Bolt Beranek and Newman Inc.

Technical Information Report No. 99

The ARPANET Pluribus IMP Program

Volume II: DDT, Program Descriptions, Data Formats

May 1978

Prepared for:
Defense Communications Agency

Bolt Beranek and Newman Inc.

Technical Information Report No. 99

THE ARPANET PLURIBUS IMP PROGRAM

Volume II

DDT, Program Descriptions, Data Formats

May 1978

Volume II

Table of Contents

Volume II

Table of Tables

Foreword

This document forms Volume II of a two-volume set which describes the Pluribus IMP program. The first volume contains descriptions of the major routines in the IMP system. Discussions in Volume I are high-level and that volume is intended to be self-contained. Volume II contains detailed descriptions of the programs that comprise the IMP system. Chapter 4 describes the DDT debugging system, Chapter 5 discusses the various program modules of the IMP system in detail, and Chapter 6 contains pictoral descriptions of the data structures used. Volume II is intended for system implementors, and should be read only with a thorough understanding of all the discussions of Volume I and with the IMP program listing handy.

Chapter 4
The Debugging System DDT

The Pluribus IMP provides within itself the diagnostic debugging system DDT to facilitate dealing with problems that might arise. The implementation makes it possible to debug the Pluribus IMP either from the terminal attached to the IMP or from a remote location. Both the local terminal handler process and the DDT process are implemented as fake Hosts within the IMP. These Hosts are "cross-patched" when the system is initialized, so that characters typed at the terminal are sent to DDT, and DDT responses print on the terminal. DDT commands are provided to reconnect DDT to another Host anywhere in the network to permit remote debugging. Further details about these fake Hosts are in sections 2.2.6.1, 2.2.6.2, and in 5.9.1-5.9.4.

DDT is a program which provides a mechanism for inspecting and changing registers of the machine. In a broader sense, however, it can be viewed as a simple operating system which controls the starting and stopping of processors and handles extraordinary conditions (QUIT and ILLOP). This section is not intended as a tutorial; some knowledge of how other DDTs work (see, for example, "DDT-10 Programmer's Reference Manual," Digital Equipment Corporation, Maynard, Massachusetts, copyright 1968, 1969, 1970) may be helpful.

Pluribus DDT versions have been developed for different configurations and applications; only the IMP version is discussed here. Regardless of the internal structure, all versions appear basically the same to the user. DDT requires a controlling device, such as a Teletype or VISTAR. Pluribus IMP DDT runs in conjunction with the STAGE subsystem. As such, it may run only when at least the STAGE Kernel is in operation (i.e. Stages LK through RC are running). DDT makes use of the Block Transfer routine to examine and deposit words in memory. Modifying locations within checksummed code automatically updates the proper checksum to permit patching of the program from DDT.

Following are descriptions of the various commands the user may type. A number is represented by "nn", and <altmode> (or <escape>) is represented by "$". A dollar-sign character is indicated by "<dollar>". A caret or uparrow "^" followed by a letter indicates a control character. The character caret (or uparrow) is indicated "<uparr>". The underscore or backarrow character is indicated by "<backarr>". The carriage return character is denoted "<cr>", and linefeed "<lf>". The word "register" generally means a location in address space; a

"processor register" is just that. Numbers are followed by "!" to indicate that they are hexadecimal (base 16).

4.1 DDT Command Summaries

4.1.1 Addresses, Opening and Closing

Whenever a register is "opened", its contents are typed out in the current mode (except as noted for certain commands). When a register is "closed", the last value typed in while open, if any, is written to that register. If nothing or <delete> is typed in, nothing is written.

nn/ Opens register nn. The processor in whose address space the reference was made types out as "Pnn" immediately following the /. The contents of the requested register then are typed in the current type out mode.

nn,nn,nn/ The first two arguments specify a processor or processors to do the reference, and a map setting if needed. The processor(s) may be specified as for the ":" command (see below). A single processor or any set of processors is permissible, and the command has the same effect as if "nn:" or "Pnn:" were typed prior to opening the location. If the address requested is a mapped reference (i.e., the address is in the range 4000-BFFF!), the middle argument is used to set the appropriate DDT pseudo-map. The effect is identical to explicit map-setting using the proper "nn,nn^F" command (see below). Either of the first two arguments may be omitted; the default is to leave the processor selection and map settings as they were prior to the command.

Rnn/ Opens processor register nn. For processors running STAGE, simulated processor registers are used, which contain the actual processor register values at the last occurence of a snapshot-triggering trap. Refer to the discussion of the DDT debugging environment below. For processors not running STAGE, the actual processor register is referenced. If the processor is running another program, all references to its processor registers result in QUITs, except for R15. The latter is the processor control register and is always visible, although references to it may cause the processor to halt.

<cr> Closes current register, if any open.

<lf> Closes current register, if any open, and opens next "instruction"; that is, if type out mode is symbolic (see below) and the current register is a double-word instruction, skip one register.

$<lf> Same as <lf> but always opens the next register; that is, a register is never skipped.

<uparr> Closes current register, if any open, and opens the previous one.

$<uparr> Like <uparr> but goes up two registers, not one.

. By itself, is the value of the address of the current register, if any open; if none, then the last current register.

/ Types out the contents of the register addressed by the current register but does not open it or change ".". The address used is the "effective" address of the symbolic instruction, if the current type-out was symbolic. For instructions that have no effective address (HLT for example), or for type out in constant or ASCII mode, the actual memory contents are used for the "effective" address.

$/ Closes the current register and opens the register addressed by the current register, as in "/".

4.1.2 Type Out Modes

There are two orthogonal type out modes. One controls the radix of type out:

^H Numbers are typed out in hexadecimal (base 16) - the default.

$^O Numbers are typed out in octal (base 8).

The other controls how register contents are interpreted:

^S Type out symbolically, that is, try to interpret as an instruction, including next word if a two-word instruction code.

^K (Konstant) type out as a number.

^A Type out as two ASCII characters.

4.1.3 Other Type Out Commands

= Retypes out the current register in the alternate mode as follows:

current	alternate
symbolic	constant
constant	symbolic
ASCII	constant

$= Retypes out the current register in the alternate mode, as in "=", and changes the current mode to the alternate mode.

nn= When preceded by a number or an expression, types out the value of that expression. The result of such expression arithmetic is not considered a value to be written to an open register when closed.

nn" opens location nn, but does not type out contents; remains in this mode until / or \ is typed.

$" analogous to $/

" analogous to /

nn\ opens location nn, but the address type out is suppressed on succeeding lines until / or " is typed.

$\ analogous to $/

\ analogous to /

nn[opens location nn, but types out contents in the alternate mode (see =, above); does not change current mode.

$[analogous to $/

[analogous to /

4.1.4 Type In

symbols — DDT contains symbols with predefined values to facilitate type in of symbolic data. All of the op codes and other instruction components of the Pluribus assembler are appropriately defined. By using <space> and/or <tab>, instructions may be entered in virtually the same format as the assembler expects. Type in routines correctly interpret displacements in branch instructions. Malformed instructions result in the type out "#", and all current type in is cancelled. There is presently no facility for user defined symbols.

NOTE: The characters <comma>, "=", "#", "+", "-", "(", and ")" have special meaning within an instruction type in, as do the symbols R0, R1, ... R7. Refer to BBN Report No. 3001, Pluribus Document 4, Basic Software, Part 2, for a description of the Pluribus assembler format.

nn — Typed in numbers are generally interpreted according to the current type out radix, except that numbers containing letters A-F are always hexadecimal. Note that some numbers look just like symbols; e.g., ADD, ADDB, BC, BE, BF1, BF2, and BF3. These are treated as symbols unless they are explicitly denoted as numbers by a leading zero or by an "!" after the number. It is a good habit to precede all hexadecimal numbers beginning with the letters A-F by a leading 0.

nn. — a decimal number

nn' — an octal number

nn! — a hexadecimal number

<delete> — echoes as "#" and cancels current input, that is, it is as if whatever is being typed in was never typed.

+ — addition

<space> — addition

<tab> — addition

- subtraction

<backarr> has the value of the last quantity typed out as a result of examining a register. This would be the value of second word of a two-word instruction when in symbolic mode. If the value of the first word is desired, use "=" followed by <backarr>.

<comma> is used to input two words at a time. Typing <comma> after the first value saves that value until the terminator is typed after the second value, then both values are written to memory. The value of "." is not changed. A <delete> typed after the <comma> aborts the entire input. If nothing is typed before the <comma>, only the second word is changed.

4.1.5 Address Spaces

Pnn: sets the number of the current processor address space. The processor number is specified according to the Pluribus convention that assigns coupler addresses to indicate the physical processor position in the machine. If the processor doesn't exist, or is inaccessible, subsequent references to its address space produce the "WHO?" diagnostic. No checking is done at the time the ":" is typed.

nn: selects a set of processors, according to the mask given in nn. Bits correspond to processors in the system, assigned right-to-left in increasing order by processor number. If nn is 0 or the character "-" or not specified at all, the mask is set to be all processors currently running STAGE. Read references go to the first processor that runs the Block Transfer subroutine; if no processor running STAGE is in the mask, then some processor in the mask is accessed by its buddy (if the buddy is running STAGE), or the lowest-numbered processor in the mask is accessed by backwards bus coupling. The identity of the processor in whose address space the location was examined types after the "/" (see above). Write references are performed by all processors in the mask, allowing simultaneous patching of all processor local memories.

nn,nn^F sets the map value of the memory page to be referenced when examining addresses in the mappable segments

(4000!-BFFF!) to the second argument. The map setting is maintained internally to DDT, but is used for all subsequent references through the corresponding address window until changed by another ^F or / command. One of the four segments is specified by the first argument, which must be "M0", "M1", "M2" or "M3" to select a particular 4K map window. If the first argument is missing, map 0 is assumed. The map setting is interpreted as follows: (1) even numbers less than 200! select a logical page type, as maintained by Stage MM. (2) even numbers 200! or greater select the physical page with that map setting. (3) 0 selects logical page 0 (the "reliability" page). (4) Any odd number selects a physical page; in particular, 1 selects physical page 0. An argument of -1 causes all four maps to be set to their default values, which are 0,10,12,10 for the reliability, variables, second variables, and variables (logical) pages. The current map settings may be examined by opening locations M0-M3 (i.e., type "M0/" to see the current setting for references to locations 4000-5FFF).

4.1.6 Control

nn^G starts the selected processor at nn. If the selected processor is running STAGE, nn is copied to its simulated R0 and its R15 is set to 2.

^G starts the selected processor at the address last specified by a ^G command. If no argument is given and no address has been specified previously, a "#" prints and nothing else happens.

^X stops the selected processor if running and types out the contents of the program counter. If not running, types out "HALTED". For processors running in the STAGE system, sets their simulated R15 to a 1, and prints the contents of simulated R0.

^P Causes the selected processor to proceed from its current state. If a processor running STAGE is selected, its R15 is set to 2.

^Z steps the selected processor one instruction and types that instruction. Prcoessors running STAGE set their R15 to 3. In this case, single-stepping is meaningless and should not be tried.

$^Z like ^Z but does not type the instruction.

nn^Z like ^Z but first sets the program counter to nn.

4.1.7 IMP Version Features

nn^L sets up a leader for a "semicolon message". Format is "<leader1>, <leader2>, <leader3>, <leader4>, <leader5>^L". Any field not specified is assumed to be zero except for <leader1>, which is set to 0F00. Thus, the command 3,6,^L sets up a normal message leader for semicolon messages to Host 3 on IMP 6, with no special leader flags and a message-id of 0.

^L clears the screen and repaints the display in the bottom portion of the screen.

nn,nn^C "crosspatches" all subsequent type in. The second argument is the IMP to send to; the first is which Host on that IMP. Host 0FD (i.e., the DDT fake Host) is assumed if the first argument is omitted.

^@ undoes ^C and directs type in to the local DDT - echoes a <cr>, <lf> pair.

^T returns as a value in the current radix the number of the last IMP Teletype to "crosspatch" to the local Teletype.

^O complements value of the override switch and echoes "^ON" or "^OF" as appropriate.

nn^O same as ^O for "sense switch" nn (1-4). The Pluribus has no physical sense switches; software sense switches are maintained in this fashion, but currently have no function.

nn<dollar> activates the operator help (OPHELP) command nn. These commands are various maintenance and debugging aids which enable day-to-day operation of the IMP system without detailed knowledge of the program. For ex
ample, OPHELP commands can loop and unloop modem lines, or look up the address of a Host parameter block.

4.1.8 Miscellaneous Commands

nn,nn,nn^B copies contents of some processor's private memory to the corresponding locations of the private memory of the currently selected processor(s). The first argument selects which processor(s) should be the source of the transfer, where nn is interpreted as in the nn: command (see above). Pnn: may be used to copy from one specific processor. The latter two arguments give the inclusive bounds on the addresses to be copied. Omitting arguments causes the last value previously specified for that field to be used; if none exists, a "#" is echoed and no copy takes place.

4.2 Control Structure of DDT

The Teletype handler process and the DDT process are polled from the operational IMP program when they have work to do (i.e., characters to process). The Teletype process is also called periodically to check for new input characters from the Teletype interface. Since crosspatching allows the local Teletype to send its characters to another IMP, and other IMPs or Hosts may send characters to the DDT process, the two processes must be independent. The ^C and ^@ commands provide control over the crosspatching of the local IMP Teletype.

DDT makes use of the Stage Block Transfer process to perform operations that involve moving data in memory. The various Examine and Deposit functions are such operations, as are the ^X, ^G, ^P, ^Z, and ^B commands. Failures within the Block Transfer process set error codes, which DDT attempts to interpret. Table 1 shows the DDT error type outs and their meanings.

The Teletype and DDT processes communicate only by passing characters through buffers. The Fake Host processes responsible for communicating these characters to and from the network manage the other side of these buffers. When the system is in debugging mode, and the IMP processes stop passing these characters for several seconds, a special mode is entered which permits the characters to be passed directly from Teletype to DDT and back. In this mode, the IMP Fake Host processes are bypassed (since they may have stopped running anyway). Refer to the discussion of debugging mode below.

Type-out	Meaning
QUIT	Block Transfer got a QUIT trying to complete the last request. If the request was performed by several processors, the mask specifying which processors got the QUIT will print in parentheses. A mask of FFFF indicates that the single processor which was selected could not complete the transfer.
FAILED	Block Transfer couldn't complete the transfer in the requested processor's address space. Again, a mask saying which processor(s) had trouble will be printed.
WHO?	Some non-existant processor was specified to Block Transfer. The mask which prints says which processor; if a single processor is selected, a mask of FFFF will print and that processor doesn't exist.
TIMEOUT	Block Transfer timed out before the requested transfer could complete. This shouldn't happen; if it does, it indicates a program bug or perhaps a very busy machine.
???	A Block Transfer error has happened which DDT can't decode. This should never occur; it would indicate a program bug in DDT.

Table 1
DDT Error Type-Outs.

4.3 Protection, Override

DDT has the power to change any location in memory in such a way that it cannot be detected (by a failing checksum, for example). As such, it could completely disable the continued operation of the operational IMP system. For this reason, all potentially dangerous actions of DDT are protected by the override mechanism. Normally, the override condition is disabled in a running IMP system. Only certain Hosts (the NCC IMP Teletype Fake Host, for example) may turn on override. Override is always disabled following a system restart. Once override is enabled, all of DDT's features are permitted, so extreme care must be used that incorrect actions are not requested. As soon as the necessary requests to DDT have been completed, override should be immediately disabled. Override is controlled by the ^O command.

Commands that can modify memory include a direct request to change an open memory location with <lf> or <cr> (although memory may be examined with override off), the <dollar> commands, the processor-control commands (^X, ^P, ^G, and ^Z), the local memory copy command ^B, and the crosspatch command ^C. Examining memory and typing <lf> to progress to the next location is permissible with override off (since nothing was typed in to be deposited).

4.4 Debugging Environment of DDT

DDT attempts to maintain a "logical" debugging environment similar to the environment the programmer is coding in when he/she is writing the program. In particular, DDT assigns special meanings to such hardware features as the memory map registers (and their effects) and the processor registers. The debugger can thus simulate step-by-step the action of a particular routine by changing the contents of these hardware registers in DDT. DDT, of course, does not change the actual registers, since it is using the registers for its own purposes. Instead, the registers and (in the case of the maps) their side-effects are simulated by DDT.

The simulated or pseudo-map registers can be accessed in DDT, either explicitly by opening locations M0, M1, M2, or M3 (or, equivalently, 0FC00!, 0FC02!, 0FC04! or 0FC06!), or implicilty in the nn,nn/ and ^F commands. The current settings in DDT's simulated registers may be examined by opening locations M0-M3. (The hardware map settings cannot be read directly!)

The processor registers can also be examined for processors running in the system. Of course, since the processor is running at the time, these have no meaning. Instead, DDT displays the registers at the time of the last "snapshot". A snapshot is triggered by a extraordinary occurence in the IMP program. An unexpected QUIT is such an occurence. In many places, traps that are not supposed to happen ever, and in particular traps that may indicate some hardware malfunction, are transformed into snapshot traps (see description of illegal instruction interrupt handler, section 5.1.14). The snapshot consists of a complete picture of the processor registers at the time of the snapshot, plus other interesting information that may have some bearing on the problem at hand. Snapshots are sent periodically to TENEX by the PLOG process when debugging mode is disabled, so that a permanent record of them may be kept and the snapshot area may be reused. Table 2 lists the simulated registers and their meanings.

Register	Contents
R0	The illegal instruction (ILLOPR) that triggered this snapshot. Refer to the trap listing for the IMP for its meaning.
R1-R7	The contents of processor registers 1-7 at the time the snapshot trap was executed. Refer to the program listing to discover their meaning at the time of this particular trap.
R8	The processor status register at the time of the trap.
R9	The contents of the program counter at the time of the trap. (i.e., the memory location containing the trap instruction)
R10	The last call to the Stage restart WST, WSTCOM, or WS. Encodes the reason for the last restart of this processor.
R11	The contents of the program counter the last time a "program in a loop" condition was detected.
R12	The address being referenced tha last time an unexpected QUIT occured.
R13	The contents of the status register at the last unexpected QUIT.
R14	The program counter at the last unexpected QUIT.
R15	The simulated processor control register. For more discussion, see the section on debugging mode.
170	(This location is right after R15; a <lf> after examining R15 will examine this location). The map0 setting at the last snapshot.
172-6	Maps 1-3 at the last snapshot. The map settings will always have the 2-bit or'ed in.

Table 2
Simulated Processor Registers.

4.5 Debugging Mode

Debugging mode is a special IMP program feature which should never be required during normal operation. It allows processors to "hang" in the early portions of STAGE under various drastic error conditions, so that the state of a computation can be examined in DDT after the occurence of the error. This facility is much like the notion of a "breakpoint" in many other DDT's. In the Pluribus, however, we have modified the notion somewhat because of the demands of the multiprocessor environment. In particular, DDT may only run when enough of STAGE is running to guarantee proper communication between the processors in the system. In other words, enough of STAGE must be running to ensure proper operation of the Block Transfer process. This comprises the "kernel" of STAGE, or Stages LK through RC.

Once these Stages are enabled, the Stage dispatcher checks the debugging mode flag. This is a flag within the local kernel checksum bit-coded by processor - if the bit corresponding to a processor is set, it is in debugging mode. Processors in debugging mode will hang at Stage RC if their simulated "halt" bit (bit 1 in R15) is set. They continue to poll Block Transfer and the Teletype and DDT processes, however, so that the state of the system may be examined by the programmer. There are known dead states in debugging mode! The DDT code page, for example, is not checked before the Teletype and DDT routines are called, so a problem in those routines could cause processors to execute a halt or worse. Use of debugging mode is dangerous, and is only intended for checkout of new systems on machines not performing an operational function. In a redundant system, debugging mode could possibly be used selectively to diagnose a failure in only one processor or processor bus, but extreme caution is advised.

In debugging mode, Stage interprets the processor "Halt" bit (i.e., the 1-bit in snapshot register R15) to mean "stop running the system and all stages later than stage RC". When all is normal, the "run" bit (the 2-bit) will be set. An occurence of a snapshot trap will turn off the run bit, but not set the halt bit (this is the "half-halted" state). A serious condition (an unexpected QUIT, or the special halt traps 0FFFF and 0FADE) will set the halt bit, and this processor will cease running the operational system. In addition, extra bits are set to indicate which condition caused the processor to hang in STAGE, and the reason will print out as "Pnn xxxx@yyyy", where xxxx is the error condition and yyyy is the location of the snapshot trap that caused the halt. The conditions and their meanings are listed in Table 3.

Type-Out	Meaning
QUIT	The processor got an unexpected QUIT at the instruction whose address is typed in the error print out.
ILOPR	The processor executed a 0FFFF instruction at the specified location.
FADE	The processor executed the special illegal instruction 0FADE at the specified location. This causes a flag to be set in common variables which in turn forces all the other processors also to hang in STAGE. This is useful when debugging a problem requires looking at the global environment of the system. Each processor checks the flag as it completes the strip it was running.

Table 3
Debugging Mode Halt Type-Outs.

NOTE: In the "halted" state, Stages LK-RC continue to run. This means that debugging those Stages is very touchy with the use of the 0FFFF or 0FADE traps. Programmers are advised to try to use snapshot traps wherever possible for debugging, and to resort to the halt traps only rarely. In most cases, enough useful data can be loaded into the registers at the time of the snapshot to permit effective debugging of bad conditions.

The snapshot area, once it contains a valid snapshot, will only be overwritten by the serious condition type of snapshot. The serious snapshot will never be overwritten, but the processor must have its "halt" bit turned off before it reenters the system. The overwriting is controlled by looking at the stets of the simulated R15 run and halt bits. Snapshots are permissable which progress from "run" state to "half-halted" or "halted" state, and from "half-halted" state to "halted" state. Causing processors to leave either "half-halted" or "halted" state is achieved by typing ^P with the appropriate processor(s) selected. This sets their R15 back into "run" state to permit more snapshots. When not in debugging mode, the TLOG process, as it sends off snapshot messages, turns the "run" bit (bit 2) back on to allow more snapshots to take place.

Chapter 5
Detailed Program Descriptions

A concise, systematic approach has been taken in presenting details of the IMP programs in the following pages. The approach is reflected by the headings of the outline used in describing the programs:

1. Function - for complex routines, each function is numbered for reference in subsequent sections. The list of functions contains those which are fundamental to major IMP operations.

2. Control Structure - A general description of the coding structure and its control flow.
   a. Entry points - locations and modes by which the program is entered.
   b. External calls - the names of subroutine or coroutine calls which the program makes.
   c. Initialization - important settings made during the initialization process.
   d. Cleanup - actions taken before exiting or during unusual situations.

3. Data Structures.
   a. Local data - variables and constants which are used only by the program.
   b. Shared - tables, variables and locks which are used by other programs as well. Care must be taken to use software locks wisely so as to insure consistency in shared data.

4. I/O Performed - Any actions taken which affect the state of any hardware interface in the IMP. Included are resets as well as input and output operations.

5.1 Stage System

Function

The Stage system is a series of processes which progressively build up a picture of the Pluribus machine configuration. The output is typically in tables which may be used by operational systems (such as the IMP program) to initiate and perform their tasks. Each of the processes is described in a subsection of this section. Dispatch control for initiating these processes is described in this section. Other basic system functions performed by the Stage routines are described in subsections following the individual Stages.

Control Structure

The Stage dispatcher polls individual Stages which have been enabled, under control of the WDIS word. In addition, the Block Transfer process is polled and, if in debugging mode, the Teletype and DDT processes. The dispatcher also maintains the Stage timing functions and consensus arrays for each Stage.

Entry Points

Stage is the first system entered on start-up or after a drastic system failure. Following are various entries to Stage, and their uses:

SETUP: a halt, so pushing RUN on the operator console proceeds at WS.

WS: restart entry, to force total system reinitialization; normal entry following a fresh paper-tape reload.

WSTINI: entry for processor that is restarted by another processor; sets up PROCNO and enters WST.

SJ7: entry if this processor believes the communication page has failed.

WSTCOM: entry to force finding of a (possibly new) communication page.

WST: entry following certain drastic local failures, such as unexpected QUITs or program loops.

External Calls

SBAD: routine to disable later Stages; called on initial entry to disable all but the first Stage (Stage LK).

STPOLL: entry to poll DDT and Teletype from Stage, for debugging only.

BLT: Block Transfer process; provides mechanism for processor reloads and restarts, DDT examine and deposit, and packet core transfers.

Initialization

Each Stage, as it succeeds, initializaes the dispatch (WSLA7) for the succeeding Stage.

Cleanup

All hardware configuration data is periodically checked and kept correct by the Stage processes.

Data Structures

Local Data

UWST: saved address of last caller to initial entry to Stage.
WSTAGE: currently running Stage index.
WTEMP: temporary register save word.
WSLA1-WSLA7: tables (by Stage) of saved register settings.
SVTIME: most recent time read from real time clock Stage is using (LCLOCK).

Shared Data

LCLOCK: address of current real-time clock for Stage to use.
UTIME: time next to run Stage LK (so all Stages run occasionally).
WDIS: Stage dispatch control; bit-coded by Stage.
CONSOL: address of operator console light registers, if any.
LTIME: time (from LCLOCK RTC) before which the Stage dispatcher should next be entered.
DEBUGM: debugging mode flag.
PROCBT: bit for this processor.
SNAP: snapshot save area.
MAPREL, MAPCOM, MAPVAR: map settings for reliability, communication, and variables pages.
STIME: local copy of Stage system time (SYTIME).
COMPTR: pointer from each common memory page to the communication page.
SYTIME, SYTIM2: time (25.6 ms ticks if IMP system running), controls many Stage timeout functions.
COMREL: communication page pointer to reliability (rely) page.
SEGCON, STGCON, BUSCON, COUCON, RCKCON, LOCCON, CKSCON, MEMCON, IOCON, INICON: consensus arrays for Stages MD, RK, BD, CD, RC, LC, MC, MM, ID, AR.

I/O Performed

While Stage is running, WDIS is displayed in the console address lights. If an external reload is in progress, the packet core address (PKCADD) is displayed in the console data lights by BLT.

5.1.1 Stage LK - Local Kernel Checks

Function

This Stage operates on configuration data local to each processor. Upon successful completion, the processor has initialized its interrupt dispatches (for QUITS, power fails and restores, and the 60-hertz ("jiffy") clock interrupts), discovered its operator console (if any), checksummed the local kernel code, and found a system real time clock (RTC) to use for timing Stage.

Control Structure

Entered on any Stage restart, and then runs forever as a coroutine with the Stage dispatcher.

Entry Points

Restarts enter at SLK00.

External Calls

CKSUBI, CKSUBR to checksum the local kernel code.
SLKSLE to dismiss coroutine (special entry to WSLEEP).
FNDCLK to search for a real time clock.
SOKAY to signal successful completion of this Stage.

Initialization

None.

Cleanup

None.

Data Structures

Local Data

None.

Shared Data

UQUIT: pointer to unexpected QUIT handler.
CONSOL: address of operator console, if any.
PROCNO: processor number for thie processor.
PROCBT: bit assigned to this processor.
LCLOCK: address of RTC for Stage to use.

I/O Performed

Processor interrupt vectors are initialized and level 1 and 4 interrupts are enabled.

5.1.2 Stage MD - Common Memory Discovery

Function

This Stage searches the common memory space to find usable common memories. In addition, it must maintain the identity of the communication page, which must be the lowest numbered page the processor can see.

Control Structure

Searches through all possible memory from map 0 to map 7E00!. Each memory is tested for proper operation with a brief memory test. The lowest page found is used as a communication page.

Entry Points

Coroutine, entered from Stage Dispatcher via WSLA7.

External Calls

WSLEEP to dismiss coroutine.
SFIXIT to reach consensus for modifying MEMSEG.
SCLEAR to remove this processor's FIXIT bit.
SOKAY to allow next Stage to run.
SBAD to disable later Stages.

Initialization

WSLA7 is initialized to dispatch to SMD00.

Cleanup

None.

Data Structures

Local Data

MYSEGS: this processor's version of common memory table.
WMLOCK: lock on memory test area.
SMDFLG: flag for timing COMTST.
SMDTIM: timer for COMTST counters.
SMDBUC: junk word (bit bucket) to store into to find memory.
SMDBLK: memory test area.
COMTST: bit-coded (by processor) timers to fix a COMPTR word.

Shared Data

MAPCOM: map setting for communication page.
MEMSEG: common memory table (bit-coded).
MEMTOT: total number of 4K memory pages found.

STIME: Stage time (local copy).
SLFPTR: page self-pointer (copy of map setting).
COMPTR: pointer from each page to the communication page.
PROCBT: processor bit for this processor.

I/O Performed

None.

5.1.3 Stage RK - Reliability Page Kernel Check

Function

This Stage searches through common memory for a viable common memory kernel. The kernel, and the whole page, are checksummed. The result is recorded (via consensus) in COMREL on the communication page.

Control Structure

Searches through memory pages discovered by Stage MD. Existence of a kernel is denoted by the password 0ACE! being stored at location RKEPAS. Checksums of the kernel, and the whole (reliability) page are verified.

Entry Point

Coroutine, dispatch from Stage dispatcher via WSLA7.

External Calls

SCLROK to pass the Stage and remove fix-it bit.
WSLEEP to dismiss the coroutine.
MEMTS2 to check for existence of a memory page.
CKSUBI, CKSUBR to compute checksums.
SFXBAD to hang this Stage and await consensus for modifying COMREL.

Initialization

WSLA7 is initialized to dispatch to SRK00.

Cleanup

None.

Data Structures

Local Data

SRKKER: kernel that has been found by this stage, if any.
RKEPAS: location of password constant identifying the kernel.
RKERCK: checksum and limit words for kernel checksum.
SRKREL: copy of COMREL, used to limit search loop.

Shared Data

COMREL: pointer from communication page to reliability page.
CKSUM: checksum and limit words for common code page.
TYPE4K: core type word for common code page.

I/O Performed

None.

5.1.4 Stage BD - Common Bus Discovery

Function

Check variables area in common memory for this and later Stages for bad parity and currency; reinitialize it if there is bad parity or if the variables are no longer current. Find which common busses exist (for I/O or memmory). Note which I/O busses have real time clocks. Results are recorded in USEBUS.

Control Structure

Scans the common variables area for bad parity, which is indicated by the occurrence of a QUIT. Checks for existence of common busses. I/O busses are checked to see if they have a PID and real time clock. Memory busses exist if their lowest-numbered page is in MEMSEG.

Entry Points

Coroutine, called from Stage dispatcher via WSLA7.

External Calls

SCLROK to pass the Stage and remove fix-it bit.
WSLEEP to dismiss the coroutine.
SFXBAD to hang this Stage and await fix-it consensus.
WST to restart Stage.
WSTMEM to remove a failing memory page from usage and restart Stage.

Initialization

WSLA7 is initialized to dispatch to SBD00.

Cleanup

None.

Data Structures

Local Data

STGTIM: timeout on the Stage common variables area.

Shared Data

COMAR: Stage common variables area.
BBCLOK: lock on backwards bus coupling privelege.
BLTLOK: lock on Block Transfer parameters.
BUSCON, COUCON, RCKCON, LOCCON, CKSCON, MEMCON, IOCON, INICON: Consensus arrays for Stages BD, CD, RC, LC, MC, MM, ID, AR.
STIME: local copy of system time.

MEMSEG: table of existing memory pages from Stage MD.
PIDGET: local table of PID read addresses.
USEBUS: bit-coded table of existing common busses and real time clocks.
LKERCK: local kernel code checksum.
MAPREL: map for reliability code page.

I/O Performed

None.

5.1.5 Stage CD - Bus Coupler Discovery

Function

Search for all couplers in the system (processor to memory, processor to I/O, and I/O to memory). Search for all processors in the system.

Control Structure

Scans all of coupler addresses on I/O and memory busses. Couplers appearing on I/O busses must be from processor busses; if memory busses have couplers that correspond, they are assumed to be from processor busses also. Couplers appearing only on memory busses are assumed to be from I/O busses, and are remembered separately. Processors are assigned numbers based on their coupler addresses (which generally correspond to their physical location in the Pluribus machine).

Entry Points

Coroutine, called from Stage dispatcher through WSLA7.

External Calls

SFIXIT to achieve consensus for modifying results of this Stage.
SCLROK to pass this Stage and remove this processor's fix-it bit.
WSLEEP to dismiss coroutine.
SFXBAD to hang this Stage and await consensus to fix results.
WST to restart Stage if this processor discovers its own number was wrong.

initialization

WSLA7 is initialized to SCD00.

Cleanup

None.

Data Structures

Local Data

PROCD: accumulates set of existing processors.
SCDIOI: current index to non-processor coupler table.
SCDBUS: accumulates bits by bus for which common busses this coupler exists on.
AMPCOM: table of common bus amputation states.
COUBUS: which busses each processor has couplers on.

IOCTBL: table of non-processor couplers and which (memory) busses they exist on.

Shared Data

PROCEX: bit-coded table of existing processors.
PROCNO: my own processor index in COUTAB.
COUTAB: table of processor coupler addresses.
MYPROC: my processor number = coupler index (bus address) plus odd bit if I'm the odd processor.
PROIOR: bit table of processors that should be removed from system.
AMPROC: amputate words by processor coupler.
SEGCON: Stage MD consensus.
USEBUS: bit-coded table of existing busses.
BBCLOK: lock on backwards bus coupling privelege.
QUITV: pointer to most recent QUIT vector used.

I/O Performed

None.

5.1.6 Stage RC - Reliability Page Check

Function

Decides (based on result of Stage RK) whether the checksum on all of the reliability page code is good. If it isn't, trigger a reload. If okay, see whether debugging and this processor is in the simulated halt state; if it is, hang in this Stage.

Control Structure

Examines COMREL to determine whether reliability page has a good checksum. DEBUGM and BLTMYC (simulated processor control register) control the halted debugging state.

Entry Points

Coroutine, called by Stage dispatcher via WSLA7.

External Calls

WSLEEP to dismiss the coroutine.
SBAD to hang the Stage pending a reload.
STEST to conditionally pass this Stage.
RELTRY to initiate a Block Transfer for a reload.

Initialization

WSLA7 is initialized to SRC00.

Cleanup

None.

Data Structures

Local Data

None.

Shared Data

PROCBT: my assigned proessor bit.
SNAP: snapshot save area.
COMREL: pointer from communication page to reliability code page.
TLIMIT: limit of checksummed area on each code page.
CKSUM: checksum on a code page.
RCKCON: consensus array for this Stage.

I/O Performed

None.

5.1.7 Stage LC - Local Page Checksum

Function

Verifies the checksum on all the local memory code for this processor. If checksum fails in every processor (i.e., consensus is reached), trigger an external reload.

Contorl Structure

Computes additive checksum on all constant words of local memory, which should be zero. External reload is initiated by setting appropriate Block Transfer parameters.

Entry Points

Coroutine, called from Stage dispatcher via WSLA7.

External Calls

SCLROK to pass this Stage and remove fix-it bit for this processor.
WSLEEP to dismiss coroutine.
CKSUBI, CKSUBR to compute local code checksum.
SFXBAD to hang this Stage and reach consensus to reload.
RELTRY to initiate an external reload of local memory.

