the preceding pages of this manual. Some commands need the OVERRIDE capability set, this is true with the diagnostic software also, but it is not required that NOC intervention is needed to obtain the OVERRIDE capability. With this diagnostic software, all functions of IMP DDT can be accomplished without NOC intervention.

10.2 BUILDING VHA BY HAND

A. Turn OVERRIDE ON <cntrol-o>.

B. VHA Locations

2,4A20/	SERIAL NUMBER OF VHA
2,4A22/	LENGTH OF VHA TABLE
2,4A24/	ALWAYS ZERO (0)
2,4A26/	VHA 1 ALWAYS ONE (1)
2,4A28/	VHA 2
2,4A2A/	VHA 3
ETC.	ETC.

Starting with location 2,4A28 the VHA would be entered as:

Host Port = 0 IMP = 5

<2,4A28/> 0 <005> would be VHA 2

or

Host port = 6 IMP = 21

<2,4A2A/> 0 <616> would be VHA 3

All locations must be entered as hexadecimal numbers.

INDEX

1 LOCAL DISPLAYED INFORMATION

1.1 CRT DISPLAY

The top part of the terminal screen displays hosts and modems status and traps that have been sent to the NMC.

Format (all in hex):

```
Bxxx Bxxx Bxxx Bxxx Bxxx Bxxx Bxxx Bxxx Bxxx Bxxx Bxxx

Dxxx Dxxx Dxxx Dxxx Dxxx Dxxx Dxxx Dxxx

TRAP#  PROC#  COUNT  REG1  REG2  REG3  REG4  REG5  REG6  REG7
```

Each status word for hosts/modems consists of 4 digits of coded information as explained below:

MODEMS STATUS

Digit	Description	Normal Value
1 leftmost	I/O Bus Currently Being Used C = Modem Interface on E-bus D = Modem Interface on F-bus	D (F bus)
2	Modem interface number	0 thru 7
3	Interface LOOP Status 0 = NOT LOOPED 4 = INTERNALLY LOOPED 8 = EXTERNALLY LOOPED	0 (NOT LOOPED)
4	Interface (line) Status 0 = UP 1 = DOWN F = NEVER EXISTED	0 (UP)

HOSTS STATUS

Digit	Description	Normal Value
1 leftmost	I/O Bus Currently Being Used A = Host Interface on E-bus B = Host Interface on F-bus	B (F bus)
2	Host interface number	0 thru 12
3	-----NOT USED	Always 0
4	Interface (host) Status 0 = UP 1 = DOWN 2 = TARDY 3 = NON EXISTENT 4 = IMP SOFTWARE NOT INITIALIZED or HOST RECEIVED QUIT F = NEVER EXISTED	0 (UP)

1.2: CONTROL PANEL

OPERATOR PANEL: The standard display on the operator panel of a running IMP is:

REGISTER LIGHTS - not relevant

ADDRESS LIGHTS - a multiplexed display, which usually displays which processors are running the system by blinking the bit corresponding to a given processor. The lowest numbered processor is bit #0, so for a system with processors 12, 13, 32, & 33, the assignment would be bit#0=P12, bit#1=P13, bit#2=P32, bit#3=P33. Processors which are discovered (bus couplers exist) but which are not running the system fully are represented by a non-blinking bit position. If either processor on the bus with the operator panel is running STAGE, the number of the stage being run is displayed by the bit number of the left-most bit which is off. Thus, if a processor is running stage 5, bit 5 and all bits to the right of it would be off. If one processor is running stage and the other is running the system, the blinking processor display would be overlayed on the stage running display. Lines are displayed in address lights 15-8. Line 1 = bit 15, and a bit off indicates a line declared up by the IMP. A line displayed as up in the lights will typically not be declared up by the network for another minute or so.

DATA LIGHTS - usually display what hosts are up and in a running system. Hosts are displayed by bits 15-0 in the same manner as lines

are. A bit off indicates that a host is declared up by the IMP. Thus, a display of 3F1F in the data lights indicates that hosts 0, 1, 8, 9 and 10 are up. When the reloader/dumper is running, the current address is displayed in the data lights. this will count if a reload or dump is in progress. The Stage running will be displayed in the address lights.

1.3: RLD CARD LIGHTS

The RLD card has three lights arranged vertically on the front edge of the card. The top light flashes each time the special reload header is detected coming into any of the modems connected to the RLD. The middle light indicates that at least one such header has been detected since the last time the bus was reset. The bottom light flashes each time a reload packet with correct checksum is detected and used to cause an appropriate bus transaction (store to memory, write a register, etc). When a reload using the RLD card is in progress (and being successful), the top and bottom lights will flash together, and the middle light will stay on. If only the top light flashes, it indicates that the checksums are bad on the received reload packets (due possibly to a flakey line or interface).

1.4 PID CARD LIGHTS

The lights on the edge of the PID card indicate the highest pid value which has been written to the card and read yet. Its main value in a running system is to provide an indication of I/O bus activity. Thus, if the processors connected to the console are both halted for some reason (and therefore not driving the control panel), some indication of life in the system can be gotten by looking at the PID cards.

2 RUNNING IMP DDT

2.1 IMP PAGE TYPES

The IMP thinks of parts of itself as having type numbers. These parts are called pages (logical pages, as opposed to the physical page numbers for chunks of core). These are assigned as follows:

Type (Hex):	Page Name
0	Reliablility Code
2	DDT Code
4	Warm Code
6	Fake Host Code
10	Spare Reliability Code
12	Spare DDT Code
14	Spare Warm Code
16	Spare Fake Code
20	IMP Variables
22	IMP 2nd Variables
30	IMP Buffers page 1
32	IMP Buffers page 2

2.2 DDT COMMANDS

Numbers

The new DDT works solely in hexadecimal. The radix commands <esc> O, H, D and the radix specifiers "'" , "}" , "." have all been removed or applied to other purposes. Another character ($) is now used to specify decimal input rather than hexadecimal and is the only exception to the hexadecimal rule.

Commands

x,y/ This is the basic examine command of DDT to return the contents of a memory location. There are two cases of this command depending on the value of Y. If Y is a local address (ie Y is less than 4000) then X is the mask of processors whose memory is to be examined (this means that the answer returned will be from one of the processors specified by the mask X). A special case is a negative mask value and sets the processor mask to be all those that are known to exist to stage BD. In the other case, when Y is greater than 4000 and therefore a reference to common memory, iyt then specifies the map setting to use in the reference. In this case X can be either a logical page (x < 200) or a physical page (x is odd or x > 200).

x/ This is a simplified case of the above and does an examine of the address X using the last processor mask specified if X is a local address or the last map specified if X is a common memory address address.

(The following command requires the OVERRIDE CAPABILITY)

x,y<cr> Carriage return is used to insert new values into memory and close the location currently examined. X and Y will be inserted into the current location and the next location respectively if the current location is still open and then the location is closed. A location is open when it has been examined but not closed with a carriage return or linefeed.

(The following command requires the OVERRIDE CAPABILITY)

x<cr> This is just the one argument form of the above and stores just one number into memory

<cr> This is the no argument form of the above and stores nothing but closes the location.

<lf> Linefeed closes the current location and examine the next location

x y Space adds two arguments together and the result becomes one argument. eg "x y/" will open the location at x+y.

<del> Delete (=rubout) will zero all current input and will restore DDT to the state it was at the last typeout.

of typing the current address. It can no longer be used to specify a decimal number.

$ Dollar sign is now used to say that the number just typed in is a decimal number.

Typeout

All typeout from DDT are four digit hexadecimal.

Examine Formats

a,b/x y

This is the usual format of an examine. If a,b is a local memory reference, then only x will be printed. Note that tne processor that did the reference is no longer specified. If a,b is a common memory address then x and y are the contents of the main and spare pages if they differ. If some kind of error occurs in the reference then the x and y are replaced by the appropriate error messages for the corresponding pages.

Error Messages

There are two formats of error messages for the two situations where an error can occur. A system wide error can occur causing Stage to put the system into the stand-alone DDT mode (if enabled by DEBUGM). In this situation the bit of the processor reporting the error is printed first followed by the error followed by the location of the error. The second type of error is a store or read error as a result of a DDT reference. In these cases the error is typed first followed by a number specifying the mask of processors that failed the reference.

Errors

QUIT The location referenced by DDT resulted in a QUIT or an unexpected QUIT occurred in the running of the system. In the system QUIT the address returned by DDT is not the address of the instruction producing the QUIT but the address of the 001 Trap specifying the quit. The location of the QUIT can be found in the snapshot area in the Stage variables area.

NX Blt returned, a non-existent memory code, as a result of some DDT reference.

FRMT BLT returned a format error in response to some DDT reference. This usually means that DDT had set up parameters improperly for BLT or that a reference to a nonexistent processor was made.

TO Timeout- BLT took too long to complete a reference and aborted

IL An attempt was made to execute a non-instruction

FADE The halt all processors trap was encountered in the running of the system

SPECIAL FEATURES:

FC,X ^C "Crosspatch to a Node X's tty.

^D "CNTL-D - COMPLEMENT VALUE OF THE ON/OFF SWITCH
TO EITHER ALLOW OR DISALLOW TRAPS ON THE terminal.

^L "CNTL-L" - ECHOS A ^L (FORM FEED) TO THE TTY.

