60474500

CONTROL DATA
CORPORATION

NETWORK PRODUCTS

COMMUNICATIONS CONTROL PROGRAM
VERSION 3
SYSTEM PROGRAMMERS REFERENCE MANUAL

CDC® COMPUTER SYSTEMS
255X SERIES
NETWORK PROCESSOR UNIT
HOST OPERATING SYSTEM
NOS 1

REVISION RECORD

REVISION*	DESCRIPTION
A	Initial Release. CCP Version 5, PSR Level 504

Publication No.
60474500

*Revision letters I, O, Q and X are not used.

Address comments concerning this manual to:

CONTROL DATA CORPORATION
Publications and Graphics Division
P.O. Box 4380-P
Anaheim, CA 92803

or use Comment Sheet in the back of this manual.

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual, are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

Page	Rev	Page	Rev	Page	Rev
Cover	-				
Evaluation Sheet	-				
Title Page	-				
ii thru xiii	A				
1-1 thru 1-18	A				
2-1 thru 2-13	A				
3-1 thru 3-8	A				
4-1 thru 4-30	A				
5-1 thru 5-26	A				
6-1 thru 6-54	A				
7-1 thru 7-26	A				
8-1 thru 8-15	A				
9-1 thru 9-24	A				
10-1 thru 10-17	A				
11-1 thru 11-23	A				
12-1 thru 12-16	A				
A-1 thru A-13	A				
B-1 thru B-6	A				
C-1 thru C-34	A				
D-1 thru D-8	A				
E-1	A				
F-1 thru F-2	A				
G-1 thru G-37	A				
H-1 thru H-92	A				
I-1 thru I-13	A				
Index-1 thru Index-19	A				
Comment Sheet	-				
Mailer	-				
Back Cover	-				

PREFACE

This manual describes those externals of the Communications Control Program (CCP), Version 3.1, necessary to aid a systems programmer in making minor modifications to standard CCP software. The manual also provides a sufficient basis to understand those standard programs which interface to any new terminal interface program (TIP) that the user writes for a nonstandard terminal. CCP is used with the CDC® 255x Series Network Processor Unit (NPU).

This manual is intended for the user who is familiar with CCP basic functions and the role of CCP in network processing; these functions are described in the CCP 3 Reference Manual. The user should be experience with the PASCAL programming language and the CYBER CROSS support system software. The user should also be familiar with the state programming language.

CONVENTIONS USED

Throughout this manual, the following conventions are used in the presentation of statement formats, operator type-ins, and diagnostic messages:

ALN Uppercase letters indicate words, acronymns, or mnemonics either required by the network software as input to it or produced as output.

aln Lowercase letters identify variables for which values are supplied by the NAM or terminal user, or by the network software as output.

ooo Ellipsis indicates that the omitted entities repeat the form and function of the entity last given.

Square brackets enclose entities that are optional; if omission of any entity causes the use of a default entity, the default is underline.

Braces enclose entities from which one must be chosen.
These delimiters indicate elements of the virtual terminal format.

Unless otherwise specified, all references to numbers are to decimal values; all references to bytes are to 8-bit bytes; and all references to characters are to 8-bit ASCII-coded characters.

RELATED MANUALS

The publications listed below contain additional information on both the hardware and software elements of the 255x Series Network Processor Unit and the CCP and related software. These publications can be ordered from Control Data Literature and Distribution Services, 304 North Dale Street, St. Paul, MN 55103.

Publication Title	Publication Number
Network Products Communciations Control Program Version 3 Reference Manual	60471400
CYBER CROSS System Version 1 PASCAL Reference Manual	96836100
CYBER CROSS System Version 1 Macro Assembler Reference Manual	96836500
CYBER CROSS System Version 1 Micro Assembler Reference Manaul	96836400
CYBER CROSS System Version 1 Link Editor and Library Maintenance Programs Reference Manual	60471200
Network Products UPDATE Reference Manual	60342500
State Programming Reference Manual	60472200
Macro Assembler Reference Manual Mass Storage Operating System NOS Version 1 Installation Handbook	60435700

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or parameters.

CONTENTS

APPENDIXES

INDEX

FIGURES

TABLES

OVERVIEW 1

This section describes Communications Control Program (CCP) on a conceptual level. The description gives the programmer an overview of how CCP functions in a Network Processor Unit (NPU). For a more complete description of how CCP functions in a network, refer to the CCP 3 Reference Manual.

CCP provides the software necessary to process data (messages) through the network communications portion of a Control Data network. The network communication functions that are moved from the host (a CYBER 70/170) to the NPU allow an application program in the host to process data as if program was connected to a virtual terminal that was connected directly to a host port. Since virtual terminals must be either batch or interactive, host processing becomes almost independent of terminal type.

The network communications tasks that have been moved into the NPU are of four types:

- Multiplexing data to and from the terminals
- Demultiplexing data and storing it in buffers for buffered high-speed transfers to and from the host
- Converting all terminal protocols into either an interactive virtual terminal protocol or into a batch virtual terminal protocol
- Regulation of the volume of message traffic handled

CCP is divided into several major subsections to handle these tasks. See figure 1-1.

- Base modules to provide NPU control and general services to other major subsections
- Network communications subsystem modules (internal processor and service module) to provide routing and network configuration services
- A host interface (HIP and coupler) subsection
- Terminal interface (TIP or LIP) subsections for each major class of terminal, including an interface to a remote NPU and the interface from a remote NPU to a local NPU. (A local NPU is coupled directly - by hardware - to the host. Any NPU lacking this coupler is a remote NPU.) Terminal interfaces are handled by a TIP; NPU to NPU interfaces are handled by a LIP at each end of the interface.
- A multiplex subsystem that provides the hardware and software interface between the NPU and the various types of terminals (it also provides the interface between local and remote NPUs)

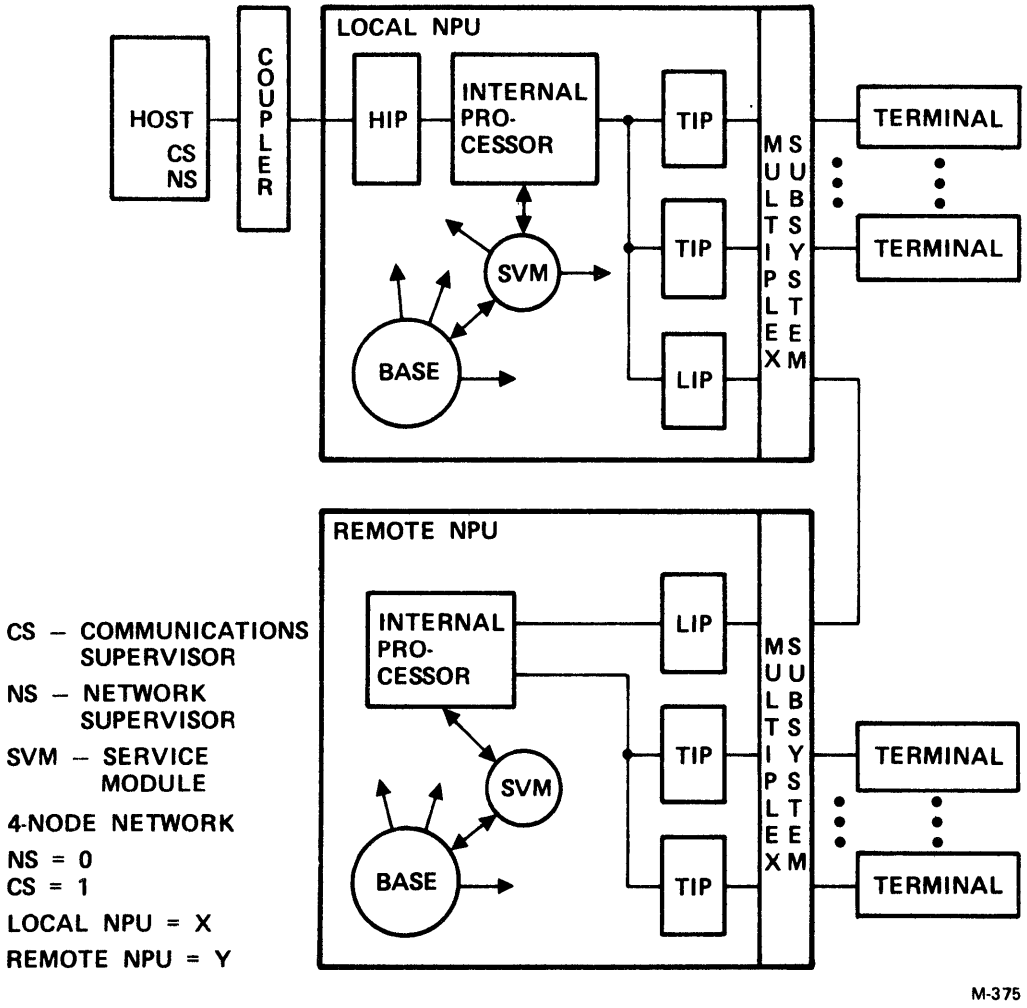


Figure 1-1. Role of NPU in a Network

CCP passes ASCII messages to and from the host in interactive virtual terminal (IVT) or batch virtual terminal (BVT) format. CCP passes messages to and from the terminals in a code and format appropriate to the terminal. Downline messages (output from the host) are switched to the proper terminal and translated from ASCII IVT/BVT to terminal format and code. Upline messages are normally received from the terminals, converted to IVT/BVT ASCII, and passed to the host.

NOTE

> A transparent mode is available. In this case, the message remains in the terminals code and format throughout the network.

CCP DESIGN

CCP can be classified as a responsive (driven) system rather than an active system. The external stimuli that drive the system come (1) from the host in the form of downline messages and commands and (2) from the terminals in the form of upline messages. At the two principal interfaces (HIP or LIP on the upline side; multiplex subsystem on the downline side), hardware and formware do much of the preparation for a message or command transfer.

PRIORITY PROCESSING AT THE INTERFACES

At the interfaces, CCP is largely interrupt-driven and operates at priority levels. Interrupts are processed immediately unless a higher priority task is already being performed. The interrupt can be processed completely at that time. However, many tasks take so much time that it is preferrable to defer part of the task processing until later. This is done by generating a worklist that defines the parameters for the task and then queuing that worklist (task request) to the module that must process it. The multiplex subsystem works this way and has its own worklist processor to schedule the appropriate modules at a priority level.

The principal priority tasks in order of decreasing importance are as follows:

- Memory errors
- Multiplex loop errors
- Host coupler events
- Real-time clock count
- Output data demands (multiplex subsystem)
- Input data frame received (multiplex subsystem)

The output of the priority level is either a message that the NPU can route to the specified destination, or a command for the NPU which CCP interprets to change its own processing mode.

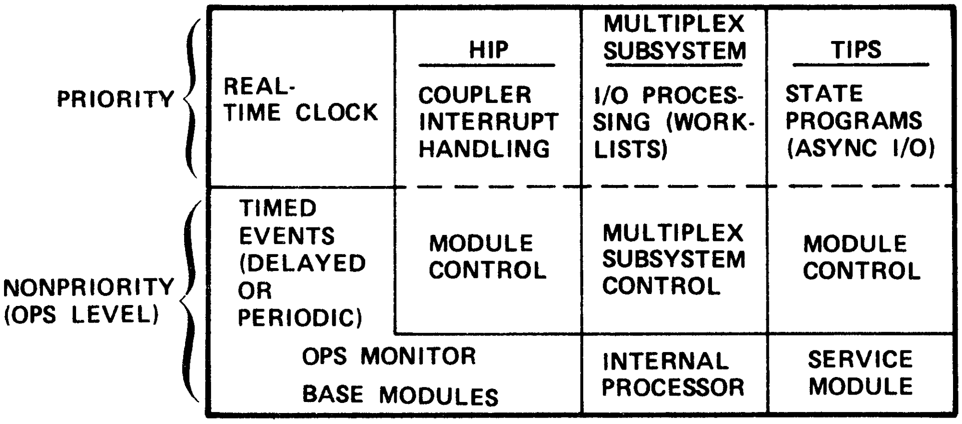
Some major modules operate largely on the priority level (the multiplex subsystem, for example); others have portions that operate on a priority level while the remainder of their processing is on a nonpriority (OPS) level (HIP, TIPs, for example). A few of the major modules do almost all of their processing on the OPS level (internal processor and service module).

OPS-LEVEL PROCESSING

When no priority tasks are pending, CCP processes OPS-level tasks. There is an OPS Monitor which assigns tasks by scanning all the nonpriority worklists. These worklists are queued to one or another of the major system modules. Each of these major modules (such as a TIP, LIP, HIP, internal processor, or the service module) has its own internal worklist scanner that determines the exact task to be performed on the basis of a workcode in the worklist.

OPS-level worklists can originate either from a priority task or from another nonpriority task. For example, a downline message from the host is first handled on a priority basis as the HIP and the coupler set up to receive the message and actually input the message into the assigned buffers in the NPU. When the message (or part of a message called a block) has been completely received, CCP is ready to process it. This block is passed on a nonpriority basis to the internal processor with a worklist. The internal processor routes the block to the proper TIP with a worklist. The TIP passes the message (still at OPS-level) to the multiplex subsystem. The multiplex subsystem sets up the transfer on the OPS level and then outputs the message to the terminal, one character at a time, on a priority basis.

Figure 1-2 shows the processing levels for most of the major modules.



M-379

Figure 1-2. Priority and Nonpriority Tasks in CCP

DOWNLINE MESSAGE PROCESSING

Downline messages originate serially from the host in blocks. A block is a full message or one part of a message treated as a unit. The block is passed to the NPU via the host interface program (HIP), which is responsible for all transfers across the coupler. See figure 1-3. The HIP passes the block to an internal processor, which examines the block header to gain information about the terminal receiving the message. Each category of terminal is serviced by one of the terminal interface programs (TIPs). The internal processor passes the message to the appropriate TIP. The TIP processes the message (translates it to terminal code and format) and passes the message to the command driver in the multiplex subsystem. Before this, the TIP requests the multiplex subsystem to prepare the NPU-to-terminal line for a transmission.

At the multiplex subsystem, the output message block is multiplexed (along with other message blocks in the process of being transmitted to the terminals) and sent to the terminal one character at a time. Actual timing of the character transmission depends on an output data demand (ODD) signal sent by the communications line adapter (CLA) to the NPU. An output processor in the multiplex subsystem handles this activity. The host is informed of message transmission progress twice: first, when the block is completely accepted by the NPU, and again after the block is completely transmitted to the terminal.

UPLINE MESSAGE PROCESSING

Upline messages (input to the host) originate at the terminals and are sent one character at a time to the input loop of the multiplex subsystem. An input processor picks up all characters and stores them in a temporary buffer called the circular input buffer. The TIPs are responsible for furnishing the multiplex subsystem a set of programs which are used to demultiplex the data into line-oriented input buffers. Code and format conversions are performed along with the demultiplexing. Since block size is a CCP/host build-time parameter, any message that exceeds the maximum block size is divided into blocks. Each block is then treated as a separate message unit by CCP. The message is converted from terminal code and format to ASCII IVT/BVT. (A transparent mode is also available for upline messages, but it is restricted to interactive terminals.) After a complete block has been assembled, the multiplex subsystem notifies the appropriate TIP, which finishes processing the message. Then the TIP passes the message block to the HIP, which in turn passes the block to the host. Terminals are notified of processing progress according to the demands of the terminal protocol. Figure 1-4 shows simplified upline message processing.

CCP FEATURES

CCP provides several message processing features:

- IVT/BVT relieves host application programs of needing to handle terminal protocols. The TIPs convert messages to/from ASCII IVT/BVT for the host.

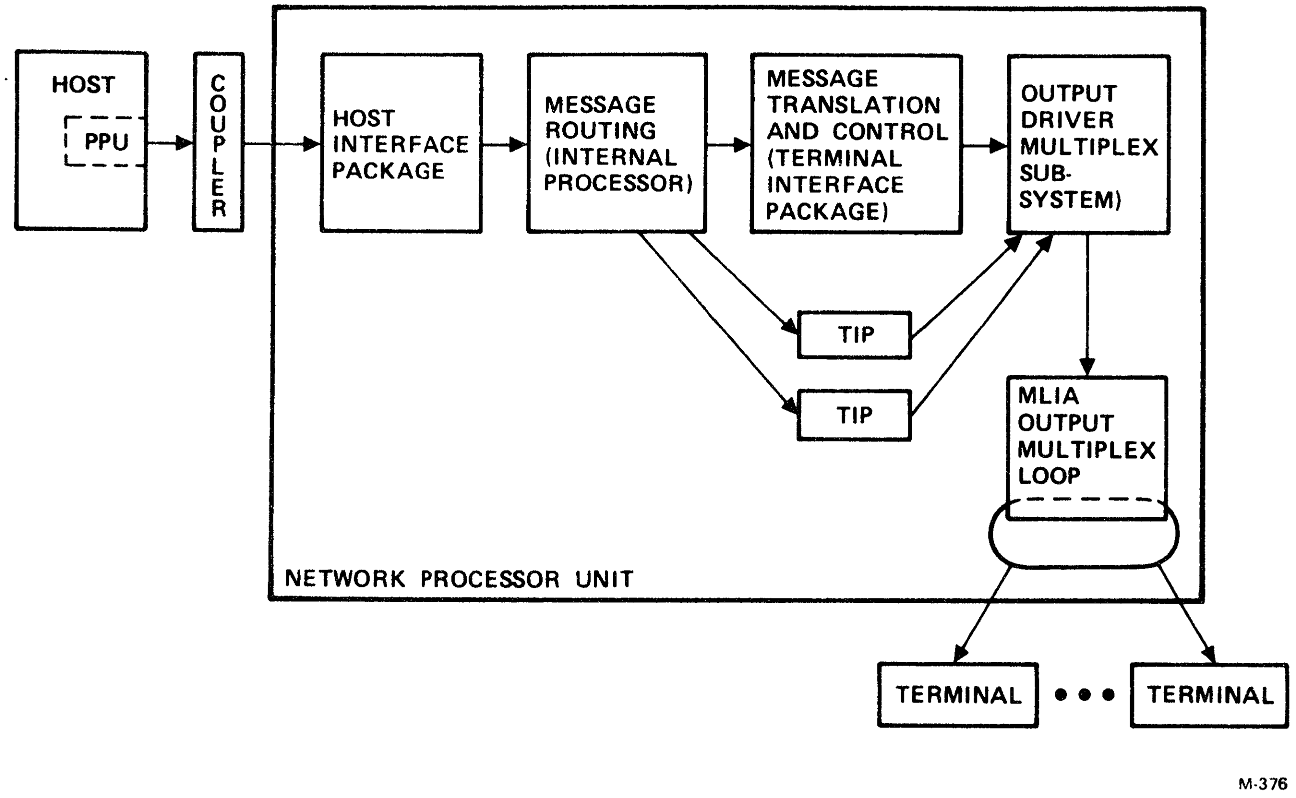


Figure 1-3. Downline Message Processing

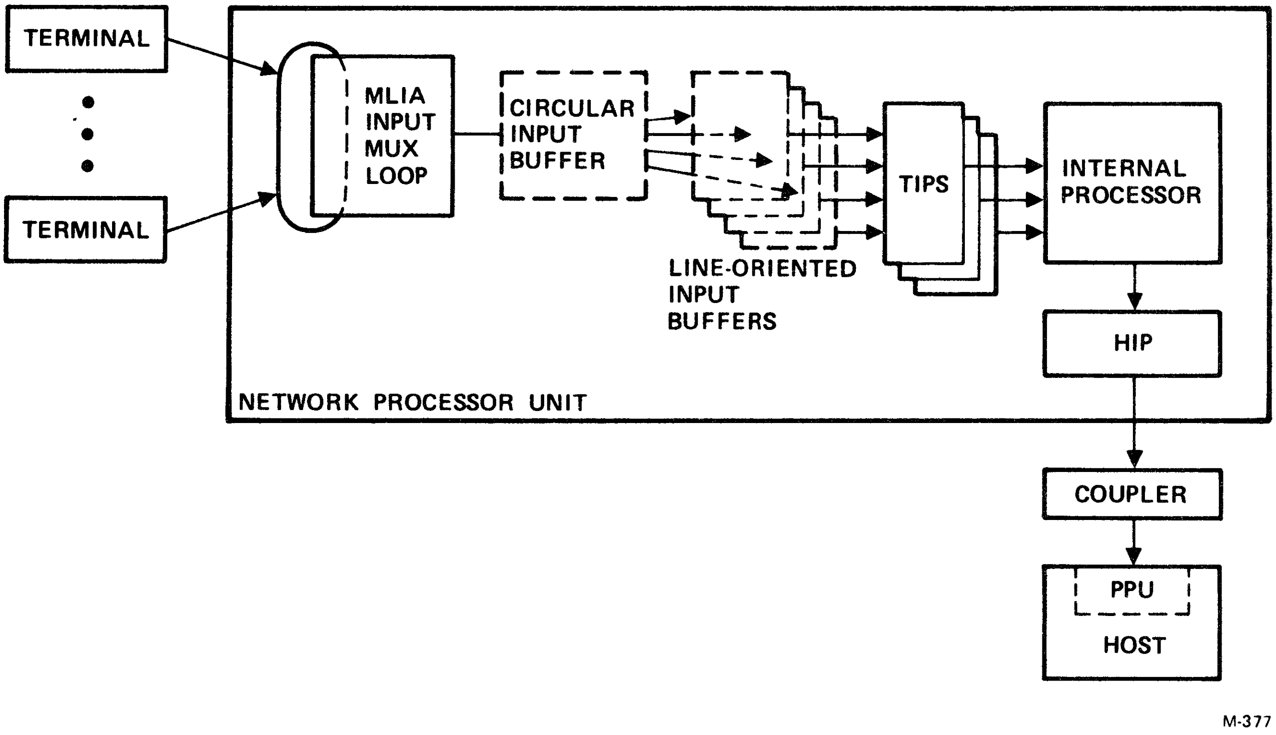


Figure 1-4. Upline Message Processing

- Block protocol relieves the NPU and the host of upline message length restrictions. Any size input message is accepted; when the normal maximum number of input characters has been received (2048 bytes including NPU-added header bytes), the block is declared full. It is processed for shipment to the host and another block is started. Blocks are designed so that the only block or the last block of a message is clearly designated (MSG type block).

- The multiplex subsystem provides hardware and software which makes the terminal hardware characteristics invisible to the TIPs. The TIP needs to know only the terminal type.

- The NPU regulates its input (rejects incoming messages) under one of several conditions - The entire NPU is short of assignable space (buffers) for message processing - An individual TIP is using too many buffers at any one time - An accept input/accept output flag is being set by the NPU or by the host - Message priority is lower than the current logical link regulation level.

  In this way, the NPU rejects messages directed to it when those messages might cause peak loading problems severe enough to stop the NPU.

- Priorities exist so that time-critical tasks can interrupt non-time dependent tasks. The time-critical tasks are concerned with either the multiplex subsystem (input and output processing at the lines to the terminals plus various errors that occur during this processing) or the NPU console. Since the console is rarely used, these latter interrupts have minimal system impact. The lowest priority is not interrupt-driven. It is called the operations (OPS) level. Most processing occurs on the OPS level.

- Programs are written in PASCAL or using state programming instructions. (A few frequently used routines are written in macroassembly language.) There is no correlation between language used and operating priority. PASCAL was chosen for its simplicity of use and because it is an effective language for manipulating table entries. Much of the CCP processing depends on information saved in tables. The OPS level of any program (TIP or otherwise) uses PASCAL code.

  For some purposes, it is more effective to write code on the firmware level (also called multiplex-level processing). State programming instructions are used for this. Such programs demulti-plex data and translate code and format. Every TIP has at least two firmware level programs: a downline text processing program and an upline input state program.

  The HIP does not use firmware programs directly; the LIP does not have a text processing program. However, several of the general support programs that are written in macroassembly language contain portions that are written in firmware. These programs should not be altered by any user.

- Three methods of communication between modules are provided: direct calls, queued calls (using worklists), and setting global variables in tables, which are then accessed by other programs.

- A special program (LIP) handles communications between a local and a remote NPU. The remote NPU handles most functions that a local NPU handles in a system without a remote NPU. Downline blocks in the local NPU are sent to the remote NPU by means of a special protocol (CDCCP). The remote LIP reconverts the blocks to normal format and passes them to the internal processor for normal routing and processing by the TIPs, etc. The upline blocks are prepared in the remote NPU as if for the HIP. Then the blocks are reformatted in CDCCP protocol and sent to the local NPU. The LIP in the local NPU reconstitutes the blocks and passes them to the HIP.

CCP MODULAR STRUCTURE

CCP can be considered as a group of generalized modules that provide saervices for the TIPs, which interface the terminal protocol to the host (block) protocol. Terminal-oriented programs are called Terminal Interface Packages (TIPs). The modularization of CCP is shown in tables 1-1 and 1-2.

Most of CCP is always resident in the NPU. It is downline loaded from the host. After loading is complete, there is additional communication between host and CCP to configure all the tables which hold line-and terminal-oriented information. A few programs use an overlay area (appendix E).

- On-line diagnostics, a series of closed loop tests available only if the user has purchased a network software maintenance contract.
- Control for loading a remote NPU (if any exists) if this is the local NPU.

CCP PROGRAMMING METHODS

CCP provides the interface for the network between terminal protocols and the host (block) protocol. It also provides multiplexing to match the high-speed block transfers at the host interface with the low-speed character-by-character transfers at the line interfaces to the terminals.

BLOCK PROTOCOL

Block protocol defines three principal types of block:

- BLK and MSG blocks carry data. No block can have more than 1048 bytes. The host is responsible for block size downline; the TIPS (input state programs) are responsible for block size upline. MSG blocks carry a full message or the end of a message. BLK blocks carry all segments of a message except the last or only segment.
- CMD blocks carry commands and status. The service module (SVM) handles generalized commands. Some commands can also be directed to and from TIPs; these do not use SVM.
- All other blocks carry communications protocol information such as acknowledgements, breaks, and restarts.

TABLE 1-1. CCP MODULES

Module	Major Function	Normal Calls
Terminal-Oriented		
Mode 4 TIP	Handles synchronous Mode 4A/4C terminals	PT4...
ASYNC TIP	Handles asynchronous terminals using teletypewriter protocols	PT... AP... AF...
HASP TIP	Handles synchronous HASP work-stations	HS... HASP...
Link Interface Program (LIP)	Handles link protocol between local and remote NPUs (NPU-to-NPU link is treated as a line by the multiplex subsystem)	various
Host-Oriented		
Host Interface Program (HIP)	Handles block protocol between host and NPU; transfers use the host coupler	PTHIP...
General Support		
Base system	Includes a monitor, timing, standard subroutines, NPU console services, and task calls (worklists)	PB...
Multiplex subsystem	Part of the base system; contains command driver, and input/output multiplex loops. (The multiplex subsystem consists of hardware, software, and firmware.)	PM...
Network communications	Message routing, service messages, and common TIP subroutines including POIs	PN... PT...

TABLE 1-2. SUPPORT PROGRAMS FOR TIPS

Programs	Location[†]	Comments
HOST INTERFACE		
Host Interface Program (HIP)	In local NPU only	
LINK INTERFACE		
Link Interface Program (LIP)	In both local and remote NPUs	
GENERAL SUPPORT		
Operating system	B	(Includes program execution, space allocation, and interrupt handling)
Worklist handling	B	Interprogram task requests
Timing services	B	
Standard subroutines	B	
Internal processor maintenance	B	Building directories
Command driver	M	
Output processor	M	
Input processor	M	
Other multiplex subsystem routines	M	
Message routing	N	
Service module, SVM	B	Handles most commands between host and NPU
TIP support	N	Includes point of interface (POI) programs, block handlers, regulation, and IVT command processor
Inline diagnostics	N	
NPU console services	B	
Initialization programs		Released when initialization is complete

[†] B Base system
M Multiplex subsystem
N Network communications

A special class of block (ACTL) is defined for data assurance over trunks (links). It is used only by the LIP.

Each block header has information relating to routing: source/destination modes (SN and DN), which are related to the host and NPU, and a connection number (CN), which is related (through directories) to lines and terminals.

An internal processor handles downline routing by use of the directories. Upline, the originating terminal is known. Using this information, the multiplex subsystem can provide the SN/DN/CN information. Only the destination code information is used during upline routing, indicating that the data is to be shipped to the host.

All host/NPU transfers are controlled on the NPU side by the HIP. The HIP operates at OPS-level and does not process blocks except to the extent that it assures that a complete block is sent or received. The HIP can reject a request to send an input block unless enough buffers can be assigned to receive the entire block at the time the transfer is requested. No effort is made to rereceive or retransmit portions of a block

The service module (SVM) handles most commands between host and NPU. For service messages, the connection number (CN) is zero. For downline commands, the SVM processes the command (such as entering fields in a terminal related table) and returns an acknowledgement service message to the host. In processing a service message, SVM can call on a TIP or on one or more other support routines.

A few commands (such as starting or stopping message transmission on a line) are sent directly between the host and the appropriate TIP. In this case, CN is not zero.

BLOCK ROUTING

Downline block switching is done by the internal processor. Almost all blocks pass to the receiving program (TIP, LIP, or SVM) using a worklist entry. Invalid blocks are discarded. Upline blocks are routed by the internal processor to the host (directly or through the local NPU) or, in rare cases, to the NPU console.

POINT OF INTERFACE (POI) PROGRAMS

From the standpoint of the TIPs, there are certain protocol requirements that each TIP fulfills both upline and downline. Common POI programs are provided for these tasks.

- PBIOPOI - internal output POI. Downline block switching is handled by the PBIOPOI. This POI generates the proper type of reply block (acknowledgement, break, initiate, etc.) or queues the block to the TIP or SVM for further processing.

- PBPOPOI - postoutput POI. This downline POI generates an acknowledgement to the host indicating that the block has been transmitted to the terminal. It also gathers statistics for the transfer.

- PBPIPOI - postinput POI together with PBIIPOI internal input POI. These POIs handle the upline block and switch it to the host.

- PBPROPOI - preoutput POI. This POI sets up table information for downline transfers.

DIRECT AND WORKLIST CALLS

Direct calls can be made from any PASCAL program to any other. At the OPS-level, direct calls are freely made between routines of the same kind (such as SVM routines or TIP routines for the same TIP). Calls are also made freely from the SVM, a TIP, the LIP, and the HIP to support routines (base and network types.

Direct calls pass task-oriented information in either of two ways:

- Information can be stored in one or more fields of PASCAL tables (data structures). The called program is expected to find the table and the field.

- A small parameter list accompanies the call. This type of list is ordinarily restricted to a few pointers and/or numbers. In this manual this type of call is depicted as

  MNCALL parm 1,...parmn

  MNCALL is at least the first six characters of the entry point name. Param 1...parmn are the associated parameters. Parameters can be omitted, but the delimiting commas cannot (exception: terminating comma(s)).

Calls between types of routines (such as a call from a TIP to the SVM or the reverse, or a block switching call) are usually made with worklists. A worklist is a packet of information about the requested task. Worklists are queued on a first-in-first-out basis to those few modules designated to receive them. Those modules are the following:

- TIPs
- HIP
- LIP
- SVM
- Internal processor
- Timing processor
- Multiplex loop interface adapter interrupt processor
- NPU console handler

All of the named modules execute at the OPS-level. Worklists are also queued for certain priority routines in the multiplex subsystem (multiplex level). A worklist is considered to be an event that requires the CCP to take appropriate action.

The monitor scans the list of OPS-level programs to find the next event (task) which must be processed. It then passes control to that module together with the worklist. The worklist contains a workcode that most receiving modules (such as a TIP) use as the index to an internal switch determining the module entry point appropriate to the requested task.

The multiplex subsystem has its own worklist processor which runs at multiplex level (priority 3). The worklist processor handles the following functions:

- Communications line adapter status
- Output buffer transmitted
- Buffer threshold reached in multiplex subsystem
- Unsolicited input or output on a line
- Bad communications line adapter address
- Illegal frame format
- Timeout of output data demand (ODD)
- Termination of input
- CE error message generation
- Hardware errors
- Calling the TIP at OPS level for further processing

The event workcodes in the worklist define the internal switching for the multiplex worklist processor.

DIRECT CALLS ON FIRMWARE LEVEL

Input state programs and text processing programs can branch during processing. The branching calls are embedded in the code. Whenever state programs are suspended for any reason (such as finishing processing on the current input character and having to release control until the next input character is available for processing), the state programs save a pointer to the next entry point in a global table (NAPORT, MLCB, or TPCB: these are defined later). When firmware processing resumes, the appropriate table is checked for the pointers to the firmware entry point. Since the table is an OPS-level data structure, the pointers can be readily used by software on any priority level, as well as by firmware.

SPECIAL CALL TO MULTIPLEX SUBSYSTEM

TIPs or SVM call the multiplex subsystem directly to save processing time. This call to the command driver (PBCOIN) has a special parameter list called a command packet which holds information used by the multiplex subsystem to set up the table controlling this message transfer (MLCB). During the transfer, additional information is added to the MLCB, and all programs concerned with the transfer (whether software or firmware) refer to the MLCB for transfer control information. The MLCB for the transfer is released when the transfer is completed.

SPECIAL CALL TO FIRMWARE INTERFACE

A support routine (PTTPINF) is called directly by the OPS-level TIP when firmware-level text processing is to be done. All text processing for a block occurs in a single pass, although PTTPINF returns to OPS-level (within itself) frequently so that interrupts can be processed. (While processing on the firmware level, interrupts are inhibited.) For text processing, the OPS-level TIP defines a table to control the transfer (TPCB) and fills all the necessary fields before calling PTTPINF. The firmware accesses TPCB for control information and adds status information used by the OPS-level TIP after PTTPINF returns control to the TIP. The TPCB is discarded by the OPS-level TIP when it passes the block to the next program (command driver downline, HIP upline).

NOTE

Space is reserved in the TPCB for the contents of the first 16 microprocessor file 1 registers. This provides 16 full words for communication in addition to the words already defined in the TPCB.

COMMUNICATIONS USING PASCAL GLOBALS (TABLES)

Several instances of communications between modules and between different levels of programs (OPS-level/firmware level) have already been cited: worklists, MLCBs, TPCBs. Use of PASCAL globals (tables) is a way of passing information between programs or saving information for later use. CCP defines several major data structures as shown in table 1-3. Some of these are defined temporarily, to be used only for one task (such as sending a message block to a terminal) or for one sequence of tasks (such as defining terminal information from the time when the line is enabled until the line is disabled). Few structures are defined permanently. Even permanent structures may need to be reconfigured each time the NPU is downloaded from the host.

All principal data structures are defined in appendix H.

LINE INTERFACE HANDLING

Much of the line interface is the responsibility of the multiplex subsystem.

Important aspects of message transfer are as follows:

- Setting up the communication line adapter (CLA) for the transfer is accomplished by a command originating in the host and passed to the command driver via the TIP that controls this type of terminal (line). The whole process can be started by a sign-on from the terminal. Low-speed lines can use autorecognition features (part of the TIP code) to establish line speed and code type.

- Polling synchronous lines for the next input character is initiated by the command to start polling which originates in the host. The TIP, however, determines the exact moment of sending each successive polling message. The line polling message is passed to the terminal via the multiplex subsystem. It is a timed output so that failure to supply another input character in the specified period is treated as a hardware error. Unsolicited input characters are also treated as hardware errors.

- The NPU may reject input when the entire NPU is running out of buffers.

- Output data is sent to the multiplex subsystem as a block of data in terminal format and code. The output processor sends each character in response to an output data demand (ODD) interrupt from the CLA. This is a timed operation. If the ODD request does not appear in one second, this is treated as a hardware error.

- The multiplex subsystem has limited error recovery logic. If the attempt to send or receive a character fails n times, the line is declared down and the TIP and SVM are called to take the appropriate internal action and to notify the host of the line failure.

TABLE 1-3. PRINCIPAL DATA STRUCTURES

Structure	Major Functions	Principal Users
Block format	Provides vehicle for NPU-to-host communications	All modules
Service message formats	Part of block format; passes commands, status, and statistics between NPU and host	SVM, all modules
Console request packet	Controls transfer to and from NPU console	Base modules
System buffers and buffer control block (BCB)	Controls space for processing. BCBs locate assignable buffers in each of four pools of assignable buffers. Nominal buffer sizes are 8, 16, 32, and 64 words(2 bytes per word)	Base modules; all modules use buffers
Worklists, worklist control block (WLCB)	Make major task request calls from module to module. WLCB locates worklists queued to a single module	Base modules; all modules that call other modules
Timing tables	Provide periodic and delayed calls; some timing is embedded in LCBs	Base modules; TIPs, SVM
Logical link control block (LLCB)	Directory information for the link (trunk) and regulation level for the trunk; one static block per link	Routing modules; SVM, LIP
Line control block (LCB)	Line-related information, timing, pointers to TIPs and terminal-related structures (TCBs); statistics information for the line; one static block per line	SVM, timing module, TIPs, LIP, HIP, multiplex subsystem
Terminal control block (TCB)	Terminal-related information, including terminal and device type, cluster and terminal addresses, statistics, pointers, and flags for data in the current transfer. Dynamically assigned when terminal is configured; released when line disabled or terminal deleted	SVM, TIPs, LIP, HIP, multiplex subsystem
Command packet (NKINCOM)	Controls information for a multiplex subsystem I/O; builds the MLCB	Sent from TIP to multiplex subsystem
Port table (NAPORT)	Current line (port) status; pointers to MLCB and state programs controlling a transfer at the multiplex port; one static entry per line	Multiplex subsystem

TABLE 1-3. PRINCIPAL DATA STRUCTURES (Contd)

Structure	Major Functions	Principal Users
Multiplex line control block (MLCB)	Controls information for a message transfer to and from a terminal major device used by OPS level and firmware level (input state programs) to exchange information; dynamically assigned for a single block transfer (downline) or message transfer (upline)	Multiplex subsystem
Text processing control block (TPCB)	Controls information for converting code and format (downline or second pass upline) of data blocks; dynamically assigned for a single block	Responsible TIP
TIP type table	TIP related addresses	SVM, base modules
Line table	Defines principal characteristics of a line	Multiplex subsystem
Modem/CLA tables	Defines modem and communications line adapter physical characteristics	Multiplex subsystem
Terminal/device type tables	Defines physical characteristics of terminals and devices at a terminal	Multiplex subsystem

The generation of the ODD and polling messages, and the use of worklists for calls is sometimes referred to as an event driven processing system.

Physical positioning of CLAs in the loop multiplexer card cage generates a preferential processing scheme. Since only one line frame (input or output) is on the multiplex loop at any one time, the CLA farthest from the loop multiplexer has first chance to use the loop. As viewed from the front, the loop multiplexer is in the next to last slot on the right-hand side of the cage (the last slot is not used). The CLA which has first chance to use the loop is in the leftmost slot, and is the half of the CLA card associated with the switches for the top half of the card. If this NPU's version of CCP contains a LIP, the port servicing the LIP is usually placed in this preferred position since the LIP is the highest speed line in the NPU.

CCP PROGRAMMING LANGUAGES

Commonly used base programs, especially those with firmware portions, are written in macroassembly language for speed of execution. These programs should not be altered. Such programs are listed in an assembly listing.

OPS-level support programs, most priority level multiplex subsystem programs, and the OPS level of each TIP are written in PASCAL language. Altering these programs can require altering the data structures (tables) which these programs use to store and pass programming control information. These programs are listed in an MPEDIT Listing and are especially usable in a PASCAL EDIT XREF listing.

NOTE

> These programs can escape directly to firmware processing using the PASCAL INST instruction together with the firmware address of the firmware program.

The firmware parts of the TIP are called input state programs or text processing state programs. The multiplex subsystem has special firmware programs called the modem state programs. These are used to process CLA-generated status. If this status word occurs, it is usually in the same frame as an input message character.

These programs are written using a predefined set of macroassembly language macroinstructions called state instructions and are called in one of three ways:

- A direct call from the OPS-level TIP to PTTPINF for a text processing program.
- An event-driven cell, triggered by the placement of data in the circular input buffer, to the modem state programs.
- A call from a modem state program to an input state program.

The firmware programs communicate with the multiplex subsystem by releasing control (input state programs or modem state programs) and by storing information in data structures. Worklist calls can be made to the OPS-level and multiplex-level multiplex subsystem programs, or the OPS-level or multiplex-level TIP. (Multiplex-level calls to the TIP are ordinarily immediately converted to OPS-level calls to the same TIP.)

Text processing programs communicate with the calling TIP by releasing control and by storing information in the TPCB. Worklist entries to the OPS-level TIP can be made also.

INITALIZING AND CONFIGURING THE NPU 2

This section describes the loading, initializing, and configuring of the NPU.

Before the CCP can be loaded into the NPU, the host must prepare the load file. Two cases of load file preparation in the host must be considered. The normal case assumes released installation tapes and the associated installation materials. Use the techniques described in the NOS Installation Handbook (see preface) to generate a CCP load file and to update a load file using corrective code release (CCR) tapes.

The special case occurs when the user initiates his own changes to CCP. This case assumes the use of a system configure file (SCF) or the equivalent. New modules sometimes have to be generated and prepared as change tapes. In all cases, changes may need to be made to the SCF itself and to the CCP tables. Table changes are normally entered by MPEDIT statements. Such changes should be made only by qualified analysts. Consult the CDC publication index for TIP Writer's Guide bulletins.

Assuming a load file is ready, a three-step process is used to make the NPU into a fully operational network node:

- Dumping the contents of the failed NPU to the host. This is an optional procedure but is normally used. If the user has purchased network maintenance from CDC, a host application program (Network Dump Analyzer, used through Interactive Facility (IAF)) is available for a quick analysis of the dump. Refer to the CCP 3 Reference Manual for standard dump formats. If the user has not purchased this maintenance, he should devise his own programs to make the dumps readily available for later analysis.

- Loading the NPU from the host. A special overlay loading capability is available for the dump/load process.

- Configuring the NPU by specifying the network logical link, line, and terminal connections for this NPU.

INITIALIZING THE NPU

Initialization takes place in two phases: the first to load and initialize the micromemory, the second to load and initialize the macromemory.

PHASE I INITIALIZATION

BEGINA starts initialization after the following occurs:

- The macromemory is downline loaded with the phase I load file
- The host sends the start signal
- The processor starts execution at location 0000_{16} (routine BEGINA).

BEGINA first executes PIRAM to load the firmware microcode into the micromemory. Then BEGINA calls PIEX to send a coupler idle status to the host. CCP loops while waiting for the phase II load file.

PHASE II INITIALIZATION

The system initialization routine (PINIT) receives control after the following occurs:

- The phase II load file is downline loaded into the NPU.
- The host sends a start signal.
- The NPU starts execution at memory location 000_{16} (a jump to routine BEGINX). BEGINX loads general-purpose registers 1 and 3 with parameters for dynamic stack management (used during initialization of recursive routines). Register 1 contains the dynamic stack last word address; register 3 contains the dynamic stack first word address.
- BEGINX executes the PASCAL routine MAIN$. This routine disables interrupts, loads the interrupt mask, and calls PINIT.

Pinit

PINIT controls the remaining macromemory initialization. The routine resets the deadman timer for host transfers, sets the page registers, and zeroes page mode. It then calls each of the other initialization routines. Before each routine is called, a specified bit is set in the initialization status word. This word can be checked for debugging purposes if the initialization procedures fail (see CCP Reference Manual). The routines are called in the sequence given in the following paragraphs.

PIPROTECT

PIPROTECT sets memory protect bits. Before setting or clearing these bits, PIPROTECT calls PISIZCORE to determine the last addressable memory location and the last word of the buffer area. The protect bits are cleared from every buffer word and set for all other words. Use of the protect system prevents DMA devices from writing into any area but buffers. The protect system can also be used with the Test Utility Program (TUP) for debugging purposes. See appendix I.

PIBUF1

PIBUF1 starts buffer initialization. PIWINIT is called to determine DN limits, and to allocate the first node in the DN table to the NPU's local node. The IDLNK and IDTBL tables are allocated and initialized, as is the ORG DN table. An entry to TUP is allowed if the TUP option has been selected.

PIGETABLE calls PILCBS to create port and circular input buffer tables. The PIGETABLE determines the pointers to the timer, port, LCB, and subLCB tables. SubLCBs for the MLIA, console and coupler are initialized, and the first LCB is also initialized. The address variables for these subLCBs are then filled.

PIBUF1 sets the address limits of the buffer area and calls PIFR1 to initialize the file 1 (firmware) registers. A 256 word array is used. Dynamic values are assigned $FFFF_{16}$. Any nonused registers are set to zero. PBEF transfers the array contents into the file 1 registers. Next, some file 2 registers are loaded using assembly language (INST) commands.

Finally, PIBUF1 initializes the buffer maintenance control block. For each buffer size, the pool boundary is forced to an even boundary, each word in the buffer area is cleared, each buffer is released to the pool, and the normal buffer threshold is set.

PIWLINIT

PIWLINIT initiates worklists. Each active worklist is allocated one worklist-sized buffer. The put and get pointers are set. Zero-sized worklists are assumed to be inactive, and a default size of three is used, but no buffer is assigned.

PIINIT

PIINIT sets the NPU console to write mode so that the CCP banner message can be displayed. PIINIT also sets up the branch-to-low-core halt routine. This routine consists of 14 no-op instructions followed by a jump to PBHALT. The routine starts at memory location 0000_{16}. Next, PIINIT sets the time of day clock to the operator-assigned value (month, day, hour, minute, second).

PIAPPS

PIAPPS initializes any trunks in the system, using the LIP. The banner message is sent to the NPU console.

PIMLIA

PIMLIA initializes the MLIA and the CLAs. The routine checks for duplicate CLA addresses. If any are found, PBHALT is called. The svstem is also halted if the MLIA cannot be initialized correctly.

PILININIT

PILININIT sets up the multiplexer and coupler timing services by adding the MLIA and coupler subLCBs to the list of active LCBs. The data buffer size is set up for the coupler. The deadman timer is reset.

PIBUF2

PIBUF2 clears and releases the last of the data buffers. The real-time clock is started, the NPU initialized message is sent to the host, interrupts are enabled, and the deadman timer is reset. PIBUF2 passes control to PBMON (the OPS monitor routine) to start normal operation of CCP.

LOAD AND DUMP NPU

A detailed description of loading and dumping an NPU, whether local or remote unit, is given in the CCP 3 Reference Manual.

CONFIGURING THE NPU

After loading and initializing the NPU, the host configures it by establishing all logical links and logical connections for that NPU. This is done in the following sequence:

- Logical links (LL) are configured by building the LLCB.
- Trunks are configured by building the LCBs assigned to the lines treated as trunks.
- Lines are configured by building the line LCBs.
- Terminals are configured by building the TCBs.

See appendix H for the definition of the data structures known as LLCB, LCB, and TCB. Format for the service messages to configure the LLCB, LCB, and TCB are given in appendix C.

Figure 2-1 shows the sequence of configuring the NPU and the service messages and blocks used for the operation.

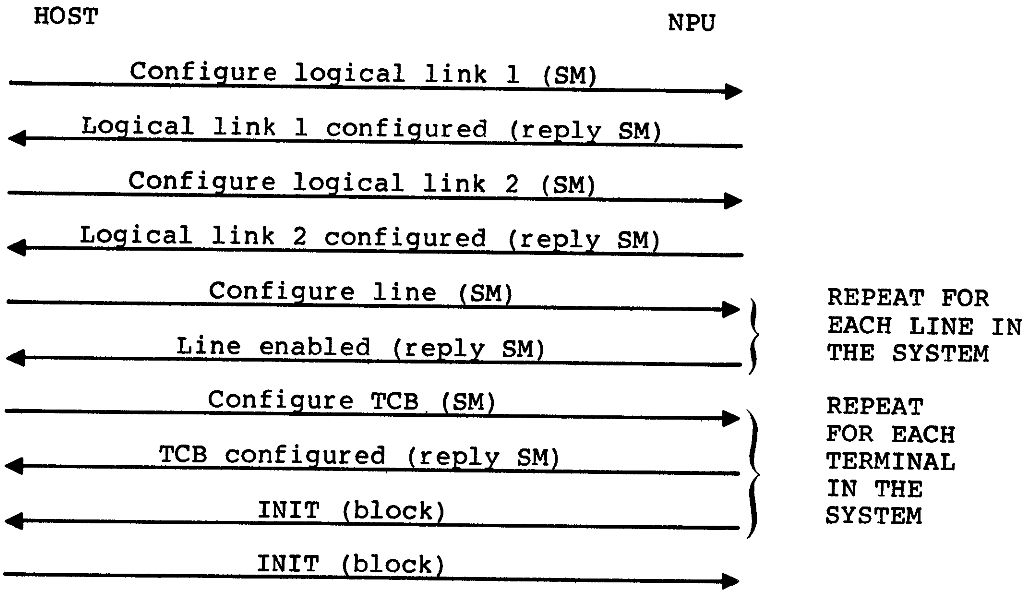


Figure 2-1. NPU Configuration Sequence

A logical connection is the association of two stations made by the assignment of a network logical address. The network logical address is a set of three numbers: two node IDs followed by a connection number. (Refer to Block Protocol portion of section 6). The two node IDs represent the nodes at which each station interfaces to the network. The order in which they appear in the network logical address specifies the direction of the connection (the destination node appearing first, then the source node). The connection number specifies a full-duplex logical channel connecting the stations. Connection number zero is reserved as a permanent service channel for service messages.

NOTE

> The network supervisor (NS) and the communications supervisor (CS) mentioned in this section are host programs. These programs are described in the CCP 3 Reference Manual (see preface).

The network supervisor in the host is informed of an NPU entering this active state by arrival of an NPU initialized service message (SM) (restoring a failed NPU) or by the arrival of the first trunk status response SM (indicating the trunk is operational). The latter occurs when an operational NPU rejoins the network.

CHANGING/DELETING LOGICAL CONNECTIONS

A change to a logical connection may be required when a TCB is already configured. This is accomplished with a reconfigure TCB SM (appendix C). The communications supervisor in the host does not change the connection number but sents the reconfigure TCB SM to reinitialize the block protocol on the logical connections.

A logical connection sails when an element (line, logical link, or application) required to support it fails or is disabled by a NOP or LOP command. (NOP is the network operator, LOP is the local operator). The NPU is informed of the termination of the logical connection either explicitly by a reconfigure TCB SM changing the connection number to zero or implicitly by deleting the TCB or the LCB on the logical link configuration. Neither changing nor deleting connections is a normal part of the initial NPU configuration process.

LINK CONFIGURATION

Two types of logical link configurations are possible in CCP:

- A link from host coupler to local NPU
- A link from local NPU to remote NPU

The functional steps in configuring a logical link are shown in figure 2-2.

The link configuration process starts when one of the following occurs:

- The NPU sends an NPU initialized SM. This is the normal configuration situation when the NPU is successfully loaded.

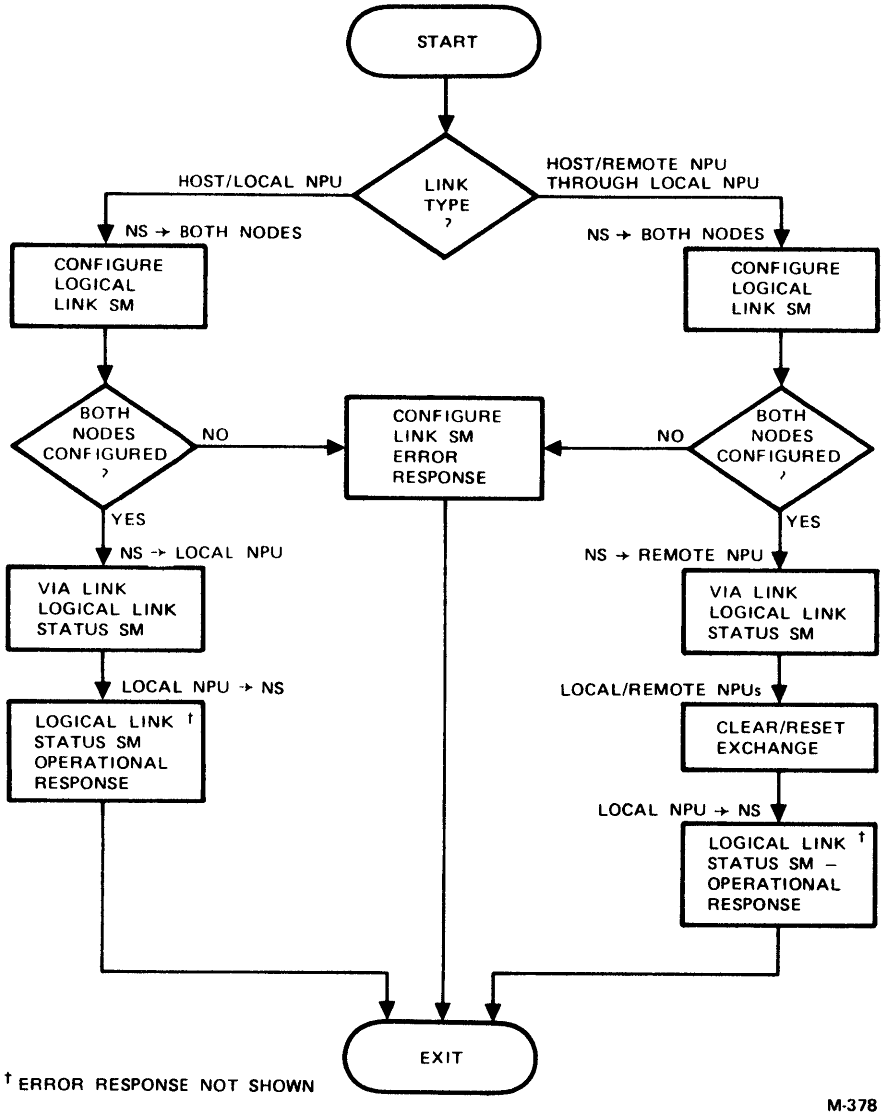


Figure 2-2. Configuring Logical Links Flowchart

- The NPU sends a trunk status operational SM. This occurs as the result of an operator-entered command.
- The network operator generates an enable trunk SM by reenabling the logical link at the host control console.

Configure Logical Link SM

NS responds to any of these situations by sending a configure logical link SM to both ends of the logical link. Message parameters include ID1 and ID2, the nodes comprising the link. ID1 is the source node for the link and ID2 is the destination node. The association between node IDs and the coupler is predefined. The SM has a destination node corresponding to the primary node ID of the NPU supporting the link.

The destination noce (CS in the host) establishes the data structure necessary to support the host end of the link. The destination node in the NPU also establishes the data structure necessary to support the link.

NOTE

> Service messages to a remote node are sent over a trunk. Once reconstituted in the remote node, such messages are treated the same way as messages received over the coupler in a local NPU.

When the link is established, a normal response SM informs NS that the link is operational. If an error occurs, the reason code in the error response message specifies the cause of the failure to configure the link.

Logical Link Status SM

NS in the host sends a logical link status SM over the newly configured link. The response SM always originates in the local NPU. Determination of response type (normal or error) is made directly within the NPU if this is a host/local NPU link, or indirectly by the clear/reset protocol over the trunk if this is a host/remote NPU link. Regulation level for the trunk in the SM reply is defined in the CCP 3 Reference Manual. An unsolicited logical link SM reply message is sent to CS when the NPU needs to change the regulation level on the trunk.

Enable Trunk SM

The enable trunk SM has two possible origins:

- Usual origin - NS in the host is notified by the unsolicited trunk status SM response that trunk protocol is established.
- Diagnostics origin - NS in the host is notified that the operator at the network console has entered a command to reenable a trunk previously disabled for diagnostic tests.

Parameters are the port connecting the local to the remote NPU and the host ordinal.

When the SM is processed, the local NPU initializes the communications line adapter and conditions the modem for line operation. The normal response includes information about communications line adapters and modem operation and identifies the node of the remote NPU, which returns to on-line condition.

LINE CONFIGURATION

Following logical link configuration, NS/CS in the host sends SMs to the terminal NPU to configure the lines between the NPU and terminals. These configure line SMs are handled by the service module in the receiving NPU. Format of the SM is the same as for the configure trunk SM.

Line configuration requires sending the following line control block (LCB) information to the NPU in the FN/FV pairs:

- Port ID for the line
- Host identifier
- Line type - includes type of duplex, communications line adapter, modem, carrier, and circuit; answering and turnaround mode; and type of transmission (synchronous, asynchronous, or CDCCP).
- Terminal type (TIP or sub-TIP required to process the terminal's data, device type, and terminal class).
- Data necessary to fill the selected fields of the LCB.

Processing of each line is governed by LCB fields. Format of the LCB is shown in appendix H.

A simplified flowchart for line configuration is shown in figure 2-3. Terminal configuration consists of configuring the terminal control block (TCB). TCB configuration is shown on the same diagram to emphasize the fact that a network cannot use the terminal until both the terminal's associated LCB and TCB are configured. After configuration, the following events occur:

- The host identifies the terminal and ascertains that it either uses an IBT or a BVT transform. The host also finds the proper regulation level to use.
- CCP identifies the protocol necessary for the data transfers and assigns a proper TIP to handle that protocol.
- The hardware in the communications line adapter and modem are prepared for data transfers.

A terminal NPU is any NPU which has a terminal attached to its I/O ports. A terminal NPU that is a local NPU can also be linked to a remote NPU.

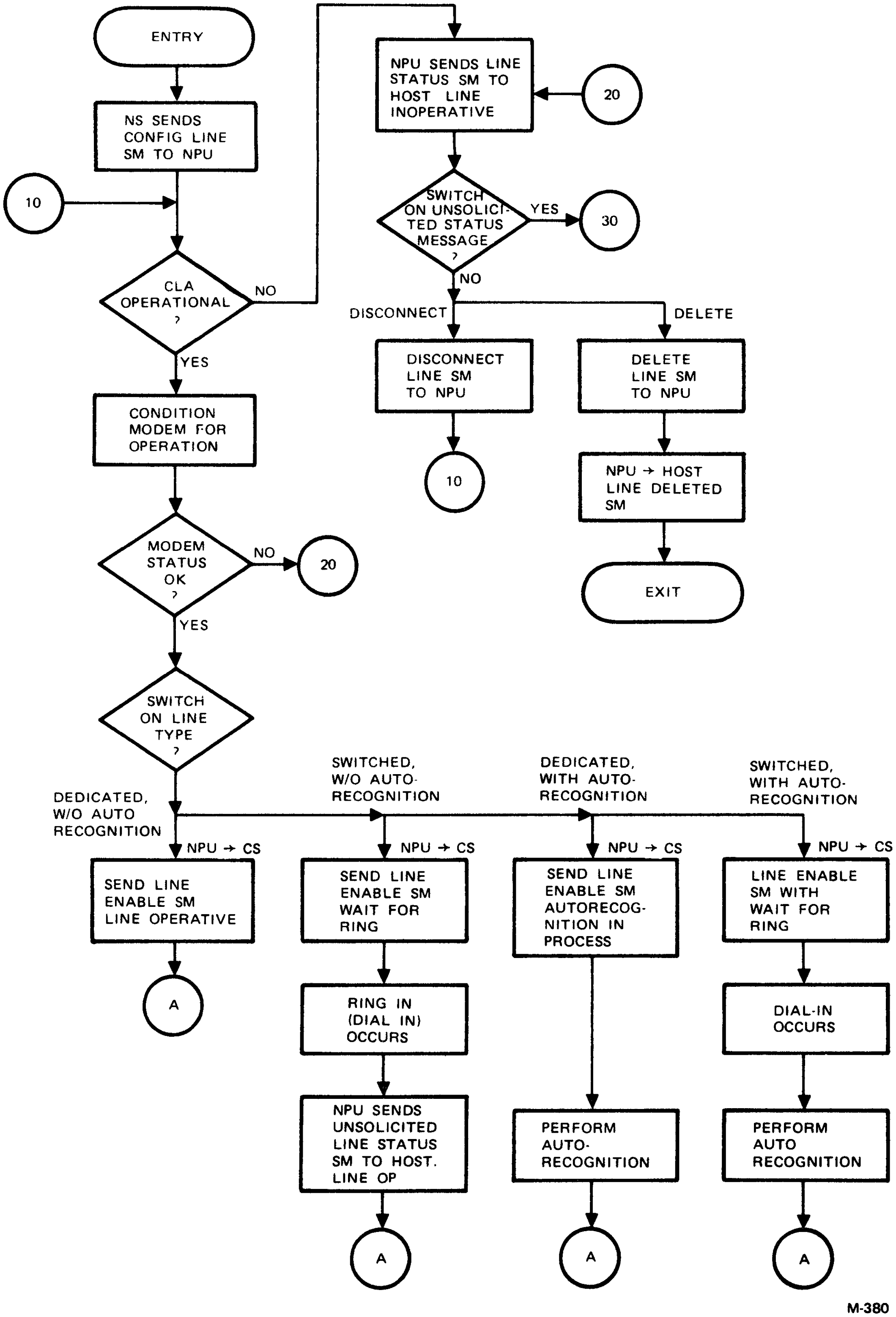


Figure 2-3. Line/Terminal Configuration Flowchart (Page 1 of 2)

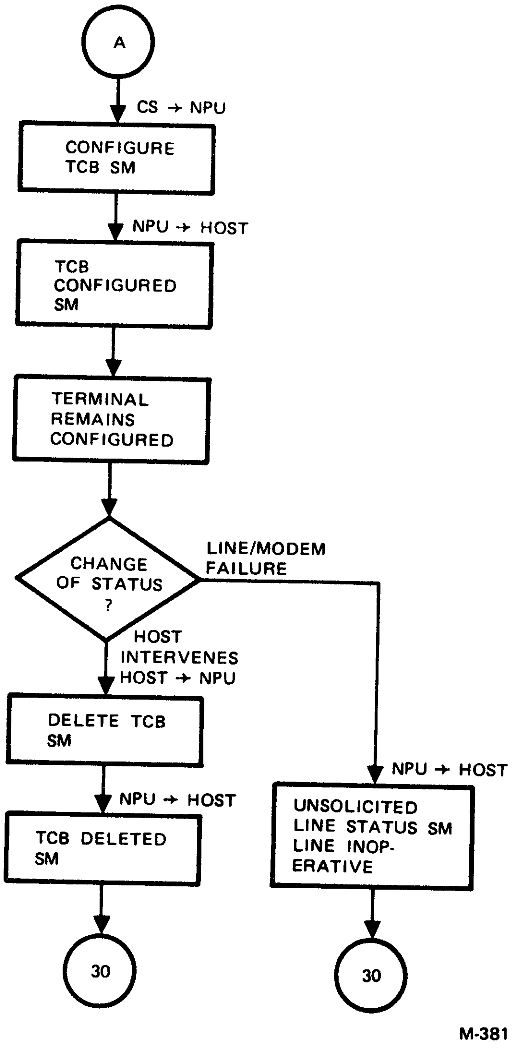


Figure 2-3. Line/Terminal Configuration Flowchart (Page 2 of 2)

After line is configured, it is automatically enabled by the service module. This allows the line to be monitored. Normal response is made using the enable line SM response message. When the line is reported operational, TCBs are configured. CS starts the line configuration process whenever an NPU is loaded and all links are configured; or a network operator enters a command generating a specific supervisory message in the host.

Configure Line SM

For each line to be configured, CS sends a configure line SM to the NPU connected to that terminal. All configure SMs contain a control block descriptor string (FN/FV). There is one such descriptor string for each type of configurable block in the NPU. The descriptor string equates a field number to a field position within the control block, and allows the associated field value to be entered into that field. Additionally, an optional action can be defined for the field number. The action allows such operations as validating the field value, assigning chains to other structures, and other actions appropriate to the newly entered field.

After performing the configuration defined by the control block descriptor string together with any defined actions, the service module attempts to enable the newly configured line. At the completion of the enable process, the line enabled response SM is returned.

The response message contains a reason code. If the response is normal, the code specifies either that the line is enabled and operational, or that the line is enabled but must wait for ring indicator/autorecognition results. If the response is an error type, the reason code specifies the type of error.

The four normal types of response messages correspond to the four major line types:

- Dedicated line, no autorecognition
- Switched line, no autorecognition
- Dedicated line, autorecognition
- Switched line, autorecognition

The response to configuration of a dedicated line is line enabled (1) if the modem of a dedicated line indicates data set ready, and (2) if (for a constant carrier) both clear to send and data carrier detect are on. Otherwise, line inoperative is reported.

Line operational is reported if autorecognition is not specified. A 30-second timer is started if autorecognition is specified. If no response is obtained within the 30 seconds the TIP responds with line not operational; the host then disconnects the line at the earliest opportunity. If a response is obtained, line operational is reported containing the results of autorecognition.

The response to configuration of a switched line is line enabled if a ring indicator is present. This normal response is generated immediately. Line enabled with no ring indicator is generated immediately if no ring indicator is present. This is followed by a line operational SM when a dial-in connection occurs. At this time, ring indicator is signalled and the NPU returns a data terminal ready to answer the call. If, when ring indicator is signalled, the host or logical link is not available, the NPU ignores the dial-in.

Autorecognition for switched lines is the same as for dedicated lines.

CONFIGURED LINE DELETION

The delete line SM changes the LCB status to not configured. CCP also deletes all TCBs for the line. The delete line SM is also treated as a positive response to an unsolicited line inoperative SM.

TERMINAL CONFIGURATION

When the line is operational, the host configures terminals for the line by issuing one or more configure terminal service messages. CCP responds to the configure terminal SM by generating the TCB. The amount of information in a TCB varies as a function of terminal or TIP type.

A TCB is built only when a line is enabled and operational. The block remains in existence until a delete terminal SM, a disconnected SM, or delete line SM is processed.

Terminals are identified in service messages by specifying the line, the hardware address, device type, terminal class, and host ordinal. Cluster and terminal address ranges are as follows (in hexadecimal):

	Cluster Address	Terminal Address
Mode 4A	70-7F	60
Mode 4C	70-7F	61-6F
ASYNC	0	0
HASP	0	1-7

The hardware address varies with the protocol being used by the terminal. Mode 4A can have one or more cluster controllers on a line but only a single console terminal on the cluster. Mode 4C can have one or more cluster controllers per line and one or more console terminals per cluster. The ASYNC TIP does not support any terminal addressing capability. The HASP TIP uses the terminal address as the stream number and does not use the cluster address. For HASP, the device type is combined with the terminal address to form the hardware identifier. Card readers and line printers use the full range of stream numbers, but plotters share the range with card punches.

A single line can have numerous terminals and therefore numerous TCBs. Each terminal has its own TCB and each TCB is usually established at the close of the initialization process.

Each terminal is configured with a host ordinal. The terminal host ordinal consists of a 4-bit integer value (0 through 15) and a toggle bit (2^4). The integer value is validated each time a service message is received for the terminal and is included in each service message sent to CS referencing the terminal. The toggle bit is validated each time a reconfigure TCB SM is received and must oppose the setting currently held in the TCB. The setting in the TCB is then reversed. This prevents inadvertent reinitialization of the block protocol on a logical connection in the event that a prior reconfigure TCB response SM was lost.

Configure Terminal SM

The configure terminal SM requires the service module to configure the TCB. Message parameters include terminal address, cluster address, device type, and the FN/FV pairs such as were defined for the configure line SM. The FV values are used in the specified fields of the TCB.

The service message is sent to the NPU by CS in the host either as the result of a line operational SM received and processed by CS, or as the result of an operator command to configure the terminal when the line has previously been reported as operational. As in the line configuration message, the FN/FV pair designates the field number and the value to be used in the field, and has an optional action associated with entering the field in the TCB. The SVM sets the fields in the TCB as directed.

A response SM is sent to CS indicating whether the fields were set or not.

TCB Reconfiguration

Terminals are reconfigured to establish or delete a logical connection number in an existing TCB, or to reinitialize the block protocol on an existing logical connection. This occurs when CS detects a need to establish or change a connection or modify other values in the TCB.

The format of the reconfigure terminal SM is the same as that given for the configure terminal SM except that the subfunction code (SFC) differs. The resulting operation in the NPU is the same except that the TCB should already exist. The TCB is modified as specified in the SM. The optional action is usually inhibited by the reconfigure TCB operation. The response formats are the same as those for the configure terminal SM.

The reconfigure terminal SM provides a general mechanism for CS to control terminals. Any action required coincident with the field change is also provided by the reconfiguration mechanism. If the toggle bit setting in the host ordinal byte does not change, an error response is generated. If the connection number is not zero, the block protocol is initialized or reinitialized on the connection.

TCB DELETION

When the operator requests that a terminal be deleted from the network, CS sends a delete terminal SM to delete the TCB and to clean up all table and data space associated with the TCB. CCP removes the connection from the logical connection directory. The service module responds to CS with a TCB deleted SM. CS is responsible for correctly deleting both ends of a connection.

Format of the delete terminal SM is the same as the configure terminal SM except the SFC code differs and there are no FN/FV pairs in the message. Normal response format is similar to that of the configure terminal SM response.

FAILURE, RECOVERY, AND DIAGNOSTICS 3

Failure and recovery of CCP depends on a number of factors:

- Host Failure - If a host fails, the NPU and its software stop message processing.
- NPU Failure - If an NPU fails, it must be reloaded and reinitiated from the host. Off-line diagnostic tests are useful during this period to help identify the cause of failure.
- Logical Link Failure - Host failure was mentioned above. Link CDCCP protocol failure leads to higher and higher levels of regulation until message traffic ceases on the link.
- Line Failure - Lines are disconnected and terminal control blocks associated with the lines are deleted.
- Terminal Failure - Terminal status is reported and message is discarded.

To aid recovery and to assure dependable network operations involving the CCP, three sets of diagnostic programs are available:

- In-line Diagnostics - These include CE error and alarm messages, statistics messages, halt code messages that specify the reason for an NPU failure, and off-line dumps.
- Optional on-line Diagnostics - These allow checking of circuits tso terminals. These aids are available only if a network maintenance contract is purchased.
- Off-line Diagnostics - These hardware tests for NPU circuits are described in detail in the Network Processor Unit Hardware Maintenance Manual.

HOST FAILURE

If the NPU fails to receive a coupler interrupt within 10 seconds, the NPU assumes a host failure and declares the host is unavailable (see HIP description, section 7). Host unavailability is communicated to the other end of all logical links (local or remote) by means of a disable trunk service message (SM). (However, the remote NPU does not allow its last trunk to be disabled - see section 8, LIP). The NPU also sends an informative SM to all connected interactive terminals.

HOST RECOVERY

After host recovery, the host assures that logical links are reinitialized and new connections are made.

The host recovers the existing configuration status by means of a status request SM to the NPU.

NOTE

All SMs are shown in appendix C of this manual.

The network repeats unsolicited line status changes that are not executed in the NPU. Most SMs sent to the network have a possibility of being rejected; in many cases the rejection code allows the network supervisor (NS)/ communications supervisor (CS) to determine the state of the line, device, or terminal that could not be configured.

NPU FAILURE

The host might not be aware of this condition, depending on its own state and availability of network paths. However, the peripheral processor unit (PPU) of the host has a 10-second deadman timer. If the PPU connected to a local NPU fails to receive an anticipated input or an idle response during this period, a timeout occurs. The host declares the NPU dead, and the NPU dump-and-load (or load only) operation is entered to start NPU recovery. Failure of a remote NPU is detected locally as a failure of the remote NPU to send data or idle blocks during a period longer than the timeout period. The local NPU informs NS of the inoperative local/remote trunk with an unsolicited trunk status SM, causing the host to dump and load the remote NPU through one of the local NPUs. See section 8 for a full description of the trunk protocol for detecting the failure and soliciting the loading of the remote NPU.

NPU RECOVERY

The host dumps (optional) and reloads an NPU after receiving a request for load. Stimulus for reloading comes from either the host PPU driver or the NPU bootstrap program. The reasons for requesting a load are as follows:

- Software failure caused PPU hardware deadman timer to expire.
- Hardware failure caused PPU deadman timer to expire.
- Trunk protocol failed between local and remote NPUs.
- Operator initiated a software halt, forcing reloading.
- Operator pressed MASTER CLEAR pushbutton on the NPU maintenance panel, causing a reload request.

The host does not request a dump after the second or subsequent reload attempt. After n successive attempts to load, the loading operation is aborted. The NPU is thereafter ignored until manually reactivated. After the NPU is successfully loaded and initialized, NS sets up all logical links for that NPU that the present state of the network allows. The methods of loading and initializing local and remote NPUs are described in the CCP 3.1 Reference Manual. NS reports the presence of each logical link that is to be established to CS. CS examines its configuration tables for elements that have been affected by the change in status. CS configures and enables lines supported by the NPU. For any line reported as operational, an examination of the configuration table reveals those terminals that can be connected. For each such terminal, both terminal and host support tables are configured and thereby connected.

HALT CODES AND DUMP INTERPRETATION

Unless NPU stoppage resulted from host failure or was initiated by operator action, some fault in the NPU caused the failure. If a dump is a normal part of the reloading cycle (and the network is normally set up that way), a dump is sent to the host. The CCP 3 Reference Manual describes the mechanics of transmitting the dump. Appendix B of that manual (Diagnostics) describes dump format and its interpretation with or without the use of halt codes.

LOGICAL LINK SUSPENSION

A logical link suspension is detected either by the local NPU determining that the channels to the host have been inactive or by an NPU detecting that the CDCCP protocol on the trunk supporting the logical link has failed. In the first case, the presumed host failure is communicated to the distant and local ends of all logical links. When a loss of ability to communicate is detected at the end of a logical link, all sources of data connected to that logical link are prohibited from accepting new data. If the host is the data source, a logical link regulation SM informs the host of the suspension of each logical link. Interactive terminals with connections on the logical link are informed of the suspension by an input stopped message.

LOGICAL LINK RECOVERY

A logical link either recovers spontaneously (e.g., return to service on a failed channel) or is reinitialized by host (NS) action. In the case of spontaneous recovery, the logical link protocol allows restart without loss of data. Otherwise, all logical connections are re-made and the terminal session restarts. Logical link recovery is described in detail in the CCP 3 Reference Manual.

TRUNK FAILURE

A failure of a trunk is detected by failure of the protocol as described in the LIP description (section 8). At this time, data in queue for the trunk is discarded. A trunk failure causes the NPU to report the failure of the logical link supported by the trunk. An unsolicited trunk status reply SM reports the failure.

TRUNK RECOVERY

Recovery of a trunk is detected by the trunk protocol using the LINIT elements of the trunk protocol (see sections 6 and 8). The logical link protocol determines when the trunk is used for data other than SMs to/from NS. Regulation of traffic on the trunk is discussed in detail in the CCP 3 Reference Manual.

LINE FAILURE

Line failure is detected by abnormal modem status or by line protocol failure. The change of status is reported to CS with an unsolicited line status reply SM. CS deletes all terminal control blocks (TCBs) supported by the line using the disconnect line SM.

LINE RECOVERY

A line cannot recover from a failure spontaneously. CS, which owns the lines, must first process the unsolicited status reply (line inoperative) SM by deleting the supported TCBs. CS then disables and reenables the line, using the appropriate SM. At this time, the TIP commences to check for a change. When the line status changes to operational, this is reported to CS with an unsolicited line status reply SM (line operational). When CS receives a message indicating that line status has changed to operational, CS attempts to configure the supported terminals.

TERMINAL FAILURE

Where the protocol is capable of determining terminal status, the protocol maintains records of such status. Terminal failure status is reported to CS for network management purposes. An unsolicited terminal status reply (terminal inoperative) SM reports the failure. The correspondent to which the terminal is logically connected is informed of the failure by the stop element of the block protocol (STP). This is discussed in section 6 (block types), Undeliverable traffic is discarded. The logical connection is not broken on terminal failure.