Initialization

WSLA7 is initialized to dispatch to SLC00.

Cleanup

None.

Data Structures

Local Data

None.

Shared Data

BLTST: Block Transfer state.
LOCALC, HOTLIM: local code checksum and limit.
LOCCON: consensus for this Stage.

I/O Performed

None.

5.1.8 Stage MC - Common Memory Checksums

Function

For each common page in the system except the reliability page, compute its checksum. Force page with invalid checksum to be empty (signalled by an improper page type) but with a good checksum.

Control Checksum

Computes additive checksum for each page, using a limit provided on each page. A special checksum value (password) signals that the checksum should be recomputed, for system initialization after paper tape loading.

Entry Points

Coroutine, called from Stage dispatcher via WSLA7.

External Calls

MEMTS2 to see whether a page exists in MEMSEG.
CKSUBI, CKSUBR to compute code checksum.
WSLEEP to dismiss subroutine.
SFXBAD to hang this Stage and reach consensus to reset a page with bad checksum.
SCLEAR to clear this processor's bit from a fix-it array.
SOKAY to signal successful completion of verifying all checksums.
WSTMEM to remove a failing memory page from usage.

Initialization

WSLA7 is initialized to dispatch to SMC00.

Cleanup

Pages with bad checksums are set to be unused (invalid page type). Pages with parity failures that can be fixed are marked to be reinitialized. Pages with solid memory parity failures are removed from usage.

Data Structures

Local Data

SMCNEX: next page to checksum.
SMCLAS: page we checksummed last.
CKSMEM: flag to signal a page whose checksum may have failed.

Shared Data

MAPREL: map setting for reliability page.
CKSUM: page checksum word.
BLTST: Block Transfer state.
INTIME: page initialization flag.

I/O Performed

None.

5.1.9 Stage MM - Common Memory Management

Function

This stage consists of four subparts, each to deal with one class of page type:

1) Find and maintain the active code pages. If can't find one with a given type, and haven't found the minimum number of code pages needed to run the system, hang in this stage, awaiting a reload. If a spare of the given type exists, copy a fresh page from it. The code pages are allocated starting on the lowest-numbered end of memory. All other page types will be allocated from the high end downwards.
2) Find and maintain the required variables pages. If can't find one with a given type, create one and zero it. If not enough memory for all these pages, hang in this stage.
3) Find or construct spares of all the code pages.
4) Find or construct optional variables pages, up to the maximum ths system can use.

Pass this stage if we get at least to part 3.

Control Structure

1) Starting at the bottom of common memory, each map is checked to see if it is a) in the MEMSEG table and b) its CMAP and LMAP entries agree with its type. This is repeated for all the required code pages until either the necessary code pages have been checked or until memory runs out. If memory runs out, STAGE will hang here.
2) Starting at the top of common, the same checks are repeated for the required variables pages. If a page is missing, another is usually found or made into a vars page.
3) The spare code pages are next found and entered into LMAP and CMAP tables. If no spare page exists, the original, if it exists, may be copied and saved.
4) The optional variables are now checked and set up, if there is room. If there is enough room to allocate all of the previous pages, then any excess pages are marked free.

Entry Points

Coroutine, called from the Stage dispatcher via WSLA7.

External Calls

SMMHTS, SMMLTS to check maps and supply free pages or spares if map wrong.
MEMTST to check if a page is in the MEMSEG table.
SCLROK to say this stage passed and OK.
SMMCOP to copy a page of memory to a new page.
SMMCP2 to clear and check a variables page.
SFXBAD to set and check the consensus.
FIXJIF to cleanly reenable jiffy interrupts.
WSLEEP to dismiss the coroutine.
WST to restart Stage if too little common memory exists for the system to run.

Initialization

WSLA7 is initialized to dispatch to SMM00.

Cleanup

Pages that the system doesn't need are marked with an invalid page type to indicate that they are unused. If memory runs out first, the remaining entries in CMAP and LMAP are marked empty.

Data Structures

Local Data

SMMHI: the high limit of unchecked pages.
SMMLOW: the low limit of unchecked pages.
SMMSPA: a word used to hold the location of the spare page of desired type.
SMMFRE: remembers any free page that has been found.
SMMTOT: total pages allocated so far, biased by count of desired variables.

Shared Data

LMAP: local copy of CMAP.
CMAP: system map table by page type.
CKSUM: common page checksum.
TLIMIT: common page checksum limit.
MAPCOM: map setting for communication page.
MEMTOT: total number of memory pages, from Stage MD.
INTIME: page initialization flag.
BLTST: Block Transfer state.
COMREL: communication page pointer to reliability page.
TYPE4K: common page type word.

I/O Performed

None.

5.1.10 Stage ID - I/O Interfaces Discovery

Function

Search through I/O address space for common I/O interfaces. Results are built as a bit-coded table.

Control Structure

Devices are checked for whether their device status register exists. Existing interfaces are tabled by bit in USEIO. Majority agreement among processors is required to decide that an interface exists.

Entry Point

Coroutine, called from Stage dispatcher via WSLA7.

External Calls

WSLEEP to dismiss the coroutine.
SCLEAR to remove this processor's fix-it vote.
STEST to conditionally pass this Stage.
SBAD to hang this Stage.

Initialization

WSLA7 is initialized to dispatch to SID00.

Cleanup

None.

Data Structures

Local Data

IOFIX: fix-it array for voting to change the USEIO table.

Shared Data

PIDGET: table of PIDs from Stage BD.
USEIO: bit-coded table of existing devices.
PROCBT: this processor's bit.
IOCON: this Stage's consensus.
CONFLG: flag to tell Configuration that USEIO changed.
CONPNT: pointer to current Configuration page.
CONA7: current Configuration dispatch.

I/O Performed

None.

5.1.11 Stage AR - Application-dependent Checks

Function

1) Check for any processors that have failed local memory checksums, and initiate Block Transfer to reload them.
2) Check for any processors that should be running and aren't, and initiate Block Transfer to reload and restart them.
3) Poll initialization checking routines on each code page, if any.
4) Check for occurences of successful QUIT retries or unsuccessful real time clock readings, and report them (via traps). Report if jiffy interrupts have stopped occuring.
5) If any traps have been saved in the local buffer, tally them in the common memory trap tables.

Control Structure

Functions are performed in order given. If an initialization check routine indicates that initialization needs to be performed, reach consensus and call the specified initialization routine.

Entry Points

Coroutine, called from Stage dispatcher via WSLA7.

External Calls

1-2) RELTO to initiate Block Transfers.
3) FIXJIF to cleanly reenable jiffy interrupts following an initialization.
WSLEEP to dismiss coroutine.
4) SCLROK to pass this stage and remove this processor's fix-it bit.
RCLOCK to read the Stage real time clock.
5) ILLCNT to tally a trap in common memory tables.

Initialization

WSLA7 is initialized to dispatch to SAR00.

Cleanup

4) Counters for QUIT retries and unsuccessful real time clock readings are cleared.
5) Local trap table is cleared.

Data Structures

Local Data

2) TEMP1: bit-coded set of processors to start.

3) TEMP2: index of page being called for initialization check.

Shared Data

1) BLTST: Block Transfer state.
   LOCFIX: Stage LC fix-it array.
   LOCALC: local memory code checksum and limit.
   CKSCON: Stage MC consensus.
2) PRTIME: time next to check for processors to start up.
   STIME: local copy of Stage system time.
   PROIOR: processors that shouldn't be started.
   SEGCON: Stage MD consensus.
   PROCEX: bit-coded array of processors that exist.
3) LMAP: local table of page maps by type.
   PGINIT: pointer to initialization check for common code page.
   MAPREL: map setting for reliability code page.
4) QUITRT: count of successful quit retries.
   CLOKRT: count of unsuccessful real time clock reads.
   PROCNO: index of this processor name in COUTAB.
   JTIME: real time clock reading at last jiffy interrupt.
5) MAPVAR: variables page map.
   LOCIPT: pointer into local trap table.
   LOCILL: local trap table.

I/O Performed

Jiffy interrupts are disabled while initialization is performed.

5.1.12 Block Transfer

Function

Performs a variety of functions associated with moving blocks of memory around in the machine. In particular, this process performs reloads from external sources, as in packet core, reloads of individual processors from other processors, dumps of portions of memory for diagnosis or verification, and examine and deposit operations for DDT.

Control Structure

Block Transfer is a large subroutine. It checks the state of the Block Transfer parameters and performs any indicated actions.

Entry Points

BLT, or BLTLKD if BLTLOK is already locked. Called from the Stage dispatcher, and from the operational system if there is a transfer in progress.

External Calls

BLTPRM, BLTPRS to do some checking for transfers to or from individual processors. BLTRLD is the entry to the external reload (packet core) process.

Initialization

None.

Cleanup

A variety of error conditions will terminate the transfer in progress early.

Data Structures

Local Data

TEMP1: saved subroutine return.
TEMP2: which processors were done this call (bit-coded).
TEMP3: index of bus being used for backwards bus coupling.
TEMP4: correct bus coupler password to use.
BBCRST: processor last started.
BBCBAD: processor to which backwards bus coupling last failed.

Shared Data

BLTST: Block Transfer state plus error bits.

BLTLOK: Block transfer lock.
BLTTO: timeout on Block Transfer actvity.
BLTADD: current Block Transfer address.
BLTSIZ: how many bytes to transfer still.
BLTDON: how much was transferred in this piece.
BLTBFM: map setting for buffer for Block Transfer.
BLTBFA: address for buffer.
BLTPRO: number of processor that did the Block Transfer.
BLTBMK: bit table of processors that had trouble in Block Transfer.
BLTDID: number of processor whose address space last transfer was to or from.
BLTPOK: PID level for Block Transfer to poke when it's done.
BLTSTY: source core type for transfer.
BLTSPM: source bit table of processors.
BLTDTY: destination core type for transfer.
BLTDPI: destination bit table for processors.
BLTDPM: destination bit table for this piece of transfer.
BLTBUF: 36 word buffer for doing Block Transfers.
RLDDEV: current device to reload from.
RLDINI: flag to signal initialization of reload parameters.
MYPROC: my processor's coupler number.
STIME: local copy of Stage system time.
LMAP: local table of map settings by type.
PRTIME: time when next to try a processor restart.
BLTMYM: simulated maps for Block Transfer.
COUTAB: table of coupler addresses for processors.
PROCEX: bit table of processors in system.
PROCBT: my processor bit.
PROCNO: my processor number (index to COUTAB).
AMPROC: table of coupler passwords by processor.
USEBUS: bit table of busses in system.
BBCLOK: lock on backwards bus coupling privelege.

I/O Performed

None.

5.1.13 Quit Handler

Function

Services Non-existent Memory interrupts, also know as QUITs. Each QUIT is retried once in case it was intermittent. If it is solid, check for a "password", which is a special form of the NOP instruction, at the location following the instruction that got the QUIT, and transfer control to its target branch address if it is there. Otherwise, enter the unexpected QUIT handler, which will restart the system in Stage.

Control Structure

Obtains control via the processor QUIT vector in low address of local memory. Returns to main program or enters unexpected QUIT handler if the QUIT is solid and there is no password. Instruction fetch QUITs and QUITs within the Quit Handler are always treated as unexpected.

Entry Points

Q50 if even processor (key 0); Q70 if odd (key 1).

External Calls

QSUBR, QSUB0 to set up parameters during unexpected QUIT, QUIT on instruction fetch, or QUIT in Quit Handler.

Initialization

Local memory interrupt vectors are initialized to dispatch to Q50 and Q70.
UQUIT is initialized to dispatch to SYSUQ, the system unexpected Quit Handler.

Cleanup

Unexpected QUITs, QUITs on instruction fetches, and QUITS in the Quit Handler all cause system restarts via the unexpected QUIT routine.

Data Structures

Local Data

QX: register save area.
QUITFL: flag that is normally set only while inside Quit Handler.
QUITAD: address of most recent QUIT.
QUITPC: instruction address at most recent QUIT.
QUITST: program status at most recent QUIT.

QUITTM: real time clock reading at most recent QUIT.

Shared Data

QUITV: address of QUIT vector used (to distinguish main, alternate processors on a processor bus).
LCLOCK: pointer to Stage real time clock to use.
QUITRT: count of QUIT retries; counts up each retry and down each solid QUIT.
UQUIT: pointer to unexpected QUIT routine for system.

I/O Performed

None.

5.1.14 Illegal Instruction Interrupt Handler

Function

Processes interrupt caused by executing illegal instruction codes. Certain illegal instructions are used by the system to signal particular error conditions; these are called traps. Some traps merely save their number; others cause production of a snapshot of a portion of local memory with important parameters of the machine state at the time of the trap. The special traps 0FFFF! and 0FADE! have special effects when in debugging mode; the former causes the processor to enter the simulated halt state and the latter does this plus setting the common password to cause all processors to enter this state.

Control Structure

Decide whether this is a trap, snapshot, or true illegal instruction. For traps and snapshots, save the required data. For true illegal instructions, restart the Stage system.

Entry Points

ILLP0 for even (key 0), ILLP1 for odd (key 1) processor.

External Calls

SAVMAP to save maps in the snapshot area.
ILLCNT to save trap number in common memory table.
WST to restart Stage system.

Initialization

The local memory interrupt vectors for illegal instructions are initialized to dispatch to ILLP0 and ILLP1.

Data Structures

Local Data

ULIIOP: last illegal instruction, ignoring traps.

Shared Data

LX: register save area.
IRET: place to set up return vector from interrupt.
MAPVAR: variables page map.
DHALT: password for halting all processors (if in debugging mode).
BLTMYC: simulated control register for debugging.

SNAPBG: beginning of area to copy into snapshot.
SNAP: snapshot buffer in local memory.
TYPE4K: common memory type word.
LOCIPT: current pointer into local trap table.
LOCILL: local trap table.

I/O Performed

None.

5.1.15 Level 1 Interrupt Handler

Function

Process level 1 external interrupts. If a remote power failure, stop buddy processor, wait a while for power to go away, then restart this processor and its buddy in Stage. If an attention interrupt following a reset, set the simulated halt state for debugging and restart Stage. Otherwise trap and restart Stage.

Control Structure

Started via local interrupt vector for level 1 interrupt. Cause of the interrupt is distinguished by a device code stored in the interrupt vector.

Entry Points

Entered at LEVEL1. Entered from Level 4 Handler at HALTUS to perform restart function for this processor and its buddy.

External Calls

WST to restart in Stage system.
DSTAND to restart with simulated halt condition for debugging.

Data Structures

Local Data

None.

Shared Data

IX: register save area.

I/O Performed

None.

5.1.16 Level 4 Interrupt Handler

Function

1) Process Level 4 interrupts.
2) For local power fail interrupts, call HALTUS (see Level 1 Handler).
3) For local power restore interrupts, restart the system in Stage.
4) For sixty-cycle interrupts ("jiffies"), check if the system real time clock is operating properly; if not, find a new one and restart Stage. Else, check if the program is hung up by examining local time; if not, return to main program. If it is hung on a locking instruction, unlock the indicated lock. Otherwise the program is in a loop, and the system is restarted in Stage.
5) Merely trap on other level 4 interrupts and return to program.

Entry Points

Entered at SJIF.

External Calls

2) HALTUS to stop this processor and its buddy.
3,4) WST to restart Stage
4) FNDCLK to start using a new real time clock.
RCLOCK to read the system real time clock.

Initialization

Level 4 interrupt vector is initialized to dispatch to SJIF.

Cleanup

Hung locks are unlocked. Abnormal conditions cause a restart in Stage.

Data Structures

Local Data

4) JTIME: reading of real time clock at previous jiffy interrupt.
UJIFFY: program counter at latest program in a loop condition.

Shared Data

1) IX: register save area.
4) LTIME: time to run Stage next.
IRET: place to build return vector.

I/O Performed

None.

5.2 IMP System Central Dispatch

Function

Dispatch to routines needing service according to their PID levels.

Control Structure

At entry check maps for their default values. If it is time to run Stage, transfer control to the Stage dispatcher. If running in a processor with an operator console, maintain the light words. If in debugging mode, check for system halts and return to Stage if halted. Otherwise, poll the system PIDs in order, dispatching through the BASE dispatch table by PID level.

Entry Points

LOOP is the nominal entry point (return from most strips).
LOOPM to reset map 0 before checking.
LOOPMV to set all maps properly and not check.
BAD is dispatch from BASE for an unexpected PID.
NOPIDS is dispatch from simulated last PID (to end the polling of real system PIDs).

External Calls

SJ6 is the return point to the Stage dispatcher.
BASE contains dispatches for each PID level in use in the system.

Initialization

None.

Cleanup

None.

Data Structures

Local Data

OLDP: saved copy of last value read from PID.

Shared Data

BASE: the dispatch table.
MAPCOD: proper map setting for map 0.
SLFPTR, FMAP2, SLFLK page self-pointers for map-checking.
MAPVAR, MAPV2: proper settings for maps 1-3.
LCLOCK: pointer to clock to time Stage.

LTIME: next time Stage should run.
CONSOL: pointer to processor console (operator panel), if any.
WATM1, WATCH1, WATM2, WATCH2: map settings and light pointers for console lights.
DEBUGM: debugging mode flag.
PROCBT: my processor bit.
SNAP: snapshot save area.
DHALT: location to check for halt password to halt all processors.
PIDGET: table of PIDs to poll.
WATCHS: light word; each processor complements its bit as it enters Stage, nominally displayed in ADDRESS lights in console.
IDLEC: count of number of times all PIDs were empty.

I/O Performed

Each time a processor references a PID, the highest level in that PID which is set is returned, and that level is cleared. PID levels may correspond either to hardware devices needing service or software processes that should be run. For processors having an attached operator console, the ADDRESS and DATA lights are made to display the values requested via the light pointer words.

5.3 Modem to IMP

Function

1) Process input PIDs and initiate new modem inputs.
2) Verify packet checksums and lengths.
3) Place routing packets on routing input queue and poke routing.
4) Pass data packets to Task and poke it.
5) Free acknowledged packets and poke modem output to send acks.
6) Place packet core buffers on the packet core queue and poke the packet core fake host JAM side.

Control Structure

All modem inputs are handled by reentrant code. Variables for each modem are obtained from the proper parameter block. The hardware is first checked and, if idle, a new buffer is put up for input. Otherwise, a queue of input buffers previously completed is polled and processed in order.

Entry Points

Each input PID dispatches to M2I.

External Calls

1) FREGET to get a new buffer for input after next (double-buffered).
   FLUSHB to free buffers with hardware input errors.
2) CKQPUT to queue a buffer to send to the NCC diagnostic terminal.
   WHEORB to modify buffer use bits.
   DEQUE to get packets off the modem input queue.
   SUBCHN the shared software checksumming subtract chain.
4) TSKPUT to enqueue buffers for Task, and poke it.
5) UNPACK to access buffer (if tracing).
   TRCDUN to copy trace data to trace buffer.

Initialization

The first input after a reset and initialization is discarded by reading into the shared bit sink JUNK.

Cleanup

When a line is declared down by Line State Timeout, input is stopped via a reset and input buffers freed, if any.

Data Structures

Local Data

4) TEMP3: input subchannel octet and bit for Task.
5) TEMP2: bit table of ACKs to do.

Shared Data

1) MBLKS: the table of parameter blocks by PID.
   IOBLOC: the hardware interface address for this modem.
   M2ILOC: lock on modem input hardware.
   NXTBF: pre-allocated buffer to read into.
   POINT: high-order buffer address bits.
   FILLING: input buffer currently in use.
   LMIQ: lock on queue of completed input buffers.
   SMIQ, EMIQ: queue of completed input buffers.
   JUNK: address of junk buffer.
   CHAN: buffer word used for input endpointer.
2) LOCKM: lock on modem software variables.
   BUFE: end of data in buffer.
   INCH: which modem this buffer is from.
   CKERRS: count of hardware checksum errors.
   MLOOP: modem loop state.
   MNOBUF: count of missed inputs.
   CLOCK: pointer to current system real-time clock.
   IT: input time this packet (for trace).
3) MINE: my IMP number.
   LENDR, LENDT: end bit values, for detecting looped lines.
   LSTATE: lint up/down state.
   MODEM: index to neighbor table.
   M2NGHB: modem neighbor table.
   NEIGH: neighbor this modem.
   SIHY: flag to send I-heard-you.
   LOCKR: lock on routing parameters.
   ERUTQ: routing input queue endpointer.
   THD: pointer to line to leader for time synchronization.
   SYNC: local copy of network time for statistics.
   LAC: line alive count, drives LSTATE in DEDL.
4) INFREE: input channels for which we have no input (bit table).
   RSEX: state of input subchannels (bit table).
5) MAXCHN: number of logical subchannels this line.
   TSEX: state of output subchannels (bit table).
   SSENTQ, ESENTQ: queue of packets awaiting acknowledgements.
   CHNBSY: output channels that are assigned (bit table).
   SLOTS: count of busy output channels.

THRUPT: modem output throughput.
ITRACE: trace on/off flag.
SNDING: current output packet.
LATER: flag to flush current output packet when done sending.

6) CORIOB, LOCKF: lock on packet core fake host.
EFHCQ: packet core input queue endpointer.
BHPID: PID for packet core fake host process.

I/O Performed

Input of data packet into a packet buffer; junk buffer used if first input after a reset or no free buffer available. Input buffers are preallocated to achieve double-buffering.

5.4 IMP to Modem

Function

1) Process output modem PIDs.
2) Free the packet last sent if acknowledged while being sent.
3) Send packet core messages, reload demands.
4) Send routing messages and nulls after routing.
5) Retransmit packets unacknowledged for MRTIME (* 100 microseconds). Verify their checksums.
6) Send new priority packet if channel available.
7) Send new regular packet if channel available.
8) Send null packet of acknowledgements.

Control Structure

The routine checks the modem output state. If it is still busy, it ignores the PID and dismisses. If not, it performs functions 1 and 2. It then attempts to perform functions 3 through 8 as required, in that order. If none are required, the output is marked inactive by clearing SNDING.

Entry Points

The IMP to modem PIDs dispatch to I2M, which loads the appropriate parameter block address from MBLKS. All modems share reentrant code. I2M is poked by fast timeout every 25.6 milliseconds.

External Calls

2) FLUSH to free the buffer.
3) WHEORM to adjust use bits for core packet.
4) UNPACK to access routing message packet.
5) DEQUE to get next packet awaiting retransmission.
   SUBCHN to checksum retransmitted packet.
   WHEORB to adjust buffer use bits if bad checksums.
   CKQPUT to send retransmissions with bad checksums to NCC diagnostic terminal.

6-7) DEQUER to get next priority or regular packet.

Initialization

None.

Cleanup

When a line enters its "hold-down" line state, no outputs are initiated for a specified time, while queues are cleared and pending packets rerouted onto other lines.

Data Structures

Local Data

None.

Shared Data

1) MBLKS to access proper parameter block.
   IOBLOC: address of hardware status registers.
   I2MLOC: modem output hardware lock.
   LOCKM: modem software lock.
   RUTTIM, RUTRAT: line speed timing variables.
   CLOCK: pointer to active system real-time clock.
2) SNDING: packet last sent, if any.
   LATER: flag to flush previously sent packet.
   ESENTQ: end pointer of queue of packets awaiting acknowledgement or retransmission.
   LOCKRO: lock on routing output buffer queue.
   CHAN: buffer word used as routing output use count.
   MAXCHN: number of logical subchannels on this line.
   SNULL: flag to send nulls by octet (16 octets for 128 channels).
3) SROUTE: flag to send routing or reload packet.
   SBLK: pointer to reload packet to send.
4) LSTATE: line state for this line.
   RUTOBF: present routing output buffer to use.
5) SSENTQ: head of queue of packets awaiting acknowledgement or retransimssion.
   ST: time this packet was sent.
   BUFE: length of packet.
   STIMER: retransmission count for this packet.
6) SPRIQ: head of priority queue.
7) SREGQ: head of regular queue.

6-7) CHNBSY: bit table of busy output channels.
   TSEX: output odd/even sex by channel (bit table).

3-7) POINT: high-order memory address of packet.

8) SIHY: flag to send I-heard-you with null.
   SYNC: network time locally.
   MINE: my IMP number.
   NULLHD: buffer for null, one dedicated to each modem.

4-8) RSEX: receive odd/even bits, sent as scknowledgements to other end of line.
   LENDT: proper end bit ("I am high IMP").

I/O Performed

3) Output of packet core message from specified buffer.
4) Output of routing from buffer from RUTOBF.

5-7) Output from specified data buffer.

8) Output of null from fixed buffer for this modem.

5.5 Host to IMP

Function

1) Process Host input PID
2) For control message, take appropriate action and initiate input.
3) For regular message leader, begin processing of message and initiate first packet input.
4) For input of first packet, if destination has not allocated space, initiate request and input of second packet (if any).
5) For first packet, if destination has allocated space, process packet and initiate input of second packet.
6) For second through final packet, process the packet and initiate input of subsequent packet.
7) For all packets, check length, generate software checksum, and pass them to Task.

Control Structure

Function 1 is performed, then one of the remaining functions is resumed from the last coroutine exit.

Entry Points

All Host input PIDs come to HI, which picks up the appropriate parameter block address from MBLKS. Coroutine control returns via the parameter block entry HILO.

External Calls

1) HILEDI, HILEDM to read the next leader.
   HINOPT to check if a NOP.
   HIERC to set up IMP to Host error message.
   LEDPO to queue error message.
   HINBWT to await I/O completion.
   FLUSH to free input buffers.
   HIWM, HIWFE to exit coroutine.
   HITTGO to initialize wait timer for hardware.
   TRNPUT to allocate a transaction block for next leader input.
3) HINBUF to initiate next packet input.
4) FNDHAC to get Host access words for raw packets.
   MESGET to get a message number on an open end-to-end connection.
   TALLYG to get an 8-packet allocate.

4-7) HIPKTR, HIPKT, HISET, HIPKTE to set up packet header and checksum it.
   HI2TSV, HI2TSK to give packet to Task.

HOTHRU to count Host input throughput.

Initialization

HILO is initialized to HIGO. The first input from real Hosts is always discarded.

Cleanup

The above initial state is entered and all resources freed whenever a hardware error is detected, when the Host ready line first comes up, or an input (subsequent to a message leader) takes more than 15 seconds.

Data Structures

Local Data

TEMP1, TEMP2: used as arguments and temporary returns for various subroutines.
HISAV7, HTEMP7: save returns for routines that can sleep.
HILO: main coroutine sleep location.
HISP: current buffer in progress.
HIBF: next buffer to process (double-buffered).
HIPKTH: builds PKTH word for each packet.
HIHAND: remembers handling type for packet.
HIENDI: remembers hardware endpointer for packet in HISP.
HTEMP: saves certain subroutine parameters.
HIOLDB: points to current transmit message (TM) block.

Shared Data

DEADSC: Host dead subcodes (from Host going down message).
TRMIDL: message-id field in leader in transaction block.
IOBLOC: pointer to hardware for this Host.
JUNK: shared bit-sink to throw away input.
FAKE: flag for Fake, Back, or VDH Host.
FAKESI: input pointer for Fake "hardware".
BHPID: Fake Host JAM PID.
HIHD: Host Status.
OPHGO: flag for Host state change.
MYIMP: local IMP coming up counter.
HIBITS: Host status bits; communicated to Task.
HITRAN: pointer to current transaction block.
HOMODE: Host new/old leader format flag.
TRNETL: beginning of leader area in transaction block.
TRTYPL: leader type word in transaction block.

TRHSTL: handling type and destination Host in leader, replaced after MESGET by message block (TM) number and message number.
TRDSTL: destination IMP in leader.
TRNTIM: transaction block timeout byte.
TRSTAT: transaction block status.
HIPAD: shared Host padding bit sink.
RUT: table of best delay routes by IMP.
ISTATE: IMP state from routing.
HACCOM: table of Host access - communicate words.
TRLUSE: transaction block local use number for TM block.
TRPACK: transaction block packet pointer.
HTPMTN: Host input throughput counters.
TRDEDS: reason for destination dead in transaction block.

I/O Performed

Initiates all inputs from Hosts. Leaders are read into transaction blocks and data into packet buffers.

5.6 IMP to Host

Function

1) Process Host output PIDs.
2) Reset Host output hardware if error or timed out.
3) If just completed sending a message, mark RM block to send rfnm or rfnm with allocate.
4) For each packet of a message, trace it if necessary and flush it.
5) Send next control message if any.
6) Send next priority message unless next regular message is too old.
7) Send next regular message if any.

Control Structure

Functions 1-4 are performed each time the output PID occurs if neccssary. Then functions 5-7 are performed in order as needed.

Entry Points

All IMP to Host PIDs dispatch to IH, which sets up the appropriate parameter block address and resumes the coroutine through IHLO. All IMP to Host routines are performed by shared, reentrant code. In addition to the hardware (simulated for Fake Hosts), IMP to Host is poked by Host to IMP, Task, Host timeout, and itself.

External Calls

1) IHDB, IHDBA are the Imp to Host sleep routines, called from various points (IMP to Host runs as a coroutine).
2) LEDGET to get the next control message.
   IHDUMP to place unprocessed messages on the discard Host output queue when resetting.
3) HOTHRU to compute IMP to Host throughput.
4) DEQUE to take packets off output queues.
   TRCDUN to trace each packet.
   FLUSHB to free each buffer.
5) IHLS to send a control message.

6-7) IHLSN to send message leader.
   UNPACK, UNPCKC to access buffers.

Initialization

IHLO is initialized to dispatch to IHIDLE. When a Host comes up, IMP to Host first sends 4 NOPs and a "Interface was reset" control message.

Cleanup

If any output takes more than 30 seconds to complete, all pending control messages are freed, all messages on the Host's regular and priority output queues are discarded and the corresponding RM states marked "dead", the Host's ready line is flapped, and the initialization state (above) is entered. The Host is declared tardy or down, depending on whether its ready line is asserted or not.

Data Structures

Local Data

3) TEMP1: local argument to HOTHRU.

Shared Data

1) MBLKS: table of parameter blocks by PID.
IHLOC: Host output hardware lock.
IOBLOC: Host interface address.
IHGOIN: Host timeout/reset flag.
LOCKIH: Host output software lock.
IHLO: dispatch address.
IHWQ: pointer to current queue to send message from (regular or priority).

2) HIHD: Host state.
OPHGO: flag for Host state change.
FAKE: flag for Fake, VDH Hosts.
SPECAL: flag for initialization control messages.

3) HTPMFN: Host throughput counters.
IHLEDR: IMP to Host leader buffer.
BUFB: pointer to RM block.
RMLOCK: lock on RM block.
RMMESS: RM message number.
RSTATE, RMTYPE: RM message state bits.
HOTPID: IMP to Host PID.

4) ITRACE: global trace flag.
CLOCK: system real-time clock.
IHLSTP: last-packet flag.

5) IHTT: Host output timeout counter.
HOMODE: Host new/old leader mode flag.
HIHOST: Host number.
HIMINE: Host's IMP number.

6-7) CHAN: buffer age word.
SHQ, SHPQ: Host regular, priority queues.
TIME: global time in 25.6 millisecond ticks.
RMCTL: RM block handling type.
RMHOST: RM block remote Host.
POINT: buffer high-order address.

BUFE: end of buffer.
FAKESO: Fake Host output pointer.
HBPID: Fake IMP to Host PID.

I/O Performed

2) Host interface output side is reset.

5-8) Initiate output of control and data messages. Control messages are sent from transaction blocks, message leaders from a dedicated buffer in the Host parameter block, and data from packet buffers.

5.7 Task

Function

1) Process Task PIDs and service the Task queue.
2) For input for Host on this IMP, pass packet to Task For Us, and ACK its source.
3) For store-and-foreward packets, select output line.
4) If space is available and an output modem subchannel is free, queue the packet on modem priority or regular queue, poke modem output, and ACK source of packet.
5) If the packet is a packet from store/forward, rerouting, or a reply, and space is available but the output modem subchannels are full, queue the packet on the auxilliary Task queue to retry later.
6) If space is not available, NACK its source, and poke it.

Control Structure

Task is entered once for each packet on its input queue. If its queue is still empty after removing the first packet, it pokes itself immediately, permitting multiple processors to work concurrently on separate packets.

Entry Points

Entered at TSK. Poked by Host to IMP, Modem to IMP, Back Hosts, and Fast and Slow Timeout, in addition to itself.

External Calls

1) DEQUE to get packet from Task queue.
   FLUSHB to discard packets with discard bit (to clear input subchannel).
2) FORUS is entered directly (next section).
4) CMOVE to allocate buffer space.
   TRYM1 to poke output and input modem routines.
   WHEORB to adjust use bits for packet.

Initialization

None.

Cleanup

Flush packets bound for dead IMPs.

Data Structures

Local Data

1) TSKCHN: copy of CHAN word for buffer (modem: input subchannel octet and bit; Host: 0).

Shared Data

1) LTQ: lock on Task queue.
   STQ, ETQ: Task queue.
   INCH: address of parameter block for packet source.
2) ISTATE: table of IMP states from routing.
   RUT: table of best-delay paths to each IMP.
   DELTIM: delay hold-down routing table.
4) LSTATE: output line state.
   SLOTS: count of available logical channels on output line.
   LOCKM: lock on output modem software variables.
   SREGQ, SPRIQ, EREGQ, EPRIQ: modem regular and priority output queues.
   CLOCK: system real-time clock.
   QT: time this buffer was queued for output.
5) ERQ: auxilliary Task queue endpointer.
3-6) INFREE: bit table of free Modem to IMP subchannels.
   RSEX: bit table of odd/even state by subchannel.
   MITHRU: Modem to IMP throughput (packets).
   SNULL: bit-coded (by octet) flag to send null for acknowledgements.
   HINPID: Host to IMP PID to poke.
   LOCKHI: lock on Host to IMP parameters.
   HIBITS: Host to IMP state bits for communicating with Task.
   HITRAN: pointer to current Host to IMP transaction block.
   TRSTAT: state bits for transaction block.

I/O Performed

None.

5.7.1 Task For Us

Function

1) Set up local variables.
2) Check for raw packets and process them.
3) Check message block. If it mismatches, check for and process get-a-block, incomplete query, reset and reset reply messages. Ignore other block error packets.
4) Check status of local Host, and set flags if down or access failure.
5) Check message number and set flags for out of range, next to go, in next 8, and in previous 8.
6) For eight-packet message packets, find the reassembly block for the message and add this packet if not a duplicate. Ignore if reply message state indicates message completed already.
7) For single-packet messages, find the reassembly block for this message and mark it complete.
8) For eight-packet requests, mark the reply state table to show the request received.
9) For single-packet requests that are the next to go, mark the reply state and queue them directly on the output Host queue.
10) For single-packet request that are not the next to go, find a free reassembly block and mark it complete (like 7).
11) In either 9 or 10, if there is insufficient reassembly space, mark the reply state table to show a single-packet request received.
12) For incomplete message packets, free any associated reassembly resources and mark reply table.
13) For givebacks, find and free a reassembly block and mark the reply table.
14) For incomplete queries, if message number out of range send an out-of-range reply; else if in last 8 messages and reply state is idle, send correct duplicate reply; else if in last 8 and not idle state, send out-of-range; else if in current message window perform cleanup of reassembly resources and mark reply state for incomplete reply.
15) In each of functions 6-14, if marking reply state bits or completing a message, search for completed messages to queue for the IMP to Host routine, and poke it if any.
16) For rfnm messages, mark the transmit message number complete.