^. "CNTL SHIFT P" UNDOES CROSSPATCHS AND RESUMES TALKING TO DDT.
(This feature will be useful when users are using the
crosspatch to talk to another Node in the Network)

****
The site user must notify the NOC before they can use the
following command. This capability is required to perform
some of the privileged commands in this document.
****

^O "CNTL-O - COMPLEMENT VALUE OF THE ON/OFF SWITCH
TO EITHER enable or disable the override capability.

3. THE OPHELP COMMANDS

#NH nice stop and halt forever

#NS nice stop and restart

#NR nice stop and reload

The "nice stops" cause the IMP to observe all network protocols before going down (send IMP going down message, turn off hosts before modems, etc.). Nice stops should be used whenever possible if the IMP is on an operational net to avoid perturbing the rest of the net. After a 'halt forever' stop, DDT may be started by pressing RESET and ATTN, or by 308 (R0=308, R8=FC00, RUN, from the console), provided the system is in DEBUG mode.

#PH panic stop and halt forever

#PS panic stop and restart

#PR panic stop and reload

Panic stops do exactly that, and are a good way to quickly cause the machine to halt or reconfigure (or reload) either when there is no network to worry about, or when things have to happen in a hurry.

<ESC> C clear illopr tables: zeroes all trap information saved in common memory

(The following commands in this section require OVERRIDE CAPABILITY)

#HU unloop host #

#HL loop host # at interface

Host looping and unlooping changes will be visible immediately in the console lights (in contrast to modems), unless the host is broken.

#MU unloop modem #

#ML loop modem # at interface

Looping or unlooping a modem will result in some combination of 5C1, or 5C4 traps being generated. Looped state will not be visible in the lights until the line is declared up by the IMP (1-2 minutes). Modems are numbered from 0 to 7 and correspond to switch settings from 15 to 8 on the interface.

#HB return address for param block of host # (0,1,...)

#MB return address for param block of modem # (0,1,...)

The address is typed out and DDT is left in the same condition it would be in if you had typed the address in. Therefore, to open loc +20 in the parameter block for host 2, type "2,HB" and then " 20/". (see description of parameter blocks.)

KI return address of iokill

useio and iokill are tables which describe what I/O to ignore and not ever try to discover under any circumstances (iokill).

Both tables are structured in the same way:

Each table consists of 4 words, each of which controls 16 possible device addresses (one per bit) starting from a corresponding base address in a table called iobase.

iobase E100

E200

F100

F200

The least significant bit (=bit 0 or the "1 bit") selects the base address; since each device occupies 10H locations, the device at the "2 bit" (or bit 1) selects the device at <base+10H>.

Therefore, to iokill a device at E220, turn on the "4 bit" (bit 3) in the second word in iokill (base = E200) by loading an 0004 into it. To iokill F1C0, turn on the 1000 bit of word 3 in iokill (by entering 1000). To select F100,F120,F1C0,F1E0, add all the bits together (0001+0004+1000+4000+8000) and load a D005 in word 3.

Use of the features: useio is useful both as information about what the system has discovered, and as a means of forcing the system to switch to the other interface of a doubled pair. This is accomplished by removing the bit for the member of the pair currently being used. The removed interface will be rediscovered by the system and reentered in the tabel in about 1 minute. Adding bits to useio may cause the system to go through stage (since if the device doesn't exist we will get quits) and is NOT recommended in general, but can speed discovery of devices that do exist. Removing F devices in doubled M/I machines may cause the system to stage because of the asymmetry of the m/i-m/i path. iokill provides the mechanism for permanently removing devices from the system. Setting a bit in iokill kills the device and causes the corresponding bit to be taken out of useio. Turning off the bit causes the system to be able to discover the device again.

4. RELOADING

4.1 RELOADING USING CASSETTE

All cassette tapes are set up to load and start the Imps without operators' intervention. To load from cassette, place the cassette tape in the reader and depress the RESET and LOAD buttons on the operators' panel. The Imp will automatically startup after the cassette tape is read in and rewind to load point.

4.2 RELOADING USING THE RELOAD CARD

Since this operation is only initiated by the NMC using a NMC NU system, there is nothing to do at the site. The RLD card was meant to be used when it is not possible to get help in reloading a dead IMP. It is generally more reliable and faster to use IMPSYS cassette tapes.

5. MANUAL RESOURCE SWITCHING

5.1 PROCESSOR CONTROL

*** This section requires the OVERRIDE capability. ***

Turning processors on and off

There are several ways to change a processor's interactions with the system. All involve setting the processor mask bit (the bit which the processor would blink on the control panel) in one of four words in common memory. These words and their effects are described below.

prokil 0,40C6 This causes the specified processor not to be restarted if he stops. Furthermore, the system will not try to discover whether his control register (R15) is there. This allows machines to run in split mode without interference (reading R15 can halt a running processor).

prohng 0,40C8 This will hang the specified processor at a late stage. He will participate in checksumming, etc., but not in running the IMP (or blinking his bit).

ampman 0,40CA Ampman is a block of three words allowing various kinds of manual amputation. Prohlt and proamp are the second and third words in the block. The first word (ampman) is copied by each processor

into its buskil word (see below. This is the preferred method of removing common buses.

prohlt 0,40CC Setting a processor's bit in prohlt also causes it not to be started. Also, if it is already running, it will halt itself cleanly (may take 1-2 minutes).

proamp 0,40CE Proamp causes the whole processor bus to be amputated by disabling forward transfers in its bus couplers. This is the positive way to turn off a processor who may be causing damage to the system. (but both processors on the bus get turned off when you do this.)

5.2 MEMORY CONTROL

There are two ways to turn off memories in the IMP. The first is by setting bits in a block called memkil to turn off individual 4K pages, and the second is by amputation of a whole memory bus using a word called buskil. [[MEMORY KILLING USING buskil NOT IMPLEMENTED YET]]

memkil 032A (4 word block) Setting a bit causes the corresponding 4K memory segment to not be used. Correspondence is as follows:

memkil - 0 (bit 0) to 1E00 (bit 8000)

memkil+2 - 2000 - 3E00

memkil+4 - 4000 -5E00

memkil+6 - 6000 - 7E00

(so to turn off the 4200 and 4400 pages, set memkil+4 to 0006)

5.3 BUS CONTROL

buskil 0328 word for removing busses (type -: to DDT first to change all locals) I/O busses (NOT memory busses yet) may also be turned off by setting the appropriate bits in a word in local memory called buskil. The procedure is described below. Since ampman (see above) is copied into buskil, set the bits into ampman first (see above).

Bit assignment: The bit assignments for buskil are:

0 - 1bit - E000 bus

1 - 2bit - F000 bus

2 - 4bit - 0 memory bus

3 - 8bit - 4000 memory bus

buskil 0328 word for removing busses. (Type -: to DDT first.)

5.4 DEVICE CONTROL

useio and iokil Turning devices on or off is usually done by setting or clearing bits in the useio or iokill blocks. The KI ophelp command returns the starting addresses of the useio and iokill blocks.

6. HOST AND MODEM CONTROL

6.1 LOOPING AND UNLOOPING

*** This section requires the OVERRIDE capability. ***

Changing the looped state of modems or hosts is done by use of the appropriate OPHELP command:

#HU - unloop host #

#HL - loop host # at interface

#MU - unloop modem #

#ML - loop modem # at interface

6.2 SWAPPING DOUBLED INTERFACES

There are two ways to swap to a spare interface for machines with separate M and I busses, but only the first of these will work with M/I machines: The first way (works for all types of machines with double interfaces) is by setting the mask bit of the device to be killed into the appropriate word in the iokill block. The starting address of the iokill block is gotten by use of the "KI" OPHELP command (see Section 3). Note that this is the ONLY way to swap on M/I machines, since the program tries very hard to swap back to an F bus interface if one exists. The second way is to delete the mask bit of the device to become the spare from the appropriate word in the useio block (get the starting address of useio by the "KI" OPHELP command, as explained in Section 3.)

6.3 HOST PARAMETER BLOCKS

Parameter blocks are where the state of each logical interface (doubled is one logical interface) is maintained by the program. The #HB OPHELP command is used to find the starting address of the parameter block # (see Section 3). Interesting entries: (see the listing for a complete description)

001F - (low byte of 001E) Host State.

0 - up

1 - ready line down

2 - tardy

3 - nonexistent

4 - IMP software not initialized

000E - Transmit pid (hardware switch settings)

000F - Receive pid

0020 - Interface address (on I/O bus) (if this address doesn't look anything like an I/O address, you may have gotten into the parameter block of one of the fake hosts or a VDH) This is the address of the interface currently being used, if it is doubled.

0022 - Spare interface address (on I/O bus). Or 0 if there isn't an acceptable spare. [NOTE - SUCCESSFULLY SWAPPING INTERFACES CAUSES THE PREVIOUS SPARE TO APPEAR AT 0020, AND THE PREVIOUS MAIN TO BE AT 0022]

003C - Host throughput counters. These are a block of 8 throughput counters which are sent to the NMC and then zeroed host every minute. They are:

3C - internode messages host-to-IMP

3E - internode messages IMP-to-host

40 - internode packets host-toIMP

42 - internode packets IMP-to-host

44 - internode messages host-to-IMP

46 - intranode messages IMP-to-host

48 - intranode packets host-to-IMP

4A - intranode packets IMP-to-host

002C - Host dead subcodes. This has the reason why a host went down. Or 0 if the host is up. see 1822 manual.