TERMINAL RECOVERY

When terminal failure is detected, possible terminal recovery is monitored. Typically, this is performed by a periodic status or diagnostic poll from the NPU to the terminal. Terminal recovery status is reported to CS with an unsolicited terminal status replay SM.

INLINE DIAGNOSTIC AIDS

Four types of inline diagnostic aids are provided with CCP:

- Alarm messages sent to the Network Operator (NOP). These messages alert the NOP that numerous hardware errors have occurred and that the engineering file in the host should be examined to find the NPU error history.

  NOTE

  If the user has purchased a network maintenance contract from CDC, the Hardware Performance Analyzer (HPA) in the host is the most convenient means of obtaining the contents of the engineering file. Otherwise, the user must devise his own method of analyzing the host engineering file.

- CE error SMs - These messages, which report individual hardware errors, are sent to the host engineering file. Such messages should be examined periodically.

- Statistics SMs - These messages are generated periodically for each NPU, line, and terminal. Statistics SMs are also generated when frequent errors cause the error counters for the device (statistic block counters) to overflow. All statistics SMs are sent to the host engineering file. These messages should be processed and displayed periodically.

- Halt messages, dumps, and dump interpretation - When the NPU stops, halt messages are sent to the NPU console. The message contains a code indicating the cause of the halt (a halt message indicates the NPU came to a soft stop; in a hard stop situation, the message cannot be generated) and the program in control when the halt command was generated. Dumps are part of the initialization process and are discussed in detail in appendix B of the CCP 3 Reference Manual. Note that the halt message is delivered using PBQUICKIO; the message does not use a SM.

Format of the SMs used to generate alarm, CE error, and statistics messages are given in appendix C. The basic format of all three SMs is shown in figure 3-1.

1	2	3	4	5	6	7
DN	SN	CN	BT	PFC	SFC	Data (one or more bytes)

DN - Destination node

SN - Source node, the originating NPU

CN - Connection number, 00 = services messages

BT - Block type, 04 = CMD (see section 6)

PFC - Primary function code

0A - CE Error or Alarm

07 - Statistics

SFC - Secondary function code

00 - CE error message }
01 - Alarm message } with PFC = 0A

00 - NPU statistics }
01 - Trunk/line statistics } with PFC = 07
02 - Terminal statistics }

DATA - see table 3-1.

Figure 3-1. Format of Alarm, CE Error, and Statistics Messages

TABLE 3-1. INLINE DIAGNOSTIC SERVICE MESSAGES

Message	PFC	SFC	Data Bytes
CE Error	0A	00	First: Error Code (EC)[†] Subsequent: data (if any) - up to 27 bytes
Alarm	0A	01	Message text
NPU Statistics	07	00	Error words 1 thru 11; 2 bytes per word[†]
Trunk/Line Statistics	07	01	First: P - port } from local NPU Second: 00 } to line/trunk Third: 00 - host ordinal Fourth: LRN - link remote node Subsequent: explanation words 1 thru 4; 2 bytes per word[†]
Terminal Statistics	07	02	First two bytes: P/00 as for trunk/line statistics Fourth: CA - cluster address } see appendix C Fifth: TA - terminal address } for values Sixth: DT - device type } Seventh: CN - connection number Subsequent: explanation words 1-3; 2 bytes per word[†]

[†] Refer to appendix B of CCP 3 Reference Manual for details.

ALARM MESSAGES

For each alarm sent, a previous series of messages (CE errors) has generated entries in the host engineering file for this device. These messages are used to determine the cause of the failure and to perform maintenance to correct the failure. See CE error codes portion of appendix B of the CCP 3 Reference Manual.

At the network operator's console, the alarm SM appears as follows:

```
FROM NPU xx/RESIDENT...(text)
```

Currently, three alarm SM texts can be generated (text is the 50 characters allowed for the SM text):

MAINTENANCE ALARM PORT xx (0 xx FF$_{16}$)

MAINTENANCE ALARM MLIA

MAINTENANCE ALARM COUPLER

Within an NPU, a group of counters is maintained in the statistics block for each hardware device. Each time a CE error SM is sent, its associated statistics counter is incremented. Periodically, each counter is compared to a threshold value. Whenever a threshold value is exceeded, an alarm SM is sent to the NOP. If a threshold is not exceeded at the periodic check time, the counter resets to zero. Threshold value is a CCP build-time variable. The suggested period is 15 minutes. To prevent multiple alarm messages for the same condition, the following alarm SM restrictions are provided:

- Lines and trunks - Only one alarm is sent after the line is enabled. A subsequent disable/enable sequence allows another alarm to be sent.
- Coupler - Only one alarm SM can be sent per NPU load.
- MLIA - Only one alarm SM can be sent per NPU load.

CE ERROR MESSAGES

This category of diagnostic service message reports the occurrence of hardware-related abnormalities. This includes all NPU-related hardware (coupler, MLIA, loop multiplexers, CLAs), and (indirectly) all connected hardware: modems, lines, and terminals. The creation of the service message is separate from and in addition to the statistics accumulated in the NPU and periodically dumped to the host.

To prevent swamping the NPU or host with error messages when an oscillatory condition arises, an error counter is incremented with each error message generated. When the counter reaches the limit specified at build time, the event is discarded rather than recorded. The counter is periodically reset to zero. This period is another system build-time parameter.

Six types of CE error messages are used. The types and text portion of the messages are in appendix B of the CCP 3 Reference Manual.

STATISTICS MESSAGES

Three forms of statistics messages are used: NPU statistics, line statistics, and terminal statistics. Each type is sent upline to the host engineering file. The host does not reply to statistics messages.

Statistics data is placed in the statistics block for the appropriate device (NPU, line, or terminal) by a call to PNSGATH. The call comes from either a TIP (via the postinput or postoutput POI) or from a LIP. The HIP places statistics information in the NPU statistics block directly. The statistics information for NPU and terminals is kept in the TCB for the terminal (NPU has its own TCB). Statistics information for lines is kept in the LCB for the line.

One stimulus for a statistics report is a request form the time module PBTIMAL. The period for this timeout is a system build-time parameter. PNSGATH handles the periodic request. Two other stimuli cause PNDSTATS to generate the message: one stimulus arises when any one of the counters that keep the statistics overflow. In that case, the message for the NPU, line, or terminal is immediately generated. The other stimulus arises when a line disconnect SM, a delete line SM, or delete terminal SM is received by the NPU. The affected line and/or terminal statistics blocks are dumped and the appropriate statistics SM is sent before the normal response SM is sent. When any statistics message is sent upline, the statistics counters in that statistics block of the TCB or LCB are cleared.

The search by PNSGATH for periodic statistics is conducted as follows: The search cycle begins at the permanently assigned TCB for the NPU. The statistics from this TCB are dumped if any are available. The next search is set to begin at the first active LCB. If no NPU statistics are available, the current search moves to the first active LCB. These statistics are dumped, if available. The next search is set to begin at the first TCB attached to this LCB. If the LCB has no statistics available, the search moves to the first TCB. Its statistics are dumped, if available. The next search is set to begin at the next TCB for this line. This continues until all the TCBs for the first active line are checked. Then, the second active line and all its TCBs are checked. This continues until all TCBs and all active lines are checked. The next cycle again starts with the NPU TCB.

BASE SYSTEM SOFTWARE 4

The support software can be divided into three categories: the base system, the multiplex subsystem (technically a part of the base system), and the network communications software. This section describes the support software for the base system only. The HIP and the LIP can be considered as support programs for the TIPs.

The functional grouping of support tasks is as follows:

- Base system - Operating system functions (program execution, buffer (space) allocation, interrupt handling), timing support, data structures support. NPU console handling is also described in this grouping.
- Multiplex subsystem - drivers for the multiplexer I/O lines.
- Network communications software - message routing, command interpretation (the service module), common TIP support routines (including statistics gathering, CE error messages to the host, and regulation assistance).

The major base subsystem components are the following:

- Monitor, also called OPS monitor
- Space (buffer) allocation
- Timing services
- Direct program calls
- Indirect (worklist-driven) program calls
- Interrupt handling
- Directory maintenance
- Global structures
- Standard code and arithmetic support routines

SYSTEM MONITOR

The NPU is a multiple-interrupt-level processor. Interrupts are serviced in a priority scheme in which all lower priority interrupts are disabled during execution of a program that is operating at a higher priority level. When no interrupt is being processed, the NPU runs at its lowest priority, known as the operations (OPS) monitor level. (Refer to interrupt lines/priorities in appendix H.)

NOTE

> This priority is not to be confused with the regulation level priority for trunks (discussed in the CCP 3 Reference Manual) nor with the host interface priorities (discussed as a part of the HIP).

The system monitor (PBMON) controls allocation of time to programs running at the OPS level. The monitor gives control to a program by scanning the table by worklist control block (WLCB) that defines the OPS level programs that can be called with a worklist. Control is released to the first program encountered with a queued worklist waiting to be serviced.

Scanning starts at entry 8 of the table (table 4-1) and continues until the first program is encountered with a worklist attached (figure 4-1). The monitor then determines whether the program can be called with more than one worklist (N > 1). Worklist control block BYLISTCB contains parameter BYMAXCNT that defines the number of worklist entries to be processed by the OPS-level program in one pass. If N is greater than 1, the program is given control successively until either all the worklists for that program are serviced or until the maximum number of consecutive executions for that program has been reached. If N is 1, the scan pointer moves to the next entry each time the program is executed, even though there may be more worklists attached to this program's queue.

The scan pointer automatically recycles to the B0CHWL entry when B0DUMMY is reached. If new worklist-driven OPS-level programs are added to the list, they precede B0DUMMY. A worklist must be established to drive the new program.

Each time a program completes, PBMON initializes a timer (BTTIMER). This timer is advanced and checked by the interrupt level timer routine (PBTIMER) at specific system-defined intervals. If the timer expires, it indicates that an OPS-level program has been abnormally delayed. PBMON execution then terminates and a call to PBHALT is made. This is called an OPS timeout condition.

BUFFER HANDLING

This function allocates any of the four types of buffers (each type has its own free buffer pool) and returns buffers to the appropriate free buffer pool when users are finished with the buffers. As an option, the function also stamps buffers to keep a record of the buffer's usage and the address of the program requesting the buffer.

Standard buffers are also assigned for the following:

- Data buffer for special TIP application
- Integer overlay
- Buffer chaining overlay
- Terminal control blocks (TCBs)
- Physical I/O request packets
- Active ASYNC LCB list
- Statistics (NPU, line, or terminal)
- Type 1 table entries
- Type 4 table entries
- Timeout buffers
- Diagnostic control block (DCB)
- Multiplex line control block (MLCB) and text processing control block (TPCB)
- Special application flags

Figure 4-2 indicates the types of buffers assigned. Each buffer type has its own field definitions. The figure also shows the stamping techniques.

TABLE 4-1. OPS MONITOR TABLE

	Table Entries	Entry No.	Program	No. Entries	Calls Program	WLG Size (Word)
BYWLCB	B0FSWL	1 2 3 4 5 6 7	These entries not serviced by monitor. Reserved for generating worklists			
...to here	B0CHWL	8	Console	1	PBCONSOLE	2
	B0INWL	9	Internal processing	1	PBINTPROC	2
	B0MLWL	10	MLIA interrupt handler	10	PBMLIAOPS	5
Current	B0SMWL	11	Service module (SVM)	2	PNSMWL	4
pointer	B0TIWL	12	Timing services	1	PBTIMAL	1
position	B0TYWD	13	TIP debug	1	PBTIPDBG	6
	B0LIWL	14	Line initializer	1	PTLINIT	3
	B0DGWL	15	(On-line diagnostics)	0	------	-
	B0COWL	16	HIP	1	PTHIPOPS	3
	B0HLDC	17	LIP	1	PLTKOPS	3
	B0M4WL	18	Mode 4 TIP	1	PTMD4TIP	3
	B0ASYNC	19	ASYNC TIP	1	PTASNOPS	3
	B0HASP	20	HASP TIP	1	PTHSOPSTIP	3
Monitor	B027WL	21	Reserved	0	------	-
pointer	B0HHWL	22	Reserved	0	------	-
recycles...	B0DUMMY	23	Dummy for console; recycles to entry 8	0	------	-

Word	15	14 8	7 0
0	*	BYCNT (count)	
1	Put Pointer		
2	Get Pointer		
3	BYWLINDEX First entry index		BYINC Not used
4	Not used		
5	**	BYMAXCNT	BYPAGE
6	BYPRADDR		

* Multi-WLCB flag

** BYWLREQ, worklist required flag

BYCNT - number of entries in the worklist queue

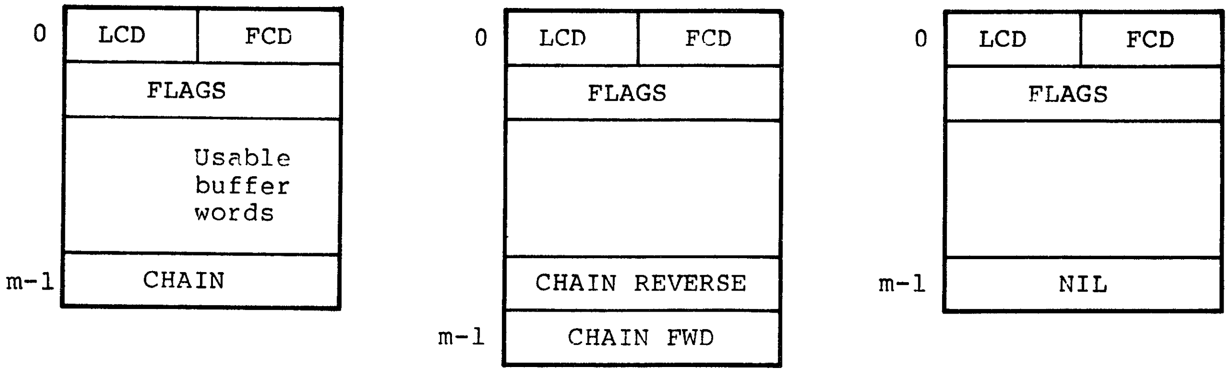
BYMAXCNT - number of entries to process in one pass

BYPAGE - program page address

PYPRADDR - program address

BYWLINDEX - WLCB index

Figure 4-1. OPS Monitor Table Format



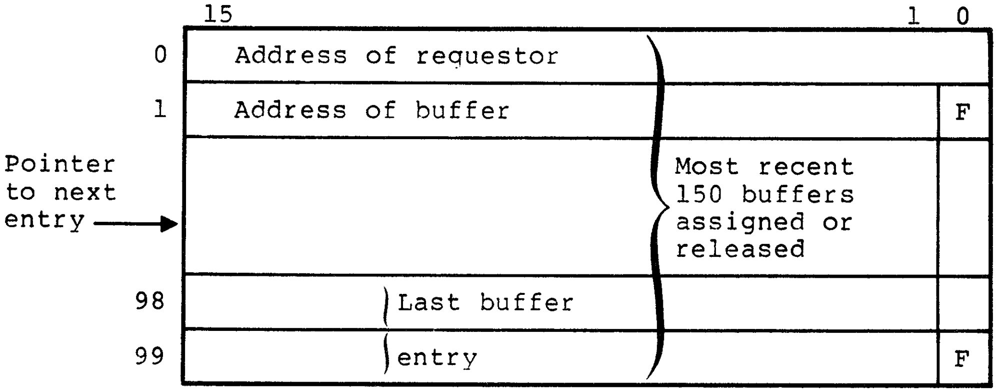
Buffer of size m

LCD - last character displacement
FCD - first character displacement
FLAGS - end indications, transparent text, queuing, etc.

Buffer before assignment. Chains of free buffers both forward and reverse

Buffer after assignment. No chain, but word m-1 reserved for chaining

Buffer Stamping area*



* Circular buffer, two words/entry

F status flag
0 = put
1 = get

Figure 4-2. Buffer Formats and Stamping

Buffer splitting continues until enough buffers of the size needed are made available from progressively larger buffer pools or until all possible buffer splits have been made from all larger buffer pools and not enough buffers are available.

When testing buffer availability against a specified threshold number, buffer maintenance attempts to adjust distribution of buffer sizes by using buffer mating or buffer splitting to replenish buffer pools that are below the threshold level. If buffer cannot be made available, the buffer requester is notified that the requested operation cannot occur for lack of buffers. Buffer mating is the converse of buffer splitting.

Buffers are potentially available in six sizes: 4, 8, 16, 32, 64, and 128 words. At installation time, the user chooses any four contiguous sizes; for instance, 8, 16, 32, and 64 words.

In the standard system, buffers are assigned in following sizes, for the uses indicated:

- 8 words - timing
- 16 words - MLCB and WLCB
- 32 words - TCB and TPCB
- 64 words - data

Buffers are assigned from a buffer pool of the appropriate size and are assigned one at a time; buffers can be released singly or in a chain of buffers. Buffers are released to the buffer pool from which they were originally drawn.

Buffer stamping is available as a build-time option. If this option is selected, a buffer stamping area is reserved to save diagnostic information on the assignment and release of buffers. The circular stamping buffer, 100 words long, can save information on the most recent 50 buffer assignments/releases. Each two-word entry consists of the address of the routine that requested the assignment/release, and the address of the buffer. A flag in each entry indicates whether the buffer is currently assigned or in a free buffer pool. Information concerning the use and location of the buffer stamp area and the pointer to the next entry to be used is found in appendix H, the buffer subsection.

OBTAINING A SINGLE BUFFER

The calling sequence to obtain a single buffer of a specified size is

```
PBGET1BF (parm)
```

Parm is the address of the pointer to the buffer control block. PBGET1BF is a PASCAL function and returns the value of BOBUFPTR that points to the base address of the buffer obtained. PBGET1BF also uses the buffer control block for the specified size buffer. The chain word and flag word of the newly assigned buffer is cleared and the LCD/FCD are set to their initial values.

Interrupts are inhibited during execution. A system halt occurs if the buffer pool is down to the last buffer and there are no buffers in larger-sized pools available to be split. A halt occurs if the next buffer has a bad chain address.

RELEASING A BUFFER

The following calling sequences are used, respectively, to release a single buffer or a specified size to release one or more buffers of a specified size, or to release a chain of buffers. After checking for no buffers, the system returns the released buffer to the free pool of other same-sized buffers. The buffer handler also ensures that the address is a valid buffer address and determines if the buffer has already been released to the free buffer pool. Contents of released buffers are not altered except for chain words.

Releasing a Single Buffer

The calling sequence to release a single buffer is

```
PBREL1BF (parm1, parm2)
```

Parm1 is a pointer to any address within any word of the buffer to be released and parm1 is the address of the pointer to the buffer control block. Parm1 is a PASCAL VAR parameter that is altered by the procedure so that, upon completion, parm1 contains the chain value of the last buffer released.

Releasing Several Buffers

Two methods are available to do this. The first method requires a pointer to the first buffer in the chain to be released. The second method will not return an error indication if the buffer address is zero. In both cases, the release mechanism is actually performed by firmware. The two methods are called by PBRELCHAIN (parm1, parm2) and PBRELZRO (parm1, parm2).

In both cases, parm1 designates a pointer to the first buffer in the chain to be released and parm2 designates (indirectly) the address of the buffer pool to which the buffers will be returned. If parm1 for PBRELZRO is zero, no action is taken.

TESTING BUFFER AVAILABILITY

The calling sequence to test buffer availability is

```
PBBFAVAIL (parm1, parm2, parm3)
```

PARM1 specifies the number of buffers required, parm2 pointer specifies the buffer control block required, and parm3 specifies the total free space threshold. PBBFAVAIL is a PASCAL function; it returns a true value if the test indicates that sufficient buffers are available. This calling sequence can be used at any interrupt level.

BUFFER COPYING

The BBCOPYBFRS routine allows copying data from a chain of any type of buffers to a chain of data buffers. The call is

```
PBCOPYBFRS (parm rcd).
```

The parameter record (parm rcd) requires the following:

- The number of source buffers to copy
- Source buffer size
- Data buffer size
- A release flag

The source chain can be released after the copying operation.

OTHER BUFFER HANDLING ROUTINES

PBDLTXT deletes data from a buffer by advancing the first character displacement (FCD) pointer in the buffer header. See figure 4-2. PBSTRIP returns the empty buffers to the free buffer pool of the appropriate size.

TIMING SERVICES

Timing services provide the means for running those programs or functions which are executed periodically or following a specific lapse of time. Seven timing services are available:

- A firmware program handles the 3.33 ms microinterrupt to provide a 100-ms timing interval. This real-time clock interrupt is handled by PBTIMER. PBCLKINIT restarts the real-time clock following the interrupt.

- Every 100 ms, PBTIMER calls PBTOSRCH to search the chain of time-lapsed buffer entries. These entries are assigned as needed in response to calls from any module. If an entry's time period elapses, and if the release flag for that entry is set, the entry is deleted from the chain. In all cases, a worklist call is made to the program which requested the delayed call. Timing services uses PBTOQUE to add entries to this chain of delayed calls.

- Every 500 ms, PBTIMER checks the deadman timer. The timer is reset and the timer monitor routine is executed. If the deadman timer expires, the monitor has spent too much time in one OPS-level program. The NPU stops.

- Every 100 ms, PTMSCAN (a part of the ASYNC TIP) scans the list of active line control blocks (LCBs) for asynchronous terminals. If a character is received, the timeout is set for the next character. If no character has been received during the 100-ms period, a timeout is declared, the LCB is removed from the list of active LCBs, and the ASYNC TIP is notified by means of a worklist.

- Every second, a timing routine checks all active output lines to find whether an output data demand (ODD) interrupt has been generated for the next character to output. If one second has passed with no new ODD interrupt, the multiplex subsystem worklist processor is called to declare a hardware failure for the line.

- A time-of-day routine, PBTIMEOFDAY, is called every second. The time of day is incremented and, if necessary, recycled to the start of day time (00 hour, 00 minute, 00 second).

- Every 500 ms, PBLCBTMSCAN scans all active lines for periodic requests. If a line's period for a specific request has elapsed, the appropriate TIP is called, using a worklist entry. Input or output is terminated for the line if this is requested. Inactive LCBs are unchained from the set of active LCBs. Timer services provides the means for chaining LCBs to this list of LCBs that require periodic action.

DIRECT CALLS

Most OPS-level programs call other programs directly for performing minor tasks. A few major task calls use indirect (worklist) calls. For direct calls, the last program in the calling chain is usually PBCALL. It is used for direct calls among OPS-level programs, for transferring between programs on different pages, for timed or periodic calls, for service message switching, for overlay execution, and by PBMON when that program places a program into execution.

PBCALL calls a procedure from PASCAL by address, rather than by name. Unlike other procedure calls, PBCALL can pass a variable number of parameters, corresponding to the number of parameters expected by the calling procedure. Example:

```
type pgms = (pgml...pgmn);
var    table:  array {pgms} of integer:
       index:  pgms;
addr   ( {programl} , table {pgml});
.
.
.
addr   ( {programn} , table {pgmn});
.
.
.
 {set up index}
PBCALL (table {index});   {call program, no parameters}
```

The PBCALL calling sequence is

```
PBCALL (addr, parml,...parmn)
```

addr is the address of the program to be called and parml through parmn are optional and are parameters passed to the called program as shown:

```
procedure PBCALL;

begin
(store return address in called procedures entry point)
(jump to procedure)

end;
```

Other switching programs of importance are as follows:

- PBPAGE (parml) switches control directly from one OPS-level program to another. Parml is a worklist index to OPS PROGRAMS SET INTO AN INTERMEDIATE ARRAY.

- PBXFER (parm1, parm2) transfers control to a program that may be on another page of main memory. Parm1 is the called program's address and parm2 is the dynamic page register base address. Both are global variables.

- PBTIMAL (parm) controls all time-dependent OPS-level programs. Parm is the array of time dependent programs (CBTIMTBL).

WORKLIST SERVICES

Worklists provide a convenient method to handle communications between software modules that do not use direct calls. Figure 4-3 depicts the worklist organization. The list services function manipulates worklists with variable entry sizes. Functions provided by list services include the following:

- Make (PUT) worklist entries from any priority level (including OPS level).

- Make OPS-level worklist entries by terminal type.

- Extract (GET) an entry from a list.

Characteristics of lists managed by list services are as follows:

- First in, first out.

- Entries may be from one to six words in length, but all entries in a particular list must be the same length.

- Lists are maintained in dynamically assigned space.

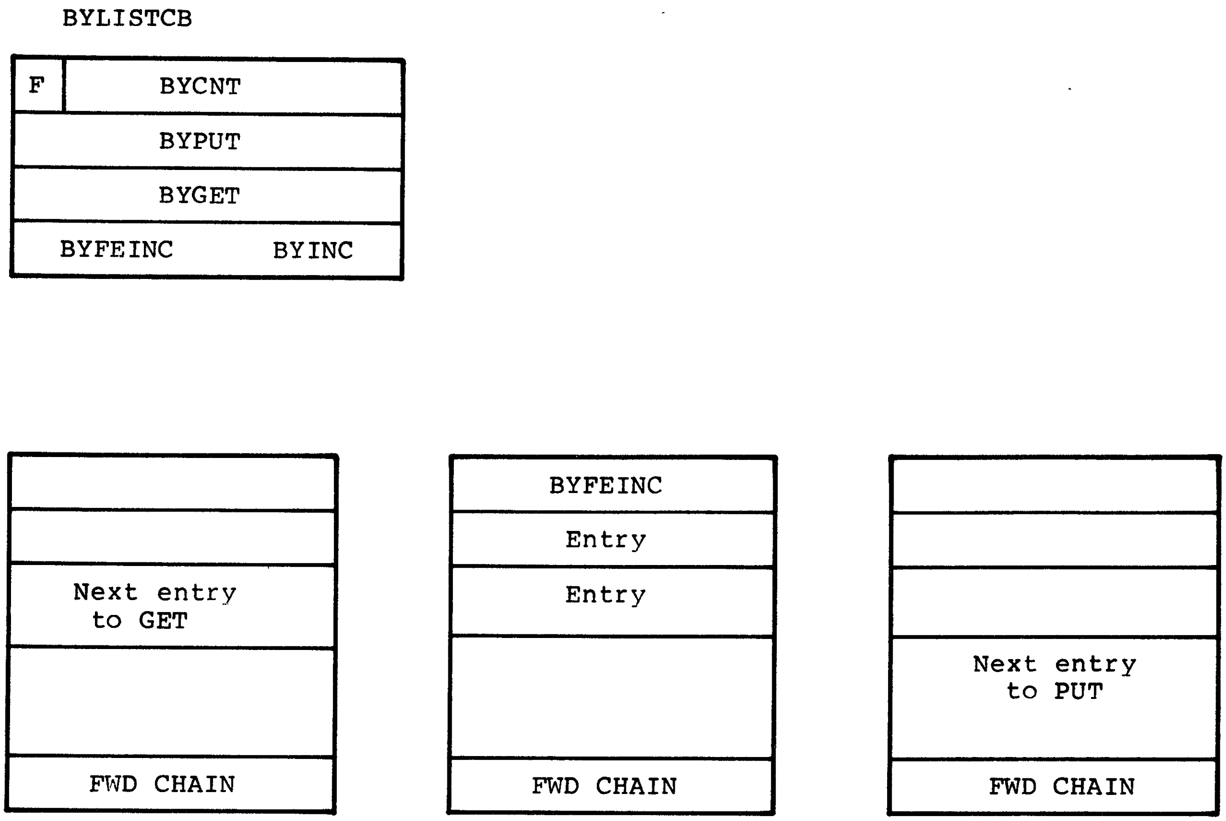
- There is no maximum on the number of entries in a list or on the number of lists serviced.

Contention between priority interrupt levels is resolved by defining an intermediate worklist array (BWWLENTRY) with 6-word entries for each possible system interrupt level. Worklist entry parameters are assembled and extracted in the intermediate worklist area corresponding to their interrupt level. (A user can design his own programs to perform this function, however.)

A worklist entry is passed to PBLSPUT and data is normally obtained from PBLSGET through a global array named BWWLENTRY. Each element of the array has a variant record structure consisting of one case for each logical entry structure. When each new worklist-driven program is created, the format of the new worklist is added as another case to the PASCAL-type definition BOWKLSTS. Thus, each worklist has unique fields and names.

There are 17 elements to the array BWWLENTRY, one for each priority interrupt level. To access the proper interrupt level, the global variable LEVELNO is used. For example, to access a field of a particular worklist entry at the proper interrupt level, the following expression is used:

```
BWWLENTRY [LEVELNO]. FIELDNAME
```



F - Not used

BYCNT - Entry count

BYINC - Entry size (uniform in any one worklist)

BYFEINC - Displacement in buffer to first entry

Figure 4-3. Worklist Organization

The fields of the worklist entry are accessed to store information before calling PBLSPUT or to obtain information after calling PBLSGET. For programs that always run at a specific interrupt (e.g., OPS, CPL, and RTC), constants can be used to increase efficiency.

If a program using PBLSPUT or PBLSGET calls a program also using PBLSPUT or PBLSGET, information in the worklist entry BWWLENTRY might be changed upon return. In such cases, one of the following techniques must be used to ensure proper data integrity:

- Put all information in the worklist entry and call PBLSPUT before calling the second program.
- Call PBLSGET and access all pertinent information from the worklist entry before calling the second program.
- Save and restore the worklist entry from BWWLENTRY.

MAKING A WORKLIST ENTRY

PBLSPUT puts an entry into a worklist from any interrupt priority level. The calling sequence is

```
PBLSPUT (parml, parm2)
```

Parml is the address of the worklist entry and parm2 is the address of the proper worklist control block.

PBPUTYP makes a worklist entry after calculating the worklist index from the line number. Firmware makes the actual worklist entry. Format of the call is

```
PBPUTYP (parm)
```

Parm is the entry to be made, either in an intermediate array or in a local save area.

NOTE

The second word of the entry is always a line number.

Two other important worklist entry builders are actually a part of network supervision.

- PBTWLE parm - This makes a worklist entry for the specified terminal control block (TCB). The parm is the work code. The entry made contains the line number and the TCB pointer. PBPUTYP moves the entry from the intermediate array to the worklist.
- PBSWLE - This makes a worklist entry for SWITCH, the procedure used for switching. PBSWLE puts the pointer to the block to be switched in a worklist entry for PRINTPRC. That routine calls SWITCH. PBLSPUT moves the entry from the intermediate array to PBINTPRC's worklist.

EXTRACTING A WORKLIST ENTRY

The PBLSGET routine moves entries from a worklist to an intermediate array (BWWLENTRY). The routine is available at all priority interrupt levels. A special firmware sequence speeds up execution and eliminates contention between software and firmware. Format of the call is

```
        PBLSGET (parml, parm2)
```

Parml is the address of the worklist entry and parm2 is the address of the worklist control block. If the list is not empty, the next entry is moved into the specified worklist area.

BASIC INTERRUPT PROCESSING

The two types of interrupts that are processed are the macrointerrupts and the microinterrupts.

MACROINTERRUPTS

The interrupt mask register is set by an interregister command and the interrupt system is activated by the enable interrupt command. Upon recognizing an interrupt, the hardware automatically stores the appropriate program return address in a storage location reserved for the activated interrupt state. This ensures that the software returns to the interrupted program after interrupt processing.

With the return address stored, the hardware deactivates the interrupt system and transfers tcontrol to an interrupt handler program that begins at the address specified for that interrupt state. The program thus entered stores all registers (including the interrupt mask register and overflow) in addresses reserved for the interrupt state. The interrupt mask register is then loaded with a mask to be used while in this interrupt state, with a one in the bit position indicating interrupt lines with higher priority than the interrupt state being processed. The program then saves the current software priority level, sets the new software level, activates the interrupt system, and processes the interrupt.

During such interrupt processing, an interrupt line with higher priority may interrupt. However, such interrupts also cause storage of return address links to permit sequential interrupt processing according to priority level with eventual return through the return addresses to the mainstream computer program.

When processing is completed at that level, the computer exits from an interrupt state by inhibiting interrupts, restoring registers to their pre-interrupt states, and executing the exit interrupt state command (EXI). This command retrieves the return address stored when the interrupt state was entered. Control is transferred to the return address and the interrupt system is again activated.

Interrupt Priority

Interrupt priority is under control of the computer program. Priority is established by an interrupt mask for each interrupt state that enables all higher priority interrupts and disables all lower priority interrupts. When an interrupt state is entered, the mask for that state is placed in the mask register. Bit 0 of the mask register corresponds to interrupt state 00, bit 1 corresponds to interrupt state 01, etc. A bit that is set means that the corresponding interrupt state has a higher priority than the interrupt state to which the mask belongs. Thus, there can be as many as 17 levels of priority.

NOTE

> Priority of any interrupt state can be changed during program execution.

Standard subroutines are provided for servicing the interrupt mask. These subroutines are as follows:

- Set Interrupt Mask
- Reload Interrupt Mask
- Perform a logical AND with the mask
- Perform a logical OR with the mask

PBSMASK - SET INTERRUPT MASK

This routine loads a specified interrupt mask value into the M register to become the new interrupt mask. The calling sequence is

```
PBSMASK (parm)
```

Parm is a value parameter specifying the new interrupt mask value to be loaded into the M register. The resultant mask becomes the new mask value in the M register.

PBAMASK - AND INTERRUPT MASK (AND PBLMASK)

PBAMASK, in conjunction with PBLMASK, is used to selectively disable and enable one or more software interrupt levels. The calling sequence is

```
PBAMASK (parm)
```

Parm is a value parameter specifying the value to be logically ANDed with the current interrupt mask.

PBOMASK - OR INTERRUPT MASK

PBOMASK employs a logical OR function to combine a given interrupt mask with the current mask in the M register, the result becoming the new interrupt mask value in the M register. The calling sequence is

```
PBOMASK (parm)
```

Parm is a value parameter specifying the mask value to OR with the current interrupt mask.

User Interface

Because each interrupt handler is an independent program, there are no specific user interfaces. However, pertinent information is necessary to enable modification of, and additions to, the interrupt handlers.

An array contains interrupt masks for the 16 interrupt states. To access a particular interrupt mask, use the interrupt state number as an index. LEVELNO is the global variable where the current software priority level is saved.

Table 4-2 lists the 16 interrupt states, gives the value for the delta field for its exit instruction, the storage location for its return address, and the location of the first instruction of the interrupt handler program. Current interrupt assignments and their associated software priority are listed in table 4-3. The seventeenth state (no interrupt line associated) is the OPS level.

TABLE 4-2. INTERRUPT STATE DEFINITIONS (PBINTRAPS)

Interrupt State	Exit Instruction Delta Field Value	Location of of Return Address	Location of First Instruction of Interrupt Handler Program
00	00	0100	0101
01	04	0104	0105
02	08	0108	0109
03	0C	010C	010D
04	10	0110	0111
05	14	0114	0115
06	18	0118	0119
07	1C	011C	011D
08	20	0120	0121
09	24	0124	0125
10	28	0128	0129
11	2C	012C	012D
12	30	0130	0131
13	34	0134	0135
14	38	0138	0139
15	3C	013C	013D

TABLE 4-3. INTERRUPT ASSIGNMENTS

Interrupt Line	Software Priority	Interrupt Description	Handler Name
0	P1	Memory parity, program protect, power failure, software breakpoint	PBLN00
1	P6	NPU console	PBLN01
2	P2	Multiplex loop error (MLIA)	PBLN02
3	P3	Multiplex subsystem - Level 2	PBLN03
4			
5	P7	Coupler 2	PBLN05
6	P7	Coupler 1	PBLN06
7	P8	Spare	
8	P9	Real-time clock	PBLN08
10	P11	Spare	
11	P12	Spare	
12	P13	ODD input parallel	PBLN0C
13	P14	Input line frame received (MLIA)	PBLN0D
15	---	Macro breakpoint	PBLN0F

MICROINTERRUPTS

Three microinterrupts are also serviced.

- The output data processor processes the output data demand (ODD) interrupt that each communications line adapter generates to indicate that it is ready to output another character. The output data processor (part of the multiplex subsystem) gets the next character from the appropriate line-oriented output buffer and puts the character on the output loop. The requesting communications line adapter picks the character from the loop and transmits it.
- The input data processor processes the interrupt produced when the entry of either a data character or communications line adapter status into the circular input buffer is completed. The input data processor (also part of the multiplex subsystem) gets the next character from the appropriate line-oriented output buffer and puts the character on the output loop. The requesting communications line adapter picks the character from the loop and transmits it.

- The input data processor processes the interrupt producued when the entry of either a data character or communicatons line adapter status into the circular input buffer is completed. The input data processor (also part of the multiplex subsystem) uses the designated input state program to demultiplex the character into the appropriate line-oriented input buffer.

- The timing services firmware processes the 3.3-millisecond clock interrupt, which is used as the time base for all timed NPU functions.

PASCAL GLOBALS

CCP provides a number of PASCAL globals, frequently in the form of fields embedded in tables. Appendix J shows the tabular form of the principal data structures and describes the fields. A complete listing of the CCP PASCAL globals is in an MPEDIT listing.

STANDARD SUBROUTINES

Standard subroutines are a miscellaneous group of support routines which perform the following tasks.

- Convert and handle numbers
- Maintain paging registers
- Perform block functions
- Set or clear protect bit
- Perform miscellaneous other tasks

Table 4-4 lists these standard subroutines. Some of these frequently used routines are written in macroassembly language rather than in PASCAL.

CALLING MACROASSEMBLY LANGUAGE PROGRAMS FROM PASCAL PROGRAMS

A procedure call to a macroassembly source code program from a PASCAL-coded program is the same as a call to any other PASCAL program. The same calling sequence code is generated, that is:

```
RTJ     program
ADC     parml
 .       .
 .       .
 .       .
ADC     parmn
```

A macroassembly program handles parameters as PASCAL parameters. To treat a parameter as a value parameter, the user loads the contents of the parameter and stores it locally and then passes the address of the store location to the called program. To treat a parameter as a variable parameter, the user loads the address of the parameter and uses this as a pointer. Packed record parameters that are fields less than full word length are unpacked into a temporary word and the address of the temporary word is passed to the called program.

TABLE 4-4. STANDARD SUBROUTINES

Subroutine Name	Description	Type**	Language*	Type Checking Defeated
PBCLR	Clear block of main memory	NI	PP	Yes
PBCLRPROT	Clear protect bit	NI	MA	Yes
PBCOMP	Compare two blocks	NI	MA	Yes
PBFILE1	Load/display file 1	O	MA	Yes
PBFMAD	Convert from ASCII to binary	R	PF	No
PBFMAH	Convert from ASCII to birary	R	PF	No
PBGETPAGEX	Reads page register from specified bank	NI	MA	Yes
PBHALT	System halt	NI	PP	Yes
PBILL	Illegal call - passes to TIP for CCP variants	NI	PP	Yes
PBLOAD	Load a canned message	R	PP	Yes
PBMAX	Get max of 2 numbers	NI	PF	No
PBMEMBER	Test ASCII set membership	NI	PF	No
PBMIN	Get min of 2 numbers	NI	PF	No
PBPSWITCH	Loads page registers 30 and 31	NI	MA	Yes
PBPUTPAGE	Writes page registers to either bank	NI	MA	Yes
PBRDPAGE	Reads dynamic page register	NI	MA	Yes
PBSETPROT	Set protect bit	O	MA	Yes
PBSTPMODE	Sets page mode	NI	MA	Yes
PBTOAD	Convert to ASCII decimal	R	PP	No
PBTOAH	Convert to ASCII hexadecimal	R	PP	No
PB18ADD	Adds to 18-bit address (paging)	R	PP	No
PB18BITS	18-bit address functions (paging)	R	PP	No
PB18COMP	Compares two 18-bit addresses (paging)	R	PP	No
TOTIME	Programs execution timer	R	PP	No
TOSTART	Starts program execution timer	R	PP	No
TOSTOP	Stops program execution timer	R	PP	No

**NI = Noninterruptable
O = OPS level only
R = re-entrant

*PP = PASCALL procedure
PF = PASCAL function
MA = Macroassembler

A functional call to a macroassembly program differs in that a PASCAL forward reference describing the calling sequence must appear before all function calls in the source code so that type-checking on the function return value can be performed.

Defeating Type-Checking in PASCAL Procedure Calls

The PASCAL compiler is a one-pass compiler. When it encounters a procedure call in source code, it may or may not have processed the calling sequence of the called program. If the calling sequence has been processed, all parameters of the user's procedure are error checked. The type of each parameter corresponds to the type specified in the calling sequence and the number of parameters must be the same. No expressions and no fields of less than a word in length in a packed record can be variable parameters.

If the calling sequence of a program has not been processed when a call to it is encountered, the PASCAL compiler generates a subroutine jump to an external symbol. The standard calling sequence is then generated; however, no error checking is done on the parameters. This situation defeats type-checking in the procedure call.

If used carefully, defeating type-checking can be a useful technique. For example, arrays with the same element types but of different lengths are treated as different types by PASCAL. Therefore, any program needing variable length array input as a variable parameter must defeat type-checking. Ramifications of defeating type-checking are as follows:

- All calls from PASCAL programs to macroassembly procedures automatically defeat type-checking unless defined as FORWARD.
- PASCAL and macroassembly functions cannot defeat typechecking.

HANDLING ROUTINES

Seven handling routines for number conversion are listed below and described in the following paragraphs.

- PBFMAD - converts from ASCII decimal to binary
- PBFMAH - converts ASCII hexadecimal to binary
- PBMAX - finds larger of two numbers
- PBMEMBER - tests number to find whether it is a member of the user defined subset of ASCII code
- PBMIN - finds smaller of two numbers
- PBTOAD - converts binary to ASCII decimal
- PBTOAH - converts binary to ASCII hexadecimal

PBFMAD – Converts from ASCII Decimal to Binary

PBFMAD converts up to five ASCII decimal characters in a buffer into binary number contained in one 16-bit word. The calling sequence is

PBFMAD (parm1, parm2, parm3).

Parm1 is integer type; the converted word is returned in parm1. Parm2 is a pointer specifying the buffer address where the decimal digits to be converted are located. Parm3 is an integer variable specifying the index where the first decimal digit to be converted is located within the buffer.

PBFMAD is a Boolean function. If PBFMAD is true, the conversion was successful; otherwise, there was either bad data or a bad index.

PBFMAH – Converts from ASCII Hexadecimal to Binary

PBFMAH converts up to four ASCII hexadecimal characters in a buffer to a binary number stored in one 16-bit word. The calling sequence is

PBFMAH (parm1, parm2, parm3).

Parm1 is a variable parameter of type BOOVERLAY; the converted word is returned in parm1. Parm2 is a pointer to the buffer address where the hexadecimal characters to be converted are located. Parm3 is an integer parameter specifying the index where the first hexadecimal character to be converted is located within the buffer.

Like PBFMAD, PBFMAH is a Boolean function. If true, PBFMAH indicates the conversion was successful. Otherwise, there was either bad data or a bad start/stop index.

PBMAX – Funds the Larger of Two Numbers

PBMAX is a function that returns the larger (maximum) of two given numbers. The calling sequence is

PBMAX (parm1, parm2).

Parm1 and parm2 are integers to be compared. The larger of parm1 and parm2 is returned by PBMAX.

PBMEMBER – Tests ASCII Set Membership

PBMEMBER determines whether or not a given ASCII character is a member of a user-defined set of ASCII characters. PBMEMBER overcomes the 255X PASCAL restriction of having one-word, 16-element sets by accessing an array of one-word sets. A character is broken up for testing by the following format:

7	6 4	3 0
	Index into array of sets	Element number in set

In an array of type JSACIISET, 128 bits are reserved (one for each possible ASCII character), where JSASCIISET = <u>array</u> (0..7) <u>of</u> SETWORD. Characters are located in the set by bit number; for instance, a blank (20_{16}) is bit number 20_{16}. Bits of the JSASCIISET array are numbered as follows:

Word 0	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7
F 0	1F 10	2F 20	3F 30	4F 40	5F 50	6F 60	7F 70

Bit Numbers (hexadecimal)

Therefore, the value initialization for testing hexadecimal characters is

<u>var</u> JSHEXSET: JSACIISET;
<u>value</u> JSHEXSET = (0, 0, 0, $\underbrace{3F_{16}}_{\text{digits 0-9}}$,
$\underbrace{7E_{16}}_{\text{characters A-F}}$, 0, 0, 0);

The calling sequence is

PBMEMBER (parm1, parm2).

PARM1 is a value parameter of type BOOVERLAY containing the character to test. Parm2 is a variable parameter of type JSASCIISET and is the set to test parm1 for membership. PBMEMBER is a Boolean function; it returns a true value if the character is in the set and a false value otherwise.

PBMIN — Funds the Smaller of Two Numbers

PBMIN is a function that returns the smaller (minimum) of two given numbers. The calling sequence is

PBMIN (parm1, parm2).

Parm1 and parm2 are integer value parameters. The smaller number of parm1 and parm2 is returned by PBMIN.

PBTOAD — Converts Binary to ASCII Decimal

PBTOAD converts a binary number contained in one 16-bit word to as many as five ASCII decimal characters. Leading zeros are suppressed. The converted digits are stored in a specified position in a buffer, followed by a blank. The calling sequence is

PBTOAD (parm1, parm2, parm3, parm4).

Parm1 is an integer containing the word to be converted; parm2 is a pointer to the buffer that stores the converted ASCII digits. Parm3 and parm4 are integers specifying the start and stop indices for storing the converted ASCII digits in the buffer. The JMCNVTO (convert to ASCII) system table is used by this routine.

PBTOAH – Converts Binary to ASCII Hexadecimal

PBTOAH converts a binary number contained in one 16-bit word into four ASCII hexadecimal characters. The converted characters are stored in a specified position in a buffer, followed by a blank. The calling sequence is

PBTOAH (parm1, parm2, parm3, parm4)

Parm1 is a hexadecimal value and contains the word to be converted. Parm2 is a pointer to the buffer that stores the converted hexadecimal characters. Parm3 and parm4 are integers specifying the start and stop indices for storing the characters in the buffer. The SMCNVTO (convert to ASCII) system table is used by this routine.

MAINTAINING PAGING REGISTERS

Five subroutines maintain the paging address system for an NPU with more than 65K words of main memory. (The maximum allowable address is $3FFFF_{16}$ and requires 18 bits.) Three other subroutines allow arithmetic and functional operations on 18-bit paging type addresses.

PBSTPMODE – Sets Paging Mode

PBSTPMODE sets the page mode for one of the three possible types of operation: no paging, paging with bank 0 page registers, or paging with bank 1 page registers. Calling sequence is

PBSTPMODE (parm)

Parm is the input index:

0 - use page mode 0; bank 0 registers
1 - use page mode 1; bank 1 registers
2 - absolute; no paging

PBPSWITCH – Performs Page Switching

PBPSWITCH loads the two dynamic page registers (30 and 31) using the input specified page register base value. Calling sequence is

PBPSWITCH (parm)

Parm is the page register base value for the program to be executed (programs must execute within a single 2K-word page). Output of the subroutine is that the dynamic paging registers are ready for use.

PBRDPGE – Reads Dynamic Page Register

PBRDPGE reads the contents of the dynamic page register (30) and returns the base address in the register to the requestor. Calling sequence is

PBRDPGE

There are no input parameters.

PBPUTPAGE – Write Specified Page Register

PBPUTPAGE loads a specified page register (number and bank) with a specified value. Calling sequence is

PBPUTPAGE (parm1, parm2)

Parm1 contains the page number; a bank flag uses the leftmost bit (flag = 0 indicates bank 0; flag = 1 indicates bank 1). Parm2 is the 9-bit value to be loaded in the designated register. Upon return, the specified page register is loaded.

PBGETPAGE – Reads Specified Page Register

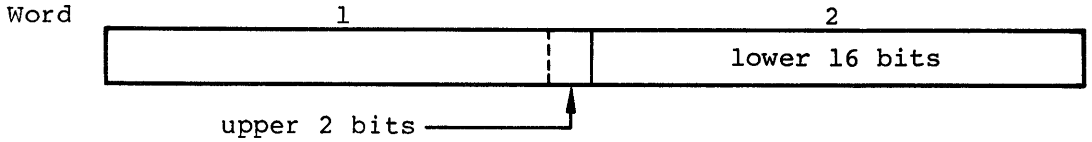
PBGETPAGE reads th contents of the specified page register and returns them to the user. Calling sequence is

PBGETPAGE (parm1, parm2)

Parm1 designates the number of the register and uses the leftmost bit as a bank flag (flag = 0 indicates bank 0; flag = 1 indicates bank 1). Parm2 is the location used to return the page register contents to the caller.

PB18ADD – Add Bit Addresses

PB18ADD adds two 18-bit addresses together. Format of an 18-bit address is as follows:



The calling sequence is

PB18ADD (parm1, parm2)

Parm1 and parm2 are the two addresses to be added in B018BITS format. Output is the single 18-bit address.

PB18BITS — 18-Bit Address Functions

PB18BITS performs one of five possible functions:

- Stores a number into an 18-bit address
- Reads the specified 18-bit address
- Clears the protect bit in an 18-bit address
- Sets the protect bit in an 18-bit address
- Forms an 18-bit address from a 17-bit address

The calling sequence is

```
PB18BITS (parm1, parm2, parm3)
```

Parm1 is an 18-bit address, parm2 is the read/store word address and parm3 specifies the function to be performed. The output is a properly performed function.

PB18COMP — Compares Two 18-Bit Addresses

PB18COMP makes a comparison between two 18-bit addresses. The calling sequence is

```
PB18COMP (parm1, parm2, parm3)
```

Parm1 is the A address, and parm3 is the B address. Parm2 specifies the type of comparison: A COMP B, where COMP is one of =, ≠, , , , or . The output is a Boolean function: true if A COMP B1; false if any other condition exists.

BLOCK FUNCTIONS

Two standard block function subroutines are provided: PBCLR clears the contents of a block, and PBCOMP compares the contents of two blocks.

PBCLR — Clears a Block of Main Memory

This subroutine is used to clear any block-sized area in main memory. Calling sequence is

```
PBCLR (parm1, parm2)
```

Parm1 is the starting address of the block to be cleared; parm2 is the number of consecutive words to be zeroed. Output is a cleared block of memory.

PBCOMP — Compares Two Equal Length Blocks

After block comparison, a Boolean answer (1 represents true, 1, false) is returned to the caller. The calling sequence is

```
PBCOMP (parm1, parm2, parm3)
```

Parm1 and parm2 are the starting address of the two blocks to be compared; parm3 is the number of words compared in each block. Output is the Boolean true-false function, which depends on whether the blocks had identical contents.

SET/CLEAR PROTECT BITS

The protect bit is bit 17 of the main memory word. It cannot be used for data, but it can be used to deny unprotected programs access to the word. The bit (as well as the parity bit) is dropped by most interregister transfers.

PBSETPROT – Set Protect Bit

PBSETPROT sets the protect bit at a specified address. Calling sequence is

```
PBSETPROT (parm)
```

Parm is the address of the protect bit to be set.

PBCLRPOT – Clear Protect Bit

PBCLRPOT clears the protect bit at the specified address. Calling sequence is

```
PBCLRPOT (parm)
```

Parm is the address at which the protect bit is to be cleared.

MISCELLANEOUS SUBROUTINES

PBFILE1 – Load/Display File 1

PBFILE1 consists of two routines: PBEF (load file 1) and PBDF (display file 1). Both programs execute specified firmware sequences to perform the load or display operations. Because of formware timing constraints, a maximum of 12 transfers per call can be specified during on-line operation. During off-line operation, as many as 256 transfers can be specified.

PBEF transfers the contents of memory to file 1 starting at a specified register. Calling sequence is

```
PBEF (parm1, parm2)
```

Parm1 is a value paramter formatted as follows:

15	7 0
Number of words to load	First File 1 register to load

To load all 256 registers, set parm1 to 0. Parm2 is a value parameter specifying the address of the first memory location to transfer.

PBDF transfers the contents of file 1, starting at register n, to memory. Calling sequence is

PBDF (parm1, parm2)

Parm1 is a value parameter formatted as follows:

15	7 0
Number of words to move	First File 1 register to transfer

To display all 256 registers, set parm1 to 0. Parm2 is a value parameter specifying the memory address to receive the first register transfer.

PBHALT – Stops the NPU

PBHALT stops the system after a serious error has occurred. The following information is saved, starting in consecutive words at address 30_{16}.

- Return address of program calling PBHALT, or a value relating to a halt code
- Halt code (indicates a reason for the halt)
- Software registers

Calling sequence is

PBHALT (parm)

Parm is an integer value parameter specifying the halt code. The halt message printed at the local console is

*HALT xxxxx yyyy

xxxxx is the return address of the program calling PBHALT and yyyy is the hexadecimal halt code or a value relating to the halt code.

PBILL – Illegal Calls

This subroutine is used to stop the NPU when calls are made to TIPs that are not a part of the CCP system. Calling sequence is

PBILL

PBILL calls PBHALT with the halt code for an illegal TIP call.

PBLOAD – Load a User-Defined Message

The PBLOAD module loads a user-defined message into a buffer starting at the designated character position. The calling sequence is

PBLOAD (parm1, parm2, parm3, parm4)

Parm1 points to the location where the user-defined message is to be loaded and parm2 specifies the text of the message to be loaded. Parm3 specifies the starting position in the buffer of the first character in the message and parm4 specifies the position of the last data character in the message after it is loaded in the buffer. Parm4 overrides the message length. Example:

```
VAR      Buffer:  B0BUFPTR:  (assume a 32-word buffer)
         MSG  :  J0ML10:
Value MSG = ( Ξ 0123456789 ] Ξ );
         .
         .
         .
PBLOAD (BUFFER, MSG, J1FRSTCHAR, J1LST32);
```

NOTE

All user-defined messages must have a right bracket ( ) as the end of message delimiter unless parm3 minus parm4 is less than the message length.

PROGRAM EXECUTION TIMERS

Three subroutines (TOTIME, TOSTART and TOSTOP) provide execution timing analysis for programs. TOSTART sets a status mode (flag bit 206) which can be used by an external hardware instrument to start a timer. TOSTOP resets the status bit. TOTIME measures the elapsed time. Output is the total execution time as measured by an external hardware instrument.

CONSOLE SUPPORT

This group of modules provides the terminal interface package (TIP) for the NPU console. Console devices communicate with the NPU via the A/Q register interface, rather than through the multiplex subsystem interface. Two categories of subroutines are discussed in the following paragraphs.

- General peripheral processing: these modules assign device, start, read, and write.
- Console processing: this set of routines forms the console TIP.

GENERAL PERIPHERAL PROCESSING

These subroutines provide for general peripheral functions.

- Starting I/O and (if necessary) assigning a device. Two routines perform these services: PBIOSER and PBSTARTIO.

  PBIOSERV reformats the logical request packet (LRP) from the user into a physical request packet (PRP). A device code is assigned and the subroutine tests whether there are too many messages awaiting delivery. If so, the new message is discarded. Then PBSTARTIO is called.

PBSTARTIO either starts the I/O, using the LRP packet from PBIOSERV, or it queues the logical request packet to the appropriate driver, using a worklist entry. If immediate I/O is requested but cannot be accomplished, the request is rejected. This subroutine sets up the device controller table parameters and issues the I/O start command. The individual driver interrupt handler then takes control.

- Testing whether device is ready, PBTCSTIORDY. Input to this routine is the device number. If the device status indicates it is ready for I/O, a ready indication is returned to the caller.

- Off-line quick output, PBQUICKIO. This permits one buffer (a short message) to be output while the NPU is in off-line mode (such as initialization breakpoint or during halt operations). As input, the caller specifies the device to be used and the location of the message to be sent.

- Timeout: PBIOTMP and PBTMEOUT are discussed in this section with other timing services.

- Ready and write a character to a peripheral device. PBWRITE and PBREAD handle the single character transfers. Characters passing over the A/Q channel are in unpacked format, right justified in the A register. (Q register usually carries peripheral addressing information.)

PBWRITE writes data or director functions to a local peripheral device. The subroutine uses the macroassembler routine PBPUTCHAR, to write the character. Attempts are made to write until a retry threshold is reached. At that time, the attempts cease and the reject error is counted by the reject counter. This can cause a peripheral device timeout. In any event, Q and A values are saved for debugging.

PBREAD reads data or status from a peripheral device. The routine uses the macroassembler routine, PTGETCHAR, to read the character until a retry threshold is reached. At that time, the attempts cease and a reject error is added to the count in reject counter. This can cause a peripheral device timeout. In any event, Q and A values are saved for debugging.

- Common driver completion PBDRCOMPL. This routine uses a completion code in the logical request packet. It requires device identification and a physical request packet address as input. Completion actions can include one or more of the following:

    - Releasing message output buffers
    - Changing I/O request flags
    - Starting another message transfer
    - Releasing current messages physical request packet

CONSOLE SUPPORT SERVICES

For certain applications, a local console is used as a communications supervisory position. Two console functions can be selectively activated or deactivated by the console operator (or at build time). These functions are orderwire and diagnostics. When one or both of these functions are transferred to a remote console, the corresponding functions must be deactivated at the local console.

The orderwire function is employed for both input and output traffic messages. The diagnostic function is used for input of diagnostic commands and output of hardware diagnostic messages.

CONSOLE WORKLIST ENTRY

A type B0CHWL worklist entry is made by the internal process output procedure for every message placed in an empty console queue. Such entry contains the console TCB address.

CONSOLE CONTROL MESSAGES

All console control messages begin with a slash (/) and end with an end-of-transmission code, control D (this consists of pressing the CONTROL and D keys simultaneously). Table 4-5 contains console control messages and the results of each.

Several routines consititute or support the console TIP.

- PBDISPLAY queues a message of 300 characters or less for output on the local console. The input parameter is the location of the message to display. This routine is a part of the base and is not technically a part of the console TIP. The routine could be used to support other devices.

NOTE

Every canned message must have a right bracket ( ]).

Canned messages use 32-word buffers.

PBDISPLAY uses the PBLOA and PBIOSERV subroutines to load a canned message and to provide I/O services. PBDISPLAY also uses system structure JCOPSLRP (OPS-level console legical request packet).

- PBOFMT formats the output for the console. Characters are converted to hexadecimal and stored in a new buffer chain.

TABLE 4-5. NPU CONSOLE CONTROL COMMANDS

Command	Function
/SUP	Puts console in supervisory mode
/ORD	Puts console in orderwire (diagnostic) mode
/OVL	Puts NPU in overlay mode
/REQ	Message interrupted by manual interrupt is requeued to console
/CAN	Message interrupted by manual interrupt is cancelled
/MTQ	Flushes console queue
IN OUT LOC	Controls routing of service messages (input, output, and locally generated messages)
MSNOP	Generates message to NOP

- PBTTYSETMODE switches the console (keyboard/display or teletypewriter) between read and write modes. If the console is in TUP mode a TUP message flag is set. If the output interrupt flag is already set, the subroutine restarts the message output. Otherwise, the message is sent to the console primary output device. A 5-minute timeout period is set when entering read mode.

- PBTTYINT is the interrupt handler for the console. Interrupts clear the I/O timer. Action depends on the interrupt type, such as one of the following:

Type	Action
spurious	count as spurious interrupt
alarm	clear console
manual	change mode
data (read)	read character
data (write)	write character
other	clear interrupt

This interrupt handler is composed of several local subroutines.

- PBSUPMSG decodes and executes supervisory (/SUP) input messages from the NPU console. The subroutine routes to the NPU console input service messages (SMs), output SMs, locally generated SMs, and messages that are directed to the network operator (NOP). An error message is generated if the messages cannot be routed.

- PBIFMT formats input messages from the console. Supervisory messages (/SUP) are specially flagged. Messages are converted from hexadecimal and the buffer headers are prepared. conversion takes place in a new chain of buffers. This subroutine uses other local internal subroutines. Otherwise the output is a message in normal network block protocol. If this is a /SUP message, the action directed by the /SUP message has been performed.

- PBQCONSOLE sets a format flag for the console format (message heading) and then calls PBQ1BLK to queue the message to the console TCB. This routine is called from PBSWITCH which detects that the message is to be sent to the console, rather than upline to the host, or that the message is to be sent both upline and to the console.

MULTIPLEX SUBSYSTEM 5

The multiplex subsystem contains the hardware, microprograms, and software elements necessary to provide data and control paths for information interchange between the various protocol handlers (TIPs and LIP) and all communications lines. Design of the subsystem is based on the multiplex loop concept, which is a demand-driven system for gathering input data and status from the communications lines, and distributing output data and control information to the communications lines. All of this is done on a real-time basis. Figure 5-1 shows the basic elements of the multiplex subsystem.

A major purpose of the multiplex subsystem is to transfer the task of processing lines according to physical characteristics from the TIPs to the multiplex subsystem programs. The TIPs need only command the multiplex subsystem according to the logical characteristics of a line; the physical characteristics are handled by the multiplex subsystem and are transparent to the TIPs.

Line-oriented input and output buffers provide temporary storage for data. The input data is placed in the circular input buffer (CIB) from which it is later extracted (demultiplexed), transformed to IVT/BVT ASCII format by the appropriate TIP and moved into a line-oriented input buffer. The part of the TIP that does this (called input state programs) is controlled by the multiplex subsystem. The OPS-level TIP informs the command driver where the programs are located; the multiplex subsystem's input processor controls execution of the input state programs. For trunks, the frames are removed from the block formatted data, and the blocks are reconstituted.

Output data is picked by the output processor from an output data buffer. The address of this buffer and other transfer information is supplied by the OPS-level TIP to the command driver. Data is in terminal format or (for LIP frames only) in downline frame format.

The multiplex subsystem is event-driven by interrupts: an output data demand (ODD) for the next character of output data, or the input line frame received interrupt which indicates that data (and possibly CLA status) is contained in the CIB ready for demultiplexing.

The interrupts are handled with global information stored in various tables. The subsystem processes data on a character-by-character basis while user programs (TIPs) process data on a message or block basis. Circuit, modem, and subsystem status is detected and transferred to the TIPs using OPS-level worklist calls. Control information is received from the TIPs in the form of a call to the command driver with an attached command packet. This command packet is used to set up the multiplex LCB (MLCB), which is the principal table used to control the transfer.

HARDWARE COMPONENTS

The multiplex subsystem includes the multiplex loop interface adapter (MLIA), loop multiplexers, and communications line adapters (CLAs).

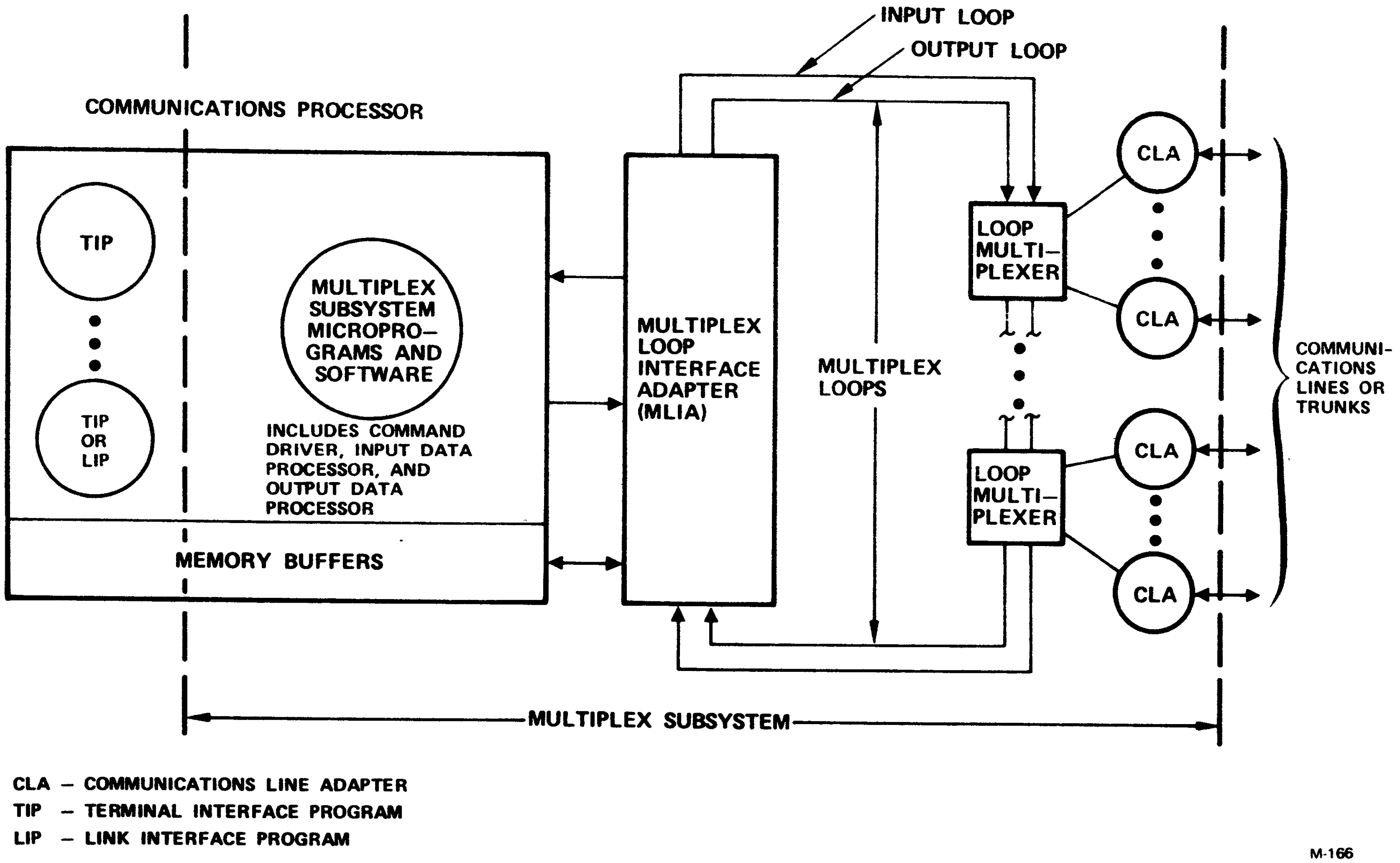


Figure 5-1. Basic Elements of the Multiplex Subsystem

MULTIPLEX LOOP INTERFACE ADAPTER

The MLIA provides hardware interface between the multiplex input/output loops and the multiplex subsystem software. The major functions are as follows:

- Management of the I/O loops
- Input data buffering - compensates for the difference in rate at which characters are removed from the input loops and the rate at which they are stored in the main memory
- Output data demand (ODD) detection and buffering
- Multiplex loop error detection
- Generation of interrupts for the multiplex subsystem microprograms and software for functions such as:
    - Output data demand received
    - Line frame received
    - Loop error conditions

LOOP MULTIPLEXERS

Each loop multiplexer provides an interface between a group of as many as 32 CLAs and the demand-driven multiplex loop. Its primary function is to receive parallel data from the CLAs and present it to the serial input loop in the loop cell format. Conversely, it assembles serial data in the loop cell format from the output loop and presents it to the CLAs in parallel form.

COMMUNICATIONS LINE ADAPTERS (CLA)

The CLAs provide the interface between the loop multiplexers and the communications lines. The primary functions of the CLAs are to assemble serial data from the communications line into parallel data and present this data to the loop multiplexer or, conversely, to disassemble parallel data from the loop multiplexer and present it in serial form to the communications line. The CLA operating characteristics can be altered under program control for such functions as signal rate, character length, parity, and stop bit duration.

SYSTEM AND USER INTERFACES

The system and user interfaces are described in detail in the following paragraphs to promote a better understanding of the internal multiplex subsystem interfaces.

SYSTEM INTERFACES

A TIP or a LIP is a multilevel program that executes at three processing levels:

- Multiplex level 1 (firmware or microcode level)

- Multiplex level 2 (macrocode level)

- OPS level (processing to satisfy network protocol such as service message handling and timing)

Control passes to the TIP or multiplex control OPS level by use of worklist entries. Direct calls are used for the other two levels. The TIP or LIP must handle the worklist entry according to the program's current processing state. State programs operate on firmware levels. State instructions provide a type of reentrant processing where the states are related to entry points, which are in turn related to the various stages of processing a message. Each TIP or LIP decision logic that switches processing to the entry point determined by a combination of the worklist and the program state.

Figure 5-2 shows the multiplex level 2 worklist codes and the programs responsible for handling and generating these codes. Table 5-1 summarizes workcode functions for level 2 and table 5-2 describes the workcode functions for OPS level.

Multiplex Level 1 (Firmware)

This level of interface program processing handles all incoming characters and status. Worklist entries generated by the input state programs are directed to either multiplex level 2 or to OPS level for processing. For preliminary handling of CLA status, states 0, 1, 2, and 3 are reserved to handle special status, as follows:

- 0 is reserved for CLA status such as parity errors and data transfer overruns.

- 1 is reserved for DCD dropped.

- 2 is used when a TIP uses too many system buffers.

- 3 is used when buffer threshold is reached.

CLA status is analyzed by Modem State Programs and status that indicates a hard error is sent to level 2. For a two-wire line the transition of data carrier detect signal can be used as a logical end of text (ETX); that is, instead of generating a good block worklist entry, the input states wait for data carrier not detected to generate a good block received. This eliminates an extra worklist entry. The good block that is received is issued to OPS level for processing. For more information, refer to section 12 and the State Programming Reference Manual (see preface).

Multiplex Level 2 (PMWOLP)

This processing runs at the multiplex interrupt level. It is entered by means of worklist entries received from the modem state programs, the multiplex subsystem firmware, and the command driver. Processing at this level is primarily of an error nature. Each interface program provides code to process the workcodes at this level (MNOBT, MMCHOUT, MMFES, MMBREAR) plus any of its own that are generated in level 1. For synchronous TIPs and LIPs, no processing is required since the MMOBT entry is optional.

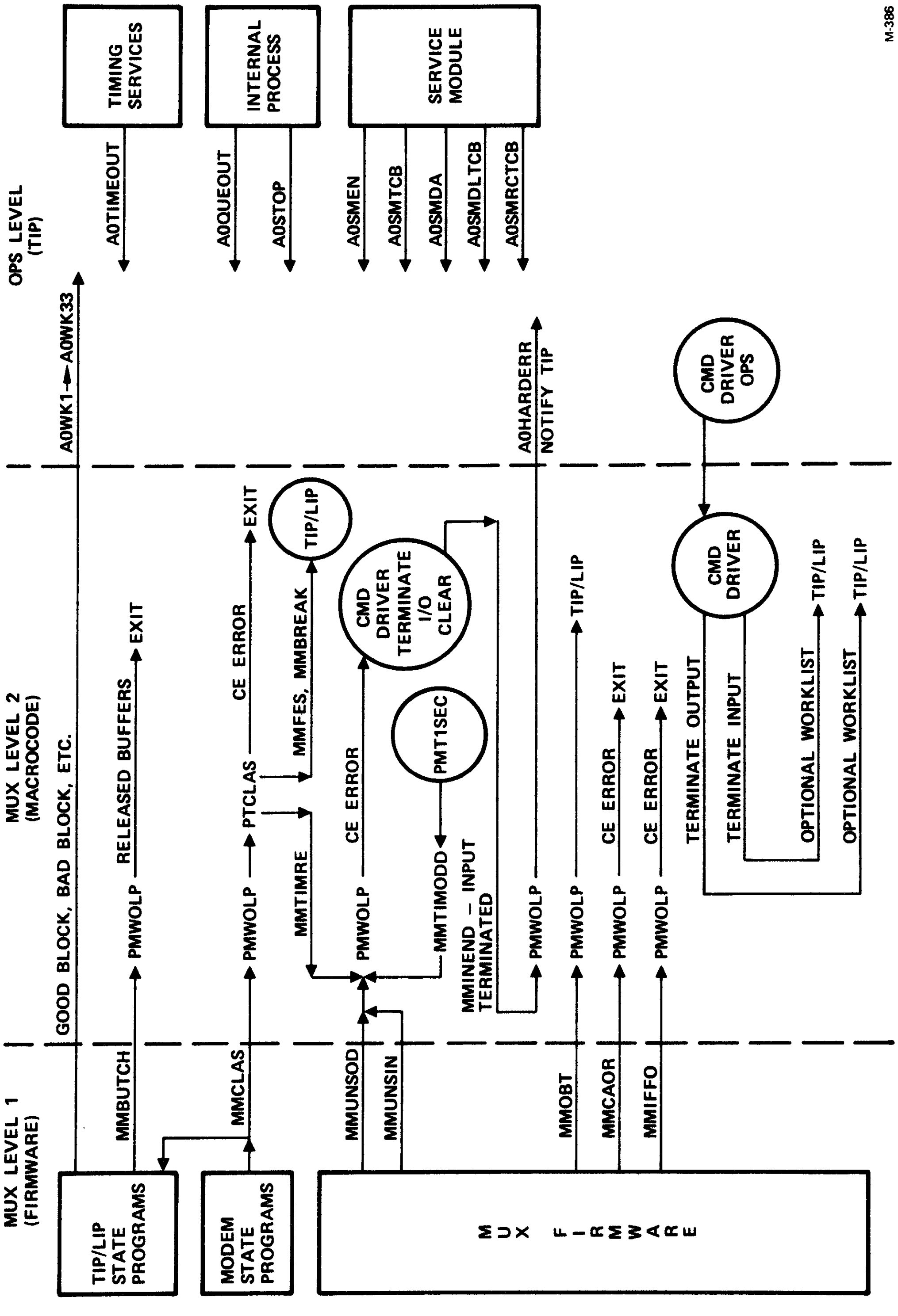


Figure 5-2. TIP and LIP Multiplex Worklist Communications

TABLE 5-1. MULTIPLEX LEVEL 2 WORKLISTS

Workcode	Workcode to TIP/LIP	Functions
MMCLAS	-	CLA status error, implies line error to IP[†]
MMUNSOD	-	Unsolicited output, implies hard error to PMWOLP, which disables the line
MMUNSIN	-	Unsolicited input, implies hard line error to PMWOLP, which disables the line
MMTIMODD	-	ODD timeout, implies hard line error to PMWOLP, which disables the line
MMTIMRE	-	Modem response timeout, implies hard line error to PMWOLP, which disables the line
MMOBT	MMOBT	Output block transmitted
MMBUTCH	MMBUTCH	Multiplex subsystem buffer threshold reached; buffers released
MMCHOUT	MMCHOUT	100-ms timeout
MMCAOR	-	CLA address out of range - not seen by IP
MMIFFO	-	Illegal lineframe format - not seen by IP
NMINEND	A0HARDERR	Input buffer terminated, response to PMWOLP command for hard errors
MMFES	-	Framing error status, TIP causes command driver to send delimiter to line (asynchronous lines)
MMBREAK	-	User break, TIP is called (asynchronous line)

[†] IP = appropriate interface program: TIP or LIP

TABLE 5-2. TIP/LIP OPS LEVEL WORKLISTS

Workcode to TIP/LIP	Description
A0WK1	Good block received from IP input states
.	
.	
A0WKn	Other workcodes from IP input states
A0HARDERR	Hard error detected from IP at level 2
A0TIMEOUT	Line timeout from timing services
A0QUEOUT	Output buffer queued to IP's TCB
A0SMEN	Line enabled from service module
A0SMTCB	TCB configured from service module
A0SMDA	Disable line command from service module
A0SMDLTCB	Delete TCB command from service module
A0SMRCTCB	Reconfigures TCB command from service module

INPUT STATE PROGRAM WORKLISTS

Input state program worklists from firmware level are passed directly to the TIP or LIP at OPS level.