17) For rfnm with allocate messages, if in reply to a single-packet request, pick up the saved copy of the message and send it, else (was 8-packet request) increment the allocate count in the TM block.
18) For dead replies, set the stop bit.
19) For incomplete replies, adjust the condition codes in the transaction block.
20) For each of 16-19, if the reply was for a regular message, queue the transaction block to send a reply to the originating Host and poke its IMP to Host routine.
21) For out-of-range replies, send a reset message right away.
22) For got-a-block messages, modify TM block state appropriately to open conversation or send destination dead to Host.
23) For reset request message, set age of TM block to maximum to signal the request.
24) For reset message, mark RM block idle and send reset reply right away.
25) For reset reply message, mark the TM block idle.

Control Structure

Task For Us is structured as some common setups (functions 1-5), followed by a dispatch based on the message-type for functions 6-25. There are many subroutines for shared functions after the dispatch.

Entry Points

Entered directly from Task at FORUS.

External Calls

HOSTNM, HOSTNO to set up local Host status and access control.
RALLYP to modify reply state bits.
REGCHK, REGCH2 to check state of reply bits for transmissions (5-15).
FIXNRE, FIXNAE to adjust reassembly and allocate counts.
REPCHK to check transmit message state for replies (16-21).
RFLEDP to mark a transaction block for a reply to the Host.
LEDPC to either free or queue a transaction block for a reply to a Host, depending on whether the Host is up.
REPFIX to mark a transmit message complete in the TM block.

REASF to free a reassembly block and any buffers it contains.
REASGT to find a specific reassembly block (by message number).
REASF1, REASF8 to find allocated reassembly blocks for 1, 8 packet messages respectively.
FTRNGT to find a transaction block to go with a reply.
CMOVE to allocate buffer space for immediate replies, raw packets, or one-packet requests.
TSKPUT to requeue immediate replies for Task.
WHEORB to modify use bits for buffers for Hosts.
REASAL to allocate a free reassembly block for out of order one-packet requests.
UNPACK, UNPCKC to access packets of a message to give to Host.
FLUSH to free packets in a reassembly block.

Initialization

None.

Cleanup

Reset TM and RM blocks are marked free. Transaction blocks for outstanding messages from Hosts that have gone down are freed. Reassembly blocks for incomplete messages are freed, together with any packets they contain.

Data Structures

Local Data

TSKBUF: address of buffer in process.
TSKBTS: state bits for Host state, message range, etc.
TSKHST: pointer to local Host parameters for this message.
RALSHF: amount for RALLYP to shift bits for RSTATE, RMTYPE.
REPBIT: bit for TM message for replies.
TEMP1, TEMP2: local returns used in various subroutines and main line.

Shared Data

RMBLKS: reply message (RM) blocks, including:
- RMLOCK: lock on the block.
- RMIMP: foreign IMP number.
- RMHOST: foreign and local Host numbers.
- RMCTL: handling type, foreign use number, and foreign block number (TM).
- RMMESS: message number, local use number, and age.

RSTATE: reply state bits for 8 messages in window.
RMTYPE: auxilliary reply state bits (with RSTATE).
RMLHN: index of local Host parameter block in H2PBLK.
TMBLKS: transmit message (TM) blocks, including:
TMLOCK: lock on the block.
TMIMP: foreign IMP number.
TMHOST: foreign and local Host numbers.
TMCTL: handling type, foreign use number, and foreign block number (RM).
TMMESS: message number, local use number, and age.
TSTATE: reset and init bits, allocate count, and message state bits.
TMSTP: stop bit and allocate timer for givebacks.
TMLHN: index of local Host parameter block.
MESSTK: pool of reassembly blocks, each containing:
REASLK: lock on the block.
RSF: number of packets so far.
REASST: reassembly block state.
RID: receive message (RM) block number.
REMESS: receive message number.
RUSE: RM block use number.
RMAX: highest packet number to get.
RAL: number allocated in this block.
REASQ, REASQE: queue of message packets.
RSFBT: count of total words of message so far.
TRNBLK: pool of transaction blocks, each containing:
TRSTAT: transaction block state.
TRTYPL: reply type to Host.
TRHSTL: message number and TM block for this transaction block.
TRDEDS: dead subcodes for control message to Host.
DEADSC: Host dead status codes.
HIHD: Host state.
HINPID: Host to IMP PID.
INCH: packet source, set to 0 for immediate replies.
CHAN: set to non-zero for immediate replies (can't refuse).
QT: queue time for trace.
BUFB: pointer from buffer to RM block.
CNTRS: buffer counters for reassembly and background.
WHERE: buffer use bits and count index.
NAL: allocate count.
EHPQ,EHQ: Host priority, regular output queues.
LOCKIH: IMP to Host software lock.
HOTPID: IMP to Host PID.
MYIMP: IMP coming up counter.
RUT: best delay route by IMP.

MINE: my IMP number.
HACSPC, HACCOM, HACMEM: Host access control tables.
H2PBLK: table of Host parameter blocks.
HOMODE: Host new/old leader format mode switch.

I/O performed

None.

5.7.2 Back Hosts

Function

The Back Hosts (named from "background") perform various functions related to end to end message processing. When control messages require reserving resources, or timing out idle resources, Task itself can't perform the function. The Back Hosts were created for these resource-reserving and freeing functions.

Control Structure

Each Back Host has a parameter block, which is very similar to the beginning of a real Host parameter block. Indeed, the Back Hosts share quite a bit of code with the Host to IMP process for real Hosts. This is because Back Hosts can produce packets to give to Task just as real Hosts can.

Entry Points

All Back Host PIDs dispatch to BACK, which obtains the proper parameter block from MBLKS and returns to the proper point in the Back Host. Each Back Host runs as a coroutine.

External Calls

See individual Back Host descriptions following.

Initalization

Each Back Host dispatch is initialized to the top of its loop.

Cleanup

See individual descriptions.

Data Structures

Local Data

None.

Shared Data

LOCKHI: lock on each Back Host.
BMESSB: index of current message block.
HITT: Back Host timer.

I/O Performed

None.

5.7.2.1 Back Host 5

Function

Send RFNMs, allocates, destination deads, and incomplete replies.

Control Sturcture

Scans all receive message (RM) blocks, starting one after the last block serviced (for fairness). For a block needing a reply sent, obtain the necessary resources (allocate, reassembly block, and buffer) and send the reply. If unable to obtain allocation resources after one half second, proceed to the next RM block (for piggyback attempts, send a RFNM with no allocate).

Entry Points

Coroutine, dispatch is through HILO.

External Calls

HIWM to dismiss coroutine.
BSET to set up common fields of message in parameter block.
BGETA to try to get allocation and a reassembly block.
REASAL to find a free reassembly block.
BSEND to obtain a buffer, construct the message, and give it to Task.

Initialization

HILO is initialized to dispatch to BACK5.

Cleanup

None.

Data Structures

Local Data

BMESSB: current RM block being tested.
BPKTH: PKTH word for control message.
BSEQH: SEQH word for control message.

Shared Data

RMLOCK: RM block lock.
RSTATE: RM block state, 2 bits per message.
RMIMP: RM block remote IMP number.
RMTYPE: RM block substate, 2 bits per message.
RMMESS: RM block message number, use number, and age.
CNTRS, NRE: reassembly count.

NAL: allocate count.
NF: number of free buffers MINF: number of guaranteed buffers total (lower limit for NF).
RAL: reassembly block allocate count.
RID: reassembly block RM block pointer.
RUSE: reassembly block RM use number.
RSF: reassembly block count of buffers received.
REASST: reassembly block state.
REASLK: reassembly block lock.

I/O Performed

None.

5.7.2.2 Back Host 6

Function

Send incomplete query messages for transmit message (TM) blocks whose incomplete timers have reached zero.

Control Structure

Each time it is entered, Back Host 6 checks all the TM blocks for any whose incomplete timers have reached zero. If one is found, Back Host 6 marks the corresponding transaction block for the oldest outstanding message, gets a buffer and sends an incomplete query message for that message.

Entry Points

Coroutine, entered from BACK via HILO.

External Calls

HIPOK to poke itself and dismiss.
HIWM to sleep until the next 25.6 ms wakeup from Back Host Timeout.
BFIXT to set up the common message fields in the parameter block.
BSEND to get a buffer, construct the control message, and give it to Task.

Initialization

HILO is initialized to dispatch to BACK6.

Cleanup

The message timeout field for idle RM blocks, or blocks that have no outstanding messages, is set to 1 if it was 0.

Data Structures

Local Data

BMESSB: current TM block to work on.

Shared Data

TMLOCK: lock on TM block.
TSTATE: TM block state bits.
TMIMP: TM block remote IMP number.
TMMESS: TM block message number, use number, and age.
TRNBLK: transaction block pool.
TRSTAT: transaction block state.
TRHSTL: transaction block message number and TM block.
TRMIDL: transaction block message-id and subtype.

I/O Performed

None.

5.7.2.3 Back Host 7

Function

1) Compute age "clips" beyond which to reset RM and TM blocks, based on how many free blocks there are left.
2) Send resets (for TM blocks) or reset requests (for RM blocks) for any blocks that have reached the corresponding clip.

Contorl Structure

Each time it is awakened, Back Host 7 recomputes the two age clips. It then scans all message blocks for any old enough to reset. If any are found, the reset or request is sent, and control returns to the beginning. If no more resets or requests need to be sent, Back Host 7 waits 640 ms before trying again (since this is how often the blocks can be aged).

Entry Points

Coroutine, awakened via HILO from BACK.

External Calls

1) B7SUB to maintain the clip values.
   HIPOK to poke itself and dismiss.
2) BFIXT, BFIXR to set up common message fields in parameter block.
   BSEND to obtain a buffer, construct the message, and give it to Task.

Initialization

HILO is initialized to dispatch to BACK7.

Cleanup

None.

Data Structures

Local Data

BSEQH: SEQH for control message.
RCLIP: RM block age clip.

Shared Data

TCLIP: TM block age clip.
TMIMP: TM block remote IMP, or minus if free.
TSTATE: TM block state bits.
TMLOCK: lock on TM block.
TMMESS: TM block message number, use number, and age.
RMIMP: RM block remote IMP, or minus if idle.

RMLOCK: RM block lock.
RMMESS: RM block message number, use number, and age.
HITT: Back Host software timer.

I/O Performed

None.

5.7.2.4 Back Host 9

Function

Back Host 9 sends giveback messages for allocates that haven't been used for 150 ms, or when more than 2 are being held in a TM block.

Control Structure

Back Host 9 scans all the TM blocks for any allocates that should be given back to the remote IMP. If it finds any, it obtains a transaction block, removes the allocate from the TM block, reserves a message number, and constructs and sends the giveback message.

Entry Points

Back Host 9 is a coroutine, awakened from BACK via HILO.

External Calls

HIWM to dismiss.
BFIXT to set up the common message fields from the TM block.
TRNPUT to allocate a transaction block.
BSEND to obtain a buffer, copy in the message, and give it to Task.

Initialization

HILO is initialized to dispatch to BACK9.

Cleanup

TSTATE is cleared for idle blocks to speed up the search.

Data Structures

Local Data

BMIDH: MIDH word for control message.
BPKTH: PKTH word for message.
BSEQH: SEQH word for message.

Shared Data

TMSTP: TM block stop flag.
TMLOCK: TM block lock.
TSTATE: TM block state.
TMIMP: TM block IMP number.
TMMESS: TM block message number, use number, and age.
TCLIP: TM block age clip (from Back Host 7).
TRLUSE: transaction block use number.

TRHSTL: transaction block message number and TM block.
TRSTAT, TRNTIM: transaction block state and timeout.

I/O Performed

None.

5.8 Routing

Function

1) Process buffers on the routing input queue.
2) Update internal routing state tables.
3) Recompute new routing output buffers as needed, and free old ones as they fall into disuse.

Control Structure

Routing runs once for each entry of each input message. It pokes itself if there is more to do, once it has picked an entry to process. Modem to IMP pokes it each time a new routing buffer is queued. Slow Timeout pokes it once every 640 milliseconds.

Entry Points

Entered at ROUTE.

External Calls

1) DEQUE to get next routing buffer.
   FLUSHB to free completed routing buffers.
   UNPCKC to access current buffer.
3) FREGET to get a new buffer for next routing output buffer.
   SUBCHN to compute checksum on it.
   FLUSH to free old output buffers.

Initialization

None.

Cleanup

Old routing output buffers and completed routing input buffers are freed.

Data Structures

Local Data

1-2) RUTJOB: index of current entry to process.
     RUTBUF: current input buffer in process.

Shared Data

1) SRUTQ: routing input queue head.
   LOCKR: lock on routing parameters.
   MINE: my IMP number.
   LSTATE: line state of input modem.
2) HOPDEL: table of current hops and delay by IMP.
   LOCALD: delay on input line (queue length).
   RUT: current best delay route by IMP.

DELTIM: delay hold-down timer by IMP.
THSDEL, OLDDEL: tables of "deltas" for smoothing delay.
MODEM: input modem number.
HOPRUT: current best hop route.
HOPTIM: hop route holddown timer.

3) LOCKRO: lock on routing output queue.
RUTOBF: queue of routing output buffers.
RNOBUF: number of missed routing updates due to no buffers.

I/O Performed

None.

5.9 Fake Hosts

Function

Fake Hosts are processes that simulate the action of real Hosts on the IMP, in that they accept and produce messages through simulated "1822" interfaces. Each Fake Host consists of two processes, one for the IMP to Host messages and one for the Host to IMP messages. Each is a coroutine, and is assigned a PID level. The main dispatch for the IMP to Host routines is WAITW, and for the Host to IMP routines DOZEW.

Emtry Points

PIDs for the Fake Host IMP to Host routines come to WAITW, which sets the Fake page map and jumps to FWAITW. Here, the appropriate Fake Host parameter block is loaded from MBLKS and the process resumed through WAITT. The corresponding Host to IMP side routine is DOZEW, which sets maps and jumps to FDOZEW. Again the proper parameter block is accessed and control passed through DOZET.

External Calls

None.

Initialization

See individual Fake Host descriptions.

Cleanup

None.

Data Structures

Local Data

WAITT: saved return to coroutine (IMP to Host side).
WAITT3: temporary storage for R3 over coroutine dismisses.
DOZET: saved return to coroutine (Host to IMP side).
DOZET3: temporary storage for R3.

Shared Data

MBLKS: table of parameter blocks by PID level.
LOCKF: lock on Fake Host parameters.

I/O Performed

None.

5.9.1 TTY Fake Host to IMP

Function

Accepts characters from the teletype input process and sends them as messages (one character per message) to the current crosspatch destination. If a semicolon is read, uses a separate leader and sends characters as a single message until a second semicolon is read (so-called "semicolon message"). If a null (code 80!, a control-@) is typed, it is sent and the crosspatch destination is reset to be the DDT Fake Host in the same IMP.

Control Structure

Characters are taken from the teletype input buffer and passed through the Fake Host interface.

Entry Points

Coroutine, entered via DOZET from DOZEW.

External Calls

T2FG to get next input character.
JAMLED to send leaders.
JAM to send each word of message (one character per word).
JAMEND to send last word of message and padding.

Initialization

DOZET is initialized to dispatch to FTH. Crosspatch leader is initialized to have this IMP's DDT as destination.

Cleanup

Crosspatch leader is reinitialized if a null was typed.

Data Structures

Local Data

TTCR: next character to send.

Shared Data

CCLED: current crosspatch leader.
MINE: my IMP number.
CLLED: current leader for semicolon messages.

I/O Performed

None.

5.9.2 TTY Fake IMP to Host

Function

Messages to the TTY fake (number 252 decimal) are accepted one word at a time and characters passed eight bits at a time to the teletype handler to be printed on the IMP terminal. If octal print is specified, input is interpreted as octal numbers and their values are printed (including the leader) eight words per line.

Control Structure

Each message is read via the Fake Host interface and passed in order to the teletype output buffer.

Entry Points

Coroutine, entered from WAITW via WAITT.

External Calls

SUCLED to read the message leader into a six-word save area.
SUCK to read the message data one word at a time.
FHTP to pass a character to the teletype output buffer.
PRINTC to print a carriage-return linefeed combination.
WORDP to print the octal value of a message word.

Initialization

Initial dispatch is to FHT.

Cleanup

None.

Data Structures

Local Data

FHTHLD: six-word leader buffer.
FHTT1: temp for first word of message.

Shared Data

TTYWHO: IMP number of last TTY to crosspatch to this IMP.
FLAGOP: octal print flag.

I/O Performed

None.

5.9.3 DDT Fake Host to IMP

Function

Accepts characters from the DDT process and sends them to the originator of the last message to DDT from the network. Characters are sent as s single message until terminated by semicolons, which the DDT Fake IMP to Host process sends through DDT. This ensures that a multicharacter response to a single DDT command all is sent to the proper source.

Control Structure

Characters are read from DDT via a single-character buffer, and passed through the Fake Host interface.

Entry Points

Coroutine, entered from DOZEW via DOZET.

External Calls

JAMLED to send a leader.
JAM to send each character as a word of the message.
JAMEND to send the last word of zero and padding.

Initialization

DOZET is initialized to dispatch to FDH.

Cleanup

None.

Data Structures

Local Data

FDHBSY: flag for state of message (awaiting first character or not).

Shared Data

D2FL: lock on character buffer from DDT.
D2FB: character buffer from DDT.
DDTLED: leader buffer, read in by DDT Fake IMP to Host.
BHPID: PID for this process.
D2FPOK: PID for DDT to poke when it puts a character into the buffer.

I/O Performed

None.

5.9.4 DDT Fake IMP to Host

Function

Reads messages from the network and passes them to DDT. As each message terminates, send a semicolon which, after DDT echoes it back, will cause the DDT Fake Host to IMP process to send a message.

Control Structure

The message leader is read into the DDT leader buffer. Then characters of the message, and the final semicolon, are passed one at a time to DDT.

Entry Points

Coroutine, dispatch from WAITW through WAITT.

External Calls

SUCLED to read a leader.
SUCK to read each word of the message.
F2DP to store a character in the buffer to DDT, and poke DDT.

Initialization

WAITT is initialized to dispatch to FHD.

Cleanup

None.

Data Structures

Local Data

None.

Shared Data

DDTLED: DDT leader buffer.

I/O Performed

None.

5.9.5 Packet Core Fake Host to IMP

Function

Accepts packet core packets from FHCQ that have arrived from our neighbor IMPs and sends them as messages into the network. Also polls block transfer for packet core messages from us if our own packet core process is active.

Control Structure

Checks block transfer state to see if packet core is active in our IMP, and if so, gets a free buffer and polls the process that constructs packet core messages from us. Otherwise, checks the packet core queue for any buffer from our neighbor to send. If any packet core message is found, it is sent via JAM and JAMEND into the network.

Entry Points

Coroutine, dispatch is through DOZET.

External Calls

FREGET to allocate a buffer for a possible packet core message from us.
PKCOC to construct a packet core message to send.
UNPCKC to access the message buffer.
FLUSHB to free the buffer if not needed, or when sent.
DEQUE to get buffers from the modem input packet core message queue.
JAMLIN to force modems out of line hold-down state.
JAMLED to send the message leader.
JAMEND to send the last word of message.
JAM to send each word of the message.

Initialization

DOZET is initialized to dispatch to FH2J.

Cleanup

None.

Data Structures

Local Data

FHCQBF: current buffer in process.
FHCQND: end pointer for that buffer.
FHCQLD: buffer to build message leader from packet core message.
FHCJA6: temporary for buffer pointer while sending.

Shared Data

BLTST: block transfer state.
BLTSTY: block transfer source type.
BLTDTY: block transfer destination type.
BLTLOK: lock on block transfer parameters.
CORIOB: parameter block for packet core Fake Host.
BHPID: PID for our process.
BLTPOK: PID for block transfer to poke when it's done.
SFHCQ: queue of received packet core messages.
INCH: modem from which buffer came.
MODEM: number of this modem.
M2NGHB: modem neighbor table.
BUFE: buffer end pointer.
MINE: my IMP number.
PCHELP: flag - destination to which to send "empty" packet core messages from our neighbors, cleared when used.

I/O Performed

None.

5.9.6 Packet Core Fake IMP to Host

Function

Accepts packets which control the loading and dumping of core areas within the IMP. When converted to special packet core messages, they may be sent to a specified malfunctioning neighbor IMP.

Control Structure

The message leader is read, and then the first word of data is read. If this word is negative, the message is processed as a packet core message, else it is for parameter change and thus is ignored (parameter change is not implemented yet).

Entry Points

Coroutine, dispatch is through WAITT from WAITW.

External Calls

SUCLED to read the message leader.
SUCK to read each word of the message.
FREGET to get a new buffer to construct the packet core message.
UNPCKC to acces this buffer.
WAIT to await a free buffer if none is available.
SUBCHN to compute the checksum on the message.
JAMLIN to prevent the modem state from entering line hold-down.
PKCIC to process packet core messages for us.
FLUSHB to free the buffer when the packet core message is processed.

Initialization

WAITT is initialized to dispatch to FH2S.

Cleanup

None.

Data Structures

Local Data

FHCSBF: current packet core buffer.
FHCSAD: where we are in this buffer.
FHCSTM: timeout counter for sending buffer to a neighbor IMP.

Shared Data

FH2LED: leader buffer area.

M2NGHB: neighbor table by modem.
M2PBLK: modem parameter block table.
SBLK: modem packet core buffer to send.
SROUTE: modem send-routing flag.
CORIOB: parameter block for packet core Fake Host.
BLTLOK: lock on block transfer parameters.
BLTST: block transfer state.
BLTSTY: block transfer source type.
BLTDTY: block transfer destination type.
HBPID: out PID value.
BLTPOK: PID for block transfer to poke when done processing the packet core input.

I/O Performed

None.

5.9.7 Statistics Fake Host to IMP

Function

1) Check active statistics routines to see if it is time to send them now. If so, send the leader and do the statistic.
2) Call one of the statistics routines:
   a) GENM, the message generator.
   b) TLOG, the TENEX logger routine.
   c) TRBL1, the status report.
   d) TRBL2, the throughput report.
   e) TRBL3, the anomalies report.
   f) DIAGRP, diagnostics report, containing bad packets.
   g) HOLDWD, a zero length packet to hold down the software watchdog timer.
3) Maintain a table of when each statistic was last called.

Control Structure

All the statistics routines are multiplexed onto one single Fake Host. Each routine is controlled by an on/off flag in the leader buffer area and network global time offset by the IMP number.

Entry Points

Coroutine, entered from DOZEW via DOZET.

External Calls

JAMLED to send the leader.
DOZE to sleep 25.6 miliseconds.

Initialization

All statistics are initialized to be off except TLOG and HOLDWD which are on.

Cleanup

None.

Data Structures

Local Data

STATN: the index into tables CAWL and STBP being used at the moment.
OLDS: when this statistic was done last.

Shared Data

MGSB, TLSB, ARSB, TRSB, SRSB, DGSB, HWSB: leader buffers and parameters for each statistic.

SYNC: global time.
ANOM: IMP anomalies word.
STATF: the frequency, in 25.6 ms clock ticks, of the statistic.
IMPOFF: the time offset of a report from this IMP.

I/O Performed

None.

5.9.8 Discard Fake IMP to Host

Function

Discard is a message sink - all incoming messages are ignored. Incoming RFNM's hold off the software watchdog timer; the statistics process HOLDWD sends messages that should periodically cause RFNM's if the system is operating normally.

Control Structure

Leaders are read into the discard leader area for checking for RFNM's. Messages are ignored by directly modifying the simulated Host interface parameters.

Entry Points

Coroutine, entered from DOZEW via DOZET.

External Calls

SUCLED to read a leader or control message.
SUCK to read single words of a message.
SUCPKT to signal end of reading a buffer of data.

Initialization

DOZET is initialized to dispatch to DISCRD.

Cleanup

None.

Data Structures

Local Data

COUNTD: total buffers discarded.
DISLED: leader buffer area.

Shared Data

SOWDTM: software watchdog timer.

I/O Performed

None.

5.10 Very Distant Host (VDH) Interface

Function

Implement the Very Distant Host interface described in Appendix F of BBN Report 1822. This includes handling the VDH modem interfaces, implementing the Reliable Transmission Package described in Appendix F, and efficiently interfacing the Reliable Transmission Package to IMP to Host and Host to IMP while at the same time simulating the Regular Host Interface as far as Host to IMP and IMP to Host are concerned. The VDH routine can be used for any Host as determined by hardware switch settings on the associated modem card. Modems with switches set to a number greater than the contents of location VD.CLP are VDH modems; their Host number is calculated by subtracting the contents of location VD.OFF from the number in the switches.

Control Structure

VDH is a separately loadable module within the IMP. When loaded VDH consists of four principal PID driven coroutines, plus seven special exits from standard IMP routines. These are described in detail below. Basically, Modem to VDH (M2V) accepts packets from a modem and puts them in 'slots'. VDH to Host (V2H) takes packets in order from the slots and passes them to the normal Host to IMP code (HI). When a packet has been accepted by the IMP, V2H arranges for an acknowledgement to be sent (an ACK). In the other direction, IMP to Host (IH) passes packets to Host to VDH (H2V) which puts them into (other) slots. VDH to Modem (V2M) transmits (and if necessary retransmits) packets in these slots until they are acknowledged. The exit routines provide for proper handling of VDH devices in configuration and checking routines, provide timing functions (both 25ms and 625ms intervals), and intercept Host timeouts to simulate hardware resets.

Data Structures

All VDH specific variables are contained in VDH blocks. These blocks are allocated, one per VDH Modem, from the dynamic allocation area. The start of each VDH block has the format of an IO-block, as described in the section on Fake Hosts. Following the IO-block section are variables which determine the state of the VDH process at any given time. These include a pointer to the associated Host block, the hardware address of the

modem being used, counters which record unusual events to aid in checking out new VDH code in the Hosts, the 'slots' mentioned above, and a Host-IMP Leader work area for each slot. (See Section 6.9 for detailed layout.)

Each (VDH) Host block contains a pointer to its VDH block in the variable IOBLOC. In addition, V2PBLK is an array of pointers to all VDH blocks, organized by internal host number, parallel to the similar H2PBLK array of pointers to Host blocks.

5.10.1 VDH Line Initialization Subroutine

Function

This subroutine is called to initialize a particular VDH interface. Initialization forces a line to go down and stay down long enough for the other side to recognize the problem and resynchronize. This state is called 'holddown'. During initialization the line state is cleared, slots are cleared, buffers returned, and fake device ready and busy bits are cleared to cause the HI and IH to reset as necessary. If the modem still exists, the modem hardware is reset.

This subroutine is called under several conditions:

1) HI or IH times out (25 ms timer IHTC); simulates a host hardware reset by entering holddown.
2) M2V recognizes the line has been looped (or unlooped); enters holddown.
3) The 625ms timer recognizes that too few I-Heard-You messages have received recently; enters holddown.
4) Configuration notices that the modem hardware (including spare if any) has disappeared; releases all resources. Configuration will report to the NCC that the host has disappeared.

Entry Point

Entered at VDINIT.

External Calls

Calls FLUSH to return outstanding buffers.

Data Structures

Local Data

None.

Shared Data

VLS, VCHNI, VCHNO, VCBO, VCBI, VOEBI, VOEBO are all cleared.
VLSTT is set to start holddown timing.
VACKI is cleared except for the Host/IMP bit.
STATIH and STATOH have their HBUSY and HREADY bits cleared.

I/O Performed

The VDH modem is reset.

5.10.2 VDH Exit Routines

Function

These exits allow Configuration to handle VDH modems and VDH fake hosts.

1) ~~VDHDV~~ extends FINDEV to recognize and accept VDH blocks as fake IO blocks.
2) ~~VDHCON~~ is an exit early in CONFIG. It goes though all VDH blocks checking that their modems still exist. Primary modem switch settings are checked; secondary modems are removed if non-existent or equal to the primary modem. When a primary modem disappears this routine switches in the secondary modem if any. If there is no secondary, we Trap and deconfigure the VDH fake device.
3) VDHCHK is called from CONFIG for each physical modem. It decides if the modem is a VDH modem; if not it returns immediately. Otherwise, it checks the host designated in the switches; if the host number is too big the modem is ignored. If the host is up it is checked to be sure that it really is a VDH host; if not we Trap. The BASE and MBLKS entries are established or checked; errors Trap. If the modem is not the primary device it is noted as the secondary modem in the VDH block. On the other hand, if the host is not up, this routine allocates and initializes a VDH block, calls BLDHST to set up a corresponding VDH Host, then starts over to set up and check the BASE and MBLKS entries.
4) ~~VDCBS~~ is an exit from the BASE and MBLKS checking routine in CONFIG. If BASE points to a VDH routine, the corresponding MBLKS entry is located in V2PBLK and the PID level is verified against the VDH block. The BASE entry is reset on failures.
5) VDHF is an exit from IHTC, and occurs every 25ms. It has several functions:
   a) Initialize the VDH state if IH times out (checks IHGOIN).
   b) Copy the IHLOOP (host looping control) word to VLOOP and to the modem.
   c) Poke the modem to hold it active.
   d) Count down packet retransmit times (STL timers in the VDH block) and add one to VSDUP whenever the timer goes to zero.
6) VDHITO is an exit from IHTC when HI times out. For VDH devices, it calls VDINIT to put the line into holddown.
7) VDHS is a normal exit routine from slow timeout (625ms). For each VDH host it handles the

Hello/I-Heard-You protocol timing, and simulates the fake host ready bit. The line can be in holddown, down, or up, as indicated by VLS. Seeing several I-H-Y messages brings the line up if down; missing several brings the line into holddown; holddown times out to down. While in holddown no Hellos are sent. Assembly parameters control the timing and number of successes required. When the line is up HREADY in STATIH in the VDH block is a one, otherwise it is a zero. Both VDH PIDs are set in part to clear possible lockups.

Entry Points

See above.

External Calls

VDHCON calls FINDEV to check if modem exists.
VDHCON, VDHF, VDHITO, and VDHS all call VDINIT to initialize the line state.
VDHCHK calls INBASE to set up BASE and MBLKS, BLDBLK to allocate a VDH block, and BLDHST to initialize the VDH host.

Data Structures

Local Data

None.

Shared Data

VDH block variables, Host block variables.
V2PBLK and H2PBLK arrays.

I/O Performed

Calls to VDINIT reset the modem hardware.
VDHF loops/unloops the modem and holds its watchdog timer active.

5.10.3 Modem to VDH Coroutine

Function

This coroutine (with V2H) is responsible for operating the data input side of VDH modems, and passing the information it receives to the other parts of VDH. Specifically, it:

1) Initiates a modem read operation into an IMP buffer.
2) Waits for the operation to complete.
3) Checks for and handles certain unusual events, such as checksum errors.
4) Restarts the modem input into a new buffer area (if possible), to provide overlap of I/O with processing.
5) Checks for a change in Loop state, and if so initializes the line.
6) Handles the four types of VDH message: Hello, I-Heard-You, Null, and Regular Data Packet.
   a) For Hello messages, sets a bit to tell V2M to send an I-H-Y.
   b) For I-H-Y messages, sets a bit which tells VDHS that an I-H-Y has been received.
   c) Nulls are handled like regular data packets until their ACKs have been processed, but are then discarded.
   d) Regular data packets are checked for validity, then their ACK word is saved in VACKI (in the VDH block). The ACKs are processed later by V2M. Duplicate packets are discarded; there are two cases, depending on whether or not V2H has as yet accepted the original packet. If VDH is out of its buffer allocation, step 4) has reused the current buffer area, so the contents are discarded at this point. But ordinarily, the packet is placed in the appropiate 'slot' depending on its channel number, to be processed further by V2H in turn; this ensures the correct packet order is maintained in the presence of garbled packets and retransmissions.
7) Maintains various counters in the VDH block:
   a) Too-long packets and modem quits Trap; these plus checksum errors (or overruns) are counted in VIMERS; the modem is reset.
   b) Too-short packets just Trap.
   c) All NOPs received are counted in VRNOP.
   d) All duplicates received are counted in VRDUP.
   e) If a packet is lost (will need to be retransmitted) due to M2V being out of buffer allocation, we Trap and add one to VRUIBF.

Control Structure

M2V is a strip which shares its PID with V2H; if M2V finds the modem has not yet read in a packet, it passes control to V2H. Logically, M2V is a coroutine driven by the modem hardware and driving V2H through the 'slots' data structure.

Entry Point

M2V is entered from LOOP at the tag 'M2V' when a modem input PID is set. M2V passes control on to V2H if the modem input has not yet completed; this allows M2V and V2H to share a PID level.

External Calls

FREGET to allocate a free buffer.
VDINIT to initialize on change of Loop state.
FLUSH to deallocate messages not put into 'slots'.

I/O Performed

VDH modem input is started and tested.
The modem input is reset on certain error conditions.

5.10.4 VDH to Host-code Coroutine

Function

This coroutine passes VDH packets to HI, by copying leader packets into transaction blocks or exchanging buffers containing data packets for empty HI buffers. V2H is polled every 25ms and when either M2V or HI want it to try to do something: When HI is ready for data, and a packet is available in the next channel's 'slot', the ACK bit for that channel is complemented, and the VSSACK bit is set in VSTAT to ensure that a NOP is sent soon even if there is no other traffic to send.

1) If HI wants a leader, the packet is copied into the transaction block supplied by HI, the HI transfer is marked complete, and the packet is discarded.
2) If HI has supplied a real buffer, it is exchanged for the data packet, and HI's buffer is freed. The allocation for the data packet is transfered from M2V to HI, making it possible for M2V to accept another packet. Finally, the HI transfer is marked complete.
3) If HI has supplied JUNK as its buffer, the data packet is discarded and the transfer is marked complete.

Control Structure

V2H is a strip which shares its PID with M2V; M2V gives V2H control if M2V has nothing to do. Logically, V2H is a dual coroutine driven by V2H through the 'slots' data structure, and driving HI through a fake I/O interface, simulating normal host hardware.

Entry Points

M2V transfers to the tag 'V2H'. The return is always to LOOP.

External Calls

FLUSH to deallocate buffers.

I/O Performed

None.

5.10.5 Host-code to VDH Coroutine

Function

This coroutine accepts IH leaders and data packets and passes them on to V2M in slots. H2V is polled every 25ms, or when either IH or V2M want it to do something. If IH has started its (fake) I/O transfer, the VDH line is up, and the next channel (in sequence) has its slot free, then data is accepted by H2V. There are two cases:

1) The data is in the IH leader send area; in this case the leader is copied to the leader holding area for this channel in the VDH block (VLDR).
2) The data is in a buffer; in this case the H2V buffer ownership bit is set for the buffer.

In either case the word before the data area is initialized to become the VDH header word. Then a pointer to the data area is stored in the channel's slot, the retransmission counter (STL) is cleared to indicate the information should be transmitted by V2M, the (fake) device complete status is set for IH, and the current channel number is advanced.