002E - IMP number for this host. This is of interest if the IMP uses multiple IMP numbers.

6.5 MODEM PARAMETER BLOCKS

The start of the modem parameter block for a given modem is gotten by the #MB ophelp command. Remember that modem numbers start with 0, and the IMP adds 1 to the device number set into the switches of the modem to ensure that this is so. Interesting locations (displacements into the block):

0002 - line state word. This word contains the line state in the left byte and -a count- in the right byte. The bits in the left byte have the following meanings:

100	master bit: on if we are higher # imp on this line or this line is hard down
200	---line is down and in software reset
400	line up bit
800	heard a hello
1000	heard a hello-up
2000	take line down if this is set
4000	send a hello
8000	send routing

If on a normal line that is up, bits 800, 1000, 4000, 8000 should be flashing, perhaps imperceptibly. Bits 100 (if our imp number is higher than our neighbors) and 400 should be solidly on.

The right byte is a counter whose use varies and whose value is useful mostly to the guys.

0004 - neighbor on logical line in upper half and old neighbor in lower half

0006 - checksum error count This is the count of hardware-detected errors which are presumed to be checksum errors. These errors do not directly trigger any kind of trap, and may result from any kind of problem between the front end of the transmitting modem interface and the front end of the receiving interface. These errors are seen anytime the modem is reset, such as a line going down and then up.

000C - logical modem number (right byte of 000C)

000E - transmit pid (switch setting) (left byte)

000F - receive pid (switch) (the low byte of 000E)

0020 - interface address (on I/O bus). This is the address of the interface currently being used if it's a double.

0022 - spare interface address (on I/O bus). Or 0 if there isn't any acceptable spare. [NOTE - SUCCESSFULLY SWAPPING INTERFACES CAUSES THE PREVIOUS SPARE TO APPEAR AT 0020, AND THE PREVIOUS MAIN TO BE AT 0022]

6.6 HOST TESTING

At the moment, the only ways to test a host consists of looping the host at various places using either internal software loops or external looping plugs. The NMC can then run their host testing procedure via NMC NU and return results. This of course requires the imp to be up and running on the network.

6.7 MODEM TESTING

Modem testing done at the site is limited to variously looping interfaces cables or modems and seeing if the imp considers them good enough to declare up. For lines that are intermittently bad, patches can be installed by software guys to count packets or errors over periods of time so a performance evaluation can be made.

7. RUNNING MESSAGE GENERATOR

*** This section requires the OVERRIDE capability. ***

Message generator is a routine which runs as a fake host on the local IMP and sends data in selectable length messages at selectable rates to a selected host(s) on a selected IMP. The starting address of the parameter block is obtained by using location 6,5e66 . Entries in the block are as follows: (adresses are displacements into the block)

00 - length in words. Initialized to be 8. Set to max of 1F7, or negative for torture test (see below)

02 - first leader word - don't change unless you know what you are doing. Initialized to be 0F00.

04 - second leader word - don't change. Initialized to be 0

06 - third leader word - destination host #. (Or set to OFF to send to discard. Initialized to be OFF.)

08 - fourth leader word - destination IMP #. Initialized to be ourself.

0A - fifth leader word - Initialized to be 0. Set to 3 for raw packets.

0C - sixth leader word - don't touch. Initialized to be 0.

0E - control word. 0 = off, 1 = on. Initialized off. Set to 1 to run it.

10 - frequency. Set to 0 to go as fast as possible. Set to 1 to send a message every 25 msec. Shift 1 left one place to half rep rate. (So setting a 2 in gives 50 msec rep rate,

4 = 100 msec, 8 = 200 msec...etc.) Initialized to be 1.

(see Section 11.C.)

If the IMP # is set to be us, no traffic goes out over modem lines.

"Torture test" - The torture test is a special case of message generator and sends messages of standard lengths to four specified Host-IMP pairs. The four Host-IMP pairs are stored in table STATDT (6,5E48) and have simply the format Host, IMP, HOST, IMP ---. The torture test is specified by setting the first word in the message generator block (length) negative (8000 or greater). The lengths used are 8,72, 496 and 504 words for Host-IMP pairs 1 through 4 respectively. The frequency at which a message is sent is the same as in other message generator stuff.

8. UNDERSTANDING THE STAGE SYSTEM

8.1 A BRIEF DESCRIPTION

The stage system is a basic chunk of software upon which the Pluribus IMP system is built. The stage system can be thought of as fulfilling two purposes: in one sense it is an initialization module that passes configuration data to the imp, eliminating the need for configuration data to be loaded into each machine; In addition, stage maintains a watch on the hardware and software in use, thus acting as a reliability module for the imp. In the initialization sense, STAGE is the first module to run when an IMP is (re)started. The STAGE software is broken into nine sequential modules, or stages, each responsible for determining the useability of some set of hardware or software. The stages are run on a round-robin basis, with the requirement that no stage can run unless all the preceding stages are also running. Each stage runs and if successful, it enables the next stage. If any stage fails to come an acceptable conclusion, it disables all the stages following it and reruns until it is happy with the system. When all the stages have been enabled, the IMP system is started and uses the configuration information provided by stage to build machine dependent tables, parameter blocks etc. While the IMP system is running, the stage system is also running all its stages at background level. Changes in the system, such as the loss/gain of memory or I/O devices are detected and subsequently can be fixed by this mechanism. In fixing something, a processor does not just change configuration information arbitrarily. One processor may see things differently from others on the machine, and so in order to have

a coherent system, the processors in the system must decide in unison before making a significant change. The implementation of this consensus mechanism is simple; each stage has a consensus word on a common communication page, and each processor wanting to change something adds (IORs) its bit to that word. The consensus concept then allows a processor to change something if, by adding its bit, the consensus word matches the word containing bits for all the processors known to exist. In this way, any adjustments are done only by the last processor to join the consensus and therefore with the approval of the other processors.

8.2 USEFUL STAGE VARIABLES

Local STAGE variables of interest:

clokrt - 50 - number of bad RTC reads (answers differ by >300 micro sec) (zeroed by 2D trap) see lclock below.

quitrt - 56 -number of successful quit-retries (got a quit the first time, but ok the second) (zeroed by 2C trap)

uillop - 78 - last F-illop

ujiffy - 8E - location of last program-in-a-loop

uquitd - 90 - got unexpected quit trying to look here

uquitp - 94 - address of place that did reference

stime - A0 - local copy of system time (sytime)

stim2 - A2 - high order time, 27, 96min/tick, 51, 50tick/day

oldp - A4 - last pid dispatch

myproc - A6 - processor name, this proc (coupler address and odd bit)

procbt - A8 - processor bit (the bit he would blink)

procno - AA - processor number, this proc

maprel - B0 - map for RELY page

mapddt - B2 - map for DDT page

mapcod - B4 - map for WARM page

mapfak - B6 - map for FAKE page

mapvar - D0 - map for VARS page

mapv2 - D2 - map for V2 page

wstage - 188 - what stage running

wdis - 18C - stage control word - bits on disable stages. (same as address lights). A way to tell where a proc is hung in stage.

iclock - 1A4 - address of the RTC this processor is using for timing various STAGE things.

8.3 STAGE DIAGNOSTICS - what a processor is complaining about

If a processor is stuck in STAGE, the address lights (for the processors connected to that bus) or wdis will tell what stage the processor is stuck in. The last bit number off to the left is the highest stage number which has been entered (and is the one we are stuck in).

Currently the stages are:

0 - LK - Local memory Kernel checksum

1 - MD - common Memory Discovery

2 - RK - Reliability page Kernel checksum

3 - BD - common Bus Discovery

4 - CD - processor Coupler Discovery

5 - RC - Reliability page Code checksum

6 - LC - Local memory Code checksum

7 - MM - common Memory Map management

8 - ID - I/O device Discovery

9 - AR - Application Reliability and initialization

The operational system

Possible causes of being stuck in various stages:

0 - LK - a bad local kernel checksum here will cause a halt. We may also not be able to see any RTC (or maybe no I/O busses)

1 - MD - We will hang here if we see less memory that the system (but not if we see more). We may also have a bad common memory pointer.

2 - RK - We were not able to find a common Kernel, or had a different idea about it in that the system.

3 - BD - We could hang here by getting a quit from the VARS page (for instance because of a memory coupler failure, or actual memory failure). It could also be that the bus discovery answer is changing, or that we have an I/O coupler failure of some sort.

4 - CD - We could hang here if our coupler tables are bad (compared to the system, of course), or because BBC is broken to one processor. If the prohlt bit for us is set, we will come to this stage and halt.

5 - RC - Hanging here says we need a reload because the RELY page checksum is smashed. (RELY is in common, and usually on the lower numbered memory bus, if we have both busses and have had time to stabilize.)

6 - LC - this says local is smashed, and we need to reload (reload the system, if no other processors have good locals) hangs in both stages 5 and 6 are characterized by a number displayed in the data lights (for the console processors) which cannot be cleared.

7 - MM - We hang here because of a broken common memory code checksum, waiting for a reload or we can hange here if the cmap table is bad, or if the typeword of a page changed, or if we want to reload common. Previous comment about data lights applies here also.