The primary workcode generated is the CLA status workcode. After the modem state programs have analyzed the CLA status for soft errors (data carrier detect dropped and others) and determined that this is not a soft error, the input processor modem state program generates a CLA status worklist to this processing level. The CLA status handler (PTCLAS) analyzes the status and generates the appropriate CE error code. If a hard error is detected on the line, PMWOLP terminates input and output over the line. All multiplex level worklists for the line are discarded until a response from the terminate input logic is received. At that time the TIP is sent an OPS-level A0HARDERR worklist.

MULTIPLEX SUBSYSTEM FIRMWARE WORKLIST ENTRIES

The multiplex subsystem firmware generates nine worklists to the interrupt level. These can be divided into three categories:

- Hard errors for unsolicited input or output, and timeouts for output data demand or modem response.

- System notices that the output buffer has been transmitted, the buffer threshold has been reached so no more buffers can be assigned, or 100 ms have elapsed since the last input character was received.

- Multiplex loop errors that the CLA address is out of range or an illegal line frame format was detected.

COMMAND DRIVER WORKLIST ENTRIES

The command driver generates worklist entries at the request of the interface program. Two optional entries are generated: input terminated and output terminated.

OPS Level

The OPS level portion of the interface program handles all line or terminal polling, output block preparation, input block processing, service module interface for configuring lines and terminals, and line error handling. Worklists are generated to the interface processor by four different programs: 1) interrupt programs multiplex level 1 and 2; 2) timing services; 3) internal process; and 4) service module.

- Multiplex level 1 worklist normally indicates a good block has been received on input. The block is passed to the point of interface (POI) program and the interface program resumes its processing at the initial entry point or at the saved entry point where processing was suspended.

- Multiplex level 2 worklist indicates a hard error has occurred on the line. Normally a line nonoperational service message is sent to the host. Service on that line is discontinued until the host takes continuation action.

- Timing services worklist is generated whenever the line control block timer expires (BZLTIMER). It can be used as a means of delaying service on a line or indicating a line failure (failure to respond).

- Internal process worklist indicates that output is queued to the terminal control block (TCB) for this interface program. This is a worklist for interface programs that stop processing when there is nothing to do; it must therefore be restarted when the next output arrives.

- The service module (SVM) maintains the interface between the host and the interface program. SVM worklists indicate to the interface program those lines and terminals that are to be configured or are to be deleted from service.

USER INTERFACES

User interfaces to the multiplex subsystem can be divided into three categories:

- Command driver interface (PBCOIN and PMCDRV). These modules command communications to the multiplex subsystem and control data flow to and from the communications lines. These include setting up the hardware to start or stop transmissions.

- Common multiplex subroutines for TIPs are provided. These subroutines allow the multiplex subsystem to communicate input events to the user.

- State programs. PMCDRV sets up the operation and calls PMCOIN to escape to the firmware. On the firmware level, the input state programs provide processing on a character-by-character basis. State programs and their OPS-level interfaces are described in section 12.

Command Driver Interface

The command driver calling sequence from the OPS level is

PBCOIN (parm)

where parm is the command packet (NKINCOM). The command driver calling sequence from level 2 is

PMCDRV (parm)

where parm = NKINCOM is the name of the command packet. The general format of a command packet which is used for most commands (NKCMD type) is shown in figure 5-3.

WORD	15	7 0
0	Command	Parameter
1	Line Number	
2	Parameters	
3	Parameters	
4	Parameters	
5	Parameters	
6	Parameters	
7	Parameters	

Figure 5-3. Command Packet General Format

The following commands are available to the user for controlling the flow of data to and from the communications lines:

- NKCLRL - Clear line
- NKINIL - Initialize line
- NKCONTROL - Control line
- NKENBL - Enable line
- NKINPT - Input
- NKDOUT - Direct output
- NKINOUT - Input after output
- NKENDIN - Terminate input
- NKENDOUT - Terminate output
- NKDISL - Disable line
- NKTURN - Turn line around (not used)
- NKSPECIAL - Diagnostic interface

Individual subroutines handle the various requests. PMCOIN is the interface between the command driver and the firmware. PMCOIN can be used by other software users to clear a CLA. If it is so used, the it must be followed by a clear line command. Inputs to PMCOIN are the two global variables NGA and NGQ that hold command and port information for use in the A and Q registers by the firmware.

CLEAR LINE COMMAND

The clear line command (NKCLRL) causes the subsystem to clear (reset) all line-oriented software and hardware (CLA) functions associated with the line specified by the line number. The command format is as follows:

WORD	15	7	0
0	NKCMD	NKLTYP	
1	NKLINO		

NKCMD - Command code (NKCLRL)

NKLINO - Line number, identifies port and subport

NKLTYP - Line type; specifies line-type entry; defines physical characteristics of port, modem, and circuit type

INITIALIZE LINE COMMAND

The initialize line command (NKINIL) establishes the line type of the specified port and places the line in a mode in which the subsystem monitors and processes modem and circuit related status. Other line-related functions, such as processing of input and output characters, are inhibited while the line is in the initialize mode. The command format is as follows:

WORD	15	7	0
0	NKCMD	NKLTYP	
1	NKLINO		

NKCMD - Command code (NKINIL)
NKLINO - Line number
NKITYP - Line type; specifies line-type table entry

CONTROL COMMAND

The control command (NKCONTROL) serves a twofold purpose. It can define the character transmission characteristics of a given line according to the transmission characteristics key (NKTCKY) for input/output signaling rate, character length, parity type, stop bit duration, and sync character. The command can also specify up to five modem/circuit control functions, such as echo, break, terminal busy, or resync. Such control functions are specified in the optional fields of the command packet.

Generally, the command is used to initialize or alter the character transmission characteristics of the line or to generate circuit control functions. This command must not be issued before the initialize command. The control command format is as shown in figure 5-4. Optional modem/circuit functions are defined in table 5-3.

ENABLE LINE COMMAND (NKENBL)

The enable line command directs the subsystem to activate, as a function of line type, the necessary modem signals to allow the local modem to connect to the specified communications line. The command also conditions the subsystem to monitor and analyze any changes in the modem status for signals indicating that a line connect occurred. Character processing functions are inhibited during the time the line is in the enable mode. The format for the enable line command is shown in figure 5-5.

<table>
<tr><th>WORD</th><th>15</th><th>14</th><th>7</th><th>6</th><th>0</th></tr>
<tr><td>0</td><td colspan="2">NKCMD</td><td colspan="3">NKTCKY</td></tr>
<tr><td>1</td><td colspan="5">NKLINO</td></tr>
<tr><td>2</td><td>F1</td><td>NKFUN1</td><td>F2</td><td colspan="2">NKFUN2</td></tr>
<tr><td>3</td><td>F3</td><td>NKFUN3</td><td>F4</td><td colspan="2">NKFUN4</td></tr>
<tr><td>4</td><td>F5</td><td>NKFUN5</td><td colspan="3">NKZERO</td></tr>
</table>

NKCMD - Command code (NKCONTROL)

NKTCKY - Optional character transmission key. If nonzero, references the character transmission characteristics table.

NKLINO - Line number

F1 thru F5 and NKFUN1 thru NKFUN5 - Optional modem/circuit function; if the associated flag (NKSRF1 - NKSRF5) is set, the function is to be implemented.

1 = Function to be implemented
0 = Function disabled

NKZERO - Delimits end of options. NKZERO is placed in the byte following the last requested modem/circuit function; five functions can be specified.

Figure 5-4. Control Command Format

WORD	15	14	11	7	0
0	NKCMD			NKTCLS	
1	NKLINO				
2	Not used				
3	NKUOPS			NKIFCD	
4	F1		NKBLKL		
5	Not used				
6	NKSCHR				

NKCMD - Command code (NKENBL)

NKTCLS - Terminal class

NKLINO - Line number

NKUOPS - Eight user flags (NKUOP1 - NKUOP8) can be accessed either individually or as an 8-bit field

NKIFCD - First character displacement (FCD) of first buffer of input block; optional FCD or zero. If zero, use value from the terminal characteristics table (NJTECT)

F - NKNOXL, the code translate flag

1 = translate
0 = do not translate

NKBLKL - Block length; optional block length or zero. If zero, use value from NJTECT

NKSCHR - Special character (optional character or 0)

Figure 5-5. Enable Line Command Format

TABLE 5-3. OPTIONAL MODEM/CIRCUIT FUNCTIONS

Function Mnemonic	Function Provided	Description
NOISR	STATUS[†]	Input status request
NORTS	RTS	Request to send
NOSRTS	SRTS	Secondary request to send (Supervisory Channel)
NOOM	OM	Originate mode/auxiliary modem control
NOLM	LM	Local mode/auxiliary modem control
NOLT	LT	Local test
NODTR	DTR	Data terminal ready
NOTB	TB	Terminal busy (line busy out)
NORSYN	RSYN	Resynchronize
NONSYN	NSYN	New sync
NOBREAK	BREAK	Send break
NODLM	DLM	Data line monitor
NOECHO	ECHO	Echoplex mode
NOLBT	LBT	Loopback test
NOION	ION	Input on
NOOON	OON	Output on
NOISON	ISON	Input supervision on
NOPON	PON	Parity on
NOPSET	PSET	Parity set (1 = even, 0 = odd)
NOCLLS	CLLS	Character length (LSB)
NOCLMS	CLMS	Character length (MSB)

[†] Pulsed functions, provide momentary signal and need not be reset

INPUT COMMAND (NKINPT)

The input command directs the multiplex subsystem to initiate the processing of data on the specified input line (i.e., turn on the input side of the communications line adapter. The processing functions provided by the subsystem are determined by the input processing state program index. Additional information is passed by a pointer table address for the input processing states. If this option is not used, the information is taken from the terminal characteristics table (NJTECT). Parity is stripped for normal processing or passed for test purposes. Format of the input command is shown in figure 5-6.

OUTPUT COMMAND (NKDOUT)

The output command permits output messages to be directed to a specified output line. Line, modem, and control functions, as defined in the line type tables, are generated by the subsystem as a function of the physical line requirements.

Output continues until the character specified by the last character displacement is transmitted. At that point, the subsystem chains to the next output buffer, if the chain address in the buffer is nonzero. Output stops if the chain address is zero or if the suppress chaining flag (BFSUPCHAIN) is set in the flag word of the first output buffer.

The subsystem generates an optional worklist entry for the user program for each data block output by the subsystem. If the buffer output is the last data buffer of a transmission block and line turnaround is required, 1) the subsystem generates the proper modem control signals to turn the line around, 2) monitors modem status for line turnaround, and 3) notifies the appropriate terminal dependent subroutine that the line is ready for input. Modem signals and modem status analysis functions are specified by the line type tables.

Either the terminate output or disable command can also be used to terminate output processing functions on a specified line. Receipt of either command causes the subsystem to immediately cease all processing functions associated with the specified line.

The format of the output command is as follows:

WORD	15	7 0
0	NKCMD	Not used
1	NKLINO	
2	NKOBP	

NKCMD - Command code (NKDOUT)
NKLINO - Line number
NKOBP - Output buffer pointer

<table>
<tr><th>WORD</th><th>15</th><th>14 11</th><th></th><th>7</th><th>6</th><th>0</th></tr>
<tr><td>0</td><td colspan="3">NKCMD</td><td colspan="3">Not used</td></tr>
<tr><td>1</td><td colspan="6">NKLINO</td></tr>
<tr><td>2</td><td colspan="6">Not used</td></tr>
<tr><td>3</td><td colspan="3">NKUOPS</td><td>F1</td><td>F2</td><td>NKISTAI</td></tr>
<tr><td>4</td><td>F3</td><td>F4</td><td colspan="4">NKBLKL</td></tr>
<tr><td>5</td><td colspan="6">NKISPTA</td></tr>
<tr><td>6</td><td colspan="3">NKSCHR</td><td colspan="3">NKCNT1</td></tr>
<tr><td>7</td><td colspan="6">NKCXLTA</td></tr>
</table>

NKCMD - Command code (NKINPT)

NKLINO - Line number

NKUOPS - Eight user flags (NKUOP1 - NKUOP8). NKUOP1 is bit 15 in the MLCB user flag field,...NKUOP8 is bit 8 in that field. NKUOPS is moved into MLCB if NKMVB is 1.

F1 - NKMVB, move block of user flags into MLCB

F2 - NKRPRT, strip parity flag

1 = strip parity
0 = do not strip parity

NKISTAI - Input state program index

F3 - NKNOXL, code translate flag

1 = translate
2 = do not translate

F4 - NKSCENBL, change special character flag

NKBLKL - Block length. If nonzero, this replaces CC2 in the MPCB.

NKISPTA - Pointer to input state program pointer table address. Optional address or zero. If zero, use NJTECT value.

NKSCHR - Special character, moved to MLCB if NKSCENBL flag is set.

NKCNT1 - Character count, moved into the CC1 field of the MLCB if the value is nonzero.

NKCXLTA - Code translation table address. If nonzero, this replaces the current code translation table address in MLCB.

Figure 5-6. Input Command Format

INPUT AFTER OUTPUT (NKINOUT)

This command permits interactive terminals (such as a display/keyboard combination) to be immediately ready to receive input data in response to a message displayed at the terminal. An index to the input state process table indicates the treatment of the returned data. The format for this command is shown in figure 5-7.

TERMINATE INPUT COMMAND (NKENDIN)

This command enables the TIP to direct the multiplex subsystem to immediately stop input processing functions on the specified line. All input characters and buffers are discarded. The TIP program can, by issuing an input command, direct the subsystem to resume input on the line. Transmission line characteristics are not altered by the terminate input command and therefore the TIP need not generate a control command. The format for the terminate input command is shown in figure 5-8.

After processing the terminate input command, the subsystem optionally generates a worklist entry to the TIP as specified in the worklist and workcode.

TERMINATE OUTPUT COMMAND (NKENDOUT)

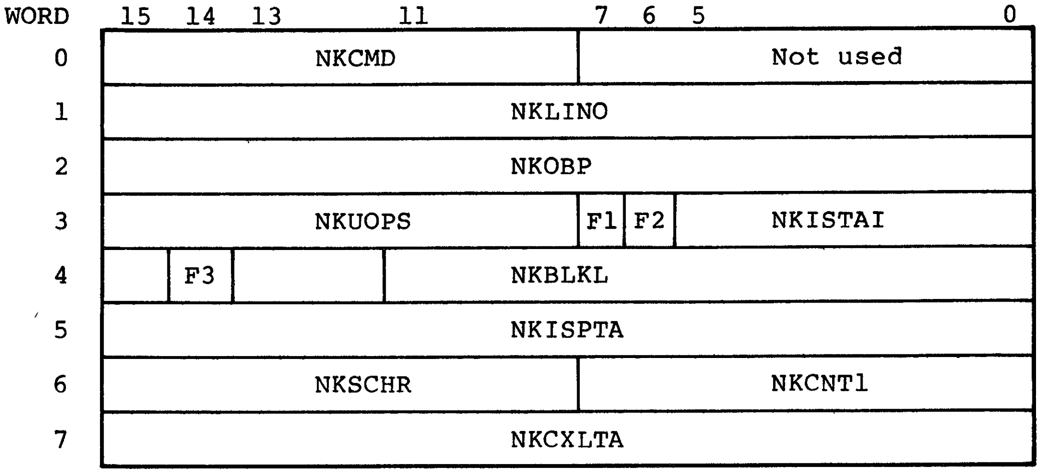
This command enables the TIP to direct the multiplex subsystem to terminate output processing functions on the specified line immediately. After processing the terminate command, an optional worklist entry is generated to the TIP, using the specified worklist and workcode. This command is used when the TIP interrupts an outgoing message for a higher priority message, or when an abnormal line condition occurs. The format of the terminate output command is shown in figure 5-9.

DISABLE LINE COMMAND (NKDISL)

The disable line command directs the multiplex subsystem to terminate all processing functions of the specified line. Modem control signals are generated to inhibit further exchange between the local modem and the communications line. The subsystem also releases all data structures defining the character processing functions for the line. To reactivate, a control, initialize, and enable command, followed by either an input or output command, must be issued. The format for the disable line command is as follows:

WORD	15	7 0
0	NKCMD	Not used
1	NKLINO	

NKCMD - Command code (NKDISL)
NKLINO - Line Number



NKCMD - Command code (NKINOUT)

NKLINO - Line number

NKOBP - Output buffer pointer

NKUOPS - Eight user flags (NKUOP1 - NKUOP8). NKUOP1 is bit 15 in the MLCB user flag word; NKUOP8 is bit 8 in that word. NKUOPS is moved into MLCB if NKMVB is 1.

F1 - NKMVB, move user flags to MLCB

F2 - NKRPRT, strip parity flag

1 = strip parity
0 = do not strip parity

NKBLKL - Block length (CC2). Moved into MLCB if nonzero; replaces current MLCB block length

F3 - NKSCENBL, special character flag. If set, move NKSCHR into the MLCB

NKISTAI - Input processing state index

NKISPTA - Input processing state pointers table address (optional address or 0; if 0, NJTECT value is used)

NKSCHR - Special character, moved into MLCB if NKSCENBL flag is set

NKCNT1 - Character count (CC1). If nonzero, this replaces the current character count in the MLCB

NKCXLTA - Code translation table address. If nonzero, this replaces the current translation table address in MLCB

Figure 5-7. Input after Output Command Format

<table>
<tr><th>WORD</th><th>15</th><th>7</th><th>6</th><th>5 0</th></tr>
<tr><td>0</td><td>NKCMD</td><td>F1</td><td>F2</td><td>NKWLINDX</td></tr>
<tr><td>1</td><td colspan="4">NKLINO</td></tr>
<tr><td>2</td><td>NKUSRBY</td><td colspan="3">NKWKCOD</td></tr>
</table>

NKCMD - Command code (NKENDIN)

F1 - NKRELBFS, release buffer flag (release buffer if set)

F2 - NKWKFL, send worklist to user (if set)

NKWLINDX - Worklist index, used if NKWKFLG is set

NKLINO - Line number

NKUSRBY - User-supplied byte, returned in field MMWTCOUNT in worklist

NKWKCOD - User workcode in worklist (MMWKCOD)

Figure 5-8. Terminate Input Command Format

<table>
<tr><th>WORD</th><th>15</th><th>7</th><th>6</th><th>5 0</th></tr>
<tr><td>0</td><td>NKCMD</td><td>F1</td><td>F2</td><td>NKWLINDX</td></tr>
<tr><td>1</td><td colspan="4">NKLINO</td></tr>
<tr><td>2</td><td>NKUSRBY</td><td colspan="3">NKWKCOD</td></tr>
</table>

NKCMD - Command code (NKENDOUT)

F1 - NKRELBFS, releases buffer when flag is set; these are buffers specified in BZLBTOMUX

F2 - NKWKFLG, sends worklist to user when set

NKWLINDX - Worklist index; used if NKWKFLG is set

NKLINO - Line number

NKUSRBY - User-supplied byte to be returned in field MMWTCOUNT in worklist

NKWKCOD - User workcode in worklist (MMWKCO)

Figure 5-9. Terminate Output Command Format

Common Multiplex Subroutines for TIPs

The multiplex subsystem provides a number of common subroutines for the interface programs; these are as follows:

- PMWOLP, the worklist processor on the multiplex level
- PTCLAS, the CLA status analyzer
- PTLINIT, the line initializer
- PMT1SEC, the timing supplier for the output data demand (ODD) function

PMWOLP, MULTIPLEX WORKLIST PROCESSOR

PMWOLP processes each multiplex worklist by workcode type. Most workcodes concern error processing. Workcodes that PMWOLP does not recognize are passed directly to the responsible TIP at multiplex level 2.

If the workcode is a hard error, the line is cleared and input and output are terminated. The terminate input command to the command driver causes the driver to return a worklist to PMWOLP. All hard errors from the line are discarded until the terminate input worklist is received. The input terminated worklist is changed into a hard error worklist (A0HARDERR = MMHARDERR) and the worklist is sent to the responsible tip at OPS level.

If the line is active, all errors, hard or soft, are reported to the CE error file.

The multiplex level workcodes are summarized in table 5-1. The actions that PMWOLP takes in response to the workcodes are as follows:

- MMCLAS - CLA Status. This workcode is generated for selected CLA status words by one of the modem state programs (refer to section 12). PMWOLP calls PTCLAS to analyze the status word. PTCLAS returns information to PMWOLP in three ways: (1) The function is set true if the worklist is to be sent to the TIP, (2) NRCODE is set to nonzero if a CE error is to be reported, or (3) the workcode in the intermediate array is changed to A0HARDERR (or MMHARDERR) if a hard error is found.

- MMOBUX - Output buffer terminated. This is an optional worklist generated by the multiplex firmware after the completion of an output message. If the line is to be turned around, PBTOQUE is called to provide a 200-ms delay. The worklist is passed to the TIP at level 2 either immediately (if the line does not require a turnaround delay) or when the delay timeout period is completed.

- MMBUTCH - Multiplex buffer threshold reached. This worklist is generated by the TIP's input state program 3 (see section 12) when the multiplex firmware notifies that state program that the buffer threshold has been reached. PMWOLP releases any input buffers and stops processing.

- MMCAOR - CLA address out of range. The multiplex firmware reports this error whenever the CLA address is out of range. The CLA is cleared and the error is reported to the CE error file.

- MMUNSOD - Unsolicited output data demand (ODD). The multiplex firmware reports this error when an ODD is received on a line that is not in output state. The error is reported to the CE error file and a hard error is declared.

- MMUNSIN - Unsolicited input. The multiplex firmware reports this error in two cases: (1) a status character is received and input status flag (ISON) is not set, or (2) a data character is received and the input on (ION) flag is not set. In either case, the error is reported to the CE error file and a hard error condition is declared.

- MMIFFO - Input framing error. The multiplex firmware reports this error when it cannot recognize the input frame. The error is reported to the CE error file and no further action is taken.

- MMTIMOD - Modem Timeout. PTCLAS reports this error after the 10-second timeout for dedicated lines has elapsed without a response from the modem. The error is reported to the CE error file and a hard error condition is declared.

- MMINEND - Input terminated. PMWOLP generates this error worklist to itself after the terminate input command is sent to the command driver. The worklist informs PMWOLP that no more worklists will follow. PMWOLP sends a hard error (A0HARDERR) worklist to the OPS-level TIP.

- MMTIMOD - ODD timeout. The multiplex subsystem timing routine (PMT1SEC) generates this worklist when an active output line has not requested a new character (ODD) within the allotted 1-second period. The error is reported to the CE error file and a hard error condition is declared.

- MMFES - Framing error for synchronous lines. PTCLAS generates this error after examining the status word. The error is reported to the CE error file and control is passed to the responsible TIP at multiplex level 2. The TIP should send a command to the command driver to clear this condition.

- MMBREAK - User break on synchronous lines. PTCLAS generates this condition after examining the status word. The user break indicates that the user has requested output to be terminated. The condition is reported to the CE error file and control is passed to the responsible TIP at multiplex level 2.

PTCLAS, CLA STATUS ANALYZER

Analyzing CLA status is a joint task of the modem state programs and PTCLAS. All incoming two-word status entries (8 bits per word) are combined into one 16-bit status word by the multiplex firmware. Control is passed to the responsible modem state program for that line. The modem state program checks for one of the necessary modem signals:

- To initialize or enable the line
- To give control to the TIP's appropriate input state program
- To detect line error conditions

If the modem state program generates a worklist to PTCLAS, PMWOLP calls PTCLAS to analyze the status word. The format of the worklist is as shown:

15 12	11 8	7 0
Line inop code	Status indicator	Workcode
Line number		
Status word		

The line inoperative code is supplied to PTCLAS for the TIP whenever a hard error is detected. When PTCLAS detects a hard error, it changes the workcode to MMHARDERR. The status condition indicator is set by the originator to indicate the type of status that was detected. PTCLAS analyzes the status word and takes one of the following actions:

- Causes control to be given to the line initializer (PTLINIT) or to a TIP
- Causes PMWOLP to request a CE error file entry
- Starts the timeout period for a CLA status overflow condition or for a modem signal loss condition (modem timeout)

See MMCLAS workcode in the PMWOLP subsection, above. Table 5-4 lists the status condition indicators and the action that PTCLAS sets up for PMWOLP.

CLA Status Overflow Handling

Each time a status word is received, the firmware increments a CLA status word overflow counter in the port table (NAPORT). This overflow count is cleared by any of the following conditions:

- Output buffer terminated (OBT) generated
- Terminate input buffer state instruction executed
- Terminate input command issued
- Terminate output command issued

When the counter overflows, the firmware builds a MOOVRT status worklist and turns off input supervision for the CLA. When PTCLAS receives the first status overflow entry, it starts a 10-second timeout period and sets flags in the port table. When the 10 seconds expire, PTCLAS receives control with a MOOVTO worklist from PBTOQUE. PTCLAS resets the overflow counter in the port table, issues a command to turn on input supervision for the CLA, and resets the wait bit. If the timeout occurs before another status overflow is detected by the firmware, status processing continues normally. However, if another overflow entry is received during the timeout period, PTCLAS reports the status overflow to the TIP as a hard error. If at any time there are not enough buffers available to start the timeout, PTCLAS reports the status overflow to the TIP as a hard error.

TABLE 5-4. PTCLAS WORKLIST ANALYSIS AND ACTION

Indicator	Reported By	Meaning	Condition Detected	Action
MOCLAON (0)	Modem state (MSTLNI)	Line initialized	Any status	Control to line initializer
MORING (1)	Modem state (MSTLNI)	Ring indicator	RI status	Control to line initializer
MOENBL (2)	Modem state (MSTENB)	Line enabled	DSR or DSR and DCD status	Control to line initializer
MOHERR (3)	Modem state (MSTCHK)	Hard error	ILE, OLE, INVALID RI, loss of DSR†	Control to TIP (supply INOP code and change workcode)
MOSOER (4)	Modem state (MSTOUT)	Soft output error	NCNA status†	Control to TIP (change workcode)
MOSIER (5)	Modem state (MSTINP)	Soft input error	DTO, FES, loss of DCD status†	Control to TIP (change workcode)
MOSTRT (6)	Modem state (MSTCHK)	Start modem timeout	Loss of DCD on constant carrier line†	Call PBTOQUE to start 15-second timeout
MOSTOP (7)	Modem state (MSTCHK)	Stop modem timeout	DCD status during modem timeout	Cancel timeout
MOOVRF (8)	Firmware	CLA status overflow	Overflow of status counter	
MOOVTO (9)	PBTOQUE (TIMEOUT)	Status overflow timeout	10-Second timer expired	
MOMRTO (A)	PBTOQUE (TIMEOUT)	Modem response timeout	15-Second timer expired†	Refer to control to TIP (change workcode)
MOBREAK (B)	Modem state (MSTINP)	Break condition	FES with null character†	Control to TIP (change workcode)

†C.E. error messages generated on these conditions

Modem Response Timeout Handling

When DCD on constant carrier lines drops, a MOSTRT status worklist is generated by the modem state program, and a bit is set in the MLCB indicating that a modem timeout is in progress. When PTCLAS receives this worklist, it causes a 10-second timeout entry to be generated. If the timeout period elapses before DCD comes up, PTCLAS reports a hard error (modem timeout) to the TIP. If, during the timeout period, the modem state programs receive a status word with DCD set, a MOSTOP worklist is generated for PTCLAS. When PTCLAS processes the worklist, it resets the timeout in progress flags and cancels the timeout. If, at any time there are not enough buffers to start the timeout, PTCLAS immediately reports the condition to the TIP as a hard error.

PTLINIT, LINE INITIALIZER

PTLINIT initializes conditions on a line for input and output operations. The program acts like a TIP and is composed of several subroutines. Figure 5-10 shows the relationship of PTLINIT with other multiplex modules, the service module, timing services, and the TIPs.

Upon receiving control, the line initializer executes the Clear-Initialize-Control sequence. As the initializer is state driven, BZSTATE is set accordingly.

On a dedicated line, a check for CLA on is made before issuing the enable line command. When the line is enabled, the initializer builds a line operational worklist message for the service module and the associated TIP.

For enabling a switched line, three conditions must be met: (1) the ring indicator (RI) must be detected, (2) the host must be up, and (3) buffers must be available. If no RI is present a timer is started. A worklist (line status nonoperational; no ring indicator) is issued if this timer expires before an RI is detected. If buffers are not available or if the host is down, another timer is started. If this timeout period expires, program control is returned to the Clear-Initialize-Control sequence. If the timeout period has not expired and RI is received in a status word, PTLINIT again checks for buffer availability and whether host is up. With an RI present, the host up, and buffers available, the enable line command is issued. Line operational worklists are built for the service module and for the associated TIP.

Error messages are generated under the following conditions:

- A timeout period has expired and a required status has not been detected.

- The status indicates that the line is not operational.

PTLINIT is state driven with each state defined in table 5-5.

PTLMUX2, the multiplex level 2 program, merely passes control by generating worklist entries to PTLINIT. This is reached through PBXFER.

After a line has been enabled, a 1-second delay is made before notifying the TIP. This allows time for line/modem transients to settle.

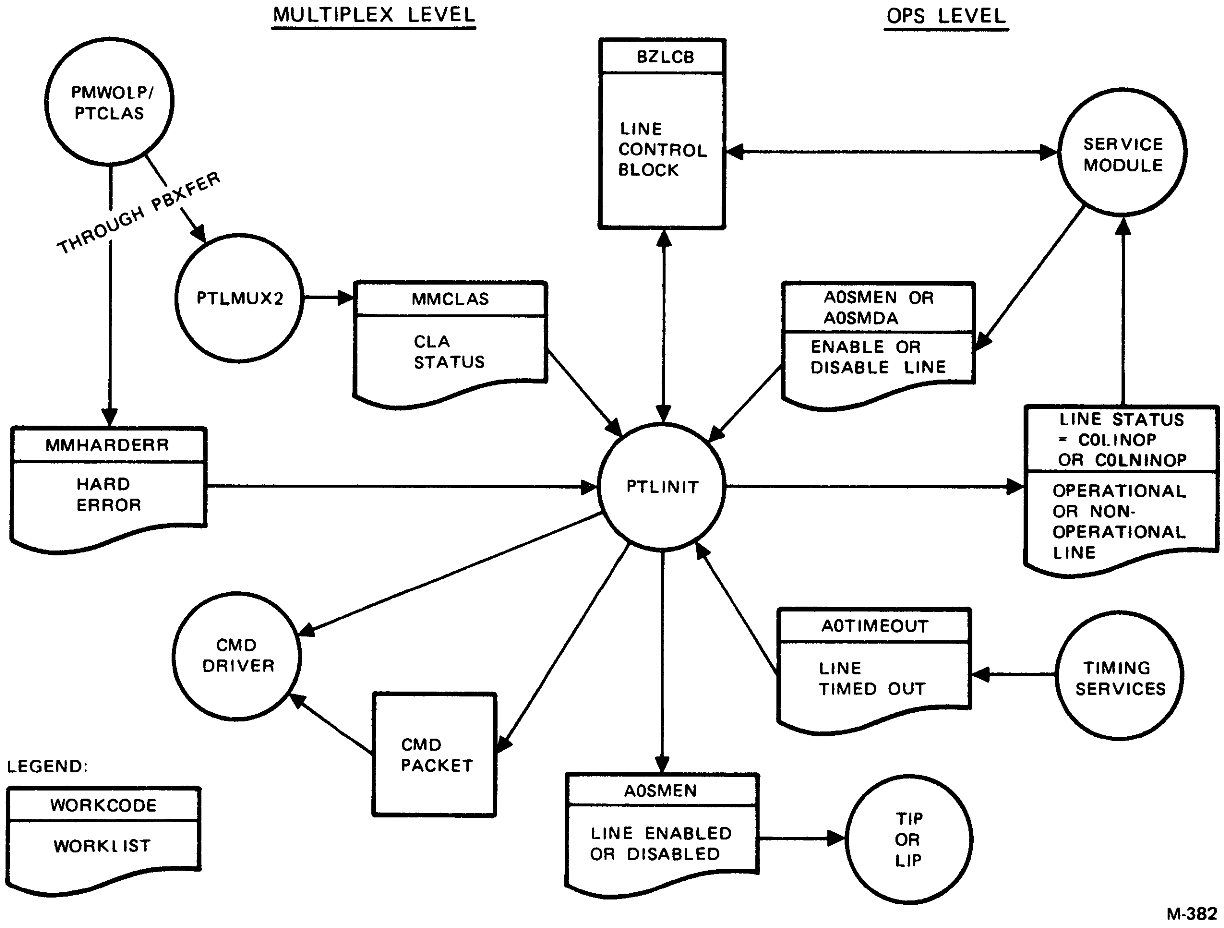


Figure 5-10. PTLINIT Relationships with Major CCP Modules

TABLE 5-5. PTLINIT STATE TRANSITION TABLE

State Event	CLAON	SWCK	SWRING	SWRDY	CLARDY	All States	SWDLY
Status	Ded: Enable Line. State=CLARDY Timer=30 seconds SW: State=SWCK Timer=1 second	Buf Avail/ Host Up Enable Line State=SWRDY Timer=30 seconds Buf Not Avail or Host Down No Operation	Buf Avail/ Host Up Enable Line State=SWRDY Timer=30 seconds Buf Not Avail or Host Down Start Timer, if timer is off	Set Up Timer for 1-second Delay	Timer=0 Autorecog. Send Line Enable-Nonop Msg. Other Send Line Oper Msg. Restore TIP Type.	----- Build WL for TIP Type.	-----
Timeout	Clear Line. Send Inop Message. State= Inactive Timer=0	Send No Ring Message. State=SWRING	Condition Line. State=CLAON Timer=1 second	Disable Line. Clear Line. Send Inop Message. State= Inactive Timer=0	Disable Line. Clear Line. Send Inop Message. State= Inactive Timer=0	-----	Send Enable WL to TIP. Restore TIP Type.
Hard Error	-----	-----	-----	-----	-----	State= Inactive Send Line Inop Message	State= Inactive Send Line Inop Message
Enable Line	-----	-----	-----	-----	-----	Save/Set TIP Type. Condition Line. State=CLAON Timer=1 second	-----
Disable Line	-----	-----	-----	-----	-----	Send Line Disable Message. Clear Line. State= Inactive Timer=0	Send Line Disable Message. Clear Line. State= Inactive Timer=0

PMT1SEC, OUTPUT DATA DEMAND TIMING HANDLER

This program supplies the timing for the ODD function. If 1 second elapses on an active output line without an ODD signal being received, PMT1SEC times the line out. A hardware error is declared by generating a multiplex worklist, which requests an interrupt to process the error.

NETWORK COMMUNICATIONS SOFTWARE 6

This section describes the block protocol and the functions of the network communications software programs. The functions include some command execution (when the service module executes the command), and common TIP subroutines. The virtual terminal formats (IVT and BVT) are also discussed in this section; the virtual terminal transforms are used as a part of the multiplex level (state program) part of the TIPs.

MAJOR FUNCTIONS

The major functions performed by the network communications programs are the following:

- Defines the types of blocks that are acceptable for data transfer, internode and intranode.
- Routes the blocks. This includes checking the validity of incoming blocks and attaching the blocks to an NPU program that will continue processing the block, or reading the block to be queued to the next using network node.
- Provides and processes a special type of block reserved for command, status, and statistics information. All service messages use this kind of block. The modules that process service messages are collectively called service modules. CE error, statistics, and alarm messages are special classes of service messages.
- Provides the ability to alter the interactive virtual terminal formatting parameters.
- Provides standard TIP support programs. These include the point-of-interface (POI) programs and other standard routines that can be used by any TIP.

BLOCK PROTOCOL

Block protocol is used to communicate commands and information between the NPU and the host. Blocks are composed of consecutive bytes. The shortest block consists of only a header (four bytes); the longest block consists of 2047 bytes, including the four-byte header.

Block protocol assumes the logical connection between processes in the host and the NPU is error free (a supportive, lower level protocol provides delivery assurances between the processes). However, the logical connection can be abnormally broken, either process can fail, or the processes can become temporarily congested, leading to regulation of information transfer.

Failure of a process is usually reported by means of a service message. Temporary bottlenecks at a destination process are usually a result of inability to deliver data to an associated terminal or to the host. Block handling provides a standard method for informing the transmitting process of a temporary problem so that any subsequent data transfers on that connection can be held in abeyance until the problem is corrected.

The paths between the two processes are fully symmetrical as shown in figure 6-1. Blocks belong to one of three categories:

- Forward supervision (FS) functions are performed by INIT and RST blocks.
- Reverse supervision (RS) functions are performed by BACK, BRK, STRT, and STP blocks.
- Forward data (FD) functions are performed by BLK, MSG, and CMD blocks.

BLOCK FORMAT

The first two bytes of any block are reserved for a link header (which is used when sending/receiving data from a remote NPU). The next four bytes of any block constitute the block header. Format of the block header is as shown in figure 6-2.

The current release consists of nine principal block types plus an additional assurance control block type used only for NPU to NPU transmissions. Characteristics of each type are summarized in table 6-1.

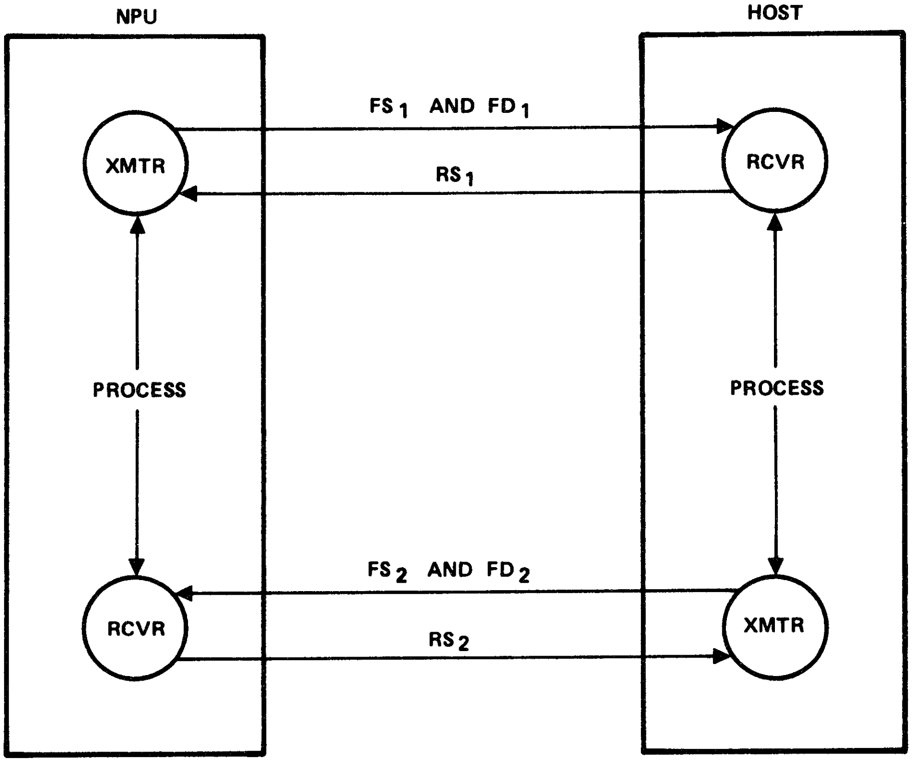
The first three bytes of the block header provide a standard network address. Byte 4 contains the block priority (P), block sequence number (BSN), and block type (BT). The content of the remainder of the block, if any, varies with the block type.

The priority of the block is only significant when the block is required to traverse a network trunk. Priority provides for preferential treatment for high-priority blocks when trunk queueing occurs. (Trunk queueing is a part of priority assignment.) All blocks (regardless of type) containing the same address must be assigned the same priority.

The BSN supplied in a downline block of type MSG, BLK, or CMD must be returned in the BSN field of the upline BACK which acknowledges that block. When a BRK or STP is sent, the BSN field must contain the BSN which was contained in the last BACK sent for this connection. The BSN is always zero on other upline and downline blocks.

Address

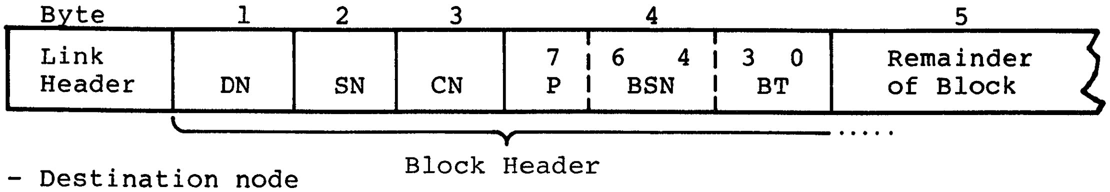
The address contains the node IDs for the source and destination of the block plus a connection number.



FS – FORWARD SUPERVISION (CONTROL/STATUS REQUESTS)
FD – FORWARD DATA (INFORMATION/COMMANDS)
RS – ACKNOWLEDGMENT AND ERROR INFORMATION

M-367

Figure 6-1. Sample Block Data Paths Between NPU and Host



DN - Destination node

SN - Source node

CN - Connection number

P - Block priority for trunk usage

1 = high
0 = low

BSN - Block sequence number (range 0 - 7)

BT - Block type (defined in table 6-1)

Figure 6-2. Block Header Format

NODE

Each NPU has a unique node ID; each interface between a host and an NPU has a unique node ID; the host has two unique node IDs. Node ID = 0 is reserved for the Network Supervisor (NS) in the host. Node ID = 1 is reserved for the Communications Supervisor (CS). The remaining node IDs (between 2 and 255) are build time parameters. For example, in a single-host, single-NPU system, the host interface (coupler of the local NPU) might be node ID two, and the terminal node (interface to the terminals) might be node ID three; this pair of nodes forms a logical link. Thus, traffic going upline (from a terminal to the host) has a destination node ID of two and a source node ID of three. Traffic going downline from NS to the NPU has a destination node ID of two and a source node ID of zero.

CONNECTION NUMBER

A logical connection is the association between a terminal control block (TCB) in a NPU and an application process in the host, by which traffic is communicated between the terminal (or a device at that terminal) and applicable process. The TCB contains all status information relative to a particular terminal (or terminal device) and the current transfer. The TCB also contains a host-assigned connection number. The connection number is one byte long, and has a range of values between 1 and 255. Every block traveling downline to a terminal device or upline from a terminal device bears the connection number of the associated TCB. Unique connection numbers are assigned to all TCBs within a given NPU node, and are associated with a particular host node, i.e., on a given logical link.

TABLE 6-1. BLOCK TYPES

Mnemonic	Name	Block Type	Traffic Type	General Function
BLK	Block	1	FD	Any data block which is not the EOM block of a multiblock message
MSG	Message	2	FD	Data block which is the EOM block of a multiblock message or the only block of a message
BACK	Block Acknowledgment	3	RS	Block acknowledgment for block transmitted in opposite direction
CMD	Command	4	FD	Command
BRK	Break	5	RS	Indicates a discontinuity in the data stream traveling in the opposite direction
STP	Stop	6	RS	Forward data stream is undeliverable and should be stopped
STRT	Start	7	RS	Forward data stream can be started
RST	Reset	8	FS	Transmitter has cleared logical connection after receiving a BRK or STRT
INIT	Initiate	9	FS	Initiate a logical connection
--	--	10 . . . 14		Not used
			Subtype	
ACTL	Assurance Control - used only in local/remote NPU communications	15	0	CLR - Local NPU clears remote NPU at initialization
			1	PRST - Remote NPU acknowledges CLR
			2	REGL - Either end of link changes regulation level
			3	LINIT - Local NPU initializes LINK
			4	LIDLE - LIP at either end of link is idle - LIDLE maintains protocol when no data is being transmitted

SERVICE CHANNEL

A block having a connection number of zero is called a service message, and the logical connection over which it is communicated is called the service channel. Unlike logical connections that can be dynamically created and released, the service channel always exists. Service messages include commands, requests for status, error information, statistics information, or replies to one of these three message categories. The service channel can also be used to send messages between terminals. Commands traveling via the service channel establish logical connections and communicate control, status, and error data. The complete summary of service messages is found in appendix C.

BLOCK TYPES

The block types are described in detail below.

BLK Block

A BLK block is a data block containing a portion, but not the last segment of a data message. All data blocks contain from 1 to 2043 bytes of data immediately following the four-byte header. The content of the data field is determined arbitrarily by the communicating processes.

MSG Block FD, BT = 2

A message is a self-contained unit of data communications. In half-duplex, two-parity communications, the transmitter signals ready-to-receive by sending end-of-message. Thus, a message is a data stream terminated with an end-of-message indicator.

If a message is 2043 bytes or less in length, it can be transmitted within a single MSG block. If a message is longer than 2043 bytes or if, as is usual, the message is segmented by the terminal or because of a desire to optimize NPU dynamic space, all segments but the last are transmitted within BLK blocks. The last segment is transmitted within a MSG block.

Back Block

A BACK block is the acknowledgment of a received block. It is returned to the transmitter by the receiver as BLK, MSG, and CMD blocks are processed to allow the transmitter to adjust the rate of issuing data to the rate of delivery to the receiver. The transmitter should not issue unacknowledged blocks in excess of a network block limit (NBL) for each connection. The BACK block that acknowledges a previously transmitted block allows the transmitter to maintain an outstanding block count to ensure that the NBL is not exceeded. NBL is established by the connection as a part of the configuration process. Note that no data bytes are associated with a BACK block.

CMD Block

A CMD block carries a network command. It allows connected processes to communicate outside of the data stream but synchronous with that stream. The command is received by the destination process in the same ordering sequence to the data stream or other commands as existed at source. If CN is 0, the command is a service message. The data bytes of the message are highly structured. Rather than using BACK blocks as acknowledgment, service messages use other service messages as acknowledgments. See appendix C.

BRK Block

The BRK block indicates a discontinuity (break) in the data stream and travels in the opposite direction. The receiving process responds with an RST to specify the point in the data stream where the BRK block occurred. Block protocol does not retain blocks for retransmission. Instead, the sender of the BRK block discards all blocks received before the RST block. A further BRK or STP block must not be sent before the RST block is received.

A single data byte, the reason code (RC), follows the BRK block header and specifies the reason for breaking the transmission. The RC byte is defined as follows:

1 = User Break 1 received (typically means queue abort occurred)
2 = User Break 2 received (typically means job abort occurred)
3 = Output device not ready
4 = Illegal or invalidly formatted block received from host

STP Block

The STP (Stop) block is similar to the BRK block except that no RST block is sent and no further blocks should be sent until a STRP block is received.

The STP block occurs when a process is unable to deliver data to the final destination such as when a terminal is inoperative or not ready, or when a line is inoperative. A reason code follows the header. This code is passed to the connected process. The sender of the STP block discards all blocks received before the next RST block received (normally caused by a STRT block issued by the sender of the STP block). The RC byte is interpreted as follows:

1 = Terminal busy
2 = Terminal failure
3 = Batch interrupted by interactive input or output

Start Block

The STRT (Start) block is used after a STP block to allow resumption of data flow to the destination sending the STRT block. The receiving process responds with a RST block to invite the connected process to resume data transmittal. No data bytes are associated with this block.

RST Block

The RST (reset) block is sent in response to either a BRK or STRT block. It serves to delimit the data stream and indicate the point in the data stream at which the BRK or STRT block occurred. From the time the BRK or STRT block was sent until the receipt of the RST block, all unacknowledged blocks and all new blocks are discarded. No data bytes are associated with this block.

Init Block

The INIT (initiate) block delimits the new data boundaries when a connection is first made. Newly established connections discard blocks from the logical connection until the INIT protocol is completed. The second end of the connection to be set up immediately sends an INIT block. Upon receipt of the INIT block, the first end to be set up responds with an INIT block and starts accepting blocks over the logical connection. Upon receipt of the responding INIT block, the second end of the connection to be set up also starts to accept blocks over the logical connection. No data bytes are associated with this block.

Bad Blocks Detected by NPU

When NPU software detects a bad block (any block with block protocol fields that contain unexpected or undefined information), the NPU discards the block. If the block is bad for some other reason, a BRK block is sent to the host. If the block is a BLK, CMD, or MSG, no BACK block is sent to the host. For any other block type, no action solicited by the block is taken and it is not acknowledged. The NPU statistics word for block-discarded-to-bad-address is incremented. The header section of a bad block is displayed at the NPU console.

ACTL Block (Assurance Control)

This protocol is not needed for NPU-to-host communications. It is used only to protect data traveling between local and remote NPUs where the possibility of line errors is relatively high.

SEGMENTATION OF BLOCKS

The block is the unit of data that is assured. Blocks are generated by the source node, passed through the network and delivered to the destination node in the order of their generation. One of two possible priorities must be assigned to a block by the source node. Obviously, if ordering is to be preserved, all blocks and all forward supervision block protocol elements on a connection traveling in the same direction must be assigned the same priority.

Block delivery across internodal physical links is performed in a manner that approximates a preemptive resume priority queue dispatch discipline. For this process, blocks transmitted in a link are segmented into subblocks to ensure that an opportunity for preemption occurs at discrete maximum intervals.

Segmentation is of functional concern only to the LIP although implementation considerations dictate that HIP and TIPs and the receive side of the LIP position the data in buffers in a manner that facilitates subblocking. Block priority for blocks arriving from the host coupler is established by the host before setting up the data transfer. The subblock boundary criteria are discussed in the section describing LIPs.

LOGICAL LINK

A logical link is the logical entity that monitors the transfer of data blocks and block protocol elements for all connections between two end points in the network. Unless both ends of a logical link are configured and operational, all such data is discarded and no connections are permitted. When both ends of a host-to-local logical link are configured, the host is notified with a logical link status operational SM immediately; this logical link remains operational until deleted by the host. When both ends of a host-to-remote logical link are configured, the host is notified with a logical link operational SM from the local NPU as soon as a clear/reset exchange occurs between the local and remote NPUs. This logical link becomes inoperative upon a physical link failure, and the host is notified with a logical link status inoperative SM from the local NPU. NS must explicitly delete the logical link. This causes all associated connections to be deleted and all data blocks and block protocol elements for these connections to be discarded. No connections are permitted on the logical link until a clear/reset sequence establishes an operational state again.

The block header format for delivery assurance over the link is as shown in figure 6-3.

SERVICE MESSAGE ASSURANCE ON TRUNKS

When a physical link fails, all blocks to be transmitted on the link are discarded by the link protocol. Any service message that must be protected across a link failure (namely, unsolicited line status SM) is retained by the service module and repeated when the link again becomes operational. While the physical link is inoperative and no alternate path is available, new service messages are retained by the service module.

DATA BLOCK CLARIFIER, DBC

The first data byte of a message is often used as the data block clarifier. In this use, the byte carries additional control information about the data, which is used internally by the TIP. CCP uses two types of data block clarifier as shown in figure 6-4.

For the downline DBC, all TIPs use format effectors. All TIPs check for transparent data, but only Mode 4C and ASYNC terminals can use the transparent (ASCII) output data.

For the upline DBC, transparent data can be used by the ASYNC TIP only; Mode 4 upline transparent data causes the TIP to lock the keyboard. Only the ASYNC TIP uses the cancel character and parity error flags.

DN	SN	CN	Type	Subtype	RL

DN - Destination node

SN - Source node

CN - Connection number

TYPE - Type of block. In this field, bit 7 is the PRID, bits 6 - 4 are reserved for the block sequence number, and bits 3 - 0 designate the BT.

PRID - Priority designator; set for high-priority blocks

BT - Block type. ACTL blocks also use subtype and RL.

Subtype - CLR - Clear = 0. Sent by local end to remote end of a logical link at initialization time; it is repeated until the PRST is received; contains the logical link regulation level in second byte of data field

- PRST - Protocol Reset = 1. Sent by the remote end of a logical link at initialization time after the receipt of a CLR. PRST contains the logical link regulation level in second byte of data field. Normal data blocks are transmitted following a PRST. Local end accepts blocks after receipt of a PRST.

- REGL - Regulation = 2. Sent by either end of logical link when local regulation level changes; contains new logical link regulation level in second byte of data field

- LINIT - Link Initialization = 3. Sent by local end to initialize the link (trunk); is repeated by local end until remote end responds with LINIT. The Local end accepts blocks following LINIT. Link initialization is done initially and after a trunk failure. Remote end sends a LINIT only in response to a received LINIT. RL field is not used.

- LIDLE - Link Idle = 4. Sent by the LIP of both local and remote ends periodically when no data is available to send to the other end so the LIP is able to monitor both directions of data flow for operational status. RL field is not used.

- RL - Regulation load for trunk

Figure 6-3. Block Header Format for Delivery Assurance

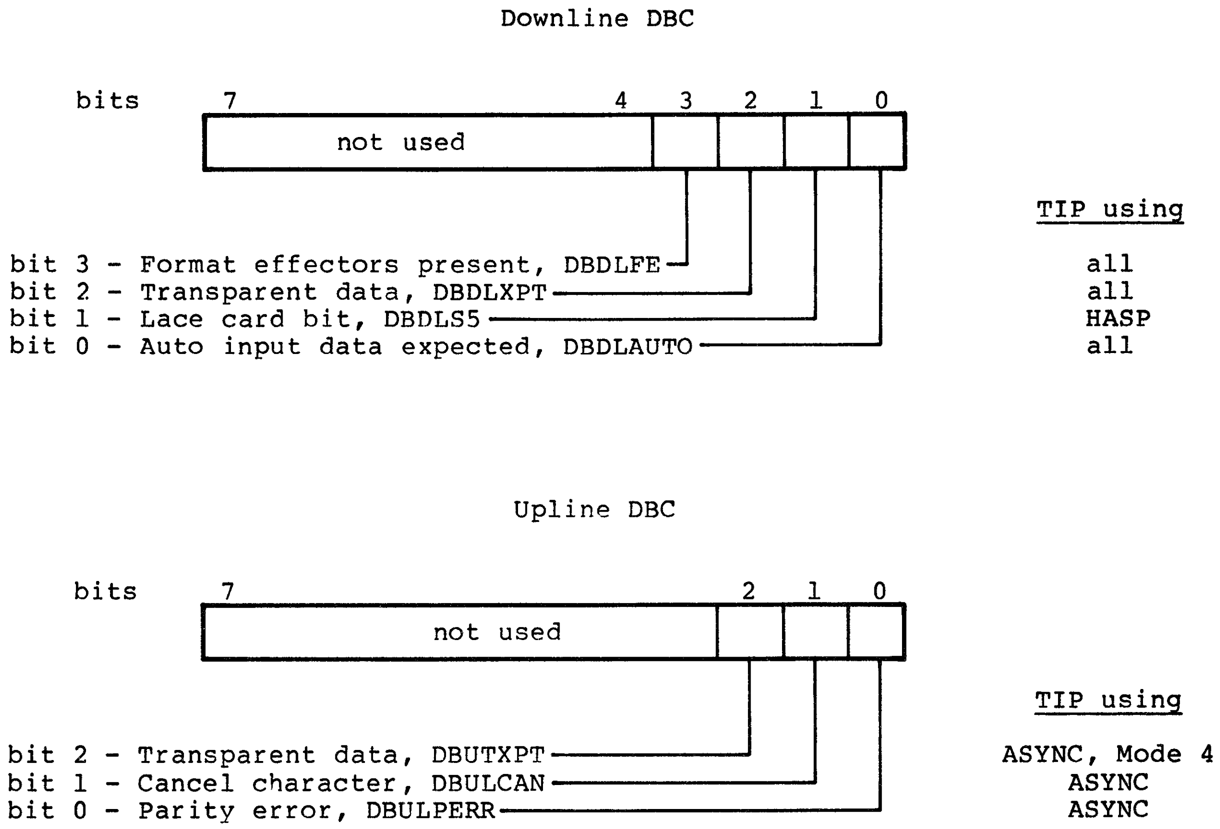


Figure 6-4. Data Block Clarifier (DBC) for CCP

ROUTING

Routing of blocks is performed by the internal processor, usually called through PBINTPRC. The internal processor call is made from the monitor with a worklist entry.

PBINTPRC passes the block to be switched to PBSWITCH, the general systems block switch. PBSWITCH uses the directories to pass the block to the program that must continue processing the block.

Upline blocks that are completely processed are passed to the HIP for transmission to the host. Downline blocks to be sent to terminals are queued to the TCB that is associated with the terminal or device to receive the message.

A second source of switching can use PNROUTE. Only the service module and utilities use this switching method.

CCP provides routing of blocks between nodes and within the NPU node. For example, in a simple system consisting of one host and one local NPU, the node assignments might be as follows:

- For host: NS = node 0; CS = node 1
- For local NPU: coupler = node 2; terminals = node 3

DIRECTORIES

Each block of information (service messages are a special subclass of blocks) has three address elements: The destination node (DN), the source node (SN), and a connection number (CN). There are three directories, one associated with each of the three address elements:

- Destination node directory
- Source node directory (LLCB for the link)
- Connection number directory

The three directories are collectively designated as the routing directories. Formats of the three directories are shown in figure 6-5.

Destination Node Directory

The destination node directory contains an integer value associated with each valid DN address (range is 0 to 255). For a local node (meaning within the same physical node), the directory provides the address of the source node directory associated with that logical node. For all external logical nodes, the directory entry provides a logical link control block (LLCB) address. A zero entry indicates a nonexistent node (an unassigned value of DN).

The destination node directory is a fixed length table with two words per entry. The first word contains the index (by node number), and the second word points to the appropriate LLCB.

Source Node Directory

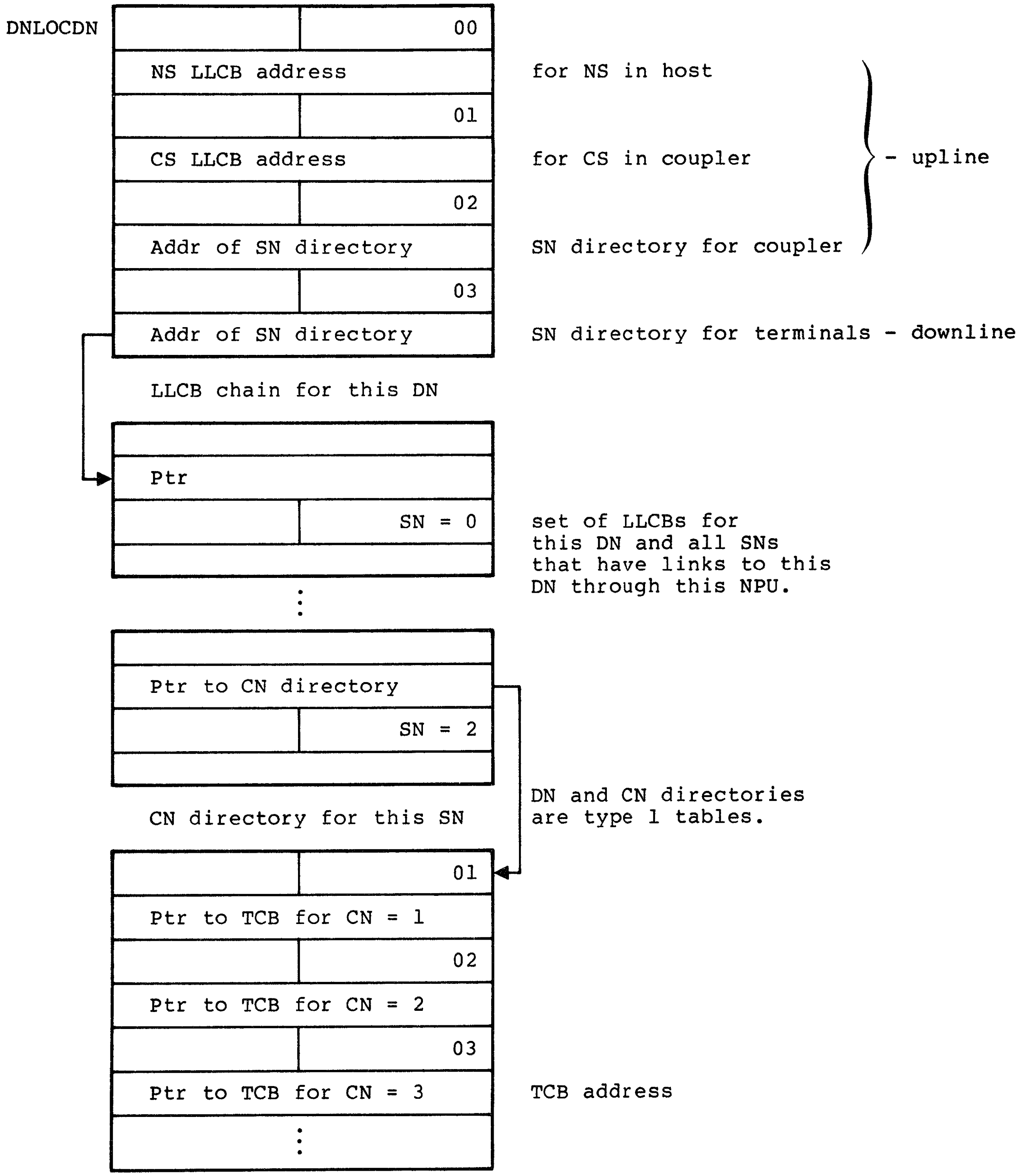
The local logical node has a source node directory for each local node address. Each SN directory is used to select the connection directory associated with the pair of nodes indicated by DN and SN. Nonzero entries point to the address of the connection directory.

Connection Number Directory

For each logical node there is a CN directory for all terminals with which there is at least one connection defined. An entry in the CN directory provides the address of a terminal control block (TCB). The directory is indexed by CN and has a pointer to the TCB for that CN. The CN directory is located in dynamic buffer space.

ROUTING PROCESS

The PBSWITCH module starts the search of the three directories to perform either internode or intranode routine (see figure 6-6).



Note: Directories shown for a one NPU network

Figure 6-5. Routing Directories Formats

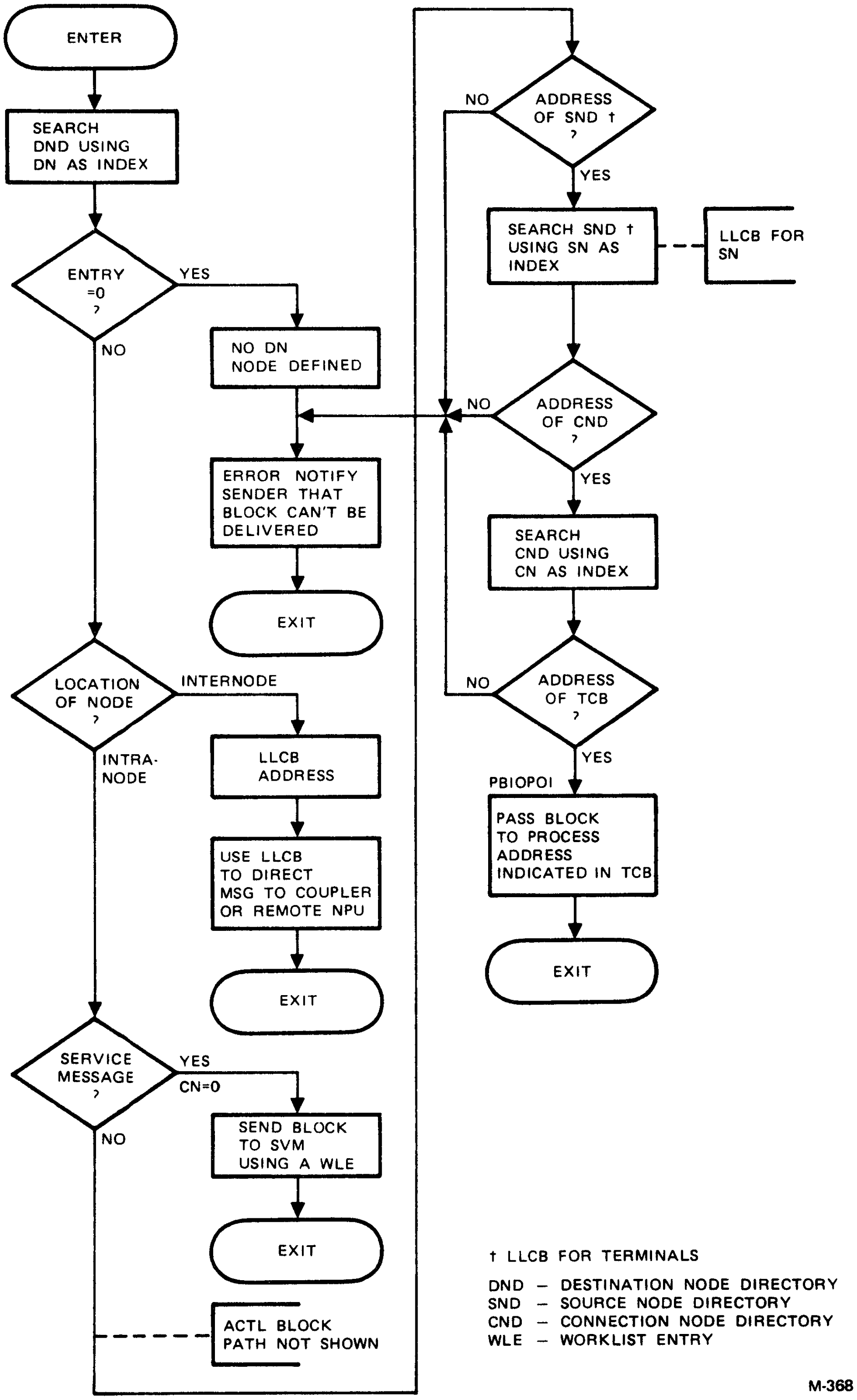


Figure 6-6. Simplified Routing Flowchart for PBSWITCH

Figure 6-6 indicates the steps of the routing search:

DN indexed the destination node directory to obtain an address. If the address obtained is zero, the destination of the block is undefined and PBSWITCH returns an indication to that effect.

If the destination is not a local logical node, the block is passed (as appropriate) to the coupler for a host process or to the remote node. If this is a locally directed service message, the message is passed to the service module using a worklist entry.

If DN is a terminal node, the LLCB for that link is searched using SN. The SN/DN LLCB has a pointer to the CN directory. This directory is similar to the DN directory. It is indexed by CN and has a pointer to the CNs associated TCB. Using the TCB address, PBSWITCH calls the internal output POI (PBIOPOI) which queues the block to the TCB.

ALTERING DIRECTORIES

The modules PNDIRADD and PNDIRDLT add or delete entries to the directories. PNDIRADD requires four input parameters:

- The first two are PASCAL values (ranges to 255) and represent DN and SN values, respectively.
- The third is a PASCAL variable (range 0 to 255) and represents CN.
- The fourth is a PASCAL variable of the buffer pointer type (range 2 - 65, 535) that points to a TCB for use in the appropriate directory.

The DN directory can have a new two-word entry. The CN directory can have new entries and, if necessary, new chained segments. LLCBs (the SN directory) are established when new links are defined. PNDIRDLT removes entries from the DN and CN directories. Three input parameters are necessary:

- The first is a PASCAL value between 0 and 255 and is the index to the DN entry to be removed.
- The second is a PASCAL value between 0 and 255 and is index to the SN entry to be removed.
- The third is a PASCAL variable in the range 0 to 255 and is index to the CN entry to be removed.

If the entry removed in the CN directory is the last remaining entry of that segment of the directory, that segment of the directory is released. Rechaining of directory segments is performed as necessary.

SERVICE MESSAGES

Service messages (SM), the special group of control messages that carry extended command, status, and statistics information between the host and NPU nodes, are processed by the Service Module (SVM). The procedures that make up the SVM are grouped into the following general categories:

- Internal SM processing

- Validating and timing out service messages
- Generating and dispatching service messages
- Configuring, enabling, disabling, and deleting control blocks. These include control blocks for logical links (LLCB), lines (LCB), and terminals (TCB).
- Generating and sending status SMs. These include logical link (trunk), line, and terminal SMs.
- Generating and sending statistics SMs
- Generating and sending broadcast one and broadcast all SMs
- Processing overlay programs and overlay data
- Generating requests for loading an NPU in response to force load SM

TASK SELECTION IN THE SERVICE MODULE

Entry to the SVM is usually made in the form of a worklist. Note that SVM is customarily one of the modules given control by the OPS-monitor with more than one worklist.

Worklist entry switching (PNSMWL) has two levels: On the first level, switching is performed according to workcode. The processed workcodes are:

- COSMIN/COSMOUT - processes or sends most SMs
- COSMDISP - sends a service message
- COLINOP - makes a line operational
- COLNDA - disables a line   Usually done in COSMIN
- CODLTCB - deletes a TCB
- COOVLDATA - processes overlay data

As can be seen, substantially all the processing is done by the COSMIN and COSMOUT codes. The second level of switching takes place in the routines handling COSMIN and COSMOUT. (This is the PFC/SFC level of switching.) A subcode (J4...) is used. Again, almost all processing occurs using one value, the J4DISPATCH subcode.

Within this subcode, the PFC (D8...) and the SFC (D9...) of the SM are used to find an entry in the DBHANDLER table (see appendix E).

The SVM trees (appendix I) show the routines responsible for each SM.

SVM also provides a few direct entries:

- The timed entry call (from PBTIMAL)
- The periodic statistics entry (from PBTIMAL)
- The SM generation, PNSMGEN, which can be used by the TIPs to send any of the eight types of service messages which this routine generates.

INTERNAL SERVICE MESSAGE PROCESSING

Four types of functions are handled by these SVM modules:

- Making worklist entries for SVM and awaiting availability of buffers for SVM processing.
- The interface to the OPS monitor so that the monitor can pass control to SVM.
- An indexing function that finds the proper point in SVM to resume processing after a pause. The necessary marking information is contained in the worklist entry.
- The logic to process the line inoperative and line operative worklist entries. The output is a line enable/disable SM or a status SM.

VALIDATING AND TIMING OUT SERVICE MESSAGES

The timeout group of modules times out SMs and responses to timeout SMs.

The validation group of modules assures that all SMs have:

- A valid primary function code (PFC) and secondary function code (SFC).
- The port identification number is within the range of ports assigned to this NPU.

NOTE

> The format for each type of service message is given in appendix C.

The general format of an SM (appendix C) is shown in figure 6-7.

GENERATING AND DISPATCHING

The following functions are handled by this group of modules:

- DN and SN of the SM are reversed for use in generating the reply SM.
- Queues SM to the local NPU console.
- Releases buffers used for SMs.
- Generates a message from the operator at the NPU console to the network operator (NOP). This process begins when the operator at the NPU console places the console in supervisory mode and enters the message text. There is no response to this type of service message.
- Generates PFC and SFC for service messages.
- Dispatches the SM to:
    1. The HIP if DN designates the local coupler.
    2. The LIP if DN designates the remote node.
    3. SVM if DN designates an action to be performed in this NPU.

Byte		1	2	3	4	5	6	
	Link Header	DN	SN	CN	P/RES/BT	PFC	EB/RB/SFC	Parameters

$\underbrace{\text{DN SN CN P/RES/BT}}_{\text{block header}}$

DN - Destination node

SN - Source node

CN - Connection number is 00 for all service messages; the SM channel is always assumed to be configured.

P - Priority flag; upper bit of block header byte 4

RES - Bits 6 and 5 of block header byte 4

BT - Block type; 4 = command block; lower 4 bits of block header byte 4

PFC - Primary function code

$00\text{-}3F_{16}$ - reserved for network use

$40\text{-}F_{16}$ - reserved for intrahost use (error for CCP to receive these messages)

$A0\text{-}BF_{16}$ - reserved for expansion

$C0\text{-}E0_{16}$ - reserved for network use

$E1\text{-}EF_{16}$ - reserved for installations

EB - Error response SM; EB = 1 (bit 7 of the byte)

RB - Normal response SM; EB = 1 (bit 6 of the byte)

SFC - Secondary function code; see appendix C (bits 5 through 0 of the byte)

Parameters - Defined in bytes. See appendix C.

Figure 6-7. Service Message General Format

CONFIGURING, ENABLING, DISABLING, DELETING CONTROL BLOCKS

This set of modules is used for initiation and changing control blocks for logical links, lines, and terminals. The format and functional effect of these messages are described in detail in the initialization section of the CCP3 Reference Manual and in section 2 of this manual.

GENERATING AND SENDING STATUS SERVICE MESSAGES

This group of modules generates and sends the logical link, trunk, line and terminal status messages. Included in these operations is the ability to count configured NS links and configured CS lines. The status indicates whether the line is operational.

Logical Link Status Request Service Message

This SM status request identifies the nodes comprising the SM link. If the nodes are not specified, the message is treated as a request for the status of all links connected through the NPU.

The response message has a reason code specifying whether the link is operational, a regulation level for the link, and a flag to indicate an unsolicited status reply. The reply also indicates the number of links checked if the message requested information about all the links.

The error response contains only the reason code. Two types of errors are recorded:

- A logical link is not configured.
- Another logical link status SM is already in progress, or the request did not originate from NS in the host.

Trunk Status Request Service Message

This SM status request specifies the port used by the trunk. If the port is not specified, the message is treated as a request for the status of all trunks connected to the NPU. The reply message contains a reason code, such as trunk operational, trunk inoperative, or no ring indicator (for dial-up lines). The reply also contains the line type, configuration states, an identifier for the remote node of the trunk, and the number of trunks checked, if the request was for status on all trunks.

An error response is sent under the following conditions:

- There are no configured trunks or the line number specified is not a trunk.
- Another trunk status SM is already in progress.
- An attempt is made to disable the last path from a remote NPU to NS. Disabling the last trunk would permanently destroy the protocol to the remote node affected when CS records are erroneous or incomplete due to a host failure.

Line Status Request Service Message

This SM status request specifies the port used by the line. If the port is not specified, the message is treated as a request for status of all lines connected to the NPU. A response status SM is sent for each line configured and owned by CS. The reply includes a response code (line operational, line inoperative, or autorecognition/no ring indicator), line type, and configuration state. If an error response is set, the reason code specifies one of the following error states:

- A port is invalid or there is a bad host ordinal.
- Another line status request is in progress.
- An illegal configuration state exists (for a single-line response message).
- No lines are configured (for an all-lines response message).

On a dial-up circuit, a line-enabled response is generated by the NPU immediately following a configure line SM. When a user dials in, the modem interface signals indicate an active line; the NPU then generates an unsolicited line status operation SM, following autorecognition, if applicable. Upon receiving the line status operational SM, the host configures the terminals for the line by sending one or more configure terminal SMs.

An unsolicited line status request SM is sent whenever the TIP senses conditions that cause the line to be inoperative, including normal disconnect on a dial-up line.

Line inoperative is reported when line or modem conditions cause the line to become inoperative; it is not reported if the line is made inactive by terminating its logical connections or by disabling the line.

The following modem signal conditions cause the line to be reported inoperative. The timeouts involved ensure that a line is not declared inoperative because of transient conditions that can be normally expected:

- Data Set Ready (DSR): If the data set ready signal drops at any time, data transmit ready (DTR) is immediately turned off and line inoperative is reported
- Clear to Send (CTS - 201 and 208 modem): If the clear to send signal does not occur within one second of the rise of the ready to send (RTS) signal; remain on for the duration of ready to send, and drop within one second of the fall of ready to send. The data transmit ready signal is then turned off, causing a switched line to disconnect, and line inoperative is reported. Clear to send is not monitored for the 103/113/202 modems.
- Data Carrier Detect (DCD - for full duplex constant carrier): Once a line is operational, if the data carrier detect signal drops and remains off for a period of 10 seconds, data transmit ready is turned off, and line inoperative is reported. Abnormal operation of a data carrier detect on a half duplex or on controlled carrier lines does not influence line status.