Control Structure

H2V is a strip which shares its PID with V2M; V2M gives V2H control if V2M has nothing to do. Logically, H2V is a dual coroutine driven by IH through the fake I/O interface, and driving V2M through the 'slots' data structure.

Entry Points

V2M transfers to the tag 'H2V'. The return is always to LOOP.

External Calls

None.

I/O Performed

None.

5.10.6 VDH to Modem Coroutine

Function

This coroutine is responsible for operating the data output side of VDH modems. It takes its data from 'slots' set up by H2V, and sends Hello, I-Heard-You, and NOP messages based on requests from other parts of VDH. Specifically, it:

1) Waits for modem output to finish (if any).
2) Checks for and counts output hardware errors (in VOMERS).
3) If necessary, sends an I-H-Y or Hello message.
4) Checks each channel for new ACKs; ACKs are checked for validity (Trap on failure), and ACKed data buffers are deallocated. Valid ACKs clear the associated channel's slot.
5) Occupied slots represent data which may need to be transmitted. A timer (STL) associated with each slot controls the retransmission interval. When a packet needs to be transmitted, V2M inserts the current ACK bits (from VOEBI, maintained by V2H) into the VDH header word, clears VSSACK in VSTAT, and starts the modem hardware.
6) If no packet needs to be transmitted, the VSSACK bit is tested to determine if an input side ACK needs to be transmitted in a NOP message. If it does, the counter VSNOP is advanced, and a NOP message is constructed and sent.

Control Structure

V2M is a strip which shares its PID with H2V; if V2M finds the modem output active, or has nothing to send, it passes control to H2V. Logically, V2M is a coroutine driven through the 'slots' data structure and various control bits in VSTAT, which drives the modem output hardware.

Entry Point

In LOOP, VDH output PIDs are dispatched to the tag 'V2M'. V2M passes control to H2V unless it starts up an output operation.

External Calls

FLUSH to deallocate acknowledged data buffers.

I/O Performed

VDH modem output is started, and is tested for completion and for error conditions.

5.11 Timeout

Function

1) Service the 25.6 ms clock PID and count the time in 25.6 ms units.
2) For dual-RTC machines, check for correct operation of each RTC.
3) Update in-core copies of the console lights.
4) Poke all modem output routines.
5) Poke all Fake Host processes.
6) Maintain line state timers, initiate sending of routing messages for each modem, and maintain modem interface hardware watchdog timers and low-order buffer address bits.
7) Poke the routines Display, Teletype, Back Host Timeout, and Host Timeout.
8) If it is not empty, append the Task retry queue to the Task queue and poke Task.
9) On "medium" timeout (every 5 ticks = 128 ms), maintain Host interface hardware watchdog timers and low-order buffer address bits.
10) Poke the slow timeout PID every 25 ticks (640 ms).

Control Structure

Timeout decides what to do based on the timer CYCLE. For each tick, functions 1-8 are performed. Function 9 occurs every 5 ticks, and function 10 every 25.

Entry Points

Entered at TOHOT, which sets maps for the RELY page and jumps to TOREL.

External Calls

6) JSRT does all the work
8) TSKPEP to unlock the Task lock and poke Task.
9) HPOKE does all the work.

Initialization

CYCLE is initialized to 9001! at startup and every 25 ticks.

Cleanup

2) If the backup clock PID wakeup determines that the main clock has stopped working, it assumes the functions of the main clock. The F-bus clock is always the preferred system RTC.

Data Structures

Local Data

2) CLKLOK: lock on checking variables.
CLK1UP, CLK2UP: timeout counters for main, backup clocks.
3) WATCH0: two-word save area for copy of console lights.

Shared Data

1) TIME: local time in 25.6 ms ticks.
SYNC: local version of network time, 25.6 ms ticks.
SYTIME: STAGE time in 25.6 ms ticks.
2) CLOCK: pointer to current system clock.
TIMEA: local time for alternate clock.
3) WATM1, WATM2: map settings for console lights pointers.
WATCH1, WATCH2: console light pointers.
CONFLG: flag to force Configuration to run before Timeout.
4) M2PBLK: table of modem parameter blocks.
MOTPID: modem output PID.
5) IOBASE: base addresses for I/O devices.
BHPID, HBPID: Fake Host software PIDs for DOZE, WAIT.
8) SRQ, ERQ: Task retry queue.
LRQ: Task lock.
ETQ: Task queue end pointer.
9-10) CYCLE: timeout state counter.

I/O Performed

Host and modem interface watchdog timers for active interfaces are held off, and their low-order buffer address bits maintained, by storing the proper values into the hardware status registers.

5.11.1 Host Timeout

Function

Performs timing functions for Host software, both IMP to Host and Host to IMP.

Control Structure

Each time it is poked from timeout, Host timeout initiates one pass through the Host parameter blocks. For fairness, each pass starts with a different block.

Entry Points

Starts at IHTC. Poked by the 25.6 ms timeout routine, and by itself until a pass is completed. Back Host Timeout enters at IHTC5 (see next section).

External Calls

Pokes Host input and output PIDs as needed.

Initialization

TCGO and TCGOA are initialized to 2.

Cleanup

Initiates resets of software when data transfers time out.

Data Structures

Local Data

TCGO: lock, current Host to service.
TCGOA: Host to end this pass with.

Shared Data

H2PBLK: table of Host parameter blocks.
IOBLOC: interface address for Host.
IHLOC: lock on IMP to Host hardware parameters.
IHTT: IMP to Host timer.
IHGOIN: flag to force IMP to Host software reset.
FAKE: flag for Fake, Back, VDH Hosts.
HIHD: Host state.
HOTPID: IMP to Host PID level.
LOCKHI: lock on Host to IMP parameters.
HITT: Host to IMP timer.
HIBITS: Host to IMP software state bits.
HINPID: Host to IMP PID level.

I/O Performed

If the Host to IMP hardware times out, it is reset.

5.11.2 Back Host Timeout

Function

Performs timing functions for Back Hosts.

Control Structure

Each time it is poked by timeout, Back Host timeout makes one pass through the Back Host parameter blocks. For fairness, each pass starts with a different block. Most of the routine is performed by code which is shared with Host Timeout (previous section).

Entry Points

Starts at BTC. Poked by 25.6 ms timeout routine, and by itself until a pass is completed.

External Calls

Pokes Back Hosts as needed.

Initialization

TBKGO and TBKGOA are initializaed to 2.

Cleanup

None.

Data Structures

Local Data

TBKGO: lock, current Back Host to service.
TBKGOA: Back Host to end this pass with.

Shared Data

BBKO-BBK3: Back Host parameter blocks.
LOCKHI: parameter block lock.
HITT: timer on Back Host.
HINPID: Back Host PID.

I/O Performed

None.

5.11.3 Slow Timeout

Function

Polls various routines which do timing functions, reliability checking, and various cleanups. The called routines are documented as subsections to this section.

Control Structure

Scans each common memory code page. The timeout table on each page, if any, lists routines to be polled every slow timeout period. Continues to poke itself until it completes a pass through all the routines.

Entry Points

Entered at TOSS, which dispatches in coroutine fashion to the current routine.

External Calls

None.

Initialization

TOA7 is initialized to dispatch to TOINIT.

Cleanup

None.

Data Structures

Local Data

SLOWTO: index into current timeout table.
SLOWMP: index into LMAP for current memory page.
TOA7: coroutine dispatch.

Shared Data

TOLOCK: Slow Timeout process lock.
LMAP: local table of map settings for common memory.
TOPNTR: pointer to timeout table for common memory page.

I/O Performed

None.

5.11.3.1 Teletype Buffer Check

Function

Keep values of teletype buffer pointers reasonable.

Control Structure

Keep teletype buffer pointers less than or equal to 31.

Entry Points

Entered at TTYFIX only.

External Calls

None.

Initialization

None.

Cleanup

None.

Data Structures

Local Data

None.

Shared Data

TTY2SS, TTY2SE, S2TTYS, S2TTYE: the teletype buffer pointers.

I/O Performed

None.

5.11.3.2 Reassembly Block Check

Function

Scan reassembly blocks for any associated with old messages or conversations.

Control Structure

For each reassembly block, lock the block and make sure it is in a legal state. Then check the associated receive message (RM) block and use number for consistency. If the RM block is idle, or has been reused for a new conversation, or the reassembly block message number is too old, the reassembly block is freed.

Entry Points

Entered at DEDREA only, from Slow Timeout dispatch.

External Calls

REASF to free a reassembly block and its associated buffers.

Initialization

None.

Cleanup

Blocks failing any of the above checks are freed.

Data Structures

Local Data

None.

Shared Data

REASLK: reassembly block lock.
REASST: reassembly block state.
RID: RM block for this reassembly block.
RMMESS: RM block message number and use number.
RUSE: reassembly block use number.
RMIMP: RM block source IMP number.
REMESS: reassembly block message number.

I/O Performed

None.

5.11.3.3 Host Access Checksum

Function

Verify checksum on Host access words tables.

Control Structure

Checksum the table and its checksum to produce zero, then clear the Host data checkum error bit in the anomalies word and return. A bad checksum causes the error in the checksum to be saved, and the Host data checksum error bit to be set in the anomalies word.

Entry Points

CONCHK, only.

External Calls

SUBCHN, the subtraction chain that produces the checksum.

Initialization

None.

Cleanup

None.

Data Structures

Local Data

CONCER: word to save checksum error.

Shared Data

ANOM: anomalies word.

I/O Performed

None.

5.11.3.4 Line State Timeout

Function

Maintain line state (LSTATE) words for each modem.

1) For each modem in the modem parameter block pointer table, infer the line speed on the basis of the elasped time to send a null message and save the speed in the line speed word.
2) If line is alive (LSTATE < 5), and received an I-heard-you (IHY), set LSTATE to 4. If no IHY, count LSTATE down one.
3) If line is coming up (4 < LSTATE < "half count", where half count depends on line speed), and got an IHY, count LSTATE down one, else count it up one.
4) If line is dead and not coming up (half count < LSTATE < twice half count) and got an IHY, set LSTATE to half count, else count it up by one and, if it reaches twice half count, set it to 80!.
5) If line is down (LSTATE > 80!) perform associated cleanup.
6) If line has been down about 8 seconds (LSTATE > 8C!) set LSTATE to half count to try to bring it up.
7) Check and clear the send-I-heard-you flag and use the result to increment (or not) the line error count.

Control Structure

Processing for each line begins with function 1. One of functions 2-6 is performed next, based on the line state and whether an IHY was received during the last slow timeout period. Then function 7 is performed.

Entry Points

DEDL only, called from Slow Timeout dispatcher.

External Calls

1) TSLEEP, provides strip break between modems.
5) LINEI1 is line initialization for lines recently in use.
LINEI2 is line initialization for lines down more than one slow timeout.

Initialization

None.

Cleanup

For any line that has gone down, all pending packets are rerouted by resubmitting them to Task. Any routes

that were using this line are marked for maximum hops or delay, and hold-down is entered, so that the "bad news" about this line will propagate to other IMPs in the network.

Data Structures

Local Data

DEDLC: temporary strip break storage.

Shared Data

M2PBLK: table of modem parameter block pointers.
RUTRAT: elasped time to send a null message.
CLOCKM: line speed word in modem parameter block.
LINCLK: clock words to synchronize routing.
LOCKM: modem parameter block lock.
LAC: IHY-received flag.
LSTATE: line state word.
SIHY: send-I-heard-you (routing received) flag.
RTRERR: line errors count (for NCC).
RTSSNT: routing sent count (for NCC).
MODEM: logical modem number.
RUT: best delay route by IMP.
HOPDEL: hops and delay word by IMP.
DELTIM: delay hold-down counter by IMP
HOPRUT: path for best hops by IMP.
HOPTIM: hop hold-down timer by IMP.
LOCKR: routing lock.

I/O Performed

None.

5.11.3.5 IMP to Host Software Check

Function

Check the IMP-Host data and dispatch structures.

Control Structure

For each Host, make sure that the last queue serviced was a legal queue.

Entry Points

RIHS only, called from Slow Timeout dispatcher.

External Calls

IHSINI to reinitialize IMP to Host parameters.

Initialization

None.

Cleanup

If an illegal queue was accessed, reinitialize the block.

Data Structures

Local Data

None.

Shared Data

H2PBLK: Host parameter block pointer table.
IHLOC: IMP-to-Host parameters lock.
IHWQ: last queue serviced in this block.
SHQ: start of Host queue of regular messages.
SHPQ: start of Host priority queue.

I/O Performed

None.

5.11.3.6 Central Dispatch Check

Function

Sets up and pokes some PIDs in common use.

Control Structure

Pokes certain other processes and then sets up the dispatches for several routines.

Entry Points

BASETP when called from Slow Timeout.
BASETO when called from Initialization (no PID pokes).

External Calls

None, but pokes the Restart, Block Transfer Poll, DDT Poll, Configuration, Task, and Routing PIDs.

Initialization

None.

Cleanup

None.

Data Structures

Local Data

None.

Shared Data

BASE: the table of dispatches, indexed by PID level.

I/O Performed

None.

5.11.3.7 Transaction Block Timeout

Function

Search for and free transaction blocks that are out of date or inconsistent.

1) If transaction block is reserved by a Host, check that some Host parameter block points to it.
2) If it is queued for transmission as a control message to a Host, be sure that Host is alive and has its pending control message count non-zero.
3) If transaction block is associated with a message in process, check that its parameters are consistent with the transmit message (TM) block indicated.

Control Structure

DEDTRN first checks the state of the transaction block to determine the present user of the block. One of functions 1-3 is then performed, based on the state, to determine if the transaction block is consistent with its "owner" (a Host parameter block or TM block).

Entry Points

DEDTRN only, called from Slow Timeout dispatcher.

External Calls

TRNFLS to free a transaction block and its associated buffer, if any.
LEDPF, to signal a Host for a control message.

Initialization

None.

Cleanup

1-2) If inconsistencies are found, a timeout counter for the transaction is decremented and, if it reaches zero, the block is freed.
3) If the block is found inconsistent, it is freed.

Data Structures

Local Data

None.

Shared Data

TRSTAT: transaction block status.
TRHSTL: message block pointer and message number.
TMLOCK: TM block lock.
TMIMP: TM block remote IMP number.

TMMESS: TM block message number and use number word.
TRLUSE: transaction block use number word.
TRHOST: Host on which this transaction block queued.
H2PBLK: table of Host parameter block pointers.
TRNTIM: transaction block timer.
LOCKHI: Host output software lock.
NXTLED: pointer to pending Host control message.
HITRAN: pointer in Host parameter block to reserved transaction block.

I/O Performed

None.

5.11.3.8 Real Host Ready Line Check

Function

1) Count down my IMP coming up timer.
2) Maintain Host status words for real Hosts.

Control Structure

1) If MYIMP is not zero, subtract one, else do nothing.
2) For each real Host, check the present value of the status word to make sure that it is not now reinitializating; then look at the hardware status for the state of the ready line. If the ready line is up, and the software status word says down, clear the software status word and clear the IMP-to-Host software timer to trigger an IMP-to-Host software reset. If the hardware is down, set the software status word to 1.

Entry Points

DEDH only, called from Slow Timeout dispatcher.

External Calls

None.

Initialization

MYIMP is set to 60 at startup.

Cleanup

When a Host first comes up, clear its saved dead status codes.

Data Structures

Local Data

None.

Shared Data

MYIMP: status of this IMP relative to the network; if not zero Hosts will be prevented from communicating with the rest of the network.
H2PBLK: table of Host parameter block pointers.
HIHD: software status word in Host parameter block.
IOBLOC: pointer to Host hardware.
DEADSC: dead subcodes, holds detailed software status word of dead Host.
IHTT: IMP-to-Host software timer.

I/O Performed

None.

5.11.3.9 Routing Timeout

Function

1) Set routes to us to zero.
2) Count down the routing hold-down timers.
3) Maintain state of each IMP (up, down, coming up, going down).
4) Maintain line for network time synchronization.
5) Synchronize line state clocks.

Control Structure

1) Using a table of our IMP numbers, the routes pointing to us are set to zero to assure that they won't be used.
2) The whole routing table is accessed sequentially, counting down the hold down timers for delay, then for hops.
3) By masking the hops/delay word to give only the hops, decide whether an IMP is reachable. Unreachable IMPs that are down get their state set to -1; unreachable IMPs that are up get counted up by two and if the count reaches 16 the IMP is declared down. Reachable IMPs that are up get their state set to two; Reachable IMPs that are not up get counted up to 7F! and then are declared up. When my own IMP number is encountered, the paths for best hops and delay are both set zero to preclude usage.
4) Finally after all IMPs in the table are done, search for the first live line and save its delay route; this line is used to maintain synchrony with the rest of the network for statistics.
5) Return to the dispatcher after synchronizing the fast clocks.

Entry Points

RTGO only, called from Slow Timeout dispatcher.

External Calls

TSLEEP to dismiss coroutine for a strip break.

Initialization

None.

Cleanup

None.

Data Structures

Local Data

RTGOT: temporary storage.

Shared Data

MINE: table of my IMP numbers.
RUT: table of best delay paths by IMP.
THSDEL: the change in delay this tick by IMP.
OLDDEL: the change in delay last tick by IMP.
DELTIM: delay hold-down timer by IMP.
HOPTIM: hop hold-down timer by IMP.
HOPDEL: hops and delay word by IMP.
ISTATE: IMP state table.
LOCKR: routing lock.
THD: modem number of line to lowest-numbered live IMP.
LINCLK: table of line clocks for sending routing.

I/O Performed

None.

5.11.3.10 Incomplete Message Timeout

Function

Time out transmit message blocks for incomplete messages.

Control Structure

Decrements message number timeout counter for transmit message (TM) blocks which are active and have outstanding messages. Three blocks are timed each slow timeout, so that each is checked every 12 seconds or so (for 56 TM blocks).

Entry Points

MESSTO only, called from Slow Timeout.

External Calls

None.

Initialization

None.

Cleanup

None.

Data Structures

Local Data

MESST: the index storage.

Shared Data

TMIMP: destination IMP number in TM block.
TMLOCK: transmit message block lock.
TSTATE: TM block message status and timeout word.

I/O Performed

None.

5.11.3.11 Routing Software Check

Function

1) Check routing delay tables for proper lines and non-zero delays except to us.
2) Check routing hop table for same.
3) Check IMP state table.

Control Structure

Checks entries for 8 IMPs each call, sleeping in between.

Entry Points

RSOFT, called from Slow Timeout dispatch.

External Calls

TSLEEP to dismiss every 8 IMPs.
MTES to see if a valid modem is given.
MINETE to compare an IMP number with our IMP numbers.

Initialization

None.

Cleanup

Any failures cause a trap and the offending entry in the table is set to -1.

Data Structures

Local Data

RSOFA1: saves place in table over strip break.

Shared Data

RUT: modem for best delay route by IMP.
HOPRUT: modem for best hop route by IMP.
ISTATE: state by IMP number.
HOPDEL: hops and delay by IMP.
DELTIM: hold down timer by IMP.

I/O Performed

None.

5.11.3.12 Buffer Counters Check

Function

1) Calculate minimum guaranteed buffer counters using the algorithms MINMI = m+21m+1, MINSF = 31m, MINHI = 1h, MINBAK = 3, MINRE = 1+81h, MINRUT = 2, where m = total modems, 1m = live lines, 1h = live Hosts. MINMI is minimum for Modem Input, MINSF for store-forward, MINHI for Host input, MINBAK for Back Host, MINRE for reassembly, MINRUT for routing output. When the sum of the counters above exceeds the number of buffers allowed, the algorithms used are: MINMI = m+1, MINSF = m, MINHI = 2, MINBAK = 1, MINRE = 9, MINRUT = 2.
2) Verify dynamic counters by scanning all buffers in use and adding to the minima from above. If discrepancies in the dynamic counters are found, smoothly adjust them until they match the results of the scan.

Control Structure

1) Modems and live lines are first counted and their contributions to MINMI and MINSF are summed. Next the Hosts are treated similarly, contributing to MINHI and VDHI (VDH Host input, 3 for each Host) if a VDH. When all are counted, MINRE is calculated from MINHI and all the counters are summed, assumming MINRUT+MINBAK = 5. The result of the sum is compared to MAXNF, the allowed buffers, and if an overflow occurs, the counters are modified as per specification above. The results, recalculated if necessary, are stored as the trial counters. Two more counters for the optional PTIP are stored and their effect on the sum accounted. The sum of the counters is put into MINF, and a strip break occurs.
2) On return from the break, count the actual buffers in use by indexing through the buffer ownership table and incrementing the proper counters. After counting all buffers, all the trial counters are compared to the actual counters and the difference adjusted by the smoothing routine. The differences are then used to update the actual counters. The updated counters are summed and the sum smoothed to give NF.

Entry Points

RCNTRS only, called from Slow Timeout.

External Calls

TSLEEP for a strip break.

FIXER is the smoothing routine.

Initialization

None.

Cleanup

Counters found to be in error are statistically modified towards their proper value.

Data Structures

Local Data

NEWCTS: storage for trial counters.
TOTMP1: temporary storage.
TRENDS: trends used in smoothing routine.
TRENDF: trend for smoothing NF.

Shared Data

M2PBLK: table of modem parameter blocks.
LSTATE: variable in modem block for line state.
H2PBLK: table of Host parameter blocks.
HIHD: Host status word in Host block.
FAKE: VDH if non-zero.
MAXNF: number of buffers allocated by Buffer Initialization.
MINF: total reserved buffers.
MAXBUF: number of buffers allocated * 2.
WHERE: buffer ownership table.
CNTRS: the actual dynamic counters.
NF: length of free list.

I/O Performed

None.

5.11.3.13 Allocate Count Check

Function

Maintain the count of allocated buffers.

Control Structure

Buffers are counted by checking all the reassembly blocks and summing assigned buffers. The difference between the new count and the previous count is smoothed by a smoothing routine and then used to update the previous count.

Entry Points

RNAL, polled from Slow Timeout.

External Calls

FIXER to smooth the buffer count.

Initialization

None.

Cleanup

None.

Data Structures

Local Data

TRENDA: the trend used in smoothing.

Shared Data

MESSTK: the reassembly block area.
REASST: reassembly block status.
REASLK: reassembly block lock.
RAL: reassembly block allocate count.
NAL: the total allocate count.
NF: free list lock and counter.

I/O Performed

None.

5.11.3.14 Modem Queue Check

Function

Count queues held by each modem to verify slots count.

Control Structure

For each block in use, starting at the end of the modem parameter blocks, first calculate the number of slots in use on the basis of the maximum channels and free slots. Then, using seperate counters, the number of busy channels is counted and subtracted from the number of slots in use, and the number of buffers in the sent queue plus the buffer being sent (if any) is also subtracted from the slots in use. The resultant two numbers are checked against each other and should be equal. The regular queue and the priority queue are next counted and subtracted from the present count to result in zero, confirming the count of slots in use.

Entry Points

MQCNT only, called from slow timeout.

External Calls

COUNTQ to count the number of buffers on a queue.

Initialization

None.

Cleanup

In the case that either of the checks fail, trap and set the line state dead.

Data Structures

Local Data

None.

Shared Data

M2PBLK: table of modem parameter block pointers.
LSTATE: line state.
CHNBSY: table containing bit flags for busy channels.
MAXCHN: maximum channels this modem.
SLOTS: free slots.
SNDING: buffer being sent.
LATER: flag to indicate current buffer (SNDING) will be flushed soon.
SSENTQ: start of retransmit queue.
SREGQ: start of regular queue.

SPRIQ: start of priority queue.
LOCKM: modem block software lock.

I/O Performed

None.

5.11.3.15 Buffer Timeout

Function

1) Check the POINT word for every buffer in use and make sure it corresponds to the areas allotted by memory management.
2) Check and see that every buffer has been flushed at least once every slow timeout.
3) Count and check the free list.

Control Structure

Runs off the slow timeout dispatch, checking eight buffers on each call. The eight buffers are specified by a counter, BUFTIM, which gets incremented by sixteen each pass. BUFTIM is maintained modulo 4096 and therefore wraps around every 256 calls, to result in a consistent loop time. Each buffer is passed to subroutine CHKPNT which performs function 1). Next the FLUSHD word of the buffer is checked and, if zero, the buffer has remained out of circulation for too long and is freed. If the FLUSHD word is non-zero, it is just cleared (normal case). After all the buffers have been checked, the list of free buffers is counted and checked. Checks of the free list consist of making sure the start pointer is legal, then checking buffer ownership and counting the number of free buffers.

Entry Points

BUFT only.

External Calls

CHKPNT: to check a buffer pointer.
FLUSH2: to flush and requeue an unflushed buffer.

Initialization

None.

Cleanup

1) Set the initialization flag, INTIME, to zero, to cause page reinitialization.
2) Free the timed-out buffer.
3) Errors in the free list cause traps and the free list is returned to a legal state.

Data Structures

Local Data

BUFTIM: a pointer keeping track of which eight buffers are to be tested next.

Shared Data

CHAIN, FLUSHD, WHERE: buffer system tables.
MAXBUF: max number of buffers * 2.
FREE, FREEND: beginning and end of list of free buffers.

I/O Performed

None.

5.11.3.16 Trace Buffer Check

Function

Check the variables used for tracing packets.

Control Structure

Find the present value of the buffer pointer and compare it with the limits of the buffer area assigned to the trace routine. Any overflow or underflow causes a trap and resets the pointer to the beginning of the buffer.

Entry Points

TRACH only, called from Slow Timeout.

External Calls

GOTRAC, to initialize and unlock the trace buffer pointer.

Initialization

None.

Cleanup

Failure results in resetting the buffer pointer.

Data Structures

Local Data

None.

Shared Data

TRCPTR: the trace buffer pointer.
TRCLOK: the trace buffer lock.
TRCBUF: the trace buffer.

I/O Performed

None.

5.11.3.17 Age Message Blocks

Function

Check and time out message blocks.

Control Structure

The timing out of each message block is done by the routines, AGETM and AGEING, called by BT0 from within a loop covering all the transmit (TM) and receive (RM) message blocks. Then, various checks on the block state are performed. Every eight blocks the routine goes to sleep.

Entry Points

BT0 only, called from Slow Timeout.

External Calls

TSLEEP for a strip break.
AGETM, AGEING to count block age.

Initialization

None.

Cleanup

Blocks with invalid IMP or local Host numbers are unilaterally freed, as are blocks associated with IMPs that have gone dead in routing. TM blocks for dead destination Hosts that have reached age 4 are freed. Blocks associated with local Hosts that have gone down are set to maximum age to trigger a reset of the conversation.

Data Structures

Local Data

BT0A1: strip break storage for timer flags.
BT0A5: strip break storage for block index.
BT0A7: strip break storage for return from AGETM or AGEING.

Shared Data

TMLOCK: transmit message block lock.
RMLOCK: receive message block lock.
TSTATE: TM block transmit status.
TMMESS (RMMESS): TM (RM) block message number, use number, and age.
TMIMP (RMIMP): destination (source) IMP.
ISTATE: IMP state table.

H2PBLK: table of Host parameter block pointers.
HIHD: Host status.

I/O Performed

None.

5.11.3.18 IMP-going-down Message Check

Function

Send an IMP-going-down message to all Hosts.

Control Structure

For each Host in operation, get a transaction block, fill it with the proper leader and code, and put the leader on the control queue.

Entry Points

Called at IMPDWN from Slow Timeout dispatch.

External Calls

TSLEEP to poke Slow Timeout and dismiss.
LEDP0 to place a leader on the control queue.

Initialization

The IMP-going-down flag is set zero.

Cleanup

The IMP-going-down flag is turned off after all the messages are sent.

Data Structures

Local Data

IGDTMP: temp for this routine. Saves a register over a strip break.

Shared Data

IGDOWN: the IMP-going-down flag.
H2PBLK: Host parameter blocks.
HIHD: Host status word in the parameter blocks.
TRNLOK: transaction block lock.
TRNBLK: pointers to transation blocks.
TRSTAT: status of a transaction block.

I/O Performed

None.

5.11.3.19 Statistics Check

Function

Maintain the IMP offset time for the statistics reports, so that each IMP reports its statistics at a time staggered from the other IMPs.

Control Structure

For each statistic that is sending on a relatively long frequency, calculate the offset for this IMP based on its IMP number.

Entry Points

GOSTAT only, called from Slow Timeout.

External Calls

None.

Initialization

None.

Cleanup

None.

Data Structures

Local Data

None.

Shared Data

MINE: my IMP number.
STATF: frequency of a statistic.
IMPOFF: offset for this statistic.

I/O Performed

None.

5.11.3.20 Restart Buffer Check

Function

Verify that the ring buffer for the Restarter process is valid.

Control Structure

Look to see if the ring pointer is greater than the begining of the ring buffer and less than the end of the structure. Then check that the counter is greater than 0 and less than the length of the structure.

Entry Points

RNGCHK, called from Slow Timeout.

External Calls

None.

Initialization

None.

Cleanup

If either failure above occurs, reinitialize the ring structure and trap.

Data Structures

Local Data

None.

Shared Data

RINGF: the ring pointer.
RING: the ring buffer.
RINGC: the ring counter.
RINGLN: the length of the ring.
RINGE: the end of the ring.

I/O Performed

None.

5.11.3.21 Fake Host Software Check

Function

Make sure fake Hosts are working right.

Control Structure

For each fake Host, rewrite the constants for that fake and check the dispatches to see that legal dispatches are being taken.

Entry Points

RFAKE only.

External Calls

BLDFH, to rewrite constants.
RETURN, to check a dispatch.

Initialization

None.

Cleanup

If a bad dispatch is found, trap and remove this fake from the IO tables.

Data Structures

Local Data

RFAKT3, saves the index to the fakes table over a strip break.

Shared Data

FAKEIO: table of fake Hosts.
IIOBL3: pointer to IO parameter block.
LOCKF: fake Host lock.
DOZET: contains the return from a strip break for this Host.
WAITT: contains the return from a strip break via WAIT rather than DOZE.
WAITT3: contains the return from SUCK if SUCK called WAIT.

I/O Performed

None.

5.11.3.22 Back Host Software Check

Function

Test the Back Host parameter blocks.

Control Structure

For each Back Host parameter block, set up various constant data such as its PID level and dispatch in BASE.

Entry Points

BKTST only, called from Slow Timeout.

External Calls

BCOMT, to do the work for each parameter block.

Initialization

None.

Cleanup

None.

Data Structures

Local Data

None.

Shared Data

MBLKS: parameter block table parallel to BASE.
BASE: main dispatch table, indexed by PID level.
FAKE: flag word within Host parameter block for Fake, Back Hosts.
LOCKHI: Host to IMP (or Back Host) software lock.

I/O Performed

None.

5.11.3.23 Trouble Report Checks

Function

1) Check the status of the NCC IMP; if up, turn on the trouble reports(TRBL1 , -2, -3), else turn off.
2) Compare the anomalies word with its value at the last anomalies report; if different, or if any Host state has changed, or if any trap has occured, send an anomalies report immediately.
3) Check to see if any traps have occurred; if so send a trap report immediately.
4) Maintain a check on how often the reports are sent; if too often, restrict the reports.

Control Structure

TRBTIM is polled from Slow Timeout and proceeds in a straightforward manner.

Entry Points

Called at TRBTIM by Slow Timeout dispatcher.

External Calls

None.

Initialization

None.

Cleanup

None.

Data Structures

Local Data

THROTC: 'throttling' count to limit number of reports.

Shared Data

BANOM: value of ANOM word sent in last report.
ANOM: anomalies word.
OPHGO: flag to send anomalies report if Host state changes.
SRSB, TRSB, ARSB: leader buffer areas for status, throughput and anomalies (traps) reports.
STATF: frequency of statistics; when set to 0, report is sent immediately.
SNON: flag to turn on/off a statistic.

I/O Performed

None.

5.11.3.24 Light Display Check

Function

Produce words to place into console DATA lights.

Control Structure

1) Using a word initially set all ones, turn off a bit for every running Host, lowest numbered Host (Host 0) being the highest order bit. Save the word (for displaying 16 Hosts' status). Reset the high order byte of the same word to all ones, and turn off lights for lines up, again lowest modem (modem 1) being highest order bit. Save this word.
2) Once these are done, check light pointer words..

Entry Points

LITES only, polled by Slow Timeout.

External Calls

None.

Initialization

None.

Cleanup

If lights pointers produce quits, reset to default values.

Data Structures

Local Data

None.

Shared Data

H2PBLK: table of Host parameter block pointers.
HIHD: Host status.
WATCHH: word for Host lights.
M2PBLK: table of modem parameter block pointers.
LSTATE: line state.
WATCH: Host and line state lights word.
WATM1: map setting for upper lights.
WATM2: map setting for lower lights.
WATCH1: light pointer for upper (ADDRESS) lights.
WATCH2: pointer for lower (DATA) lights.

I/O Performed

None.

5.11.3.25 Nice Stop Check

Function

Checks for a nice-stop or panic-stop request and performs the desired action. The nice-stop process attempts to halt IMP processing gracefully, while a panic-stop does so instantly. When stopping is complete, the system will either halt each processor, request a reload, or restart the IMP.

Control Structure

NSCHK runs as a coroutine to Slow Timeout. At initial entry NSCHK simply checks the nice-stop flag; if it is zero nothing happens and NSCHK returns. If positive, a panic stop is in process and an exit to the panic stop follows; if negative, a nice-stop is begun and the control structure is modified to run a stage of the nice-stop every five seconds. The nice-stop process consists of informing users and neighbors of the upcoming interruption of service, resetting devices, and halting operations prior to halting each processor. The first stage is simply setting the IMP-going-down flag and then waiting five seconds for the messages to be sent. Next the Hosts are turned off by setting their status to be "not initialized" and setting the IMP status word negative. Stage three of the nice-stop is halting the store/forward traffic by removing their buffers from circulation. The modems are halted next and finally the Slow Timeout dispatcher is backed up so that the next dispatch is to NSCHK again. The processor running the code then pokes the Slow Timeout PID, so that the next processor will arrive at this point too. If a halt is desired, each processor inhibits level one and four interrupts and halts. Otherwise, each processor reenters STAGE to perform a restart or reload.

Entry Points

NSCHK is the nominal entry point. If a nice-stop is already in progress, coroutine control passes via NSCRT to the proper point of the process.

External Calls

NSWT to return wait five seconds before the next stage of nice-stop.
NSRET to dismiss the coroutine.

WS, WST to reenter STAGE for restart, reload respectively.

Initialization

NSCRT is initialized to dispatch to NS1.

Cleanup

None.

Data Structures

Local Data

NSCRT: contains the next dispatch.
NSWTIM: a counter to count down the five second wait between stages of the nice-stop.
NSWRET: saves the return from NSWT.

Shared Data

NSRTF: nice-stop request flag,
IGDOWN: IMP-going-down flag.
H2PBLK: table of Host parameter blocks.
HIHD: Host state.
MYIMP: IMP status of this IMP; negative for nice stop in progress.
NF: number of free buffers.
MAXNF: total buffers in system.
NSF: store/forward buffer count.
M2PBLK: table of modem parameter blocks.
LSTATE: line state for modem.
SLOWTO: Slow TImeout dispatch index.
TOLOCK: Slow Timeout lock.
LOCALC: checksum on local code.
MAPCOM: communications page map.
MEMSEG: bit table of existant memory.
CKSUM: common memory page checksum.