8 - ID - we can hang here because our useio table is bad (we can see less I/O than others). We will also hang here if our mask bit is set in prohng.

9 - AR - We are here because we want to initialize and are waiting for enough processors to run the system (currently two for IMPs).

9. TRAPS

9.1 TRAPS - GENERAL

Traps report unexpected and expected error conditions to the NMC and the local terminal. Each trap consists of a trap identification number, a word containing the bit(s) of the processor(s) reporting the trap, a count of how many were reported, then registers 1-7 at the occurance of the first trap of this kind. The trap reporting mechanism in the imp saves up to eight different traps in a table on the first variables page. The terminal display is taken from these tables and is displayed at the top of the screen. There are two modes to the screen display: if the location of the screen cursor is at the top left (very beginning) of the screen, then up to 8 traps will be displayed. If the cursor is anywhere else, then only 3 traps are displayed. allowing more room for data printout while debugging. Generally a site terminal should be left in the first state, to allow maximum information to be on the screen. In case of IMP failure, site personnel can report the traps to the NMC. To leave the cursor at the top left, either use CR to get to the beginning or more simply type a control-L to DDT which will re-write the screen and leave the cursor in the proper place.) The processor mask is the logical OR of all processors reporting the trap, with each processor encoded as a bit postion, lowest numbered processor = bit 0. The actual mechanism for implementing traps in the program is through the illop (illegal operation) self-interrupt in the processor. Normal traps are caused by executing an illop in the code of the form Exxx, and Fxxx,

Traps sent to the NMC are in the format:

TRAP# PROC MASK COUNT REG1 REG2 REG3 REG4 REG5 REG6 REG7

traps sent to the terminal screen are in the format:

TRAP# PROC# COUNT REG1 REG2 REG3 REG4 REG5 REG6 REG7

TRAP# PROC# COUNT REG1 REG2 REG3 REG4 REG5 REG6 REG7

TRAP# PROC# COUNT REG1 REG2 REG3 REG4 REG5 REG6 REG7

The seven register reported to the NMC and the terminal screen give lots of information about the cause of the trap; e.g. device number, or line #. The trap description tells which registers are useful.

9.2 A GUIDE TO THE TRAPS

The following are descriptions of some of the more serious (expecially from a hardware point of view) of the Pluribus traps. A complete list of the traps, together with their hex locations, is given in section 9.3.

HEX DESCRIPTION TRAP OF TRAP #

1 unexpected quit

Any quit trap represents a quit which was retried once and was solid. To find where the quit came from, use DDT to look at the snapshot area (see above).

STAGE RESTART

2 program in a loop

Means that the processor didn't get through LOOP by the time it was timed out (some strip ran too long). We checked to see if were hung on a lock and weren't. This trap will work in STAGE too.

Timeout varies depending on what strip was running (range is 20msec to 150msec for timeout) Saves: (UJIFFY) E4 - p.c. at the time strip was timed out (at jiffy time)

--CAUSES STAGE RESTART

3 Completed Memory Management

This trap occurs in Stage MM if any page swapping occurred. This is expected on startup or restart but if it persists it implies that the IMP is constantly changing memory useage for some reason.

--NO STAGE RESTART

4 local clock stopped

This says we got two successive jiffies and the reading on the RTC hasn't changed. The response is to pick a new RTC. For snapshot traps: R2 - has the clock we are now using (the one we just switched to)

--FULL STATE RESTART

5 Local Kernel Checksum Broken

This says the most sacred part of local STAGE is busted. Since the desease is quite fatal, a processor with this trouble may not be able to report it before halting. If the checksum breaks while the processor is running he should be able to trap and then halt. To tell what the trouble is, put the proc bit in PROKIL (so as not to restart him), turn off his bus reset timer and verify his local memory (by comparing with some other processor).

--PROC WILL TRAP AND HALT, OR MAYBE JUST HALT

6 unexpected interrupt

This says we wether got a level 2 or level 3 interrupt, or a

level 1 which wasn't a remote power fail. This can happen because of a "reset-attn"

7 BBC MAP FAILURE

This means that a processor can't agree with the rest of the system about what common memory exists, and also can't get the right to fix it (because the consensus doesn't agree with him.) If a memory bus goes down in an n-processor system, (n-1) processors will report this trap. The nth processor makes the consensus ok and the memory table gets fixed, since the algorithm is that the processor who sees the most memory is right. This trap will also occur for things like bad bus couplers.

The bus in question will almost always be the 4000 bus. If only the 0 mem bus is on, the trap will not occur except under strange circumstances. This is because the memory table is kept in the lowest numbered memory (communications page), and if that page breaks, different traps occur.

--NO STAGE RESET

8 No PIDs in system

Means no PID cards answering (either really gone, or configured out.) The trap happens late enough that we have already found an RTC somewhere. (only one PID gone will result in the devices on that bus being configured out. If this is suspected, use OPHELP commands to see what devices

are thought to be present.)

--FULL STAGE RESTART

9 adjusted comrel

This says we moved a page of common on which we were running the higher numbered stages, and is likely when we get a broken checksum in either Kernel or the page which was moved. This will happen continuously when the Rely kernel is broken.

--USUALLY CAUSES IMP RESTART

A Jiffy clock stopped

This should only be reported by even processors and says that at least 1 sec. has elapsed without the jiffy interrupt having updated its copy of the RTC reading. It could also be due to the RTC or something in between, since the trap will be sent if the current RTC reading is more than 10000 (1 second) decimal different from the jiffy reading. (The jiffy reads the RTC every 1/60 sec)

-- [NO ACTION]

B system missed a tick

This says that the system failed to update the SYTIME word in common since the last time the reporting processor ran stage (128 msec).

The cause is not immediately obvious. Could be that we

stopped getting pids from the RTC, or have a broken memory.

--ACTION = COUNTS SYTIME BY 1

C Quit in checksum parameters

Interrupt recevied from the system during checksum rouitne.

D Quit during checksumming

D traps may accompany 3 traps, but a D trap will not always cause a 3 trap, since the quit might be in some other code during checksumming. If snapshots are turned on:

R1 - map if quit in common memory

R6 - loc+2 of the quit

All references in common mem will be through map 1.

-- NO STAGE RESTART

E Local code checksum broken

This is similar to a 5 trap but for the code page in local. Since it happens later in stage, it implies that the Kernel checksum is ok. (Local checksumming covers covers all constants in local except for the Kernel.) If other processors think their local is ok, they will start a block transfer (software) to the unhappy one, and he will participate in it.

If all processors are unhappy with themselves, they will ask

for a reload from the net.

(D traps may also occur with this one)

--TOTAL RELOAD OF LOCAL

F Not enough memory to run system

This can happen as a transient if MEMKIL is used and the new usable memory is too small. There is a remote possibility of 1 processor not seeing enough memory but still being able to run a late enough stage to produce the trap. The picture in local of what the processor sees in common:

[MYSEGS] 1st word 0-1E00
(block) 2nd word 2000-3E00
3rd word 4000-5E00
4th word 6000-7E00

(This table is bit coded by page recognized within the specified range. Thus for the first word, bit 1 on means the 200 page on 0 memory. If bit 1 of the 3rd word were on, it would correspond to the 4200 page on the 4000 memory. Bits are cleared periodically and then reappear as the processor discovers memory pages.)

10 Stage variables area quit

This trap results from the next procedure after finding the Kernel variables area, which is to scan the whole area for quits and clear the whole area if we get one. The procedure

assumes it is parity quit. Consensus is necessary, so a number of these traps will occur for multiprocessor operation. The program attempts to use the same area again, and a quit, while clearing the area, will cause the processor getting the quit to voluntarily stop using the page via MEMKIL.

ACTION: WHEN CONSENSUS IS ACHIEVED, STAGE VARIABLES ARE REINITIALIZED, PRODUCING A 12 TRAP

11 lost our communications page

This means that the page we were previously using for communications gave a quit on a write. The sequence preceding the trap is that we got a quit on a read of SYTIME during checking of the page, and then tried a write into the same location which also got a quit. (The assumption being that it was a parity quit the first time.) If the quit was fixed on a rewrite, stage proceeds normally and doesn't give any indication.

The normal cause of this trap is 0 Memory disappearing (but not if configured out)

--FULL STAGE RESTART

12 Stage Common Reinitialization

This says that the timer which says whether stage variables are current got too old (a watchdog timer), and can happen as

a result of a 10 trap. Timout time = 10-15 sec. Usually only triggered by a cold start. timer is:

STGIM - 0,5D56

--ACTION: PARTIAL STAGE RESET (already done by the time this trap produced)

13 Memory coupler quit

Coupler discovery got a quit referencing a memory bus coupler (BCM). The most likely cause is an old-style BCM on a parity memory bus.

--THE COUPLER IGNORED

14 HUNG ON BAD LOCK

This happens if: the jiffy code sees that it's time to do a 2 trap, but first it checks for a lock pattern in the code, and if it finds one, it calculates the address of the lock and checks for validity. To be valid, a lock must have either a map 1 or a map 3 address range. An invalid address range causes a 14 trap.

--FULL STAGE RESTART

14 CAN'T FIND A CLOCK

System is unable to find the real time clock.