TCBs are not automatically deleted when a line becomes inoperative. The host must terminate each logical connection explicitly with a delete terminal SM, or implicitly by sending a delete line SM or a disconnect line SM.

The unsolicited SM also contains bytes defining the number of terminals, the terminal type, the terminal address and the cluster address, the device type, and line speed and code type. For autorecognition responses, the terminal address and device type are repeated for each terminal that can be detected by the TIP. The ASYNC TIP reports only one terminal address or device type pair.

Line Count Request Service Message

The CS sends this message when it requires a count of the line which it owns. This occurs following a host failure or when the NPU causes records to be incomplete or erroneous. The reply message contains the requested count.

Terminal Status Request Service Message

The CS sends this message when its records are incomplete due to a host failure. Status can be requested for one or all terminals on a specified line, the request specifying the line to be checked.

The response can be in answer to a request or it can be unsolicited, when the NPU detects a terminal failure or a terminal recovery. Response parameters are defined in appendix C.

When terminal failure is detected, the correspondent is informed via the logical connection (if any) and the terminal status SM is sent. Terminal failure does not change the state of the TCB with regard to the logical connection, nor is the state of the line (as recorded in the LCB) modified. Operator action is required to delete the terminal if desired.

If an error response is sent, the error is one of the following:

- Invalid line number or bad HO
- No terminals configured
- Line inoperative or not enabled
- Another terminal status request SM is in progress
- LCB not configured

Generating and Sending Statistics Service Messages

Statistics SMs report on the NPU coupler, on lines, trunks and terminals. The statistical data is derived from the appropriate statistics blocks for the coupler, lines, and terminals respectively. The messages are generated periodically or when the counter for the type of failure reaches its overflow level. Statistics messages are also sent when a line is connected or disabled or when a TCB is deleted. The various types of statistics SMs are described in detail in appendix B.

Generating and Sending Broadcast SMs

The network operator (NOP) can send a message to one terminal or to all terminals. These broadcast messages are carried in service messages. This type of message identifies the cluster and terminal addresses, and the device type of the receiving terminal. The network operator produces the text. The procedures for entering this message from the NOP console are given in the NOS Operator's Guide.

A normal response uses a similar format to acknowledge that the broadcast message was received and passed to the specified terminal. If the message was not delivered, an error response is generated. The possible types of errors are as follows:

- Invalid line number, bad host ordinal or toggle bit
- Invalid device type
- Terminal or line not configured
- Terminal or line inoperative
- Host toggle bit error

A broadcast message can be sent to all interactive terminals connected to the NPU. Only the text of the message and the ID of the nodes being used are necessary in the request message. The network operator enters the message at the host console using the procedure outlined in the NOS Operator's Guide.

A normal response is sent when the message is queued to all the interactive terminals connected to the destination NPU; otherwise an error response is sent. Errors are reported in the following cases:

- no logical link established or this logical link is not established
- another broadcast SM is already in progress

PROCESSING OVERLAY PROGRAMS AND OVERLAY DATA

This group handles the overlay logic. Overlays are used for on-line diagnostics in all NPUs, and are used in a local NPU to initialize a remote neighbor NPU.

The same technique is used in either case, and is described in detail in the CCP 3 Reference Manual.

PROCESSING FORCE LOAD COMMAND

The Network Operator has the ability to force an NPU to an inoperative state, so that the NPU requests that it be reloaded.

Receipt of this force load SM causes the CCP to start the deadman timer. When the timer expires, the NPU sends a load request SM to the host. There is no response to the force load SM.

The technique for entering the force load command at the host console is described in the NOS Operator's Guide.

The initialization process resulting is described in the CCP 3 Reference Manual.

CE ERROR AND ALARM MESSAGES

CE error messages are special SMs that report hardware failures. These messages include a one-byte CE error code, and can include additional data. CE error messages are described in appendix B of the CCP Reference Manual.

Alarm messages are special SMs that report frequent errors occurring on a given hardware device and are generated whenever the number of these errors reach a threshold level. Alarm messages are described in detail in appendix B of the CCP 3 Reference Manual.

COMMON TIP SUBROUTINES

These TIP subroutines belong to one of two classes: point-of-interface (POI) routines, and other standard TIP support routines.

POINT-OF-INTERFACE ROUTINES

Five point-of-interface routines are included in the internal processor. These routines handle many of the interfaces for the LIP and TIPs to begin or to end processing of a message. The programs are as follows:

- PBPIPOI - Post input POI
- PBIIPOI - Internal input POI
- PBIOPOI - Internal output POI
- PBPROPOI - Pre output POI
- PBPOPOI - Post output POI

PBPIPOI AND PBIIPOI

PBPIPOI, the post input POI, calls PNSGATH to gather the statistics for the upline message transfer, and then calls PBIIPOI, the internal input POI, to check if a proper connection for the data exists. If not, the buffers are released; otherwise the header is added to the data (chained at the beginning of the blocks, if necessary) and the data buffers are switched to the next processing routine (presumably the HIP).

PBIOPOI – INTERNAL OUTPUT POI

This POI is called to process the output buffers according to block type. It is called from the internal processor switch (PBSWITCH) to route downline blocks to the TIPs. It is also called by the service module to switch broadcast messages.

- BLK, MSG, and CMD blocks are queued to the appropriate TIP if the accept output flag is set. Otherwise, the (chained) buffers are rejected.

- BACK blocks indicate acceptance by the receiving node, so the number of outstanding blocks is decremented and the acknowledged block is released.

- BRK blocks sent upline from the TIP to the host indicate that a transmission was interrupted. This indicates a non-recoverable error. The host aborts the output transmission.

- INIT blocks cause the terminal operating and ready flags to be set.

- RST blocks cause the accept output data flag to be set. Buffers for the current transmission are released.

- STRT blocks sent upline to the host cause the accept input data flag to be set and a RST block to be generated. The host can again send messages downline to the device.

- STP blocks are sent upline by the TIP to indicate that the terminal cannot be used now, but that the message might be transmitted later (after the TIP sends a STRT block). This is used for recoverable cases, such as a printer being currently marked down. STP blocks clear the accept input flag, release the buffers for the current transfer, and notify the TIP to stop processing.

See figure 6-8.

PBPROPOI – PREOUTPUT POI

This POI is used to get a block for output processing. This is done by updating pointers in the output message buffer that is queued to the TIP. The block serial number is extracted also.

PBPOPOI – Postoutput POI

This POI is called from the TIP's postoutput routine to generate the statistics for the block (uising PNSGATH) and to send a BACK block unless the block was internally generated. The POI then releases the buffers holding the message that the TIP has now finished processing.

STANDARD TIP SUBROUTINES

OUTPUT QUEUEING – PBQ1BLK AND PBQBLKS

Output queues are associated with a specific TCB that contains a pointer to the first block in the queue, specifically to the first buffer of that block. Figure 6-9 illustrates the queue structure. The queue contains one or more data blocks, each of which is composed of one or more buffers. The buffers are linked in the order they are removed from the chain. The last word of one buffer is the pointer to the next buffer. The last word of the last buffer contains NIL.

Blocks are chained together using the QCHN word of the buffer header (word 3 of the data buffer header). New blocks are always chained to the previous last block. The QCHN word of the newest block is always NIL.

The TCB output queue is built by two routines: PBQ1BLK and PBQBLKS:

- PBQ1BLK (parm) uses the parameter (block address) to clear the chain word of the block to be queued, then PBQ1BLK calls PBQBLKS.

- PBQBLKS (parm 1, parm 2) uses parm 1 to find the TCB output queue and parm 2 to find the buffers to be added to the chain. If the TCB queue is empty, a worklist entry is made to the TIP that controls the TCB, so the TIP can process the queue.

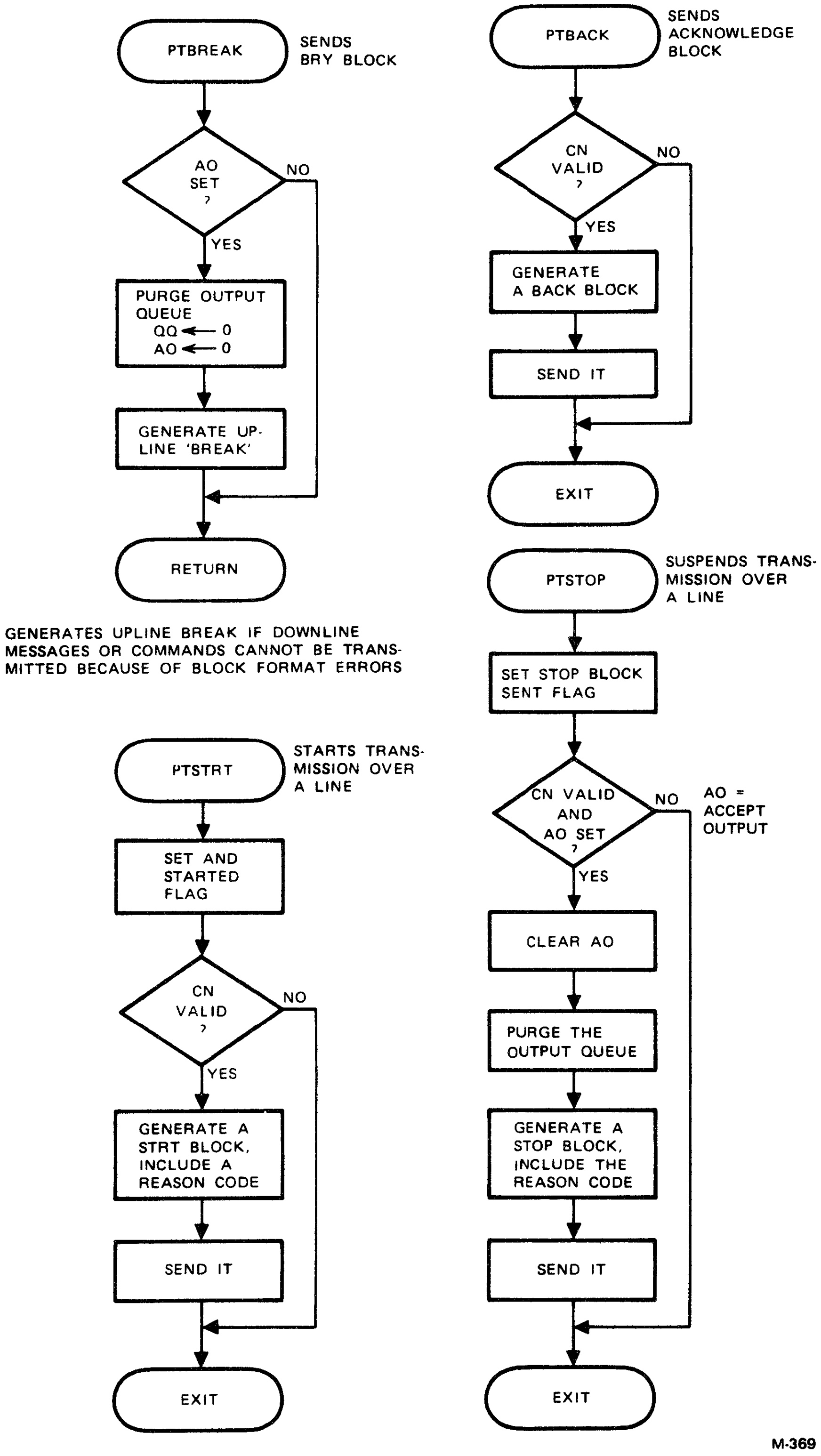


Figure 6-8. Flowcharts for Important Common Tip Subroutines (sheet 1 of 2)

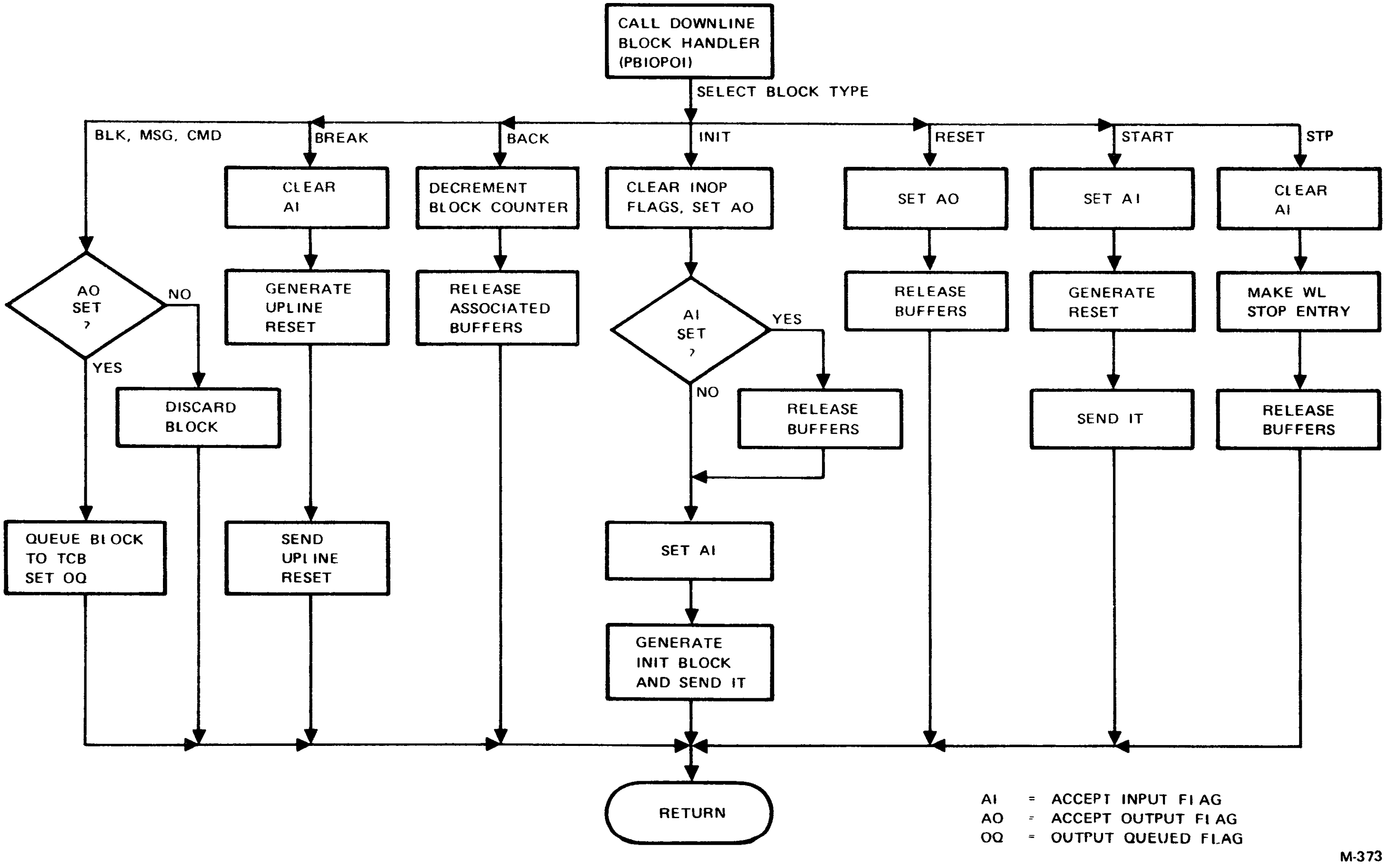


Figure 6-8. Flowcharts for Important Common TIP Subroutines (sheet 2 of 2)

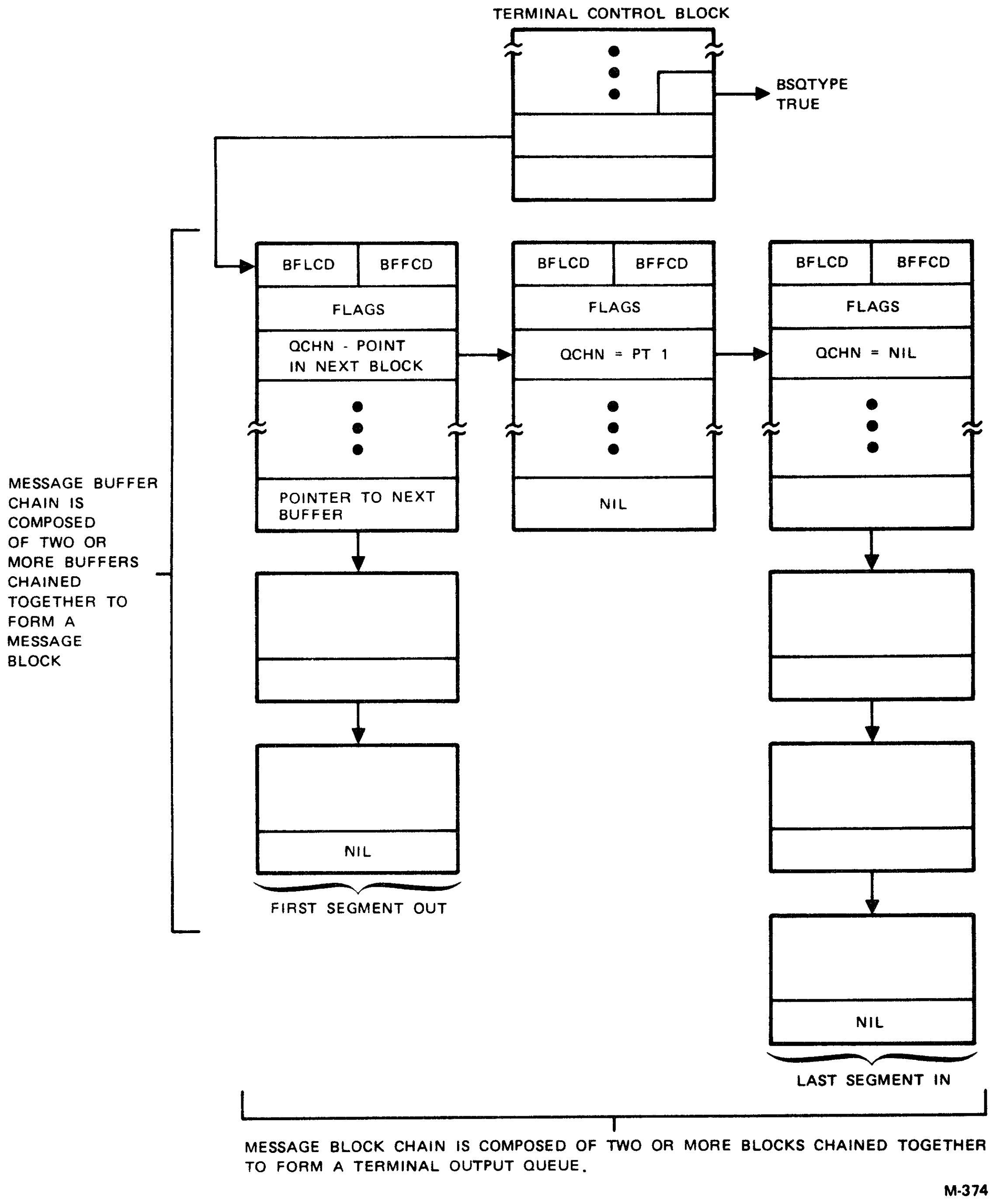


Figure 6-9. Structure of a TCB Queue

UPLINE BREAK – PTBREAK

The common send break subroutine PTBREAK (figure 6-8) indicates a discontinuity in the output stream. This routine purges the output queue described above, sets AO to zero to prevent further queueing of output information, and sends an upline BREAK block with a code indicating the reason for the break.

DOWNLINE BREAK

The host commands the TIP to stop input by sending a downline stop message (a type of CMD block). This block is acted upon, when received, without being output queued. The TIP replies with an input stopped message (also a type of CMD block). This message causes the accept input (AI) flag to be set to zero. To restart input, the host sends a start input message (a type of CMD block). This sets the AI flag to 1 and the TIP again accepts input from the terminal.

STOP TRANSMISSION TO A TERMINAL – PTSTOP

A TIP calls PTSTOP with a stop reason code. PTSTOP clears the accept output flag in the TCB and then calls PNDNABRT to clear the output queue for this terminal. PTSTOP also generates a STP block and includes the reason code for the stop. The internal processor sends the block to the host via the HIP.

INTERFACE TO TEXT PROCESSING FIRMWARE – PTTPINF

A TIP calls this interface to firmware routine to execute the upline or downline text processing state programs. Upline text processing is used only by TIPs which require two-state input processing, such as the HASP TIP. The call is

```
PTTPINF (parm)
```

where parm is the address of the TPCB.

Text processing occurs on the firmware level. Information exchange between OPS-level and firmware level uses the 32-word text processing control block (TPCB). Prior to the call to PTTPINF, the TIP sets all information necessary to execute the transfer into the MLCB. When PTTPINF is put into control, it transfers the second 16 words of the TPCB to the microprocessor file 1 registers to speed processing. The text processing state programs can save information for the OPS-level TIP either in the file 1 registers or in any other MLCB field. After the text processor (using the terminal-oriented text processing state programs) has converted the data, control returns to PTTPINF which stores the current file 1 register values in words 16 - 31 of the TPCB. After escaping to firmware processing, TPPTINF periodically returns to OPS level to process interrupts (interrupts are inhibited while firmware is executing state programs). When the entire text processing sequence is completed, TPPTINF returns control to the calling TIP. If the text could not be converted, TPPTINF notifies the TIP of the failure by using fields in the TPCB.

This module is technically a part of the base system but is discussed here since it provides a service for the TIPs.

FINDING NUMBER OF CHARACTERS TO BE PROCESSED – PTCTCHR

PTCTCHR counts the number of characters in the buffer to be processed. This count includes the complete chain of data buffers in the message. This module is also considered a part of the base system.

SAVING AND RESTORING LCBs – PTSVxLCB AND PTRTxLCB

Two sets of routines allow TIPs to mark transmissions that must be suspended until further terminal or host action occurs. The suspension address in the TIP controlling the transfer is saved in the LCB, and upon the necessary action being completed, control returns to the TIP at the specified point and transmission processing continues.

- PTSV1LCB or PTSV2LCB saves the TIP return address in the LCB and saves a wait count prior to returning control to the monitor. PTSV1LCB is used for input; PTSV2LCB is used for output. The TIP will later receive control by a worklist entry to continue processing at saved address.

- PTRT1LCB or PTRT2LCB - The TIP for this suspended transmission receives control as a result of a worklist entry to it. These routines restore TIP processing at the address (next entry point) saved by PTSVxLCB. PTRT1LCB is used for input; PTRT2LCB is used for output.

These modules are also considered a part of the base system.

COMMON RETURN CONTROL ROUTINE – PTRETOPS

PTRETOPS is called by a TIP in order to properly relinquish control to the monitor (PBMON). This module is also considered a part of the base system.

COMMON TIP REGULATION – PTREGL

The common TIP regulation checking routine is called when the TIP is ready to start processing the data (upline or downline). Even though some processing of the data may already be completed (for instance, input state processing being complete on upline data), CCP may need protection from an additional request for space or processing resources.

At the TIP's request, PTREGL checks any one or any combination of the following four regulation conditions:

- The regulation level at this end of the logical link is higher than the priority level of the block transmitted to this NPU.

- The allowable number of blocks that can be queued to this TCB (ABL) is greater than the number of blocks already queued to this TCB for processing (OBL).

- The accept input (AI) flag is not set in the TCB (upline data).

- The buffer availability level in this NPU is below the level set for this type (low or high priority) of data blocks.

NOTE

> This routine is not called by the multiplex subsystem for upline data. Instead, upline data is accepted from the input loop, stored in the CIB, and demultiplexed into a line-oriented input buffer; then the TIP is called. The TIP has the responsibility for checking whether the message should be rejected (regulation occurs). The mechanism for stopping input at the external interface is also a TIP responsibility. This is done by breaking the message (input stopped or BRK block) and commanding the multiplex command driver to turn off the CLA. Until the CLA state is changed, the multiplex subsystem must continue to accept input data.

The calling format is PTREGL (parm1, parm2). Parm 1 is a pointer to the buffer associated with the proposed input operation. Parm 2 is the type of comparison to be made.

If the type of regulation checked does not currently exist, PTREGL passes a no regulation flag to the caller.

PTREGL is also considered a part of the base system.

SAVING AND RESTORING REGISTERS

Two subroutines save and restore the R1 and R2 registers.

PBBEXIT – Save R1 and R2

PBBEXIT is used to save R1 and R2 before executing the GOTO (EXIT) when the GOTO statement occurs within one or more executable WITH statements.

NOTE

> A GOTO (EXIT) from within a noninterruptable program does not perform an UNLOCK operation before exiting.

PBEXIT then restores R1 and R2.

PBAEXIT – Restore R1 and R2

PBAEXIT is used before a GOTO (EXIT) is executed from within one or more executable WITH statements. PBBEXIT has previously saved R1 and R2 in a specified area so that they may be used as base addresses of the structures associated with the first two executable WITH statements. The calling sequence is

PBAEXIT (parm)

where parm is the name of the two-word save area for R1 and R2.

VIRTUAL TERMINAL TRANSFORM

Virtual terminal format allows the host application programs to expect only two types of input: ASCII input from a standardized interactive terminal (IVT), or ASCII input from a standardized batch terminal (BVT).

Each TIP is responsible for converting from terminal code and format to and from the ASCII virtual terminal formats. Downline, this is handled entirely in text processing state programs (see section 12). If the TIP handles several types of terminals, it must have state programs to handle the conversions for each separate type of terminal.

Upline, TIPs can use either of two ways of converting data. Usually, input state programs can be used to completely demultiplex data from the circular input buffer, to convert format, and to translate code in a single operation (one pass processing). In cases where the upline block of data from the terminal may be composed of data from several terminal devices, this single stage input state processing is impractical. Instead, the multiplex subsystem first uses input state programs for this TIP to gather all the data into an input block for the line. Then after the TIP is called at OPS level, the TIP provides a separate set of upline text processing state programs to finish demultiplexing the data into blocks for each device. At the same time, the upline text processing state programs convert format to BVT or IVT, and translate code to ASCII (two stage input character processing; used by the HASP TIP).

IVT and BVT can be considered as a special subset of the normal host/NPU block protocol.

BVT is handled entirely by the state programs within the TIPs. Most IVT transforms are handled the same way; however, IVT parameters can be varied within a narrow range. For this reason a common TIP routine, PTIVTCMD, is provided to decode the operator (or host)-entered message that changes the IVT parameters (PTIVTCMD calls PTIVTPRSR to parse the message containing the new IVT parameters).

Since the techniques used to format for IVT and BVT differ, the two types of terminals are discussed individually.

BATCH VIRTUAL TERMINAL (BVT)

Batch Virtual Terminal provides the standard interface which permits application programs in the host to exchange information with remote batch terminals without regard to specific terminal characteristics.

The additional block handling abilities needed for batch-type terminals are as follows:

- Ability to transform data to and from BVT format
- Ability to handle block protocol for each type of 9 blocks that can be passed over the host and local NPU interface

Batch Virtual Terminal Characteristics

The BVT is deemed to be a multi-device terminal operating remotely from the host. The BVT is connected to the 255X by a synchronous medium using a high-speed line. Although the protocol on the line may differ by equipment type, the BVT is assumed to be a block oriented terminal.

A separate logical connection exists for each device supported. Device types that may exist at the remote site include: card readers, printers, plotters, and card punches. The BVT is defined to allow full use of the features of Mode 4 terminals.

Features considered are: data compression, printer carriage control, code conversion, transparent data mode control, and file structure. For downline blocks, the host process ensures that downline network blocks do not exceed the allowable device block size after processing by the TIP, and that output print lines do not exceed the device printline width. Similarly, the host process is responsible for compressing data. For downline data, only, blank, zero, and duplicate character compression is permitted. Compression duplicate characters other than blanks or zeros will cause a rejection in the form of a BRK block, if such data is sent to a Mode 4 terminal - (HASP workstations, however, accept duplicate character compression). The degree of upline compression is determined by the terminal. Full compression is assumed. At any multidevice terminal the interactive devices conform to IVT and the batch devices to BVT.

BVT Block Protocol Usage

- BLK Blocks - BLK blocks transfer non-last blocks of input or output messages. The size of the upline block is determined by the terminal. It is a host responsibility to ensure that the size of the downline block does not exceed the terminal buffer size, after the protocol envelope has been added. The TIP attempts to deliver all blocks to the terminal. The effect of delivering too large a block differs according to terminal type.

- MSG Blocks - Message blocks transfer the last or only block of an input or output message. An upline message block is generated whenever an end-of-information (EOI) is encountered in the card stream. The EOI is designated by the < END OF INFORMATION > sequence. A downline MSG block designates the end of a host message.

NOTE

The < > symbols are used for delimiting elements of the IVT/BVT format.

- BACK Blocks - A BACK block acknowledges delivery of BLK, MSG, or CMD blocks, for purposes of flow control.

- BRK Block - A break block temporarily stops the data flow when an operator action occurs (interactive devices have precedence over batch devices) or when a printer-not-ready condition is detected. The application program is responsible for restarting the flow. A BRK block is sent upline when the TIP receives a block that does not conform to BVT or IVT.

- STP Block - A stop block stops the data flow when the end device becomes inoperative or otherwise incapable of accepting more data. The source process is required to protect all data which has not been acknowledged by a BACK block and to prevent new data from being sent to a device unable to accept it.

- STRT Block - A start block cancels the effect of the STP block. The source process must respond with an RST block; then the source may resume sending data.

- RST Block - A reset block indicates the point at which a BRK or STRT block affected the message block stream. A destination process issuing a BRK or STP block discards all unacknowledged blocks, as well as all new BLK, MSG and CMD blocks, until an RST is received. Additional BRK or STP blocks cannot be issued until the RST block for the previous BRK or STP block is issued.

- CMD Block - A command block causes a change of mode in the other process. A CMD block which is to affect data in the opposite direction will not take effect until all data in the same direction ahead of it has been processed. A CMD which is to affect data in the same direction affects any data in the stream that follows the CMD block.

Table 6-2 defines the MESSAGE contents of the blocks to the level needed for BVT processing. Symbols used in the table are as follows:

- PARAM indicates a necessary parameter in the message block.

- $\left\{\begin{matrix}\text{PARAMa}\\ \vdots \\ \text{PARAMn}\end{matrix}\right\}$ indicates one necessary parameter chosen from a list of possible parameters.

- $\left\{\text{(PARAM)}\right\}$ . . . indicates that a parameter is necessary or permitted at a certain place in the message stream; for instance a single MODECHANGE is allowed ahead of a physical record in an UPLINEDATA message block.

Data control bytes have several parameter names: MODECHANGE, COMPRESSEDDATA, etc. These control bytes have a common generic format: $FFnn_{16}$ where nn ranges between 00 and FF_{16}. These values are listed together in a subtable.

A sample of the use of table 6-2 is shown in figure 6-10.

Table 6-3 defines the values for the parameter FORMSCONTROL which specifies the print control action for the BVT.

Figure 6-11 shows job stream card examples for BVT data handling.

TABLE 6-2. BVT BLOCK SYNTAX (HOST/COUPLER INTERFACE)

MESSAGE	=	$\begin{bmatrix}\text{CONTROL}\\ \text{DATA MESSAGE}\end{bmatrix}$
CONTROL	=	$\begin{bmatrix}\text{DOWNLINE CONTROL}\\ \text{UPLINE CONTROL}\end{bmatrix}$
DOWNLINE CONTROL	=	NETWORK HDR COMMAND $\begin{bmatrix}\text{STOP INPUT}\\ \text{START INPUT}\end{bmatrix}$
UPLINE CONTROL	=	NETWORK HDR COMMAND INPUT STOPPED REASON CODE
NETWORK HDR	=	NETWORK ADDRESS PRI BSN
NETWORK ADDRESS	=	DN SN CN
PRI	=	$\begin{bmatrix}0\\ 1\end{bmatrix}$ Priority $\begin{cases}\text{0 - low}\\ \text{1 - high}\end{cases}$
BSN	=	$\begin{bmatrix}0\\ 1\\ 2\\ .\\ .\\ 7\end{bmatrix}$ Block Sequence Number see block protocol description at beginning of section 6
COMMAND	=	4
STOP INPUT	=	PFC - C1_{16} SFC = 05
START INPUT	=	PFC = C1_{16} SFC = 06
INPUT STOPPED	=	PFC = C1_{16} SFC = 07
REASON CODE	=	$\begin{bmatrix}00\\ 01\\ 02\\ 03\end{bmatrix}$ $\begin{cases}\text{00 - Stop input response}\\ \text{01 - Input device not ready}\\ \text{02 - Card slip error}\\ \text{03 - E01 input}\end{cases}$
DATA MESSAGE	=	(BLKBLOCK)_{0-n} MSGBLOCK
BLKBLOCK	=	NETWORK HDR BLK DBC $\begin{bmatrix}\text{UPLINEDATA}\\ \text{DOWNLINEDATA}\end{bmatrix}$
MSGBLOCK	=	NETWORK HDR MSG DBC (ENDOFINFORMATION) $\begin{bmatrix}\text{UPLINEDATA}\\ \text{DOWNLINEDATA}\end{bmatrix}$
BLK	=	01 } See block protocol
MSG	=	02 }
DBC	=	$\begin{bmatrix}\text{UPLINEDBC}\\ \text{DOWNLINEDBC}\end{bmatrix}$
UPLINEDBC	=	00
DOWNLINEDBC	=	SPARE SPARE SPARE NOTUSED NOTUSED NOTUSED BANNERCARD NOTUSED
BANNERCARD	=	$\begin{bmatrix}00\\ 01\end{bmatrix}$ $\begin{matrix}\text{Don't punch banner card}\\ \text{Punch banner card}\end{matrix}$
UPLINEDATA	=	$\begin{bmatrix}\text{(MODECHANGE) (COMPRESSEDDATA)}\\ \text{ENDOFMEDIA (ENDOFRECORD)}\end{bmatrix}_{0-n}$

TABLE 6-2. BVT BLOCK SYNTAX (HOST/COUPLER INTERFACE) (Contd)

DOWNLINEDATA = $[$(MODECHANGE) (FORMSCONTROL) (COMPRESSEDATA) ENDOFMEDIA (ENDOFRECORD)$]_{0-n}$

A single MODECHANGE is allowed ahead of a physical record. FORMSCONTROL is required ahead of each print line. COMPRESSEDDATA may be elided, e.g., FORMSCONTROL without print. ENDOFMEDIA is required at the end of each physical record. ENDOFRECORD and ENDOFINFORMATION are used to indicate logical record or file boundaries.

HEX Value	Parameter	Use
FF00-FF09	MODECHANGE	Data Modes
FF0A-FF0F	ENDOF...	Information Separators
FF10-FF2F	COMPRESSEDBLANKS	Compressed Blanks
FF30-FF3F	COMPRESSEDZEROES	Compressed Zeros
FF40-FF8F	COMPRESSEDDATA	Compressed Data
FF90	STRINGINDICATOR	Uncompressed String Terminated by FF_{16}
FF91-FFCF	STRINGLENGTH	Uncompressed String of Length 1 through 63
FFD0-FFDF	--	Not Used
FFE0-FFFE	FORMSCONTROL	Forms Control
FFFF		Data Character FF

MODE CHANGE = $\begin{bmatrix}\text{ASCII-029}\\\text{ASCII-026}\end{bmatrix}$

ASCII-029 = $FF00_{16}$

ASCII-026 = $FF03_{16}$

Each device type supported by the BVT is assigned a data mode (see device type subtable, below) which, in most cases, is unchangeable. However, downline data to a card punch may contain a MODE CHANGE requesting the TIP to perform the appropriate code translation to generate the desired punched cards. The mode selected stays in effect until the next MODE CHANGE or an ENDOFINFORMATION, which causes the data mode to be returned to the default for the device. For all other downline data and all upline data, MODE CHANGE is ignored.

ASCII-029 indicates that the data should be interpreted as ASCII, but that only the 64 character subset will appear. The data will be translated by the TIP to produce 029 cards. Similarly, ASCII-026 will produce 026 cards.

TABLE 6-2. BVT BLOCK SYNTAX (HOST/COUPLER INTERFACE) (Contd)

DEVICE DATA MODES SUBTABLE

Device Type	Data Mode
Card Reader	Data is always converted to the 64-character subset of ASCII by the TIP based on the characteristics of the card reader and/or from information punched on the job and end-of-record cards in the input stream.
Line Printer	Data is always sent to the TIP in the 64-character subset of ASCII and is translated by the TIP to produce the terminal's standard graphics.
Card Punch	Data is always sent to the TIP in the 64-character subset of ASCII and, by default, is translated by the TIP to produce 029 cards. A MODE CHANGE can be used to request that 029 or 026 be punched.
Plotter	Data is always sent untranslated by the TIP to the plotter.

$$\text{FORMSCONTROL} = FF_{16} \begin{bmatrix} E0 \\ E1 \\ . \\ . \\ . \\ FD \\ FE_{16} \end{bmatrix}$$

Forms control associated with each print line. See table 6-3 for definition of values. Forms controls which are not supported by a specific device results in a single space. See individual TIP actions for implementation.

$$\text{COMPRESSEDDATA} = \begin{bmatrix} \text{COMPRESSEDZEROES} \\ \text{COMPRESSEDBLANKS} \\ \text{REPLICATIONCOUNT} \quad \text{BYTE} \\ \text{STRINGLENGTH} \quad \text{STRING} \\ \text{STRINGINDICATOR} \quad \text{STRING} \end{bmatrix}_{\text{1-n words}}$$

$$\text{COMPRESSEDZEROES} = FF_{16} \begin{bmatrix} 32 \\ 33 \\ . \\ . \\ . \\ 3E \\ 3F_{16} \end{bmatrix} \begin{array}{l} 32_{16} - \text{ 2 zeroes compressed} \\ 33_{16} - \text{ 3 zeroes compressed} \\ \\ \\ \\ \\ 3F_{16} - \text{ 15 zeroes compressed} \end{array}$$

$$\text{COMPRESSEDBLANKS} = FF_{16} \begin{bmatrix} 12 \\ 13 \\ . \\ . \\ . \\ 2E \\ 2F_{16} \end{bmatrix} \begin{array}{l} 12_{16} - \text{ 2 blanks compressed} \\ \qquad . \\ \qquad . \\ \qquad . \\ \qquad . \\ \qquad . \\ 2F_{16} - \text{ 31 blanks compressed} \end{array}$$

BYTE = (0..255)
00 through FF_{16} (8-bit byte)

TABLE 6-2. BVT BLOCK SYNTAX (HOST/COUPLER INTERFACE) (Contd)

REPLICATIONCOUNT = $FF_{16} \begin{bmatrix}42\\43\\\cdot\\\cdot\\\cdot\\8E\\8F_{16}\end{bmatrix}$ Second byte represents the number of times the byte following the count is to be repeated. Value may range from 2 $(42)_{16}$ to 79 $(8F)_{16}$. Upline compression is determined by terminal; full compression capability should be assumed. Not used for downline blocks.

STRINGINDICATOR = $FF90_{16}$ Used for upline only, this indicates that the following byte string consists of uncompressed data of indeterminate length. The string is terminated by the first non-data FF_{16} encountered. Any data FF_{16} patterns must be doubled by the TIP and the added FF_{16} must be deleted by the host.

STRING = $BYTE_{1-n}$ bytes - n is limited by the physical record length of the terminal device.

STRINGLENGTH = $FF_{16} \begin{bmatrix}91\\92\\\cdot\\\cdot\\\cdot\\CE\\CF_{16}\end{bmatrix}$ This indicates that the following byte string consists of uncompressed data of length 1 (91_{16}) through 63 (CF_{16}). This method of representing uncompressed data is always used downline but is used upline only when a count is provided by the terminal, such as HASP.

The following three elements allow file structure to be retained during transfer.

ENDOFMEDIA = $FF0A_{16}$ - This represents the end of a physical record, for instance: card, print line.

ENDOFRECORD = $FF0B_{16}$ nn $FF0A_{16}$ - This represents the end of a logical record and may occur at other than block boundary.

nn = logical record level number

ENDOFINFORMATION = $FF0CFF0A_{16}$ - This occurs only in a MSG block as the last four characters in the block.

Use of BVT Block Syntax table: Example to generate an upline, input stopped because of card error, BVT block.

MESSAGE = CONTROL
CONTROL = UPLINE CONTROL
UPLINE CONTROL = NETWORK HDR COMMAND INPUT STOPPED REASON CODE
NETWORK HDR = NETWORK ADDRESS PRI BSN
NETWORK ADDRESS = DN SN CN - Codes of destination source, and connections are given earlier in section 6
PRI = 0 - low priority assigned to batch terminals
BSN =
COMMAND = 4 - CMD type of block
INPUT STOPPED = C1 - 7_{16} - see appendix C for the primary and secondary function code assignments
REASON CODE = 2

Formatting the syntax into a byte format:

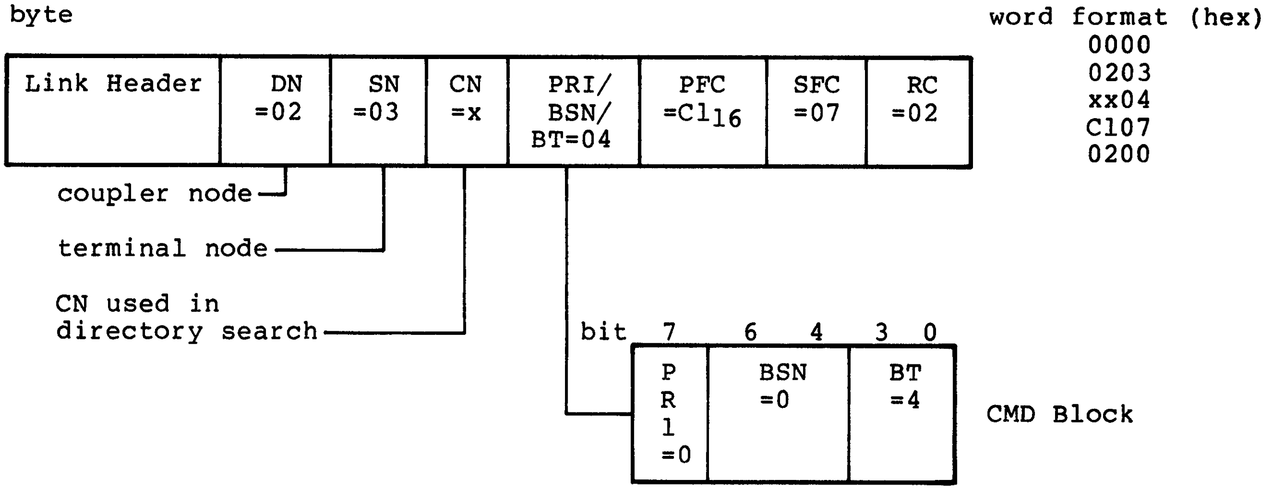


Figure 6-10. Use of the BVT Block Syntax Table

TABLE 6-3. FORMSCONTROL VALUES FOR BVT BLOCKS

FORMSCONTROL (Hex)	Action Before Printing	Action After Printing
E0 (1)	Space 1	No Space
E1 (1)	Space 2	No Space
E2 (1)	Space 3	No Space
E3 (1)	Suppress Space	No Space
E4 (1)	Skip to Channel 1 (2)	No Space
E5	Skip to Channel 12 (3)	No Space
E6	Skip to Channel 6	No Space
E7	Skip to Channel 5	No Space
E8	Skip to Channel 4	No Space
E9	Skip to Channel 3	No Space
EA	Skip to Channel 2	No Space
EB	Skip to Channel 11	No Space
EC	Skip to Channel 7	No Space
ED	Skip to Channel 8	No Space
EE	Skip to Channel 9	No Space
EF	Skip to Channel 10	No Space
F0	No Space	Skip to Channel 1 (2)
F1	No Space	Skip to Channel 12 (3)
F2	No Space	Skip to Channel 6
F3	No Space	Skip to Channel 5
F4	No Space	Skip to Channel 4
F5	No Space	Skip to Channel 3
F6	No Space	Skip to Channel 2
F7	No Space	Skip to Channel 11
F8	No Space	Skip to Channel 7
F9	No Space	Skip to Channel 8
FA	No Space	Skip to Channel 9
FB	No Space	Skip to Channel 10
FC-FE	Reserved	

Notes:

1. Supported on all devices
2. Page eject
3. Bottom of page

See individual TIP sections for exceptions.

```
Terminal Input                        BVT

                 (2)
Jobcard:
                 ┌29┐
                 │26│                  (1)               ┌29┐        (3)
JOBNAME....      └xx┘                 FF90 JOBNAME....   │26│       FF0A
                                                         └xx┘
End of Record Card:

      (5)
┌7                    ┐
  8              ┌29┐ │
   9 nn          │26│ │               (4)
└/* EOR nn...    └xx┘ ┘               FF0B nn FF0A

End of Information Card:
┌6     ┐
  7
   8
    9                                 (6)
└/* EOI┘                              FF0C FF0A
```

Notes:

1. Uncompressed stream terminated by FF flag.
2. Columns 79/80 of JOB card may contain 26/29 code sequence.
3. End of physical record sequence.
4. EOR sequence.
5. EOR card may contain octal logical level number following EOR designator.
6. EOI sequence. This is not valid for HASP.

Figure 6-11. Sample CYBER Job Stream Card Inputs for BVT Data Handling

INTERACTIVE VIRTUAL TERMINAL (IVT)

Four types of additional block handling are needed for interactive type terminals:

- Ability to place data in IVT format.
- Ability to handle block protocol for each of the 9 block types that can be passed over the host/NPU interface.
- Special service messages for the CS node are needed.
- Special service messages are needed to change interactive terminal operating modes and terminal parameters.

Details of the user interface and virtual-to-real transforms are described in the appropriate TIP section.

The variety of terminals that may be used to access interactive processes causes a problem of incompatibility. This problem is of greater concern on the output side where the use of format effectors produces undesirable and unintended effects. The NT code solves these problems.

Because of the TIP's state programs, an application program in the host may expect compatible input from a terminal, and may issue output to a terminal with confidence that the intended results will occur. IVT provides the necessary transforms between selected types of terminals and one of the designated virtual terminal subtypes. IVT also provides a method for varying these transforms to widen the variety of terminals which may be accommodated.

The choice of functions provided by the IVT modules has been restricted to ensure that significant intelligence will not be lost even when transforming to the real terminal with the lowest capability. Where the application program requires features not provided by IVT, but known to exist on the connected real terminal, the application program may use those features in one of two ways:

- The application may embed appropriate control characters in the output text or, conversely, scan for significant control characters in the input text. Due regard must be made for the control characters which are significant to IVT and, therefore, are possibly transformed by the TIP.
- By transferring data within formatting changes allowed (transparent mode), the transforms are inhibited and the application has direct access and responsibility for all real terminal features. Transparent mode is separately selectable in each direction. Transparent mode is not allowed for HASP interactive devices.

Interactive Virtual Terminal Characteristics

An IVT always has an input device and an output device. The input device is typically a keyboard, but may be a paper tape reader or cassette reader. When the input device is not a keyboard, IVT normalizes reader input so that it appears to be identical to keyboard input.

The output device is typically a printer or display, but may be a paper tape punch or cassette recorder. The host application program does not normally concern itself with the output media. Optional additional equipment supported includes a paper tape reader or punch. Paper tape can be used anytime, but the user must declare if X-OFF is to be put on this output tape. The user must also declare if the tape is to be turned on again when an X-OFF is input from the tape.

The IVT does not provide a method of switching between a display and a printer, but assumes that local hard copy facilities may exist. IVT device parameters, as seen by a host application, are as follows:

Line width - Infinite (subject to block limit)
Page size - Infinite
Parity - None (set to zero)
Code set - ASCII - 128 characters available
Format effector delays - None

IVT format effectors (FE) are an optional feature of downline data blocks. A flag in the data block clarifier (DBC) determines whether FEs are present or not. If the flag is set, FEs are not present and each output logical line is defined as single-spaced, prior to printing; and the first character is printed. If the flag is zero, FEs are assumed as the first byte of each logical line of text. Undefined FEs default to the single-space prior to print condition. The interpretation of FEs is given in table 6-4.

TABLE 6-4. FORMAT EFFECTORS

<table>
<tr><th>FE</th><th>Action</th><th>When Action Occurs</th></tr>
<tr><td>SP</td><td>Single Space</td><td>preprint</td></tr>
<tr><td>0</td><td>Double Space</td><td>preprint</td></tr>
<tr><td>-</td><td>Triple Space</td><td>preprint</td></tr>
<tr><td>+</td><td>Position to start of current line</td><td>preprint</td></tr>
<tr><td>*</td><td>Position to top of form or cursor home</td><td>preprint</td></tr>
<tr><td>1</td><td>Home cursor and clear screen</td><td>preprint</td></tr>
<tr><td>,</td><td>No action</td><td>preprint</td></tr>
<tr><td>.</td><td>Single space</td><td>postprint</td></tr>
<tr><td>/</td><td>Position to start of current line</td><td>postprint</td></tr>
<tr><td colspan="3">Two additional format control symbols may be passed downline in text:</td></tr>
<tr><td>< CR ></td><td>Carriage return</td><td>preprint</td></tr>
<tr><td>< LF ></td><td>Line feed</td><td>preprint</td></tr>
</table>

Other potential control characters, or control character sequences, are translated one for one. Thus the application can detect special input control sequences and transmit special output control sequences by taking note of the translation performed for a specific terminal. Idle fill can be inserted for <CR> and <LF>, however. IVT operational controls which may be passed via flags in the data block clarifier of downline blocks are as follows:

Auto-input	- Return this output with next input (only effective for a MSG type block)
Transparent	- Inhibit IVT transform for this output
FE	- FEs present or absent

IVT operational controls which may be passed via flags in the data block clarifier of upline blocks are as follows:

Transparent	- This block remains in the terminal format
Parity Error	- This block had one or more parity errors
Cancel	- Cancel the message of which this MSG block is a part

IVT mode control, which may be affected by downline synchronous commands, is shown in table 6-5 under the description of a MESSAGE.

The basic format of the command block is as follows:

	MESSAGE = NETWORK HDR				COMMAND	UPLINE OR DOWNLINE CONTROL
Byte		1	2	3	4	5
	Link header	DN	SN	CN	BT=04	CONTROL TEXT

IVT uses upline synchronous commands to communicate that input has been stopped. (Note that CN defines that the command is for this terminal.)

TABLE 6-5. IVT BLOCK SYNTAX

MESSAGE = $\begin{bmatrix} \text{CONTROL} \\ \text{DATA MESSAGE} \end{bmatrix}$

<u>CONTROL MESSAGE PARAMETERS</u>

CONTROL = NETWORK HDR $\begin{bmatrix} \text{COMMAND} \\ \text{BREAK} \end{bmatrix}$ $\begin{bmatrix} \begin{bmatrix} \text{DOWNLINE CONTROL} \\ \text{UPLINE CONTROL} \end{bmatrix} \\ \text{REASON CODE} \end{bmatrix}$

NETWORK HDR = DN SN CN PRI BSN - See block protocol description at the beginning of section 6

COMMAND = 4

BREAK = 5

TABLE 6-5. IVT BLOCK SYNTAX (Contd)

REASON CODE (for break)	= $\begin{bmatrix}00\\01\\02\\03\end{bmatrix}$ - User 1 break received (usually abort queue) - User 2 break received (usually abort job) - Output device not ready - Illegal/invalid block sent by host	
DOWNLINE CONTROL	= $\begin{bmatrix}\text{TERMINAL CONTROL}\\\text{TERMINAL PARAMETERS}\end{bmatrix}$	
TERMINAL CONTROL	= $\begin{bmatrix}\text{STOP INPUT}\\\text{START INPUT}\end{bmatrix}$	
STOP INPUT	= PFC = $C1_{16}$ SFC = 05	
START INPUT	= PFC = $C1_{16}$ SFC = 06	
UPLINE CONTROL	= INPUT STOPPED REASON CODE	
INPUT STOPPED	= PFC = $C1_{16}$ SFC = 07	
REASON CODE (for input stopped)	= $\begin{Bmatrix}00\\01\\02\\03\\04\end{Bmatrix}$ - Stop input response - Input device not ready - Card slip error - EOI input - Interactive interrupt	
TERMINALPARAMETERS = PFC=C1 SFC=04	$\left[\begin{array}{l}\text{TC} = \begin{bmatrix}1\\2\\\cdot\\\cdot\\\cdot\\14\\15\end{bmatrix}\\\text{PW} = \text{NNN}\\\text{PL} = \text{NNN}\\\text{PA} = \begin{bmatrix}\text{Z}\\\text{O}\\\text{E}\\\text{N}\end{bmatrix}\\\text{CN} = \text{SELECTED CHAR}\\\text{BS} = \text{SELECTED CHAR}\\\text{CT} = \text{SELECTED CHAR}\\\text{CI} = \begin{bmatrix}\text{CA}\\\text{NN}\end{bmatrix}\\\text{LI} = \begin{bmatrix}\text{CA}\\\text{NN}\end{bmatrix}\\\text{SE} = \begin{bmatrix}\text{N}\\\text{Y}\end{bmatrix}\\\text{DL} = \text{(XHH) (,CNNNN) (,TO)}\end{array}\right]$ ⋮ ⋮	Meaning of terminal parameters are discussed below under the heading: Commands for Terminal Parameterization. PFC and SFC values are given in appendix C

TABLE 6-5. IVT BLOCK SYNTAX (Contd)

⋮

IN = $\begin{bmatrix}\text{KB}\\ \text{XK}\\ \text{PT}\\ \text{XP}\\ \text{X}\end{bmatrix}$

OP = $\begin{bmatrix}\text{PR}\\ \text{DI}\\ \text{PT}\end{bmatrix}$

CD = A

EP = $\begin{bmatrix}\text{Y}\\ \text{N}\end{bmatrix}$

PG = $\begin{bmatrix}\text{Y}\\ \text{N}\end{bmatrix}$

AL = SELECTED CHAR

B1 = SELECTED CHAR

B2 = SELECTED CHAR

MS = TEXT

NNNN = (0...4095) - One to Four Decimal Digits

NNN = (0...255) - One to Three Decimal Digits

NN = (0...99) - One to Two Decimal Digits

SELECTEDCHAR = ASCII Representation of Selected Character

HH = 00...FF - Selected Bit Pattern as Sent by Terminal

TEXT = One through fifty ASCII characters message composing the text

END OF CONTROL MESSAGE PARAMETERS

DATA MESSAGE PARAMETERS

DATAMESSAGE = $\begin{bmatrix}\text{TRANSMODEMSG}\\ \text{CHARMODEMESSAGE}\end{bmatrix}$

TRANSMODEMSG = BLKBLOCK** MSGBLOCK

BLKBLOCK = NETWORKHDR BLK TRANSBLKCONTENT

MSGBLOCK = NETWORKHDR MSG TRANSBLKCONTENT

NETWORKHDR = DN SN CN PRI BSN

BLK = 1

MSG = 2

TRANSBLKCONTENT = $\begin{bmatrix}\text{DBCUPLINE}\\ \text{DBCDOWNLINE}\end{bmatrix} (\text{BYTE})_{0\text{-}n*}$

*The number of bytes in a BLKBLOCK is a system parameter separately declarable upline and downline; n ≤ 2043.

**0-m lines

TABLE 6-5. IVT BLOCK SYNTAX (Contd)

```
BYTE               = (0...255) - CLA mode or terminal may not support
                                 full range.  If terminator is specified,
                                 then that value will not appear upline

                     SPARE    SPARE    SPARE    SPARE
DBCDOWNLINE        = FEUSAGE   TRANSPARENT   NOTUSED
                     AUTOINPUT    Binary flags

                     [00] - This output is not autoinput
AUTOINPUT          = [01] - This output is to be returned ahead of
                            next input

FEUSAGE            = [00] - Format effectors (FEs) used
                     [01] - Format effectors not used

                     SPARE    SPARE    SPARE    NOTUSED
DBCUPLINE          = NOTUSED   TRANSPARENT   CANCELLED
                     PARITYERROR    Binary Flags

CANCELLED          = [00] - No Action Required
                     [01] - Cancel any incomplete upline message

TRANSPARENT        = [00] - Block is in character mode
                     [01] - Block is in transparent mode

PARITYERROR        = [00] - Parity errors not detected
                     [01] - Parity errors detected

CHARMODEMESSAGE    = CHARMODEBLK**   CHARMODEMSG

CHARMODEBLK        = BLKADDRESS   BLK  [ UPLINE CONTENT   ]
                                       [ DOWNLINE CONTENT ]

CHARMODEMSG        = BLKADDRESS   MSG  [ UPLINECONTENT   ]
                                       [ DOWNLINECONTENT ]

BLKADDRESS         = DN    SN    CN    See normal block header

UPLINECONTENT      = DBCUPLINE   [ PHYSICALLINE ]
                                 [ LOGICALLINE  ]

PHYSICALLINE       = 128ASCIICHARSETWITHPARITYSETTOZERO** - Blocking may
                     occur at physical line boundary or logical line
                     boundary.  See individual TIPs for a discussion of
                     upline blocking.

LOGICALLINE        = 128ASCIICHARSETWITHPARITYSETTOZERO** - Except total
                     of all bytes in a block must not exceed n.

DOWNLINECONTENT    = [[FE   LOGICALLINE   US]* ]
                     [[LOGICALLINE   US]**     ]

US                   1F16 - US must not appear in a downline logical line

                     [ SINGLESPACEPRE            ]
                     [ DOUBLESPACEPRE            ]
                     [ TRIPLESPACEPRE            ]
                     [ STARTOFCURRENTLINEPRE     ]
FE                 = [ FORMFEEDPRE               ]
                     [ HOMEANDCLEARPRE           ]
                     [ NULL                      ]
                     [ SINGLESPACEPOST           ]
                     [ STARTOFCURRENTLINEPOST    ]
```

*The number of bytes in a BLKBLOCK is a system parameter separately declarable upline and downline; n ≤2043.

**0-m lines

TABLE 6-5. IVT BLOCK SYNTAX (Contd)

SINGLESPACEPRE	= SP	Preprint Format Effectors (defined earlier)
DOUBLESPACEPRE	= '0'	
TRIPLESPACEPRE	= '-'	
STARTOFCURRENTLINEPRE	= '+'	
FORMFEEDPRE	= '*'	
HOMEANDCLEARPRE	= '1'	
NULL	= ','	No Action
SINGLESPACEPOST	= '.'	Postprint Format Effectors (defined earlier)
STARTOFCURRENTLINEPOST	= '/'	

IVT Block Handling at Host Interface

When a TIP in the NPU communicates with the application program in the CYBER host, the communication between the two is subject to processing by an intermediate process in the host called the Network Access Method (NAM). NAM exists to provide a common logical interface to the communications network.

The IVT interface to the host is necessarily described at two levels - the interface to NAM, and the overlying interface to the interactive application. The interface to NAM uses block protocol. Its special application to IVT is defined below:

IVT Block Protocol Usage

BLK Block

- The BLK block is a non-last segment of a message. It is used for transferring data both upline and downline. When a message is greater than m bytes (M n), then the message is divided into blocks of n bytes long. All non-last segments are sent as BLK blocks. Blocks have a maximum of 2043 bytes, but are normally smaller to conserve 255X resources.
- Upline, a character mode block is a partial logical line, (typically a physical line), sent at the convenience of the TIP. A transparent mode block consists of a system-defined number of bytes.
- The optimum block size for the IVT is a small number of physical lines for the specific terminal. For special application, such as graphics or paging, the optimum block size is a single display.

MSG Block

The last or only segment of a message is sent as a MSG block. For transparent downline data, if page wait is selected, the MSG block indicates the end of the page.

BACK Block

The BACK block is used for flow control. A BACK is sent by the receiving process (NAM/TIP) when it has delivered, or otherwise disposed of a BLK, MSG or CMD block.

CMD Block

The command block (CMD) provides a means of passing control information synchronously with the data stream, but apart from the BLK and MSG blocks which constitute the data stream. The CMD block functions available are specified later in this section.

BRK Block

A TIP sends the break block (BRK) when:

- User Break 1 is received from the terminal (typically this means abort the queue).
- User Break 2 is received from the terminal (typically this means abort the job).
- The downline block does not conform to IVT format.

In all cases, the TIP discards all locally queued output data and all newly arriving data until a reset (RST) block is detected. Data discarded includes synchronous blocks. Downline BRK blocks are not used.

STP Block

The TIP may send a stop block (STP) to the application program to request suspension of output.

STRT Block

The start block (STRT) cancels the effect of the STP block.

RST Block

A reset block (RST) is sent by a process when it has received a BRK or STOP block. A RST block specifies the point in the data stream when the break or stop occurred. A further STP or BRK block must not be issued until the previous RST block has been processed.

Block usage is defined in the TIP sections.

Table 6-5 defines the contents of the message blocks to the level needed for IVT processing. Symbols used are the same as those used for table 6-2. Symbol definition and an example of table use are given in the BVT portion of this section. Familiarization with syntax for block usage can be enhanced by reviewing the sample in figure 6-10.

The following restrictions apply to the use of the IVT block syntax:

- All upline character mode messages consist of zero or more BLK blocks and a single MSG type block. Each block typically contains a single physical line. The whole series of BLK and MSG blocks comprise a single logical line.

- Downline character mode messages may be multiblock. Each block may contain multiple logical lines. Logical lines may not cross block boundaries.

- For downline character mode messages, a flag in the data block clarifier indicates whether format effectors are present. If so, all logical lines are preceded by an <FE> byte. A logical line in a block is terminated by a <US> ($1F_{16}$).

- A logical line may contain any of the 128 ASCII character set, except <US>.

- In character mode, all ASCII characters consist of 7 bits, right-justified, in an 8-bit byte with the parity (bit 8) set to zero.

- All bytes of a transparent mode block can contain any of the 256 possible bit combinations. Exception: if a terminator character is defined for an upline block, this terminator does not appear. Note that terminal or CLA configuration may restrict the significant number of bits in the byte to less than eight.

IVT Block Handling for Communications Supervisor

IVT uses a special subclass of command messages for communicating changes of IVT parameters to CS (node 1) in the host. The types of messages needed are:

- Messages to define terminal class, page width, and length.

- Broadcast messages allowing the network operator to communicate with the operator at one or all of the controlled terminals.

- Messages allowing a terminal operator to communicate with the local NPU operator.

TERMINAL CLASS, PAGE WIDTH/LENGTH

This NPU-originated message provides CS with the current terminal class and page width/length information for this class. The byte format for the message is as shown in figure 6-12.

BROADCAST MESSAGES

The two types of broadcast messages allow the network operator to communicate to a designated terminal or to all the terminals supported by an NPU. These messages and their replies are described in detail in the service message portion of this section. The format of the message is also summarized in appendix C.

Block Header

Link Header	DN =01	SN	CN	PRI/ BSN/ BT	PFC $=0C_{16}$	SFC =03	P	00	HO	CA	TA	DT	ORIG	PW	PL

DN - The CS in host

SN - NPU coupler node

CN - Connection number

PRI - Priority

BSN - Block serial number

BT - Block type; 4 = CMD

PFC
SFC - Primary and secondary function codes for this SM

P - Port for this terminal

HO - Host ordinal

CA - Cluster address
TA - Terminal address   defined in appendix C
DT - Device type

ORIG - Originator of message

00 - terminal user
01 - applications program

TC - Terminal class - defined in appendix C; $1 \le TC \le 15$

PW - Page width in characters/line; $0 \le PW \le 255$

PL - Page length in lines; $0 \le PL \le 255$

Figure 6-12. Format for Terminal Class, Page Width, Page Length Messages

OPERATOR MESSAGE

This message originates at a terminal and allows the terminal operator to communicate with the network operator. The byte format of the message is:

Byte

Link Header	DN =01	SN	CN	PRI/ BSN/ BT	PFC =$0C	SFC =02	P	SP	HO	CA	TA	DT	HO	TEXT

where all the fields except TEXT are defined above.

TEXT is sent in response to a <CTL>MS = TEXT message previously delivered to the terminal

Commands for Changing Terminal Parameters

As noted in table 6-5, a special subclass of IVT service messages is used for terminal parameters. These commands belong to the <TERMINAL PARAMETERS> class of messages. In table 6-5, the parameters are left undefined; they are defined below.

Each control message consists of a synchronous command with a single command embedded as an ASCII test string. All control messages from the host to the TIP may also be entered by a terminal user. Three of the commands were discussed above: terminal class, page width and page length. When these are entered by the terminal user, or issued by the host, they are reported to the communication supervisor. All terminal user-entered commands result in an acceptable or unacceptable response to the user. Host commands that are invalid or illegal are rejected with a BRK block, and are printed on the NPU console.

Terminal parameter definitions are as follows:

- Terminal Class (TC)

  TC establishes a class for the terminal, with default values for all parameters, as defined in table C-7. A TIP does not execute a command if the class is not supported. This change must be reported to CS in the host.

- Page Width (PW)

  PW establishes the physical line width in characters for output. For nontransparent blocks, the TIP inserts a character to move the carriage or cursor to the next line. This insertion occurs at the point where the number of characters to be transmitted equals the page width. This character sequence differs on each terminal class. The parameter NNN varies between 0 and 255; 0 means new line and is never inserted. This change must be reported to CS in the host.

- Page Length (PL)

  PL establishes the number of physical lines in a page for output. The TIP inserts the character sequence defined for the terminal class to advance the carriage or cursor to the next page length. Also, if the page wait feature is selected, the TIP will wait for an operator input before continuing. The parameter NN varies between 0 and 255; 0 means no paging. This change must be reported to CS in the host.

NOTE

> None of the remaining IVT parameter changes need be reported to the host (CS).

- Parity Selection (PA)

  PA specifies the type of parity that the TIP expects on input and generates on output. See the description of parity in the asynchronous TIP section of this manual.

- Cancel Character (CN)

  CN establishes the character that is used to delete the current logical input line.

- Backspace Character (BS)

  BS establishes the character that is used to delete the previous input character from the current input buffer.

- Control Character (CT)

  CT establishes the character that is used to enter operational control messages.

- Carriage Return Idle Count (CI)

  CI establishes the number of idle characters to be inserted in the output stream following carriage return (CR). The use of CI-nn overrules the default value and CI-CA restores the default value.

- Line Feed Idle Count (LI)

  LI establishes the number of idle characters to be inserted in the output stream following line feed (LF). The use of LI-nn overrules the default value and LI-CA restores the default value.

- Character Set Detect (CD)

  This restarts the character set recognition logic when changing a character set during a message exchange sequence. First, the terminal operator enters the IVT command: CD = A. Then the operator has 60 seconds to (1) physically change the terminal's code set (for instance, by changing the type element on a typewriter), and (2) activate the TIP's code set recognition sequence by pressing the carriage return key.

- Transparent Text Delimiter (DL)

  DL establishes the transparent text delimiter for input. The delimiter may be a character, a character count, or a timeout of 300 ± 100 ms. One or more of the delimiters may be active simultaneously. Default values are shown in table C-7.

- Input Device (IN)

  IN specifies the input device as a keyboard or paper tape reader, in character or transparent mode. Note that paper tape input is allowed in keyboard mode, but that the TIP does not send the X-ON characters to start the paper tape reader.

- Output Device (OP)

  OP specifies the output device as printer, CRT display, or paper tape punch. Printer and CRT display are functionally equivalent. The user may punch a paper tape in any mode, but the TIP provides the X-OFF character only if OP = PT and if data is not transparent.

- Special Edit Mode (SE)

  A SE = Y selection places the terminal in special edit mode; an SE = N selection returns the terminal to the normal character edit mode. Special edit mode provides two types of special operations: (1) backspace (BS), linefeed (LF), and cancel input control symbols are sent upline as data; and (2) a character delete sequence (one or more backspaces followed by a linefeed) causes the TIP to issue a caret prompt to the terminal, and then to continue with input processing.

- Echoplex Mode (EP)

  EP specifies where input character echoing will take place. EP = N implies the terminal is doing its own input echoing. EP = Y causes the TIP to set the CLA, to provide character echoing.

- Page Wait (PG)

  PG selects the page wait feature. It allows the user to control output by demanding each page explicitly after the previously page has been viewed for the desired period of time.

- Abort Output Line Character (AL)

  AL selects the character which, when input followed by a carriage return, results in the current output line being discarded.

- User Break 1 (B1)

  B1 selects the character which, when input followed by a carriage return, causes the TIP to send an upline BRK block, with reason code specifying user break 1. Conventionally, user break 1 is used to abort the queue.

- User Break 2 (B2)

  B2 selects the character which, when input followed by a carriage return, causes the TIP to send an upline BRK block, with reason code specifying user break 2. Conventionally, user break 2 is used to abort the job.

- Message (MS)

  MS defines the character used to delimit messages to the LOP. Up to 50 characters of text may be inserted between the MS delimiters.

For any of these parameter changes entered from a terminal, the TIP can accept or reject the command. If the TIP accepts the command it does not usually return a positive acknowledgment to the interactive terminal. If, however, the TIP rejects the command, the TIP sends the following error message to the terminal:

ERR...

Table 6-6 shows the IVT terminal parameters as used by the standard TIPs.

TABLE 6-6. TERMINAL PARAMETERS AS USED BY STANDARD TIPS

Command	MD4	HASP	ASYNC
TC	AR[††]	B	AR
PW	AR	AR	AR
PL	AR	B	AR
PA	B	B	A
CN	A	A	A
BS	B	B	A
CT	A	A	A
CI	B	B	A
LI	B	B	A
SE	B	B	A
DL	B	B	A/B[†]
IN	B	B	A
OP	B	B	A
EP	B	B	A
CD	B	B	A
PG	A	B	A
AL	B	B	A
B1	A	A	A
B2	A	A	A
MS	C	C	C
Other or invalid parameters	B	B	B

A = Action
AR = Action and Report to CS
B = No Action; Send BRK or ERR block to host
C = Valid only from User

[†] These commands are only valid for certain terminal classes. DL is not a valid command for terminal class 4 (IBM 2741). A BRK block is sent to the application if any of these commands are received for a terminal in a class which does not support the command.

[††] An error occurs for any attempt to change the mode from 4A to 4C, or vice versa.

HOST INTERFACE PROGRAM 7

This section describes the operation of the Host Interface Program (HIP).

The CYBER 70/170 channel coupler provides the hardware interface between the NPU and the PPU of a CYBER 70/170 host processor. This coupler is operated through the cooperation of two programs; one resident in the host, the other resident in the NPU. The NPU program, called the Host Interface Package (HIP) is described in this section. The HIP provides logic to support the following functions.

- Interrupt processing for coupler-generated interrupts
- Initiation and control of data transfers across the coupler
- Coupler status processing and error recovery
- Communication with the host coupler control program to support the transaction protocol
- The standardized logical (as opposed to physical) interface for all NPU resident software involved with data transfers between host and NPU

TRANSACTION PROTOCOL

A special protocol is used for transfers between the NPU and the host. The block portion of this protocol was discussed in section 6. The directives which pass the blocks across the coupler are discussed here.

TRANSFER FUNCTIONS

The coupler's transfer path is half-duplex: thus it is bi-directional, but transmission occurs in only one direction at a time. Both the host and the NPU can bid for the right to transmit over the transfer path. The following conventions govern the transfers:

- When both the PPU and NPU simultaneously bid for the transfer path, output from the host takes precedence over input to the host. Input to the host is called an upline transfer. Output from the host is called a downline transfer.
- The NPU may reject an output request if it has insufficient space to assign for receiving the message. This is called an overload condition.
- Both the host and NPU coupler control programs operate in one of three states: idle, sending, or receiving.

- When an error occurs during a transaction, the receiving processor discards all data associated with the transaction and returns to an idle state.

- During periods of inactivity, the NPU coupler program generates a periodic IDLE INQUIRY status word to verify that the host is still operating. The host must respond by reading the NPU status word. If the host does not read the word within 10 seconds, the NPU assumes a host failure.

DIRECTIVES USED

Five directives govern the data transfers:

- OUTPUT REQUEST specifies that the host has data to send to the NPU.

- INPUT REQUEST specifies that the NPU has data to send to the host.

- READY FOR OUTPUT specifies that the NPU is ready to accept the data transfer designated by the current OUTPUT REQUEST. This is a response to an OUTPUT REQUEST.

- NOT READY FOR OUTPUT specifies that the NPU cannot accept the data transfer designated by the current OUTPUT REQUEST because there are not sufficient buffers to store the data. This is a response to an OUTPUT REQUEST.

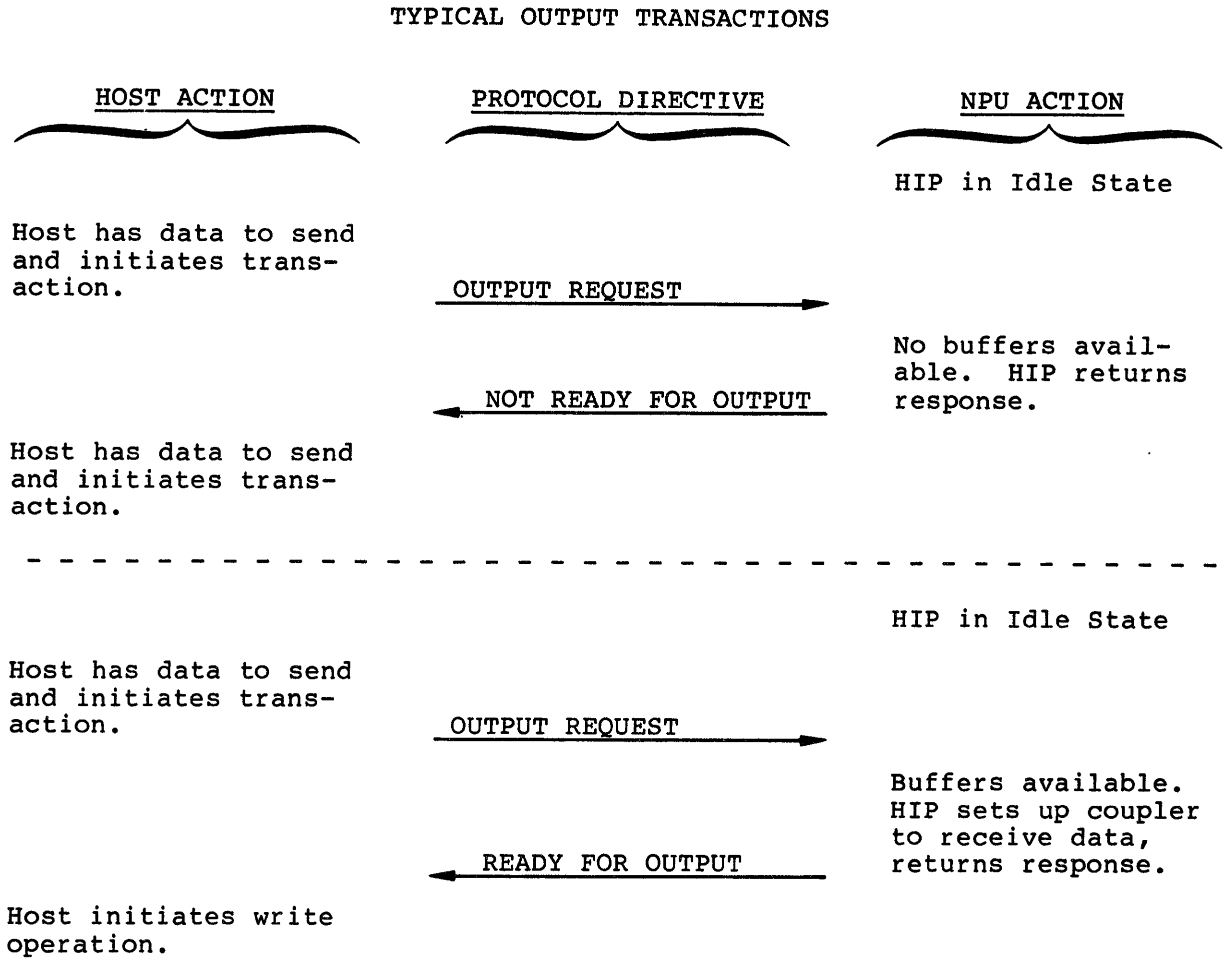
- IDLE INQUIRY indicates that the preestablished timeout period for another transfer to or from the host has expired without activity. The NPU issues this directive to verify that the host is still operating.