I/O Performed

None.

5.12 Initialization

Function

Prepare variables and data structures for use by IMP system; also reset same in case of drastic failure.

Control Structure

Before initialization, checks are made to see if initialization is required. The checks determine timeout status, debugging status, map table consistency and look for manual requests for initialization. If initialization is not needed, and the common memory snapshot area is vacant, copy any local snapshot to common and clear the local snapshot buffer. If initialization is desired, the procedure is as follows:

1) Return to Stage AR and check FIXIT word. If allowed to do the fix, return to the init routine and start.
2) Look for other running processors, if insufficient numbers, wait for more.
3) Zero variables page, buffer tables.
4) Empty base table.
5) Fill configuration tables and routing tables with -1s.
6) Zero variables on fake page.
7) Set up the Fake Host leader buffers.
8) Unlock locks.
9) Set queues up; point endpointer at begin pointer, set begin pointer to 1.
10) Set up constant dispatches in the central dispatch table.
11) Find and initialize a real-time clock (RTC); find IMP number and check it.
12) Initialize reassembly blocks.
13) Initialize message blocks.
14) Initialize buffer tables.
15) Initialize trace blocks.
16) Initialize Back Hosts.
17) Remove I/O blocks for the Fake Hosts.
18) Initialize statistics Fake Host.
19) Calculate the statistics offset time for this IMP.
20) Trap to note completion of initialization.

Entry Points

Entered from Stage AR at INITCK. If initialization is to be performed, return after that stage reaches consensus is at INIT.

External Calls

PCOUNT to count procs.
BASETO to initialize certain invariant software PID level entries in BASE.
CONCLK to find the RTC and IMP number.
CONINI to initialize the Configuration dispatch.
BUFINI to initialize the buffer structures.
FLUSH2 to flush the newly created buffers.
GOTRAC to initialize trace routines.
BREINI to initialize each Back Host.
GOSTAT to set up statistics constants.

Initialization

None.

Cleanup

None.

Data Structures

Local Data

VMAP: map settings for variables and buffers pages at initialization time.

Shared Data

All variables on the common variables page, second variables page, and fake code page are cleared. Then the following are initialized:
BASE: the PID dispatch table - "BAD", the dispatch for unexpected PIDs.
H2PBLK, M2PBLK, V2PBLK: tables of Host, modem, VDH modem parameter blocks - -1 (for no block).
RUT, HOPDEL, HOPRUT, ISTATE: Routing best delay route, hop and delay counts, best hop route, and IMP state by IMP - -1 for "dead".
WATCH1, WATCH2: light pointers - WATCHS, WATCH (normal light display).
DYNXT: dynamic parameter blocks pointer - DYBLKS (beginning of free area for blocks).
CYCLE: Timeout counter - 9001!.
TCGO, TCGOA: Host Timeout control - 2.
TBKGO, TBKGOA: Back Host Timeout control - 2.
TOA7: Slow Timeout dispatch - TOINIT.
MYIMP: my IMP coming up counter - 60.
NSCRT: nice-stop coroutine dispatch - NS1.
CCLED: teletype Fake Host leader - DDT Fake Host on this IMP.

RINGF: Restart ring buffer pointer - RING (beginning of the buffer).
TRSB, ARSB, SRSB, DGSB: leaders for throughput reports, anomalies reports, status reports and diagnostic reports - NCC Host address.
MGSB, HWSB: message generator, software watchdog statistics - Discard Fake Host on this IMP.
TLSB, TLTTCP: TENEX Logger process leader and internet header - BBN System E TENEX address.
DSPLOK: Display lock - 1.
TOLOCK: Slow Timeout lock - 1.
CLKLOK: lock on RTC reliability in Timeout - 1.
NF: number of free buffers (lock) - 1, increased as buffers are freed.
FREE, FREEND: free list head and tail - 1, then build initial free list.
LTQ: lock on Task queue - 1.
LOCKR: lock on Routing parameters - 1.
LOCKRO: lock on Routing output buffers - 1.
RUTOBF: list of Routing output buffers - 1.
TRNLOK: lock on free transaction blocks - 1.
RINGLK: lock on Restart process parameters - 1.
D2FL, F2DL: DDT to Fake Host buffer locks - 1.
T2FL, F2TL: Teletype to Fake Host buffer locks - 1.
DDTLOK: lock on DDT process - 1.
TTYLOK: lock on Teletype process - 1.
STQ: Task queue - empty.
SRUTQ: Routing input queue - empty.
SRQ: Task retry queue - empty.
SCKQ: diagnostic queue of bad buffers - empty.
SFHCQ: queue of incoming packet core messages - empty.
BASE: dispatches for Restart process - RSTGO; Block Transfer Poll - BLTCAL; DDT Poll - JJDDT; Configuration - CON; Task - TSK; Routing - ROUTE; Display - JDSPLY; Back Host Timeout - BTC; Teletype Poll - JJTTY; Host Timeout - IHTC; Slow Timeout - TOSS; the "last" (simnulated) PID - NOPIDS; and each empty PID - EMTY.
REASLK: lock on each reassembly block - unlocked.
REASST: reassembly block state - 1.
RMLOCK, TMLOCK: lock on each RM, TM block - unlocked.
RMIMP, TMIMP: RM, TM block remote IMP - -1.
PKCLIM: destination IMP number for reload/dump in packet core - 0.
FAKEIO: table of Fake Host "devices" - 0.
STATDT: table of destinations for message generator torture test - assorted destinations.

INTIME: initialization flag on fake code page - 3 (to stop future initializations).
In addition, the following are also referenced:
MAPVAR, MAPV2: map settings for variables, second variables pages.
MAPB1-6: map settings for 6 all-buffers pages.
PROCBT: my processor bit.
SNAP: local memory snapshot area.
STAIOB: statistics Fake parameter block.
SNAPBF: common memory snapshot buffer.
MAXBUF: 2 times total number of buffers in system.
MAPREL: map for reliability code page.
MINE: my IMP number.

I/O Performed

None.

5.12.1 Buffer Initialization

Function

Set up buffer pointers on the six buffers pages and on the leftover room on the second variables page.

Control Structure

BUFINI is called on initialization to divide the buffers pages into buffer sized blocks, storing packed buffer addresses into the table POINT. First the six all-buffers pages are divided and then an attempt is made to fit buffers between the tables POINT, CHAIN, WHERE, FLUSHD and CHAN.

Entry Points

Entered at BUFINI.

External Calls

DOBUFS to divide a page into buffers and set up the POINT words.

Initialization

The buffers pages have been allocated (in Stage MM) by placing the maps for those pages (or -1 if no room is available for that buffer page) into the map table.

Cleanup

None.

Data Structures

Local Data

None.

Shared Data

MAPB1, MAPB2, MAPB3, MAPB4, MAPB5, MAPB6: map settings for up to six all-buffers pages.
MAPV2: map for second variables page.
POINT: the table of packed addresses for each buffer.
MAXNF: the count of buffers.
MAXBUF: the length of the buffers tables.
JUNK: the last buffer.
BUFLEN: the length of a buffer.

I/O Performed

None.

5.12.2 DDT Page Initialization

Function

Initializes dispatch vectors, locks and variables for DDT.

Control Structure

DDT initialization is entered via a subroutine call, and checks the page reinitialization word. If the word is non-zero, nothing happens and a return proceeds. If the word is zero, a subroutine call to the skip return is used to allow the calling routine to participate in the consensus. If the consensus agrees, the calling routine returns to the initialization and initialization proceeds. Initialization consists of unlocking locks in a table and filling some locations with values found in a table. The return after initialization is directly to stage AR, where more page initializations are done.

Entry Points

DDTINC is the top level entry and checks the page initialization word.

External Calls

CNTLFS, a routine to set up DDT's dummy maps to their default values.
CHKPRO, to set up processor and page type masks.

Initialization

None.

Cleanup

None.

Data Structures

Local Data

None.

Shared Data

D2FL: DDT to fake Host lock.
F2DL: Fake Host to DDT lock.
T2FL: Teletype to fake Host lock.
F2TL: Fake Host to Teletype lock.
DDTA7: is initialized to print the DDT herald.
TOMODE: the type out mode flag, is set for hexadecimal.

TT3+<6*words>: the display dispatch, gets set so that the display is immediately refreshed.
INTIME: the page init word, is set non-zero so that initialization won't be called again.

I/O Performed

None.

5.13 Configuration

Function

The configuration code manages several functions related to the initialization of variables depending on real or simulated ("Fake") I/O devices. The configuration routines generally do some testing of devices and then set up tables as needed by the devices.

Control Structure

Configuration routines are driven by a central dispatch located in local memory. This dispatch uses the pointer to the timeout dispatch table on every common memory code page (TOPNTR) to locate the configuration routine and transfers there. Configuration then proceeds until either completion or a strip break is reached. DDT, LOCAL and WARM pages have no configuration routines on them. Reliablilty page routines test modems, test real Hosts, test dispatch tables, look for and test devices, check VDH modems if any, check and test PID levels for the devices, and build parameter blocks for the devices if necessary. FAKE page routines make sure Fake Hosts are operational, and set up clocks.

Entry Points

CON, the central dispatch mechanism. Coroutine control returns via CONA7.

External Calls

CSLEEP to dismiss coroutine at a strip break.
MTEST to test a modem.
HOTEST to test a Host.
PIDTST to see if a PID level is in use.
BLDMOD to build a modem or host parameter block.
GOFH to initialize a Fake Host.
CONCLK to set up the clocks.
RSTART to cause Configuration to dismiss until PID reaches low priority.

Initialization

CONA7 is initialized to dispatch to CONINI.

Cleanup

Parameter blocks are deallocated when the devices they correspond to disappear. Blocks are reinitialized when reliability checks fail.

Data Structures

Local Data

CONA1: a temp to store register 1.
CONA7: coroutine dispatch.
CONLOK: lock on Configuration process.
CONPNT: index in LMAP of present code page to use.
CONFTI, CONFTB: temporary storage for index values.
FAKCT3: temporary index storage over subroutines.

Shared Data

LMAP: table of map settings for each common memory page.
TOPNTR: pointer to timeout table (containing configuration routine dispatch) on each code page.
CONFLG: flag to force Configuration to complete before Timeout will run.
M2PBLK, H2PBLK: tables of modem, Host parameter blocks.
BASE: central dispatch table by PID level.
MBLKS: table of parameter blocks, parallel to BASE.
USEBUS: bit-coded word indicating which common busses exist.
MINPID, MOTPID: modem input, output PID levels.
USEIO: bit-coded table of which I/O interfaces exist.
IOBLOC, ALTIO: main, alternate I/O interface addresses for a modem or Host.
HINPID, HOTPID: Host to IMP and IMP to Host PID levels.
MAPREL: map setting for reliability page.
FAKEIO: simulated extension of USEIO (device table) for Fake Host "hardware".
MAPV2: map setting for second variables page.

I/O Performed

Device interfaces are reset when (1) they are first allocated, (2) they are reinitialized, and (3) they are removed from use.

5.14 Miscellaneous Routines

5.14.1 Packet Core Reload

Function

Implements the Packet Core protocol, which is used to dump or reload failed IMPs, and is used for debugging and network control functions in general.

Control Structure

Runs as a special part of the Block Transfer Process (see above). Controlled in part by Block Transfer parameters.

Entry Poinhts

BLTRLD is the entry point from Block Transfer.

External Calls

PKCIC to process a received packet core message.
PKCOC to check if a packet core message should be sent.

Initialization

None.

Cleanup

The Packet Core Process retries periodically if it is awaiting a reload. Each modem interface is used in sequence, with several tries on each interface.

Data Structures

Local Data

BLTRCT: count of tries remaining for current device.
PKCMYI: My IMP number for packet core messages.
PKCFID: foreign message-id for setup messages.
PKCTYH: TYPH word for packet core messages.
PSDATA: first data word of packet core message; setup from core message.
PKCLHA: handling type and host destination for packet core message.
PKCTYP: packet core type.
PKCLEN: length of transfer in words.
PKCADD: packet core word address; 8000! bit indicates common memory, 4000! bit indicates logical page type.
PKCSSF: flag to force sending of a setup message.
RLDTIM: timeout on packet core process.
PKCLID: local message-id for packet core messages.

BLTRIB, BLTROB: buffers for packet core input, output.
PKCNIM: packet core neighbor IMP number.

Shared Data

RLDDEV: current device to try reload from.
RLDINI: flag to trigger reload parameters reinitialization.
LCLOCK: current Stage system clock.
BLTSTY, BLTDTY: Block Transfer source, destination types.
USEBUS: bit table of existing busses.
BLTSIZ: size in bytes of Block Transfer.
BLTADD: Block Transfer address; odd for common memory, 8000! bit for logical page type.
STIME: local copy of Stage system time.
CONSOL: address of this processor's operator console, if any.
MAPREL: map setting for reliability page.
BLTST: Block Transfer state.

I/O Performed

While a packet core transfer is in progress, the current address (PKCADD) is displayed in the operator console DATA lights so progress of the transfer may be monitored.

5.14.2 Block Transfer Polling Process

Function

Poll the Block Transfer (BLT) Process.

Control Structure

Set up the code map for block transfer (MAPREL) and check the state of block transfer to see if anything needs to be done. If nothing, exit; otherwise call block transfer. On return from block transfer, restore maps 1, 2, and 3. Call subroutine RSTART to reschedule this process to run when priority is low. An exit to the main loop completes the routine.

Entry Points

BLTCAL only.

External Calls

BLT, the block transfer subroutine.
RSTART, to place a pid into the ring buffer.

Initialization

None.

Cleanup

None.

Data Structures

Local Data

None.

Shared Data

MAPREL: code map for BLT.
MAPVAR: first variables map.
MAPV2: second variables map.

I/O Performed

None.

5.14.3 Restart Process

Function

Poke PIDs for processes that have gone to sleep until the PID priority is low (by calls to RSTART).

Control Structure

Checks count of PIDs in its ring buffer and, if non zero, picks up the next one and stores it into the PID.

Entry Points

RSTGO, dispatch is from main dispatch loop. Poked whenever RSTART is called, and by itself if still more entries are left in the ring buffer.

External Calls

None.

Initialization

The ring buffer is initialized to be empty.

Cleanup

None.

Data Structures

Local Data

None.

Shared Data

RINGLK: lock on the ring buffer.
RINGC: count of PIDs in the buffer.
RINGF: current ring buffer pointer.
RING: ring buffer.

I/O Performed

None.

5.14.4 DDT Polling Process

Function

Poll DDT process periodically while the IMP system is running.

Control Structure

Restores registers for the DDT process and resumes it at the last coroutine dismiss point.

Entry Points

JJDDT sets up the code map for the DDT page, and then proceeds to DDTWAK to resume the DDT process.

External Calls

DDTPOL to resume DDT process.

Initialization

None.

Cleanup

None.

Data Structures

Local Data

None.

Shared Data

IMPDDT: flag to show IMP is polling DDT.

I/O Performed

None.

5.14.5 Teletype Handler Polling Process

Function

Poll the teletype handler process.

Control Structure

Restores registers for the teletype process and resume from previous coroutine dismiss point.

Entry Points

JJTTY, which sets up the code map for DDT and jumps to TTYWAK.

External Calls

TTYPOL to poll the teletype handler process.
RSTART to cause this routine to be poked when the PID level reaches low priority.

Initialization

None.

Cleanup

If the teletype lock timer reaches 16, the lock is ignored and operations proceed as if it were unlocked upon entering.

Data Structures

Local Data

TTYLTO: teletype lock timer to prevent overly long lockings.

Shared Data

RSTRSW: restart switch, indicating teletype process needs to be polled again.
STIME: local copy of system time.
POLTIM: time at last polling.

I/O Performed

None.

5.14.6 Display Process

Function

1) Update information for bottom line of terminal.
2) Write out display information to teletype output buffer.

Control Structure

Using a table of words to display on the bottom line, the present value of the words are compared to the value now on the screen. A difference causes a flag to be set and the new value stored. When the results for all the entries are completed, a table of buffers containing display information (including the results just added) and the flag word are used to print those buffers that have changes. When all the buffers are done, sleep for about six seconds and run the process again.

Entry Points

JDSPLY, which sets up code map for DDT page and jumps to DISPLY which resumes the display handler coroutine.

External Calls

F2TPED places a character in the teletype output buffer if room, else sleeps until room is available.
DSPSLP is the coroutine dismiss (sleep) routine.

Initialization

None.

Cleanup

None.

Data Structures

Local Data

TT1: variable to save cursor location.
TT3: register save area.

Shared Data

DINIT: word containing flags for data buffers by display line.
MAPCOM, MAPREL, MAPDDT, MAPCOD, MAPFAK, MAPVAR, MAPV2, MAPV2+2, MAPB1, MAPB2, MAPREL+NVARSP, MAPDDT+NVARSP, MAPCOD+NVARSP, MAPFAK+NVARSP, MAPFAK+NVARSP+2: locations (map settings) to be displayed in bottom line of display.

CILLOC, CILLPR, CILLCT, W1: buffers of words to be displayed (first three are tables of trap numbers, processor masks, and counts).

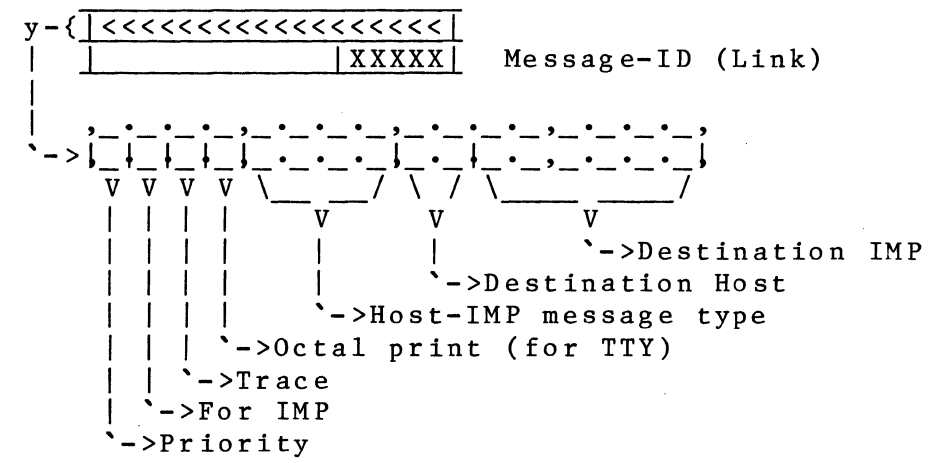
I/O Performed

None.

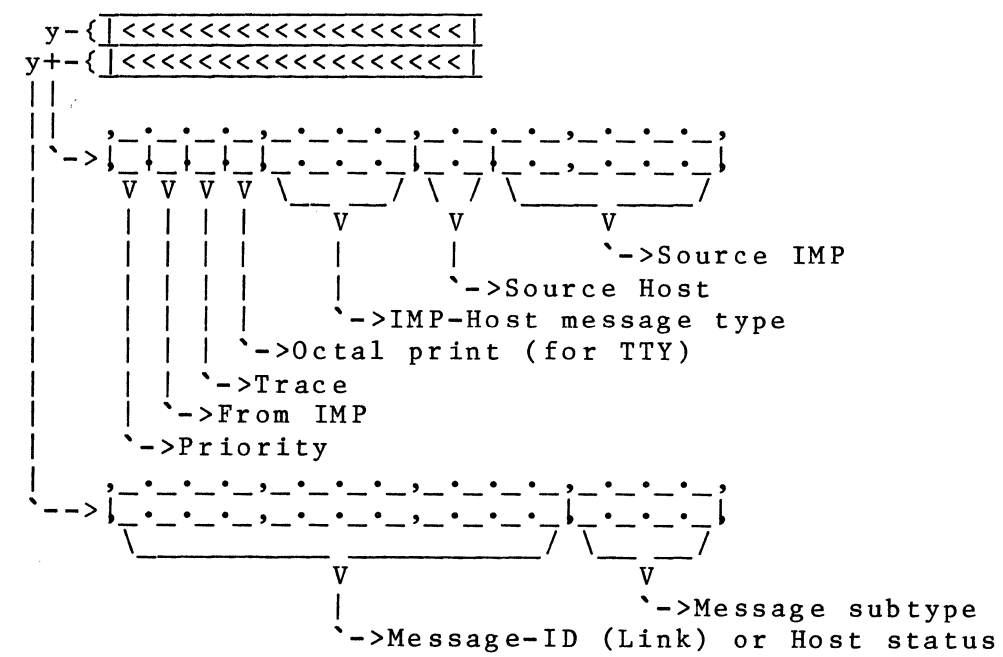
Chapter 6
Data Formats

6.1 Old-style Leader Format

Host to IMP

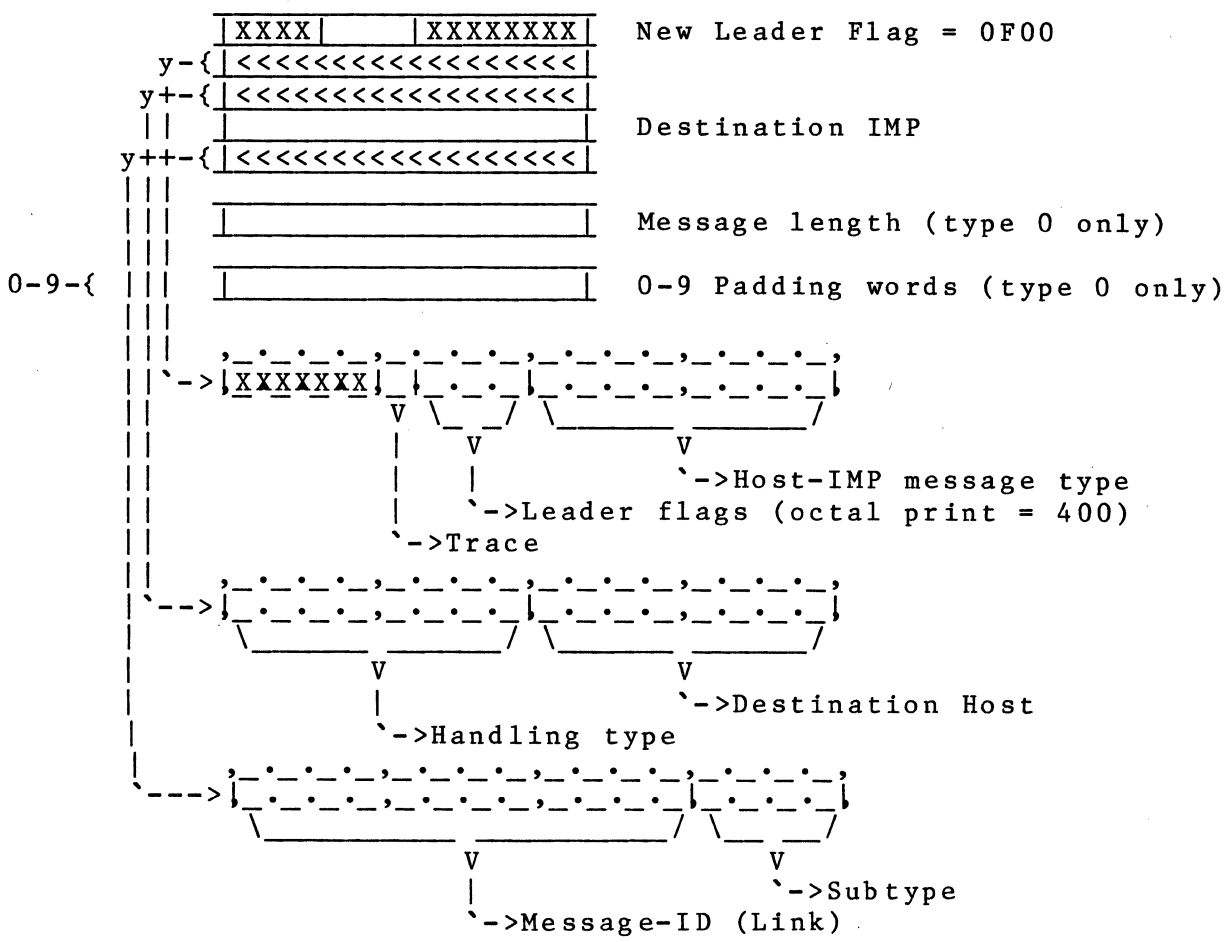


IMP to Host

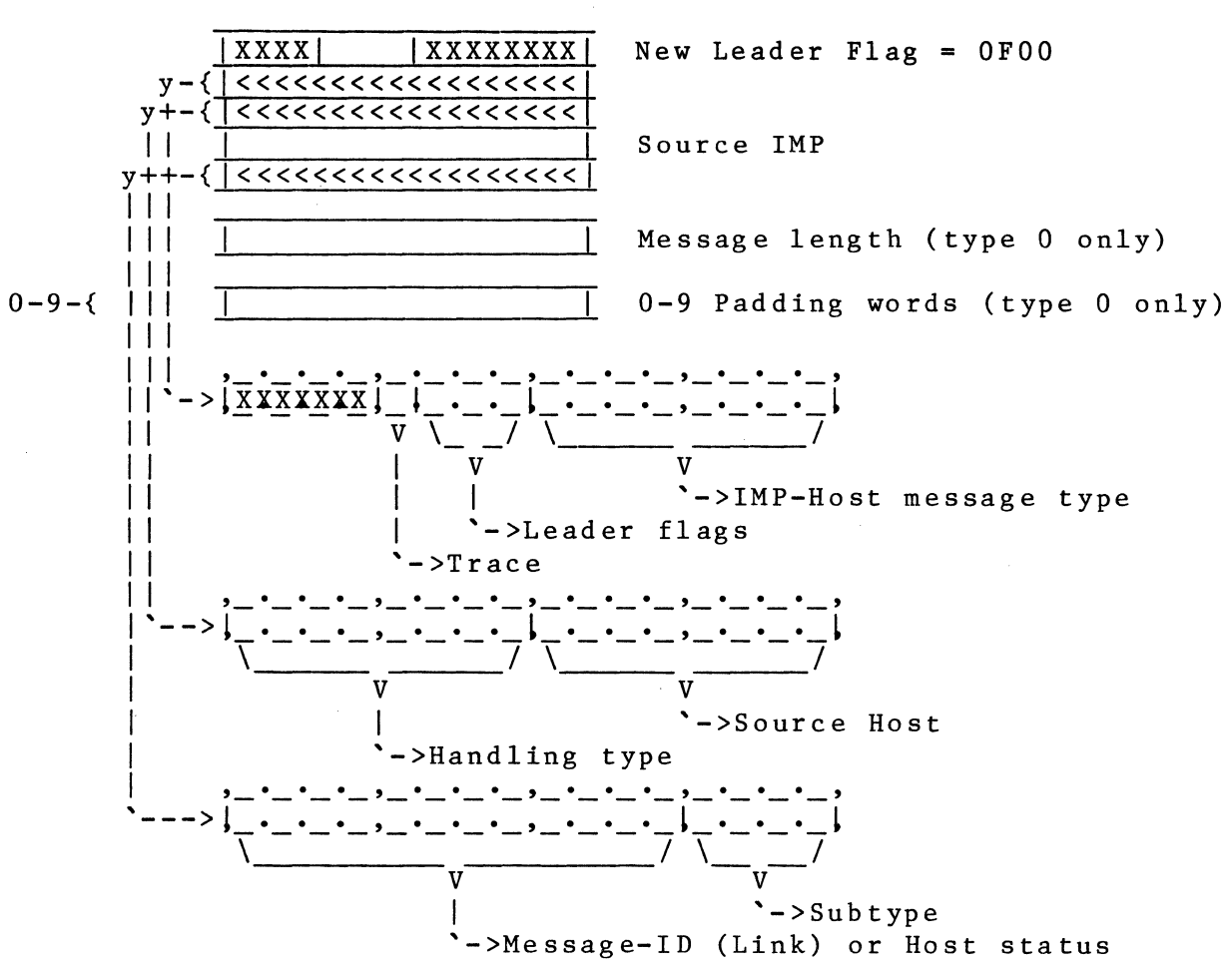


6.2 New-style Leader Format

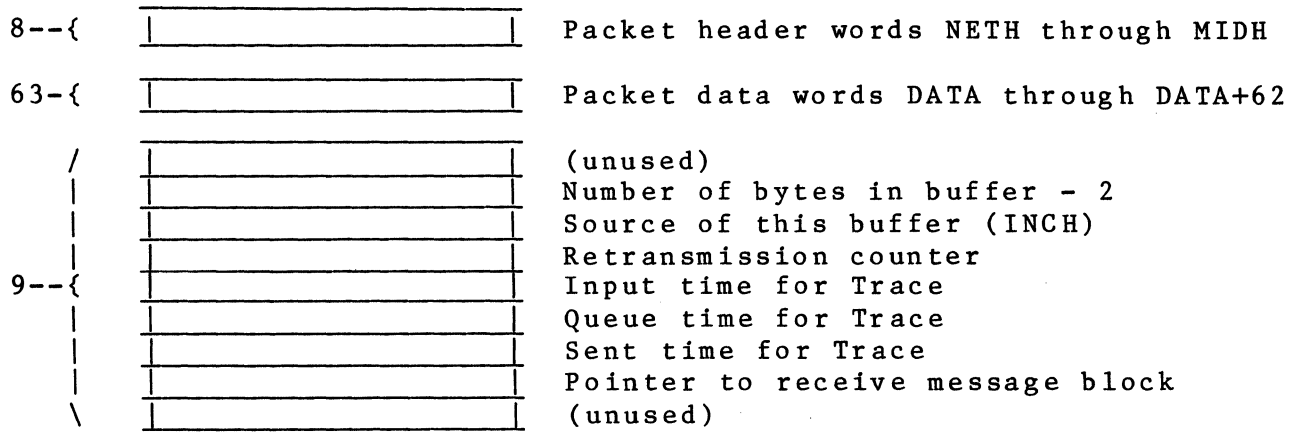
Host to IMP



IMP to Host

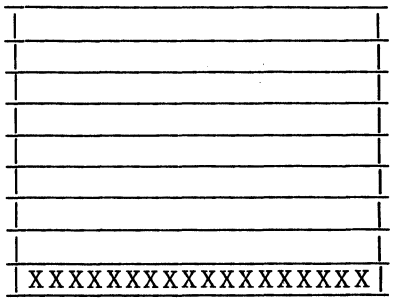

|XXXX| |XXXXXXXX| New Leader Flag = 0F00
y-{|<<<<<<<<<<<<<<<<<<<<|
y+-{|<<<<<<<<<<<<<<<<<<<<|
| | | Source IMP
y+-{|<<<<<<<<<<<<<<<<<<<<|
| | |
| | | Message length (type 0 only)
| | |
0-9-{ | | | 0-9 Padding words (type 0 only)
	`->	XXXXXXXX
	V _ _/ _____ _____/
		V V
		V `->IMP-Host message type
		`->Leader flags
	`->Trace
`-->
_____ _____/ _____ _____/
V V
	`->Source Host
`->Handling type
`--->|
_____ _____/ __ __/
V V
| `->Subtype
`->Message-ID (Link) or Host status

6.3 Buffer Format



6.4 Basic Packet Structure

Not all of the various messages have all of these fields, but if they do have them they will appear as diagrammed here.



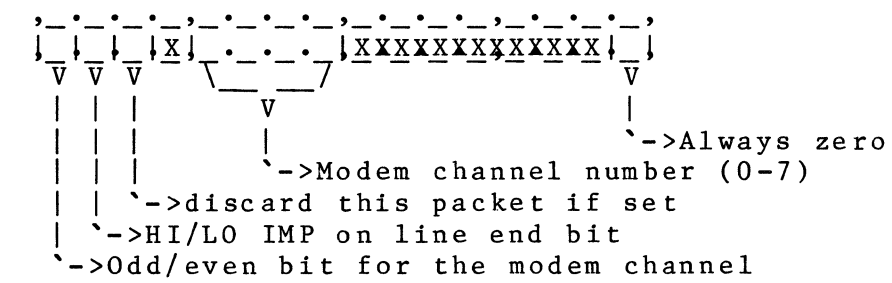
Every packet has SRCH, CHKH, and the fields Packet type and Compat.

Packet type is the high order two bits of TYPH.
The basic types are:

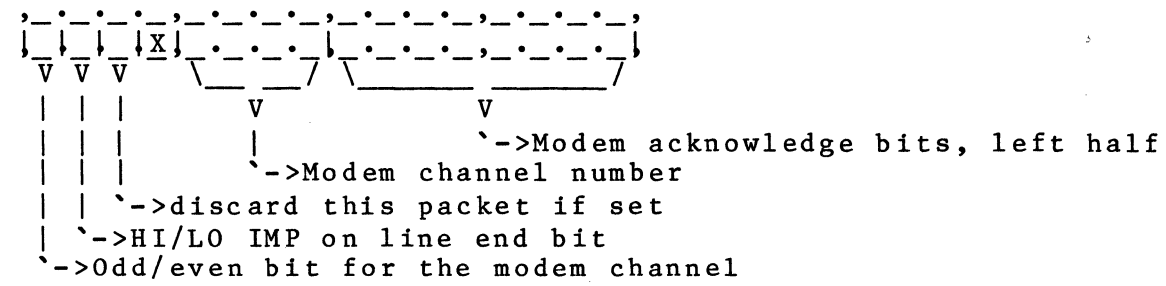
0 Data : Message, RFNM, etc
1 Network control: Get-a-block, Reset, etc
2 Switch control: Routing, Null, etc
3 Special: Packet core, Demand reload, etc

Compat is the next bit below packet type and is used as an odd/even bit within packet type so that an incompatible release can be propagated.