R3 - address of real time clock

R7 - bus used

--FULL STAGE RESTART

16 remote power fail interrupt

Getting one of these causes us (even proc) to first try to stop our buddy, then wait 1 sec and try to restart him at the beginning of stage, then restart us at the beginning of stage.

--FULL STAGE RESTART

17 Can't find an RTC

This happens in the routine which looks for an RTC and is started if the main RTC stops, or if our previous idea of where an RTC might be was wrong (for instance if we are told not to use an RTC on start-up).

--FULL STAGE RESTART

18 BBC processor started

This is reported by the processor doing the restarting. To find out who was restarted, check the snapshot: R2 - address of the coupler of the restarted processor. If R2 is odd, then the odd processor was the one who was restarted.

--RESTART OF THE TARGET PROCESSOR

19 Buddy Processor started

This reports that this processor was successful in restarting the other processor bus.

1A Bad processor identity

This says that we found our own coupler address and had the wrong idea about who we were before. The algorithm for finding ourselves is to see if a given coupler address answers '2100' on all busses. We then try writing the same password as we last wrote to that coupler in I/O space, and if it quits, it's us. This trap would also happen if we never discovered our couplers.

--FULL STAGE RESTART

1B Block Transfer Timout

This says that the job of BLT is not progressing fast enough. One possible cause is the system believing a processor is ok enough to participate in his own BLT of local (to him) when he really isn't running.

--FREES BLT FOR NEW USE

1C BLT proc not in table

This says that we can't find the BLT target processor in the table of coupler addresses cleaned in an earlier stage. Not finding it stops the BLT and causes the trap. No normal IMP activity is likely to trigger this currently, but things like DDT or reloading might.

--STOPS BLT PROCESS (NO STAGE RESET)

1D Non-existent proc in BLT??

This can't really happen for hardware reasons. Getting it implies a program bug.

--STOPS BLT PROCESS

1E No I/O bus for BBC

Before doing BBC, we test the path a little by doing a store through the BBC window. If we get a quit (20 trap), we retry through the other I/O bus, and if that doesn't work either, we give a 1E trap. This trap implies that we can at least read the processor's control register (since we had already discovered him.) We try to halt the processor before BBC, and quit from the write to his control register will also cause this trap in the same manner.

--STOPS BLT IN PROGRESS (may change this later)

1F One of my couplers is broken

This occurs in the processor discovery code and says that we got a quit trying to read a coupler whose name matches our name. We can probably run okay, though the trap will continue to happen from time to time.

Snapshot:

R2 - Proc # we were looking for.

R7 - address of coupler.

--KEEPS TRYING, BUT PROBABLY WON'T JOIN SYSTEM

20 BBC transfer failure

This is caused by a quit during either a read or a write through a BBC window.

Snaps:

R1 - BBC map

R3 - value stored

R6 - I/O bus plus index into BBC window (0-6)

R7 - I/O bus BBC was done through

--TRY TO USE OTHER I?O BUS. IF ALL FAIL, GIVE A 1E TRAP.

21 Power restore interrupt

We (the even guy) got a level 4 dev 2 interrupt.

--HALT (US AND OUR BUDDY)

22 Local power fail interrupt

The processor detected that it was restarted after power was off.

--FULL STAGE RESTART--

23 Illegal level 4 interrupt

We got a level 4 interrupt which wasn't dev1, 2 or 4.

--IGNORE

24 Stage variables memory failure

Interrupt received from system while clearing common memory variables.

--CAUSES STAGE RESTART

25 Spare code page's checksum disagrees

This is not a hardware trap. Primarily it is a reminder to someone patching the code page that maybe he hasn't patched both copies. (trap will occur about once every 1.4x (# of pages in use) sec. in each processor.

--NO ACTION

26 Fixed bad memory parity

The stage (stage MC) which checksums common memory also does a loop to read the rest of common (outside the checksummed area). A quit during this test causes us to try to write 0 to the location, and if successful, will cause this trap and also zero the cell on the page which may tell the system to reinitialize if appropriate. The trap may trigger various kinds of restarts.

For snapshot:

R1 - page (map setting)

R6 - location of quit (always through map1)

--MAY REINITIALIZE. MAY TRIGGER VARIOUS RESTARTS

27 Solid memory parity error

This results from the same loop as the 26 trap above, but says that we got a quit and couldn't fix the location (either we got a quit on write, or got a quit when we read the loc again, or failed to reread the 0 we wrote there.)

Snapshot: (same as above)

R1 - page (map setting)

R6 - location of quit (always through map1)

The processor getting the trap stops using the page and does a full stage restart (and may drop out of consensus if he is the only one seeing the problems.) This process is intended to help remove processors writing bad parity from the system, but may have trouble doing that unless the bad parity writes are relatively frequent or solid. If all processors see the failure, then all will stop using the bad memory.

--STAGE RESTART

28 No useable common memory

This says we haven't found the communication page (the lowest numbered page) on any memory bus. It could happen with 2 bad BCMs, or as a result of some error in switching cables, cards, etc., or if the memory went away.

--HANGS IN STAGE 1 (if no other proc is running and holding off a timer, we will reinitialize and retry about every 1-3 minutes if we see any memory at all.

2A Quit in quit handler

Entering the quit handler for the first time sets a flag. If the flag is already set, we make a 2A trap. This trap can happen as the result of a software bug if the entry flag is initialized the wrong way when a processor is restarted after

being stopped.

--CAUSES STAGE RESTART

2B Quit on instruction fetch

This happens in the same loop as the 2A trap. If the quit target is within 4 bytes of the saved quit P.C., we call it 2B trap.

--CAUSES STAGE RESTART

2C Quit retry(ies) succeeded

This trap occurs if we got a quit which succeeded on a second try at the instruction. This seems to happen a lot, and is indicative of some sort of design problem. Unfortunately, we can't notice where we retried in general, though a special patch might be able to do it.

Snaps:

R1 - number of retries

--Since retry worked, program continued normally

2D RTC read retry(ies) succeeded

This trap occurs if we got inconsistent readings on two successive reads of the real-time clock. The readings are allowed to differ by 3 (300 microseconds). If they differ by more, we count a counter, which is then reported later by this trap. This trap probably indicates some sort of design

bug in the hardware.

Snaps:

R1 - number of retries

--Just retries until it succeeds

2F Copy/Cleared Failed

Unable to copy/clear address located in register 4 of TRAP.

40 Start pointer write failed

Bad start pointer given to routine to start the I/O input routine.

R1 - address in error

R3 - Length of message/packet

R5 - Device page used

41 4 start pointer failures

System software made more than 4 attempts to write the I/O buffer to the address specified in R1.

R1 - address failing

R3 - length of the message/packet

R5 - device

42 got illegal pid value

indicates that the routine all ready exists ot that that area in the BASE table is inuse.

R1 - pid level

R2 - address in BASE table

43 map error

system software received error while verifying the maps for the software.

44 LSTACK overflow

Pointer to addresses in maps exceeded limits.

45 INBASE failed

Error received while attempting to initialize the BASE table.

50 software watchdog timer expired

Timer exceeded limit. Generates an interrupt to keep software from looping.

CO tty changed psb

TTY initialize routine wither restarted or didi a reset due to quits received by the systems software.

R4 - psb

100 IMP reinitialization

This means what it says. Nothing much about the previous state of the machine remains after the reinitialization, but some clues may be gotten from previous traps, if they are still around.

--FULL REINITIALIZATION

101 smashed a buffer pointer

The table of buffer pointers (starting address and number of bytes, etc.) has had an entry smashed.. The buffer (or packet is lost).

102 Changing buffer page allocation

Reported as a result of the Node reconfiguring after a device failure in a M/I bus machine. System is now using spare device. Notify maintenance.

108 Main clock has stopped

This says that the main RTC didn't tick for three ticks (25 msec ticks) of the backup clock. It also says that the other clock became the main clock. The F clock is always preferred for the main clock if it exists CLOCK - 6238 - address of current main clock.
--STARTS USING OTHER RTC (has already started by trap time)

109 Backup clock working again

This trap says that we had one clock (either one) and now have two. If the F clock was the one that came back, the system starts using it as main clock instantly.

10A No working backup RTC

This trap happens when the backup RTC goes away for any reason. The trap will reoccur every 27 minutes until the backup reappears.

10B IMP number invalid

This says that the IMP number set into the switch register on either or both RTCs is zero or too big (>67). The most likely cause is that somebody forgot to set it when the RTCs were last installed, but it could also be a flakey switch or

associated logic on the RTC. We don't currently check for different IMP numbers in the two RTCs (unless one is zero) and will always use the IMP number on the E006 RTC when two are present. ((this aspect will be changed))

--THE IMP WILL COME UP (NOT TO THE NET) BUT WON'T RUN WELL (DDT WILL BE INACESSIBLE FROM THE terminal) AND WE WILL KEEP TRAPPING.

10C RTC gone away

This says that we ran through the bus discovery code in stage and removed the dispatch for one RTC from the table (as a result of not finding it). This trap will not occur if both RTCs have the same pid settings (having them be the same is now ok, but not such a hot idea). This trap may occur along with a 10A or 108 trap.

--NO ACTION (will cause KI trap)

110 Modem hardware gone away

(FOR 110, 111, 112, 113, 114, 115 TRAPS:) These traps are all the result of the IMP reconfiguring (the table changes in STAGE, but doesn't trap then, and this is the result of the IMP noticing the change.) (We will only give one of the three types of traps for each interface unless two halves of an interface disappear at the same time.)