TRANSFER INITIATION

Upline data transfers are initiated by the HIP when the CCP notifies the HIP that there is input data queued for transfer to the host. This is an OPS-level event. Downline data transfers are initiated when the HIP receives an OUTPUT REQUEST orderword from the host. This is an interrupt-level event.

If either the upline or the downline data transfer occurs while the HIP is in idle state, the HIP immediately begins to process the request. Requests for upline data transfers are queued if the HIP is already sending or receiving data. Requests for downline data transfers are accepted if the HIP is not already receiving data from the host.

Figure 7-1 shows typical input and output transactions over the coupler. Figure 7-2 shows the resolution of I/O contention at the coupler. Figure 7-3 shows the division of the HIP tasks between the OPS and interrupt levels. The PTxxxxx labels designate HIP subroutines. For further details, see a HIP listing.



TYPICAL OUTPUT TRANSACTIONS

HOST ACTION	PROTOCOL DIRECTIVE	NPU ACTION
		HIP in Idle State
Host has data to send and initiates transaction.	OUTPUT REQUEST →	
	← NOT READY FOR OUTPUT	No buffers available. HIP returns response.
Host has data to send and initiates transaction.		

- -

		HIP in Idle State
Host has data to send and initiates transaction.	OUTPUT REQUEST →	
	← READY FOR OUTPUT	Buffers available. HIP sets up coupler to receive data, returns response.
Host initiates write operation.		

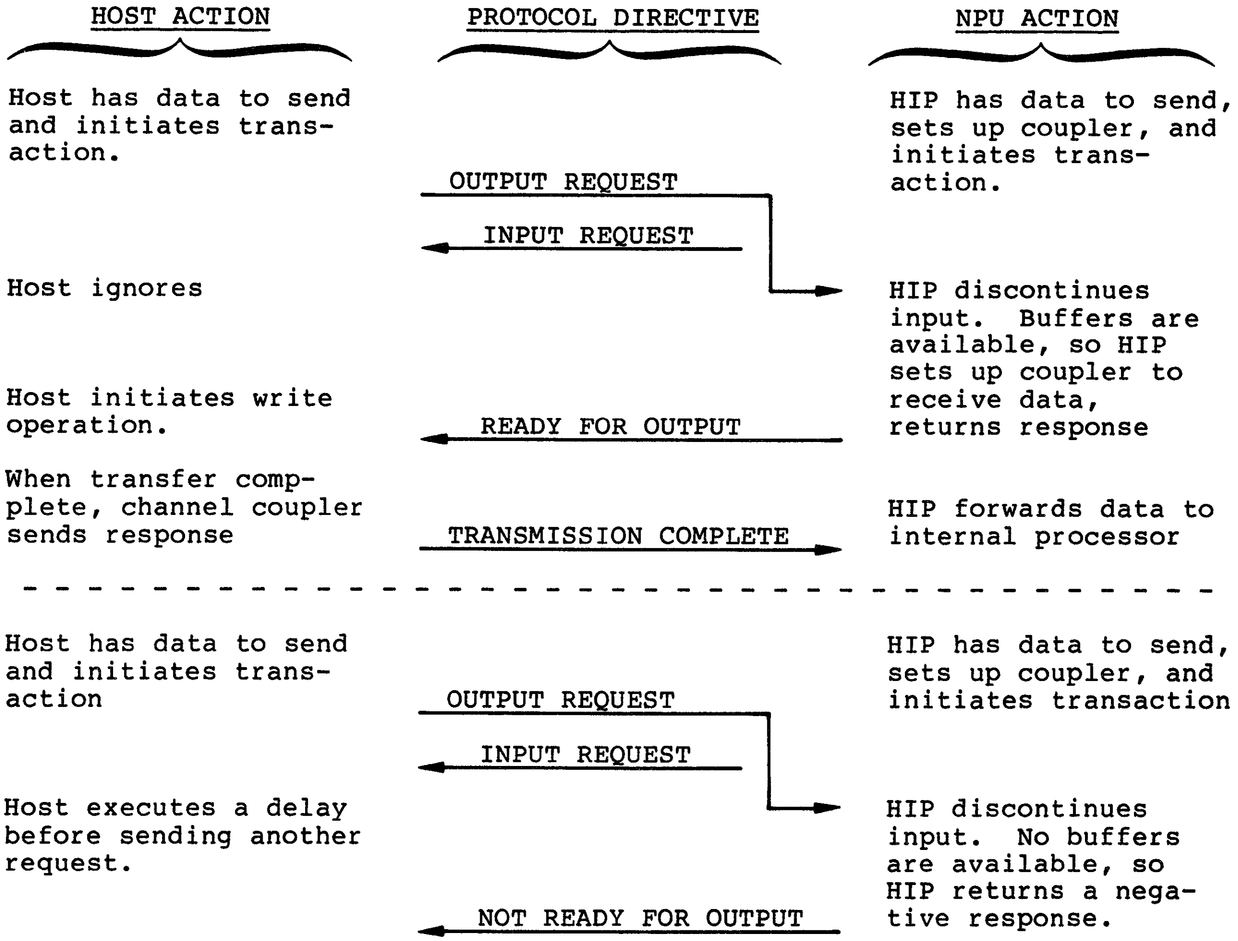
The transaction is ended when the coupler generates the completion interrupts to the host and NPU. If a transfer error occurs, the data is discarded by the HIP and the host must initiate the transfer again.

Figure 7-1. Coupler I/O Transactions (sheet 1 of 2)

TYPICAL INPUT TRANSACTIONS

HOST ACTION	PROTOCOL DIRECTIVE	NPU ACTION
	← INPUT REQUEST	HIP has data to send, sets up coupler and initiates transaction.
Host unable to accept data	NOT READY FOR INPUT →	
		HIP waits up to 1-2 ms before trying again.
- - - - - - - -	- - - - - - - -	- - - - - - - -
	← INPUT REQUEST	HIP has data to send, sets up coupler and initiates transaction.
Host can accept data and initiates a read operation.		
When transfer completes, channel coupler sends interrupt	TRANSACTION COMPLETE →	
		HIP releases data buffers.

Figure 7-1. Coupler I/O Transactions (sheet 2 of 2)



HOST ACTION	PROTOCOL DIRECTIVE	NPU ACTION
Host has data to send and initiates transaction.		HIP has data to send, sets up coupler, and initiates transaction.
	OUTPUT REQUEST →	
	← INPUT REQUEST	
Host ignores		HIP discontinues input. Buffers are available, so HIP sets up coupler to receive data, returns response
Host initiates write operation.	← READY FOR OUTPUT	
When transfer compplete, channel coupler sends response	TRANSMISSION COMPLETE →	HIP forwards data to internal processor

Host has data to send and initiates transaction		HIP has data to send, sets up coupler, and initiates transaction
	OUTPUT REQUEST →	
	← INPUT REQUEST	
Host executes a delay before sending another request.		HIP discontinues input. No buffers are available, so HIP returns a negative response.
	← NOT READY FOR OUTPUT	

HIP then starts a normal input sequence.

Figure 7-2. I/O Transaction Contention at The Coupler

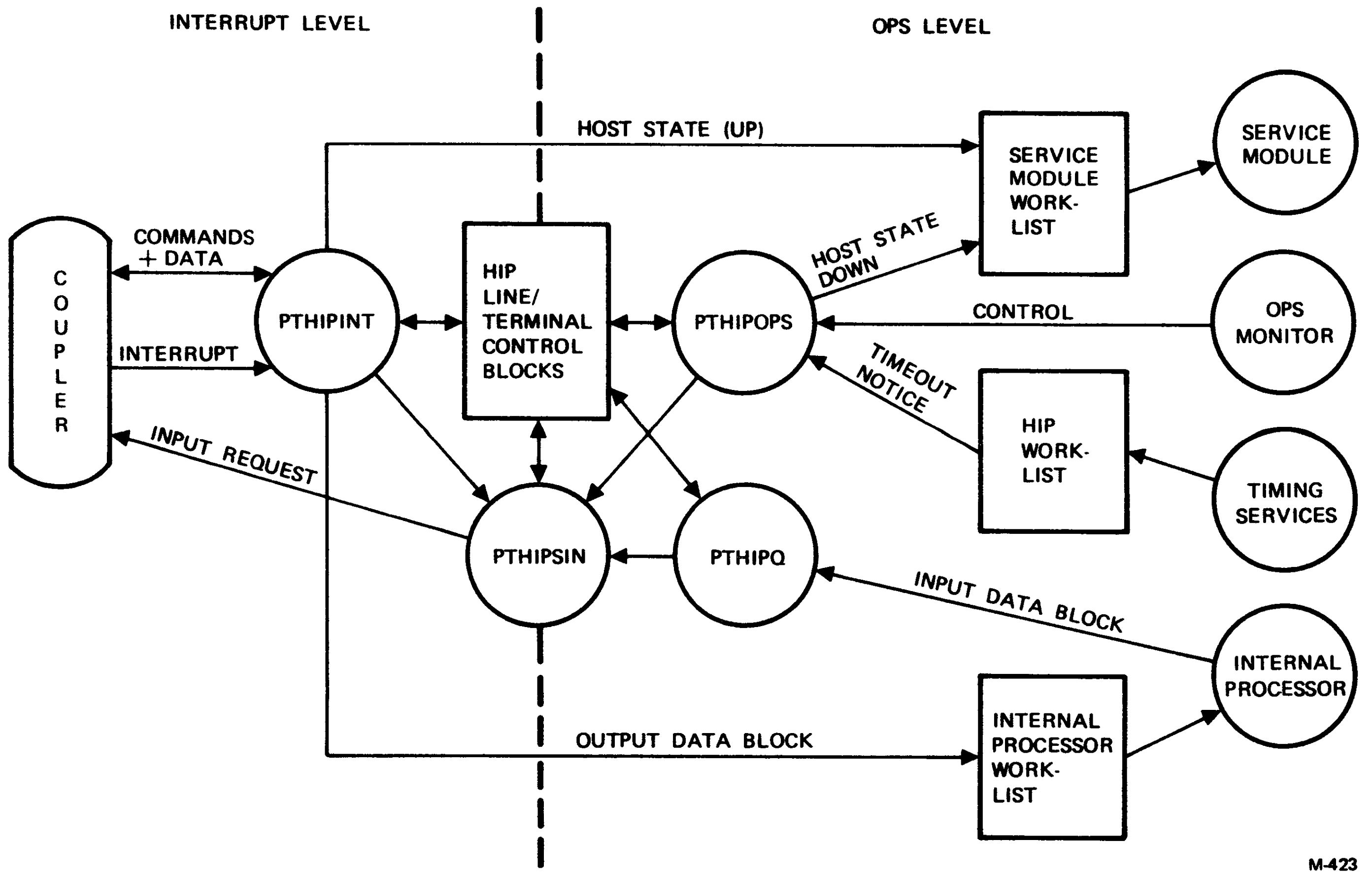


Figure 7-3. OPS and Interrupt Levels for the HIP

TRANSFER TIMING

All coupler transfers are timed by means of a deadman timer which is set for ten seconds. If the scheduled transfer fails to complete during that period (a timeout condition), the HIP declares that the host is down. The HIP then causes the service module to send the HOST UNAVAILABLE message to all interactive terminals. The NPU rejects all further input from terminals. The HIP also discards any output if an output transfer was in progress. If an input transfer was in progress, the current block is replaced at the head of the output queue. It will be the first block transmitted when the host recovers.

The HIP recognizes that the host has recovered when a valid orderword is received. All terminals are notified by a message sent through the service module. Input is again accepted from the terminals.

ERROR PROCESSING

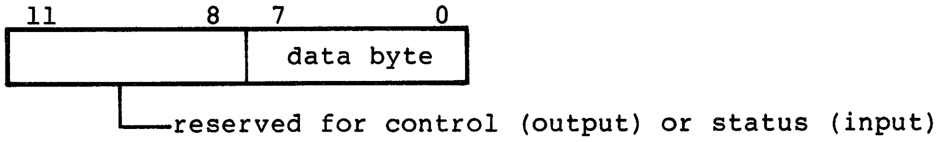
The HIP provides two types of error processing:

- For recoverable errors, the HIP retries the transfer. The HIP provides an unlimited number of retries to accomplish the transfer. However, in practice the number of retries is limited by the host stopping the transfer or stopping the NPU and reloading the CCP. The recoverable errors are data parity error, hardware timeout, and abnormal termination.
- For unrecoverable errors the HIP aborts the transaction. The unrecoverable errors are memory parity error, memory protect error, and chain address zero (the condition which occurs when the HIP expects to find a chained data buffer, but finds a zero address for that buffer). All of these cause an NPU halt and are therefore unrecoverable errors. The NPU processor must be downline loaded from the host to continue message processing.

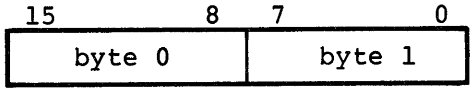
When an error is detected during a downline transfer, the HIP discards the data associated with the transfer and returns to the idle state.

HOST/NPU WORK FORMATS

The host uses a 12 bit byte at the PPU interface. Format is as shown:



The NPU uses a 16 bit word composed of two eight bit bytes. Each NPU word requires two PPU words. Data transmission to the host is made only over the direct memory access (DMA) path. Format is as shown:



Other transfers are made through four sets of special registers in the coupler. The NPU uses the internal data channel (IDC) for loading and reading these registers. The registers have a 16 bit interface on the NPU side and a 12 bit interface on the host side. Transfers to the registers are discussed below under coupler interface hardware programming.

COUPLER INTERFACE HARDWARE PROGRAMMING

Figure 7-4 shows the coupler hardware which constitutes the host/NPU interface. A PPU may interface to one or two couplers, but each coupler must connect to different NPU. An NPU can also have two couplers. If there are two couplers, the NPU determines which host loads the NPU at initialization time.

The coupler has three transmission circuits:

- A half-duplex data circuit for transmission of programs or data between the memory of the PPU and the main memory of the NPU. On the NPU side, this circuit uses the direct memory access mode of transmission. This channel also provides an execution control method (function command) used by the PPU to start or stop NPU microprogram execution. Micromemory execution must be started at address 0. This method is used for initial loading and dumping of the NPU.
- A full-duplex control circuit which the NPU and the PPU use to perform transaction setup (handshaking).
- A supervisory circuit which is set up and monitored by both NPU and PPU. Transaction status is made available to both sides of the interface by this circuit.

COUPLER REGISTER USE

It must be recognized that the names of some of the registers (coupler status, orderword, NPU status word) and some of the circuits (supervisory, control) do not adequately define coupler operations. For instance, the control and set up of the NPU involves the following:

- The host loads the orderword register, and examines the coupler status word to determine if the NPU status word is available for examination. The NPU status word is then checked.
- The host sends a function word address to the coupler channel and executes an output command for a single word transfer.
- At a later time, the host sends service messages for further control of the NPU, using block transfers on the data channel. The NPU replies using service messages.
- In all cases the host and/or NPU checks and changes coupler status register bits to indicate the current status of the transfer activities.
- The host or NPU transmits data (messages) after properly setting up a block starting address in the NPU, using the memory address registers in the coupler.

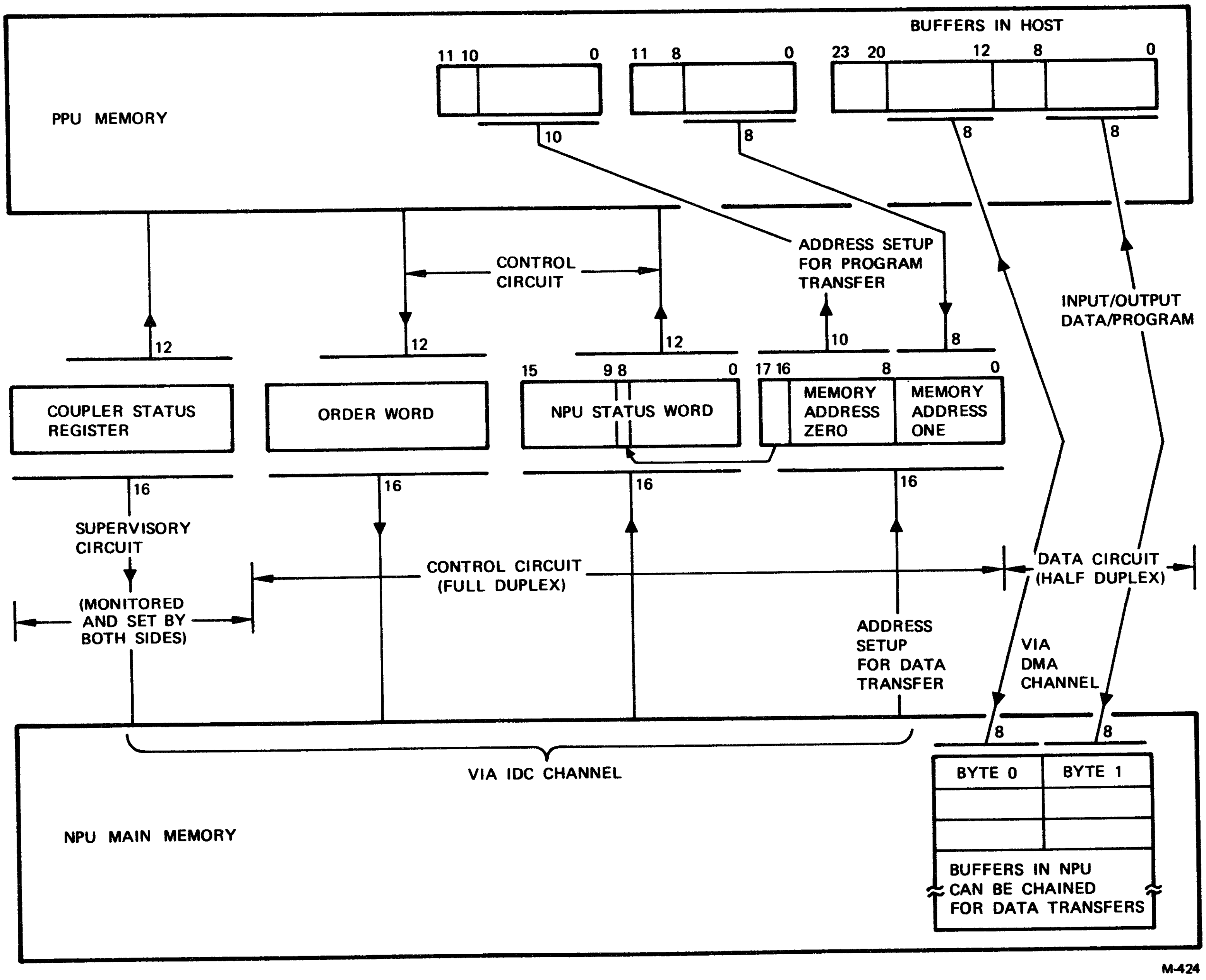


Figure 7-4. Coupler Registers

The coupler registers shown in figure 7-4 directly accessed by the PPU program for normal data transmission are as follows:

- Coupler Status Register - A group of 16 hardware-defined flags, the low order twelve bits can be read by the PPU. The flags inform the NPU of the reason for interrupt, and indicate to both the NPU and PPU the status of the transaction and the status of other coupler registers.

- NPU Order Word - A 16-bit register, the low order twelve bits are written by the PPU to communicate a software-defined order code to the NPU. This code determines the order of regulation across the coupler.

- NPU Status Word - A 16-bit register, the low order twelve bits can be read by the PPU. The NPU uses this register to communicate a software-defined status code to the PPU. This code indicates the type of transfer that the NPU is ready to perform.

- NPU Address Register - An 18-bit register, the PPU can write all 18 bits for the purpose of loading or dumping the NPU. The high order 10 bits (address register bits 17-8, plus bit 8 of the NPU status register) are called memory address zero. The low order 8 bits, address register bits 7-0, are called memory address one. The PPU must perform two function operations to write the entire register. Since the highest order bits of the address register (bits 17, 16) are actually implemented as bits 9, 8 of the NPU Status word, those bits cannot be used for other purposes.

  The NPU address register is also set by the NPU to indicate to the host the address of the first word to be transferred during a data transfer.

The code/bit assignment for each of these registers is shown in tables 7-1 through 7-4.

The NPU receives an interrupt when the PPU writes the order word or completes a data transfer. The coupler status register indicates the reason for the interrupt to the NPU. Therefore, the PPU does not use a separate control circuit to indicate that the transaction is complete; this information being automatically available in the supervisory circuit.

PROGRAMMING THE COUPLER BY USE OF FUNCTION CODES

The coupler may be given function codes by either the PPU or the NPU. In either case, the codes are treated as one word addressed to the coupler equipment. From the NPU side, functions are sent to the coupler over the internal data channel.

HOST FUNCTION COMMANDS

The coupler is programmed from the host (PPU) side by setting a function code (see table 7-5) and executing an I/O instruction. The coupler function code occupies the low order nine bits of the 12-bit PPU function code. The high order 3 bits of this PPU word contain the equipment code (coupler address on the channel). The equipment code is determined by the setting of hardware switches on the coupler.

TABLE 7-1. COUPLER STATUS REGISTER BIT ASSIGNMENT (sheet 1 of 2)

Bit Number	I/A	Flag Name	SET Condition	RESET Condition
0	A	Memory parity error	NPU memory parity error	†
1	A	Memory protect fault	NPU memory protect fault	†
2	-	NPU status word loaded	NPU writes status word	PPU reads NPU status word ††
3	-	Memory address register loaded	PPU or NPU writes memory address one	-
4	I	External cabinet alarm	Power failure	†
5	I	Transmission complete	PPU completes any input or output operation	†
6	I	Transfer terminated by NPU	NPU terminates transfer (not used)	†
7	I	Transfer terminated by PPU	PPU sets channel inactive during data I/O	†
8	I	Orderword register loaded	PPU writes orderword	NPU reads orderword
9	-	NPU status read	PPU reads NPU status word	†
10	I	Timeout	Inactive returned during a PPU data I/O operation because coupler was selected and active for more than 3 seconds	†

All flags (†† except bit 2) are reset when NPU or PPU clears the coupler. Those flags marked with †are also cleared when the NPU reads the coupler status register. All flags are cleared by a Master Clear.

I/A: I = Setting Flag causes an NPU interrupt; A = Setting Flag causes an alarm.

TABLE 7-1. COUPLER STATUS REGISTER BIT ASSIGNMENT (sheet 2 of 2)

Bit Number	I/A	Flag Name	SET Condition	RESET Condition
11	A	CYBER 170 channel parity error	12-bit word plus parity from data channel not odd parity. Enable parity switch on.	Enable † parity switch positive transition
12-13		Unused		
14		Chain address zero	Coupler finds zero in last word of NPU buffer.	†
15	-	Alarm	Positive transition of any flag marked "A"	†
All flags (†† except bit 2) are reset when NPU or PPU clears the coupler. Those flags marked with † are also cleared when the NPU reads the coupler status register. All flags are cleared by Master Clear. I/A: I = Raising Flag causes NPU Interrupt; A = Raising Flag causes Alarm.				

The coupler channel is automatically disconnected when the PPU sends the function code. The disconnect occurs within one microsecond of executing the function code. If a parity error is detected on the function code (CYBER 170), the channel is not disconnected.

NPU FUNCTION COMMANDS

The NPU commands (see table 7-6) are issued over the internal data channel. The coupler is not disconnected from the host by these commands.

HIP FUNCTIONS

There are two primary functions performed by the HIP:

- Processing single word (control/status) function.
- Processing block transfers, for control or message processing purposes.

TABLE 7-2. ORDERWORD REGISTER CODES

11 9	8 0	
Order Code	Length	Orderword Register

Order Code Value	Name	Regulation Level
1	Output Level 1 (Service Messages)	1
2	Output Level 2 (High Priority Data)	2
3	Output Level 3 (Low Priority Data)	3
5	Host not ready for input	

Length - In 8 byte increments, of the output block to be transferred. The value is rounded up when the length is not a multiple of 8.

TABLE 7-3. NPU STATUS WORD CODES

Code Value (hexadecimal)	Name	Protocol
0	Ignore value and read again	Data transfer
1	Idle	↓
4	Ready for output	↓
7	Not ready for output	↓
8	Ready for dump	Dump transfer
13	Input available, 256 bytes	Data transfer
14	Input available, 256 bytes	Data transfer
100	Bit 16 of address register (actually bit 16 of the NPU address register)	NPU address set up by host dump/load protocol

TABLE 7-4. ADDRESS REGISTER CODE

bit 16 (first word)	bit 15 - 8	bit 7 - 0
Used as bit 8 of NPU status word	Memory address 0	Memory address 1

1. Address register increments with each NPU word (16 bits) transferred.
2. Bits 11-8 of the second PPU word and bits 11-9 of the first PPU word are discarded when loading register from PPU.
3. Only 15 bits are loaded from NPU; PPU zero fills the upper sets of each word.

SINGLE WORD TRANSFERS (CONTROL)

The PPU can write the orderword at any time. The NPU reads the orderword only if it has been loaded by the PPU, as indicated by bit 8 of the coupler status register. This bit is automatically reset when the NPU reads the orderword.

The NPU can write the NPU status word at any time. The PPU can read the NPU status word only if it has been loaded by the NPU. When the PPU reads the register, it cannot read the register again until the NPU again writes the register. The PPU determines that the NPU status word has been loaded (written) by interrogating bit 2 of the coupler status register. This bit is automatically reset when the PPU reads the NPU status word.

Note that the NPU accesses the orderword NPU status word over the internal data channel (IDC).

MULTIPLE CHARACTER DATA TRANSFER (BLOCK TRANSFER)

Block transfers use the direct memory access channel.

When executing the data transfer protocol, an arbitrary number of characters are transferred between contiguous locations in the PPU and a set of chained buffers in the NPU. The location of the characters in NPU memory and the operation of the buffer chaining mechanism are transparent to the PPU.

From the point of view of both NPU and PPU, input means data flowing upline; that is, from NPU to PPU. Similarly, output means data flowing downline, from PPU to NPU.

This operation of the coupler requires concurrent action of both the NPU and PPU. Either the NPU or the PPU may initiate the operation. When both have completed the setup, the transfer takes place.

TABLE 7-5. PPU FUNCTION COMMANDS

PPU Function Code	Octal Value	PPU Usage
Clear NPU	200	Used prior to loading or dumping the NPU. Stops the NPU and sets micromemory address register to location 0.
Start NPU †	040	Starts the NPU emulator (microcode) at the location in the micromemory address register. The emulator must always be started at location 0.
Input program	007	Used to dump NPU main memory.
Output program	015	Used to load the NPU main memory. Micromemory can neither be loaded nor dumped directly from the PPU.
Clear coupler	400	Resets the coupler's control logic and most registers. The protocol defined allows only the NPU to clear the coupler.
Output memory address zero and one	010 011	Sets NPU main memory accessing for loading and dumping.
Output orderword	016	Loads the coupler orderword register. Causes an NPU interrupt.
Input coupler status	005	Used to check the state of various registers and flip-flops in the coupler. Used to test whether the NPU has loaded the NPU status word.
Input NPU status	004	Inputs the NPU status word previously loaded by the NPU.
Input orderword	006	Allows the PPU to read back the orderword it had written. Used only prior to dumping the NPU.
Input data	003	Allows characters to be input to the PPU. The coupler must have been previously set up by the NPU.
Output data	014	Allows characters to be output from the PPU. The coupler must have been previously set up by the NPU.

† Must be delayed at least 10 ms following a clear NPU function code.

TABLE 7-6. NPU FUNCTION COMMANDS

NPU Command	Hexadecimal Value	NPU Usage
Input switch status	0654	Allows the NPU to check PPU data channel device address, on-line/off-line switch setting, alarm override switch setting. Executed during initialization.
Output buffer	0658	Sets the coupler to follow the NPU buffer chains for the current buffer length in use. Executed during initialization.
Clear coupler	060C	Resets the coupler control logic and most registers. Used during protocol error processing. The contents of the NPU status word are not affected.
Input coupler status	0650	Used in the NPU interrupt handler to determine the reason for interrupt.
Input orderword	0660	Used in the NPU interrupt handler to input the orderword previously loaded by PPU.
Output NPU status	0648	Used to send control codes to the PPU.
Output memory address	066C	Used to set up the coupler for data transfer. Points the coupler to the start of an NPU buffer chain.

The PPU sends a function to the coupler, either to input data or to output data. During an output operation the PPU cannot directly determine if the NPU has set up its side of the coupler to transfer the data. The determination is accomplished by the preceding communications, during which the NPU and PPU agree that setup for output will be the next thing done by both sides. For an input operation, after the PPU has sent a function to the coupler and has activated the channel, the PPU can test the channel to determine if a first buffer address is specified for the transfer and if the NPU status indicates that the NPU has input data available. If so, the NPU is set up and the transfer can take place. If not, the NPU sets up the coupler. The channel should become ready for transfer within 12 ms of the input data function command to the coupler.

The NPU sets up its side of the coupler for data transfer by writing the address of the first buffer of a chain to the coupler address register (buffer length is set up during initialization).

The high order four bits of each PPU data word control the operation of the output transaction, although bits 10-8 are not used in the defined protocol and are always set to zero. (If any of bits 10-8 are set, NPU buffer chaining occurs at other than end-of-buffer. This causes excessive buffer use in the NPU.) Bit 11 is set to 1 on the last character of the transaction; this causes the coupler to stop storing data into the NPU memory. The PPU disconnects the channel following transfer of this flagged word.

Input transfer is terminated when the last character of an NPU buffer is transmitted, and when bit 11 in the last word of the buffer is 1. The last character transferred is stored in PPU memory with bit 11 set. The coupler automatically disconnects the channel after this word is transferred.

It should be noted that a service message is handled by block transfers, although such messages have a control rather than a message transfer function. Interpretation of service messages is discussed throughout this manual according to the type of service message.

Checking data transfers is discussed below under the timeout and error checking heading.

CONTENTION FOR COUPLER USE

The coupler performs block mode transfers in only one direction at a time (half-duplex protocol). Either the NPU or PPU can request the channel at any time. The NPU requests the channel by setting the output memory address to point to the start of the input block buffer chain, and then by setting the output NPU status with one of the input available status codes. The PPU requests the channel by sending a function to the coupler to output the orderword with one of the output codes.

If the NPU and PPU both request to use the channel at approximately the same time, output is usually favored. This is accomplished by changing the value in the coupler's memory address register to point to an output buffer chain and responding with a "Ready for output" in the NPU status word. The NPU will re-request the channel at the completion of the output transaction.

When the output transaction is completed, the PPU starts a brief (1-10 ms) output-continue timer cycle to allow the NPU to request input, if the NPU has data queued for the PPU. This timer prevents the PPU from monopolizing the channel with output operations and thereby flooding the NPU.

If the NPU has a scarcity of buffers, it rejects the PPU's request, thus regulating output data. To limit the frequency of NPU output-request-driven coupler interrupts to the NPU during this data regulation period, an output rejected timer cycle of 100 ms is used.

REGULATION OF COUPLER USE

The primary objective of host regulation is to:

- Prevent saturation or overloading of the host or network in the event of an abnormality (emergency regulation).
- Allow data flow between the network and the host to ensure that continuity of service and performance standards are maintained.
- Smooth data flow (prevent over-regulation) using appropriate feedback control techniques.

The host coupler interface is a controlled, variable bandwidth I/O channel, in which the bandwidth is increased or decreased by a combination of load-balancing and reaching regulation thresholds.

Normally, the NPU accepts all input offered by the PPU. When buffer availability levels drop below pre-defined thresholds, the NPU uses the priority level defined below to reject downline messages from the host:

Priority	Message Type
1	Service messages.
2	Data blocks and related forward and reverse supervision of the highest priority.
3	Data blocks and related forward and reverse supervision at the lowest priority.

Each of these message types is kept in a separate queue in the host. Regulation in the NPU occurs by the NPU first rejecting output offered at level 3, then rejecting levels 3 and 2, and in an extreme situation, rejecting all output offered by the PPU. As buffer levels rise above these regulation thresholds, the NPU reverses this procedure until the unit is again capable of accepting all outputs.

The order in which the PPU offers the various output levels is determined by host considerations.

There are also two classifications of upline messages:

Classification	Message Type
1	Data and supervision less than 256 bytes in length.
2	Data and supervision greater than 256 bytes in length.

Both types of message are kept on a single queue in the NPU.

There is no priority associated with the two upline classifications offered by the NPU to the PPU; the separation into two length ranges is only to allow the PPU to utilize its buffer space more efficiently.

HOST FAILURE AND RECOVERY

A special case of regulation occurs when the host fails and when it recovers.

When the NPU software determines that communication across the coupler has failed, a regulation level of zero is communicated to the other end of each logical link terminating at the coupler. This inhibits acceptance of further input traffic from terminals logically connected via the coupler. Additionally, an informative message will be sent out to each affected interactive terminal.

When the NPU software determines that communication across the coupler has been restored, a normal regulation level is communicated to the other end of each logical link terminating at the coupler. This enables input from terminals logically connected via the coupler and causes an informative message to be sent to all affected interactive terminals.

ERROR CHECKING AND TIMEOUTS

The data transfer physical protocol checks for:

- Contaminated data
- Incomplete transaction
- Failure of interface to respond

The first two types of errors are handled at the physical protocol level by accepting only good blocks, and by discarding bad blocks in their entirety. The physical level protocol does not re-transmit blocks. The coupler is assumed to provide a noise-free channel and to generate only hard (rather than intermittent) failure modes. Errors are detected and logged by the host.

Interface failure causes the interface to be declared down, but the protocol returns to the initial state and continues to wait for interface response. Both the PPU and NPU have timers implemented locally to accomplish failure detection. A keep-alive timer of one second duration generates a periodic idle status, made available to the PPU when no traffic is in progress. The PPU deadman timer provides a ten second duration signal. This timer expires only if the PPU fails to receive either an idle or input request during that period. If the timer expires, the PPU declares the NPU to be down and enters the NPU dump/reload sequence.

The NPU deadman timer also provides a 30 seconds duration signal. If the NPU fails to receive a coupler interrupt within this period, it declares the host unavailable. The NPU deadman timer is not explicitly shown in the NPU protocol flow diagram (figure 7-2), but it is implicit in all places where the NPU is waiting for an interrupt.

HOST/NPU INTERFACE SEQUENCES

Figures 7-5 and 7-6 show the interface protocol sequences as viewed from the host and NPU, respectively.

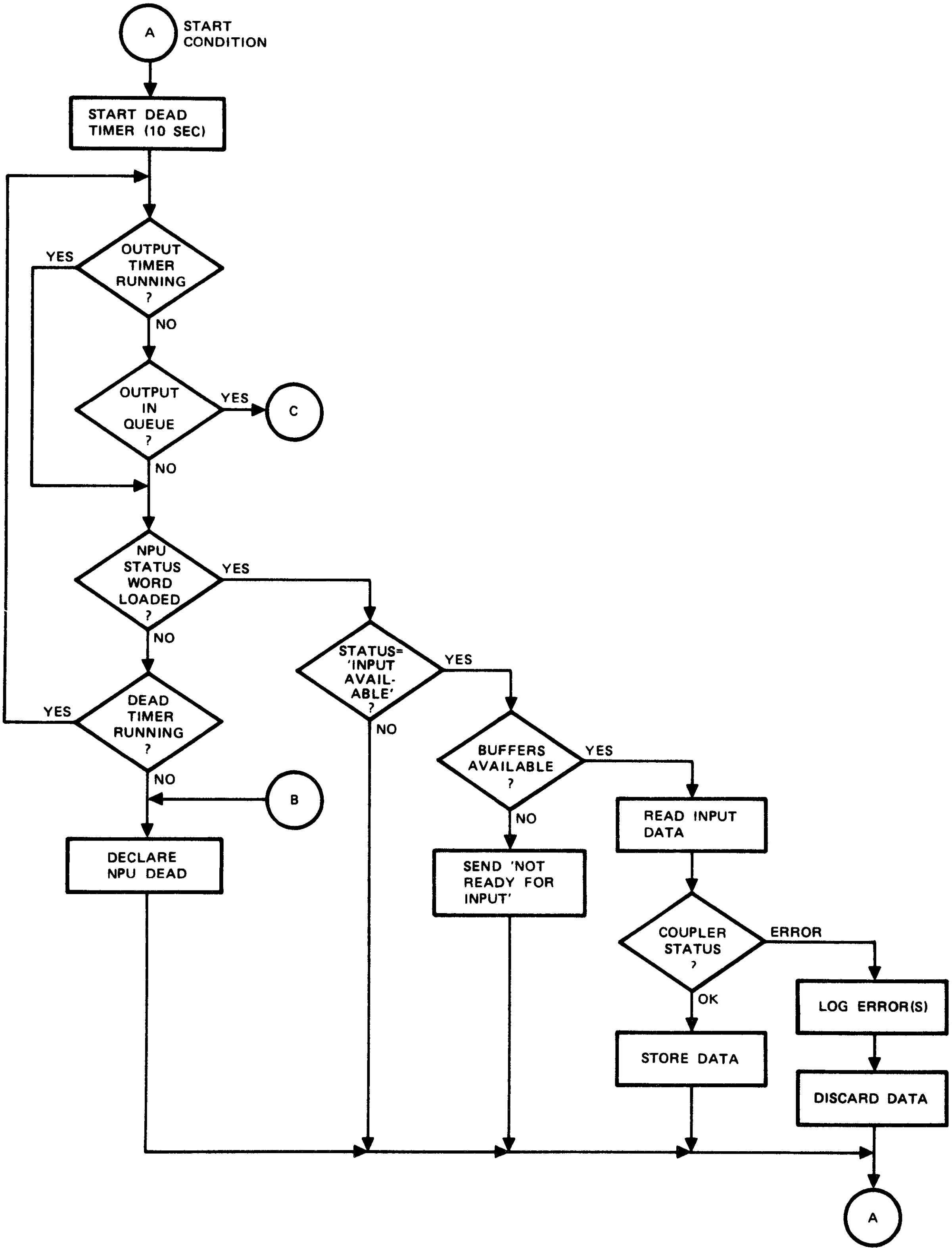


Figure 7-5. Host Interface Protocol Sequence, Host Side (sheet 1 of 2)

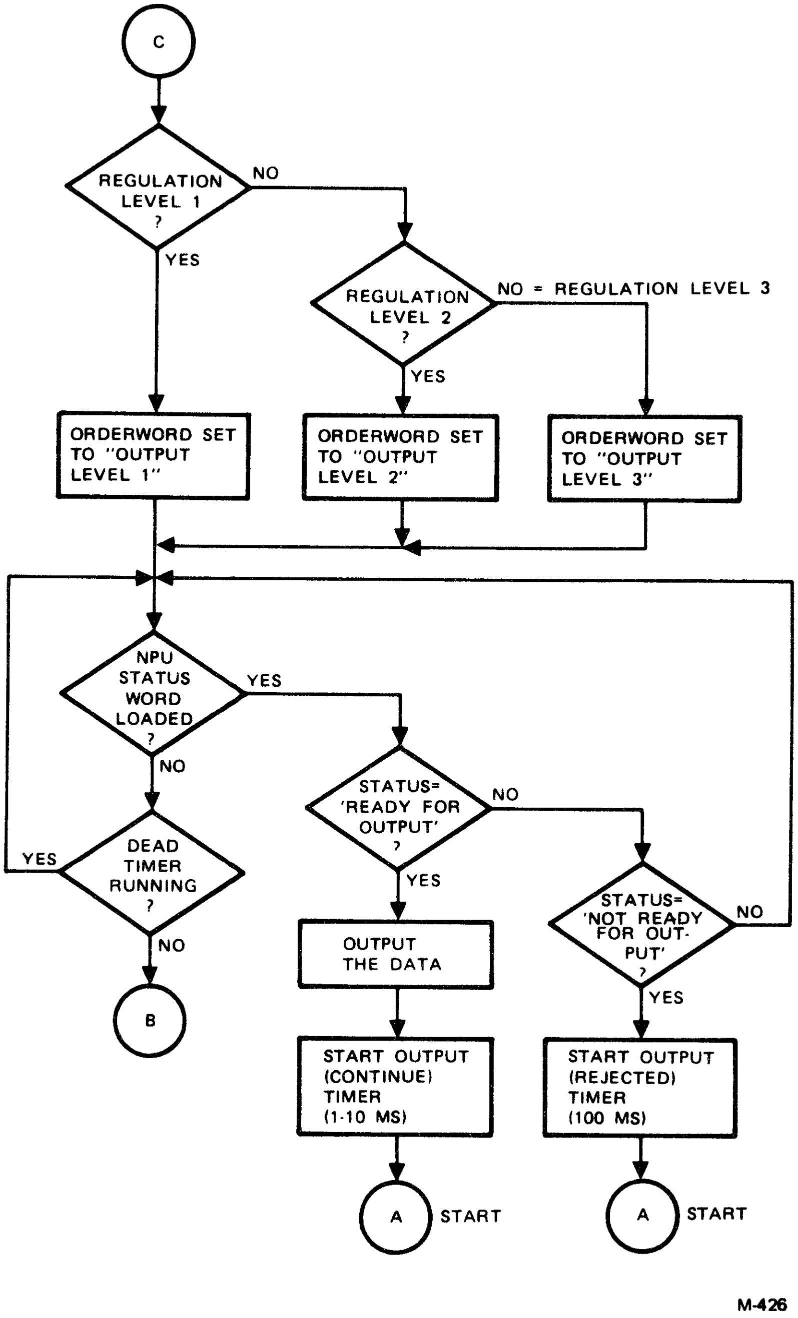


Figure 7-5. Host Interface Protocol Sequence, Host Side (sheet 2 of 2)

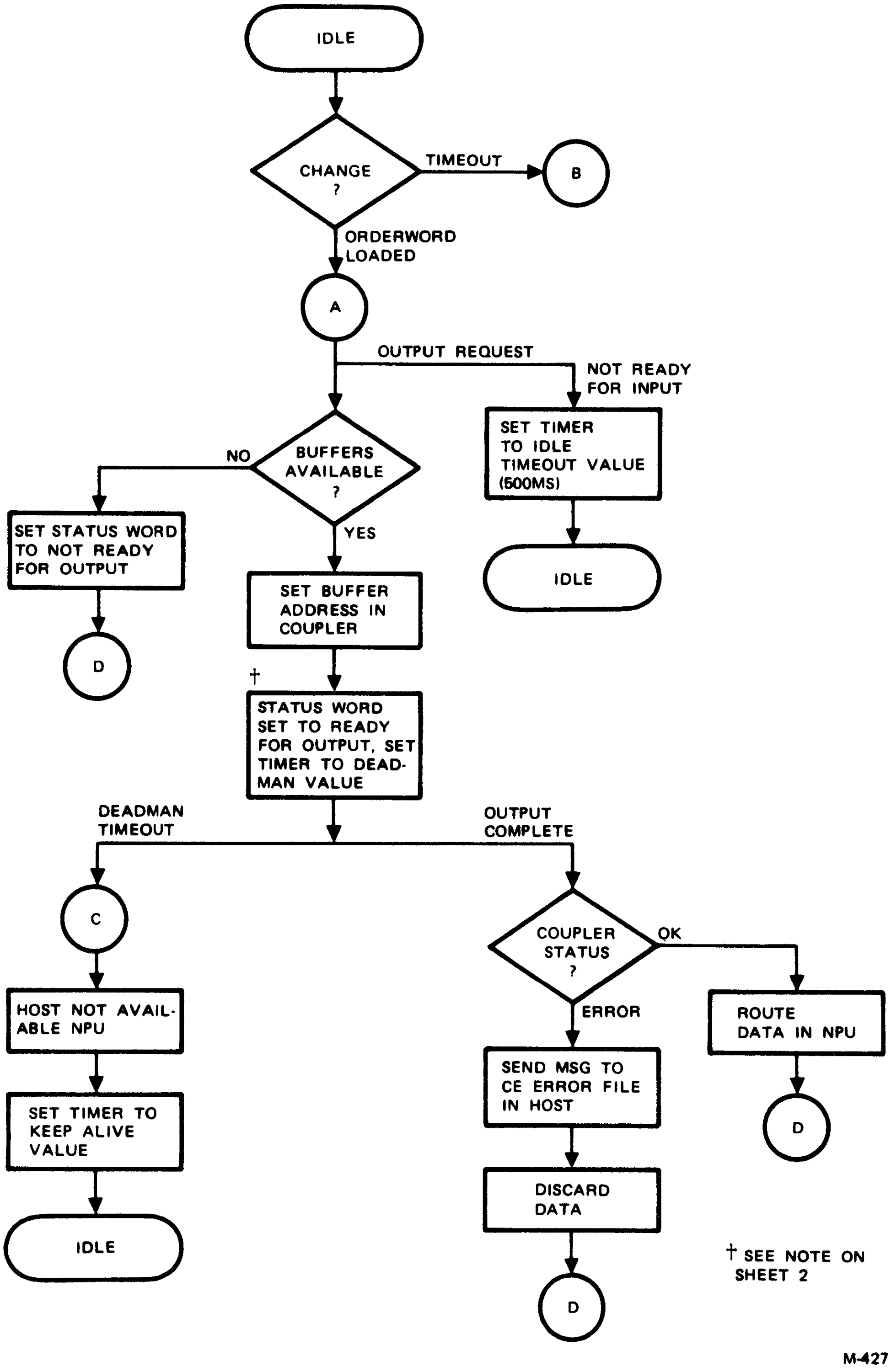
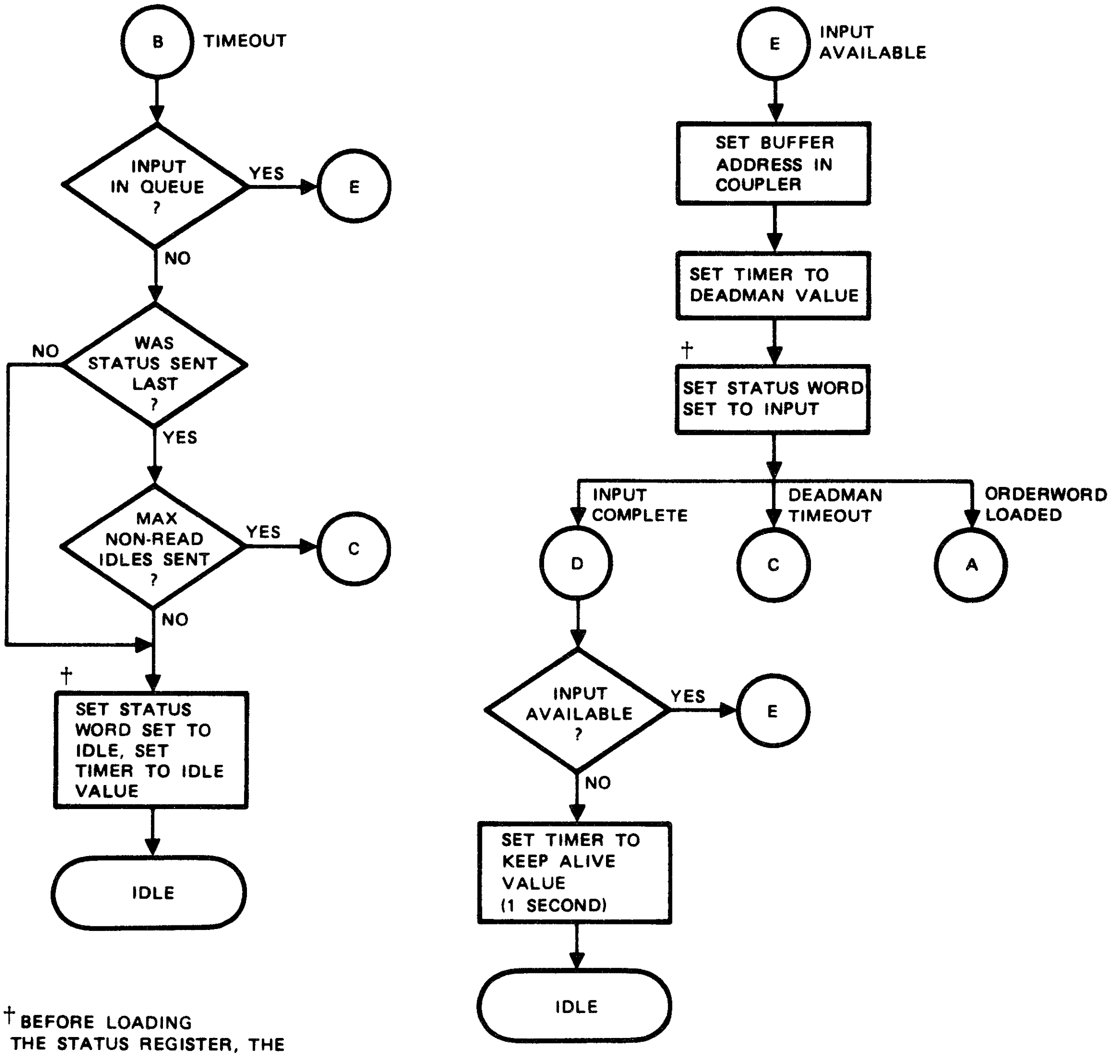


Figure 7-6. Host Interface Protocol Sequence, NPU Side (sheet 1 of 2)



†BEFORE LOADING THE STATUS REGISTER, THE STATUS IS CHECKED TO VERIFY IT IS NOT STILL LOADED FROM A PREVIOUS TIMER. IF IT IS, A WORKLIST IS MADE BACK TO THE OPS LEVEL HIP TO RE-EXAMINE THE STATUS.

Figure 7-6. Host Interface Protocol Sequence, NPU Side (sheet 2 of 2)

NOTE

> In figure 7-6 the large arrowhead (↓) indicates that the NPU is waiting for the next coupler interrupt. While waiting, the coupler program re-entry point is saved in a state vector, the deadman timer is running, and the NPU is servicing other processes. When the interrupt occurs, the NPU resumes servicing the coupler at the location specified by the state vector. If the reason for interrupt is one of the items listed below the arrow, service proceeds as shown. If the interrupt occurred for some other reason, an error has occurred. Such an error is logged in the CE error file and the protocol is restarted at Ⓐ. If the deadman timer timeout occurs before the interrupt, the HIP calls a routine to note that the host is unavailable, and then restarts the protocol at Ⓐ.

The principal features of the protocol detailed by the flow charts are as follows:

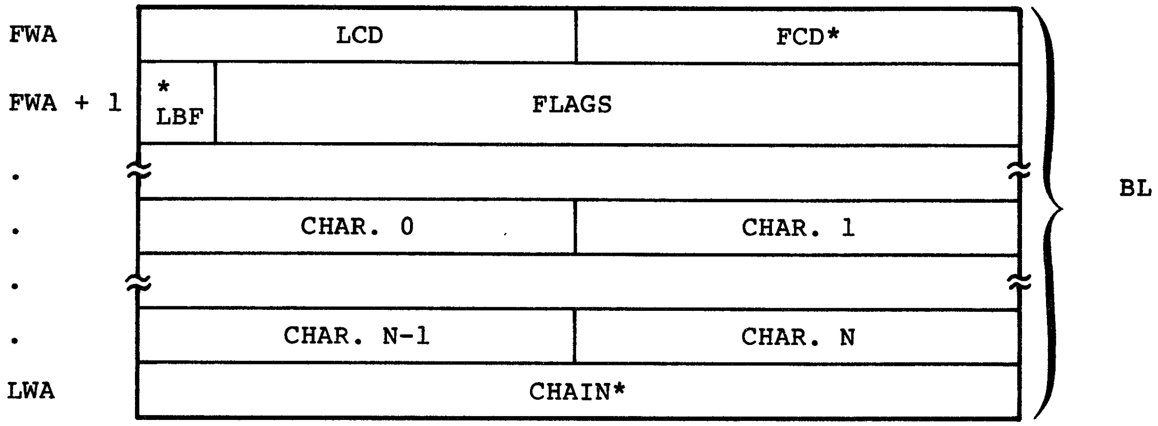
- The NPU can specify input available and set up the coupler for input data transfers at any time.
- The PPU can order output at any time.
- If conflict occurs, the NPU normally allows output from the PPU.
- The NPU can refuse to take PPU output if the NPU does not have sufficient buffer space for the transfer.
- The PPU can refuse input from the NPU by requesting output or by responding with a 'not ready for input'.
- If either the NPU or the PPU deadman timer expires, protocol is reset to the start condition, but continues.
- If a given output type is refused by the NPU, the PPU performs a short timeout before re-requesting output, to prevent swamping the NPU with interrupts. The type of output offered in succeeding attempts is determined by the host logic.
- If output is accepted by the NPU, the PPU allows the NPU to indicate if input is available, before again ordering output.
- Once data transfer is initiated, the transaction must be complete. If it does not, the entire transaction unit is discarded.
- Error checking is performed by the receiving device. If an error is detected, a CE error message is sent to the host engineering file, any received data is discarded, and the protocol is reset. No attempt is made to retransmit the data.

BUFFER FORMAT

The HIP requires all using programs to provide or accept data blocks in standard format. Figure 7-7 shows format that is a variation of standard block format.

HIP STATES

The HIP can be considered a passive program that passes from one state to the next as a result of a stimulus from an external event. Table 7-7 shows the HIP as a state driven program.



BL = Buffer Length (in 16 bit words) $BL = 2^x$, $2 \le x \le 7$

FCD = First Character Displacement (relative to FWA) $4 \le FCD \le 253$

FLAGS = Bit indicators which provide additional information about the data or data buffers.

FWA = First Word Address of Buffer (must be an integer multiple of BL)

LBF = Last Buffer Flag (1 = last)

LCD = Last Character Displacement (relative to FWA) $4 \le LCD \le 253$, BL LCD/2 + 1

LWA = Last Word Address of Buffer LWA = FWA + BL - 1

CHAIN = FWA of next data buffer (may contain zero value when LBF = 1)

Figure 7-7. Standard Data Block Format Used by the HIP

TABLE 7-7. HIP STATES AND TRANSITIONS

Event State	Transfer Complete	Transfer Terminated by PPU	Orderword Loaded	Chain Address Zero	Transaction Timeout
AOPT0 IDLE	CE=Spurious Interrupt	CE=Spurious Interrupt	Start Output (AOPT3) Invalid Orderword → Halt	CE=Spurious Interrupt	Send Idle Inquiry
AOPT1 Idle Inquiry Sent	CE=Spurious Interrupt	CE=Spurious Interrupt	Start Output or Not Rdy for Input Invalid Orderword → Halt	CE=Spurious Interrupt	CK for Idle Response (deadman timeout) Send Idle Inquiry
AOPT2 Input Completion	NORMAL INPUT COMPLETION	CE=Transfer Term by PPU Release Input Block	Terminate Input, Start Output (AOPT3) Invalid Orderword → Halt	CE=Chain Address Zero, Release Input Block	Host Down to SVC Module, Requeue Input Message
AOPT4 Output Completion	NORMAL OUTPUT COMPLETION	CE=Transfer Term by PPU Release Output Buffers	CE=End of Operation Missing Release Output Buffers Invalid Orderword → Halt	System Halt (JOCHAIN)	Host Down to SVC Module Release Output Buffers
AOPT5 AOPT6 Delay	No Action	No Action	No Action	No Action	No Action

LINK INTERFACE PACKAGE MODULE 8

The LIP module is responsible for handling transmission and reception on both ends of a trunk; therefore, a version of the LIP must exist in both the local and remote NPU.

Since the current CCP version permits only direct coupling from a remote NPU to a local NPU, the terms trunk, logical link, and physical link are synonymous for this connection. Two major types of operations are handled by the LIP:

- Loading/dumping of the remote NPU. This operation is discussed in the CCP 3 Reference Manual (see preface).
- Transmission of data (messages) over the trunk. Figure 8-1 shows the functions involved in such transmissions. Note the division of functions between local and remote NPUs. Table 8-1 contrasts local and local/remote systems.

This section discusses LIP operation in five major categories:

- Trunk protocol
- Transmit functions
- Receive functions
- Trunk enabling/disabling
- Trunk failure/recovery

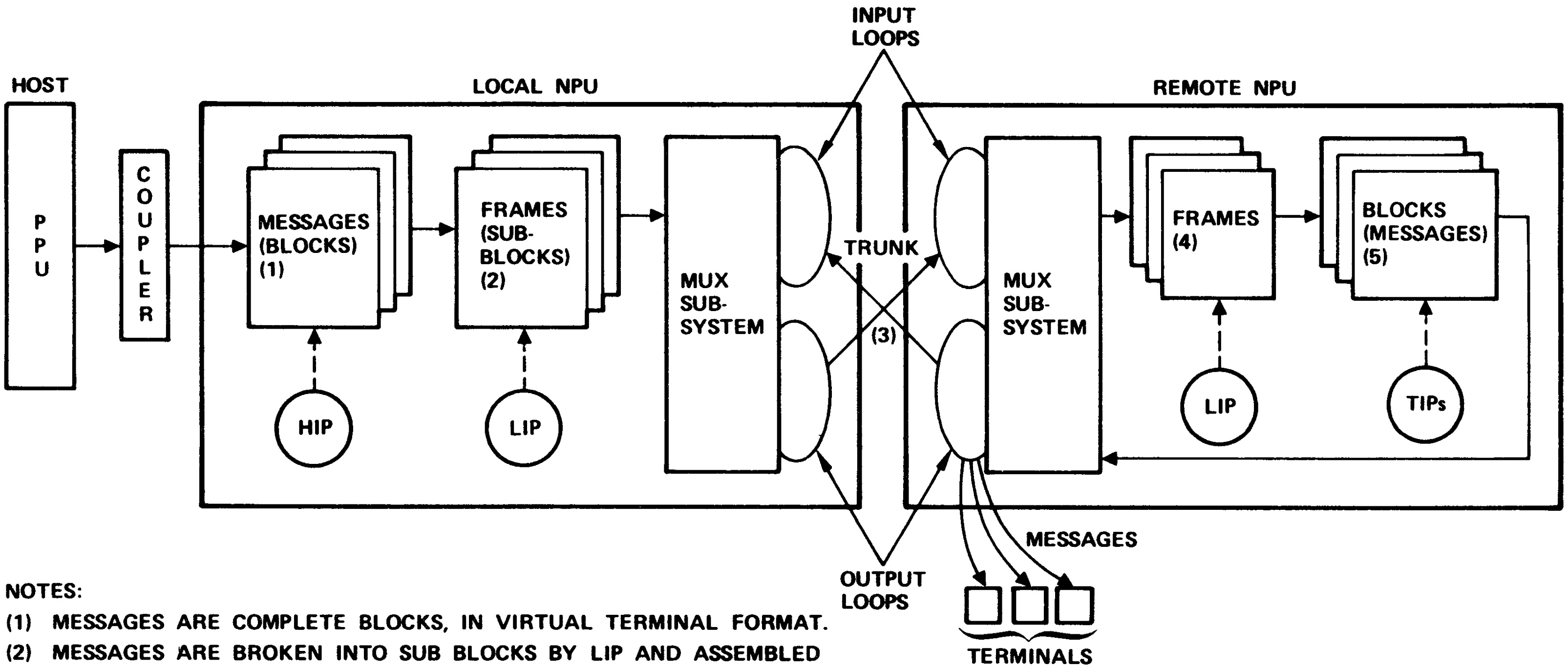
TRUNK PROTOCOL

The LIP implements a class of the Control Data Corporation Control Procedure (CDCCP) for information interchange. CDCCP treats each trunk as a separate entity and is not concerned with the contents of the information frame. The specific protocol implemented is equivalent to ISO HDLC class, using the symmetrical, asynchronous response mode, and using the basic numbering range with two-way, simultaneous reject and initialization options.

Either end of the link can initiate data transmission when conditions warrant. The interfacing LIPs first establish the nominal mode: the local NPU sends the set-asynchronous-response-mode (SARM) frame; the remote NPU replies with an unnumbered-acknowledgement (UA) message indicating that asynchronous response mode (ARM) has also been established in the remote NPU. Then data transmission begins.

The basic unit of transmission over the trunk is a trunk transmission frame (TTF). Format of the frame (8-bit bytes) is shown in figure 8-2. There are three types of frames:

- Unnumbered frames that establish the basic transmission states between the two nodes, such as initialization and command rejected.



NOTES:

(1) MESSAGES ARE COMPLETE BLOCKS, IN VIRTUAL TERMINAL FORMAT.

(2) MESSAGES ARE BROKEN INTO SUB BLOCKS BY LIP AND ASSEMBLED INTO FRAMES CONTAINING ≤ 255 BYTES EACH (FIGURE 8-2).

(3) FRAMES TRANSMITTED OVER TRUNK AT ≤ 9600 BAUD. OUTPUT LOOP OF LOCAL NPU CARRIES DATA/CONTROL INFORMATION; INPUT LOOP RECEIVES ASSURANCE CONTROL INFORMATION (ALSO IN FRAMES).

(4) RECONSTITUTED FRAMES ARE CHECKED BY LIP PRIOR TO PASSING DATA AS RECONSTITUTED BLOCKS TO TIPs FOR PROCESSING.

(5) TIPs PROCESS MESSAGES, CONVERT TO TERMINAL FORMAT, AND QUEUE FOR OUTPUT TO INDIVIDUAL TERMINALS.

Figure 8-1. Simplified Trunk Operation (Output Only)

TABLE 8-1. COMPARISON OF LOCAL AND LOCAL/REMOTE NETWORKS

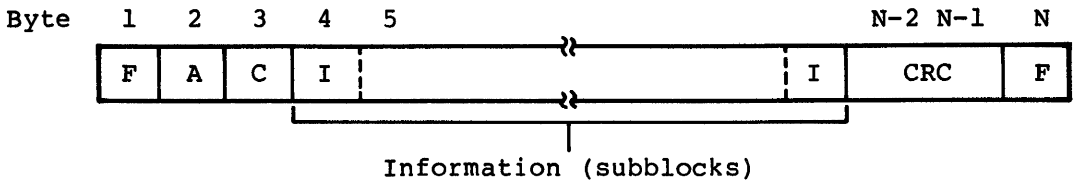
Local	Local/Remote
Terminals local only	Terminals remote; can also have local terminals.
Terminal data multiplexed locally; TIPs place data in virtual terminal format (upline) or real terminal format (downline). Upline data passed through TIPs to HIP, thence to host	Remote; same on downline. Upline data collected into frames (made up of subblocks) after convert to IVT or BVT format; transmitted via trunk to local NPU. No HIP in remote NPU. Local: Data from local terminals treated the same. Data from remote terminal treated same as for upline data except LIP reconstitutes and checks frames. Then reconstitutes subblocks into message to pass to HIP. Downline data broken into subblocks, assembled into frames, then sent to remote NPU via trunk; still in virtual terminal format.
Load/dump NPU through coupler	Load/dump local NPU through coupler; load/dump remote NPU using overlay in local NPU, transmission over trunk, and bootstrap program and cassette in remote.

- Supervisory frames that establish whether transmission or reception is currently possible (ready for data/not ready for data/rejected last data sent) and that provide frame acknowledgement information.
- Information frames used to transmit message data. This class of frames includes frames that are carrying service messages.

Before data framing, the messages (blocks) are queued in the link queues on a first-in first-out basis. Each NPU has two such queues, one for high-priority messages, the other for low-priority messages. The queues hold pointers to the blocks which can either be a single buffer or a chain of buffers (subblocks) making up the message. From the link queue, individual subblocks are passed to the text transmission queue and then to the frame. The entire subblock need not be included in the information bytes of the frame. All that is necessary is the data part of the buffer. This is the part delimited by FCD-LCD in the buffer, as shown in figure 8-3.

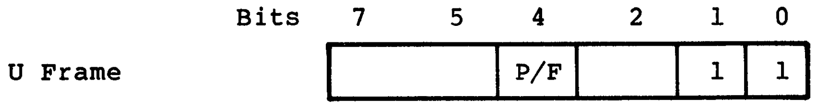
When the frame is filled (that is, the next subblock would cause a frame overflow condition), the frame CRC is generated and the frame is sent to the neighbor NPU (assuming the trunk protocol has been established).

FRAME FORMAT



C - Control byte (can be U, S, or I frame)

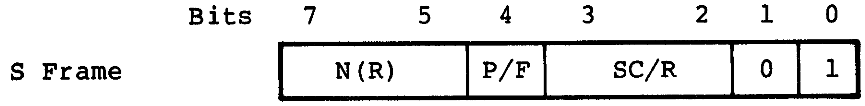
CONTROL BYTE (C Field)



P/F - Poll/final flag

0 = poll
1 = final

Bits 7 - 2	Function	Protocol Element
000P00	Unnumbered information	UI
000F01	Request initialization mode	RIM
000P01	Set initialization mode	SIM
011F00	Unnumbered acknowledgement	UA
100F01	Command rejected	CMDR
000P11	Set asynchronous response mode	SARM



N(R) - Sequence number of next frame expected in receiving NPU

P/F - Same as U frame

SC/R - Supervisory command/response

Bits	Mnemonics
00 Receive Ready	RR
01 Receive Not Ready	RNR
10 Rejected	REJ
11 Not Used	

Figure 8-2. Frame and Subblock Format (sheet 1 of 2)

Bits	7 5	4	3 2	1	0
I Frame	N(R) = 0	P/F	N(S)		0

P/F - Same as U frame
N(S) - Sequence number of frame

SUBBLOCK FORMAT FOR INFORMATION (I) FIELD

L	FLG	Message data bytes

FLG - Disassembly flags

Bits 7 - Priority 1 = high priority
6 - last subblock; 1 = true
5 - 0 - Unused

F - Flag is a unique bit pattern (01111110) to identify start and end of frame. A zero bit is inserted after every string of five L's where a frame is transmitted, and removed at the receiving NPU; F bytes are added by transmitting CLA.

A - Receiving node address

0 = local
1 = remote

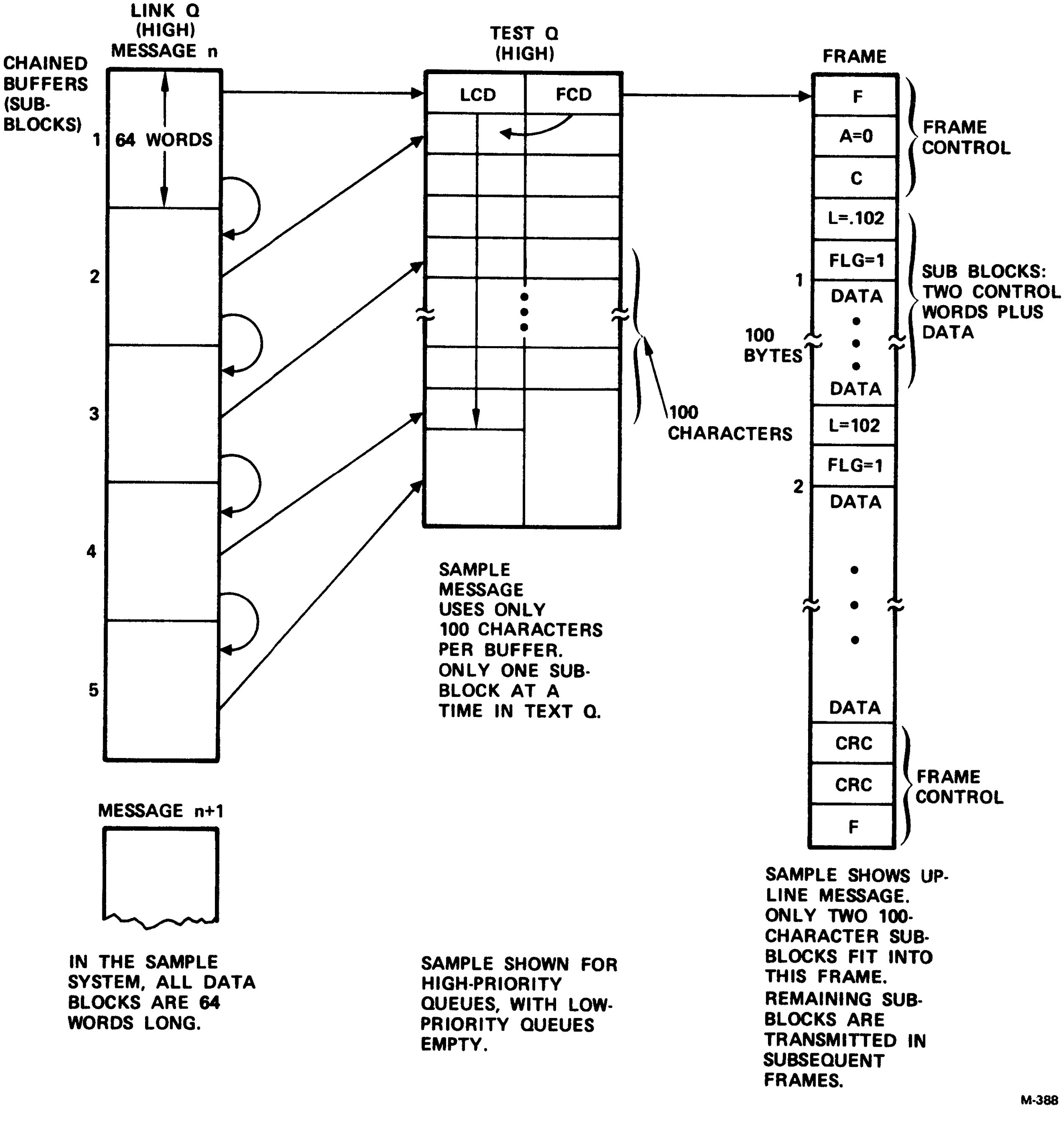
CRC - Two cyclic redundancy bytes added by transmitting CLA.

N - Maximum frame size determined by the build time parameter MAXFRMSZE (nominally set to 259); excludes the beginning and ending F bytes and CRC bytes added by the CLA when transmitted

I - Appears only when control byte is I or in a UI frame

L - Length of subblock: $3 \leq L \leq 257$

Figure 8-2. Frame and Subblock Format (sheet 2 of 2)



IN THE SAMPLE SYSTEM, ALL DATA BLOCKS ARE 64 WORDS LONG.

SAMPLE MESSAGE USES ONLY 100 CHARACTERS PER BUFFER. ONLY ONE SUB-BLOCK AT A TIME IN TEXT Q.

SAMPLE SHOWN FOR HIGH-PRIORITY QUEUES, WITH LOW-PRIORITY QUEUES EMPTY.

SAMPLE SHOWS UP-LINE MESSAGE. ONLY TWO 100-CHARACTER SUB-BLOCKS FIT INTO THIS FRAME. REMAINING SUB-BLOCKS ARE TRANSMITTED IN SUBSEQUENT FRAMES.

Figure 8-3. Sample Frame Formation

Upline Data - Instead of being passed from the TIP to the HIP (as in a local NPU only system), the message is passed to the remote NPU's LIP and through the two queues, as shown in figure 8-4. Then the message data is placed in the frame. After transmission over the trunk, the local NPU's LIP checks the transmission, strips away the frame, reconstitutes subblocks into whole message blocks, and passes these blocks to the HIP for upline transmission to the application program in the host.

Downline Data - Downline transmission is shown in figure 8-5. Messages (blocks) that are still in virtual terminal format are passed through the HIP to the local NPU's LIP. The LIP converts the chained message buffers to subblocks to be used in the frame. When the frame is filled (or no more data is queued for transmission), the frame is sent to the remote NPU over the trunk.

At the remote NPU the frame is stripped off and the subblocks are reconstituted into chained message buffers, which are passed to the appropriate TIP to be converted to the output terminal's protocol.

Two priorities are associated with frames to allow a trunk regulation scheme. These priorities are as follows:

- Priority 1 (high). Normally this priority is assigned to messages from interactive terminals. Messages tend to be short but need rapid processing to avoid delays at the terminal.

- Priority 2 (low). Normally this priority is assigned to messages from batch terminals. Messages tend to be long (1000 bytes or more).

The scan system that generates the frames from subblocks scans four queues priority 1 TEXT Q, priority 1 link queue, priority 2 TEXT Q, and priority 2 link queue in the order given. In this manner, all priority 1 information that can fit in the frame is transmitted before any priority 2 level information.

Figure 8-6 shows the logical sequence of constructing a frame from subblocks extracted from the various queues. Blocks for internodal delivery are queued by link according to priority. These queues are input to the LIP. The LIP interrupts low-priority traffic delivery at frame boundaries in order to deliver queued high-priority traffic. This, in conjunction with an appropriately small frame size, optimizes high-priority response.

Information frames are constructed from subblocks with a total length not exceeding a defined maximum frame size. A subblock can be all or part of a block. Since frames must end on a block boundary, frames of fairly constant length are constructed whenever a sufficient number of subblocks are awaiting transmission.

Each trunk has a transmit in-process text queue (TEXT Q) for each priority. If the queue is not empty, TEXT Q contains the untransmitted remainder of a block that has been removed from the link queue and partially transmitted on the trunk.

Each frame is headed by the A and C fields (figure 8-2). Each subblock in the frame is headed by (1) an L field containing the length in characters of the subblock following, and (2) an FLG field containing a priority flag and an end-of-block flag. The L and FLG fields are used by the receiving LIP to restructure the original blocks for processing by the CCP program.