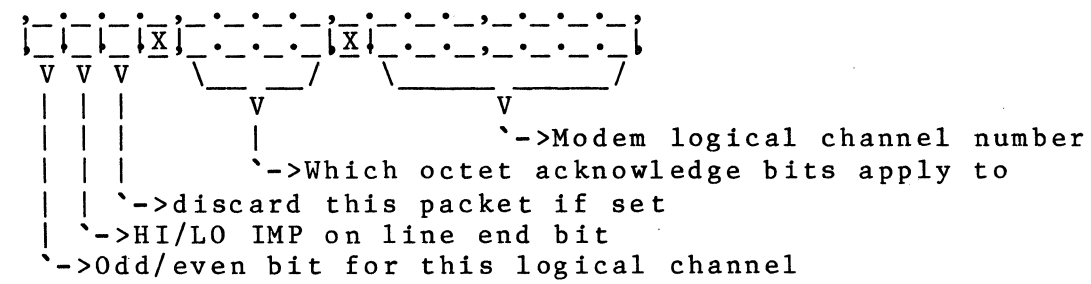
For packet types 0 and 1, and null packets (type 2), on an eight channel line, NETH has the following format:



On a sixteen logical channel line, NETH is structured as follows (packet types 0 and 1):



On a 32 to 128 logical channel line, NETH is structured as follows (packet types 0 and 1):



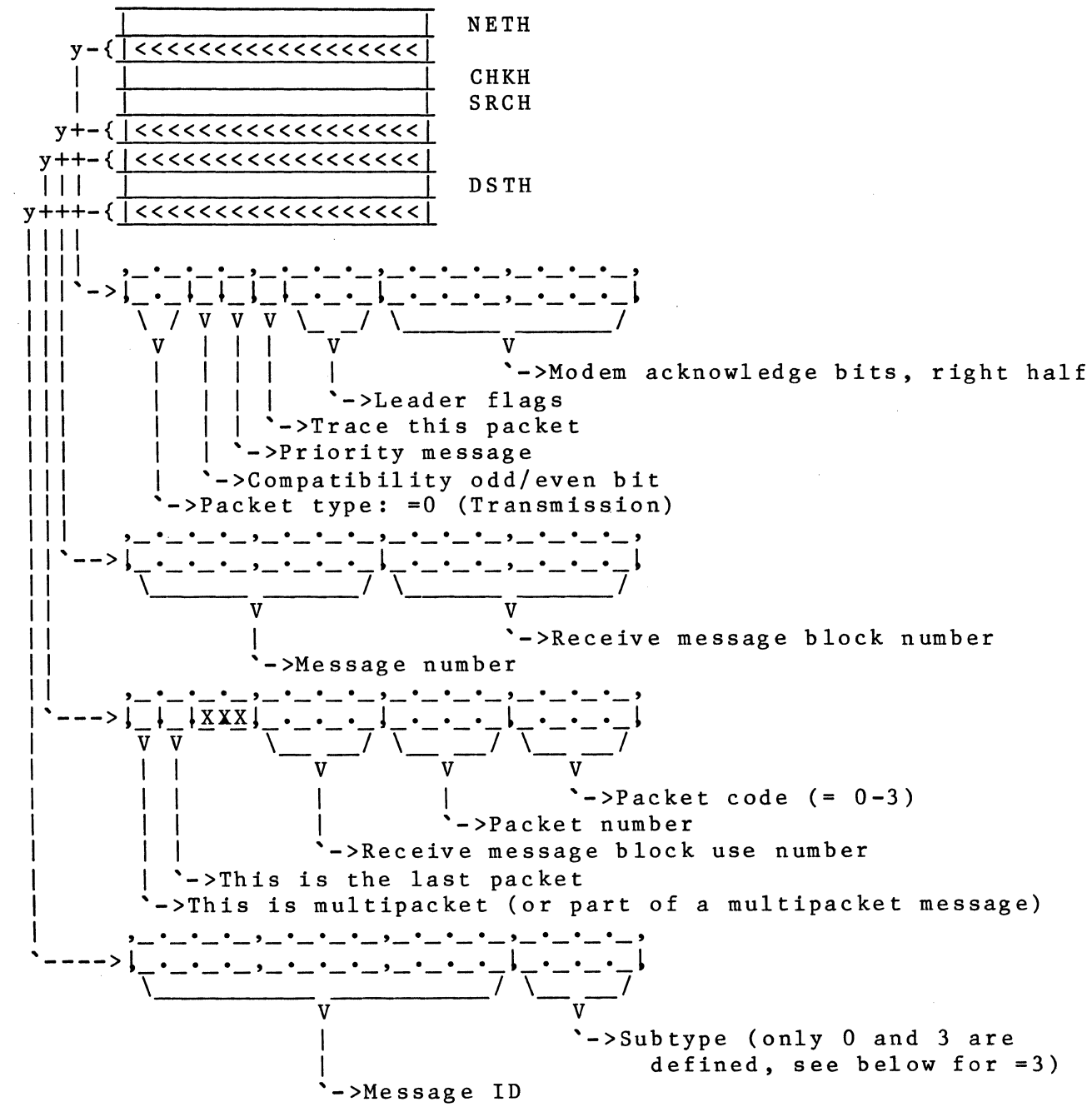
6.4.1 Packet Type 0 Formats

Packet type 0 is for messages concerned with the actual transmision of data

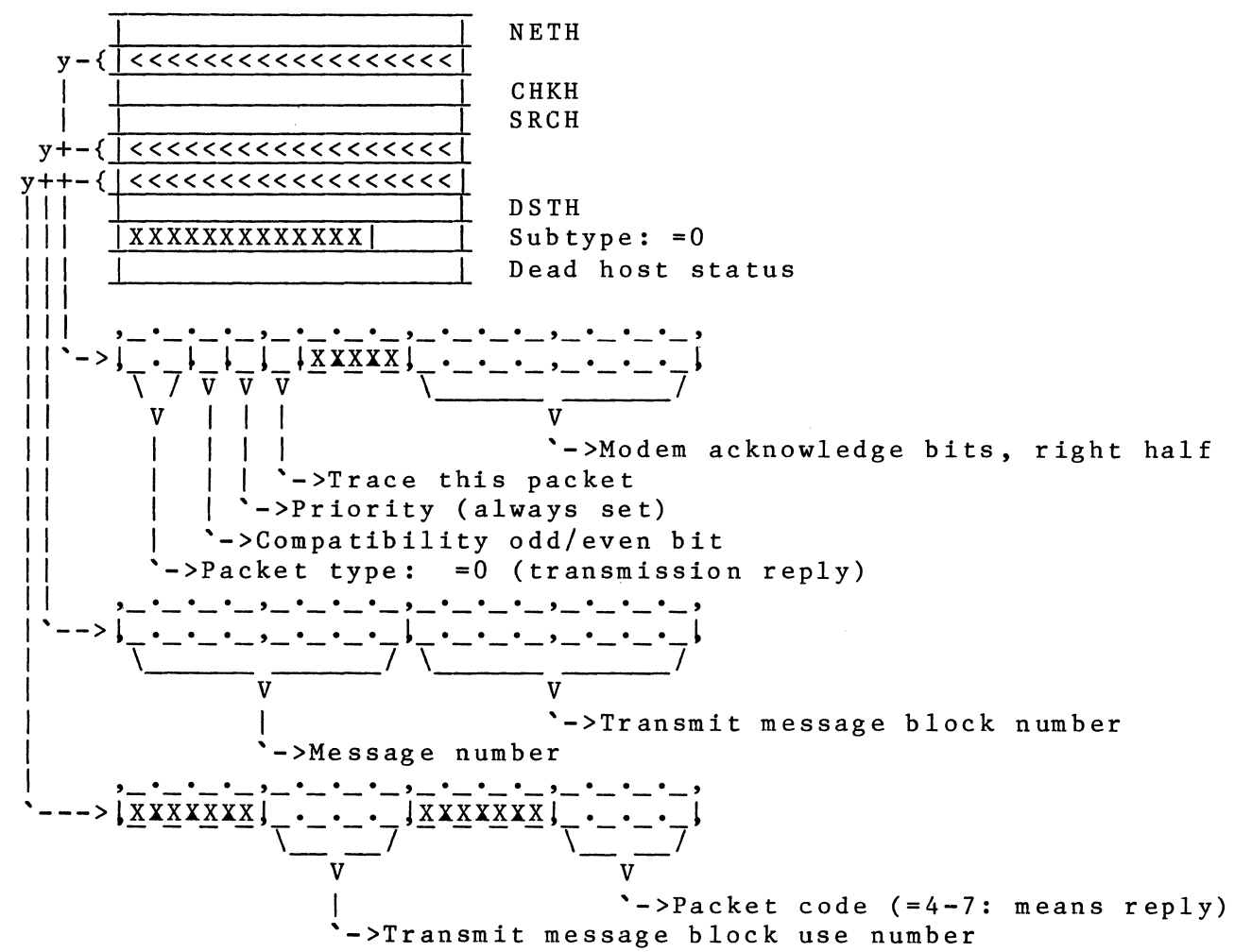
The packet codes are:

0 Message
1 Request (for 1 or 8 depending on multipacket bit)
2 Giveback (Multipacket bit always on)
3 Incomplete message
4 RFNM
5 RFNM w/allocate
6 Destination was dead
7 Incomplete reply

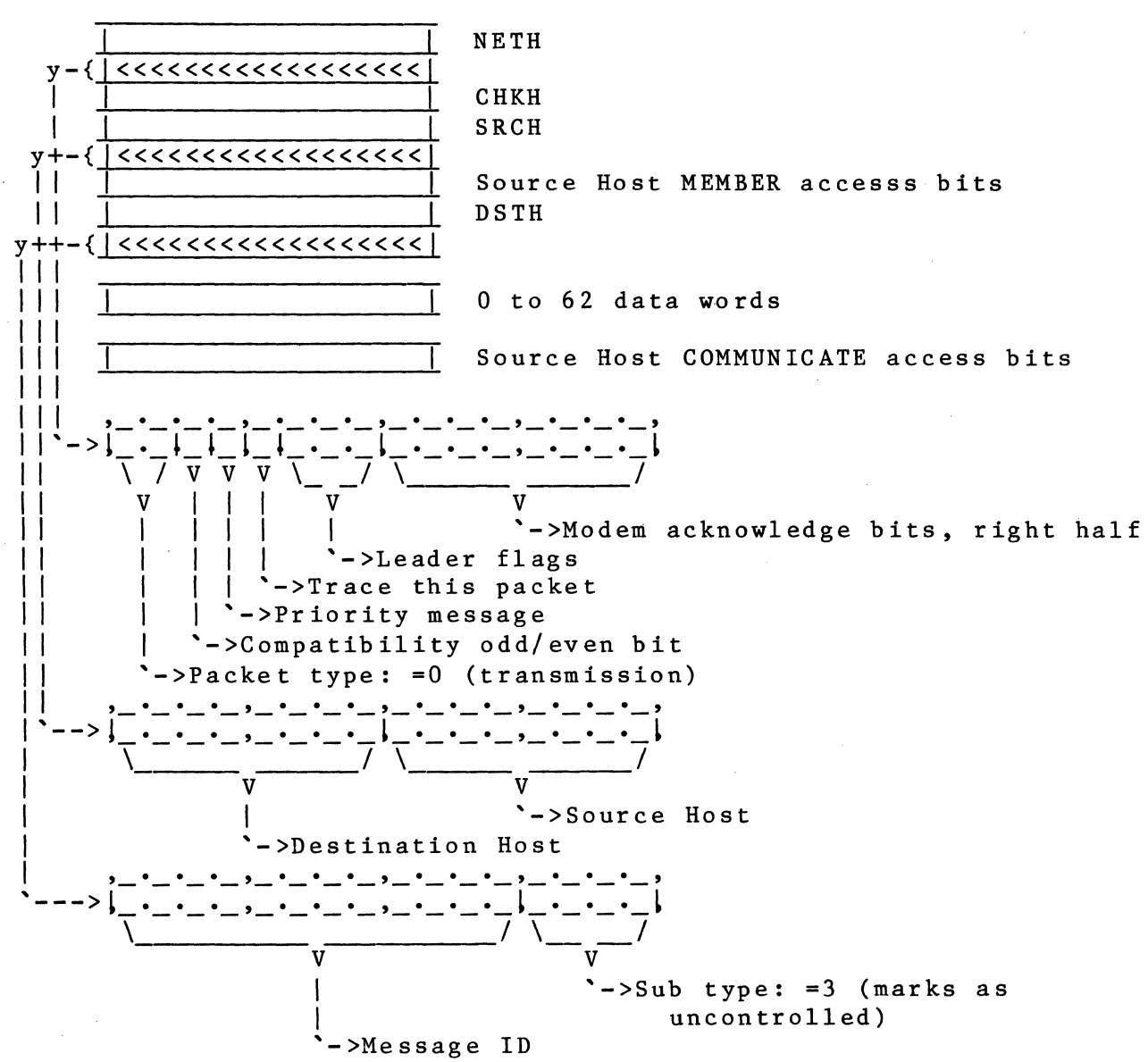
6.4.1.1 Packet format for Type 0, Codes 0-3



6.4.1.2 Type 0, codes 4-7



6.4.1.3 Type 0, subtype 3: Uncontrolled packet

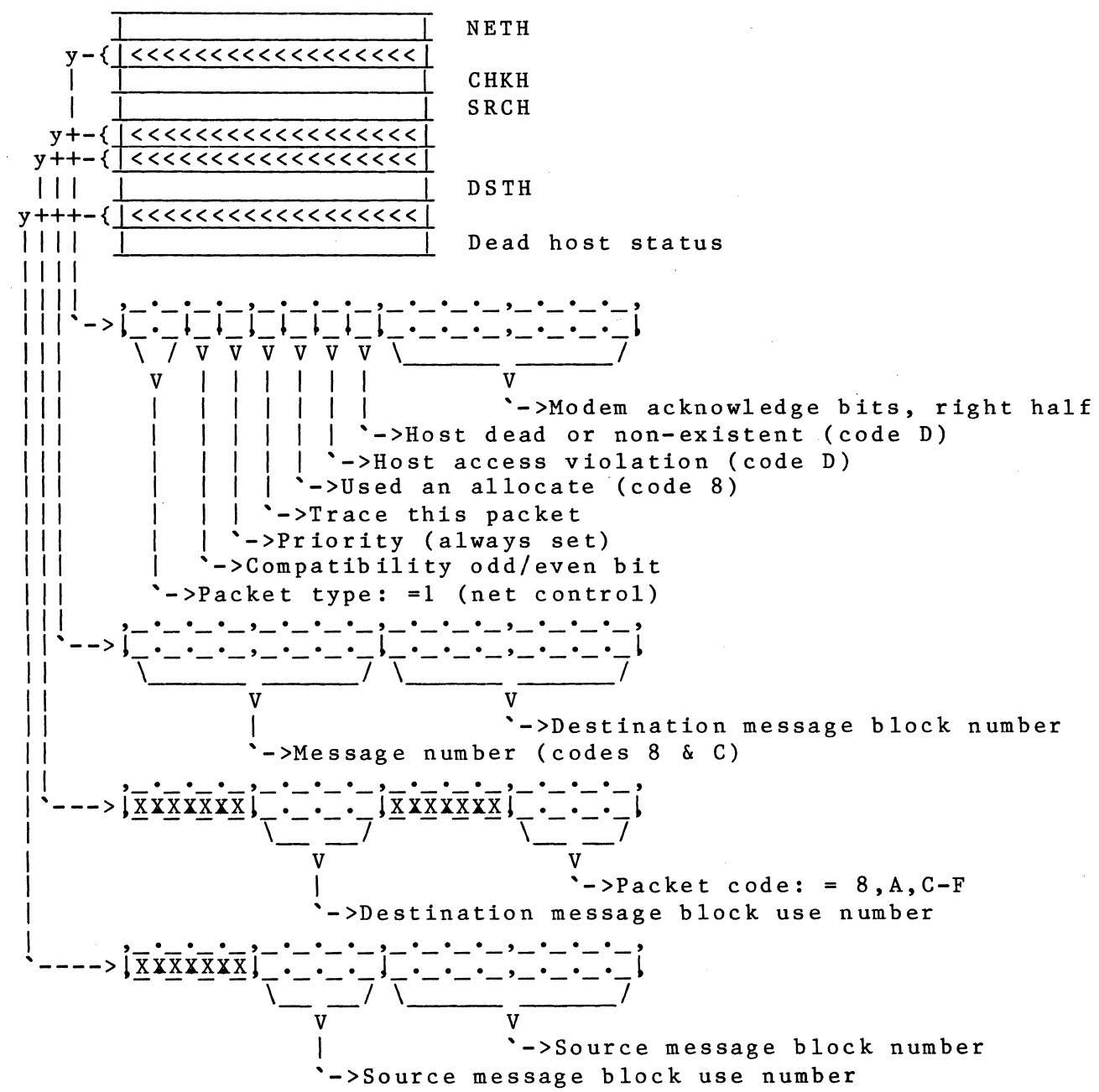


6.4.2 Type 1 Packet Formats

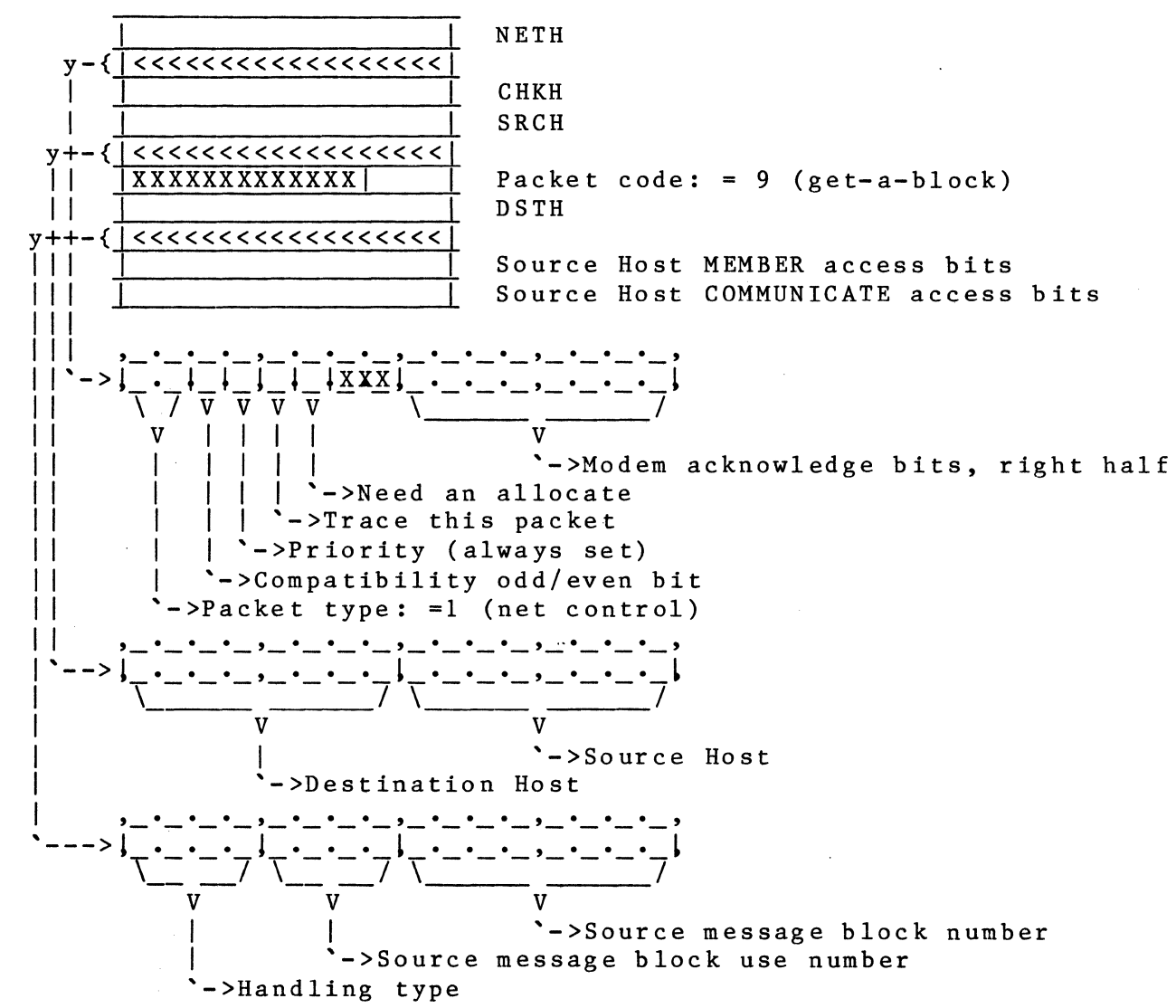
The type 1 packets use the following codes:

8 Incomplete query
9 Get-a-block
A Reset block
B .. unused
C Out of range
D Got-a-block or Got-no-block (i.e., reply to a get-a-block)
E Reset block request
F Reset block reply

6.4.2.1 Packet type 1, codes 8, A, C, D, E and F

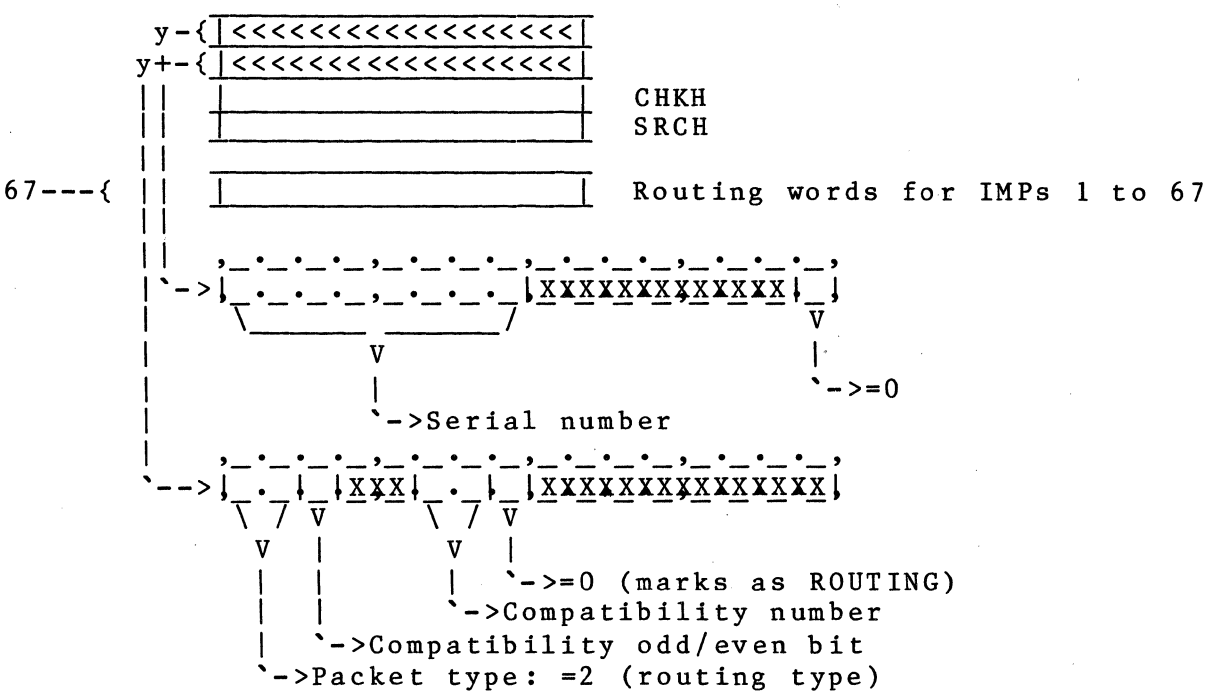


6.4.2.2 Packet type 1, code 9

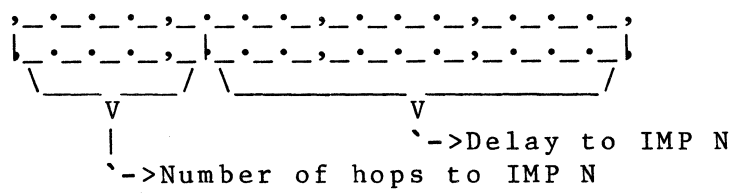


6.4.3 Packet type 2: Routing and null

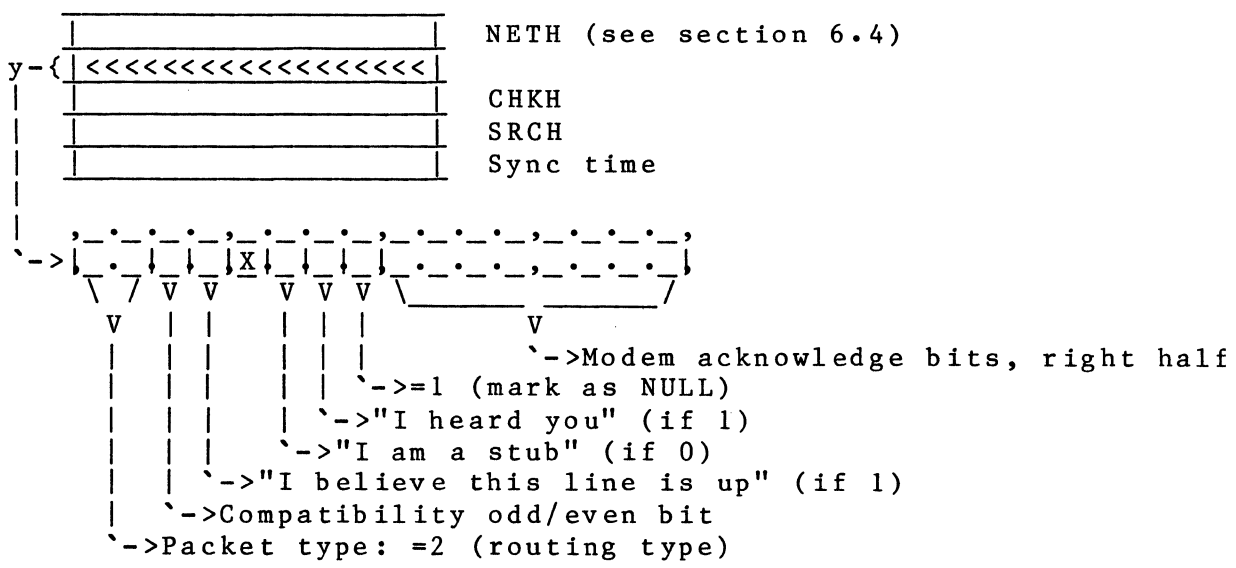
Routing packet



Format for word N of routing data in above message



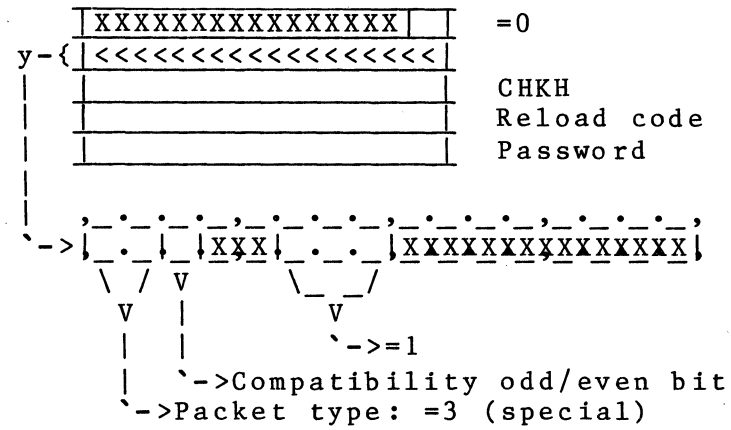
Null packet


NETH (see section 6.4)
CHKH
SRCH
Sync time
X
V
`->Modem acknowledge bits, right half
`->=1 (mark as NULL)
`->"I heard you" (if 1)
`->"I am a stub" (if 0)
`->"I believe this line is up" (if 1)
`->Compatibility odd/even bit
`->Packet type: =2 (routing type)