Snapshots for trap 110:

R1 - address of device gone away

--STOPS TRYING TO USE INTERFACE (may also cause 204 trap)

111 Host HARDWARE gone away

SEE ALSO TRAP 110

Snaps:

R1 - address of device gone away

--STOPS TRYING TO USE INTERFACE (may also cause 204 trap)

112 Spare modem interface disappeared

SEE ALSO TRAP 110

Snap:

R1 - address of device gone away

--NO ACTION

114 Swapping modem interfaces

SEE ALSO TRAP 110

R1 - new interface we are using

--SWAP

115 Swapping host interface

SEE ALSO TRAP 110

Snaps:

R1 - address of guy who went away

R3 - address of new guy we are using

--SWAP

203 Over 2 interfaces, one device

This happens if we find at least three devices with the same device type and device number.

Snaps:

R5 - device we're trying to put in the dispatch

R3 - 2nd of the 2 devices we've already found with the same type and number.

--USE THE FIRST TWO INTERFACES FOUND

204 Removed bad base dispatch

This results from a 10C, 110, or 111 trap, generally.

--CLEARS ENTRY IN BASE DISPATCH

205 Pid for doubled interface differs

This says we found two devices with the same type and device numbers but differing pid settings. This could happen for a variety of hardware reasons (flakey pid and device no switches or operator error are the most common).

Snaps:

R5 - address of second device to be found

R4 - parameter block for first device

R3 - receive pid for second device

R6 - transmit pid for second device

--USE THE FIRST OF THE TWO INTERFACES DISCOVERED

207 Dynamic blocks area full

This says there are too many devices for us to handle. The maximum number varies from assy to assy, and is currently 40 devices (single or doubled)

--SOME DEVICES WON'T GET USED (the trap will keep occurring)

301 Detected VHA table error.

The system has detected that either a virtual host entry is not assigned to a physical host, or that more than one virtual host number is assigned to the same physical host address.

Snaps:

R3 - Contains VHA (in HEX) which is being complained about.

- - HOST TRAFFIC MAY BE LOST - TABLE MUST BE CORRECTED

302 INVHA: No Virtual address found.

The system has detected that no physical address exists in the VHA table for a virtual host being referenced.

Snaps:

R3 - should contain the VHA number.

Note: Can also be caused by PLULOG ON with no dest. host to ge to.

303 TSKVHA: No Virtual Address this source.

The system has detected that it has received host traffic from a physical host port that has no virtual address assigned to it.

Snaps:

R3 - Contains the physical IMP number.

R5 - Contains the physical host port number.

304 Too many VHA numbers.

The system has detected a virtual host address that exceeds the table length. It is treated as being invalid.

--EXAMINE VHA TABLE AND MAKE APPROPRIATE CHANGES--

305 VHA IMP Number too big.

The system has detected a virtual host physical address that has an IMP number greater than current maximum.

__EXAMINE VHA TABLE AND MAKE APPROPRIATE CHANGES--

3C0 IMP going down

This trap occurs when we enter the nice-stop sequence after either a #NS or #NR command.

3C1 Got setup for no dest

This trap says one of our neighbors is asking to be reloaded.

3C2 Flushing reload packet - No room

This trap says we lost a reload packet we were trying to send to a dead neighbor.

401 Modem bad end pointer

This applies to the receive end pointer only (so far). It says that we checked the receive end pointer after we got a receive pid and it was either less than the beginning of the buffer we were using, or past where we told it to stop. Check after evey modem input.

Snap:

R4 - address of parameter block

R6 - address of the buffer (through MAP2)

(map in MAPSAV area)

R1 - length that the hardware said it gave us.

--FLUSH THE INPUT AND RETRY. RESETS THE INTERFACE

402 Modem got a quit

This applies to receive only. (we don't currently get much information out of this one)

Snaps:

R4 - address of parameter block.

R5 - device address

R1 - end of input that the hardware told us.

(receive end pointer)

R2 - contents of status register

--FLUSH INPUT AND RETRY. RESETS INTERFACE.

403 Modem input too short

This says the packet was less than the minimum size that the net should ever give us (OA bytes).

A2EA (VARS/Vars) clklok: Lock on RTC counters

A3B6 (VARS/Vars) ltq: task queue lock

A3C4 (VARS/Vars) free: free buffer list

A3C6 (VARS/Vars) freend: end of free buffer list

A3C8 (VARS/Vars) nf: size of shared buffer pool plus minf

A4A4 (VARS/Vars) lockro: routing send buffers lock

A4A6 (VARS/Vars) cycle: timeout clock counters

A4A8 (VARS/Vars) trnlok: free transaction blocks lock

A4AA (VARS/Vars) messt: message number timeout non-lock

A4B2 (VARS/Vars) ringlk: restarter ring lock

A4D4 (VARS/Vars) tcgo: host wakeup lock

A4D8 (VARS/Vars) tbkgo: back host wakeup lock

A5O6 (VARS/Vars) stolok: slow timeout lock

A55O (VARS/Vars) conlok: configuration lock

A5BO (VARS/Vars) rmlock: (and every 020) rcv mes block locks

A93O (VARS/Vars) tmlock: (and every 020) xmit mes block locks

ACBO (VARS/Vars) reas blk lock (and every H10)

AE44 (VARS/Vars) Fake 0 DOZE lock

AE46 (VARS/Vars) Fake 0 WAIT lock

AEC6 (VARS/Vars) Fake 1 DOZE lock

AEC8 (VARS/Vars) Fake 1 WAIT lock

AF48 (VARS/Vars) Fake 2 DOZE lock

AF4A (VARS/Vars) Fake 2 WAIT lock

AFCA (VARS/Vars) Fake 3 DOZE lock

AFCC (VARS/Vars) Fake 3 WAIT lock

BO38 (VARS/Vars) back host 0 (back5) lock

BO58 (VARS/Vars) back host 1 (back7) lock

B078 (VARS/Vars) back host 2 (back9) lock

B098 (VARS/Vars) back host 3 (back6) lock

B0C0 (VARS/Vars) hi host lock fake 0

B10E (VARS/Vars) ih hardware lock fake 0

B11C (VARS/Vars) ih software lock fake 0

B130 (VARS/Vars) hi host lock fake 1

B17E (VARS/Vars) ih hardware lock fake 1

B18C (VARS/Vars) ih software lock fake 1

B1A0 (VARS/Vars) hi host lock fake 2

B1EE (VARS/Vars) ih hardware lock fake 2

B1FC (VARS/Vars) ih software lock fake 2

B210 (VARS/Vars) hi host lock fake 3

B25E (VARS/Vars) ih hardware lock fake 3

B26C (VARS/Vars) ih software lock fake 3

B2A4 (DISPLY/Vars) dsplok: display variables lock

B2C6 (ROUTE/Vars) spfrtl: Lock on common SPF tables

B2C8 (ROUTE/Vars) rutlok: Lock on routing processing

B3C6 (VHA/Vars) vhalok: Lock on VHA inverse translation table

BD52 (STAGEK/RelVars) Common Bus Discovery Consensus

BD5E (STAGEK/RelVars) processor and bus coupler discovery consensus

BD68 (STAGEK/RelVars) bbclok: lock on bus coupler states

BDBA (STAGEK/RelVars) bltlok: Block transfer lock

BE2A (STAGEK/RelVars) Consensus for Rely page Checksum

BE30 (STAGEK/RelVars) Consensus for Local Checksum

BE38 (STAGEK/RelVars) memory configuration consensus

BE92 (STAGEK/RelVars) consensus for I/O discovery

BEA6 (STAGEK/RelVars) initialization consensus lock

BEAE (PKCORE/RelVars) pkclok: lock on packet core parameters

10 DIAGNOSTIC SOFTWARE

645 (WARM/LCode)	3178:	Got msg with illegal pkt code 13
646 (WARM/LCode)	30BC:	No trnblk for inc RFNM
647 (WARM/DDTCode)	4E74:	no trnblk for inc query
650 (WARM/LCode)	305E:	Got a duplicate Allocate 1
658 (FAKREL/FakCode)	55A0:	Bad local Host in message block
681 (FAKREL/DDTCode)	5480:	Flushing an old trnblk
682 (WARM/DDTCode)	521A:	recovered an old reas block
683 (FAKREL/DDTCode)	54A0:	requeueing trnblk for IH
684 (FAKREL/DDTCode)	54CE:	trnblk/tmblk mismatch
68A (WARM/LCode)	2B54:	host blocked awaiting free buffer
68C (WARM/LCode)	2D18:	host blocked awaiting mes num or blk
68D (WARM/LCode)	28B4:	host blocked awaiting all8
68E (WARM/LCode)	2AD4:	host blocked awaiting task
68F (WARM/LCode)	289C:	host blocked requesting all8
690 (WARM/Warm)	5A1A:	host blocked awaiting trnblk
691 (WARM/LCode)	2952:	host blocked middle of 8-pkt
692 (WARM/LCode)	29A2:	Task blocked incomplete message
698 (WARM/Warm)	5DB2:	Bad rm blk for mes on host q
699 (WARM/LCode)	2AB0:	gvtsk: lost buffer in hi2tsk
69A (WARM/LCode)	2AD0:	back blocked awaiting task
6A0 (WARM/LCode)	296C:	error during host input data
6C2 (WARM/Warm)	5CD8:	bad buffer on host queue
6C6 (WARM/LCode)	28A2:	clobbered hisp requesting all8
6C7 (WARM/LCode)	28B8:	hisp clobbered in packet
6C8 (WARM/LCode)	29B2:	bad hisp for bad message
6CA (WARM/LCode)	305A:	bad trnblk buffer
6D0 (WARM/LCode)	3524:	bad buffer in t2h