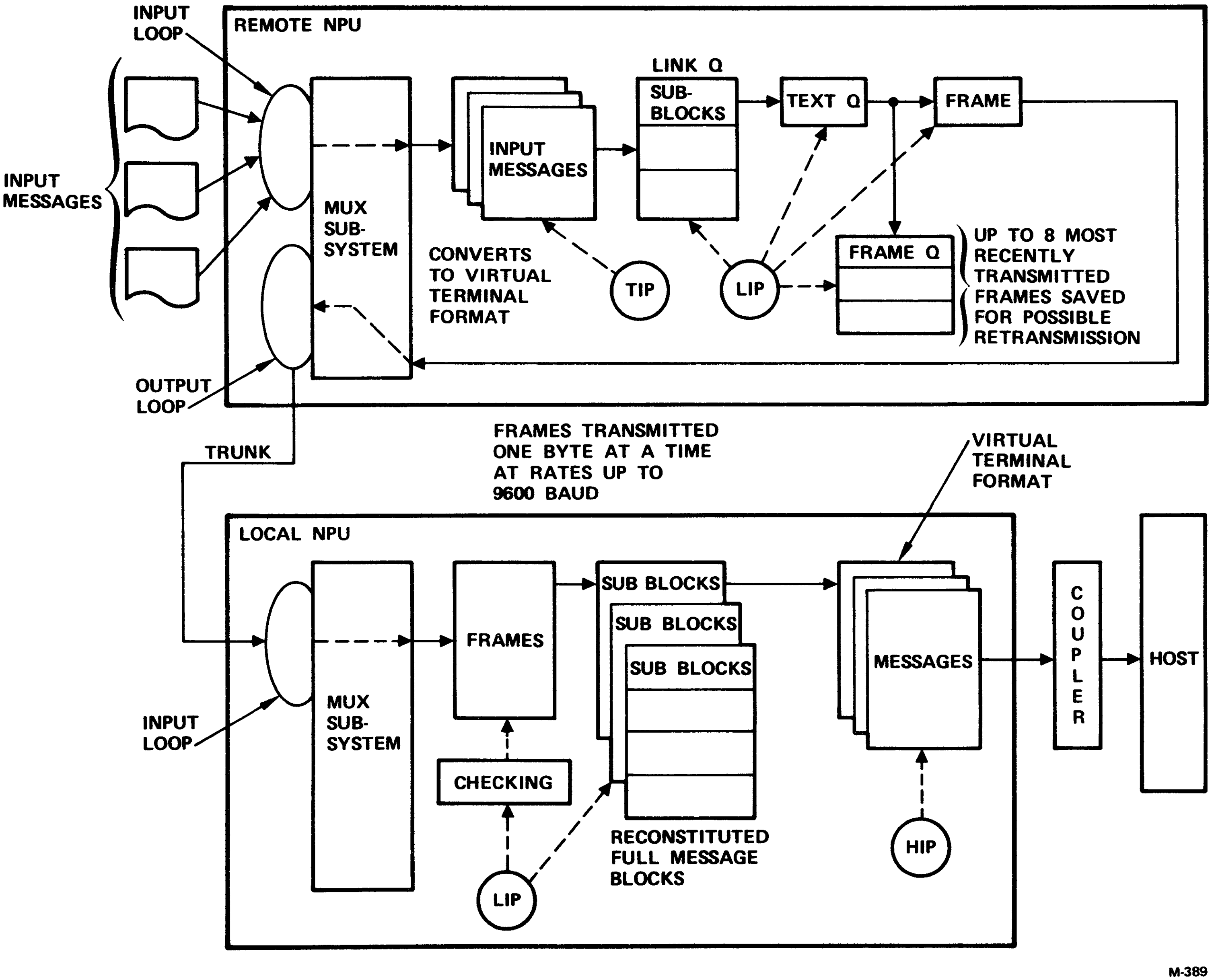


Figure 8-4. Sample Upline Message Transmission Over a Network Link

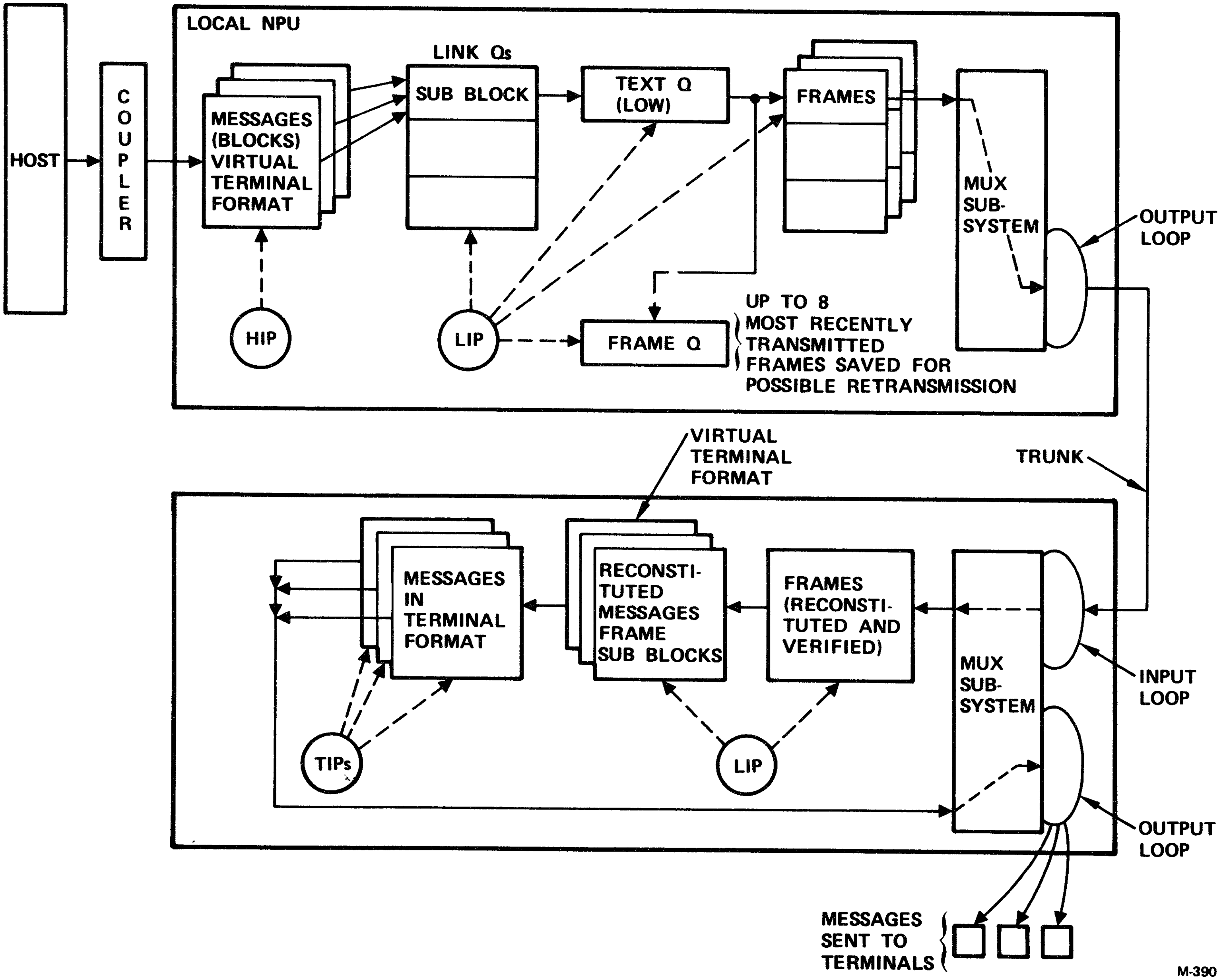


Figure 8-5. Sample Downline Message Transmission Over a Network Link

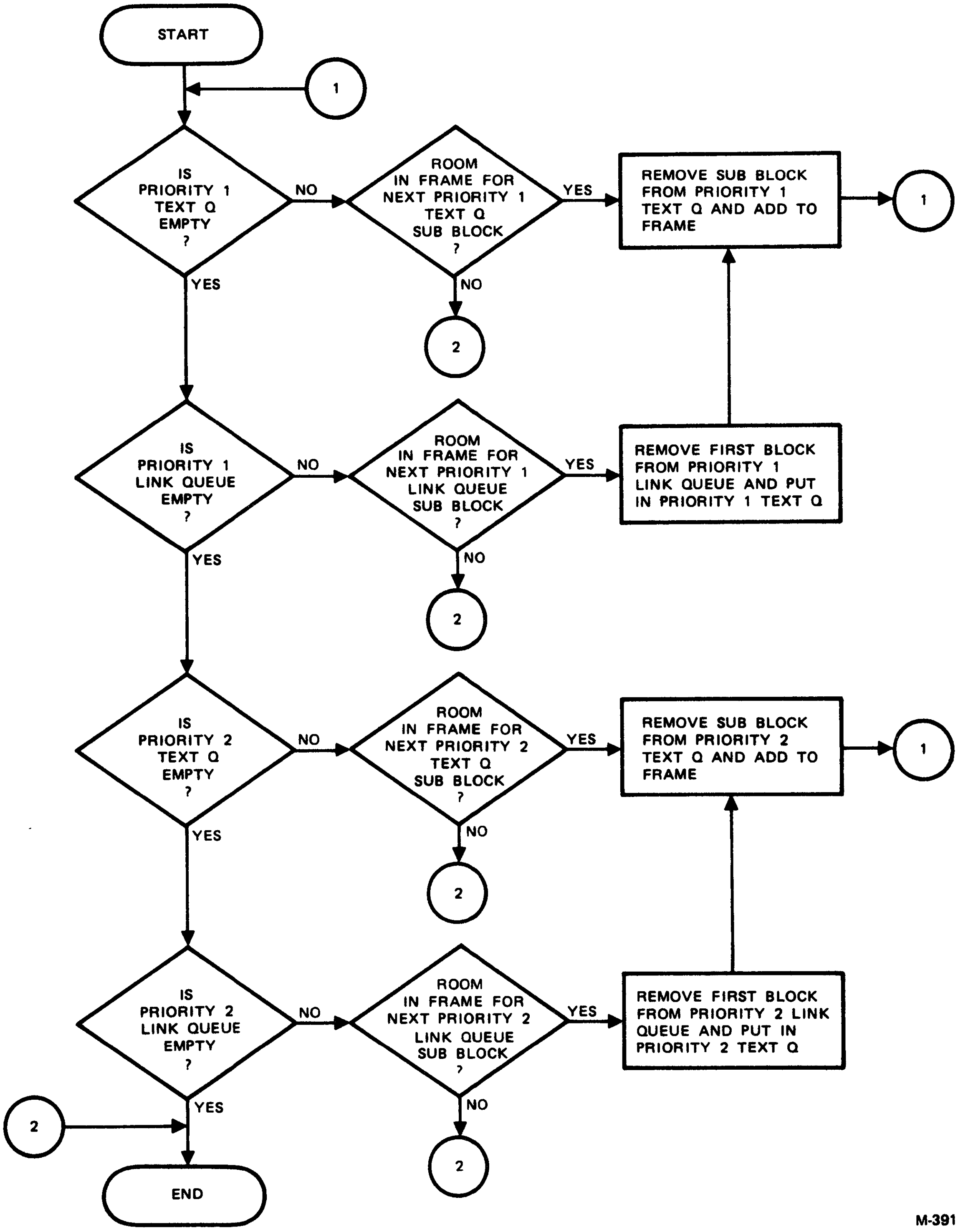


Figure 8-6. Frame Construction Flowchart

The system regulation level (0, 1, and 2 levels) as discussed in the CCP 3 Reference Manual are used in conjunction with supervisory frames to determine whether or not the receiving NPU can accept frames.

CHECKS AND RETRANSMISSIONS

Since there is a possibility that data will be garbled during transmission over a trunk, a cyclic redundancy check (CRC) is included in each frame by the communications line adapter (CLA).

Cyclic Redundancy Check

The cyclic redundancy check field is a 16-bit result of mathematical computation on the digital value of all bits in the frame (excluding inserted zeros). The transmitter performs the calculation and sends the result. The receiver performs the calculation and compares the result with the CRC received. If the comparison fails, the frame is discarded and must be retransmitted.

The CLA uses CRC procedure to determine the reliability of the incoming frame. The CRC field is the binary pattern found in multiplying the binary value of the A, C, and I fields by X^{16} and dividing the result by $X^{16} + X^{12} + X^5 + 1$. If, at the end of the received frame, the CRC field does not equal the calculated value of this remainder, the frame check sequence error (FCSE) status is sent to the controlling processor.

Retransmission is made possible by saving recently transmitted frames. If the frame acknowledgement fails to appear or indicates a bad frame, all frames up to the last properly acknowledged frame are retransmitted. These frames were previously saved in a Frame Retention Queue (FRQ) which is an eight entry list for each trunk. As an information frame is transmitted, it is entered into the frame retention queue according to its transmission sequence number. When acknowledged, the frames are released from the frame retention queue. Frames are retransmitted from the frame retention queue as necessary, starting with the oldest frame first.

TRANSMIT FUNCTIONS

Three types of frames can be transmitted (figure 8-2): unnumbered, supervisory, and information.

UNNUMBERED FRAME

The following control statements are transmitted as unnumbered frames (U frames):

- The set asynchronous response mode (SARM) message establishes normal transmission over the trunk.
- The SARM response message is an Unnumbered Acknowledgement (UA). This is also used to acknowledge UI messages.
- The request for initialization mode (RIM) message is sent when the remote NPU requires reinitializing (for instance, after a timeout).

- The response to a RIM message is a set initialization mode (SIM) message, acknowledging that the local NPU will commence the load or dump operation of the remote NPU using overlay methods.
- The unnumbered information (UI) message is used to transmit load or dump information.
- The command reject (CMDR) message is sent when the command (C) field of a received frame does not correspond to any of the legal C fields.

SUPERVISORY FRAME

Three types of supervisory frames (S-frames) are transmitted. All these frames respond to the condition of a frame just received.

- A receive ready (RR) frame is sent when either of the following occurs:
    - An information frame is correctly received and the receiving NPU can process more data (for instance, the next frame of a message).
    - A receive not ready (RNR) message was received but the poll/final flag is not set, and the regulation is not at zero (message transmission prohibited) level.
- A receive not ready frame is sent in response to an information frame or to a receive not ready message when zero regulation is in effect. This essentially causes the receive not ready message to be passed back and forth over the trunk until regulation level rises to at least level 1 or until the trunk is disconnected.
- A reject (REJ) frame is sent when an information frame is received without error but the sequence number, N(S), is not the one expected. The received frame is discarded and a reject frame is sent. All subsequent information frames are discarded until the expected frame is received.

INFORMATION FRAME

Information frames (I-frames) carry the network's message traffic over the trunk. The LIP generates an information frame (figure 8-2) by scanning the link and TEXT transmit queues as discussed previously. The information frame header consists of the address byte and the control byte. The sequence number of this frame, N(S), is placed in the control byte. This defines the slot in the frame retention queue where the pointer to this frame is to be stored.

RECEIVE FUNCTIONS

Frames received from a neighbor are processed according to type. Information frames contain information. Supervisory frames contain acknowledgements and can interrupt the flow or cause retransmission. Unnumbered frames indicate initialization is needed or an error has occurred and are processed by the LIP as necessary.

Acknowledgements come across the trunk in the control byte of a supervisory frame. The number N(R) is the neighbor's next expected number for the trunk. Thus all frames up to and including N(R)-1 that are saved in the frame retention queue may be released. Failure to receive an acknowledgement after a suitable time causes the transmitting NPU to poll for an acknowledgement. If the acknowledgement does not allow all frames to be released from the frame retention queue, the remaining frames are retransmitted. If repeated polls do not receive an acknowledgement, the trunk is declared inoperative.

An incoming receive not ready frame with the poll/final flag set causes the supervisory receiving NPU to reply as soon as possible. If regulation is not in effect, the response is a frame with receive ready; otherwise the response is a supervisory frame with receive not ready.

The value of the poll flag in a received information frame is returned to the sender in the final flag of the response generated for that frame.

Supervisory functions are performed when the following supervisory frame responses are received:

- Receive ready - Acknowledgement frames as described above.
- Receive not ready - The sending NPU inhibits further information frame transmission over the trunk. A supervisory frame with receive not ready and the poll flag is sent to inquire if the receiving NPU can again receive information frames. The trunk is declared inoperative if, after several inquiries, the receiving NPU is not ready to receive.
- Reject - After the acknowledgement contained in the reject supervisory frame is processed, all frames remaining in the frame retention queue are retransmitted, starting with the oldest frame.

Certain unnumbered commands and responses can be received during the normal protocol. Any event not mentioned causes a command reject (CMDR) to be sent. Receiving a command reject causes the trunk to be declared inoperative.

- Request initialization mode - This indicates the neighbor NPU has failed and the load/dump process is to be initiated.
- Command rejected - The information field contains the reason the command was rejected. The event is noted in the statistics block and the trunk is reinitialized using the set asynchronous response mode - unnumbered acknowledgement handshake procedure.
- Set asynchronous response mode - An unnumbered acknowledgement is immediately transmitted on the trunk.

TRUNK ENABLING AND DISABLING

Enabling is the result of normal operations that attempt to bring the trunk up. Enabling can also be operator initiated following an operator-initiated disabling command.

When a disable trunk service message (SM) is received by the local node, the protocol is stopped at that node and a normal response (trunk inoperative) SM is sent to NS. The LIP does not service the trunk until an enable trunk SM is received by the local node. Upon receiving an enable trunk SM, the local NPU sends a normal response (trunk operational) SM to NS, and the link initialization procedure is restarted. (Disabling takes place at both ends of a trunk independently.) One end of a trunk can be enabled, with the LIP at that end attempting to maintain trunk protocol, while the other end is disabled.

Receiving a disable trunk command is such a case. The remote node sends an abnormal response (cannot disable last path to NS) SM to NS. An enable trunk SM received by the remote node causes a normal response (trunk operational) SM to be sent to NS.

TRUNK FAILURE/RECOVERY

These operations result from hardware or software errors. After the trunk is declared operational, the local node and the remote node monitor both directions of data flow (receive and send). If no data is available to transmit, the LIDLE element of the link control block type (ACTL) is periodically sent to the other end. The LIDLE element format is shown in figure 8-7.

When the protocol indicates an inability to send data, or neither a data block nor a LIDLE has been received in time, the trunk is declared inoperative. The local node informs NS by sending the host an unsolicited trunk status SM. The LIP discards all data blocks upline and downline after a trunk failure.

The link initialization procedure is used to recover following a trunk failure. After a successful exchange of LINIT frames between local and remote nodes, the local node reports the trunk as operational to NS. Normal data blocks may then travel upline and downline over the trunk.

Byte	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	F	A	C	I	L	FLG	DN	SN	CN	TYPE	SUB-TYPE	RL	CRC		F

I-bytes of subblock

F - Frame flag

A - Receiving node ID

C - I-frame

L, FLG - As defined for I-frame

DN, SN - Destination and source nodes (terminal nodes of sending and receiving NPUs)

DN - Connection numbers

TYPE - Specifies priority, block type and serial number as follows:

Bit	7	6	4	3	0
	P	BSN		BT	

P - Priority
BSN - Block serial number
BT - Block Type = 15 (ACTL)

SUBTYPE - 4 = LIDLE
3 = LINIT

RL - Regulation level of sender

00 = no messages accepted
01 = high priority accepted
02 = low priority accepted

Figure 8-7. LIDLE or LINIT Frame Format

ASYNCHRONOUS (ASYNC) TIP 9

The Asynchronous (ASYNC) TIP supports dedicated and dial-up asynchronous lines that serve freewheeling terminals[†] operating at standard rates in the range between 110 and 9600 baud. The TIP provides software support for Teletype, IBM 2741, and teletypewriter-compatible CRTs that operate in an interactive mode with host applications. The TIP supports seven separate types of terminals. In addition, by means of the IVT control command, a user at his terminal can alter parameters for any of the seven standard terminals to create new terminals, which are also supported.

HARDWARE CONSIDERATIONS

The seven types of terminals supported by the TIP are the following:

Terminal Class	Manufacturer	Model Number
1	Teletype	M33, 35, 37, 38
2	CDC	713-10
4	IBM	2741
5	Teletype	M40/2
6	Hazeltine	2000
7	CDC	751-10
8	Tektronix	4014

Appendix C gives the default parameters for each of these terminals and also defines terminal class and subTIP.

The basic features of the TIP are as follows:

- Each line has a single terminal. Clusters are not supported. Multidevice terminals can include keyboard/display devices with or without paper tape reader/punch or cassette.
- Each terminal can be dedicated or dial-up.
- Nine standard line speeds are supported. These speeds range between 110 and 9600 baud and are defined in appendix C.
- Lines are considered to be full duplex.
- All terminals are interactive devices.
- The TIP supports terminals that use ASCII, External, or correspondence code as their basic code.

[†] See glossary

MAJOR FUNCTIONS

The major functions of the Async TIP are concerned with message control, code and format conversion, and line speed setting.

- The TIP interfaces terminal protocol (one of the seven defined terminals or a terminal derived from one of these seven by varying parameters) to the host interactive virtual terminal (IVT) protocol. Data is transformed to and from IVT format. For downline messages, this text processing is controlled by state programs. For upline messages, this processing is controlled by input state programs.

- The TIP simultaneously controls several transfers to terminals. Each line can have multiple messages waiting for transfer. Information for a transfer is contained in a worklist entry (WLE) which is attached to the line control block (LCB) for that line. The line must have an active terminal control block (TCB) for the terminal.

  If a terminal has a task in progress, additional tasks are queued to the TIP in the form of more WLEs. Tasks are processed on a first-in, first-out basis.

  Most of the terminal transfer functions (such as finding the next character on output, placing it in an output frame, and passing the frame to the output control loop) are performed by the multiplexer subsystem. The TIP specifies the data location on output. On input, the TIP input state programs demultiplex data under multiplex control. The TIP specifies the first of the series of state programs to be used. The TIP gains control to terminate the data transfer or to process the unrecoverable failure of a transfer.

  Fields in the TCB determine which terminal device is to be used for input and for output. These fields are changed by an IVT command from the host application or by a user IVT command entered at the terminal.

- The TIP provides transparent mode for passing terminal data to and from the host. In this mode, the host application program that receives or originates the data is responsible for handling all data interpretation, including control characters.

- The TIP converts terminal code (such as External) to and from ASCII code where necessary.

- The TIP sets line speed explicitly at TCB configuration time, or determines line speed as a part of autorecognition.

- The TIP processes autorecognition information to gather terminal configuration data for the host. This includes line speed for terminals with transmission rates up to 1200 baud. For the 2741 terminal, code type is also detected.

- The TIP is prepared to receive input at all times. The TIP attempts to deliver output whenever such data is available unless an input operation is active, a page wait condition is in force, or an auto input block has been output. When input is detected during output, the TIP suspends the output operation and processes the input. The TIP repeats the interrupted output later, from the beginning of the logical line unless the input causing the interruption was one of the special characters that cause an upline user break or the discarding of a logical line.

- The TIP processes unrecoverable errors in data transfers and reports the failure to the host. Other parts of CCP process terminal and line recovery, in conjunction with the service module.

HOST INTERFACE

The host interface uses block protocol. Data is normally formatted in IVT mode (see section 6). Most commands, status, and statistics pass through the interface in the form of service messages. These use CMD blocks with the connection number zero.

COMMAND BLOCKS

Connection-oriented commands also use four types of CMD blocks. Table 9-1 shows the command block format.

TABLE 9-1. CMD BLOCKS FOR ASYNC TIP

Name	Format						
	Block Header						Other
	DN	SN	CN	BT	PFC	SFC	Other
Start Input	NPU†	Host ID	Line ID	04	C1	05	-
Stop input	NPU†	Host ID	Line ID	04	C1	06	-
Input Stopped	NPU	Host ID	Line ID	04	C1	07	RC
Define terminal characteristics	NPU	Host ID	Line ID	04	C1	04	String

RC - Response code; if 00, stops input response

String - Conforms to the IVT requirements of table 6-5; has the form shown in TERMINAL PARAMETERS less the PFC and SFC; is one or more characters

†Downline from the host only.

The terminal parameters recognized by the IVT interface are as follows:

Command	Definition
TC	Terminal class
PW	Page width
PL	Page length
PA	Check parity
CN	Cancel character
BS	Backspace character
AL	Abort output line
B1	User break 1 character
B2	User break 2 character
CT	Control character
CI	CR idle count
LI	LF idle count
SE	Special edit mode
DL	Transparent text delimiter
IN	Select input device
OP	Select output device
CD	Select character set device
EP	Echoplex mode
MS	Operator-generated message to network operator (NOP) console
PG	Page wait

These commands can be sent at the rate of one per CMD block. There is no limit to the number of CMD blocks that can be sent to alter one or more TCBs. If an error is detected in a command from the host, a BRK block is generated and sent upline.

TERMINAL CONFIGURATION

Before a terminal can be used, the line and terminal must be configured. This is performed by service messages to configure (change) line control blocks (LCBs) and terminal control blocks (TCBs). The initial configuration of TCBs is processed by the service module (SVM). The TIP, however, finishes preparing the TCB on a worklist entry call from the service module.

When the connection between the user terminal and the host is initially established, the terminal is configured by setting up the TCB with a set of default parameters (appendix C). Host software can modify these parameters at any time using any of the parameters listed above. The terminal user also can modify the configuration of the terminal, its operational modes, and the management of the upline and downline data streams by entering these parameters in a control message.

USER INTERFACE

The Async TIP user interface has five aspects:

- Commands from the user console to alter the terminal characteristics. These commands are functionally similar to those commands received from the host which were discussed previously. As in the host interface case, the message is parsed by the IVT processor (PTIVTCMD) and the information is used to alter the TCB for the terminal, thereby altering the terminal's characteristics.

Information changing PW, PL or TC is also passed upline to the communication supervisor (CS) in the host so that network configuration remains a system constant. This assures that terminal will retain its PW and PL characteristics should the NPU fail. In this case the NPU is reloaded from the host using current configuration information.

- The format of input messages from the terminal.
- The format of output messages to the terminal.
- Modem and line control that results from the user activating or deactivating a terminal.
- Sending a break 1 or break 2 signal.

NOTE

Break 1 and break 2 signals are user-defined and are independent of the terminal's break key (if any). The host application program must provide code to utilize these break signals.

USER CONTROL MESSAGES

A user control message has three parts:

```
CTL   other   CR
```

CTL is the appropriate control character for the terminal, other is one of the terminal parameters described previously, and CR (carriage return) is the terminal's input logical line delimiter.

This message is passed through the multiplexer subsystem interface and is recognized as a user-initiated control message. The Async TIP calls PT1VTCMD to parse the message and to check for a valid parameter. If all parameters are valid, the appropriate field in the TCB is changed and the TIP responds to the user with the statement of

```
CR  LF  CR  LF
```

If the user input is incorrect, the TIP responds with the canned message

```
CR  LF  ERR...  CR  LF
```

To enable the TIP to detect operational control messages, each message must start with the defined control character and the message must be contained in one logical input line (2741 terminals must precede the control character with an attention character). Commands become effective immediately. A detailed description of each terminal parameter follows.

Terminal Class Command

The terminal class command format is as follows:

$$\text{CTL}\quad \text{TC} = \begin{bmatrix}1\\2\\4\\5\\6\\7\\8\end{bmatrix}\quad \text{CR}$$

This command establishes the terminal class and default parameters as defined in the terminal class table (appendix C).

Page Width Command

The page width command format is as follows:

```
CTL  PW = NNN  CR
```

This command establishes the line width (in characters) for nontransparent output and the maximum block size (in characters) for input.

For those terminal classes that do not use the display as the default device, the TIP inserts the character sequence deferred for the terminal to move the carriage to the point in the next line where the number of characters to be transmitted equals page width.

For those terminal classes that do use the display as the default device, the page width is assumed to be the actual physical width of the screen. The TIP does not insert a new line sequence when the number of characters output equals page width, since the TIP assumes that terminal hardware automatically starts the new line. This prevents double spacing. NNN ranges between 0 and 255; 0 means NEW LINE is never inserted.

Page Length Command

The page length command format is as follows:

```
CTL  PL = NNN  CR
```

This command establishes the number of physical lines for output. For terminal classes that do not use the display as the default device, the TIP inserts the character sequence defined for the terminal class to advance the carriage to next page when the number of physical lines transmitted equals page length. For terminal classes that use the display as the default device, the TIP assumes the page length is the actual screen size. When the page length is reached, the TIP does not output a new page because the TIP assumes that the terminal hardware will automatically move to the new page position. If the default device is display and if the page wait feature is selected, and if OP = DI, the TIP waits for operator input before continuing. NNN varies between 0 and 255; 0 means no paging.

Check Parity Command

The check parity command format is as follows:

CTL PA = $\begin{bmatrix} \mathrm{Z} \\ \mathrm{O} \\ \mathrm{N} \\ \mathrm{E} \end{bmatrix}$ CR

This command establishes the type of parity that is to be expected on input and that is to be generated on output. Parity options are discussed later in the terminal transforms subsection.

Cancel Character Command

The cancel character command format is as follows:

CTL CN = a CR

This command establishes the character to be used to delete the logical input line in process. After the line is deleted, the TIP sends a *DEL* message to the terminal.

Backspace Character Command

This command establishes the character to be used for the backspace key; that is, the character that causes the previous input character to be deleted from the input buffer in process. Note that backspacing is a one-unit-at-a-time operation. Backspacing cannot cross a logical or physical line boundary. The command format is

CTL BS = a CR

Abort Output Line Command

This command establishes the character to be used to cause the rest of the present output logical line to be discarded. The command format is

CTL AL = a CR

User Break 1 Character Command

This command establishes the character to be used to generate an upline BRK block with a user break 1 reason code. User break 1 is frequently used as an abort output queue signal. The command format is

CTL B1 = a CR

User Break 2 Character Command

This command establishes the character to be used to generate an upline BRK block with a user break 2 reason code. User break 2 is frequently used as an abort job signal. The command format is

CTL B2 = a CR

Control Character Command

This command establishes the character to be used to enter operational control messages (IVT parameter change command). The command format is

CTL CT = a CR

CR Idle Count Command

This command establishes the number of idle characters to be inserted in the output stream following a carriage return. The user of CI = nn for these terminals overrules the default value (appendix C); CI = CA restores the default value. The command format is

CTL CI = $\begin{bmatrix}\text{CA}\\\text{nn}\end{bmatrix}$ CR

LF Idle Count Command

This command establishes the number of idle characters to be inserted in the output stream following a line feed. The use of LI = nn overrules the default value (appendix C); LI = CA restores this default value. The command format is

CTL LI = $\begin{bmatrix}\text{CA}\\\text{nn}\end{bmatrix}$ CR

Default value is given in appendix C.

Special Edit

Command format:

CTL SE = $\begin{bmatrix}\text{Y}\\\text{N}\end{bmatrix}$ CR

An SE = Y selection places the terminal in special edit mode; an SE = N selection returns the terminal to the normal character edit mode. Special edit mode provides two types of special operations:

- Backspace (BS), line feed (LF), and cancel input control symbols are not treated as control characters; instead they are sent upline as data.

- A character delete sequence (one or more backspaces followed by a line feed) causes the TIP to issue a caret prompt to the terminal, and then to continue with input processing.

Transparent Text Delimiter Command

This command establishes the transparent text delimiter. The timeout value is 300 ± 100 milliseconds.

TABLE 9-2. TRANSFORMS FOR EMBEDDED FORMAT EFFECTORS (FE) IN ASYNC TIP DOWNLINE

Action	IVT	Terminal Classes						
		1	2	4†	5	6	7	8
	Virtual Interface	TTY 33, 35, 37, 38	CDC 713-10	IBM 2741	TTY 40	CDC 751-10	Hazeltine 2000	Tektronix 4014
Carriage Return	CR	CR	CR	NL††	CR	CR	not supported	CR
Line Feed	LF	LF	LF	LF	ESCB	LF	LF	LF

†Supports the APL code set.
††New Line

The command format is

CTL DL = (Xhh),(Cnnnn),(TO) CR

hh - Two hexadecimal digits representing the terminal-originated character selected as a delimiter

nnnn - A character count (0 to 4095)

TO - Input character timeout

Each field is optional, but at least one must appear. Parameters can be entered in any order and trailing commas can be deleted.

Select Input Device Command

This command allows the user to specify the input device as a keyboard or paper tape reader. It also specifies whether or not transparent mode is in effect. Note that paper tape input is allowed in keyboard mode, but that the TIP does not send the X-ON characters to start the paper tape reader.

The command format is:

$$\text{CTL IN} = \begin{bmatrix}\text{KB}\\ \text{XK}\\ \text{PT}\\ \text{XP}\\ \text{X}\end{bmatrix} \text{ CR}$$

KB - Keyboard input
XK - Transparent keyboard input
PT - Paper tape reader input
XP - Transparent paper tape reader input
X - Transparent input, any device

Select Output Device Command

This command allows the user to specify the output device as printer, CRT display, or paper tape punch. Printer and CRT display are functionally equivalent except for page wait. The user can punch a paper tape in any mode, but the TIP only provides the X-OFF character if OP = PT and if data is not transparent. The command format is:

$$\text{CTL}\quad \text{OP} = \begin{bmatrix}\text{PR}\\ \text{DI}\\ \text{PT}\end{bmatrix}\quad \text{CR}$$

PR - Printer
DI - CRT Display
PT - Paper Tape Punch

Character Set Detect

Command format:

```
CTL  CD = A  CR
```

This restarts the character set recognition logic when the terminal operator changes the message character set. After the operator enters this command, he has 60 seconds to: (1) physically change the terminal's code set (for instance, by changing the type element on a 2741 typewriter), and (2) activate autorecognition of the new code set by pressing the) and carriage return keys (in that order).

Echoplex Mode Command

This command allows the user to specify where input character echoing is to take place. The command format is:

$$\text{CTL}\quad \text{EP} = \begin{bmatrix}\text{Y}\\ \text{N}\end{bmatrix}\quad \text{CR}$$

Y - TIP sets the communication line adapter to echo the input characters
N - The terminal echoes the input characters

Operator Message Command

This command allows the user to send message text to the network operator. Any number of text characters is accepted. The command format is:

```
CTL  MS = text CR
```

Page Wait Command

This command selects the page wait condition. It allows the user to limit output to the currently displayed page until the operator provides a turn page signal. Note that this command has effect only for OP = DI. The command format is:

$$\text{CTL}\quad \text{PG} = \begin{bmatrix}\text{Y}\\ \text{N}\end{bmatrix}\quad \text{CR}$$

ACCESS CONTROL KEYS

The user can abort output processing by using a special character. Each of the following three allowable special characters must be followed by a carriage return (CR):

- Abort output line character - the predefined key at the terminal (not the ABORT key).
- User break 1 - the predefined key at the terminal (not the BREAK key).
- User break 2 - the predefined key at the terminal (not the BREAK key).

For full-duplex terminals, the special characters can be entered during output; for half-duplex terminals, a break state must first be entered by pressing the BREAK key (IBM 2741 uses ATTN key) to cause output to stop and the special character to be recognized. When break processing occurs, the user can enter data or commands.

TERMINAL ON/OFF AND BREAK CONTROL

For asynchronous lines, the modems produce the carrier signal only during active message transmission.

- Receive Carrier - The receive carrier remains on while the line is up.
- Transmit Carrier - The TIP turns the NPU transmit carrier (RTS) on for the duration of an output message delivery to the terminal. The TIP turns RTS off immediately following the last character sent or in response to a break received from the terminal.

Breaks can be initiated upline. The received (upline) break from the terminal appears in one of two ways:

- For terminals with transmission rates less than 600 baud: for at least 200 ms, a spacing condition is maintained on the receive data line while the output is being sent.
- For terminals with transmission rates of 600 baud and above: for at least 200 ms, a spacing condition is maintained on the supervisory receive channel.

USER INPUT MESSAGE FORMAT

Two standard input message formats are acceptable, one for normally processed data and the other for transparent data, as shown below:

```
        [(STX)  logical line  LLDLM  (X-OFF)]
input = [(STX)  CTL  Command  LLDLM         ]
        [       transparent data  DLM       ]
```

An X-OFF after a DLM is not seen.

STX - Start of text symbol
logical line - [physical line LF CR] physical line 0-n

The terminal user enters input as the basic unit that he wishes processed by the computer. If page mode is in effect, input can be treated as a request for next page.

Character mode inputs are logical lines as shown above.

The logical line delimiters (LLDLM) are:

LLDLM = $\begin{bmatrix}\text{CHARSEQ}\\ \text{EOT}\end{bmatrix}$

CHARSEQ - $[\text{LF} \quad \text{DEL}]$ CR 0-m

EOT - 04_{16}, the value when translated from user's code. Set to no parity (ASCII).

- A logical NOT

LF - $0A_{16}$
CR - $0D_{16}$
$\Big\}$ values - when translated from the user's code. Set to no parity (ASCII).

CTL - Control character, defined by terminal type and can be changed by user

Command - Terminal parameter commands (listed above in user control messages)

Transparent Data - $[[\text{byte}]\ 1\text{-}n_1\ (\text{X-OFF})]\ 1\text{-}n_2$ where n_1 and n_2 are positive integers

DLM - $\begin{bmatrix}\text{200-ms timeout character count}\\ \text{delimiter byte}\end{bmatrix}$

Any of these can be specified by the user. Two or more can be used in combination.

physical line [Character] 1-m where m is terminal's physical line width as defined by user

byte - bit pattern - any bit pattern that can be received from the terminal

character - member of 128 ASCII character set. When translated from user's code, it is set to no parity (ASCII)

DEL - idle fill

X-OFF - a character that turns the paper tape reader off. Meaningful only when input device is paper tape (IN = PT or XP)

USER OUTPUT MESSAGE FORMAT

Two standard output message formats are acceptable: one for normally processed data and the other for transparent data. The format is given for the message after all IVT transforms, paging, etc., have been performed.

```
output  = [page            ]
          [transparent data]

page = (FF)  [(PREFE)  physical line]

[LF   [idle]    NR   [idle]        ]]
            0-k            0-m     ]]
                                   ]]
[(POSTFE)   (X-OFF   [idle] 3)     ]]    1-n
```

k and m - Line feed and carriage return idle counts defined by terminal class or terminal parameter commands; n - Page length in physical lines. If the page length is set to zero, no form feeds (FF) are inserted by the TIP, and the page wait feature has no effect.

transparent data	- [byte]$_{1-n}$ where n is an installation time parameter for maximum block size
FF	- [home-and-clear] ; differs by terminal class; not sent if page length is zero
PREFE	- [Single Space Double Space Triple Space Start of Current Line Home Home-and-Clear] Pre-print format effectors; differ by terminal class. See table 9-2.
physical line	- [character]$_{0-n}$ where n is defined by page width
LF	- 0A$_{16}$ value when translated from no parity ASCII to user's code set; causes the cursor or platten to move down one line
idle	- [DEL NULL]
POSTE	- [Single Space Start of Current Line] Postprint format effectors; differ by terminal class. See table 9-3.
X-OFF	- A character that turns the paper tape reader off; used only when the output device is paper tape (OP = PT) and when data is not transparent.
byte	- Any bit pattern capable of being received by the terminal; depending on the parity option selected, byte can be 7 bits plus parity or all 8 bits as received from the host.

TABLE 9-3. PREPRINT AND POSTPRINT FORMAT EFFECTORS FOR ASYNC TIP

	IVT FE	TERMINAL FE						
		TTY 33, 35, 37, 38	CDC 713-10	IBM 2741	TTY 40	CDC 751-1	Hazeltine 2000	Tektronix 4014
PREPRINT								
Position to start of next line	SPACE	CR LF	CR LF	NL	CR LF	CR LF	LF	CR LF
Position to start of current line	+	CR	CR	(N)BSs†	ESC G	CR	-	CR
Position to top of form (cursor home)	*	CR 6LFs††	EM	6NLs††	ESC H	EM	2DC	ESC FF
Home cursor and clear screen	1	CR 6LFs††	CAN	6NLs††	ESC R	CAN	FS	ESC FF
Null	,	-	-	-	-	-	-	-
Double Space	0	CR 2LFs	CR 2LF	2NLs	CR 2LF	CR 2LF	2LFs	CR 2LFs
Other	-	CR 3LFs	CR 3LF	3NLs	CR 3LF	CR 3LF	3LFs	CR 3LFs
POSTPRINT								
Single space	.	CR LF	CR LF	NL	CR LF	CR LF	LF	CR LF
Return to start of current line	/	CR	CR	(N)BSs	CR	CR	-	CR

†The number of backspaces is a function of current cursor position.

††When PL ≠ 0, the IVT logic calculates the difference between end of page and current print position. It then spaces forward the appropriate number of lines.

DATA TRANSFORMS

The following text describes the upline and downline transforms necessary to convert asynchronous vertical terminal data to and from terminal protocol format. The following transforms are described:

- Parity options
- Character mode input processing
  (1) for logical and physical lines
  (2) block mode support as the default condition
  (3) type ahead mode
  (4) keyboard input (includes processing for parity, for nulls and deletes, conversion to 7-bit ASCII code, backspacing, autoinput, line feed and new line for physical lines, carriage return and end of transmission for logical lines, store text, cancel, upper and lower case control, and line width)
  (5) paper tape character mode input
- Transparent mode input
- Character mode output processing
  - printer output (including conversion from 7-bit ASCII to printer code, processing of format effectors, line folding, and upper and lower case shift)
  - CRT output
  - paper tape output
- Transparent mode output processing
- Aborting logical lines

PARITY OPTIONS

Parity can be set in any of four ways: zero (Z), odd (O), even (E), and none (N). Four processing types (transparent and nontransparent data for input and output) must be supported. Table 9-4 summarizes the processing done on bit 7 (parity bit) of the character by the Async TIP.

CHARACTER MODE INPUT PROCESSING

Logical Lines

A logical line of input is defined to be that input line ending with the terminal's carriage return, new line, or EOT delimiter. The TIP discards the carriage return or EOT character. A line feed sequence or new line sequence, respectively, is returned to the user if the mode permits. The currently assembled block is sent to the host as a MSG block. Null logical lines are discarded only if they are used as a page turn indicator.

TABLE 9-4. PARITY HANDLING

		Parity Selection			
Data Mode	Direction	Zero (Z)	Odd (O)	Even (E)	None (N)
Nontransparent	Output	Host sends 8 bits; bit 8 is ignored. If character is translated, bit 8 is suppressed. (Virtual characters must have bit 7 = 0.)			
Transparent	Output	Host sends 8 bits; bit 8 can be anything. Bit is then set correctly to Zero parity, then character is sent to CLA	Odd	Even	Character from host is sent out unaltered.
Nontransparent	Input	Bit 8 is always set to zero before sending character to host.			
Transparent	Input	Bit 8 is always set to zero before sending character to host.			Character is sent to host unaltered.

Physical Lines

A physical line of input is defined to be an input line that ends with the terminal's line feed delimiter or when current page width is reached.

When not in APL special mode, the TIP discards the line feed delimiter character. In the case of line feed, a carriage return sequence is returned to the user. The currently assembled block is sent to the host as a BLK block containing a single physical line. When in APL special mode, the line feed is not discarded, a carriage return sequence is not sent to the user, and the block is sent to the host as a MSG block.

Note that on a 2741 terminal, line feed is effected by using the ATTN key. In normal processing, a new line is echoed to position the carriage to the beginning of the next line and the keyboard is unlocked. In APL special mode, a line feed is echoed to perform the physical line feed only; the keyboard is not unlocked.

Block Mode Support

The default condition of the TIP is block mode. This means that input has priority over output. At the end of each logical or physical line, a 300-ms timer is started by the TIP. If any new input arrives from the terminal, the output side of the TIP is locked out. Output data from the host remains queued for the terminal. Any canned response, such as echoing carriage return to line feed sequence, is discarded.

Type Ahead Mode

The TIP is always in the type ahead mode; that is, it is normally ready for input unless it is busy outputting. Output is started only if the input pauses at the end of a logical line for 300±100 ms. If an input request conflicts with output on output operation or if input starts at any time that output is active, the output is halted and input proceeds. If the user is in autoinput or special edit mode, he has the responsibility for not typing ahead.

Keyboard Input

PARITY CHECKING AND STRIPPING

The TIP services the input data stream using the default parameter appropriate to the terminal class. For the no parity checking case, the parity bit is stripped, as data characters arrive from the terminal. (This does not apply if the data is transparent and PA = N.) The user can cause parity checking by resetting the internal parameters using the CTL PA command. The communications line adapter is set to the terminal's present parity mode. As input characters arrive, the communications line adapter automatically checks and strips parity from the data characters. If a parity error occurs, the TIP stores the bad character in the input data buffer and then marks the data block clarifier (DBC) to show that a parity error exists within the data block.

NULLS AND DELETES PROCESSING

The TIP strips nulls (NUL) and deletes (DEL) from the input data stream as it receives them.

CHARACTER CODE CONVERSION

The TIP converts the terminal's input characters to 7-bit ASCII (parity bit = 0) as it receives them.

BACKSPACE PROCESSING

The TIP is capable of detecting the terminal's currently defined backspace character. One input character is discarded by the TIP for each consecutive backspace character received. Backspacing to the beginning of a line deletes the line. Backspacing past the beginning of the line is ignored.

Since the TIP may ship physical lines to the host before the end of logical line, all references to beginning of line in the preceding discussion should be understood to refer to physical lines. If the current page width is reached before receiving the end of a physical line indicator, backspacing is not permitted into the previous block since the TIP has already released that block. Backspacing is effective only if the special edit mode is not in effect. In special edit mode, the backspace is treated as any other data character.

AUTOINPUT PROCESSING

The TIP has limited ability to place data into the data block just output (autoinput mode). Logically, the previously received output data block is chained to the front of a newly arriving input data block and is sent to the host as part of the input data stream. Autoinput only applies to downline MSG blocks; it is ignored if specified in a BLK block (that is, the entire autoinput message is restricted to a single block).

After the autoinput block has been output, the TIP cannot deliver any more output until the executed input has been received. Otherwise, the input from the terminal may not be attached to the correct block. Only the first 20 characters of the output data are returned. Format effectors are stripped from the output data before returning it. If the user wishes to override the autoinput and substitute his own input, he enters a cancel input line character followed by a carriage return/EOT. This cancels any data entered by the user as well as the autoinput block being held for return to the host. When an autoinput block has been output, the TIP remains in input mode until a noncancelled input is received.

SPECIAL EDIT MODE

In Special Edit Mode input, the backspace, line feed, and cancel characters are sent upline as data. When a character delete sequence is recognized (BS ... BS LF), the TIP issues a caret prompt. Note that in special edit mode the TIP recognizes only logical lines and not physical lines.

LINE FEED AND NEW LINE PROCESSING (PHYSICAL LINE)

When not in Special Edit Mode, the TIP discards the line feed or attention (2741 terminal) character, and sends a carriage return sequence to the user. The currently assembled block is sent to the host as a BLK block containing a single physical line. In special edit mode, however, the TIP stores the line feed as data and does not send a carriage return sequence to the user.

CARRIAGE RETURN AND EOT PROCESSING (LOGICAL LINE)

The TIP discards the carriage return or EOT character. Either a line feed sequence or new line sequence is sent to the user if the mode permits. The currently assembled block is sent to the host as a MSG block. A null logical line is discarded only if it is used as a page turn.

PHYSICAL/LOGICAL LINE PROCESSING

Processing of physical and logical lines follows the general rules laid down for character mode input processing.

START-OF-TEST PROCESSING

The start-of-text (STX) character is discarded when it occurs as the first character of a logical line.

CANCEL CHARACTER (CN) PROCESSING

The TIP detects the terminal's currently defined cancel character preceding the end-of-logical line indicator, discards the current input logical line, and sends a *DEL* message to the terminal. (Note that 2741 terminals must have an attention character preceding the CN character.) If any part of the logical line has already been dispatched, a cancel MSG block is sent to the host. The cancel character is treated as any other data character if the TIP in operation is in either special edit or transparent mode.

UPPER/LOWERCASE SHIFT PROCESSING

For the 2741 terminal, the TIP records shifts between lowercase and uppercase to ensure correct translation to ASCII. The TIP assumes the lowercase condition at the beginning of each input logical line.

MAXIMUM LINE WIDTH PROCESSING

If the current line width is reached without a physical line terminator being found, the partially assembled physical line is sent to the host as a BLK block. In the case that the line width is zero (user did not specify line width), the maximum line width is set to 140 characters. Note that in the usual case, the line terminator is found before the maximum width is reached. At that time the line is sent to the host as a BLK block.

Paper Tape Character Mode Input

The TIP is capable of reading paper tape input data without forcing the user to specifically enter a paper tape mode. To accomplish this, X-OFF characters should not exist on the paper tape or, alternatively, the user must turn the reader on after each X-OFF.

For those users who have paper tape with X-OFF characters on the paper tape, paper tape input should be declared. In both keyboard and paper tape modes, the TIP detects end of physical/logical lines and processes them accordingly. The TIP then checks the next character which arrives; nulls and deletes are always stripped. If the character following a carriage return or EOT delimiter is a line feed or a new line, that character is discarded by the TIP. Similarly, if the TIP detects a line feed or new line delimiter followed by a carriage return or EOT character, that character is discarded.

In keyboard mode an X-OFF character is treated as data. In paper tape mode X-OFF is treated as data unless it is at the end of a logical line. In that case, X-OFF is discarded. If X-OFF stopped the tape, whether or not it was at the end of the logical line, the TIP sends X-ON after a MSG block from the application has been processed and there is no further output queued for this terminal.

TRANSPARENT MODE INPUT PROCESSING FOR KEYBOARD AND PAPER TAPE

Input data received by the TIP is sent to the host without character translation. When system default block size is exceeded, data is sent as BLK type blocks until one of the user transparent delimiters is reached. That data is then sent as a MSG block. In transparent paper tape mode where a special character or character count is specified, receipt of an X-OFF, which stops the tape, results in X-ON being sent to the terminal. The X-OFF character and all previously input data is sent to the host in a BLK block. When X-OFF is input due to special character or due to a timeout delimiter, and the tape stops, then the previous input is sent to the host in a MSG block, and transparent mode is terminated. No X-ON is sent in this case. If the input does not stop at the end of the transparent input, the remaining data is processed in character mode. Some of the initial character data might be lost in this case. The number of significant bits per character received from the terminal can range from six to eight depending on terminal type and parity setting. When the no-parity mode is selected (PA = N), the parity bit is passed as data. All information is passed right justified in the byte. Nonsignificant high-order bits are set to zero.

When transparent mode ends, the TIP returns to character mode. Device type remains unchanged.

CHARACTER MODE OUTPUT PROCESSING

Output delivered to the TIP can have multiple logical lines within a data block. End of logical line delimiters, as well as certain embedded format characters, are translated to the terminal's format sequence where possible. Table 9-2 lists embedded format effector conversions for various types of terminals. The TIP monitors for input message or break commands during output operations so that the user can stop the output and perform necessary input operations or terminate the output.

During automatic line folding or end of logical line processing, the TIP inserts the terminal's currently defined number of NUL characters into the output stream. During output paging, any input causes the TIP to reset the page count to the top of page. Therefore, the user must assume responsibility for inputting data, which can cause subsequent output to be improperly positioned on following pages.

Where format effectors cause the terminal to be positioned over a page boundary, a new page sequence is output. This feature can be disabled by setting the page length to zero.

Logical Line Aborting

During output, the TIP continuously monitors for a break or for input data. The user can terminate the current logical output line by entering the abort line character followed by a carriage return or EOT. Output continues with the next logical output line.

Printer Output

The printer output function includes character translation, format effector and line folding, and, for the 2741 terminal, upper and lowercase shifts.

- Character translation. Normal output data (IVT format) is delivered to the TIP from the host application in ASCII code. The TIP converts the ASCII data to the terminal character set.
- Format Effectors and Line Folding. Each logical line of output can contain a format effector as the first character. A bit in the data block clarifier defines whether or not these format effectors are present. Preprint single spacing is assumed if the format effectors are not present or are not defined. The format effectors (table 9-2) cause preformat or postformat control. The TIP converts the format effectors to the terminal's format sequence. Where applicable, the TIP automatically folds the line by outputting the terminal's line feed and carriage return sequence with the appropriate number of NULs.
- 2741 Upper and Lowercase Shifts. Current upper and lowercase shift is retained by the TIP for output. Upper and lowercase shift characters are inserted by the TIP as a function of ASCII code translation to the 2741 terminal character set.

CRT Output

CRT output is processed the same as printer output except that the TIP allows a page wait when that option is selected. After a page wait, the user enters a null line to obtain the next page. The TIP discards the null input line in page wait situations. If a non-null input line is typed by the user, it is treated as a page turn and is passed to the host unless it is a command. In that case, the TIP processes the command.

When the page wait option is selected, the page output size is one line less than the current page length, to allow space for the user input necessary to turn the page. The page wait option has no effect on hard copy devices or when the current page length is zero. If a top of form is received in the output stream before the page is full, the message OVER.. is output to notify the user to turn the page.

Paper Tape Output

When the output device is specified to be paper tape, the TIP inserts an X-OFF character (DC4) followed by three NULs at the end of each logical line sequence if that line sequence contains postprint format effectors. Line folding is performed as for printer output.

TRANSPARENT MODE OUTPUT PROCESSING FOR PRINTER, CRT, AND PAPER TAPE

Transparent mode allows the user application to inhibit the TIP transforms. In this mode the user application is responsible for all data formatting.

The application can permit page waits by sending a synchronous command to the TIP. The TIP adds page waits at the end of every MSG block. The TIP interrupts page wait responses in the same manner as character mode page waits.

LOGICAL LINE ABORTING

Logical line aborting is the same as described previously for character mode output processing.

ERROR HANDLING

The Async TIP has the following error handling capability:

- Marks lines down if the transmission fails due to an autorecognition timeout or a hardware error on the line (detected by multiplex subsystem). The TIP then requests the service module to generate a line status disabled service message. SVM sends the message to CS in the host.
- Disables the line in response to a disable line service message from CS in the host.
- Stops message processing and releases the message in response to user breaks, aborts output line commands, or cancels input line commands.
- Rejects improper commands.

The TIP does not generally check output transmissions.

REGULATION

The NPU is forced to reject input when (1) the NPU runs low on buffers, (2) the network block limit is exceeded, (3) a stop input command is received, or (4) the NPU loses contact with the host. If the reason for rejecting input is because the NPU lost contact with the host, then at the time the condition is detected in the NPU, each connected console terminal is sent a canned message to inform the user of the situation. The canned message is

X-OFF NUL NUL CR LF BELL BELL IDLES_N

INPUT STOPPED user text CR LF IDLES_N

Default for user text is HOST UNAVAILABLE. If input is received after the user has been notified of a loss of contact with the host or if any of the other reasons for rejecting input are detected, the input is discarded and the user is notified with the following canned message:

X-OFF NUL NUL CR LF BELL BELL IDLES_N

REPEAT...CR LF IDLES_N

This message is repeated every time any further input is attempted from the terminal until the situation is relieved. When communication with the host has been restored, the user is notified by the following canned message:

CR LF IDLES_N

HOST AVAILABLE CR LF IDLES_N

AUTORECOGNITION

Autorecognition allows the TIP to determine both the terminal's transmission rate (if the rate is between 110 and 1200 baud) and the terminal's current code set. To activate the autorecognition function, the user at the terminal presses the carriage return key after the connection is established. This generates the appropriate character code input from the terminal. The TIP samples the input at 800 baud. Depending on the transmission speed, the TIP will detect one or more different characters for each acceptable line speed.

The TIP resets the communications line adapter to the correct baud rate and then sends the terminal two line feeds to begin the character set recognition function. The operator presses the) key and then a carriage return (ASCII terminal operators may press only the carriage return if they wish).

To determine the code set, the TIP compares the input bits to the bits for these characters in each acceptable code set. After finding the correct code set, the TIP sends two more line feeds downline to the terminal to indicate that autorecognition is complete. Upline, the TIP sends a line operational service message to the host. This message contains the line speed and terminal character set. See appendix C.

Extended code set recognition is a build time option. If the option is not selected, the TIP sends an error message to the terminal.

Any terminal operating at a speed greater than 1200 baud must be dialed into a port where the communications line adapter is designed to operate at that particular speed.

Table 9-5 summarizes the baud rate and code set autorecognition.

TABLE 9-5. AUTORECOGNITION IN THE ASYNC TIP

Stage 1 - Baud Rate - Autorecognition after terminal connection

Rate	Terminal	Operator Input	TIP Response
110	Any but 2741	Carriage Return	2 LFs
134.5	2741	Carriage Return	2 LFs
150	Any but 2741	Carriage Return	2 LFs
300	Any but 2741	Carriage Return	2 LFs
600	Any but 2741	Carriage Return	2 LFs
1200	Any but 2741	Carriage Return	2 LFs

Stage 2 - Code Recognition

Code	Operator Input	TIP Response*
ASCII	) CR or CR	2 LFs
Teletype-paired ASCII	) CR or CR	2 LFs
Bit-paired ASCII	) CR or CR	2 LFs
External BCD	) CR	2 LFs
External BCD-APL	) CR	2 LFs
Correspondence	) CR	2 LFs
Correspondence APL	) CR	2 LFs

*If extended character set recognition is not included at build time, the error message is sent to terminal.

MODE 4 TIP

10

The Mode 4 terminal interface program (TIP) provides procedures to convert data from synchronous terminals using Mode 4 protocol to data that is compatible with the host's virtual terminal (IVT or BVT) format. The Mode 4 protocol supports both batch and interactive devices. There are three versions of the protocol:

- Mode 4A supports a group of devices, such as console, printer, and card reader.
- Mode 4B supports a console.
- Mode 4C supports several consoles.

The TIP also handles the necessary interface control tasks.

HARDWARE CONSIDERATIONS

Some of the hardware considerations for Mode 4 are the following:

- Terminal types. A typical Mode 4A terminal is the 200 User Terminal consisting of a keyboard, a display (CRT), a card reader, and a printer. This terminal has both interactive and batch devices, and uses a single line.
- Cluster capabilities. The Mode 4 terminal can be a cluster of several devices of the same types, such as a group of consoles or a group of printers. The TIP services multiple terminals in sequential order, without priority. However, the individual batch devices (card reader and printer) in a Mode 4A cluster terminal are subordinated to the interactive device. A batch transfer using such a device is preempted by an interactive device transfer.
- Line speed. The TIP supports line speeds up to 9600 baud.
- Line type. Lines are of two types: dedicated without a transceiver, or dial-up with a modem. Lines are considered to be half duplex. The TIP either transmits data over the line or receives data, but does not do both simultaneously.
- Terminal codes. The TIP supports terminals that use either ASCII or external BCD code.

MAJOR FUNCTIONS

The TIP performs the following major functions:

- It interfaces terminal protocol (some variation of Mode 4 protocol) to the host virtual terminal protocol (IVT for interactive devices, BVT for batch devices).

- It provides a transparent mode of passing terminal data to and from the host. In transparent mode, the host application program that receives or originates the data is responsible for handling all data interpretation, including control characters.
- It converts external BCD code to and from ASCII code where necessary.
- It polls terminals to receive upline data or to assure that the terminal is ready to accept downline data. The host requests the polling; the TIP controls actual timing of the polling.
- It performs autorecognition to gather terminal configuration data for the host. Autorecognition on lines with multicluster terminals report only one cluster.
- It performs terminal and line recovery for recoverable errors and reports irrecoverable errors.

NOTE

Considerable differences in terminology exist in Mode 4 documents. Table 10-1 defines the terms used in this manual and in other Mode 4A and 4C documents.

TABLE 10-1. MODE 4 NOMENCLATURE

Nomenclature used in this manual	Mode 4 Nomenclature	Mode 4C Nomenclature
NPU	data source	control station
cluster address	site address	terminal address
cluster controller	equipment controller	station
terminal address	station address	device address

DATA FORMAT FOR MODE 4

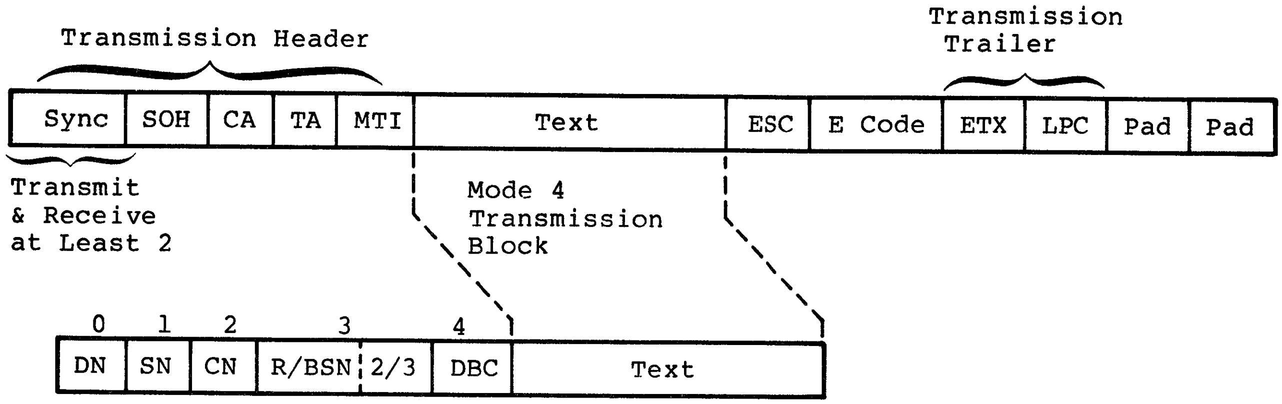
Figure 10-1 shows typical data formats for Mode 4 protocol.

HOST INTERFACE

The host interface uses block protocol. Data is formatted in IVT or BVT mode (see section 6). Most status and statistics pass through the interface in service messages. These use CMD blocks with a connection number (CN) of zero.

Four types of line-related CMD blocks are used. Table 10-2 shows the command block format.

DATA FORMAT (odd parity)



Block Format
(NPU and Host)

NONDATA FORMAT

Mode 4 Transmission Block

Sync - Sync bit = 16_{16}
SOH - Start of header = 01
ESC - Escape code; external BCD = $3E_{16}$ ASCII = $1B_{16}$
ETX - End of text = 03
MTI - Message text indicator
E Code - Equipment Code
CA - Cluster address (appendix C)
TA - Terminal address (appendix C)
LPC - Longitudinal parity check; collects parity on bits 0 - 6 of all characters except Sync bytes
DN, SN, CN - Block header address
R/BSN/BT - Response flag/block serial number/block type; BT for a data block must be 1 of 2
DBC - Data block clarifier
Pad - Byte of all 1's to assure transmission of LPC by modem

Figure 10-1. MODE 4 Protocol Message Formats

TABLE 10-2. CMD BLOCKS FOR MODE 4 PROTOCOL

Name	Format						
	DN	SN	CN	BT	PFC	SFC	Other
Start Input	NPU Terminal	Host Data Node	CN	04	C1	05	-
Stop Input	NPU Terminal	Host Data Node	CN	04	C1	06	-
Input Stopped	Host Data Node	NPU Terminal	CN	04	C1	07	RC
Define Terminal Characteristics	NPU	Host Data	CN	04	C1	04	String

RC - Response code

00 - Stop input response
01 - Input device not ready
02 - Card slip error
03 - EOI input
04 - Batch input interrupted by interactive I/O

String - Conforms to the IVT requirements of table 6-5. String has the form shown in TERMINAL PARAMETERS less the PFC and SFC. It is one to fifty characters long.

The TERMINAL PARAMETERS recognized by the IVT interface are listed below. See appendix C for default values.

Command	Definition
TC	Terminal class
PW	Page width
PL	Page length
CN	Cancel character
CT	Control character
IN	Input device for transparent mode
PG	Page wait
B1	User break 1
B2	User break 2
MS	Operator message

Each command entered from the terminal must be preceded by the control character and followed by a carriage return or an end of message: CTL parameter CR. In an input block from the terminal containing multiple logical lines separated by carriage returns, only the first logical line can be an IVT command. All other lines are treated as data and sent to the host. If the IVT command is a request for transparent input, the current terminal input continues to be treated in the current mode. The next input block, however, is treated in transparent mode.

Commands sent by the host are contained within CMD blocks and are not preceded by the control character. Only one IVT command can be sent in a CMD block.

If an error is detected in a command from the host, a BRK block is returned to the host. When errors are detected in a request from the terminal, the message ERR... is sent to the terminal.

TERMINAL CONFIGURATION

Before a terminal can be used, the line and terminal must be configured. This is performed by service messages to create control tables called line control blocks (LCBs) and terminal control blocks (TCBs). Configuring the LCBs can involve the autorecognition logic.

Most of the initial configuration of TCBs is processed by the service module. The TIP, however, finishes preparing the TCB when it is called by the service module.

The TIP processes each line as independent data channels. Each terminal on a line is checked for work in the order the terminals were configured. This method allows each terminal to be processed in order without priority. The card reader and printer of the 200 User Terminal are treated as separate terminals in this scheme, but the console is required and must be configured before the card reader and printer can be configured.

Note that each terminal can perform only one task if other terminals have work waiting. The work allocation check always moves to the next terminal after assigning the current task to a terminal.

IVT INTERFACE

The interactive virtual terminal interface to the Mode 4 TIP supports display/keyboards attached to synchronous lines. The configuration may be multicluster and each cluster may be multiterminal. The 200 User Terminal console supported by the IVT interface uses several additional features to control the card reader and printer.

The terminals are activated a (polling for input is started) either by delivery of an output message (MSG or BLK blocks) or by a start input command.

Polling for input continues until the terminal is deleted, until an error occurs, until buffer or logical link regulation occurs, or until a stop input command is received. Input stopped command is sent in response to the stop input command.

A STP block is sent upline whenever a communications error is detected. The subsequent STRT block is sent upline when the error condition is resolved.

For the 200 User Terminal, the use of the display causes STP blocks to be sent upline on the card reader and printer connections. The STP block on the card reader connection is preceded by an input stopped command if the device is reading cards. These events signal the current use of the 200 User Terminal transmission buffer since this buffer is shared by the display, the card reader, and the printer. The host is sent STRT blocks for the card reader and printer to signal the end of the interactive transactions when the TIP receives a stop input command for the console connection.

CARD READER INTERFACE

The Mode 4A card reader is activated by sending a start input command to the TIP. The TIP sends card reader data, transformed to BVT format, to the host. Each block of data is reported to the host as a BLK block until an EOI card (6/7/6/9 punch in column one) is detected. Then a MSG block is sent containing the data up to and including the EOI card. Subsequent EOI cards are discarded until the first non-EOI card is sensed. Any data following the last EOI is considered part of the next message. (A single block from a Mode 4 device might contain more than one message, which is reported as a MSG block.) An input stopped command is sent following the last data from the transmission block. No further input is allowed from the card reader until it is restarted by a start input command. An input stopped command is also sent if no further cards are present in the input hopper (not ready), if the TIP detects an error in the card data (card slip), if card reading is interrupted for I/O to the display/keyboard, or as a response to a stop input command. A reason code is supplied to distinguish the different cases (see table 10-2). Note that if an EOI card and not ready are detected in the same transmission block, then the not ready reason code is reported.

An upline STP block on the card reader channel indicates that downline data or commands must not be sent. If data or commands were sent, they are not acknowledged with a BACK block. The data or command must then be repeated. A STP block is used by the host or the terminal operator whenever the display is in use.

An upline STP block is generated when the TIP detects a communications error with the terminal. A subsequent upline STRT block is sent when the error is resolved.

PRINTER INTERFACE

The printer is activated when the host sends downline data. The printer connection is considered active until a MSG block is sent by the host or until the display is used. The TIP sends to the printer the data that has been transformed from BVT format to printer format. Each correctly delivered block is acknowledged by sending a BACK block to the host.

A STP is generated by the TIP whenever data or commands cannot be processed because the display is in use. This stop occurs either when the host sends data to the display or when the remote operator interrupts a batch operation. The host must prepare to resend any data or commands not acknowledged with a BACK block.

A STP block is also generated whenever an irrecoverable error is detected on the printer.

A BRK block is sent to the host whenever the printer is found to be not ready while the host is attempting to deliver output.

DATA TRANSFORMS

This subsection describes the upline and downline transforms that convert data to and from terminal format.

DOWNLINE IVT TRANSFORMS

The downline IVT transforms (table 10-3) apply to the following:

- Carriage return (CR)
- Line feed (LF)
- Logical line separator (US)
- Autoinput. In this case, the TIP saves the first 20 data characters of the output message and returns these 20 characters together with the reply data solicited from the operator at the terminal (also has upline transform effects). The format effector is removed if present in the downline data.
- Transparent data. Data is not transformed; it remains in terminal format.
- Format effectors (FEs). These are present in the downline IVT data (see table 10-4).

NOTE

Flags affecting the autoinput, transparent mode, and FEs are found in the data block clarifier (DBC) field of the data block. The transparent mode flag applies to both upline and downline transfers. This clarifier byte immediately precedes the first byte of data in the block.

Cursor is returned to the left margin following each input and output of a logical line. If more than one logical line exists in a block, the logical separator (US) is treated as a carriage return. This assures that output data is compatible whether logical lines are blocked or not. The fact that the cursor is returned to the left margin after each output is taken into account when processing the format effectors. Any undefined format effector is processed as a preprint position-to-start-of-next-line command.

- ASCII control characters. Any ASCII control character is replaced with blanks. For those terminals with fewer than 96 characters, lowercase is treated as uppercase.
- Data errors. If an error is detected in the IVT data, a BRK block is sent to the host.
- Code conversion. Converts ASCII to terminal code if necessary.
- Preprint format effectors for clearing and homing cursor. If preprint format effectors position-to-top-of-form or home-cursor-and-clear-screen are used, they must begin a transmission to the terminal. If more than one logical line exists in a block from the host, the block is fragmented into as many separate transmissions as necessary to achieve the proper function.

TABLE 10-3. DOWNLINE IVT TRANSFORMS FOR MODE 4

IVT Interface	Transform (all devices)	
Carriage Return	CR	CR†
Line Feed	LF	nul
Logical Line Separator	US	CR†

†ASCII is $1B41_{16}$; external BCD is $3E41_{16}$; see appendix A.

TABLE 10-4. DOWNLINE IVT FORMAT EFFECTOR (FE TRANSFORMS)

FE Type	Command	Effector Code	Transform (all devices)
Preprint	Position to start of next line	SPACE	nul
	Position to start of current line	+	nul
	Position to top of form (cursor home)	X	C††
	Home cursor and clear screen	1	12††
	Null	,	nul
	Double space	0	CR†
	Triple space	-	CR,CR†
Postprint	Single space	.	nul
	Return to start of current line	/	nul

Notes:

†CR in ASCII is $1B41_{16}$; in external BCD it is $3E41_{16}$.

††Message type indicator (MTI) codes of Mode 4 protocol where C_{16} is reset write and 12_{16} is clear write.

UPLINE IVT TRANSFORMS

The input from the terminal can include multiple logical lines separated by carriage returns with the restriction that only the first logical line can be an IVT command. Each logical line is sent to the host as an individual MSG block. Code conversion and control character blanking can occur. No other transforms are performed on the data except that escape codes are not counted in the calculation of the cursor position.

Autoinput Mode

The TIP delivers output to the terminal and retains the data buffers when the autoinput flag is set in the data block clarifier of a MSG block. The subsequent input from the terminal is attached to the end of the first 20 characters of the saved data and returned to the host. The format effector is deleted from the autoinput if it is present. If more than one logical line is present in a MSG block specifying autoinput, a BRK block is sent to the host. If more than one logical line is received from the terminal, the first received line is appended to the saved autoinput. All subsequent logical lines are transmitted to the host as received.

The terminal operator can cancel the saved autoinput data by entering a logical line ending with the cancel control (CN) character. The cancel request must be the first logical line of the transmission; subsequent logical lines are sent to the host as received.

An input logical line other than an IVT command must be received to satisfy the autoinput request before a subsequent output can be sent to the terminal. The cancelled line is not sufficient to satisfy the autoinput request.

Transparent Mode

Mode 4C terminals are interactive; no batch capability is provided. The IVT transform is not performed on transparent data. Mode 4 frame control is added to the data. No code conversion is performed. The parity bit for each character is also added before the data reaches the line.

Autoinput and page wait are supported for transparent data. However, page length calculations are not supported; page wait occurs only following each MSG block.

Format effectors are not supported. Each output is assumed to be a write with an E4 terminator. The clear-write and reset-write features of the Mode 4 protocol are not supported.

Transparent input applies only to the first input transaction following selection of the feature. The Mode 4 frame control characters are removed but no other translation occurs. The cursor is not repositioned to the left margin following each input or output and the keyboard is not unlocked. Since any further polling would result in retransmission of the previous data, polling ceases. The host must request that polling be resumed by sending output or by issuing a start input command.

Transparent mode for a Mode 4A terminal, which is a batch device, is illegal; a BRK block is sent if this is attempted.

User Break 1/Break 2

The IVT interface allows the terminal operator to request a BRK block to signal the user break 1 or break 2 condition. This BRK block is caused by entering a logical line with either the user break 1 or user break 2 character as the only data. The interpretation of these BRK blocks depends on the application program that uses them.

Page Wait

The page wait feature of the IVT interface provides a method of assuring that output is delivered at a readable rate. The data sent from the host is added to the screen until the end of the page is reached. The data remains displayed until the operator enters an input line.

Page Size

Calculations for page size are based on the page width and page length parameters, which are assumed to be the actual size of the terminal or display. It is assumed that the hardware provides an automatic carriage return at the page width boundary.

Page calculations take line folding into account. A folded logical line may span a page boundary. The clear-write and reset-write format effectors terminate a previous page. If the previous page is not full, the message OVER.. is sent to the terminal. A page is full whenever the page length less one line is filled.

Page turning is accomplished when the terminal operator enters an input line. If the page prompt consists of a null line or a line with only a control character, the line is not usually sent to the host. However, if the NPU has no more queued data to be sent to the terminal, the null line is sent to the host.

Code Conversion

In character mode, all IVT data is converted to ASCII code whether the terminal code is ASCII or external BCD. The ASCII Mode 4A translation includes folding lowercase into uppercase and substituting blanks for any control codes. The Mode 4C translation substitutes blanks for the control codes but allows the transmission of the lowercase codes.

Cursor Control

The TIP returns the cursor to the leftmost character on the next line following the end of each input or output line. A blank line appears on the screen if the ETX symbol from an input request is in the last column or if the output ends in the last column. This is required to allow positioning of the send index for the next line.

Whenever the send index terminator is detected as the first two characters (an escape/control code pair), it is deleted before sending the message to the host.

Cursor positioning to the left margin is accomplished in one of three ways, depending on terminal class (terminal class is initially configured and can be changed with the TC IVT command from the terminal user or application):

- 214 and 200 User Terminals. Each input causes the TIP to output a sufficient number of blanks to move the cursor to the left margin of the next line. Each output is padded with blanks to move the cursor to the left margin of the next line.
- 731/732 and 734 Terminals. Each input causes a clear line to unlock the keyboard. Each output is terminated with a clear line.
- Mode 4C Devices (711 and 714). Each input causes a carriage return, backspace sequence. Each output is terminated with a carriage return, backspace sequence. In either case, the cursor is at the left margin of the next line.

CANCEL CHARACTER PROCESSING

When the TIP detects a cancel character <CN> in the input line preceding the end of logical line indicator, the TIP discards that logical line. Then the TIP notifies the terminal that the line was discarded by sending a *DEL* message downline.

Message Type Indicators (MTI)

The MTI codes shown in figure 10-2 are in hexadecimal notation, exclusive of parity. The type of MTI code affixed to output data is a function of the format effector in character mode only. For transparent mode, MTI is always write.

E Codes

For downline transforms, device selection is performed by E codes which are appended to the output by the TIP. For upline transforms, E codes coming from the terminal indicate the responding device and also report status. Received E codes are stripped from the input data by the TIP. Table 10-5 shows the E codes.

UPLINE AND DOWNLINE BVT TRANSFORMS

The Mode 4 TIP converts downline data from BVT specifications to the Mode 4 protocol. This conversion is limited to the actual features of the 200 User Terminal printer as described in table 10-6.

Any BVT code pair beginning with FF_{16} is considered an error if not supported by the Mode 4 transform. Any sequence of characters not preceded by a legal BVT code pair is also considered a host error. All such errors are reported by sending a BRK block to the host.

Upline data is translated from the Mode 4 protocol to the BVT specifications as described in table 10-7. Each card other than EOR or EOI is scanned for spaces. Trailing spaces are removed. Each card is terminated with the end-of-media indicator. Blank cards send only an end-of-media indicator. Sequences of uncompressed data are preceded by the string indicator.

MTI in Transmitted Block (hexadecimal)	MTI in Received Block			
	REJECT (18_{16})	ACK (06_{16})	ERROR (15_{16})	READ (13_{16})
05 Poll	X		X	X
12 Clear Write	X	X	X	
0C Reset Write	X	X	X	
11 Write	X	X	X	
07 Alert		X	X	
31 Configuration	X		X	X

POLL, ALART, REJECT, ACK and ERROR transmission blocks are non-data blocks and have the following format.