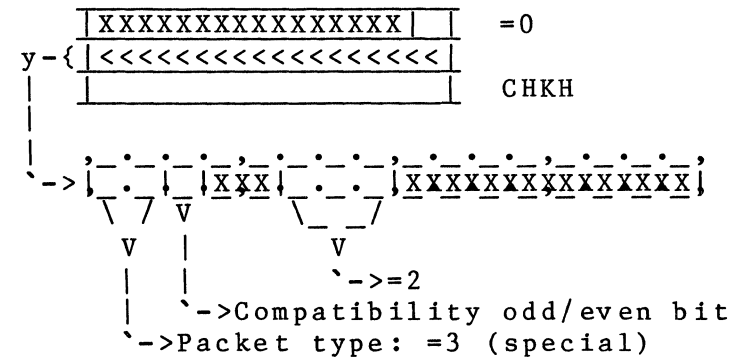
6.4.4 Packet Type 3

Type 3 packets include Demand reload, Reload request and Packet core

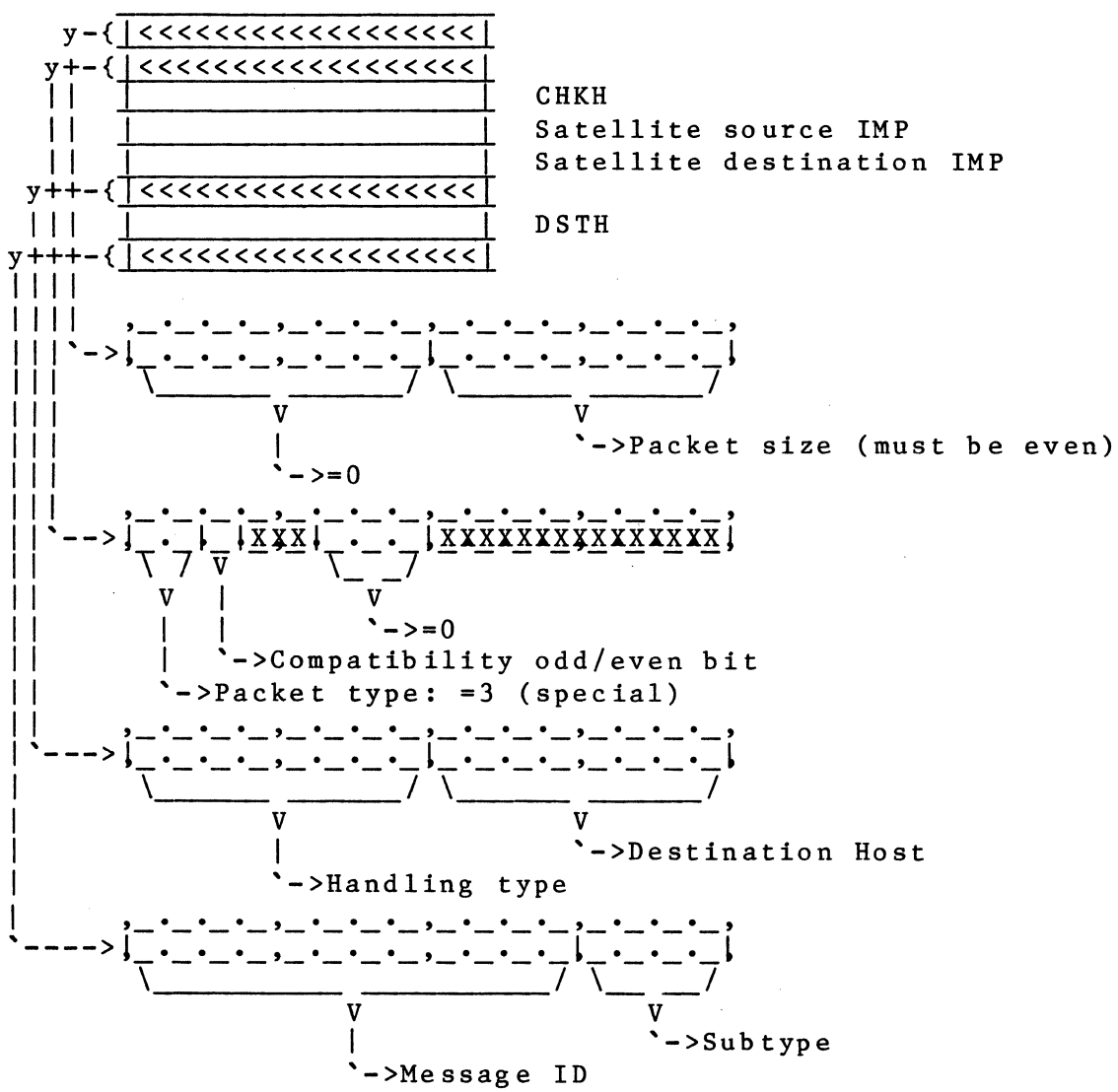
6.4.4.1 Demand reload



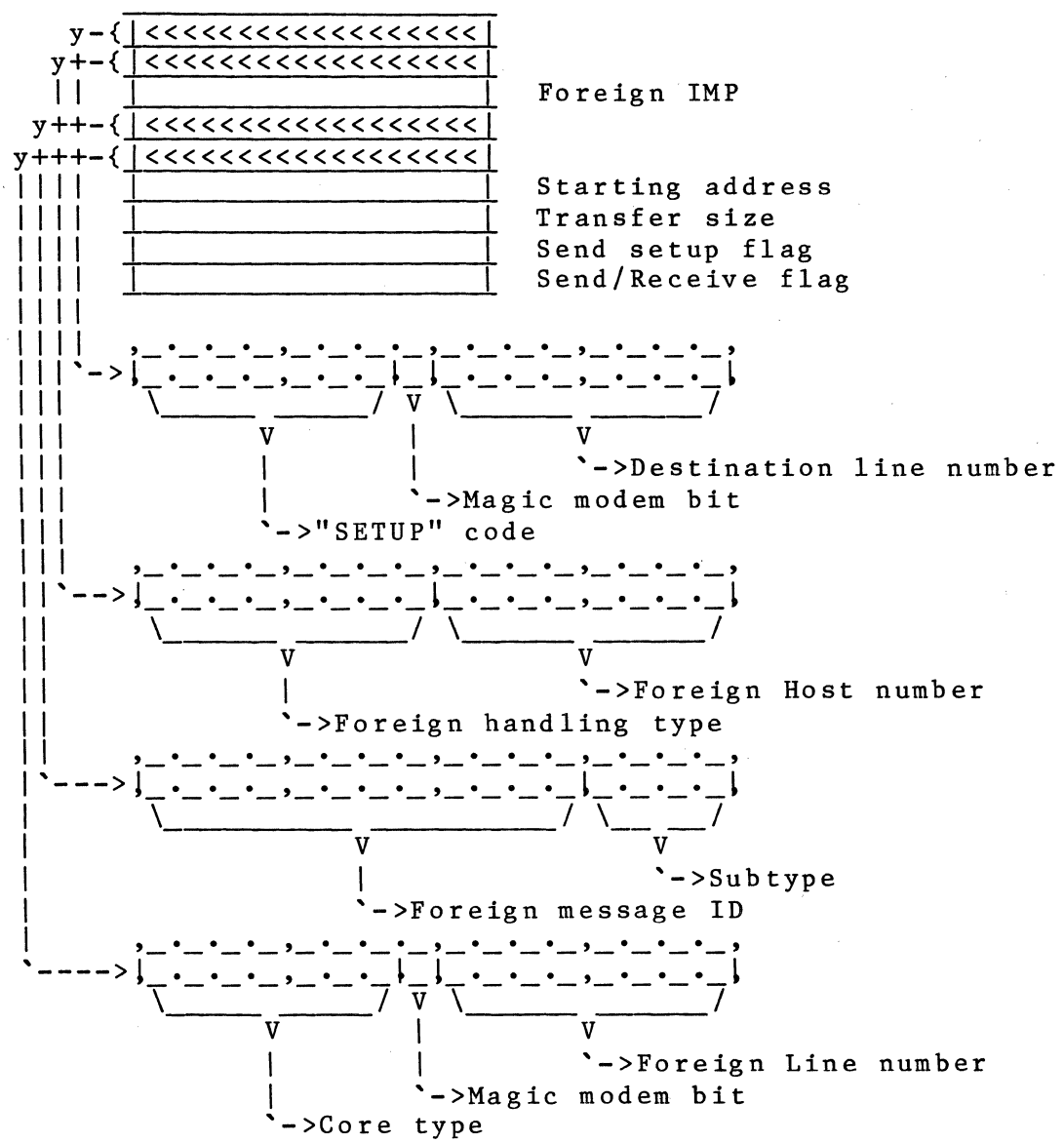
6.4.4.2 Reload request



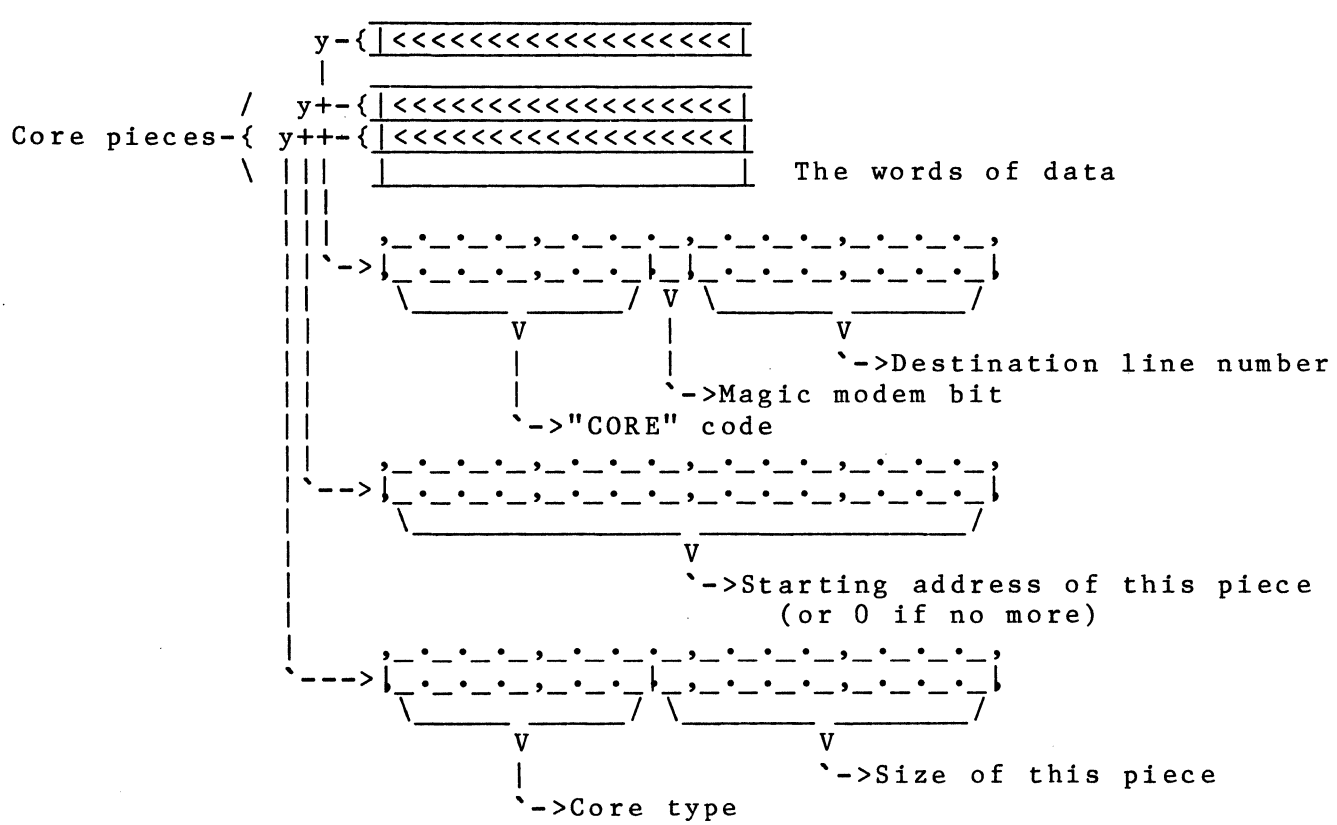
6.4.4.3 Packet core



6.4.4.3.1 Data for SETUP message



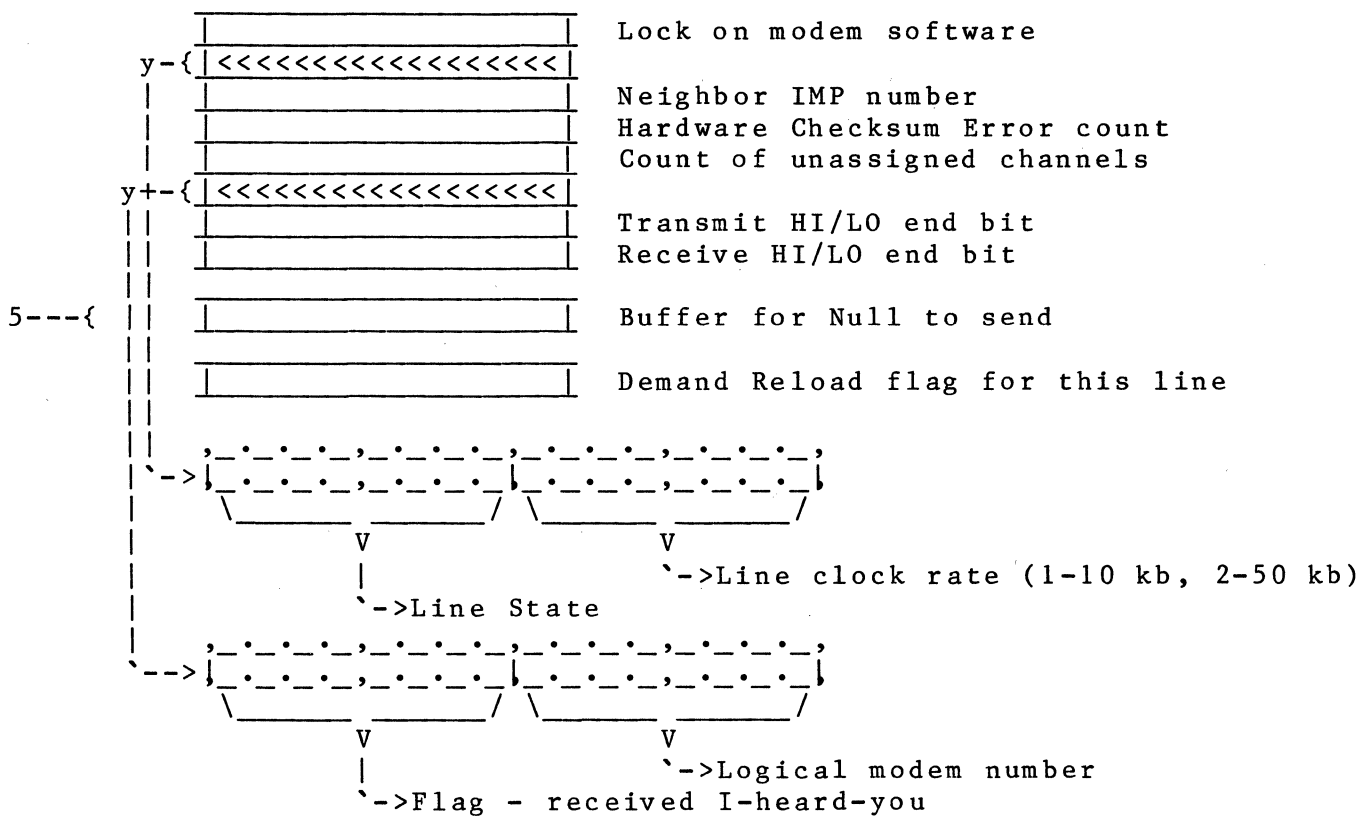
6.4.4.3.2 Data for CORE message



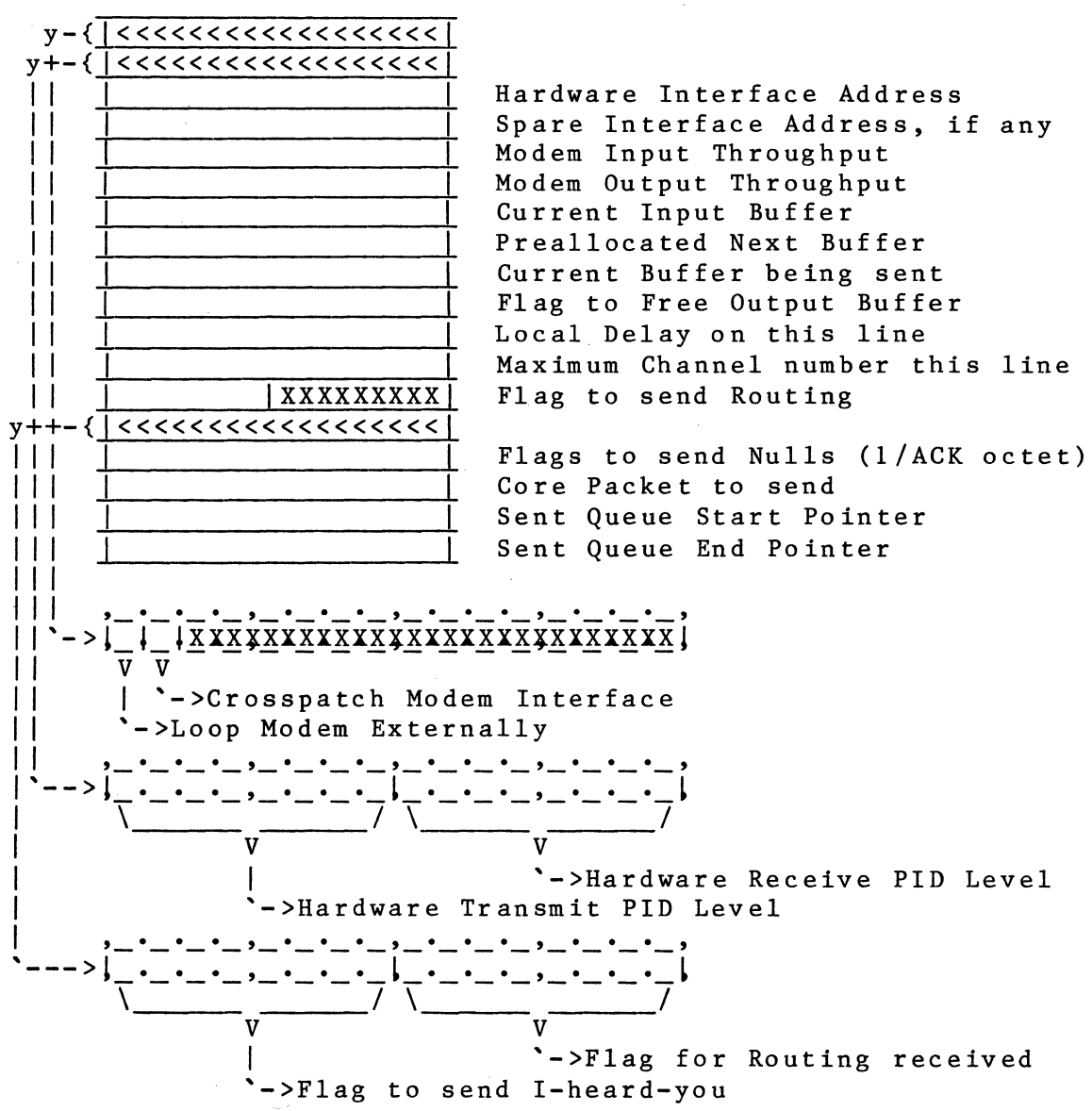
6.5 Modem Parameter Blocks

One block per modem.
Length is 56 to 80 words, depending on number of logical channels.

First 14 words (byte offsets 0-1B!):

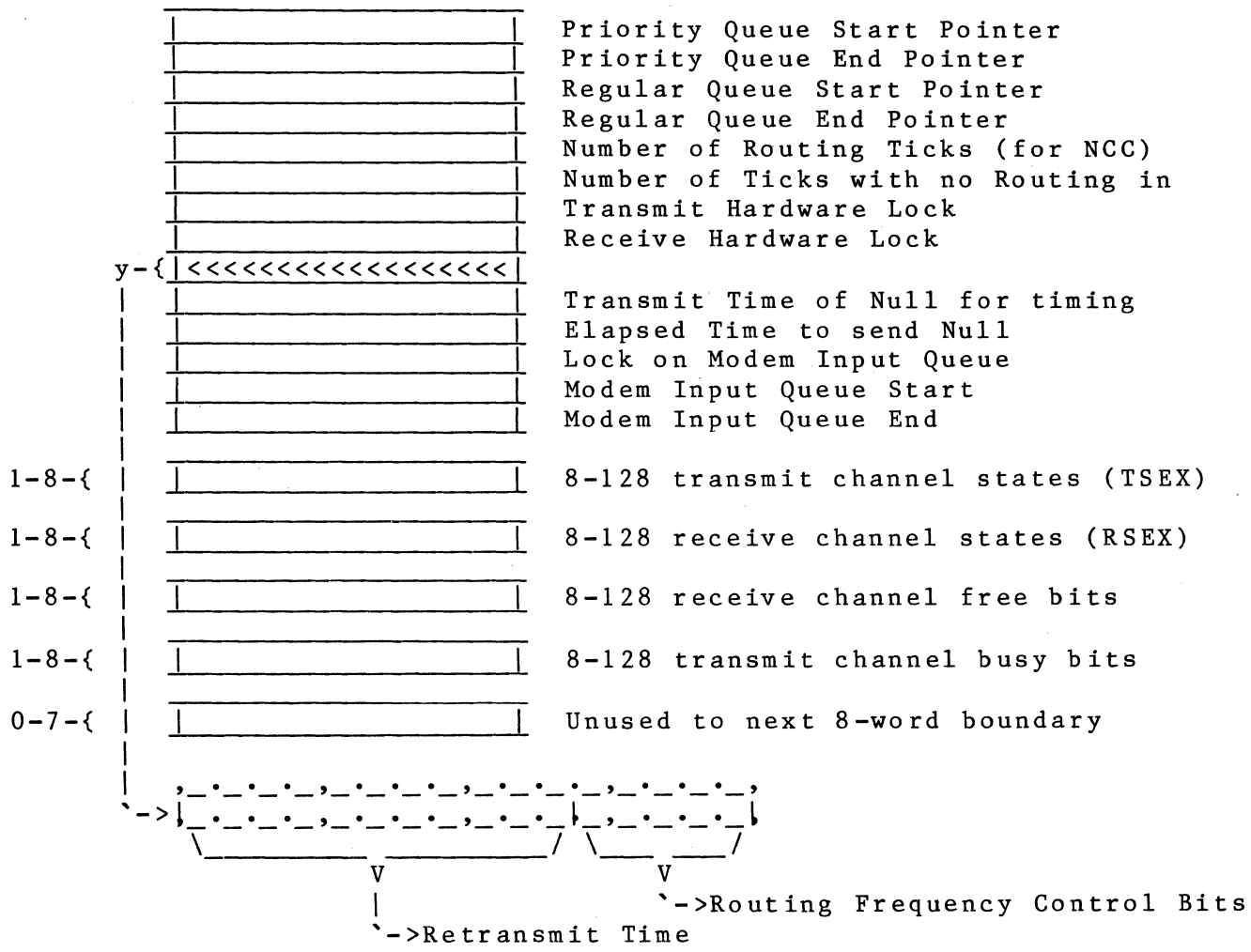


Modem Parameter Blocks (cont.) - words 15-32 (bytes 1C-3F!)



Modem Parameter Blocks (cont.) - last 24 to 48 words (bytes 40-9F!)

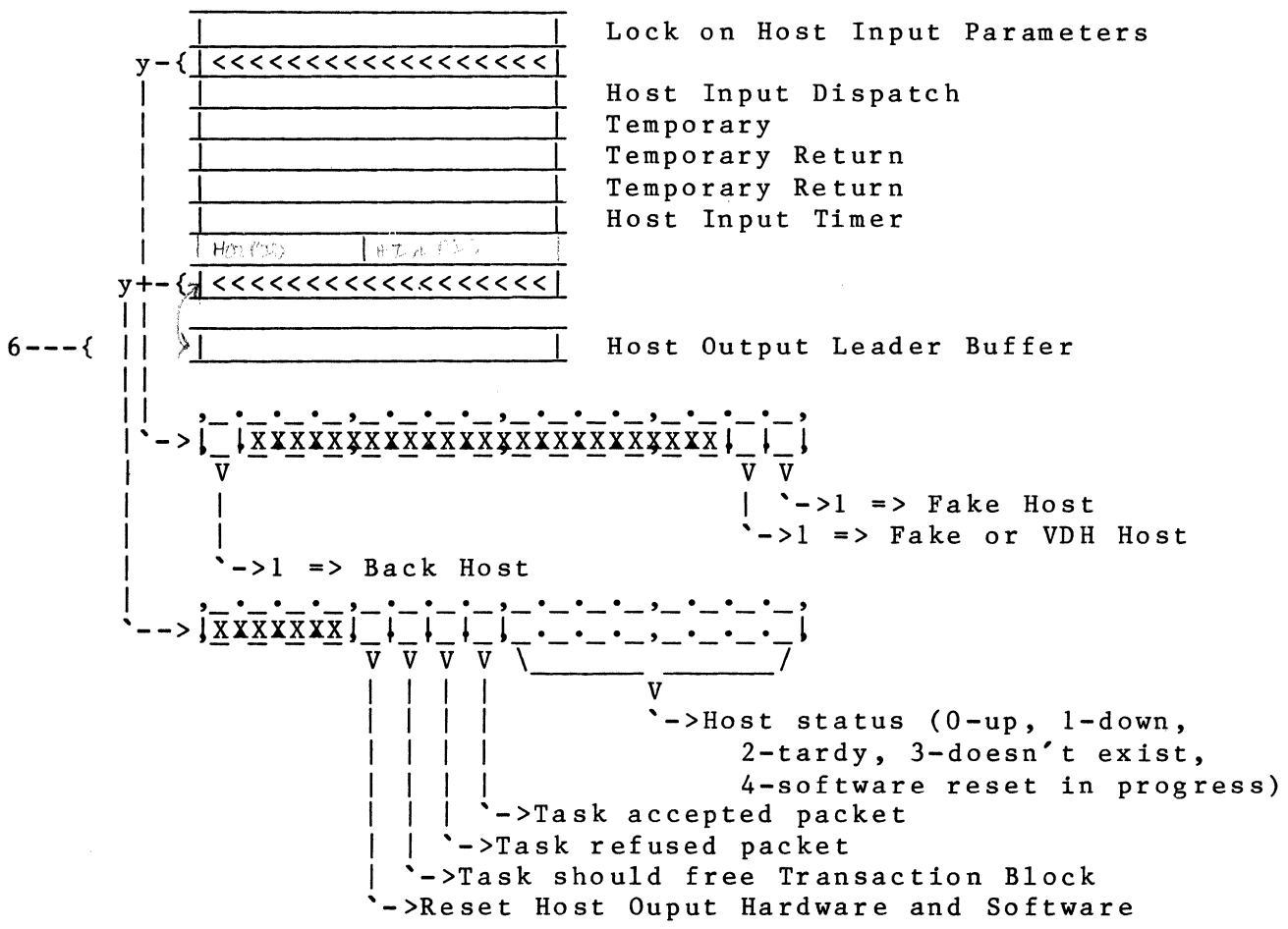
(length = 56 for 8 or 16 channels, up to 80 for 128 channels)



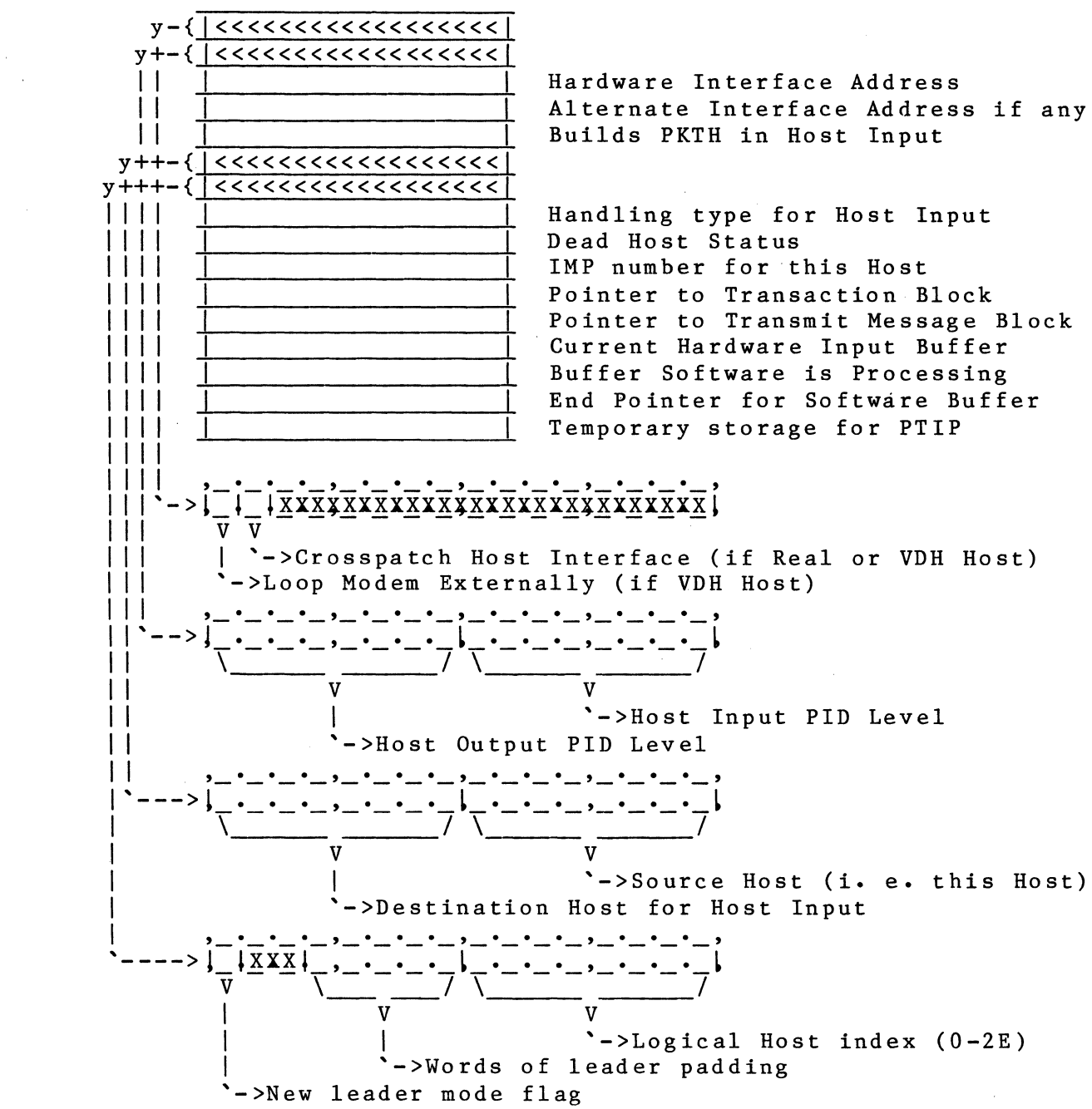
6.6 Host Parameter Blocks

One 56-word block for each Real, Fake, or Very Distant Host
First 7 words match Back Host Parameter Block (see below)

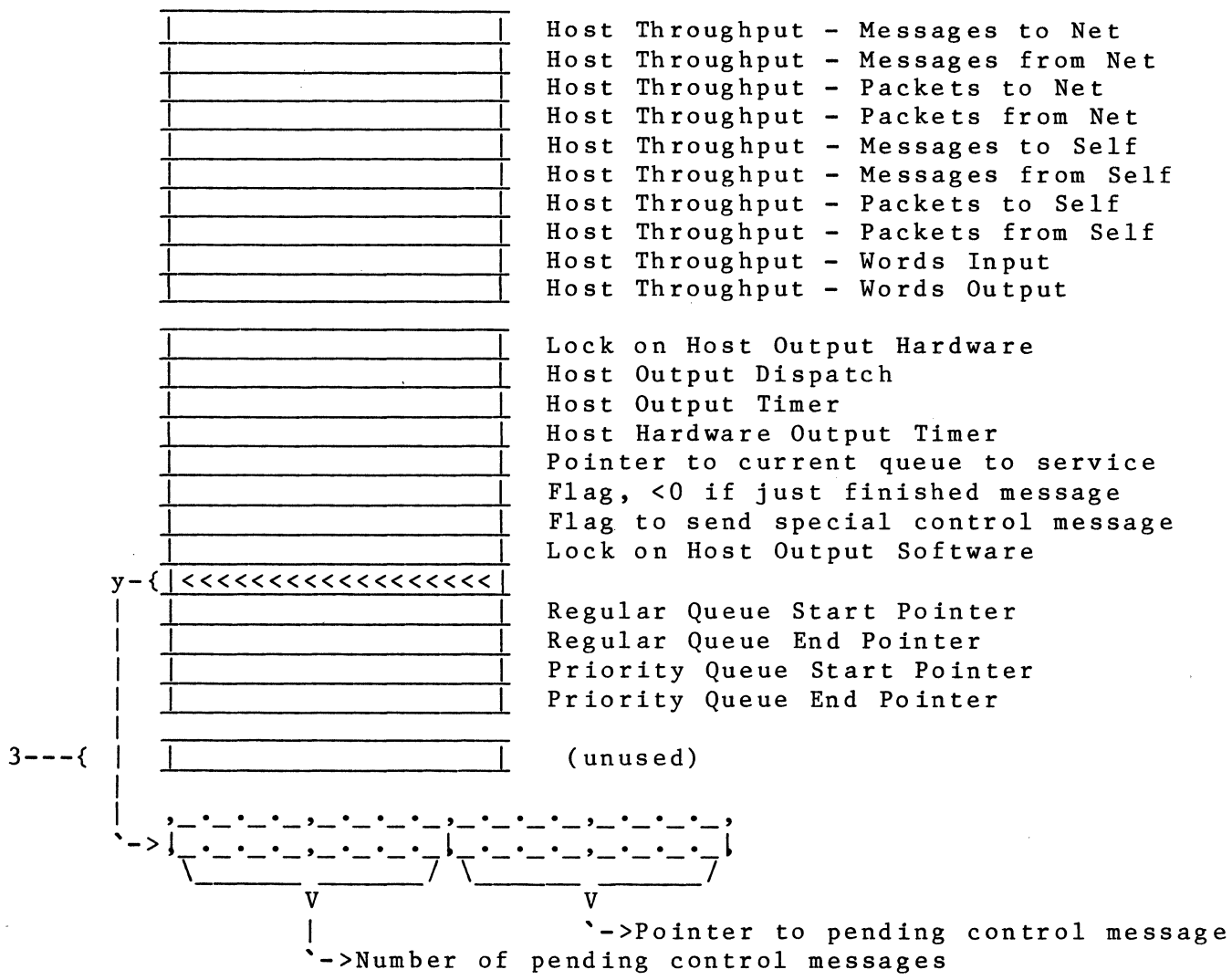
First 14 words (bytes 0-1B!):



Host Parameter Blocks (cont.) - words 15-30 (bytes 1C-3B!)



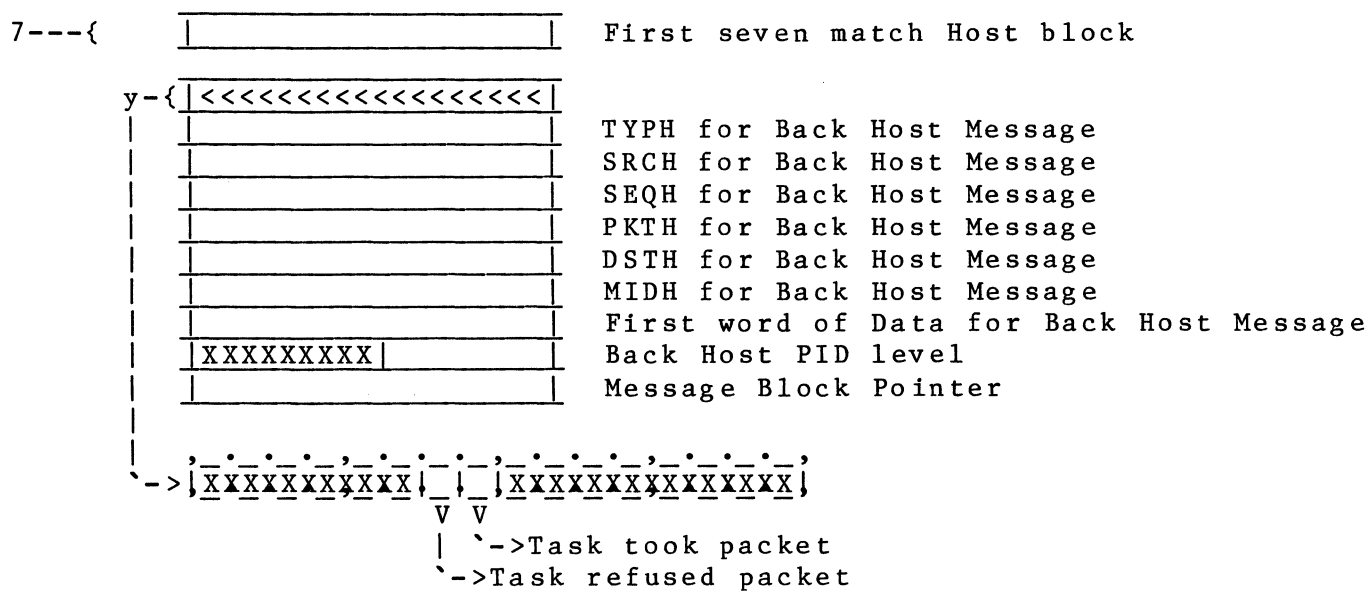
Host Parameter Blocks (cont.) - words 31-56 (bytes 3C-6F!)



6.7 Back Host Parameter Blocks

First 7 words match Host Parameter Block (above)

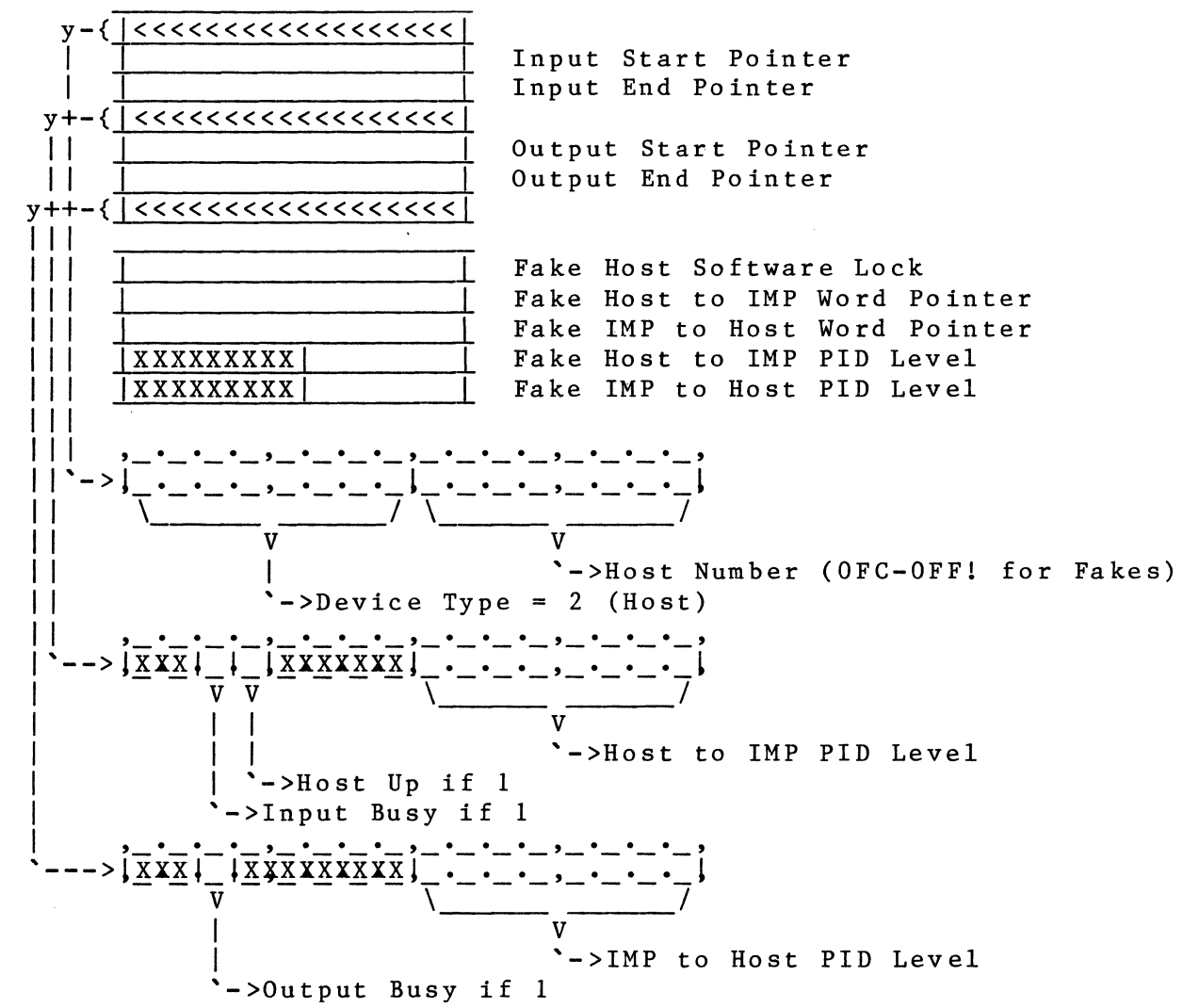
One 17-word block per Back Host



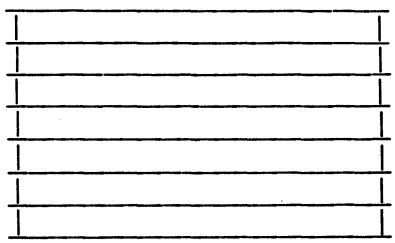
6.8 Fake Host Parameter Blocks

One 19-word block for each Fake Host.
The first seven words simulate a Pluribus Host Interface.
VDH Parameter Block is identical for first 12 words.