6D8 (WARM/LCode)	2E4O:	ih lost a trnblk
6E8 (WARM/Warm)	5CD4:	bad ih queue structure
6FO (WARM/LCode)	2CA2:	HI bad packet length
7CO (WARM/LCode)	3O66:	Got an Out-of-range
7C1 (WARM/LCode)	2FBE:	sending out-of-range
7C2 (WARM/LCode)	31D2:	no free rm blk
7C4 (WARM/LCode)	29B6:	dest died in hi
7C5 (WARM/LCode)	2F8A:	sending duplicate reply
7C6 (WARM/LCode)	31E6:	received duplicate Get-a-block
7C7 (WARM/DDTCode)	4EB4:	sending incomplete query
7C8 (WARM/LCode)	2F62:	Sending incomplete reply
7CA (WARM/LCode)	334O:	no allocate for 1-pkt msg
FC8 (WARM/LCode)	29E8:	host sent error with id
FDO (WARM/LCode)	36AE:	nal gone negative
FD8 (WARM/LCode)	3494:	Illegal rstate/type
FEO (WARM/LCode)	398C:	Back bypassing allocate
FE1 (WARM/LCode)	3988:	Back can't back up
Lock (Source/Page)	Label:	Description

AO82 (STAGEK/Vars)	wmlock:	memory test lock
AO8E (STAGEK/Vars)	slflk:	locked copy (+2) of slfptr
AO98 (STAGEK/Vars)		memory discovery consensus
AOAA (STAGEK/Vars)		Common Kernel Discovery Consensus
A25E (DDT/Vars) d2fl:		
A26O (DDT/Vars) f2dl:		
A262 (DDT/Vars) ttylok:		
A264 (DDT/Vars) ddtlok:		

406 (MODEM/Warm) 47F0: I2MFLD: Bad checksum in routing update

407 (ROUTE/Warm) 4C92: RUTSPF: bad update checksum

409 (CONFIG/RelCode) 5AD2: MTEST: scrambled modem parameter block

40A (MODEM/LCode) 21CC: bad sentq

40B (MODEM/LCode) 1C76: I2M: lost SNDING buffer

40C (UPDWN/Warm) 4670: KNMISS: master line died

40D (MODEM/Warm) 49C4: M2IHIHY: Slave obeys master down

40E (UPDWN/Warm) 460C: PHDEDL: Slave missed k in a row

410 (MODEM/LCode) 206C: modem software checksum failure

411 (MODEM/LCode) 1D6E: broken cksum on retransmission

412 (MODEM/LCode) 1DA2: 64 retransmissions: killed line

413 (MODEM/LCode) 1DAA: 32 retransmissions: discard packet

414 (MODEM/LCode) 2202: DOAK: unexpected ack

415 (UPDWN/Warm) 44F4: PHDEDL: Excessive Hardware Checksum Errors

420 (TASK/LCode) 236C: TASK: no route for packet

421 (TASK/LCode) 2344: TASK: flushing pkt with discard bit

4C1 (MODEM/LCode) 1FF8: filling buffer error

4C8 (FAKREL/DDTCode) 538E: modem state mismatch

500 (ROUTE/Warm) 5348: SPFERR: SPF error forced restart

503 (ROUTE/LCode) 22FC: RUPWHC: routing queue broken

504 (ROUTE/Warm) 5532: RUPFLS: buffer no longer owned by routing

505 (ROUTE/Warm) 5544: RUPFLS: caller's bit not on

506 (ROUTE/Warm) 5562: RUPFLS: rupq buffer missing

507 (ROUTE/Warm) 57D2: RTRGEN: retransmission with bad length or IMP

508 (ROUTE/Warm) 58D4: RUPQCK: rupqct wrong

509 (ROUTE/Warm) 58F6: RUPQCK: recovered unused buffer

50A (ROUTE/Warm) 541A: RUPENQ: queuing packet for no one

555 (ROUTE/Warm) 545A: CHKRQ: queue count too large

557 (ROUTE/Warm) 545A: SPF accounting information

5C2 (MODEM/LCode) 20E2: M2IREG: suddenly looped line

5C3 (TASK/LCode) 2396: TASK: flushing packet for dead IMP

5C5 (MODEM/Warm) 49A2: M2IHIHY: master/slave mismatch

5C6 (MODEM/Warm) 493A: M2IHIHY: Neighbor IMP number changed

5C8 (MODEM/LCode) 20EE: M2IREG: accepting pkt on dead line

5D0 (UPDWN/Warm) 4762: LINEUP: line up, r7=neighbor

600 (WARM/LCode) 3278: No message for incm or incq

602 (WARM/LCode) 2968: host input got a quit

603 (WARM/LCode) 2AE6: host input quit in leader

604 (LOCAL/LCode) 1800: IH: host output got a quit

605 (WARM/LCode) 338C: no reas block for allocated 8-pkt msg

606 (WARM/LCode) 3252: no allocate to give back

607 (WARM/LCode) 326A: incq or incm with gvb, but no alloc to gb

608 (WARM/LCode) 332E: rstate violation

60A (WARM/LCode) 3174: reply lost-no space

60B (WARM/Warm) 5CFC: HSIOUT: Start pointer write failed

611 (WARM/LCode) 2D24: illegal message blk in hi

619 (FAKREL/FakCode) 5662: ihwq is a mess

61A (CONFIG/RelCode) 5986: BLDHST: BASE/MBLKS wrong for HI/IH

628 (CONFIG/RelCode) 5B56: scrambled host parameter block

640 (WARM/LCode) 2FDA: Block error, no recovery

641 (WARM/LCode) 2FE4: Block error, trying recovery

642 (WARM/LCode) 2FF0: No trnblk for allocate

643 (WARM/LCode) 3096: no trnblk for RFNM or dead RFNM

644 (WARM/Warm) 5BB4: res rep when not resetting

10A (FASTTO/Warm)	43A6:	no working backup RTC
10B (FAKREL/FakCode)	56F2:	CONCLK: IMP number invalid
10C (CONFIG/RelCode)	5846:	RTCCHK: RTC gone away
10D (FAKREL/KakCode)	6E1D:	Node heat prob/or lost bus
110 (CONFIG/RelCode)	5A98:	MTEST: modem hardware gone away
111 (CONFIG/RelCode)	5AE2:	HOTEST: host hardware gone away
112 (CONFIG/RelCode)	5B8C:	TST2DEV: spare interface disappeared
114 (CONFIG/RelCode)	5AA2:	MTEST: swapping modem interfaces
115 (CONFIG/RelCode)	5AEA:	HOTEST: swapping host interfaces
203 (CONFIG/RelCode)	5A0A:	DEVINUSE: over 2 interfaces, one device
204 (CONFIG/RelCode)	574A:	RELCON: removed bad BASE dispatch
205 (CONFIG/RelCode)	5A02:	DEVINUSE: PID for doubled interface differs
206 (CONFIG/RelCode)	593E:	CMODEM: BASE/MBLKS wrong for M2I/I2M
207 (CONFIG/RelCode)	5A68:	BLDBLK: dynamic blocks area full
208 (FAKREL/LCode)	3C0E:	free list in loop
209 (FAKREL/LCode)	3C1E:	lost the free list
20A (IMPSUB/LCode)	1458:	FREGET: threw away free list tail
263 (IMPSUB/LCode)	1354:	map error in flush
264 (IMPSUB/LCode)	14A4:	map error in nwheom
266 (IMPSUB/LCode)	156C:	* map error in deque
267 (IMPSUB/LCode)	1530:	* map error in unpack
281 (FAKREL/FakCode)	527E:	BUFT: Recovered a timed-out buffer
2A2 (IMPSUB/LCode)	158C:	DEQUE: buffer ownership error
2C2 (FAKREL/LCode)	3BF6:	free list buffer error--WHERE nonzero
2C8 (LOCAL/LCode)	1678:	ringc overflow in rstart
2C9 (FAKREL/DDTCode)	550C:	ring structure broken in timeout
2E1 (IMPSUB/LCode)	1448:	FREGET: free list error, non-zero where