Sync	SOH	CA	TA	MTI	ETX	LPC

Figure 10-2. MTI Codes for Mode 4

TABLE 10-5. E CODES

E Code	E Code (Hexadecimal)	Write (Output)	Read (Input)
E1	42	To CRT (text)	From CRT (text)
E2	20	To printer (text)	From printer (no text); indicates possible error in printing last block From card reader (text); indicates that card reading has stopped
E3	21	To card reader (no text): enables transfer of card buffer to CRT buffer	From printer (no text); indicates that last block correctly printed From card reader (text); normal card data
E4	22	To CRT (text): position to start index	Not used

TABLE 10-6. DOWNLINE BVT TRANSFORMS FOR 200 USER TERMINAL PRINTER

BVT Interface (hexadecimal)		Conversion	
		ASCII (hexadecimal)	EBCDIC (hexadecimal)
mode change	FF00 to FF09	nul	nul
forms control	FFE0	20	50
	FFE1	4A	4A
	FFE2	4A1B4020	4A3E5050
	FFE3	50	30
	FFE4	41	41
	FFE5 - FFFE	20	50
compressed zeros	FFF32	3030	4A4A
	FF33	1B44	3E43
	.	.	.
	.	.	.
	.	.	.
	FF3F	1B4F	3E4F
compressed blanks	FF12	2020	5050
	FF13	1B23	3E23
	FF14	1B24	3E3F
	.	.	.
	.	.	.
	.	.	.
	FF2F	1B3F	3E3F
end of media	FF0A	1B40	3E50

TABLE 10-7. UPLINE BVT TRANSFORMS

Mode 4 Interface	BVT Interface (hexadecimal)	
Beginning of uncompressed data	string indicator	FF90
End of card	end of media	FF0A
esc 57_{16} in column 1†	end of record	FF0B
esc 56_{16} in column 1†	end of information	FF0C

†esc indicates escape; $1B_{16}$ for ASCII and $3E_{16}$ for external BCD.

Special processing occurs for EOI, EOR and JOB cards (first card following an EOI card) to transform them to the BVT form specified in section 6. The TIP does not interpret columns 79 and 80 of the JOB and EOR cards.

The card data is transmitted as read from the terminal. For external BCD the data is converted to ASCII as specified by the system code conversion table.

The TIP ensures that each transmission block of data received from the card reader contains a multiple of 80 characters. If it does not, the data is discarded and an input stopped command is sent to the host.

ERROR HANDLING

The Mode 4 TIP handles two types of errors:

- Short-term errors in which an error counter is incremented and the operation is retried.
- Long-term errors in which the shoft-term errors cannot be corrected; an irrecoverable error is declared and the I/O is terminated.

SHORT-TERM ERROR PROCESSING

The TIP performs short-term recovery for both input and output. The TIP retains three error counters, as follows:

Error Counter	Type of Error
1	No response: after transmitting to the terminal, a response timeout occurs; SOH is never received.
2	Bad response: Cluster Address (CA) or terminal address (TA) does not correspond to terminal addressed by transmit block invalid message type indicator invalid or missing E code ETX missing (overlength block or data carrier detected signal drops prematurely) character of longitudinal parity error text in block that should not have text
3	Error response: indicates transmit error

Whenever any error occurs, the TIP increments the appropriate counter and retries the output/input sequence. If any counter reaches threshold value (set at 5) in an attempt to complete a single transaction with the terminal, the TIP performs the long-term error handling procedures.

LONG-TERM ERROR PROCESSING

When the TIP cannot recover from a short-term error while communicating with a terminal, the host is sent a STP block. For a Mode 4A terminal, the STP block is sent for all connections on the cluster. The terminal is then polled every 10 seconds until the problem is resolved. When a read response is detected for the terminal, the host is sent a STRT block. A terminal status service message is generated each time a change in terminal status is noted.

DUPLICATE WRITE ERRORS

Those terminals which do not have separate CRT and transmission buffers (such as the 200 user terminal) write output data directly to the CRT screen as it is being received. If the terminal detects an error in the block, it sends an ERROR response, causing the TIP to retransmit the output. However, the cursor is not in the same place as it was when the original WRITE was performed, so the output block can appear two (or more) times on the CRT screen. This is not a problem with RESET WRITE or CLEAR WRITE which home the cursor before displaying the output data, and thus overwrite the bad block.

The toggle bit returned from Mode 4 terminal differs depending on terminal type:

- The 200 user terminal and compatible terminals always return the toggle state of the last good write regardless whether the terminal is responding to a write request or to a poll for status request.

- The synchronous Tektronix 4014 terminal and 711 terminals always return the toggle state of the last message received. If the last message is a poll for status request, then the terminal returns the toggle state of the poll message.

The Mode 4 TIP compensates for these Mode 4 terminal differences as follows:

- The toggle state of the terminal is initialized by writing a null message in order to guarantee delivery of the first block of output.

- When polling for status due to a lost terminal response to a write, the TIP sets the toggle state opposite to the state of the last write. If the toggle bit in the response is the same as in the poll, the block is sent again. This method guarantees that all output blocks are correctly received by the terminal. No blocks are duplicated (except for 711 terminal) since (1) the block is not sent more than once for 200 and 714 terminals and (2) the 4014 terminal discards a block if the toggle state is the same as the previous block.

- In the case of the 711 terminal, it is impractical to prevent the sending of duplicate blocks since the terminal neither supports polling for status nor contains logic for discarding duplicate blocks.

LOAD REGULATION

If the TIP is unable to acquire sufficient buffers for an input block or when the host is down, the TIP discards the partial block and repolls the terminal later when the condition is cleared. No error counter is incremented by this operation. However, a counter is incremented in the NPU statistics block to indicate the number of times that regulation has taken place.

AUTORECOGNITION

The host can request autorecognition for Mode 4 lines. This activates a procedure for determining the cluster address and terminals that exist on the line. When the host configures the line, the TIP responds with the line enable response. If the line is dedicated, autorecognition begins. The line is switched and the TIP waits until the ring indicator is present.

Autorecognition begins with a cluster poll to determine the cluster address of the caller. The first four polls are done at cluster address 70_{16} to allow the caller to hear the audible tone and to allow the modem time to stabilize after the modem data switch is depressed. All cluster addresses are attempted at least twice before a failure is declared. The timeout for a nonexistent cluster is from 1/2 to 1 second.

Once the cluster address has been determined, the TIP checks for receipt of a read message. The read message contains an escape code which determines the code set in use by the terminal. Polling continues until the read message is received. For external BCD terminals, this completes autorecognition. For ASCII terminals, the configuration poll is sent to determine the configuration. If there is an error response or no response, the terminal is assumed to be Mode 4A. If a read response is detected, the terminal is assumed to be Mode 4C.

The line status operational service message is sent to the host at the normal completion of autorecognition. This service message contains the following:

Field	Description	
TT	Terminal type	
CA	Cluster address	
TA	Terminal address	for each terminal
DT	Device type	

For all terminals, the appropriate terminal type (appendix C) is reported as one of the following: Mode 4A external BCD, Mode 4A ASCII, or Mode 4C ASCII. The actual cluster address is also reported in the range $70\text{-}7F_{16}$.

For the Mode 4A external BCD or Mode 4A ASCII, three terminals are reported; these describe the console, the card reader, and the line printer. The terminal address for all three terminals is 60_{16}.

The configuration request is used for the Mode 4C terminals to determine the actual terminal address and actual device types. Only the consoles are reported.

To complete autorecognition during the dial-up procedures, the remote operator must press the send key on at least one of the displays in the cluster. This allows the code set of the terminal to be recognized.

UNSUPPORTED MODE 4 PROTOCOL FEATURES

The following features of Mode 4 devices are not supported by the TIP:

- Status request
- Alert
- Diagnostic write
- Receipt of initialization request

HASP TIP

11

The HASP multi-leaving TIP supports HASP workstations. The protocol uses bidirectional transmission over HASP lines to terminals that have both interactive and batch devices.

The HASP protocol defines two types of blocks: data blocks and control blocks. Data blocks also contain control information. Positive acknowledgment of the receipt of each block is required.

The HASP protocol automatically attempts to resend garbled blocks. If the block cannot be successfully sent after four attempts, the line is declared inoperative.

Data blocks are composed of data records, which are in turn composed of character strings. If several consecutive identical characters occur, this character string is sent as a number (the number of identical characters) plus the character. This type of data compression can save significant transmission time. Another important feature of the HASP protocol is its ability to meter the rate of output so that fast processing devices have most of the transmission time available, yet slow processing devices have data whenever they are ready to use it.

HARDWARE CONSIDERATIONS

Some of the hardware considerations for the HASP TIP are the following:

- A typical HASP workstation consists of a keyboard, a CRT display, up to 7 card readers, up to 7 printers, a processor, and (optionally) external storage (magnetic tape or disk). The processor has computer-like functions, with upline and downline data processing.
- The terminal has its own emulation package, which is loaded from the designated storage device: magnetic or paper tape, cards, or terminal mass storage.
- The internal code of the workstation is EBCDIC.
- Any hardware (computer) that can be made to respond to HASP protocol and which uses EBCDIC internal code can be used as a HASP workstation.
- Each workstation uses one NPU port (line). Device sharing is the responsibility of the HASP TIP at the NPU and the workstation processor at the terminal.
- All terminals have interactive devices and most have batch devices.
- Transmission over the line is bidirectional.
- Line speed is determined by the modem clock.

MAJOR FUNCTIONS

The HASP TIP performs the following major functions:

- The TIP interfaces the ASCII-coded virtual terminal protocol of the host to a workstation that uses the HASP protocol and EBCDIC as its internal code.
- It handles tasks by queueing them as worklist entries (WLEs) to the OPS-level TIP. The host application programs send data to one HASP device at a time. The HASP TIP sends all output data blocks to one device at a time. There is no multileaving on downline data transfers.
- It converts code between ASCII (128-character set for interactive devices, 64-character set for batch devices) and EBCDIC (128-character subset only).
- It supports upline and downline data compression for both interactive and batch devices.
- It supports data flow control to various devices by the use of a function control sequence (FCS).
- It initiates line synchronization when the line has been configured; uses an enquiry/reply protocol to determine whether the line can currently be used for a transfer.
- It provides soft error processing (retransmitting a garbled data block) and hard error processing (declaring a line inoperative when soft error processing fails to transmit data correctly).
- It rejects all data when the host is down or the NPU's supply of available buffers has reached the threshold level. Note that there can be no regulation distinction between interactive and batch data since one HASP block can carry both types of data.
- It supports autoinput. In this mode only the first 20 characters of the output data are saved to be appended to the beginning of the solicited return data.
- It discards the terminals sign-on card. A network log-in is used instead.
- It does not process autorecognition.
- It processes control messages (IVT commands) from the terminal at the workstation. The messages change IVT parameters for the terminal. The acceptable parameters define user break 1 and 2 characters, define the cancel and control message characters, and define page width. A message can also be sent to the local operator (LOP).
- It interfaces to the multiplex subsystem. Downline, IVT/BVT data is reformatted to the terminal (HASP) protocol by the text processing state programs (reached through a call to PTTPINF). The TIP then calls the multiplex subsystem command driver. The address of the converted block plus other message processing information is placed in a command packet for the command driver (PBCOIN). The multiplex subsystem is then responsible for sending the data to the HASP workstation.

Upline, the HASP data is partially processed by the multiplex subsystem using the input state programs that are part of the firmware-level TIP. Before starting the input transfer, the TIP sets up the message processing by passing the transfer parameters (including the pointer to the first input state program to be used and an input buffer address) to the command driver. After the first stage of processing is completed by the TIP's input state programs, the multiplex subsystem calls the TIP at OPS-level using a worklist entry. The TIP then uses this partially processed data as a source buffer and calls the HASP TIP input text processing programs (via PTTPINF) to demultiplex as well as to convert the upline data to IVT/BVT format.

- It rejects any attempt to send transparent data for either batch or interactive device except that downline data to the plotter (batch mode device) is not code converted.

HASP PROTOCOL

The multileaving protocol consists of the bidirectional transmission of information blocks between an NPU and a HASP multileaving terminal using IVT/BVT data at the host interface. Transparent mode is not supported. Two types of information blocks are defined:

- Control block. This contains binary synchronous communications (BSC) characters only. Table 11-1 lists commonly used HASP mnemonics.

- Data block. This contains data records composed of character strings and their associated character string control bytes. Each data record in the data block is associated with a specific peripheral device. In order to facilitate identification, a record control byte (RCB) is used to assign a stream number and a device type to the data record. Each record control byte has an associated subrecord control byte (SRCB) to provide additional information about the data record.

  A data block can consist of several data records from one or more devices. A function control sequence (FCS) is added to each data block to control the flow of data from or to any particular device.

  To facilitate error detection, a block control byte (BCB) is added to each data block. A binary synchronous communications envelope surrounds the data block.

The HASP TIP never sends multileaved downline data to the HASP terminal. The host must send to the HASP TIP the desirable length of data for each active output stream (device) to make a single data block.

NOTE

Multileaving is a synonym for interleaving data from various devices in a single transmission block.

The HASP TIP does support multileaved data from a HASP workstation. The HASP TIP parses the input stream, relating each physical record to its associated connection number, and sends the data to the host sorted by device.

TABLE 11-1. HASP PROTOCOL MNEMONIC DEFINITIONS

Mnemonic	Definition	Use
ACK0	Acknowledge block or character	Positive acknowledgment that transmission was received
BCB	Block control byte	Used for error detection; includes block sequence number
BSC	Binary synchronous communications control characters	Any of several block control characters such as DLE, STX, and ETB
CRC	Cyclic redundancy check	Data quality checksum
DLE	Data link escape control character	BSC control character
ENQ	Enquiry control character or block	Inquiry if transmission can be started when terminal is newly configured
EOF	End-of-file block	
ETB	End-of-transmission block character	BSC control character
FCS	Function control sequence block	Controls data transmission rate from/to a device
NAK	Negative acknowledgment block	Confirms that transmission failed
PAD	Padding control character	All are 1's
RCB	Record control byte	Stream number and device type ID; contains status information
SCB	String control byte	String length and type, duplicate character
SOH	Start of header character	BSC control character
SRCB	Subrecord control byte	Additional data record information
STX	Start of text character	BSC control character

TABLE 11-1. HASP PROTOCOL MNEMONIC DEFINITIONS (Contd)

Mnemonic	Definition	Use
SYN	Sync control character	Maintains line synchronization
WLE	Worklist entry	
IVT Parameter Control Mnemonics		
CN	Cancel character	Defines symbol for cancel key
CT	Control character	Defines symbol to be used with operator-initiated control messages
B1	User break 1	Defines symbols for user breaks
B2	User break 2	
MS	Operator message	Allows terminal operator to send message to local operator
PW	Page width	Defines page width

TERMINAL OPERATIONAL PROCEDURE

The workstation software is loaded and the communications line is initialized. After the sign-on card is transmitted, the NPU and the terminal transmit idle blocks until one or the other initiates a function (data or command transfer).

When a function other than a console message or console command is desired, the process trying to initiate the function transmits a request to initiate function transmission RCB. The receiving process then transmits a permission to initiate function transmission RCB if the data from the requesting process can be handled. If the data cannot be handled, or a function is currently being processed, the request to initiate a function transmission RCB is ignored.

When a permission to initiate a function transmission RCB is received, the requesting process begins transmitting data blocks to the other process. Data blocks can be transmitted until an EOF is encountered. In order to transmit more data blocks for the same device stream, the request to initiate a function transmission RCB sequence must be repeated. If a request to initiate a function transmission is not received before data blocks are received, the data blocks are ignored.

Data blocks are transmitted and acknowledged one block at a time. Before a second block can be transmitted, the receiving process must transmit a positive response which takes one of two forms: if no data is ready to be transmitted to the sending process, an acknowledge block is sent; otherwise, the next waiting data block is transmitted to the sending process.

Console functions (operator messages and commands) do not have to follow the request-to-initiate/permission-to-initiate sequence. A console function can be initialized any time that the wait-a-bit in the function control sequence is not set and the remote console bit is set.

MULTILEAVING BLOCK DESCRIPTIONS

CONTROL BLOCKS

The multileaving protocol uses four types of control blocks:

- Acknowledge block
- Negative acknowledge block
- Enquiry block
- Idle block

Table 11-2 lists significant EBCDIC characters associated with these blocks.

TABLE 11-2. HASP SIGNIFICANT EBCDIC CHARACTERS

Char	Hex Value	Meaning
SOH	01	Start of header
STX	02	Start of text
DLE	10	Data link escape
ETB	26	End-of-transmission block
ENQ	2D	Enquiry
SYN	32	Synchronize
NAK	3D	Negative acknowledge
ACK0	70	Positive acknowledge
PAD	FF	Pad

Note: ACK0 only has significance in the sequence DLE ACK0 (as the entire message) since ACK0 is not a protocol character.

Acknowledge Block (ACK)

The acknowledge (ACK) block consists of the following control characters:

SYN, SYN, SYN, DLE, ACK0, PAD

SYN - Synchronization control character
DLE - Data link escape control character
ACK0 - Affirmative acknowledgment control character
PAD - Pad control character (all 1's)

The ACK block indicates that the previous block was received without error and no data is available for transmission.

Negative Acknowledge Block (NAK)

The negative acknowledge (NAK) block consists of the following control characters:

SYN, SYN, SYN, NAK, PAD

SYN - Synchronization control character
NAK - Negative acknowledgment control character
PAD - Pad control character (all 1 bits)

The NAK block indicates that the previous block was received in error and a retransmission is necessary. If the allotted number of retry attempts have been completed, the line is declared inoperative. A NAK block cannot be transmitted as a response to a NAK block.

Enquiry Block (ENQ)

The enquiry (ENQ) block consists of the following control characters:

SYN, SYN, SYN, SOH, ENQ, PAD

SOH is the start-of-header control character and
ENQ is the enquiry control character

The enquiry block establishes communications between the HASP terminal and the NPU at loading time. It is not used at any other time.

Idle Block (ACK0)

The idle block is an ACK0 block which is used to maintain communications and to avoid an unwanted timeout when neither process has any data to transmit. An idle block is transmitted at least once every 2 seconds. This block has the same format as the acknowledge block.

CONTROL BYTES FOR DATA BLOCKS

Each data block has at least one sequence of five control bytes that define the data immediately following the last control byte. The control bytes in the order they appear are as follows:

- Block control byte (BCB); used for sequencing blocks
- Function control sequence (FCS); defines the transmission flow (suspends all data or the data for a device, or restarts data transmission for one or all devices)
- Record control byte (RCB); carries status information for following data and stream identification
- Subrecord control byte (SRCB); carries more status and data control information
- String control byte (SCB); describes data string (length and nature, whether compressed or uncompressed data)

Following the first set of five control bytes, additional data subblocks can be preceded only by an SCB or by a sequence of RCB/SRCB/SCB.

Each control byte is defined below. Figure 11-1 shows a typical transmission block and its associated control bytes.

NOTE

> The bytes in the following descriptions are described as if they appeared on a card input device. That is, the least significant bit is on the left, the most significant bit is on the right.

BLOCK CONTROL BYTE (BCB)

The block control byte format is as follows:

```
0      7
ØXXXCCCC
```

Ø - must be a 1 (on)

XXX - 000 = Normal block
- 001 = Ignore sequence count
- 010 = Reset expected block sequence count to CCCC
- 011 - 111 = Not used in this implementation

CCCC - Module block sequence count, range 0 through 15

FUNCTION CONTROL SEQUENCE (FCS)

The function control sequence (FCS) format is as follows:

```
0      7       F
ØSRRABCDØTRRWXYZ
```

Ø - Must be a 1 (on)

S - 1 = Suspend all stream transmission (wait-a-bit)
0 = Normal state

SYN	
SYN	- Synchronization characters
SYN	
DLE	- BSC leader (SOH if no transparency feature)
STX	- BSC start-of-text
BCB	- Block control byte
FCS	- Function control sequence (2 bytes)
RCB	- Record control byte for record 1
SRCB	- Subrecord control byte for record 1
SCB	- String control byte for record 1
DATA	- Character string
SCB	- String control byte for record 1
DATA	- Character string
SCB=0	- Terminating string control byte for record 1
RCB	- Record control byte for record 2
SRCB	- Subrecord control byte for record 2
SCB	- String control byte for record 2
DATA	- Character string
SCB=0	- Terminating string control byte for record 2
RCB=0	- Transmission block terminator record control byte
DLE	- BSC trailer (SYN if not in transparent mode)
ETB	- BSC ending sequence
CRC-16	- Cyclic redundancy checksum (2 bytes)
PAD	- All 1 bits

Figure 11-1. Typical HASP Multileaving Data Transmission Block

NOTE

For the following bits: a bit = 1 = continue (restart) function transmission; a bit = 0 = suspend (stop function transmission).

T - Remote console stream identifier
R - Not used
ABCDWXYZ - Various function stream identifiers

These stream identifiers are bit-defined and have two sets of definitions; one for upline use, the other for downline use. For upline use, the bits identify the card reader that is to send data:

Card reader 1 = A
Card reader 2 = B
Card reader 3 = C
Card reader 4 = D
Card reader 5 = W
Card reader 6 = X
Card reader 7 = Y
Card reader 8 = Z

For downline use, the bits identify the punch or printer that is to receive the data:

Printer 1 = A = Punch number 8
Printer 2 = B = Punch number 7
Printer 3 = C = Punch number 6
Printer 4 = D = Punch number 5
Printer 5 = W = Punch number 4
Printer 6 = X = Punch number 3
Printer 7 = Y = Punch number 2
Printer 8 = Z = Punch number 1

RECORD CONTROL BYTE (RCB)

The record control byte bit representation is as follows:

```
 0      7
┌────────┐
│0SSSSSSS│
└────────┘
```

0 - 1 (must always be on)

SSSSSSS - Additional record information, dependent upon record type (see RCB above)

For general control record:

SSSSSSS = 10000001 = Initial terminal sign-on

For request or permission to initiate a function transmission:

SSSSSSS = Stream identifier and record type identifier as described in RCB

For bad BCB on last block received:

SSSSSSS = Expected block sequence count

For print record:

SSSSSSS = MCCCCCC

M - 0 = Normal carriage control
1 = Not used

CCCCCC - Carriage control information
1000NN = Space immediately NN spaces
11NNNN = Skip immediately to channel NNNN
0000NN = Skip NN spaces after print
01NNNN = Skip to channel NNNN after print
000000 = Suppress space

For punch record:

SSSSSSS = MMBRRSS

SS - Punch stacker select information

B - 0 = Normal EBCDIC card image
1 = Not used

MM - 00 = SCB count units = 1
01 - 11 = Not used

RR - Not used

For input record:

SSSSSSS = MMBRRRR

MM - 00 = SCB count unit = 1
01 - 11 = Not used

B - 0 = Normal EBCDIC card image
1 = Not used

RRR - Not used

STRING CONTROL BYTE

The string control byte bit representation is as follows:

```
 0      7
┌────────┐
│OKTCCCCC│
└────────┘
```

O - 0 = End of record (KTCCCCC = 0)
1 = All other SCBs

K - 0 = Duplicate character string

T - 0 = Duplicate character is a blank
1 = Duplicate character is non-blank (character follows SCB)
CCCC = Duplication count

K - 1 = Non-duplicate character string

TCCCCC - Character string length

If KTCCCCC is 0 and O is 1, SCB indicates record is continued in the next transmission block. This feature is not supported by the HASP TIP and is shown for completeness only.

DATA BLOCK DESCRIPTION

Data blocks consist of data records, the control bytes, and the following text control characters:

SYN - Synchronization control character

DLE - Data link escape control character

SOH - Start of header control character; used only if nontransparent mode

STX - Start of text control character

ETB - End-of-transmission block control character

CRC-16 - Cyclic redundancy checking control characters (2 bytes)

PAD - Pad control character (all 1 bits)

A typical data transmission block is shown in figure 11-1.

Several types of blocks are specially defined. These blocks appear to be data blocks but are actually special purpose blocks containing transmission control information. They are as follows:

- Operator console blocks
- End-of-file blocks
- FSC change blocks
- Sign-on blocks
- BCB error blocks

OPERATOR CONSOLE BLOCKS

Blocks which contain operator console messages or commands do not contain any additional records in the data block following the console record.

A request to initiate a transmission function is not required to transmit console records. However, the wait-a-bit flag must not set in the FCS, and the remote console bit must be set.

END-OF-FILE BLOCKS (EOF)

Blocks that contain the end of file (EOF) indicator do not contain any additional records from the same device stream in the data block following the EOF. Data blocks terminated by an EOF contain a final record in the following format (shown for card reader 1):

(BSC header)

BCB	
FCS	
RCB	10010011 - Card reader stream 1
SRCB	10000000 - SCB count units = 1, EBCDIC card images
SCB	00000000 - EOF
RCB	00000000 - Transmission block terminator (BSC trailer)

(BSC trailer)

To transmit additional records for a device stream that contains an EOF, the request to initiate a function transmission must be transmitted again. If another device stream contains data for transmission and has permission to transmit, the last RCB in the above example would be a device stream RCB followed by data instead of a transmission block terminator.

FCS CHANGE BLOCKS

The FCS change block is transmitted when the status of one or more of the streams has changed, and there is no data ready to transmit. The FCS change block format is as follows:

(BSC header)

BCB	
FCS	- Changed FCS
RCB	00000000 - Transmission block terminator

(BSC trailer)

USER INTERFACE

The user is required to load the software into the HASP workstation processor, to execute this initializing software, to sign-on after the communications line is configured (by the HASP TIP and the workstation), and to sign off.

WORKSTATION STARTUP AND TERMINATION

The workstation startup procedure consists of three steps:

- Terminal initialization at the HASP workstation
- Communications line initialization, which involves the workstation, the NPU, and the host
- Signing-on, which involves the workstation and the HASP TIP in the NPU

WORKSTATION INITIALIZATION

The HASP workstation operator loads the terminal software and executes it. The loading medium can be paper tape, cards, magnetic tape, or mass storage depending upon the terminal hardware. The workstation initialization processor establishes I/O buffers and other necessary parameters. After initialization, a card is read from the card reader. If the card is blank, the default sign-on parameters are used (default sign-on parameters are assembled into the terminal software). If the card is a /*SIGNON card, the parameters on the /*SIGNON card are used instead of the default. In either case, the /*SIGNON card is discarded by the HASP TIP; it is not passed to the host.

COMMUNICATIONS LINE INITIALIZATION

After the terminal is initialized, the communications line is initialized by the HASP TIP upon receipt of a configure line service message (SM) from the host. When communication is established with the line, communication between the HASP TIP in the NPU and the HASP workstation is established by the following procedure:

- An ENQ block is sent from the workstation to the HASP TIP.
- The ENQ is ignored by the HASP TIP until configure terminal SM arrives from the host for the HASP console stream. The HASP TIP then sends an ACK0 to the ENQ.
- If the ACK block is received by the workstation, the sign-on record is transmitted to the HASP TIP.
- If I/O errors occur or the ACK0 block is not received, the process restarts with another ENQ block.
- After the sign-on record is transmitted and a positive acknowledgment is received (ACK0), the workstation is ready for normal processing.
- As each individual batch device stream is configured by the host, the INIT block is received and the HASP TIP allows processing of the corresponding output streams. For batch input streams, processing does not begin until a START INPUT command is received for the input device stream. For the console input stream, input is allowed after the receipt of a downline data block or a Start Input command.

SIGN-ON BLOCK

```
Column  1      16        25
        /*SIGNON REMOTEnn  password
```

NOTE

Record is shown in punched card format, with least significant character on the left, most significant character on the right.

The nn is a one or two digit number that can be used to correlate this remote terminal with information about it in the host computer. Password can be blank. The sign-on block format is shown in figure 11-2.

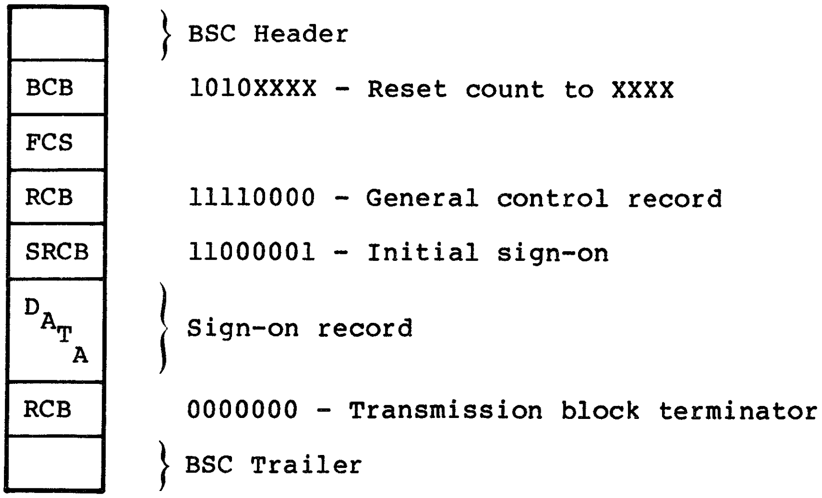


Figure 11-2. Sign-on Block Format

The sign-on record is not sent to the host since the host requires a separate logging-on procedure at the operator's console.

SIGN-OFF BLOCK

The /*SIGNOFF card, when transmitted to the HASP TIP as a record in the data block, has the same effect as an EOF block. The HASP TIP converts the signoff record to a BVT EOI and sends it to the host as a MSG data block.

HOST INTERFACE

The host interface is used for connection configuration and initialization of the workstation devices. Once the line becomes operational, the HASP TIP allows the sign-on block to be sent from the HASP workstation. The sign-on block is acknowledged to the HASP workstation but is not delivered to the host.

Upon receiving a line operational service message for a HASP workstation, the host issues a configure terminal service message to configure the workstation's console. A downline data block or a Start Input command causes the HASP TIP to permit input from the workstation console. The console connection allows the workstation operator to send and receive messages to and from the host. It also allows the operator to alter IVT parameters.

After the console is configured, the batch devices are configured. Start Input commands from the host cause the HASP TIP to allow input devices to read cards. Output device streams are initiated by the HASP TIP as soon as data arrives. Devices configured as plotters use card punch output streams.

Once the necessary initialization and configuration are complete, traffic can flow between the terminal and the host. During this traffic handling period, the HASP TIP is involved in the following functions.

- Code conversion - upline and downline
- Format conversion, HASP TO BVT/IVT upline, BVT/IVT to HASP downline
- Flow control upline and downline
- HASP error recovery procedures
- Input/output streams, to/from a HASP terminal

CODE CONVERSION

Interactive virtual terminal (IVT) data to and from the console is translated by the TIP by converting ASCII to EBCDIC or EBCDIC to ASCII code. Only the 128-character subset equivalent to the ASCII character set is converted; that is, on output the eighth bit is stripped off, and on input any character not in the subset is converted to a blank.

Batch virtual terminal (BVT) data to and from the batch devices (except the plotter) is translated by converting ASCII to EBCDIC or EBCDIC to ASCII. Here only the 64-character subset of ASCII appears in the data from and to the host. Plot data is sent untranslated to the terminal.

Upline, card reader input is translated by an EBCDIC-to-ASCII conversion by default (029) or if requested by a job card or an end-of-record card with 29 punched in columns 79 and 80. An alternate 026-to-ASCII conversion can be requested by punching 26 in columns 79 and 80 of a job or end-of-record card. Subsequent end-of-record cards with 29 or 26 punched in columns 79 and 80 change the conversion mode; the current mode is kept in effect with any other punches. The conversion mode is returned to default (029) when an end-of-information card is input. No indication of conversion mode is sent to the host.

Downline, printer data is always translated by an ASCII-to-EBCDIC conversion. Punch data is translated by the same conversion by default or if requested by a MODE CHANGE of ASCII-029. An alternate ASCII-to-026 conversion can be requested by a MODE CHANGE of ASCII-026. The requested or default mode stays in effect until changed by a subsequent MODE CHANGE request or until the receipt of an ENDOFINFORMATION. At that time, the mode is returned to the 029 default.

Note that on card input in 026 mode, 12-8-2 is read as 12-0 and 11-8-2 read as 11-0. Neither 12-8-2 nor 11-8-2 is punched on card output in 026 mode. Similarly, on card input in 029 mode, 11-0 is read as 12-8-7 and 12-0 is read as 12-8-4. Neither 11-0 nor 12-0 is punched on card output in 029 mode.

For downline data, the HASP TIP appends the BSC envelope to the data in the same manner as the BSC envelope was received from the HASP workstation.

HASP/BVT FORMAT CONVERSION

Conversion of the HASP/BVT format is required for compressed data, both upline and downline, for uncompressed data, for EOI/EOR codes, for forms control codes, and for punch banner cards.

Compressed Data (Upline)

The HASP TIP converts the string control bytes to BVT compressed format. String control bytes designating blank compression are converted directly to BVT blank compression codes. Trailing blanks are stripped.

The HASP terminal does not distinguish between compressed blanks and compressed zeros as BVT does. Zeros are treated the same as any other repeated character by the HASP TIP for conversion to BVT.

Compressed Data (Downline)

The HASP TIP converts BVT compressed format to string control byte format. BVT compressed zeros are expanded to the string control byte format for repeated characters.

EOI/EOR Codes

All data blocks between the host and the HASP TIP are BLK blocks, except for the last block of data which is a MSG block. The MSG block is the end-of-information (EOI) block and indicates no more data transmission follow.

Upline, the end-of-input block (MSG block) is sent by the HASP TIP when an end-of-file block is received from a card reader stream. Contained within the MSG block is a BVT EOI. /*EOI cards received from the card reader stream cause the HASP TIP to send the MSG block as well. Consecutive /*EOI cards are ignored by the HASP TIP.

/*EOR or 789 cards received from the card reader stream cause the HASP TIP to convert the EOR to its special BVT equivalent. The TIP obtains the level number from the card and also passes that information to the host.

In addition to the preceding, the HASP TIP scans the first card from an input device stream, the card after a /*EOI (assumed to be a job card), and the /*EOR or 789 card. The TIP checks columns 79 and 80 for code conversion (26, 29). An appropriate conversion table is selected, based on the information in columns 79 and 80.

Downline, the MSG block causes the HASP TIP to send the associated output device stream an end-of-file block.

BVT EOR/EOIs are converted to 789 cards (with the appropriate level number) and to /*EOIs. For output devices other than the card punch, these EOR/EOI symbols are ignored by the HASP TIP.

Uncompressed Data

The HASP TIP converts uncompressed string control bytes to BVT uncompressed control codes and converts BVT uncompressed control codes to uncompressed string control bytes.

Forms Control Codes

The BVT forms control codes are converted to subrecord control bytes for printer streams. For each possible BVT code, there is an equivalent preprint or postprint subrecord control byte.

Punch Banner Cards

The downline data block clarifier contains a flag which, when set in a block for a punch file, indicates that a laced card should be generated before sending the data to the terminal. The laced card consists of 80 columns of the EBCDIC punch 58_{16}, which punches rows 12, 11, 8, and 9.

HASP/IVT FORMAT CONVERSION

The IVT command allows the workstation operator to vary some workstation parameters. These parameters apply only to the workstation console which is the interactive HASP workstation device. The following IVT parameters can be changed:

- CN - designates key to be used as cancel character
- CT - designates key to be used as control message character
- B1, B2 - designate keys to be used as user breaks 1 and 2
- MS - designates key to be used to delimit a message from the workstation console to the local operator (LOP) console
- PW - designates page width on the workstation display

Format of the message as entered at the workstation is

<CTL><OTHER> < >

CTL is the control symbol

OTHER designates one of the six allowable parameters above

< > is the terminator character for the console as defined by the CT parameter

HASP compressed data is expanded to IVT format. Page width (PW) line folding past the column specified by the PW parameters (or the default value if PW has not been specified) is performed by sending multiple output lines.

Autoinput is supported. In the autoinput mode, only the first 20 characters of the output block are appended to the solicited input data. Autoinput is confined to a single MSG block on output. Input longer than one line is ignored.

Transparent data is not allowed in IVT mode. Any attempt by the user application to send transparent data downline causes the HASP TIP to send a BRK block to the host, terminating the output attempt.

If a cancel input line character is sent to the TIP, the TIP deletes the previous input line and sends a *DEL* message to the terminal.

Downline IVT format effectors (FEs) are defined in table 11-3. No parsing of any FE within the output line is performed.

For output data blocks that contain a format effector only, the HASP TIP ensures that a blank character is inserted into the output stream. This prevents the HASP workstation from changing a format-effector-only data block into an end-of-file block.

TABLE 11-3. DOWNLINE IVT FORMAT EFFECTORS FOR HASP TERMINALS

Preprint	FE	Action
Single Space	Space	No action required
Double Space	0	Generate one blank line
Triple Space	-	Generate two blank lines
Start of Current Line	+	Ignore
Home	'*'	Ignore
Home and Clear	1	Ignore
Null	,	Ignore
Postprint		
Single Space	.	No action required
Start of Current Line	/	Ignore

ERROR HANDLING

The NAK block is the basic method of informing the receiving process that an error occurred. The TIP saves the last downline data block for the terminal so that it can be retransmitted if needed. Retransmission of a data block is attempted three times following the initial NAK. For output blocks that are undeliverable to the terminal, the HASP TIP causes the service module to generate a line status service message with line inoperative indication. This service message is sent to the host.

The HASP TIP, after receiving the same block incorrectly four times from the terminal, considers the data as unreadable. The TIP causes the service module to generate and to send a line inoperative SM to the host.

The error conditions recognized by the HASP TIP are as follows:

- CRC-16 error
- Illegal block make-up
- Unknown response
- Timeout
- BCB error

CRC-16 ERROR

Cyclic redundancy checking (CRC) occurs only on data blocks. If a CRC-16 error occurs, the receiving process transmits a NAK block to the transmitting process. This indicates that a retransmission of the last block is required. If the retransmitted block is correct, the processing continues.

ILLEGAL BLOCK MAKE-UP ERROR

A data block must end with an ETB control character; if it does not, an illegal block make-up error occurs. The receiving process transmits a NAK block to the transmitting process, which informs the transmitting process that a retransmission of the last block is required. If the retransmitted block is correct, the processing continues.

UNKNOWN RESPONSE ERROR

An unknown response error occurs when the response received from the transmitting process is not one of the following:

- A data block beginning with the DLE and STX control characters in transparent mode

- A data block beginning with the SOH and STX control characters in nontransparent mode

- An ACK0 block

- A NAK block

If an unknown response error occurs, the receiving process transmits a NAK block to the transmitting process. This informs the transmitting process that a retransmission of the last block is required. If the retransmitted block is correct, processing continues.

TIMEOUT ERROR

If the maximum number of retries has been used or there is a hard error, the HASP TIP declares the line to be down. Otherwise, the TIP tries to resent the block.

BLOCK CONTROL BYTE ERROR

Every data block has a block control byte (BCB) that contains a block sequence count. The data blocks are transmitted in sequentially ascending order unless an ignore or reset block control byte is transmitted. If the block sequence count in the data block is not the same as the expected block sequence count, a block control byte error occurs.

If a block control byte error occurs and the block sequence count is the same as a block sequence count previously received (the expected count minus received block sequence count ≤ 2), the data block is ignored and processing continues as if a function control sequence change block or ACK0 block was received.

If a block control byte error occurs and the block sequence count is not the same as the count previously received, a block control byte error block is transmitted from the receiving process to the transmitting process. The block control byte error block informs the other process that a block sequence count error has occurred, and that the transmitting process must either return to the missing block or must transmit a reset block control byte. The format of the block control byte error block is shown in figure 11-3.

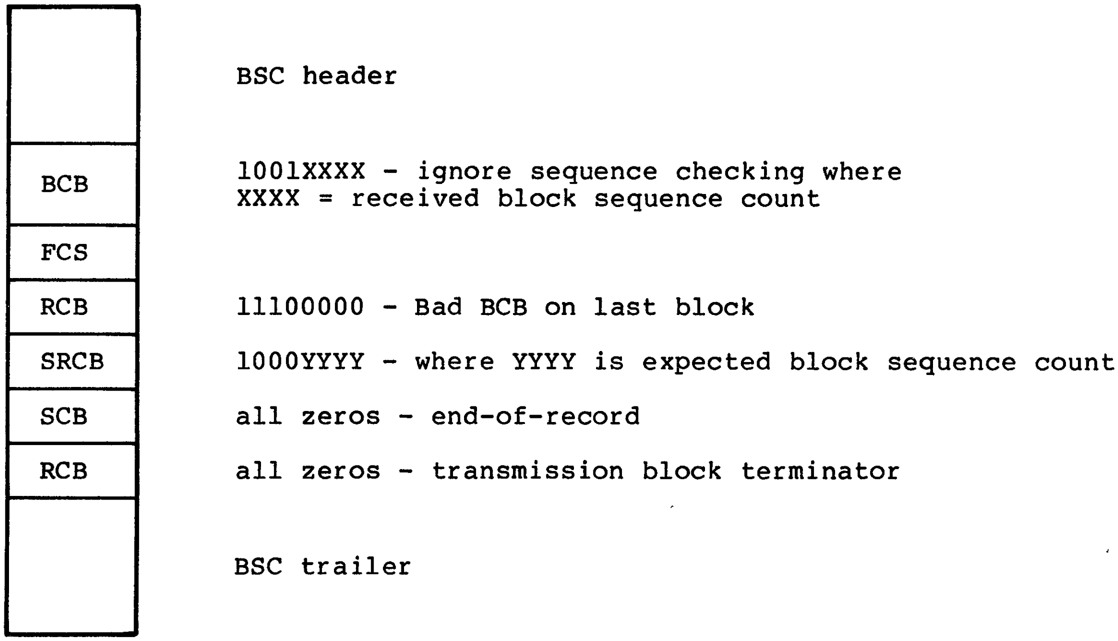


Figure 11-3. Format of Block Control Byte Error Block

REGULATION AND FLOW CONTROL

The NPU regulates upline input from the HASP workstation when the NPU runs out of buffers, when the host stops, or when data transmission is not ready. The workstation regulates downline data output from the host or NPU as a function of the busy state of the workstation device that uses or produces the data.

UPLINE REGULATION

In response to the Stop Input command, the TIP sends an Input Stopped command to the host. If data continues to arrive from the terminal, that data is discarded. No permission to transmit is granted by the TIP.

Upon receipt of an end-of-file block from the terminal, the TIP sends an Input Stopped command to the host following the data. Permission to send more data is not granted until a Start Input command is received from the host.

If the host becomes unavailable or if the NPU runs out of buffers, the TIP stops all input from the terminals by setting the wait-a-bit in the function control sequence (FCS) control byte.

DOWNLINE DATA FLOW CONTROL

The FCS fields control the flow on each of the streams (terminal devices) by the use of the bits assigned to control each stream. The FCS sent by the terminal to the TIP controls the TIP's downline delivery of records related to each stream.

The TIP correlates the FCS bits with the applicable connection numbers. If a bit is set to the suspend transmission state, the TIP sends an upline STP block on the related connection after a timeout occurs. In a subsequent upline block from the terminal to the TIP, the function control sequence bit for the specified downline stream is set to change transmission from the suspend state to the continue state. This causes the TIP to send a STRT block upline on the related connection number. This block causes the host to resume delivery of downline traffic to the TIP for that stream.

If a request to initiate function transmission sent from the HASP TIP is denied by the terminal, then a STP block is sent upline for this device's connection number (CN) after a timeout occurs. If permission is granted, a STRT block is sent.

HASP POSTPRINT

HASP printers vary in their terminal carriage control actions. Some perform carriage control, then print the data; others print the data, then perform carriage return actions. The former are called preprint terminals; the latter are called postprint terminals. The preprint terminals are designed to receive the data in the following format:

[CARRIAGE CONTROL] [DATA]

The terminal action is perform the carriage control action first, then print the data.

Initially, CCP treated all printers as preprint terminals. However, postprint terminals cannot perform these actions in one step; they use the following sequence of actions: print the data, then perform the carriage control actions.

To handle both preprint and postprint terminals with the same data format, CCP divides HASP printer output data into two records:

[CARRIAGE CONTROL] [DATA]

[2 BLANKS] [CARRIAGE CONTROL] and [DATA] [no CC]

where no CC is no carriage control character

Preprint terminals handle this as a carriage control, then a print data sequence. Postprint terminals print out two blanks from the first record (that is, nothing is printed) and then perform the carriage control action. For the second record, the postprint terminal prints the data, but performs no carriage control action.

STATE PROGRAMS

12

This section describes the firmware-level state programs, which are used by the TIPs and the multiplex subsystem to speed programming. One set of state programs controls upline transfers (input state programs, sometimes augmented by upline text processing programs) and another set controls downline transfers (text processing). Each program is composed of a series of state processes. Each state process is composed of a series of state instructions.

Each TIP (in some cases, each type of terminal serviced by a TIP) has upline and downline state programs to process control characters, assign buffers, perform error processing for garbled characters in the transmission stream, and (if necessary) to translate code. The entire group of state processes that comprise a state program has a state pointer program associated with it. To execute a program, the TIP sets an index in this pointer table to specify the first state process to be used when the next character is to be processed. The pointer table index is then moved as appropriate for the next anticipated character. This is usually done by the state programs themselves.

The multiplex subsystem also controls a set of state programs called the modem state programs.

EXECUTION OF STATE PROGRAMS

All state programs are executed on the firmware level. Message processing itself is under the control of the appropriate TIP, which is executed on the OPS level. That TIP, before starting processing of the message, sets up a multiplex line control block (MLCB) for upline messages or a text processing control block (TPCB) for downline messages. Since most of the message processing is normal (for instance, the modem is set up in the same way each time, buffers are assigned, a sequence of control characters delimit the message, termination is generally the same), this kind of processing can be handled entirely within the state programs.

As the message is processed on the firmware level, the state program index is changed on the firmware level by the state programs themselves. The state programs process the data without further communication with the OPS-level part of the TIP. For upline data, processing consists of moving data from the circular input buffer to a dynamically assigned, line-oriented input buffer. When the line buffer is ready, the OPS level TIP is called to process it. For downline data state processing consists in taking all the data from the line-oriented output buffer, translating and reformatting it for the terminal, and placing it in an output buffer. Control returns to the OPS-level TIP to continue processing the message. Usually, the TIP notifies the multiplex subsystem that the message is ready for outputting.

The ideal case summarized above makes few provisions for special problems such as error processing. In such a case, the state programs might inform the TIP that message transmission failed, and the TIP would then activate one of its OPS-level routines for handling that situation based on the type of error encountered.

State program processing is usually more complicated than in the ideal case. Processing may shift several times between firmware-level processing by the state programs and OPS-level TIP. Communication between the TIP and the multiplex subsystem is needed to set up the input state program. This communication uses the command packet. The multiplex subsystem then starts the input state programs when the first character of the message is placed in the CIB. Whenever the TIP passes control to the multiplex subsystem, the new input state index must be set in the MLCB.

Figure 12-1 shows the pointers that initially are needed to locate the first state process in a state program sequence. As a state process is completed and requires another, the index in the state pointer table is changed so the TIP or multiplex subsystem can find the next state process of the state program to be executed.

CLASSES

Functionally, there are three classes of state programs:

- Input state programs for upline processing. An input data processor handles the character processing.

  The input data processor is a multiplex subsystem level 1 microprogram which has the basic task of removing loop cell data from the input multiplexer loop, stripping away the multiplex loop control fields, and packing the resulting characters into a circular input buffer (CIB). Then the input state program is called to store an input character into a line-oriented input buffer. The current input state process determines whether any special action (code or format conversion) is required for the character and processes the character as needed. When all the input characters for that block are processed, input is terminated and a worklist entry is made to call the TIP at OPS level.

  The input data processor is interrupt driven (priority 2) by the multiplex loop interface adapter whenever a line frame is stored in the CIB. Unless pre-empted by a priority 1 interrupt, the input data processor causes the appropriate state program (input or modem) to remove all unprocessed entries from the CIB prior to relinquishing control. In this way, the CIB's pick pointer is moved up to the put pointer position whenever possible. Running out of space in the CIB causes the NPU to stop.

- Text processing state programs for downline processing. Output text processing is always required unless the output sent by the host is in transparent mode. Normally, the OPS-level TIP calls the text processing state program to convert data to terminal format. The TIP makes a direct call to the state programs using the firmware interface program, PTTPINF. The text processing program reformats and converts to terminal code where necessary. This operation moves the data from the buffers holding the output data in virtual terminal format to buffers holding the data in terminal format. This data conversion must be accomplished before calling the multiplex subsystem to initiate output on the line.

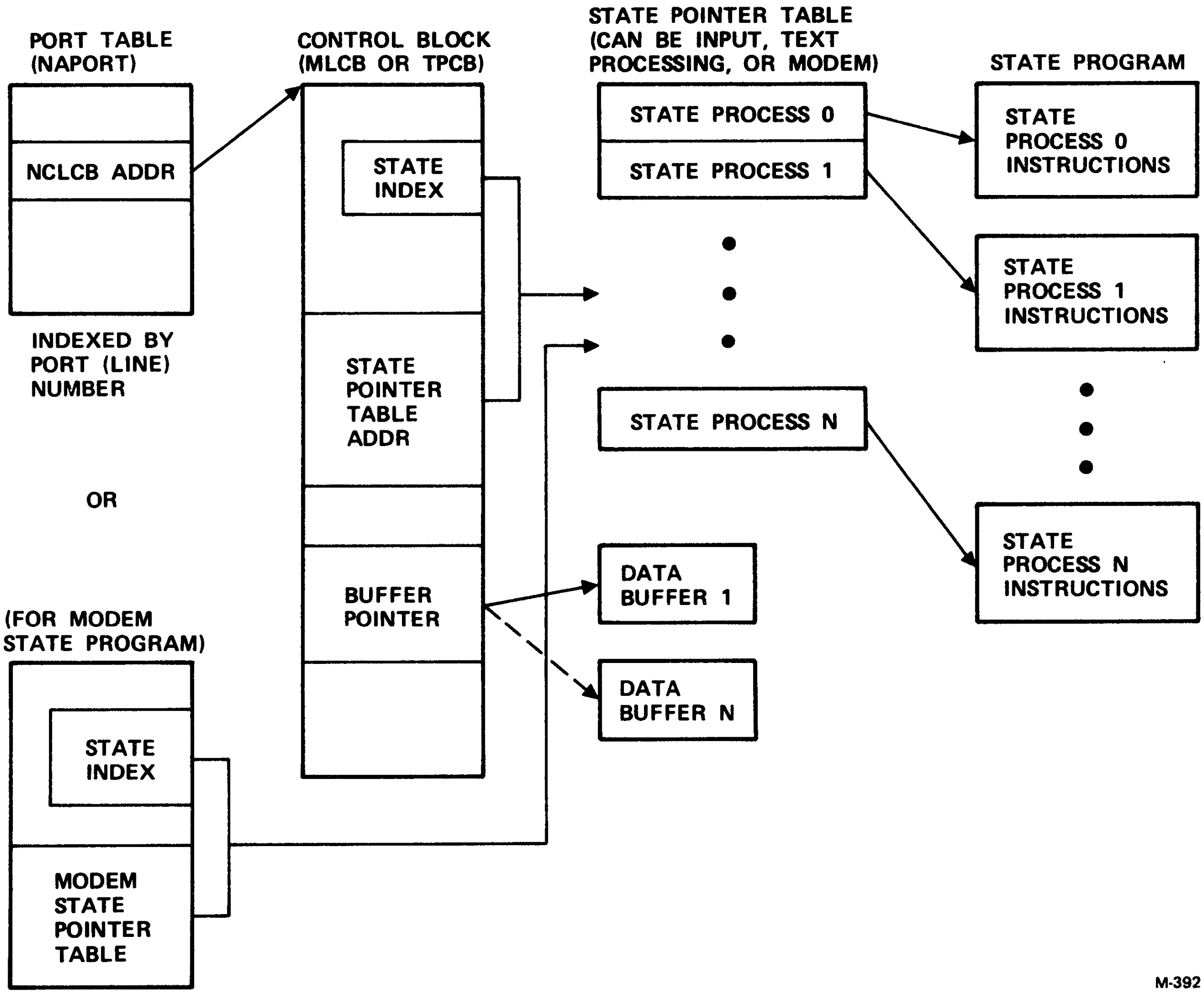


Figure 12-1. Locating a State Process

After the text is converted to terminal code and format by the text processing state program, the output data processor (ODP) in the multiplex subsystem handles the character output to the line. The output data processor is an interrupt-driven (priority 1) level 1 microprogram that is activated when an output data demand (ODD) is generated by the CLA on that line. The output data processor's primary function is to obtain a single character from line-oriented output buffer, to place this data into line frame format, and to transfer the line frame onto the multiplex output loop. This process is repeated, driven by the ODD interrupts, until the entire message is transmitted.

Text processing is also performed on some upline data. This occurs where the input block is composed of data from several devices at the same workstation, as in the case of the HASP TIP. In this case, the input state programs partially demultiplex the data into a line-oriented input buffer. Then the multiplex subsystem calls the OPS-level TIP. The OPS-level TIP calls TPPTINF to convert this block of terminal data to one or more blocks of device-oriented data in IVT/BVT format. Different sets of text processing programs are needed for upline and downline conversions.

- Modem state programs. The IDP and ODP described above handle those tasks that are protocol dependent. Modem state programs handle those tasks that are performed for all line protocols, such as processing CLA status.

COMPONENTS OF A STATE PROGRAM

There are three components of a state program:

- A state program consists of one or more state processes. The number and variety of state processes defined for a state program is a function of the particular terminal protocol. Each state program is assembled as a sequential table of coded state processes.
- A state process is composed of one or more state instructions (firmware macroinstructions). The set of these macros forms the language of state processing. For a complete description of the macros and their use, refer to the State Programming Reference Manual (see preface).
- The state pointer table contains the address of each state process defined for a particular protocol or line type. A state process is selected by setting the state index to the process number.

FUNCTIONS

The functions of the input state, text processing, and modem state programs are described in this subsection.

INPUT STATE PROGRAMS

Input state programs demultiplex characters into line-oriented input buffers. This is done in two ways:

- One-pass processing. These buffers of fully converted data are passed to the host via the TIP and the HIP.

- Two-pass processing. These buffers of partially demultiplexed data become the source buffers for input text processing. The OPS-level TIP is called to finish the demultiplexing. Then the TIP passes the fully converted data to the host via the HIP.

An input state program consists of a maximum of 64 state processes. These processes handle tasks such as data conversion, CRC generation, character compression, and message blocking. Since all state processes are reentrant, lines with a similar protocol can share some state processes.

The TIP must provide programs for the four reserved input state processes (0, 1, 2, and 3). State 0 handles parity errors or data transfer overrun. State 1 is called when the data carrier detect (DCD) signal is dropped. This condition can be used as a logical end of text for controlled carrier lines. Both state 0 and 1 are given control by the modem state program (regardless of the current input state) when the stated condition occurs. States 2 and 3 are called by the input data processor to process buffer-related conditions. State 2 is given control when the number of input buffers currently in use exceeds the system limit. State 3 receives control when the available buffer minimum threshold is reached. States 4 through 63 are defined by the TIP.

The 16-word multiplex line control block (MLCB) stores control information for the message. Numerous flags and fields are defined for the transfer, including the state process pointer and the state program index. Together, these locate the next state process to be executed. The MLCB fields are defined in appendix H.

The input data processor has three interfaces: to firmware, to modem state programs, and to text processing state programs.

Firmware Interface to Input Data Processor

When the firmware input data interrupt causes the multiplex subsystem to pass control to the designated input state process for the line or terminal. Before executing the first state input state instruction, the firmware loads a selected register with the current (untranslated) character. The contents of this register can be changed by state macroinstructions.

If parity stripping is specified, the parity bit is stripped when the register is initially loaded. If and when the register contents are changed, parity stripping is ignored. Exit options allow the TIP to store the character from the register without changing the register contents.

Modem State Program Interface to Input Data Processor

When a data character and CLA status occur in the same line frame of the CIB, the firmware transfers control to the current modem state process. The modem state program is responsible for passing control to input state process 0 or 1 upon detecting status conditions for which the input state program should get control.

Flags in the MLCB are used for communication between the modem state program and input state program. One flag indicates that a workcode has been saved for use when the carrier drops. Another flag is set by the line initializer when a controlled carrier line is detected.

The input state program must set the modem state index to the modem state process that handles status while input is in progress. That is, upon detecting start of input, the input state program must change the modem state index to the modem state process that handles status when inputting. Then, upon detecting end of transmission, the input state program must set the modem state index to the modem state process for idle.

For the controlled carrier type of line, an output message cannot be transmitted until data carrier detect drops on input. To eliminate the possibility of a TIP starting output before data carrier detect has dropped during input, the input state program has the ability to terminate the input buffer and save the workcode in the MLCB (the alternative would be building the worklist at the time of the termination). The input state program then sets a user flag indicating this saved workcode condition.

A worklist entry can be built immediately if the line type is not a controlled carrier line.

The modem state program jumps to input state process 1 when the saved workcode flag is set, data carrier detect has dropped, and the idle modem state exists. The TIP does not get control until data carrier detect has dropped, eliminating the possibility of starting output before data carrier detect has dropped during input.

Other input/modem states interfaces can be defined as needed by the user.

Text Processing State Program Interface to Input Data Processor

The input state program creates interim (source) buffers to be used by the text processing state program only when more than one pass is required to process the input from the CIB.

TEXT PROCESSING STATE PROGRAMS

These state programs handle all protocol-oriented output processing and some input processing (where several devices on the same line have data to convert within a single upline block).

When handling characters for output text processing, the buffer received from the host is referred to as the source buffer. A character from this buffer is known as a source character. For input text processing, the source character is obtained from the source buffer that was created by the input state program at the end of the first pass. The source character is placed in the current character register by the firmware.

A text processing state program consists of a maximum of 64 state processes. Since all state processes are reentrant, lines with a similar protocol can use the same state processes.

Text processing state process 0 is reserved for handling the end of a source-reached condition and state process 2 is reserved for handling buffer overflow processing. States 1 and 3 through 63 are defined by the TIP.

The selection of the text processing state process to execute is determined by combining the value of the state process index with the state pointer table address. Both fields are in the text processing state pointer table entry points to the associated text processing state process. See appendix H for a definition of TPCB fields.

The state pointer table address and state process index fields are set by the OPS-level TIP program. State program macroinstructions allow the firmware program to change the state process index while executing text processing state programs.

Before text processing is initiated, a group of 16 firmware registers (file 1 text processing registers) are initialized from the last 16 words of the TPCB by PTTPINF. This action allows the firmware to operate entirely within micromemory.

The 16 file 1 registers are accessed by specifying a displacement to the selected file 1 register. A displacement of 0 selects the first file 1 register and a displacement of 15 selects the last file 1 register.

Firmware Interface to the Output Data Processor

The destination buffers generated by the output text processing program can be accessed by the output data processor when an output data demand (ODD) is received from the communications line adapter. The output data processor gets the next character from line-oriented buffers, moves the character into multiplex output loop frame, and transfers the frame to the MLIA for transmission on the multiplex output loop.

The TIP support program, PTTPINF, provides the interface between the OPS-level TIP and the firmware which performs state-driven text processing. PTTPINF performs the following functions:

- Initializes the file 1 registers for text processing with the lower 16 words of the text processing control block (TPCB) array.
- Initiates output state processing instructions.
- Releases unused destination buffers created by the save and restore conditions state instruction upon return to macrolevel processing.
- Restores the text processing TPCB array with the file 1 registers upon return to macrolevel processing.

PTTPINF is called with a parameter containing the address of the TPCB.

After detecting a character but before executing the first state instruction, the firmware loads file register 0 and a selected register with the current (untranslated) character. The programmer can change the contents of file register 0 by using the state program macroinstructions.

If parity stripping is specified, the parity bit is stripped when the register is initially loaded. If the contents of the register are changed, parity is ignored. Exit options can store this character without changing the register contents.

MODEM STATE PROGRAMS

The modem state programs process modem status as a function of modem control signals. The programs (which are called by the firmware when communications line adapter (CLA) status word enters the subsystem) use a worklist entry to forward the logical CLA status to the multiplex level status handler (PTCLAS). PTCLAS analyzes the status and uses a worklist entry to report line conditions to the OPS-level TIP modem state program.

A modem state program consists of a maximum of 16 state processes. There are modem state processes defined for each line type based on line condition. Thus, the modem state program can have one or more processes for each condition or one state process to handle more than one line condition, depending on the line type.

The modem state programs report status conditions to the line initializer and to the TIPs. These programs are based on line type. The states defined for each line analyze the status as a function of the current state of the line (for example, line idle, output in progress, input in progress, and initializing line).

State 0 is the starting state of the modem state programs when a CLA status word is detected in the circular input buffer. This state checks for hard errors and any other signals that are common to idle, input, and output states. Control passes to the current state program if no errors are detected or if the current state is discard, initializing line, or enabling line.

State 1 discards all status. This state is selected following any hard error worklist generation or by a clear line or disable line command to the command driver.

State 2 is the common line initialization state. Upon receiving any status, this program checks the ring indicator. A worklist is generated if it is found. If the ring indicator is not included in the status word, a CLAON worklist is generated.

State 3 is the enable line state. It is selected whenever an enable line command is issued. The modem signals that indicate that the line is ready for data transfer are checked. If these are found, a worklist indicating the line is enabled is generated. The modem state program changes to state 4 (idle) after the worklist is generated. Either of two signals indicate the line is enabled: data set ready (DSR) alone, or a combination of DSR and data carrier detect (DCD).

NOTE

> States 0, 1, 2, and 3 are similar for all line types. Any new modem state programs must perform these same functions. New programs should also check the three hard error indicators: input line enabled, output line enabled, and DSR.

State 4 is the idle state. It checks for any error conditions that are not checked in state 0.

NOTE

> States 5 and 6 are unique by line type.

State 5 is the output state. It checks for output-related errors not checked in state 0, such as next character not available.

State 6 is the input state. It checks for input-related errors not checked by state 0, such as parity error status. The program also provides a jump to the TIP input state that handles the data character that accompanies the status indicator for any status condition that requires such a character (for example, PES, data transfer overrun, and SDLC character status).

NOTE

> States 4, 5, and 6 can be separate states if the line does not use full-duplex transmission. With full-duplex transmission lines, these states can be performing the same functions for handling status while input and output are simultaneously in progress.

State 7 is ready for output, reverse channel. It is not used.

The modem state index in the port table (NAPORT) can be set by the command driver, an input state program, or a modem state program. The modem state program address field is set by the command driver when a line is initialized. The command driver sets the index to the modem state process according to the command being issued. The input state programs control the setting of the modem state program index for handling status while input is in progress.

The modem state program is initially entered by accessing modem state process 0. Modem state process 0 sets the modem state index according to the status information it receives. Subsequent selection of a modem state process is determined by the modem state program address and modem state index of the port table. This combination of the index and address selects the state pointer table entry that points to the associated modem state process.

The modem state programs have three interfaces.

Firmware Interface to the Modem State Programs

CLA status is moved into the circular input buffer (CIB) along with the input data. When the firmware's input data processor detects CLA status, it passes control to modem state process 0 for that line.

Multiplex Level Status Handler (PTCLAS) Interface to the Modem State Programs

After the modem state program builds a worklist entry containing the logical CLA status, the multiplex level worklist processor routes the priority worklist entry to the multiplex level status handler, PTCLAS. Upon receiving control, PTCLAS analyzes the status condition indicator and acts accordingly. The appropriate action may be to generate a CE error message, to start a timer for modem response or CLA status overflow, or to make a worklist entry to the associated TIP at OPS-level.

Input State Program Interface to the Modem State Programs

This interface was described in the Input State Program subsection.

MACROINSTRUCTIONS

There are nine classes of macroinstructions:

- Status of the two assignable counters
- Character manipulation (store, replace, etc.)
- Index manipulation
- Skips
- CLA status handling
- Flag control (set and reset)
- Worklist handling (build, terminate, use fields)
- Text processor operations
- Miscellaneous (addresses, timers, backspace, resync, CRC, buffer allocation, block length, move fields)

The state program macroinstructions are summarized in table 12-1. The general format of a state program macroinstruction is

```
MACRO NAME  parm1,parm2,...,parmn
```

The instruction in this call format is closed up and all defined parameters must be present. If a parameter is inapplicable to the current call or if the default value is to be used, the parameter value can be omitted, but its delimiting commas must be present.

Example:

```
MACROX  parm1,parm2,parm3,parm4
```

could appear as

```
MACROX  parm1,,parm3,
```

if parameters 2 and 4 are to have default values.

TABLE 12-1. STATE PROGRAM MACROINSTRUCTIONS

Name	Function	Parameters
STATUS OF ASSIGNABLE COUNTERS		
INTCC	Initialize character counters (CC)	COUNT, ACTION
INTCC1	Initialize CC1 with packet size	ACTION
INTCC2	Initialize CC2 with maximum block length	ACTION
SETCC	Set CC to value (CV)	COUNT, CV
SETCC1	Set CC1 to CV	CV
SETCC2	Set CC2 to CV	CV
CHRCC	Mask and set CC	COUNT, IMASK
CHRCC1	Set CC1	IMASK
CHRCC2	Set CC2	IMASK
MOICC	Set CC with modulus function. Modulus = CV	COUNT, CV
ICC	Increment CC	COUNT, ACTION
ICC1	Increment CC1	ACTION
ICC2	Increment CC2	ACTION
DCC	Decrement CC	COUNT, LABEL, ACTION
DCC1	Decrement CC1	LABEL, ACTION
DCC2	Decrement CC2	LABEL, ACTION
CNTNE	Compare CC with value (CV)	COUNT, CV, LABEL
CNT1NE	Use count 1	CV, LABEL
CNT2NE	Use count 2	CV, LABEL
BLCNE	Compare CC to block length	COUNT, LABEL
BLC1NE	Use count 1	LABEL
BLC2NE	Use count 2	LABEL
STORC	Store CC in destination buffer	COUNT, ACTION
STORC1	Use count 1	ACTION
STORC2	Use count 2	ACTION

TABLE 12-1. STATE PROGRAM MACROINSTRUCTIONS (Contd)

Name	Function	Parameters
CHARACTER MANIPULATION		
STORE	Store current character in destination buffer with or without CRC	CRCA
RCHAR	Make specified character the current (untranslated) character	CHAR, ACTION
RPLACE	Make specified character the current character, store it (combines RCHAR and STORE)	CHAR, CRCA
ADDC	Insert (add) character to destination buffer	CHAR, ACTION
RADDC	Add CHAR to destination buffer the number of times specified in count 1	CHAR
CHRPT	Add current character to destination buffer the number of times specified in count 1	none
INDEX MANIPULATION		
MSTATE	Set modem state index in port table to value (STATE)	STATE, ACTION
MJUMP	MSTATE, then execute indexed program	STATE
STATE	Set input index in MLCB to value (STATE) or set TP index in TPCB to value	STATE, ACTION
RTRN	Execute currently indexed input or TP state programs	none
JUMP	Optionally update state index, then execute indexed input or TP state program	STATE, RTN
SKIPS		
SKIP	Skip forward to LABEL	LABEL
SKIPB	Skip backward to LABEL	LABEL
CRCEQ	Skip to LABEL if CRC check is good	SB, LABEL
STATLS	Skip to LABEL if current input/TP state index < LABEL	STATE, LABEL

TABLE 12-1. STATE PROGRAM MACROINSTRUCTIONS (Contd)

Name	Function	Parameters
MSTLS	Skip to LABEL if current modem state index < LABEL	STATE, LABEL
CHARNE	Skip to LABEL if current character ≠ CHAR	CHAR, LABEL
SPCHEQ	Perform ACTION if current character ≠ special character, skip to LABEL otherwise (special character in control block)	LABEL, ACTION
CHARLS	Skip to LABEL if current character < CHAR	CHAR, LABEL
<u>CLA STATUS HANDLING</u>		
TSTCLA	Check unmasked CLA status bits, skip to LABEL unless bits match. Use AND function	CMASK, LABEL
CMPCLA	Same as TSTCLA but use exclusive OR function	CMASK, LABEL
<u>FLAG CONTROL</u>		
SETRAN	Set translate flag	ACTION
RSTRAN	Reset translate flag	ACTION
SETINP	Set message in process flag	ACTION
RSTINP	Reset message in process flag	ACTION
SETMXF	Set specified flags	MFLAGS, ACTION
RSTMXF	Reset specified flags	MFLAGS, ACTION
TSTMXF	Skip to LABEL if any of MFLAGS is set	MFLAGS, LABEL
SETFLG	Set flags in destination buffer	MFLAGS, BUFFER, ACTION
SETPAR	Set parity flag in control block (strips parity from subsequent current characters)	ACTION
RSTPAR	Reset parity flag	ACTION

TABLE 12-1. STATE PROGRAM MACROINSTRUCTIONS (Contd)

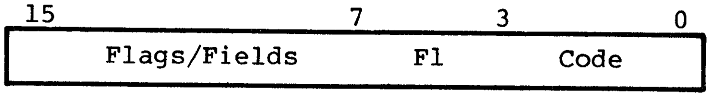
Name	Function	Parameters
<u>WORKLIST HANDLING</u>		
TIBWL	Terminate input buffer, build a worklist entry (WLE) for TIP	WC, WL, EOT, ACTION, EP
TIBSWC	Terminate input buffer, save workcode (WC) in MLCB	WC, EOT, ACTION
BLDWL	Build WLE for OPS or multiplex level	WC, WL, ACTION, EP
BLD01	Generate CLA status WLE for multiplex level 2	SCI, ACTION
<u>TEXT PROCESSOR OPERATIONS</u>		
TPADDR	(SF1R+DF1R) → DF1R. F1R is a file 1 register, S is source, D is destination	SD, DD
TPSUBR	(DF1R-SF1R) → DF1R	SD, DD
TPCMPR	SF1R < DF1R, execute P+1 instruction SF1R = DF1R, execute P+2 instruction SF1R > DF1R, execute P+3 instruction	SD, DD
TPINCR	Increment specified F1R by VALUE	SD, VALUE
TPDECR	Decrement specified F1R by VALUE	SD, VALUE
TPMARK	Mark (save processing parameters) source and destination buffers at level (LV)	LV
TPBKUP	Return to the specified buffers at level	LV, SRC, DST
TPSTLC	Store left byte of F1R (SD) into destination buffer (with or without CRC check)	SD, CRCA
TPSTRC	Store right byte of F1R	SD, CRCA
TPRSTL	Restore untranslated character registers from F1R, left byte	SD
TPRSTR	Restore untranslated character register from F1R, right byte	SD
TPEXIT	Exit from TP state program to OPS level	none

TABLE 12-1. STATE PROGRAM MACROINSTRUCTIONS (Contd)

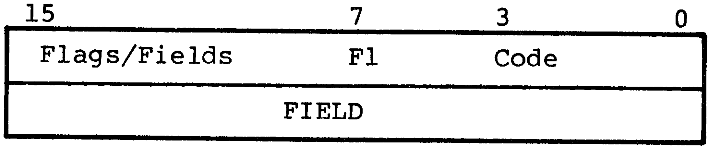
Name	Function	Parameters
MISCELLANEOUS		
STRNTB	Store translation table address in control block	TA, ACTION
RSTIME	Reset line control timer value (TIME); is a function of line type	TIME, ACTION
BKSPAC	Backspace destination buffer pointer one word	none
RESYNC	Send resync command to CLA	ACTION
ICRC	Initialize CRC	ICRC, ACTION
ALNBUF	Allocate and initialize a buffer	FCD, ACTION
NOPR	Specify ACTION parameter	ACTION
TPMOVE	Move SF1R contents to DF1R	SD, DD
TPST	Move SF1R to specified CB word	SD, DD
TPSTR	Move right byte of SF1R to specified CB word	SD, DD
TPSTL	Move left byte of SF1R to specified CB word	SD, DD
TPLD	Move specified CB word to DF1R	SD, DD
TPLDR	Move right byte of specified CB word to DF1R	SD, DD
TPLDL	Move left byte of specified CB word to DF1R	SD, DD
SBLC	Adjust block length count and then store new count in CB	ADJ, ACTION

The number of parameters varies. Macroinstructions are represented in either a one word or a two word instruction (parameter list). The usual word oriented format is as follows:

One-word



Two-word



Flags - usually in bits 14 and 15

Fl - a set of frequently used parameters, including ACTION, a parameter that specifies the actions to take prior to exiting from the instruction sequence

Code - the instruction ID (index): 00 CODE $1F_{16}$

Field - any additional control or address field

Each code can have several variations, defined by use of flags and fields.

NOTE

Flags, Fields, and Fl are all parameters. The order of the parameters in the call is not usually the same as the packed order in the instruction words.

For a detailed description of the macroinstructions, refer to the State Programming Reference Manual (see preface).

GLOSSARY

ADDRESS - A location of data (as in the macro or micro NPU memory) or of a device (as a peripheral device or terminal). The NPU main memory is paged.

APL - A specific programming language characterized by powerful operations defined as simple keyboard symbols.

APPLICATION PROGRAM - A program resident in a host computer. The program provides an information storage, a retrieval, and/or processing service to a remote user via the data communication network and the Network Access Method.

A/Q CHANNEL - The internal data channel of the 255X NPU. Peripheral devices located on the A/Q channel ordinarily use the A register for data and status transfers and the Q register for command and addressing information.

ASYNC PROTOCOL - The protocol used by asynchronous, teletypewriter-like devices. For CCP, the protocol is actually the set of protocols for eight types of real terminals. The NPU and terminal interface is handled by the ASYNC TIP.

AUTORECOGNITION - A capability for selected terminals which allows the TIP to recognize some device characteristics for the terminal, rather than having the terminal or the host specify the information.

BANDWIDTH - For CCP, bandwidth indicates the transfer rate (in characters per second) between the NPU and the terminal.

BASE SYSTEM SOFTWARE - The relatively invariant set of programs in CCP that supply the monitor, timing, interrupt handling, and multiplexing functions for the NPU. Base software also includes common areas and debugging utilities.

BLOCK - A unit of information used by networks. A block consists of four or more 8-bit characters and contains sufficient information to identify the type of block, its origin, destination, and routing. Differing block protocols apply to the host/NPU and the NPU/terminal interfaces.

BLOCK PROTOCOL - The protocol governing block transfers of information between the host and the NPU.

BREAK - An element of a protocol indicating an interruption in the data stream.

BROADCAST MESSAGE - A message generated by the system or by an operator using the systems. The message is sent to one (broadcast one) or all of the terminals in the system (broadcast all).

BUFFER - A collection of data in contiguous words. CCP assigns one size of buffers for data and two other sizes of buffers for internal processing. A buffer usually has a header of one or more words. Data within a data buffer is delimited by pointers to the first and last characters (data buffers are character oriented). If the data cannot all fit into one buffer, an additional buffer is assigned and is chained to the current buffer. Buffer assignment continues until the entire message is contained in the chain of buffers. Buffers are chained together only in the forward direction.

BUFFERING - The process of collecting data together in buffers. Filled buffers include the case where data is terminated before the end of the buffer and the remaining space is filled with extraneous information.

BUFFER THRESHOLD - The minimum number of buffers available for assignment to new tasks. As the buffer level falls toward the threshold, new tasks are rejected (regulation).

BVT - Batch Virtual Terminal. See virtual terminal.

BYTE - A group of contiguous bits. For data handling within the NPU/host interface, a byte is 8 bits, usually in the form of a 7-bit ASCII character with the eighth bit reserved for parity.

CASSETTE - The magnetic tape device in an NPU used for bootstrap loading of off-line diagnostics and (in remote NPUs) the bootstrap load and dump operation.

CCP - Communications Control Program. This set of modules performs the tasks delegated to the NPU in the network message processing system.

CE ERROR MESSAGE - A diagnostic message sent upline to the host from the NPU. The message contains information concerning hardware and/or software malfunctions.