Words 1-12 (bytes 0-17!):



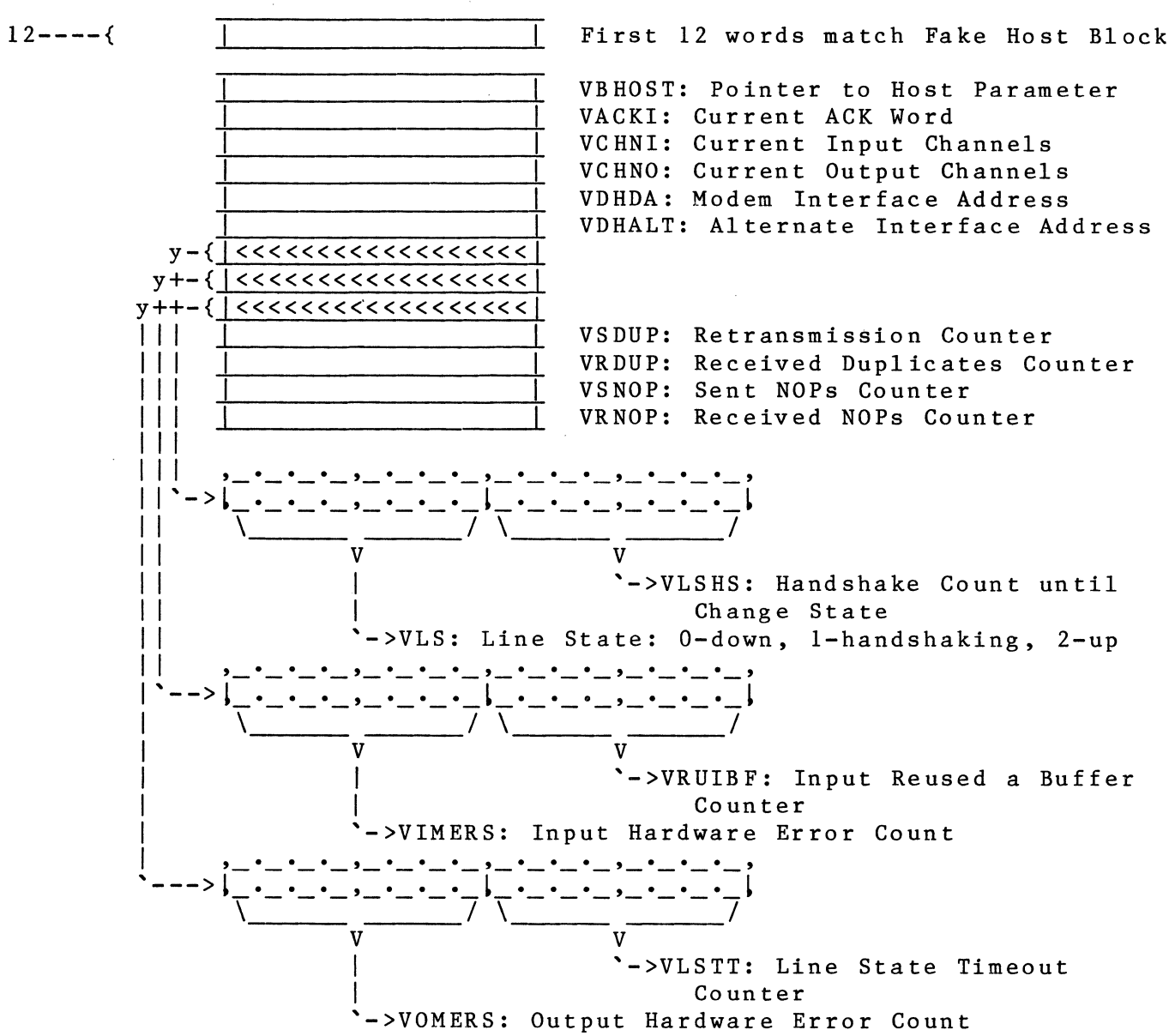
Fake Host Parameter Block (cont.) - Words 13-19 (bytes 18-25!)



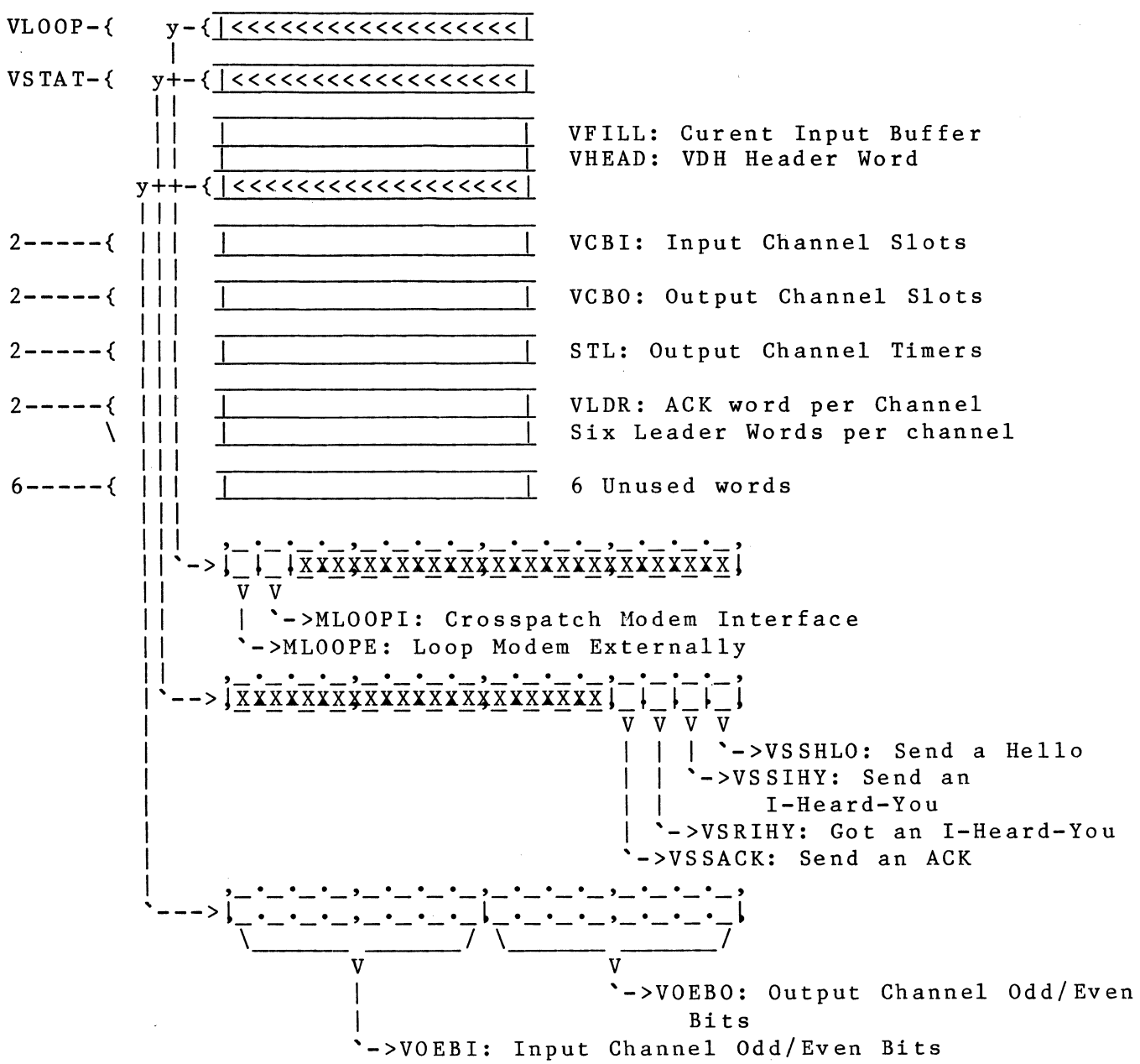
Return for JAMLED
Return for JAM
Return for DOZE
Return for SUCK
Return for WAIT
Saved Data Word for DOZE
Saved Data Word for WAIT

6.9 Very Distant Host (VDH) Parameter Blocks

First 12 words match Fake Host Block (above)
Words 1-25 (bytes 0-31!):

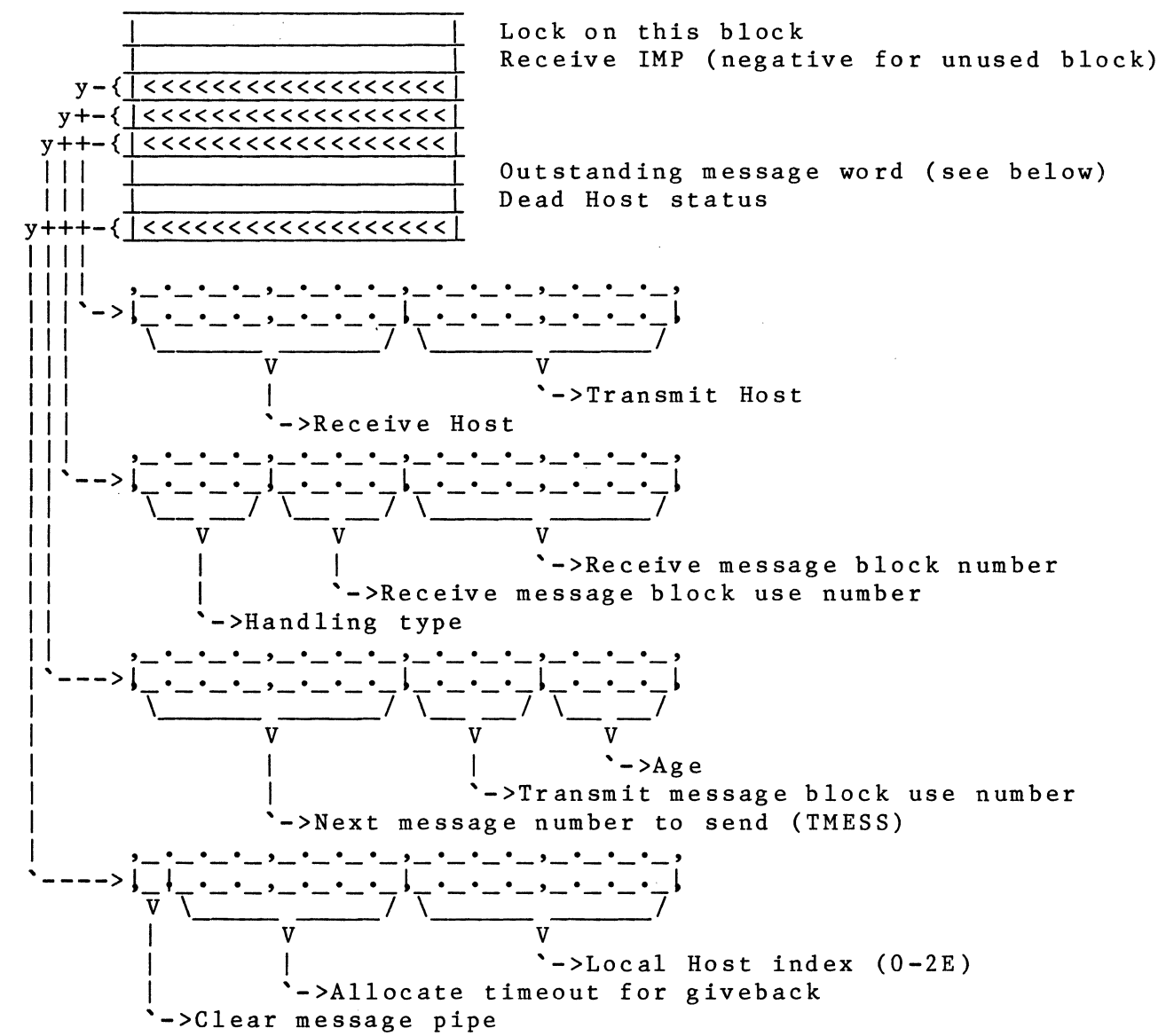


VDH Parameter Blocks (cont.) - Words 26-56 (bytes 32-6F!)



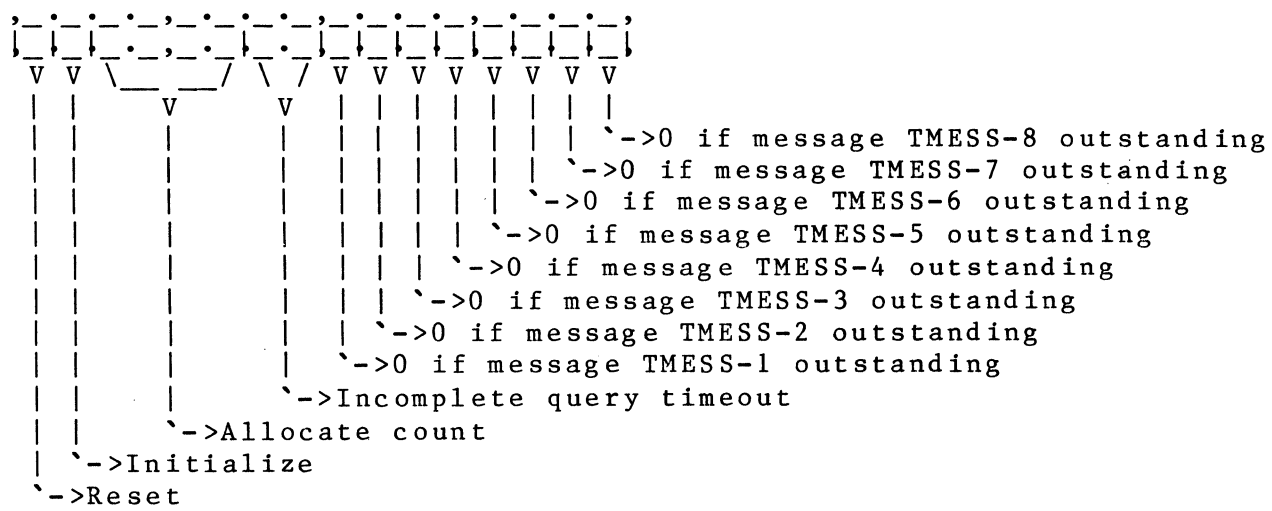
6.10 Transmit Message Block Table

56 8-word blocks



Transmit Message Block Table (cont.)

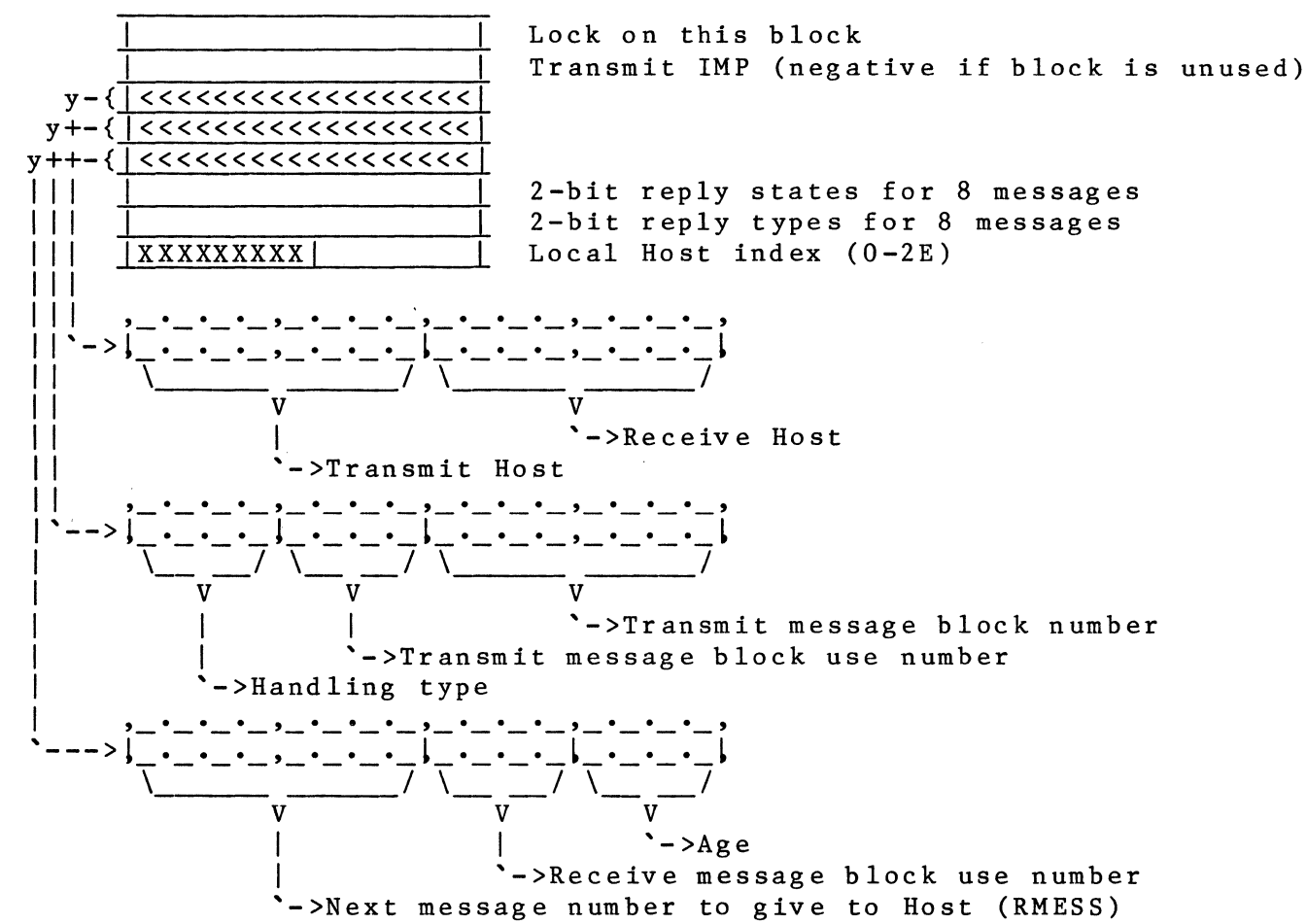
Outstanding message word:
(sixth transmit message block word)



6.11 Receive Message Block Table

56 8-word blocks

(see below for reply states and types)



Receive Message Block Table (cont.) - State and Type words

State and type words contain 8 2-bit entries, numbered i=1,...,8 (left to right), where entry number i corresponds to either message number RMESS-i (previous window, *P*), or to message number RMESS+8-i (current window, *C*), depending on the values of state and type. The meanings of various combinations, along with the window (*P* or *C*), is given by the following table:

STATE->	IDLE 00	REQUEST 01	MESSAGE 10	REPLY 11
TYPE ! V				
00	RFNM sent *P*	(unused)	ALL1 sent/ msg rcvd *C*,*P*	RFNM1 to be sent *P*
01	ALL8 sent *P*	REQ8 rcvd *C*	GVB rcvd *C*	RFNM8 to be sent *P*
10	DEAD sent *P*	ALL1 to be sent *C*	DEAD rcvd *C*	DEAD to be sent *P*
11	INC sent *P*	ALL8 to be sent *P*	INC rcvd *C*	INC to be sent *P*

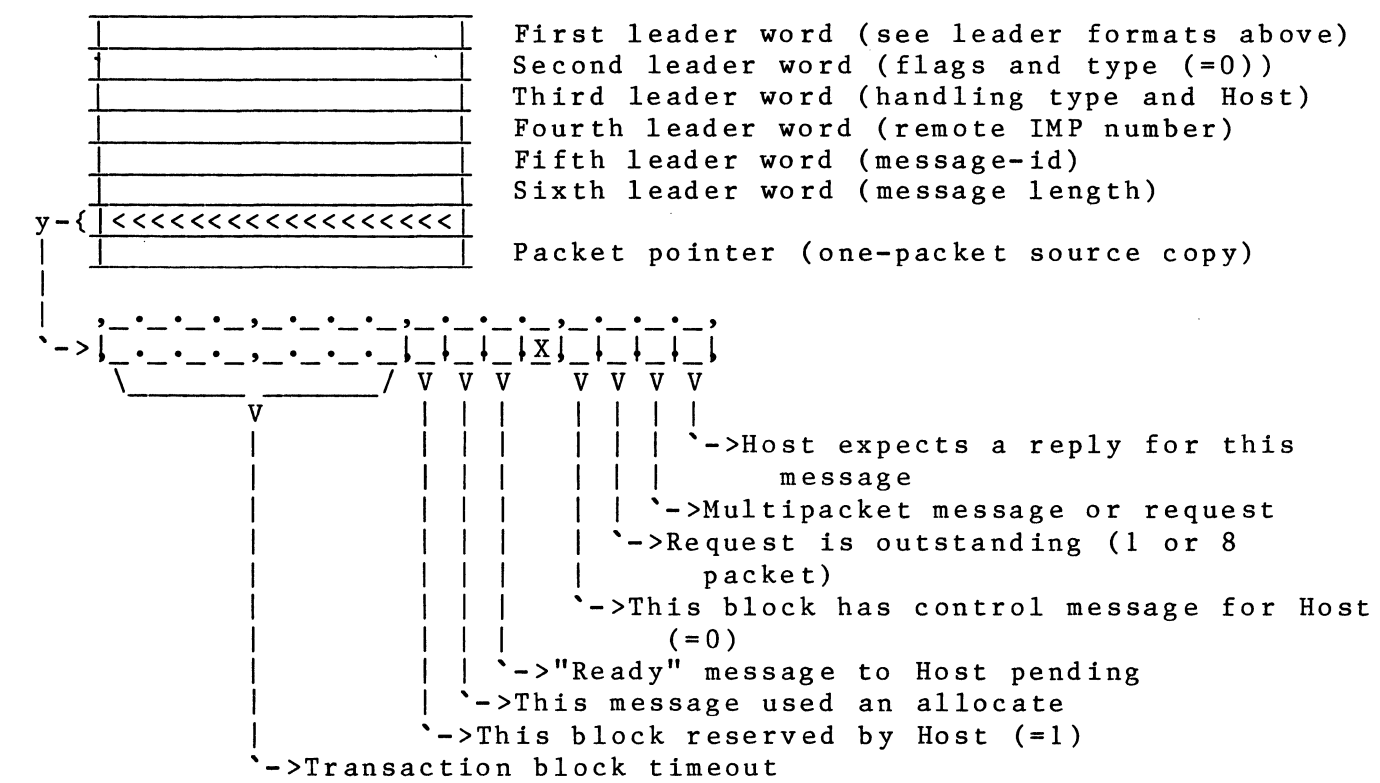
6.12 Transaction Block Table

40 8-word blocks

Transaction Block states depend on the status bits in the sixth word. Blocks generally progress through the following three states in order. Each Transaction Block keeps track of the status of a single message from the time of reading in the message leader until the RFNM is sent to the Host.

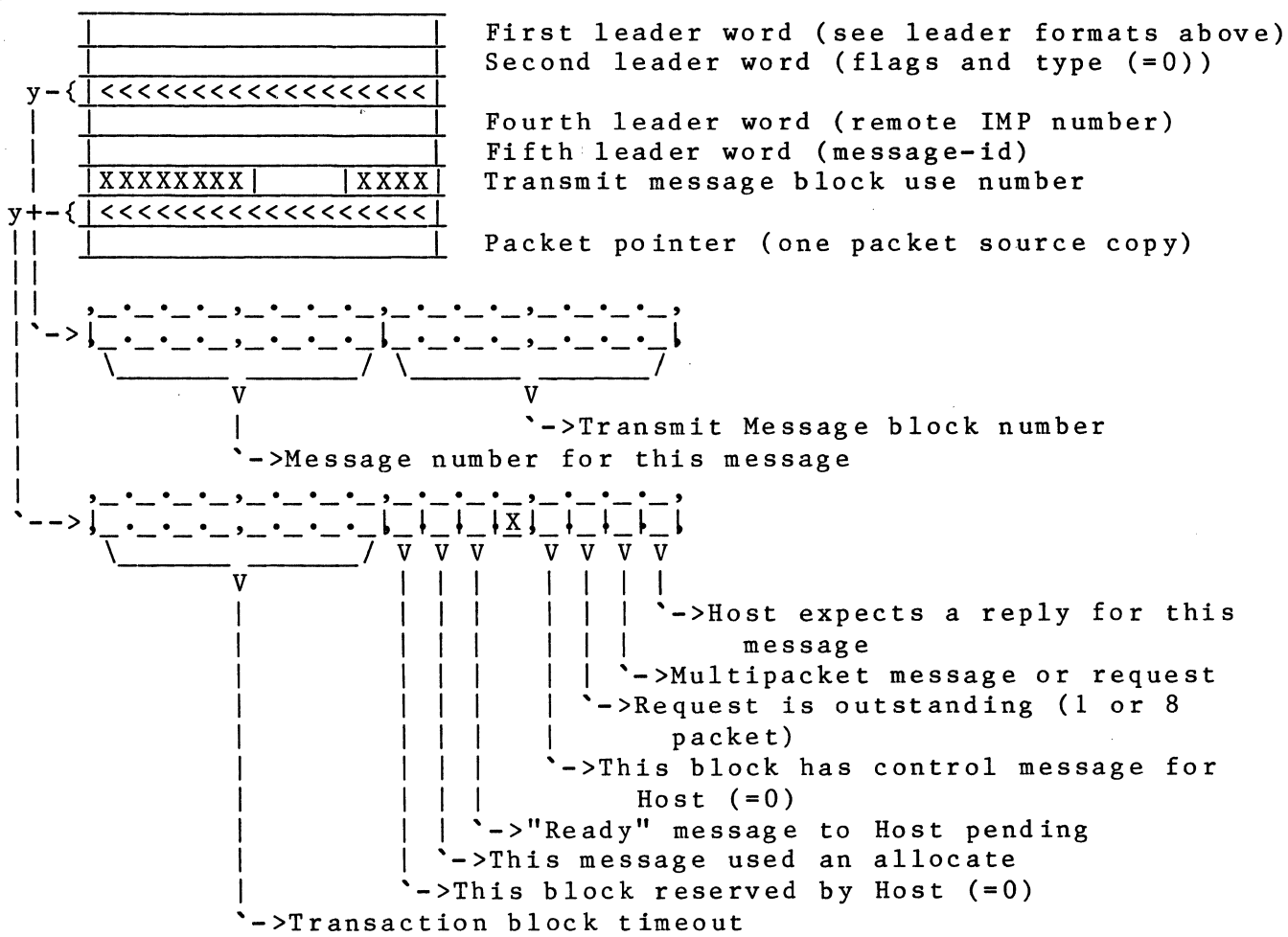
6.12.1 Reserved Transaction Block Format

Blocks reserved by Host to IMP processes



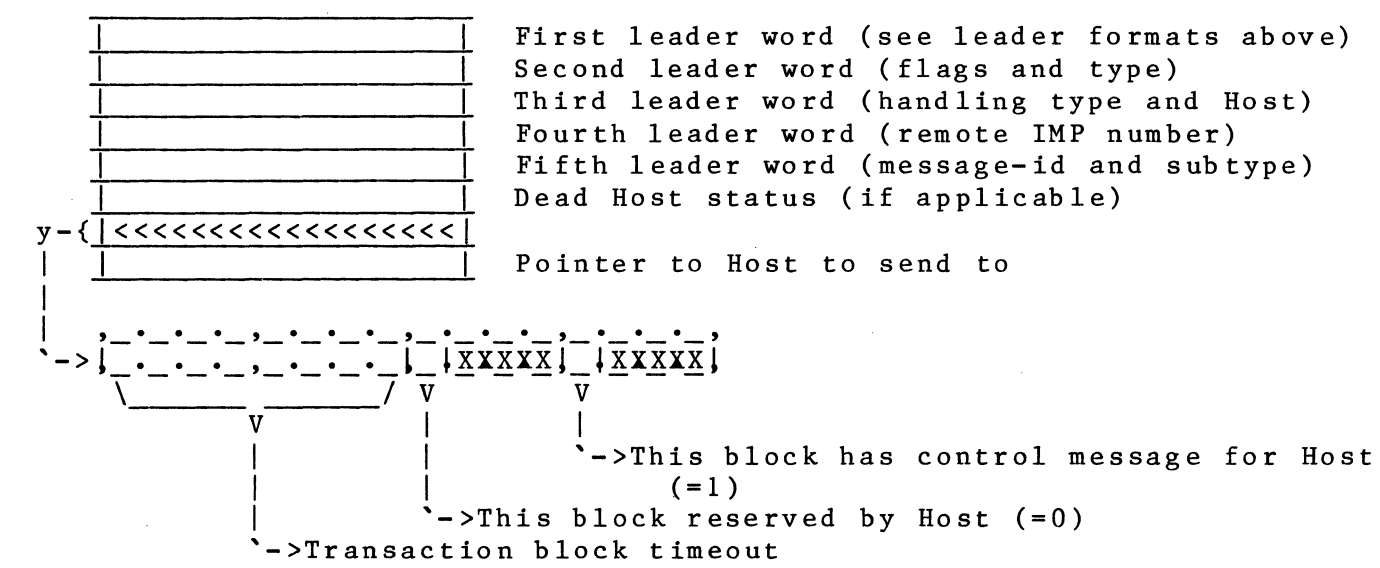
6.12.2 Outstanding Message Transaction Block Format

Messages awaiting replies from destination IMP



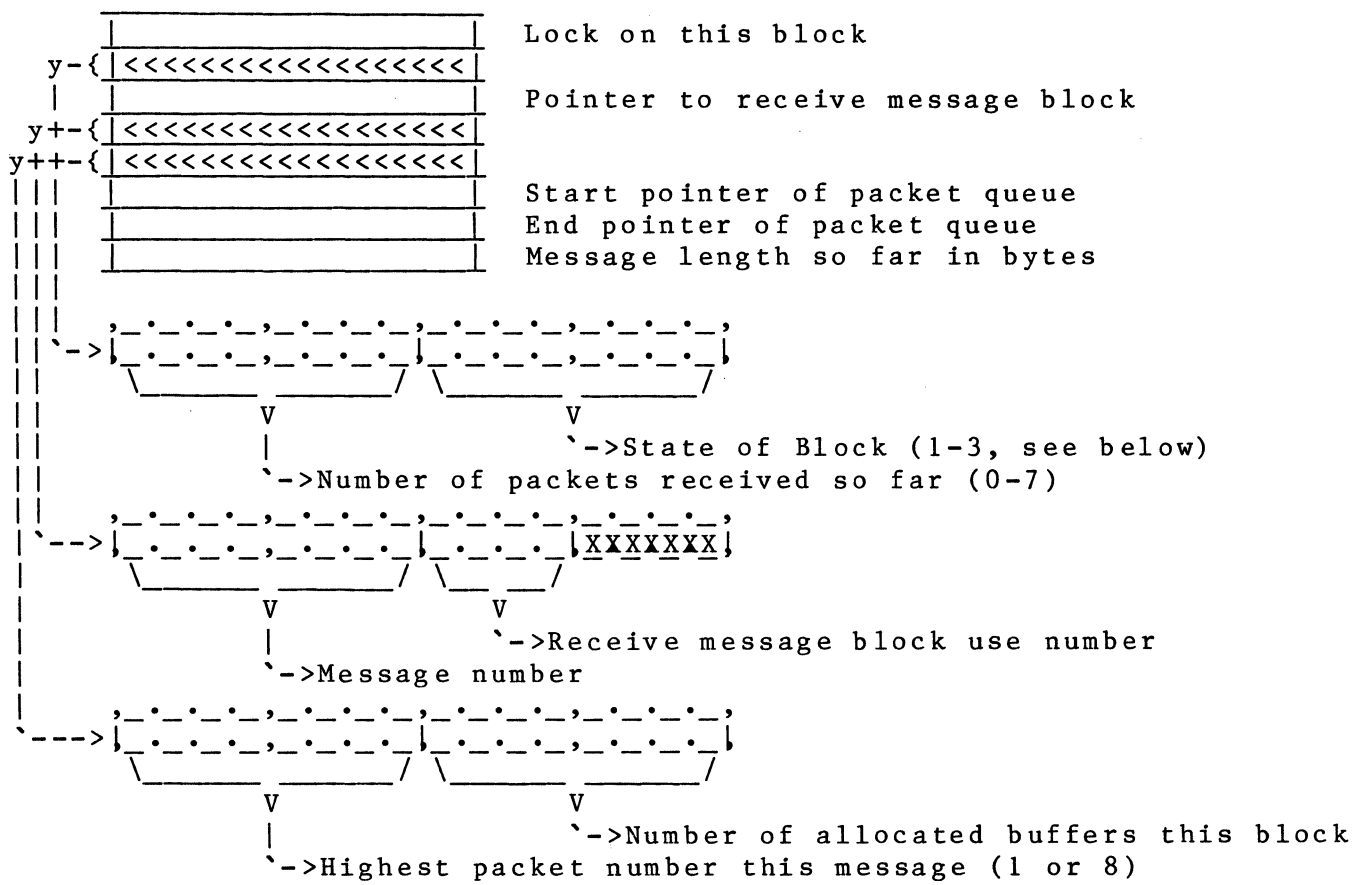
6.12.3 Control Message Transaction Block Format

Control Messages awaiting transmission to Host



6.13 Reassembly Block Table

24 8-word blocks



Reassembly Block Table (cont.) - Block states:

Unused - State byte = 1, all other words (except lock) = 0

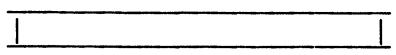
No-name - State byte = 3, number allocated = 1 or 8, receive message block and use number set up.

Partial - State byte = 2, number allocated = original allocation less number so far, packets so far = 1 to 7, message block, use, and number set up, highest packet number set up once last packet is received, length is total for packets received so far.

Complete - State byte = 2, number allocated = 0, packets so far = highest packet number, else like partial.

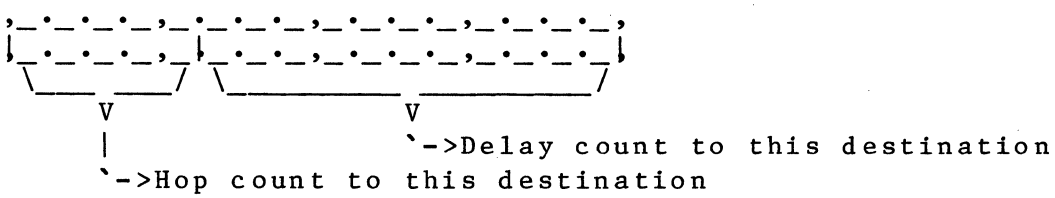
6.14 Routing Tables

RUT, Route Use Table, best line directory - 1 word/IMP

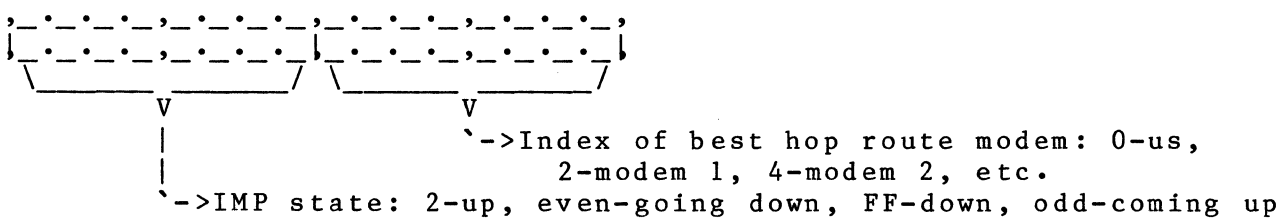


Pointer to modem line for best route to IMP

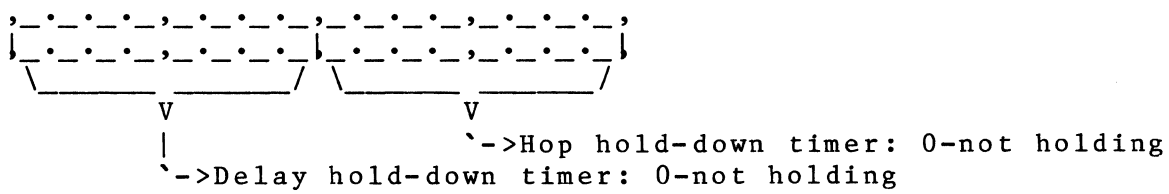
HOPDEL - Hops and Delay Table - 1 word/IMP



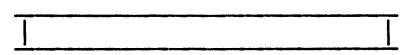
ISTATE and HOPRUT - IMP state and best Hop Route - 1 word/IMP



DELTIM and HOPTIM - Delay and Hop Route Hold-down Timers - 1 word/IMP

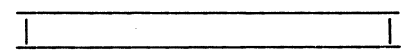


THSDEL - Change in Delay for This Routing Tick - 1 word/IMP



Accumulated change in delay, this routing tick

OLDDEL - Change in Delay for This and Last Routing Tick - 1 word/IMP



Accumulated change in delay, latest two ticks