2E3 (IMPSUB/LCode)	1366:	tried to flush non-buffer
2E5 (IMPSUB/LCode)	1374:	tried to flush non-owned buffer
2F0 (IMPSUB/LCode)	1598:	fixed half-empty queue
300 (VHA/DDTCode)	5648:	VHAREL: finished VHALIS recomputation
301 (VHA/DDTCode)	556C:	VHAREL: Detected VHA table error
302 (VHA/LCode)	3D40:	IHVHA: No virtual address found
303 (VHA/LCode)	3D6C:	TSKVHA: No virtual address this source
304 (VHA/DDTCode)	55A4:	VHAREL: Too many VHA numbers
305 (VHA/DDTCode)	5594:	VHAREL: VHA IMP number too big
3C0 (FAKREL/FakCode)	5BDE:	imp going down
3C1 (FAKES/FakCode)	485C:	FH2: got setup for no dest
3C2 (FAKES/FakCode)	4814:	FH2: Flushing Reload Packet
3F0 (FAKSUB/FakCode)	4124:	FDOZEW: Initialized Jam Fake Host
3F1 (FAKSUB/FakCode)	4328:	FWAITW: Initialized IMP Fake Host
3F8 (FAKSUB/FakCode)	41B0:	JAMLEADER: Host wanted a buffer
3F9 (FAKSUB/FakCode)	41A6:	JAMLEADER: no host block
3FA (FAKSUB/FakCode)	43BC:	SUCKLEADER: Host sending a buffer
3FB (FAKSUB/FakCode)	4466:	FSUCBUF: Host sending leader
3FC (FAKSUB/FakCode)	4282:	FJAM1B: Host wanted a leader
3FD (FAKSUB/FakCode)	4278:	FJAM1B: No host block?
3FE (FAKSUB/FakCode)	4474:	FSUCBUF: No host block?
3FF (FAKSUB/FakCode)	448C:	FSUCBUF: Bad buffer from IH
401 (MODEM/LCode)	205A:	modem bad end pointer
402 (MODEM/LCode)	1F7E:	modem input got a quit
403 (MODEM/LCode)	2056:	modem input too short
404 (MODEM/LCode)	1C5A:	modem output got quit
405 (MODEM/LCode)	1F14:	I2MXMIT: Start pointer write failed

8	(STAGEK/RelCode)	41EE:	No PIDs in system
9	(STAGEK/LCode)	D1E:	adjusted comrel
A	(STAGEC/RelCode)	5580:	Jiffy clock stopped
B	(STAGEK/LCode)	908:	WSLEEP: System missed a tick
C	(STAGEK/LCode)	E8A:	Quit in cksum parameters
D	(STAGEK/LCode)	E60:	quit during checksumming
E	(STAGEC/RelCode)	503E:	Stage LC: Local code checksum broken
F	(STAGEC/RelCode)	5112:	Stage MM: Not Enough Memory
10	(STAGEK/RelCode)	4264:	Stage variables area quit
11	(STAGEK/LCode)	8C8:	WSLEEP: Lost our communications page
12	(STAGEK/RelCode)	4250:	Stage Common Reinitialization
13	(STAGEK/RelCode)	444E:	Stage CD: Memory Coupler End QUIT
14	(STAGEK/LCode)	670:	hung on bad lock
15	(STAGEK/LCode)	A8E:	Stage LK: can't find a clock
16	(STAGEK/LCode)	4BE:	remote power fail interrupt
17	(STAGEK/LCode)	58A:	Can't find an RTC
18	(STAGEK/RelCode)	49D4:	BBC processor started
19	(STAGEK/RelCode)	45FC:	Buddy Processor started
1A	(STAGEK/RelCode)	44F0:	Stage CD: Bad Processor Identity
1B	(STAGEK/RelCode)	4680:	Block Transfer Timeout
1C	(STAGEK/RelCode)	484A:	BLT proc not in table
1D	(STAGEK/RelCode)	48DC:	Non-existant proc in blt??
1E	(STAGEK/RelCode)	4902:	No I/O bus for BBC
1F	(STAGEK/RelCode)	43F2:	Stage CD: One of my Couplers is broken
20	(STAGEK/RelCode)	49EC:	BBC transfer failure
21	(STAGEK/LCode)	564:	Power restore interrupt
22	(STAGEK/LCode)	55C:	Local Power Fail Interrupt

23 (STAGEK/LCode)	56E:	illegal level 4 interrupt
24 (STAGEK/RelCode)	4228:	Stage variables memory failure
25 (STAGEC/RelCode)	51DE:	Stage MM: Spare page checksum differs
26 (STAGEC/RelCode)	53E6:	fixed bad memory parity
27 (STAGEC/RelCode)	53EE:	solid memory parity error
28 (STAGEK/LCode)	B3C:	SMD: no useable common memory
2A (STAGEC/RelCode)	5558:	QUIT(s) in QUIT handler
2B (STAGEK/LCode)	478:	Quit on instruction fetch
2C (STAGEC/RelCode)	5544:	Quit retry(ies) succeeded
2D (STAGEC/RelCode)	556A:	RTC read retry(ies) succeeded
2F (STAGEC/RelCode)	537A:	Stage MM: Copy/clear failed
40 (XSIOIN/LCode)	F00:	XSIOIN: Start pointer write failed
41 (XSIOIN/LCode)	EF6:	XSIOIN: 4 start pointer failures
42 (OPSYS/LCode)	106C:	got illegal pid value
43 (OPSYS/LCode)	110A:	LOOP: map error
44 (OPSYS/LCode)	10C2:	LOOP: LSTACK overflow
45 (OPSYS/LCode)	1286:	INBASE failed
50 (STAGEC/RelCode)	5658:	SARWDG: software watchdog timer expired
C0 (DDT/DDTCode)	416A:	TTYINI: tty changed psbs
100 (FAKREL/FakCode)	5AB0:	IMP reinitialization
101 (FAKREL/FakCode)	524E:	smashed a buffer pointer
102 (CONFIG/RelCode)	57E4:	RELCON: Changing buffer page allocation
102 (FAKREL/FakCode)	590A:	FAKINI: stopped pktcore
103 (CONFIG/RelCode)	5B7A:	TST2DEV: Swapping to F device
104 (STO/LCode)	1B7A:	STO: lock timed out
108 (FASTTO/Warm)	4384:	main clock has stopped
109 (FASTTO/Warm)	435E:	backup clock working again

Snaps:

R4 - address of parameter block

R6 - address of the buffer (through MAP2)

(map in UMAP area)

R1 - length that the hardware said it gave us.

--FLUSH INPUT AND RETRY. RESETS INTERFACE

404 Modem output got a quit

The modem output status register reported a quit.

Snaps:

R3 - output status register we read

R4 - modem parameter block address

R5 - device interface address

--CONTINUES WITH NEXT INPUT

405 Start Pointer Write Failed

Snaps:

R4 - start of modem parameter block for the offending

modem (or pair).

--FLUSH INPUT

410 Modem software checksum failure

The systems software detected a checksum error on a packet (after the packet apparently was accepted by hardware checksum logic). This could indicate a software problem, but most likely is the result of a bad DMA card in the modem

interface.

5C5 Master Slave Mismatch.

Usually caused when lower number IMP on other end of line fails to hear our "HELLO" packets. It could then be our modem output failing or the other IMP's modem input failing.

Snaps:

R4 - Points to the modem parameter block.

602 Host input got a quit

Host input detected a quit during the input of data.

Snaps:

R1 - receive status

R4 - host parameter block

R5 - host interface address

--RETURNS "ERROR DURING DATA" MESSAGE TO HOST

603 Host input quit in leader

The hardware reported a quit while we were reading in the leader.

Snaps:

R2 - receive status

R4 - host parameter block

R5 - host interface address

--RETURNS "ERROR IN LEADER" MESSAGE TO HOST

604 Host output got a quit

The host output hardware reported that it got a quit. The host is reset, and host ready line flapped to indicate the failure.

Snaps:

R3 - transmit status

R4 - host parameter block

R5 - host interface address

--FLAPS HOST READY LINE AND RESETS SOFTWARE

6A0 Error during host input data

This says the error bit came on in the host receive end pointer (bit 0). The usual cause is either the host or the IMP dropping its ready line while active.

Snaps:

R7 - end pointer we got

R4 - parameter block of host

R5 - interface hardware address

--FLUSHES THE MESSAGE AND GIVES HOST AN "ERROR DURING DATA" MESSAGE.

FC8 Host sent error with ID

This says that the host computer thinks his ready line (imp ready) flapped while he was reading his leader in. Although this is specified in 1822, it is very unlikely that any real hosts will do it.

9.3 TRAP LOCATIONS FOR IMP <1200> (NOT NECESSARILY TRUE FOR PSE)

The following is a list of Pluribus traps. For IMPs on the PLATFORM the hex trap number is reported to the NMC. For machines not on the net, the traps are displayed at the bottom of the terminal. When the Pluribus times out a software lock, its address is reported as if it were a trap. The locks are listed following the traps. Note - some locks are contained in dynamically-allocated parameter blocks; thus, their addresses depend on the individual machine configurations. If you need to find out what dynamic lock has timed out, ask for help from a software person.

The names in parentheses below are the source file name and the logical page. The logical page information is used for obtaining the correct common memory page if you need to patch a trap for any reason. (see Section 2.1.)

Page Trap (Source/Page) Loc: Description

Trap (Source/Page) Loc: Description

1 (STAGEK/LCode)	426:	Unexpected Quit
2 (STAGEK/LCode)	5CE:	program in a loop
3 (STAGEC/RelCode)	5340:	Stage MM: Completed memory management
4 (STAGEK/LCode)	58E:	local clock stopped
5 (STAGEK/LCode)	A76:	Local Kernel Checksum Broken
6 (STAGEK/LCode)	9DC:	unexpected interrupt
7 (STAGEK/RelCode)	44B2:	Stage CD: BBC map failure