CHARACTER - A coded byte of data. In the CCP program, a character is ordinarily in 8-bit ASCII format (7 bits plus an eighth bit reserved for parity).

CIB - Circular Input Buffer. The fixed buffer used by the multiplex subsystem to collect all data passing upline from the multiplexer. The buffer is controlled by a put pointer for the multiplexer and a get pointer used to demultiplex data to individual line-oriented data buffers.

COMMAND DRIVER - The base system program (PMCDRV) that controls the multiplex subsystem.

COMMON AREA - Areas of main memory dedicated to system and global data. These are usually below address $1D50_{16}$.

CONFIGURATION - See System Configuration.

CONNECTION NUMBER (CN) - A number specifying the path used to connect the terminal through the NPU to the host. For each NPU/host pair, there are 255 available connection numbers.

CONSOLE - (1) A terminal devoted to network control processing. There are three such terminals: the Network Operator's (NOP) terminal, the Local Operator's (LOP) terminal, and the NPU console. (2) Any standard interactive device on a terminal.

CONTENTION - (1) The state that exists in a bidirectional transmission line when both ends of the line try to use the line for transmission at the same time. Most protocols contain logic to resolve the contention situation. (2) The situation that exists when an interruptable program and the program that may interrupt it share data elements.

CONTROL BLOCKS - (1) The types of blocks used to transmit control (as opposed to data) information; (2) Data structures assigned for special configuration/status purposes in the NPU. The major control blocks are line control blocks (LCB), logical link control blocks (LLCB), terminal control blocks (TCB), queue control blocks (QCB), buffer maintenance control blocks (BCB), multiplex line control blocks (MLCB), text processor control blocks (TPCB), and diagnostics control blocks (DCB).

COUPLER - The hardware interface between the local NPU and the host. Transmissions across the coupler use block protocol.

CRC - Cyclic redundancy check. A check code transmitted with blocks/frames of data. It is used by several protocols including the HASP, Mode 4, BSC and CDCCP protocols.

CROSS - The software support system for CCP. It supports PASCAL coding and is run on the host computer. One output is a CCP program in 255X Machine Code format ready for execution in the NPU.

COMMUNICATIONS SUPERVISOR (CS) - A portion of the network software resident in the host. CS is written as an application program; the Communications Supervisor coordinates the network-oriented activities of the host computer and of the lines and terminals logically linked to it.

DATA - Information processed by the network or some components of the network. Data usually has the form of messages, but commands and status are frequently transmitted using the same information packets as data (for instance, system messages).

DATA COMPRESSION - The technique of transmitting a sequence of identical characters as a control character and a number representing the length of the sequence. HASP and Mode 4 protocols support data compression, as do virtual terminal formats.

DATA SET - A hardware interface that transforms analog data to digital data and the converse. A data set is used to connect a remotely located terminal to the NPU.

DDLTs - Special diagnostic documentation that uses a highly structured table technique to aid the troubleshooter in isolating a problem.

DEBUGGING - The process of running a program to rid it of anomalies. CCP supplies debugging aids for programs (TUP, PBTIPDG, and PBDEBUG) and for run-time PASCAL programs (QDEBUG and its associated programs).

DIAGNOSTICS - Software programs or combinations of programs and table that aid the troubleshooter in isolating problems.

DIRECT CALLS - The method of passing control directly from one program to another. This is the usual control transfer mode for CCP. Some CCP calls are indirect, through the monitor. Such OPS-level indirect calls pass information to the called program through parameter areas called worklists. See Worklist.

DIRECTORIES - Table in CCP that contain information used to route blocks to the proper interface and line. There are directories for source and destination node (SN and DN) and for connection number (CN). A routed message is attached to the TCB for the line over which the message will pass.

DN - Destination node. The network node to which a message is directed; for instance, the DN of an upline message might be the host process (CS) which processes line-related service messages.

DMA - Direct memory access. The high-speed I/O channel to the NPU main memory. This channel is used by the coupler for host/NPU buffered transfers and by the multiplex hardware (MLIA) for line-NPU transfers.

DOWNLINE - The direction of output information flow, from host to terminal or NPU.

DUMP - The process of transferring the contents of the NPU main memory, registers and file 1 registers to the host. The dump can be processed by the Network Dump Analyzer in the host to produce a listing of the dumped hexadecimal information.

EXTERNAL BCD - A type of binary-coded decimal (BCD) code used by some asynchronous and Mode 4 terminals.

FE - Format Effectors. Control symbols used by certain protocols (for instance, the IVT protocol).

FILE REGISTERS - The two sets of microregisters (file 1 and file 2) in the NPU. File 1 registers contain parameter information that is reloaded whenever the NPU is initialized. Microprograms using these registers can also change values in them. File 2 registers are invariant firmware registers that come preprogrammed with the NPU.

FRAMES - (1) The basic communications unit used in the HDLC or CDCCP protocol for trunk (NPU to NPU) communications. Frames are composed of control bytes, a CRC sum, and (in some cases) data bytes in subblock sequence. A subblock may be a block protocol block or a part of a block. Frames are transmitted as a sequence of bytes through the multiplex subsystem. (2) A sequence of data bytes used internally by the multiplex subsystem hardware (see Line Frame).

FREEWHEELING - A terminal that can input information at the discretion of the user. Input rate cannot be controlled directly.

FRONT END - A computer that performs network communications functions (such as terminal multiplexing) for a host computer. The local NPU is a front end for a CYBER host.

FULL DUPLEX (FDX) - A transmission mode allowing data transfer in both directions at the same time.

FUNCTION CODE - Code used by the service module to designate the type of function (command or status) being transmitted. Two codes are defined: Primary function code (PFC) and secondary function code (SFC). See appendix C.

GLOBAL VARIABLES - Variables that are defined for use throughout CCP. Contrast global variables with local variables, which are identified only within a single program.

HALT CODE - Code generated by the NPU when it executes a soft stop. A halt indicates the cause of the stoppage; it is delivered at the NPU console in the form of a halt message.

HALF DUPLEX (HDX) - A transmission mode allowing data transfer in one direction at a time. Normally, a single set of data lines carry input, output, and part of the control information. Contention for use is possible in HDX mode and must be resolved by the protocol governing line transfers.

HASP - Houston Automatic Spooling Process is a protocol used by the HASP workstations. The standard code of a HASP workstation is EBCDIC. The HASP TIP in the NPU processes the HASP protocol and normally performs EBCDIC/ASCII conversions since the host uses ASCII in IVT or BVT format for its processing.

HEADER - A word or set of words at the beginning of a block, record, file, or buffer which contain control information for that unit of data.

HIP - Host Interface Package. The CCP program that handles block transfers across the host/local NPU interface. The HIP normally operates with IVT or BVT data and uses CCP block protocol.

HOST - The computer that controls the network and that contains the applications programs that process network messages.

ID - Identifiers. This can refer to ports, nodes, lines, links, or terminals. Any hardware element or connection can have an ID, normally a sequentially assigned number.

INITIALIZATION - The process of loading an NPU and optionally dumping the NPU contents. After downline loading from the host, the NPU network-oriented tables are configured by the host so that all network processors have the same IDs for all network terminals, lines, and trunks.

INPUT BUFFER - A data buffer reserved by CCP for receiving an upline message for the host. These buffers are assigned and released dynamically. Contrast with the CIB on the multiplex subsystem interface.

INTERFACE (NPU) - The set of hardware and software that permits transfers between the NPU and an external device. There are four principal interfaces: to the host (block protocol in IVT or BVT format handled by a HIP), to the peripheral devices (CDT printer protocol handled by base system software), to a neighbor NPU (CDCCP protocol handled by a LIP), and to the terminals via the multiplex subsystem (various protocols; standard protocols are handled by the Mode 4, ASYNC, and HASP TIPs).

INTERNAL PROCESSING - A group of CCP modules that provide routing capability.

INTERRUPTS - A set of hardware lines and software programs that allow external events to interrupt NPU processing. Interrupting programs are allowing preferential processing on a priority basis. The lowest priority level is processed by the OPS monitor.

IVT - Interactive virtual terminal. A block protocol format for interactive terminals. See Virtual Terminals.

LCB - Line control block. A table assigned to each active line in the system. It contains configuration information as well as current processing information.

LINE - A connection between an NPU and a terminal.

LINE FRAME - A sequence of data bytes used within the multiplex subsystem as the means to transfer data and status in both directions between the CLA and the MLIA.

LINK - A connection between two NPUs or an NPU and a host. In this release (CCP3) a link is the same as a trunk.

LIP - Link interface package. The CCP program that handles frame transfers across a trunk; that is, across the connection between a local and a remote NPU. A LIP uses CDCCP protocol and interfaces on the local NPU side to the HIP. On the remote NPU side, the LIP interfaces with the appropriate TIP. In both local and remote NPUs, the LIP interfaces with the multiplex subsystem for transfers across the trunk.

LLCB - Logical link control block. A table assigned to each logical link in the system, which includes this NPU. The table contains configuration information as well as current processing information. A logical link is an association between a pair of nodes in the network.

LOAD - The processing of moving programs downline from the host and storing them in the NPU main memory and micromemory. Loading of a remote NPU is accomplished by the host through the use of overlays in the local NPU.

LOCAL NPU - An NPU that is connected to the host via a coupler. A local NPU always contains a HIP for processing block protocol transfers across the host/local NPU interface.

LOGICAL CONNECTION - A logical message path established between two application programs or between a network terminal and an application program. Until terminated, the logical connection allows messages to pass between the two entities. Not all logical connections are used (for instance, a remote NPU may be actively connected to local NPU1 rather than local NPU2; however, if NPU1 fails, the potential logical connection to NPU2 becomes an active connection and traffic is routed to the host via NPU2).

LOGICAL LINK - See Link.

LOGICAL REQUEST PACKET (LRP) - A parameter/data packet for a peripheral device. The LRP attached to a real peripheral control block is transformed to a physical request packet and is delivered to the assigned NPU console device.

LOCAL OPERATOR (LOP) - The operator of that terminal in the network that is connecting a specific application program in the host to the messages being processed. The terminal by default is the host system console, but the LOP can be transferred to any other interactive terminal in the network other than an NPU console. The operator manages the communications elements of the network within the local computer system by communicating with the Communications Supervisor in the host computer. Contrast with network operator. The local operator is an administrative operator within the network and need not be the host computer's operating system operator.

LOOP MULTIPLEXER (LM) - The hardware that interfaces the CLAs (which convert data between bit serial digital and bit parallel digital (character format) and the input and output loops of the MLIA).

LPC - Longitudinal parity character. A form of check character which is formed by exclusive OR of all the preceding characters. It is used by the Mode 4 and ASCII BSC protocols.

MAIN MEMORY - The macromemory of the NPU. It is partly dedicated to programs and common areas; the remainder is buffer area used for data and overlay programs. Word size is 16 data bits plus two additional bits for parity and program protection. Memory is packaged in 16K and 32K word increments.

MASK REGISTER - A register used in the interrupt subsystem to determine whether an interrupt is of sufficiently high priority to be processed now. Each bit in the mask register (M) corresponds to an interrupt line. The register operates under program control.

MESSAGE - A logical unit of information, as processed by an application program. When transmitted over a network, a message can consist of one or more physical blocks.

MODE 4 - A communications line transmission protocol for synchronous terminals. The protocol requires the polling of sources for input to the data communications network. CCP supports Mode 4A, 4B and Mode 4C equipment. Mode 4A equipment is polled through a single hardware address (usually that of the console device), regardless of how many devices use the address as the point of interface to the network. Mode 4C equipment is polled through several hardware addresses, depending on the point each device uses to interface with the network. The Mode 4 TIP processes the interface between the NPU and the Mode 4 terminals.

MODEM - A hardware device for converting analog levels to digital signals and the converse. Long lines interface to digital equipment via modems. Modem is synonymous with data set. The term modem is derived from modulator-demodulator.

MICROMEMORY - The micro portion of the NPU memory. This consists of 2048 words of 64-bit length. 1024 words are read-only memory (ROM); the remaining 1024 words are random access memory (RAM) and are alterable. The ROM memory contains the emulator microprogram that allows use of assembly language.

MICROPROCESSOR - The portion of the NPU that processes the programs.

MLIA - Multiplex loop interface adapter. The hardware portion of the multiplex subsystem that controls the multiplex loops (input and output).

MODULE - See program.

MONITOR - The portion of the NPU base system software responsible for time and space allocation within the computer. The principal monitor program is PBMON (commonly known as OPSMON) which executes OPS-level programs by scanning a table of programs that has pending tasks (worklist entries).

MULTILEAVING - Interleaving data from various devices in a single transmission block. It is used by the HASP protocol.

MULTIPLEX SUBSYSTEM - The portion of the base NPU software that performs multiplexing tasks for upline and downline data and also demultiplexes upline data from the CIB and places the data in line-oriented input data buffers.

NAM - See Network Access Method.

NEIGHBOR NPUs - Two NPUs connected to one another by means of a trunk. The NPU connected to the host via a coupler is designated as the local NPU. The other NPU is a remote NPU; it is not connected directly to the host in any fashion.

NETWORK - An interconnected set of network elements consisting of a host, one or more NPUs, and terminals.

NETWORK ACCESS METHOD (NAM) - A software package that provides a generalized method of using a communications network for switching, buffering, queueing, and transmission of data. NAM resides in the host.

NETWORK DEFINITION LANGUAGE (NDL) - The compiler-level language that defines the network configuration file and local configuration file contents used by the host computer.

NETWORK LOGICAL ADDRESS - The address used by block protocol to establish routing for the message. It consists of three parts; DN (the destination node), SN (the source node) and CN (the connection number).

NETWORK OPERATOR (NOP) - An administrative operator at the network operator console. This terminal by default is the host console, but the NOP function can be assigned to any other terminal in the system except an NPU console. The network operator manages the NPU hardware, linkages, and other network elements of the entire data communications network by communicating with the Network Supervisor at the host computer. Contrast with local operator. The network operator can also be a local operator, but need not be the operating system operator for the host computer at the network control center.

NETWORK PROCESSING UNIT (NPU) - The collection of 255X hardware and peripherals together with the Communications Control Program (CCP). The CCP program buffer and transmit data between terminals and host computer.

NETWORK SUPERVISOR (NS) - A portion of the network software, which coordinates all of the NPUs in the communications network. NS is written as an application program and resides in the host.

NODE - A network element that creates, absorbs, switches, and/or buffers message blocks. Typical system nodes are NS and CS in the host, the coupler node of a local NPU, and a terminal node of a remote NPU.

OFF-LINE DIAGNOSTICS - Optional diagnostics for the NPU that require the NPU be disconnected from the network.

ON-LINE DIAGNOSTICS - Optional diagnostics for the NPU that can be executed while the NPU is connected to, and operating as a part of, the network. Individual lines being tested must, however, be disconnected from the network. These diagnostics are provided if the user purchases a network maintenance contract.

OPS MONITOR - The NPU monitor (see Monitor).

OUTPUT BUFFER - Any buffer that is used to output information from the NPU to another NPU, to a peripheral device, or to a terminal via the multiplex subsystem.

OVERLAY AREA - A reserved area in main memory that is used to execute overlay programs.

OVERLAY PROGRAMS - Programs that are not normally resident in main memory but which are called into the overlay area of main memory to execute special tasks. These programs are loaded by means of service messages from the host and perform such tasks as NPU initialization, debugging, loading/dumping a remote NPU, and on-line diagnostics.

PAGING (NPU) - A method of executing programs and accessing data in the NPU main memory region above 65K. Paging is required to allow addressing where the address is larger than 16 bits (NPU word size) in length.

PAGING (Screen) - The process of filling a CRT display with data while holding additional data for subsequent displays. Changing the paged display is a terminal operator controlled function.

PARITY - A bit-oriented data assurance method. Parity in the NPU memory is word-oriented and is ordinarily not controlled by the operator. Parity bit is added when words are stored in main memory; parity bit is discarded after checking when the word is read from main memory. A parity error causes the highest priority interrupt in the system. Parity bits are also associated with ASCII characters (bit 7) and with some synchronous protocols (example: LPC, the longitudinal parity character).

PASCAL - A high-level programming language used for CCP programs. Most CCP programs are written in PASCAL language.

PFC - Primary function code. See Function Code.

PHYSICAL LINK - A connection between two major network nodes such as neighboring nodes. Messages can be transmitted over active physical links.

PHYSICAL REQUEST PACKET (PRP) - A packet of data to or from a peripheral device. Data in PRP format is ready to be processed by the peripheral device handler. A logical request packet (which see) must be converted into a PRP before sending output to the device.

POINT OF INTERFACE (POI) PROGRAMS - A special set of base system programs that interface directly with TIPs. POIs are defined for such standard functions as ending an output operation or ending an input operation.

POLLING - (1) The action of checking CLAs to find whether a port is ready to transmit or receive another word of data. The multiplex subsystem performs the polling operation for active lines. (2) The action of soliciting input from certain types of terminals. A poll message is output to the terminal. The response is input device status or an indication that no data is ready to be input.

PORT (P) - The physical connection in the NPU through which data is transferred to or from the NPU. Each port is numbered and supports a single line. Subports are possible but not used in this version of CCP.

PPU - Peripheral processor unit. The part of the host dedicated to performing I/O transfers. The coupler connects the PPU to an NPU via a data channel.

PRIORITY LEVEL - CCP uses 16 interrupt processing levels plus the OPS processing level. Priority levels are interrupt driven. The OPS monitor processes at the lowest priority level; that is, at a level below any interrupt driven level.

PROGRAM - A series of instructions that are executed by a computer to perform a task; usually synonymous to a module. A program can be composed of several subprograms.

PROTECT SYSTEM - A method of prohibiting one set of programs (unprotected) from accessing another set of programs (protected) and their associated data. The system uses a protect bit in each main memory word.

PROTOCOL - The complete set of rules used to transmit data between two nodes. This includes format of the data and commands, and the sequence of commands needed to prepare the devices to send and receive data. CCP uses the following protocols: The block protocol, the Logical Link protocol, the coupler protocol, and various terminal protocols.

QUEUES - Sequences of blocks, buffers, or messages. Most NPU queues are maintained by leaving the queued elements in place and using a combination of tables of pointers to the next queued element and pointer words within the queued elements. Most queues operate on a first-in first-out basis. A series of worklist entries for a specific terminal is an example of an NPU queue.

RECORD - For CCP: A data unit defined for the host software or for HASP workstations and HASP transmission. A HASP record contains space for at least one character of data and normally has a header associated with it. Records for HASP may be composed of subrecords.

REGULATION - The process of making an NPU or a host progressively less available to accept various classes of input messages. The host has one regulation scheme, the multiplex interface of a local NPU has another scheme, and the multiplex interface to a neighbor NPU has a third regulation scheme. Some types of terminals (for instance, HASP workstations) can also regulate messages. Message classifications are usually based on batch, interactive, and control message criteria.

REMOTE NPU - An NPU connected only to other (local) NPUs. Since a remote NPU has no coupler, it cannot be directly connected to the host.

RESPONSE MESSAGES - A subclass of service (network control) messages directed to the host that are normally generated to respond to a service message from the host. Response messages normally contain the requested information or indicate the requested task has been started or performed. Error responses are sent when the NPU cannot deliver the information or start the task. A class of unsolicited response messages are generated by the NPU to report hardware failures.

ROUTING - The process of sending data or commands through the NPU to the internal NPU process or to an external device (for instance, a terminal). The network logical address (DN, SN, CN) is the primary criterion for routing. The NPU directories are used to accomplish the routing function.

SERVICE CHANNEL - The network logical link used for service message transmission. For this channel, CN=0. The channel is always configured, even at load time.

SERVICE MESSAGE (SM) - The network method of transmitting most command and status information to or from the NPU. Service messages use CMD blocks in the block protocol.

SERVICE MODULE (SVM) - The set of NPU programs responsible for processing most service messages. SVM is a part of the network communications software.

SFC - Secondary function code. See Function Codes.

SOURCE NODE (SN) - The network node originating a message or block of information.

STATE PROGRAMS - Programs in the multiplex subsystem whose execution depends on the current state of the message being transmitted. For example, one state program is executed at the start of the message header processing, and another at start of text processing, another at end-of-text processing.

STATE PROGRAM TABLES - Tables used by the multiplex subsystem to locate the next state program to execute.

STATISTICS SERVICE MESSAGE - A subclass of service messages that contain detailed information about the characteristics and history of an element such as a line or a terminal.

STATUS - Information relating to the current state of an equipment, device or line. Service messages are the principal carriers of status information. Statistics are a special subclass of status.

STRINGS - A unit of information transmission used by the HASP protocol. One or more strings compose a record. A string can be composed of different characters or it can be a string of contiguous identical characters. In the latter case, the string is normally compressed to a single character (the only one type in the string) and a value indicating the number of times the character occurs.

SUBPROGRAM - A series of instructions that are executed by a computer to perform a task or part of a task. A subprogram can be called by several programs or can be unique to a single program. Subprograms are normally reached by a direct call from a program.

SUPERVISORY MESSAGE - A message block in the host not directly involved with the transmission of data but which provides information for establishing and maintaining an environment for the communication of data between the application program and NAM, and through the network to a destination or from a source. Supervisory messages can be transmitted to an NPU in the form of a service message.

SWITCHING - The process of routing a message or block to the specified internal program or external destination.

SYNCHRONOUS PROTOCOLS - A class of protocols which require that characters be transmitted in contiguous blocks. Synchronization for the entire block transmission is established at the beginning of the block. Synchronous Protocols include Mode 4, BSC, HASP, HDLC, and CDCCP.

SYSTEM CONFIGURATION - The process of setting tables and variables throughout the network to assign NPUs, lines, links, terminals, and devices so that all elements of the network recognize a uniform addressing scheme. After configuration, all network elements accept all data commands directed to or through themselves and reject all other data and commands.

TERMINAL - An element connected to a network by means of a communications line. Terminals supply input messages to, and/or accept output messages from, an application program. A terminal can be a separately addressable device comprising a physical terminal or station, or the collection of all devices with a common address.

TERMINAL CONTROL BLOCK (TCB) - A control block containing configuration and status information for an active terminal. It is dynamically assigned.

TERMINAL INTERFACE PACKAGES (TIPs) - NPU programs that provide the interface between real terminal format and virtual terminal format. The standard TIPs are the ASYNC TIP, the Mode 4 TIP, and the HASP TIP. TIPs are responsible for some data conversion and for error case processing.

TIMEOUT - The process of setting a time for completion of an operation and entering an error processing condition if the operation has not finished in the allotted time.

TIMING SERVICES - The subset of base system programs that provide timeout processing and clock times (examples: messages or status). Timing services provide the drivers for the real-time clock.

TRUNK - A line connecting two NPUs or an NPU and a host. The host/NPU trunk uses block protocol; the NPU/NPU trunk uses trunk protocol.

TRUNK PROTOCOL - The protocol used for communicating between neighboring NPUs. It is a modified CDCCP protocol that uses the frame as the basic communications element.

TUP - Test Utility Package. A debugging utility that supports breakpoint debugging as well as other utility type operations such as loading and dumping.

UNSOLICITED SERVICE MESSAGES - Service messages sent to the host that do not respond to a previous service message from the host. Unsolicited SMs report hardware or software failures to the host.

UPLINE - The direction of message travel from a terminal through an NPU to the host.

VIRTUAL TERMINAL - A software concept for CCP that converts all types of upline messages to one of two formats: Batch virtual terminal (BVT) or interactive virtual terminal (IVT). By this method, application programs in the host need only to be able to process data in IVT or BVT format rather than in the multiplicity of formats that real terminals use. Downline messages from the host to real terminals are converted from IVT or BVT to real terminal format. The IVT/BVT processors are a part of the NPU's network communications software.

WORD - The basic storage and processing element of a computer. The NPU uses 16-bit words (main memory) and 64-bit words (internal to the microprocessor only). All interfaces are 16-bit word (DMA and A/Q) or in character format (multiplex loop interface). Characters are stored in main memory, two per word. Hosts (CYBER series) use 60-bit words internally, but a 12-bit byte at the interface to the NPU. Characters at the host side of the NPU/host interface are stored in bits 19 through 12 and 7 through 0 of a dual 12-bit byte.

Interfacing intelligent terminals, such as a HASP workstation, can use any word size but must communicate to the NPU in character format. Therefore, workstation word size is transparent to the NPU.

WORKLISTS - Packets of information containing the parameters for a task to be performed. Programs use worklists to request tasks of OPS level programs. Worklist entries are queued to the called program. Entries are one to six words long and a given program always has entries of the same size. Worklists are also used on the multiplex (priority) level.

WORKLIST PROCESSOR - (1) Any system program that receives and processes. (2) The program within the multiplex subsystem that handles worklist entries generated by the multiplex firmware (PMWOLP).

CCP MNEMONICS

ACK0	Acknowledge block (various protocols)
ACN	Application connection number
ACTL	Assurance control block
A/Q	The A/Q (internal) I/O channel of the NPU
APL	A Programming Language
ARM	Asynchronous response mode
ASCII	American Standard Code for Information Interchange
ASYNC	Asynchronous
BACK	Acknowledgment block (element of block protocol)
BCB	1. Buffer control block 2. Block control byte (HASP protocol)
BCD	Binary coded decimal
BFC	Block flow control
BFR	Buffer
BLK	Message block (element of block protocol)
BN	Block number (overlay)
BP	Breakpoint
BRK	Break (element of block protocol)
BSC	Binary synchronous communications (protocol)
BSN	Block serial number (for blocks/SVM)
BT	Block type
BVT	Batch virtual terminal format
B1, B2	User allowed breaks for IVT protocols
CA	Cluster address
CB	Cluster block
CDCCP	CDC communications protocol (trunk protocol)
CDT	Conversational display terminal
CCP	Communications control program in NPU
CE	Customer engineer
CFS	Configuration state (for SVM)
CIB	Circular input buffer
CLA	Communications line adapter

CLR	Clear logical line (trunk protocol)
CMD	Command block (element of block protocol)
CMDR	Command reject (trunk protocol)
CN	Connection number (for blocks/SVM)
CND	Connection number directory
CR	Carriage return
CRC	Cyclic redundancy check
CRT	Cathode ray tube (type of terminal display)
CS	Communications Supervisor program in host
CTL	Control element (ASYNC protocol)
DBC	Data block clarifier (for blocks/SVM)
DCB	Diagnostics control block
DCD	Data carrier detect (RS-232 signal name)
DDLT	Diagnostic decision logic table
DEL	Delete character
DM	Disconnect mode (trunk protocol)
DMA	Direct memory access (in NPU)
DN	Destination node (for blocks/SVM)
DND	Destination node directory
DSR	Data set ready (RS-232 signal name)
DT	Device type
EBCDIC	Extended Binary Coded Decimal Interchange Code
EC	Error code
E-CODE	Device codes (Mode 4 protocol)
ENQ	Enquiry block (HASP/BSC protocols)
EOF	End of file
EOI	End of information
EOM	End of medium
EOR	End of record
ETB	End of block (HASP/BSC protocol)
ETX	End of text
FCD	First character displacement (in buffer)
FCS	Function control sequence (HASP protocol)
FD	Forward data (block protocol)
FDX	Full duplex
FE	Format effector

FE	Front end
FF	Form feed
FN	Field number (for SVM)
FRQ	Frame retention queue (trunk protocol)
FS	Forward supervision (block protocol)
FV	Field values (service module)

HASP	Houston automatic spooling process (protocol)
HDLC	High-level data link control
HDX	Half duplex
HIP	Host interface package
HO	Host ordinal

IAF	Interactive Facility Program in host
ID	Identifier (number or code)
IDC	Internal data channel (in NPU)
I-FRAME	Information frame (trunk protocol)
INIT	Initialization block (element of block protocol)
I/O	Input/output
ISO	International Standards Organization
IVT	Interactive virtual terminal format

LBN	Last block number (overlay)
LCB	Line control block in NPU
LCD	Last character displacement (in buffer)
LCF	Local configuration file in host (CS controlled)
LD	Load or dump
LF	Line feed
LIDLE	Idle element (trunk protocol)
LINIT	Line initialization element (trunk protocol)
LIP	Link interface package in NPU
LL	Logical link
LLCB	Logical link control block in NPU
LLREG	Logical link regulation
LM	Loop multiplexer
LOP	Local operator
LP	A TUP command
LRN	Link remote node (service module)
LRP	Logical request packet (I/O) for the NPU console
LT	Line type

M	Mask register
MLCB	Multiplex line control block
MLIA	Multiplex loop interface adapter
MPLINK	The PASCAL Linking Editor
MSG	Message block (element of the block protocol)
MTI	Message type indicators (Mode 4 protocol)
NAK	Negative acknowledgment block (HASP/BSC protocol)
NAM	Network Access Method program in host
NCF	Network configuration file in host (NS controlled)
NDA	Network dump analyzer (in host)
NDLP	Network Definition Language Processor in host
NHP	Network host products
NIP	Network Interface program
NOP	Network operator
NPINTAB	CCP Data structure that contains initialization status
NPU	Network Processor Unit
NS	Network Supervisor program in host
NVF	Network Validation Facility in host
OBT	Output buffer transmitted (information from multiplex subsystem to user)
ODD	Output data demand (multiplex subsystem microinterrupt)
OPS	Operations (OPS level = Monitor level programs)
OPSMON	Monitor in CCP (PBMON)
P	Priority
P	Port
PAD	Padding element (synchronous protocols)
PFC	Primary function code (for SVM)
PL	Page length (IVT)
POI	Point of interface (class of CCP programs)
PPU	Peripheral processor unit in host
PRP	Physical request packet (I/O) for the NPU console
PRST	Protocol reset (trunk protocol)
PW	Page width
QCB	Queue control block
QDEBUG	PASCAL Debugging Package

RAM	Random access memory
RBF	Remote Batch Facility program in host
RC	Reason Code (in response service messages)
RC	Remote Concentrator
RCB	Record control byte (HASP protocol)
RCV	Receive state
REGL	Regulation level
REJ	Reject (trunk protocol)
RIM	Request initialization mode (trunk protocol)
RL	Regulation level
RM	Response message (service message)
RNR	Receive not ready (trunk protocol)
RR	Receive ready (trunk protocol)
RS	Reverse supervision (block protocol)
RST	Reset block (element of block protocol)
RT	Record type
RTS	Ready to send (trunk protocol)
RTS	Request to send (RS-232 signal name)
SARM	Set asynchronous mode (trunk protocol)
SCB	String control byte (HASP protocol)
S-Frame	Supervisory frame (trunk protocol)
SFC	Secondary function code (service message)
SIM	Set initialization mode (trunk protocol)
SM	Service message
SN	Source node (for blocks/SVM)
SND	Source node directory
SPRM	System programmer's reference manual
SRCB	Subrecord control byte (HASP protocol)
STP	Stop data block (element of block protocol)
STRT	Start data block (element of block protocol)
STX	Start of text
SVM	Service module for processing service messages
SYNC	Synchronizing character (synchronous protocols)
TA	Terminal address (same as the station address used by Mode 4)
TAF	Transaction facility in host
TC	Terminal class
TCC	Trunk control character (byte) - UI frame - LIP
TCB	Terminal control block in NPU

TDP	Time Dependent Program
TIP	Terminal interface package
TIPTQ	TIP trunk queues (trunk protocol)
TO	Timeout
TOT	Total number of trunks (SM)
TPCB	Text processing control block
TT	Terminal type
TTF	Trunk transmission frame
TUP	Test utility package
TVF	Terminal Verification Facility in host
UA	Unnumbered acknowledgment (trunk protocol)
U-Frame	See UA and UI
UI	Unnumbered information frame (trunk protocol)
US	Unit separator
UT	User terminal
VAR	PASCAL keyword that marks the beginning of the variable declaration section of a PASCAL program, procedure, or function
VAR	PASCAL keyword that specifies that the parameter in a procedure or function is to be passed by name rather than by value
WACK	Wait acknowledgment block (synchronous protocols)
WL	Worklist
WLCB	Worklist control block
WLE	Worklist entry
WLP	Worklist processor
X-OFF	Stop punch character (ASYNC protocol)
X-ON	Start punch character (ASYNC protocol)
XPT	Transparent bit, paper tape (ASYNC TIP)

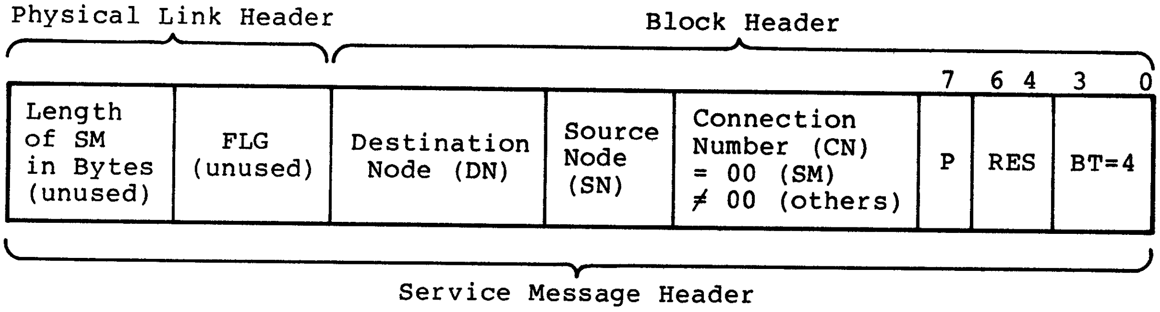
SERVICE AND COMMAND MESSAGE SUMMARY C

This appendix is divided into five parts:

- The general format for all service or command messages (SMs)
- The network SM primary and subfunction summary table
- A summary of each network SM and its normal or error response sequence
- A table of SM mnemonics
- A set of tables defining SM parameter values

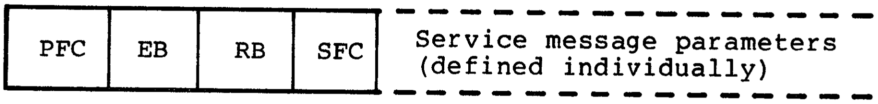
SERVICE AND COMMAND MESSAGE GENERAL FORMAT

All service or command messages described within this appendix are prefixed by the header information shown below. (This information is omitted in the individual descriptions to conserve space.) Each of the major subdivisions in the header format diagram is one 8-bit byte in length.



P - priority flag
RES - unused
BT - block type = 04 for service messages. This is a CMD block.

The general format of the service and command message body is shown below. Each of the major subdivisions in the body is also one 8-bit byte in length.



PFC - Primary Function Code
EB - 1 = Error response service message
RB - 1 = Normal response service message
SFC - Secondary function code

A0 - BF_{16} - Reserved for expansion
C0 - $E0_{16}$ - Reserved for network use
E1 - EF_{16} - Reserved for installations
00 - $3F_{16}$ - Reserved for network use
40 - $9F_{16}$ - Reserved for intrahost use

SUMMARY OF SERVICE AND COMMAND MESSAGE TYPES

Table C-1 shows the basic network SM types and the primary (PFC) and secondary (SFC) function codes associated with each type. For service messages PFC = 01 through 0C, CN = 0; for command messages PFC = C1, CN ≠ 0.

COMMANDS SENT OVER LOGICAL CONNECTIONS

The following are command blocks sent through logical connections, where connection number is not zero.

START INPUT

PFC = $C1_{16}$	SFC = 05

STOP INPUT

PFC = $C1_{16}$	SFC = 06

INPUT STOPPED

PFC = $C1_{16}$	SFC = 07	RC

RC - Reason code

00 = Stop input response
01 = Input device not ready
02 = Card slip error
03 = EOI input
04 = Batch input interrupted by interactive I/O

DEFINE CHARACTERISTICS (TERMINAL PARAMETERS) FOR IVT

PFC = $C1_{16}$	SFC = 04	String

String - defined in section 6, table 6-4. It is given in the TERMINAL PARAMETERS field of the IVT protocol.

TABLE C-1. SERVICE AND COMMAND MESSAGE TYPES

Service Message Name CN = 0	PFC (hex)	Mnemonic	SFC (hex)	Mnemonic	SVM Processing Routine
Load Request	01	D8LOAD	00	D9RQ	PNDISCARD
Force Load			01	D9FRC	PNFRCELD
NPU Initialized			02	D9INIT	PNDISCARD
Configure Logical Link	02	D8LINK	00	D9LLCNF	PNLLCNF
Delete Logical Link			01	D9LLDLT	
Configure Trunk/Line	03	D8CONFIG	00	D9LNCNF	PNLNCNF
Delete Line			01	D9LNDLT	PNDELETE
Configure Terminal (TCB)			02	D9TMLCNF	PNTMLCNF
Reconfigure Terminal (TCB)			03	D9TMLRCNF	
Delete Terminal (TCB)			04	D9TMLDLT	ANTMLDLT
Overlay Program Block	04	D8OVLOAD	00	D9OVLBLK	PNOVLOAD
Terminate Overlay			01	D9OVLTMT	PNOVLTMT
Overlay Data	05	D8OVLDATA	00	D9DATA	PNOVLDATA
Logical Link Status Request	06	D8STATUS	00	D9LLSTAT	PNLLSTAT
Trunk Status Request			01	D9TNKSTAT	PNTNKSTAT
Line Status Request			02	D9LNSTAT	PNLNSTAT
Terminal Status Request			03	D9TMLSTAT	PNTMLSTAT
Line Count Request			05	D9LCR	PNLCR
NPU Statistics	07	D8COUNTS	00	D9NPUCNTS	PNDISCARD
Trunk/Line Statistics			01	D9CNTLN	
Terminal Statistics			02	D9CNTML	
Enable Trunk/Line	08	D8LINE	00	D9ENABLE	PNENABLE
Disable Trunk/Line			01	D9DISABLE	PNDISABLE
Disconnect Trunk/Line			02	D9DISCONNECT	
CE Error	0A	DSEVENT	00	D9CE	PNDISCARD
Message to Network Operator			01	D9ALARM	
Host Broadcast One	0C	D8USER	00	D9BRD1	PN1BRDCST
Host Broadcast All			01	D9BRDCST	PNBRDCST
Operator Message			02	D9OPMSG	PNDISCARD
Terminal Characteristics			03	D9TDEF	
Service Message Name CN ≠ 0	**PFC**		**SFC**	**Remarks**	
IVT Command	C1		04	See section 6, IVT/BVT	
Start Input			05	Downline	
Stop Input			06	Downline	
Input Stopped			07	Upline	

INDIVIDUAL SERVICE MESSAGES

These messages, where the connection number is zero, are shown below.

LOAD REQUEST

PFC = 01	SFC = 00	LRN	P	00

LRN - Node ID of element to load
P - Line over which load is performed

Response

None

FORCE LOAD

PFC = 01	SFC = 01

Response

None

NPU INITIALIZED

PFC = 01	SFC = 02	CCP Version	CCP Cycle	CCP Level

Describes the current software running in the NPU

Response

None

CONFIGURE LOGICAL LINK

PFC = 02	SFC = 00	ID1	ID2	HO

ID1/ID2 - Nodes forming logical link

ID1 = Destination node
ID2 = Source node

HO - Host ordinal

Normal Response

PFC = 02	SFC = 40_{16}	ID1	ID2	HO	RC

RC - 00 = Configured

Error Response

PFC = 02	SFC = 80_{16}	ID1	ID2	HO	RC

RC - 01 = ID1 invalid
02 = Too many LLCBs
03 = LL already exists

DELETE LOGICAL LINK

PFC = 02	SFC = 01	ID1	ID2	HO

ID1/ID2 - Nodes forming logical link; ID1 to be used as the local ID at the NPU

Normal Response

PFC = 02	SFC = 41_{16}	ID1	ID2	HO	RC

RC - 00 = deleted

Error Response

PFC = 02	SFC = 81_{16}	ID1	ID2	HO	RC

SFC - Logical link does not exist

RC - 01 = ID1 invalid
02 = LLCB not configured
03 = Bad HO

CONFIGURE LINE

PFC =03	SFC =00	P	00	HO	LT	TT	FN1	FV1	...	FN_n	FV_n

P - Port
LT - Line type (see table C-3)
TT - Terminal type (see table E-2)
FN - Field number
FV - Associated field value (see table E-5)

Normal Response

The normal response is a line-enabled normal response SM.

Error Response

PFC =03	SFC= 80_{16}	P	00	HO	LT	TT	RC	FN	FV

LT - See table C-3

TT - See table C-2

RC - 01 = Invalid FN/FV
02 = Invalid line number
03 = Line control block already configured
04 = Invalid line type
05 = Invalid terminal type
06 = Diagnostics in progress

FN/FV - Pair returned if RC = 1

DELETE LINE

PFC =03	SFC =01	P	00	HO

Normal Response

PFC = 03	SFC = 41_{16}	P	00	HO	RC=00

Error Response

PFC = 03	SFC = 81_{16}	P	00	HO	RC

RC - 02 = Invalid line number
03 = Line not configured

CONFIGURE/RECONFIGURE TERMINAL

PFC =03	SFC	P	00	HO	CA	TA	DT	THO	FN1	FV1	...	FN_n	FV_n

SFC - 02 = Configure
03 = Reconfigure

DT - See table C-2

FN/FV - See table C-7

Normal Response

PFC =03	SFC	P	00	HO	CA	TA	DT	THO	RC= 00

SFC - 42_{16} = Terminal configured
43_{16} = Terminal reconfigured

DT - See table C-2

Error Response

PFC =03	SFC	P	00	HO	CA	TA	DT	THO	RC	FN	FV

SFC - 82_{16} = Configure
83_{16} = Reconfigure

DT - See table C-2

RC - 01 = Invalid FN or FV

02 = Invalid line number or terminal address

03 = Terminal already configured (configure), or not configured (reconfigure)

04 = No buffer for TCB (temporary)

05 = Invalid DT

06 = Line inoperative or not enabled

07 = HO toggle bit unchanged

08 = Logical link not established

09 = CN in use

10 = No console configured for Mode 4A cluster; cannot configure batch device

11 = Line not configured

FN/FV - Pair returned if RC = 01 or 09

DELETE TERMINAL

PFC =03	SFC =04	P	00	HO	CA	TA	DT	THO

Normal Response

PFC =03	SFC= 44_{16}	P	00	HO	CA	TA	DT	THO	RC =00

Error Response

PFC =03	SFC= 84_{16}	P	00	HO	CA	TA	DT	THO	RC

RC - 02 = Invalid line number

03 = Terminal on line not configured

04 = Cannot delete console of Mode 4A cluster while batch devices still configured

05 = HO toggle error

OVERLAY PROGRAM BLOCK

PFC =04	SFC =00	BN	LBN	Overlay ID	Checksum		. . .	

Words 1-n of overlay

Checksum - Complement of arithmetic sum of data words

Normal Response

PFC =04	SFC= 40_{16}	BN	LBN	Overlay ID	RC= 00

Error Response

PFC =04	SFC= 80_{16}	BN	LBN	Overlay ID	RC

RC - 01 = Overlay space in use
02 = Checksum error

TERMINATE OVERLAY

PFC =04	SFC =01

Response

PFC =04	SFC= 41_{16}

OVERLAY DATA (GENERAL FORM)

PFC =05	SFC =00	Overlay ID	DATA

Normal Response

PFC =05	SFC= 40_{16}	Overlay ID	DATA

Error Response

PFC =05	SFC= 80_{16}	Overlay ID	RC	Overlay ID

RC - 01 = Invalid OVID
     02 = No overlay loaded

Overlay ID - Returned if RC = 1

OVERLAY DATA (LOADING/DUMPING)

LOAD COMMAND

PFC= 05	SFC= 00	Overlay ID	1	P	00	0	BC

1 - Load

Beginning Address	Checksum	Data Words (1 - 105)

0	Register Number	Register Content	Page Displacement
23	22 18	17 11	10 0

22 - 18 - Base register address (not used)
17 - 0 - Main memory address

Response

PFC= 05	SFC= 40_{16}	Overlay ID	01	P	00	RC	0	Beginning Address

01 - Load

RC - 00 = Overlay loaded successfully
01 = Protocol error on trunk
02 = Mode error

START COMMAND

PFC= 05	SFC= 00	Overlay ID	01	P	00

02 - Start

Response

PFC= 05	SFC= 40_{16}	Overlay ID	02	P	00	RC

02 - Start

RC - 00 = Overlay started successfully
     01 = Protocol error on trunk
     02 = Mode error

DUMP COMMAND

PFC= 05	SFC= 00	Overlay ID	00	P	00	0	0

00 - Dump

Beginning Address	Ending Address

Response

PFC= 05	SFC= 40_{16}	Overlay ID	00	P	00	RC	0

00 - Dump

RC - 00 = Overlay dumped successfully
     01 = Protocol error on trunk
     02 = Mode error

Beginning Address	Data Words (1 - 105)

CLEAR COMMAND

PFC= 05	SFC= 00	Overlay ID	03	P	00

03 - Clear

Response

PFC= 05	SFC= 40_{16}	Overlay ID	03	P	00

03 - Clear

LOGICAL LINK STATUS REQUEST

PFC= 06	SFC= 00	ID1	ID2	HO

ID1/ID2 - Node IDs forming logical link; ID1 is node ID of the NPU. If ID1 and ID2 are missing, NPU returns status for all logical links supported by the NPU.

Normal Response

PFC= 06	SFC= 40_{16}	ID1	ID2	HO	RC	RL	INIT	TOT

RC - 00 = Logical link operational
01 = Logical link inoperative

RL - Regulation level (see CCP Reference Manual)

INIT - 00 = Second and subsequent responses
01 = Unsolicited response (used when NPU changes the regulation level)

TOT - Number of LL in an "all" request

Error Response

PFC= 06	SFC= 80_{16}	ID1	ID2	HO	RC

RC - 02 = Logical link not configured
     03 = Logical link status request in progress or request not from NS

NOTE

The normal response may be unsolicited (SFC = 40_{16}) or unsolicited (SFC = 00).

TRUNK STATUS REQUEST

PFC= 06	SFC= 01	P	00	HO= 00

P/00/HO - If missing, return status on all trunks

Normal Response

PFC= 06	SFC= 41_{16}	P	00	HO= 00	RC	LT	CFS	LRN	TOT

RC - 00 = Trunk operational
     04 = Trunk inoperative
     05 = No ring indicator

LT - See table C-3

CFS - See table C-4

Error Response

PFC= 06	SFC= 81_{16}	P	00	HO= 00	RC

RC - 01 = Invalid line number or no trunks configured belonging to requestor

02 = Trunk status request in progress

03 = Cannot disable last path to NS

Unsolicited Response

NOTE

Normal responses above may be sent as an unsolicited status message with SFC = 01.

LINE STATUS REQUEST

PFC= 06	SFC= 02	P	00	HO

P/00/HO - If missing, return status on all lines except trunks

Normal Response

PFC= 06	SFC= 42_{16}	P	00	HO	RC	LT	CFS	NT

RC - 00 = Line operational
04 = Line inoperative
05 = No ring indicator or autorecognition in progress

LT - See table C-3

CFS - See table C-4

Error Response

PFC= 06	SFC= 82_{16}	P	00	HO	RC

RC - 01 = Invalid line number or bad HO
02 = Line status request in progress (all lines only)
03 = Illegal configuration state (single lines only)
07 = No lines configured (all lines only)

Unsolicited Response

Only for autorecognition

PFC =06	SFC =02	P	00	HO	RC	LT	CFS	NT	TT	LS	CD	CA

TA_1	DT_1	...	TA_1	DT_1

RC - Same as other line status responses
LT - See table C-3
CFS - See table C-4
TT - See table C-2
LS - See table C-6
CD - See table C-6

For autorecognition responses, the TA DT pairs are repeated for each terminal that can be detected by the TIP. The ASYNC TIP will only report one TA DT pair. The DT may be either zero to indicate no information or four to indicate the IBM 2741. The Mode 4 TIP may report up to 15 TA DT pairs with the full range of values as shown in Table A-2 for DT. For a Mode 4A cluster, the TIP will report 3 terminals: DT=00, TC=00, DT=01, TC=00; DT=02, TC=00. Mode 4C consoles will be reported as TC=00 as it is not possible to distinguish 711 from 714.

TERMINAL STATUS REQUEST

PFC= 06	SFC= 03	P	00	HO

Normal Response

PFC 06	SFC= 43_{16}	P	SP	HO	CA	TA	DT	THO	RC	DN	SN	CN	TOT

DT - See table C-2

RC - 00 = Terminal operational
04 = Terminal inoperative

Error Response

PFC= 06	SFC= 43_{16}	P	00	HO	RC

RC - 01 = Invalid line number of bad HO
02 = No terminals configured
03 = Line inoperative or not enabled
05 = Terminal status request in progress
06 = LCB not configured

Unsolicited Response

NOTE

Normal response (see above) may be sent as an unsolicited status message with SFC = 03.

LINE COUNT REQUEST

PFC= 06	SFC= 05

Normal Response

PFC= 06	SFC= 45_{16}	NL

NPU STATISTICS (UPLINE ONLY)

PFC= 07	SFC= 00	Statistics Words

Word 1 - Service messages generated
Word 2 - Service messages processed
Word 3 - Bad service messages received
Word 4 - Blocks discarded due to bad address
Word 5 - Packets/blocks discarded due to bad format
Word 6 - Times at regulation level 4 (no regulation)
Word 7 - Times at regulation level 3
Word 8 - Times at regulation level 2
Word 9 - Times at regulation level 1
Word 10 - Times at regulation level 0
Word 11 - Network assurance protocol timeout

Response

None

TRUNK/LINE STATISTICS (UPLINE ONLY)

PFC= 07	SFC= 01	P	00	HO	LRN	00	Statistics Words 1 - 4

LRN - Trunks only; LRN = 0 for Lines

Word 1 - Blocks transmitted
Word 2 - Blocks received
Word 3 - Characters transmitted (good blocks only)
Word 4 - Characters received (good blocks only)

Response

None

TERMINAL STATISTICS (UPLINE ONLY)

PFC =07	SFC =02	P	00	HO	CA	TA	DT	THO	Statistics Words 1 - 3

DT - See table C-2

THO - Toggle HO

Word 1 - Blocks transmitted
Word 2 - Blocks received
Word 3 - Blocks in error

Response

None

ENABLE TRUNK/LINE

PFC= 08	SFC= 00	P	00	HO

Normal Response (Trunk/Line Enabled)

PFC= 08	SFC= 40_{16}	P	00	HO	RC	LT	CFS	LRN NT=0

Trunk
Line

RC - 00 = Trunk/line enabled and operational
04 = Trunk/line inoperative
05 = Line enabled; wait for ring indicator or autorecognition result

LT - See table C-3

CFS - See table C-4

Error Response (Trunk/Line Not Enabled)

PFC= 08	SFC= 80_{16}	P	00	HO	RC

RC - See trunk line status request response codes

DISABLE TRUNK/LINE

PFC= 08	SFC= 01	P	00	HO

Normal Response (Trunk/Line Disabled)

PFC= 08	SFC= 41_{16}	P	00	HO	RC= 00	LT	CFS	LRN NT

Trunk
Line

LT - See table C-3
CFS - See table C-4

Error Response

PFC= 08	SFC= 81_{16}	P	00	HO	RC

RC - See trunk/line status request responses

DISCONNECT TRUNK/LINE

PFC= 08	SFC= 02	P	00	HO

Normal Response

Normal response is line enabled normal response SM.

Error Response

PFC= 08	SFC= 82_{16}	P	00	HO	RC

SFC - Equals 80_{16} for RC ≥ 04
RC - See trunk/line status request error response codes

CE ERROR MESSAGE

PFC= $0A_{16}$	SFC= 00	EC	Text

EC - Error codes defined in appendix B of The CCP3 Reference Manual
Text - Text defined in appendix B of The CCP3 Reference Manual

Response

None

MESSAGE TO NETWORK OPERATOR

PFC= $0A_{16}$	SFC= 01	00	Text (0 - 50 characters)

Response

None

ALARM MESSAGE TO NETWORK OPERATOR

PFC= $0A_{16}$	SFC= 01	01	Text

Text - Maintenance alarm coupler
       Maintenance alarm MLIA
       Maintenance alarm port xx

Response

None

HOST BROADCAST ONE

PFC= $0C_{16}$	SFC= 00	P	00	HO	CA	TA	DT	THO	Text

Text must be 1 - 50 characters in IVT compatible format.

Normal Response

PFC= $0C_{16}$	SFC= 40_{16}	P	00	HO	CA	TA	DT	THO	RC= 00

Error Response

PFC= $0C_{16}$	SFC= 80_{16}	P	00	HO	CA	TA	DT	THO	RC

RC - 01 = Invalid line number of bad HO or bad THO
     02 = Invalid device type
     03 = Terminal not configured or line not configured
     04 = Terminal inoperative or line inoperative
     06 = HO toggle error

HOST BROADCAST ALL

PFC= $0C_{16}$	SFC= 01	ID1	ID2	HO	Text

ID1/ID2 - If 0, broadcast to console terminals supported by NPU
Text - 50 characters or less in IVT compatible format

Normal Response

PFC= $0C_{16}$	SFC= 41_{16}	RC= 00

Error Response

PFC= $0C_{16}$	SFC= 81_{16}	RC

RC - 01 = No logical link established or specified logical link not established

02 = Broadcast already in progress

OPERATOR MESSAGE

PFC= $0C_{16}$	SFC= 02	P	00	HO	CA	TA	DT	THO	Text (50 characters or less)

TERMINAL CLASS/PAGE WIDTH/PAGE LENGTH (TERMINAL CHARACTERISTICS)

PFC= $0C_{16}$	SFC= 03	P	00	HO	CA	TA	DT	THO	ORIG	TC	PW	PL

DT - See table C-2

ORIG - 00 = Terminal user
01 = Application

TC - Terminal class (see table C-2)

PW - Page width in characters per line

PL - Page length in lines per page

SERVICE MESSAGE MNEMONICS

The following table defines abbreviations used in the individual service message descriptions.

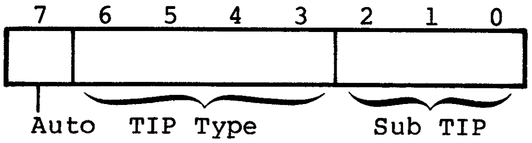
Abbreviation	Meaning
BN	Block Number - used in the overlay load SM to insure delivery of all overlay program blocks
BSN	Block Serial Number - part of the block protocol
BT	Block Type - SMs are always of type CMD
CA	Cluster Address - part of a terminal's physical identification
CD	Code type (see table C-6)
CFS	Configuration State - state of the line as known by the service module (see table C-4 for values)
CN	Connection Number - part of the block address. In the address of a SM, the CN is always zero. When used as data in a SM, the CN may be nonzero.
DN	Destination Node ID - part of the block address
DT	Device Type - part of the Terminal Type (see table C-2)
EB	Error Bit in SM response
FN	Field Number - used in line and terminal configure SMs to describe a field in the LCB or TCB (see table C-5 and C-6 for values)
FV	Field Value - used in line and terminal configure SMs as the value to be put in the field (see tables C-5 and C-6)
HO	Host Ordinal - a value (0 - 15) that is included in all SMs that refer to control structures, and provides unique element identification for the host. For terminals, an additional toggle bit that controls connection switching is included.
ID1	Node ID1 - used to identify the destination node in SMs dealing with logical links.
ID2	Node ID2 - used to identify the source node in SMs dealing with logical links.
LBN	Last Block Number - used in the overlay load SM to insure delivery of all overlay program blocks.
LRN	Link Remote Node - node ID of the neighbor node at the other end of a trunk.
LS	Line Speed Index (see table C6)

Abbreviation	Meaning
LT	Line Type - used to describe the transmission capabilities of the line (see table C-3)
NBL	Network Block Limit - the number of blocks allowed to be outstanding for any one terminal at a given time.
NL	Number of Lines - the number of configured lines belonging to a particular CS.
NT	Number of Terminals - the number of terminals configured on a line.
P	Port - the CLA address used for a communications line.
PFC	Primary Function Code - used to delineate the class of SM (see table C-1)
RB	Response Bit in SM response
RC	Response Code - used in SM responses to indicate the requested action has taken place or an error has occurred.
SFC	Secondary Function Code - used to indicate a particular SM within a class of SMs (see table C-1)
SN	Source Node - part of the block address
TA	Terminal Address - part of the terminal's physical identification
THO	Toggle HO
TOT	Total Number of Status SMs to be sent for this request. Used by the requestor to verify all responses have arrived.
TC	Terminal Class - used to describe the common characteristics of a set of terminals (see tables C-2 and C-8)
TT	Terminal Type - the combination of DT and TC.

TABLES SPECIFYING SM PARAMETER VALUES

TABLE C-2. TERMINAL TYPE/DEVICE TYPE

Terminal Type (TT)



TIP Type:
0 = LIP
1 = ASYNC
2 = Mode 4
3 = HASP
4 = 2780/3780

Auto - Autorecognition Flag

	TIP Type		
Sub TIP	1 ASYNC	2 Mode 4	0 or 3 LIP or HASP
1	110 baud ASCII	4A	not defined
2	150 baud ASCII	4C	not defined
3	300 baud ASCII	not defined	not defined
4	2741 Ext BCB	not defined	not defined
5	2741 Correspond	not defined	not defined

Stored in BZSUBTIP field of LCB

TABLE C-2. TERMINAL TYPE/DEVICE TYPE (Contd)

Device Type (DT)

7	6	5	4	3	2	1	0
Device			Terminal Class				

(TC) Class	Terminals Supported (by Device)				
	0 Console	1 Card Reader	2 Line Printer	3 Card Punch	4 Plotter
1	M33, etc.				
2	713				
4	2741				
5	M40				
6	H2000				
7	751-1				
8	T4014				
9	HASP	HASP	HASP	HASP	HASP
10	200 UT or 4014	200 UT	200 UT		
11	214				
12	711-10				
13	714				
14	731				
15	734				

Device: 5 - Reserved for internal host/NPU use
6 - Reserved for expansion
7 - Reserved for installations

Terminal Class: 16-27 - Reserved for expansion
28-31 - Reserved for installations

When the DT byte is sent in a downline SM to identify a particular TCB, the TC field need not match the field in the TCB as the latter can change at any time.

TABLE C-3. LINE TYPES (LT)

Line Type Hexadecimal Value	Trans-mission Facility	CLA Type	Modem Type	Answer Mode	Carrier Type	Circuit Type	Turn-Around Required	Turn-Around Delayed	Transmission Mode
01	HDX	2560-1	RS232-201A/2081 Compatible	Switched	Controlled	2 Wire	YES	NO	Synchronous
02	FDX*	2560-1	RS232-201B/208A Compatible	Dedicated	Controlled	4 Wire	YES	NO	Synchronous
03	FDX	2560-1	RS232-201B/208A Compatible	Dedicated	Constant	4 Wire	NO	NO	Synchronous
04	HDX	2561-1	RS232-358-1 Compatible	Dedicated	Controlled	2 Wire	YES	NO	Asynchronous
05	HDX	2561-1	RS232-202 Compatible	Switched	Controlled		YES	NO	Asynchronous
06	FDX	2561-1	RS232-103E/113 Compatible	Switched	Constant	2 Wire	NO	NO	Asynchronous
07	FDX	2561-1	RS232-103E Compatible	Dedicated	Constant	2 Wire	NO	NO	Asynchronous
08	HDX	2561-1	RS232-202S Compatible	Switched	Controlled	2 Wire RC††			Asynchronous
09	FDX	2561-1	RS232-103E Compatible	Switched	Constant	2 Wire	NO	NO	Asynchronous
0A	FDX	2563-1	RS232-201B Compatible	Dedicated	Constant	4 Wire	NO	NO	HDLC
0B	RESERVED ⟶								

* Operating with HDX Protocol

††RC = reverse channel

TABLE C-4. CONFIGURATION STATES

Value	Significance
0	LCB not configured
1	LCB configured, not enabled
2	Enable requested to TIP
3	Line operational, no TCBs
4	Line operational, TCBs configured
5	Disable requested to TIP
6	Line inoperative, no TCBs
7	Line inoperative, TCBs configured
8	Disconnect requested to TIP
9	Line inoperative. Waiting for ring indicator or autorecognition in process

TABLE C-5. LINE CONTROL BLOCK FIELD NUMBER/FIELD VALUE (FN/FV) ASSIGNMENTS

Field Number	NPU Mnemonic Name	Description	Mode 4 TIP	ASYNC	HASP
5	BZOWNER	Node ID of owning CS/NS	1-255*	1-255*	1-255*
21	BZLNSPD	Line speed index	-	0-8**	-

*Required for configuration
**Required if autorecognition not specified

TABLE C-6. LINE SPEED AND CODE SET

Line Speed (LS)	Index
800 baud	0
110 baud	1
134.5 baud	2
150 baud	3
300 baud	4
600 baud	5
1200 baud	6
2400 baud	7
4800 baud	8
9600 baud	9
Not used	A through F

Code Set (CD)	Index
Not used	0
BCD	1
ASCII	2
Typewriter-paired ASCII APL	3
Bit-paired ASCII APL	4
External BCD	5
External BCD APL	6
Correspondence	7
Correspondence APL	8
EBCDIC	9
Not used	A through F

LS/CD can occur in a single byte of a service message. In this case, LS uses the upper 4 bits of the byte and CD uses the lower 4 bits.

TABLE C-7. TERMINAL CONTROL BLOCK FIELD NUMBER/FIELD VALUE (FN/FV) ASSIGNMENTS

Field Number	NPU Mnemonic Name	Description	Values		
			ASYNC TIP	Mode 4 TIP	HASP TIP
5	BSTTYP	Terminal class	1, 2, 4-8	10-15	9
12	BSOWNER	Node ID of owning CS	1-255†	1-255†	1-255
13	BSCN	Connection number	1-255	1-255	1-255
14	-	Destination node	0-255	0-255	0-255
15	-	Source node	0-255	0-255	0-255
16	BSABL	Available block limit	0-7†	0-7†	0-7†
19	BSIPRI	Input priority	1-2	1-2	1-2
28	BSPGWIDTH	Page width	0-255	0-255	0-255
29	PSPGLENGTH	Page length	0-255	0-255	-
30	BSCANCHAR	Cancel character	0-127	0-127	0-127
31	BSBSCHAR	Backspace character	0-127	0-127	0-127
33	BSCRIDLES	Carriage return idle count	0-99	-	-
34	BSLFIDLES	Line feed idle count	0-99	-	-
35	BSCRCALC	Calculate CR idle count flag	0-1 (no-yes)	-	-
36	BSLFCALC	Calculate LF idle count flag	0-1 (no-yes)	-	-
37	BSSPEDIT	Special edit mode	0-1 (no-yes)	-	-
38	BSXPARENT	Transparent input mode	0-1 (no-yes)	0-1 (no-yes)	-
39	BSXCHM	Transparent character count delimiter (MSB)	0-15 (most significant 4 bits)†††	-	-
40	BSXCHL	Transparent character count delimiter (LSB)	0-255 (least significant 8 bits)†††	-	-

TABLE C-7. TERMINAL CONTROL BLOCK FIELD NUMBER/FIELD VALUE (FN/FV) ASSIGNMENTS (Contd)

Field Number	NPU Mnemonic Name	Description	Values		
			ASYNC TIP	Mode 4 TIP	HASP TIP
41	BSXCHAR	Transparent character delimiter	0-255	-	-
42	BSXTO	Transparent time out delimiter flag	0-1 (no-yes)	-	-
43	BSINDEV	Input device	0-1 (KB, PT)	-	-
44	BSOUTDEV	Output device	0-2 (PR, DIS, PT)	-	-
45	BSECHOPLX	Echoplex mode flag	0-1 (no-yes)	-	-
46	BSPGWAIT	Page wait flag	0-1 (no-yes)	0-1 (no-yes)	-
47	BSPARITY	Parity mode	0-3 (zero, odd-even, none)	-	-
48	BSABTLINE	Abort output line character	0-127	-	-
49	BSUSR1	User Break 1 character	0-127	0-127	0-127
50	BSUSR2	User Break 2 character	0-127	0-127	0-127
51	BSCODE	TIP code set††	4-5†	1-3††	-
52	BSXCHRON	Transparent message is delimited by a character (flag)	0-1 (no-yes)	-	-

†Required for configuration

††See table C-9 (BSCODE)

†††Pairs 39 and 40 are required together

TABLE C-8. DEFAULT PARAMETERS FOR TERMINAL CLASSES

	ASYNC Terminals						
Terminal Class (TC)	1	2	4	5	6	7	8
Terminal Supported	M33, M35 M37, M38	CDC 713-10	IBM 2741	M40	Hazeltine 2000	CDC 751-1	Tektronix 4014
Page Width (PW)	72	80	132	74	74	80	74
Page Length (PL)	0	0	0	0	0	0	0
Parity (PA)	Even	Even	Odd	Even	Even	Even	Even
Cancel Input Line Char. (CN)	CAN††	CAN††	(	CAN††	CAN††	CAN††	CAN††
Back Space (BS)	BS	BS	BS	N/A	BS	BS	BS
Control Char. (CT)	ESC	ESC	(÷)†††	CTL P	ESC	ESC	ESC
Carriage Return Idle Count (CI)	2	0	CA†	1	0	0	0
Line Feed Idle Count (LI)	1	0	1	3	3	0	0
Special Edit Mode (SE)	No	No	No	No	No	No	No
Transparent Mode (TM)	No	No	No	No	No	No	No
Transparent Delimiter (DL)	CR/ 2043	CR/ 2043	CR/ 2043	CR/ 2043	CR/ 2043	CR/ 2043	CR/ 2043
Device Mode (IN) In/Out (OP)	KB/ PR	KB/ DI	KB/ PR	KB/ DI	KB/ DI	KB/ DI	KB/ DI
Echo Mode (EP)	No	No	N/A	No	No	No	No
Page Wait (PW)	No	No	No	No	No	No	No
Abort Output Line (AL)	CAN	CAN	(	CAN	CAN	CAN	$
User Break 1 (B1)	DLE	DLE	:	ACK	DLE	DLE	DLE
User Break 2 (B2)	DC4	DC4	)	DC4	DC4	DC4	DC4

†Calculated by TIP

††Keyboards may actually be marked as follows: CTLX for CAN, CTL P for DLE, CTL F for ACK, CTL T for DC4

†††÷ for APL

TABLE C-8. DEFAULT PARAMETERS FOR TERMINAL CLASSES (Contd)

	HASP	Mode 4 Terminals					
Terminal Class (TC)	9	10	11	12	13	14	15
Terminals Supported	HASP	200UT/ 4014	214	711-10	714	731	734
Page Width (PW)	80	80	80	80	80	80	80
Page Length (PL)	N/A	13	13	16	16	13	13
Cancel Input Line Char. (CN)	(	(	(	(	(	(	(
Control Char. (CT)	%	%	%	%	%	%	%
Transparent Mode (IN)	N/A	N/A	N/A	No	No	N/A	N/A
Device Mode In/Out	N/A	KBD/ CRT	KBD/ CRT	KBD/ CRT	KBD/ CRT	KBD/ CRT	KBD/ CRT
Page Wait (PG)	N/A	Yes	Yes	Yes	Yes	Yes	Yes
User Break 1 (B1)	:	:	:	:	:	:	:
User Break 2 (B2)	)	)	)	)	)	)	)

TABLE C-9. BSCODE DEFINITIONS FOR CCP INTERNAL USE

BSCODE VALUE	ASYNC		MODE 4		HASP
	TC = 4	TC ≠ 4	TC = $\genfrac{}{}{0pt}{}{12}{13}$	TC ≠ $\genfrac{}{}{0pt}{}{12}{13}$	TC = 9
††0	UNK	UNK	UNK	UNK	UNK
1	External BCD 5	X	X	BCD 1	9 †EBCDIC
2	6 External BCD APL	2 ASCII	X	2 Mode 4A ASCII	X
3	7 Correspondence	3 Typewriter- Paired ASCII APL	2 Mode 4C ASCII	X	X
4	8 Correspondence APL	4 Bit-Paired ASCII APL	X	X	X
5-7	X	X	X	X	X

TC - terminal class
UNK - unknown or does not apply
X - illegal value for that combination of TIP type and terminal class
n - external value for code set

†HASP TIP currently does not use BSCODE since EBCDIC is the only code set supported.

††If a BSCODE = 0 is specified for an ASYNC terminal, the ASYNC TIP with default to the ASCII code set.

BLOCK PROTOCOL SUMMARY

D

Block protocol is the protocol used to communicate between the NPU and the host. It is used for data (message) transfer and for commands and status transmission. (A few commands and status transfers are confined to the handshaking routines in the host coupler; these do not use the block protocol.)

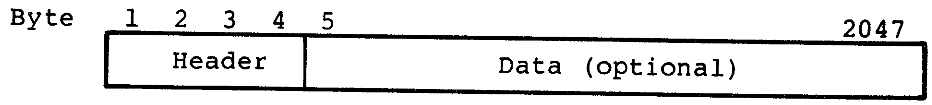
BLOCK SIZE

The minimum block size of a block is 4 bytes. The block consists of only the block header.

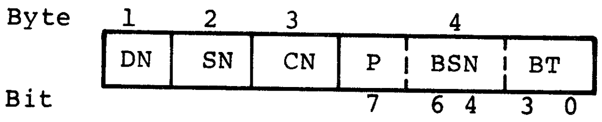
The maximum block size is 2047 bytes, which includes the block header of 4 bytes plus data bytes.

BLOCK FORMAT

The format of a block is as follows:



The format of this header field is shown below:



DN - destination node

SN - source node
00 = NS in host
01 = CS in host
02 = NPU coupler node } Only one NPU
03 = NPU terminal node } in system
02-225 = nodes in systems with more than one NPU in system

CN - connection number
00 = service message
01 = 225 = line

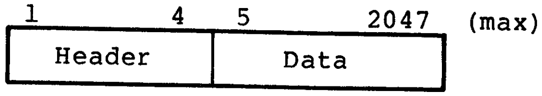
P - Priority (trunks)
1 = high
0 = low

BSN - block serial number (0 for ACTL blocks)

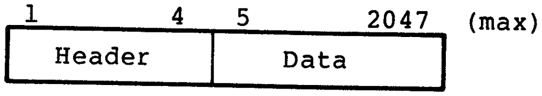
BT - block type; see table D-1 for description of types.

BLOCK TYPE

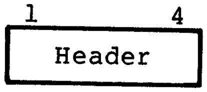
BLK = 1



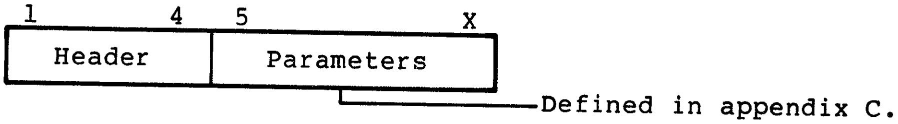
MSG = 2



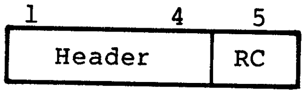
BACK = 3



CMD = 4

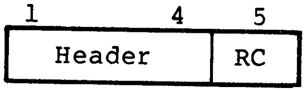


BRK = 5



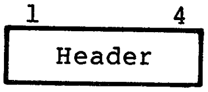
RC - Reason Code
    00 = illegal
    01 = user break 1
    02 = user break 2
    03 = output device not ready
    04 = illegal/invalid format in block received from host

STP = 6

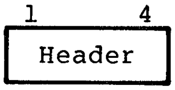


RC - Reason Code
    00 = illegal
    01 = terminal busy
    02 = terminal failure
    03 = batch interrupted by interactive I/O

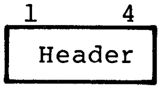
STRT = 7



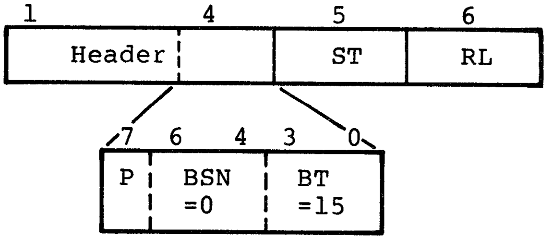
RST = 8



INIT = 9



ACTL = 15 (for use between neighbor NPUs only)

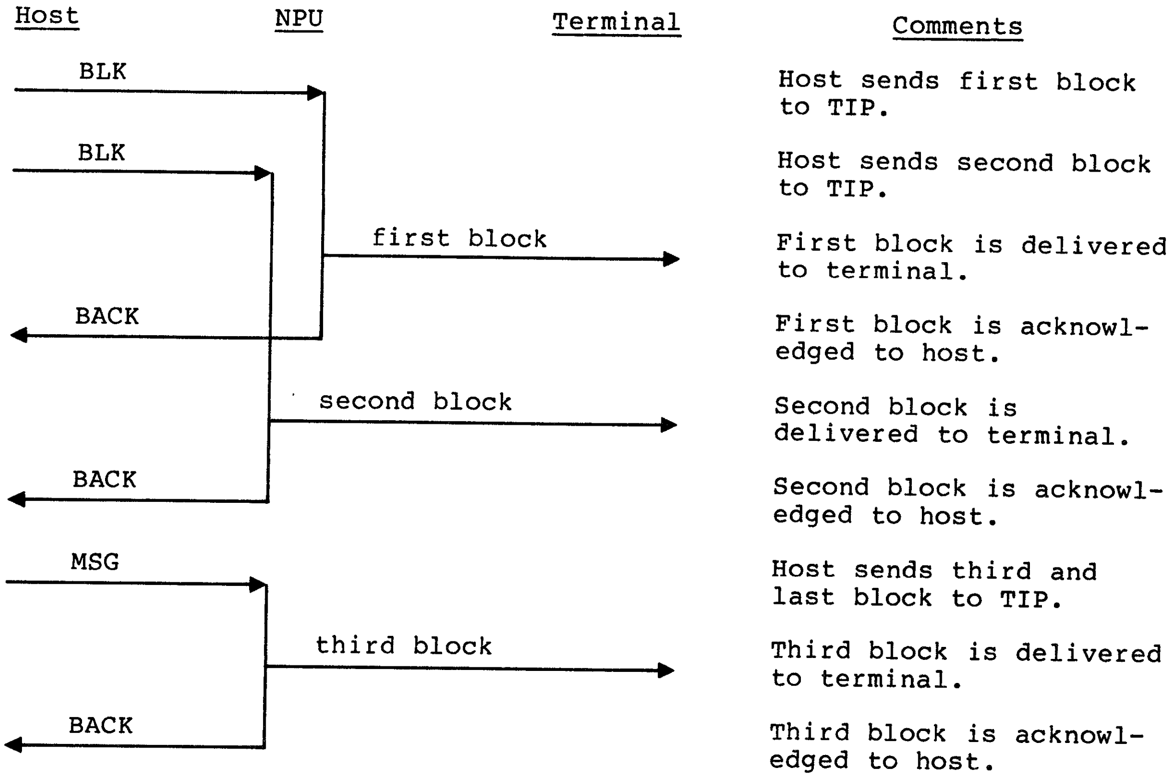


ST - Subtype
00 = clear (CLR)
01 = protocol reset (PRST) } RL byte used
02 = regulation (REGL) }
03 = link initialization (LINIT) } RL byte not used
04 = link idle (LIDLE) }

RL - Regulation level
00 = high
01 = low

BLOCK FLOW

Figure D-1 illustrates sample data block protocol flow downline and figure D-2 shows the sample data block protocol upline flow. Figure D-3 illustrates the downline flow where the TIP controls restart and figure D-4 shows downline flow where the host controls restart.



Host sends first block to TIP.

Host sends second block to TIP.

First block is delivered to terminal.

First block is acknowledged to host.

Second block is delivered to terminal.

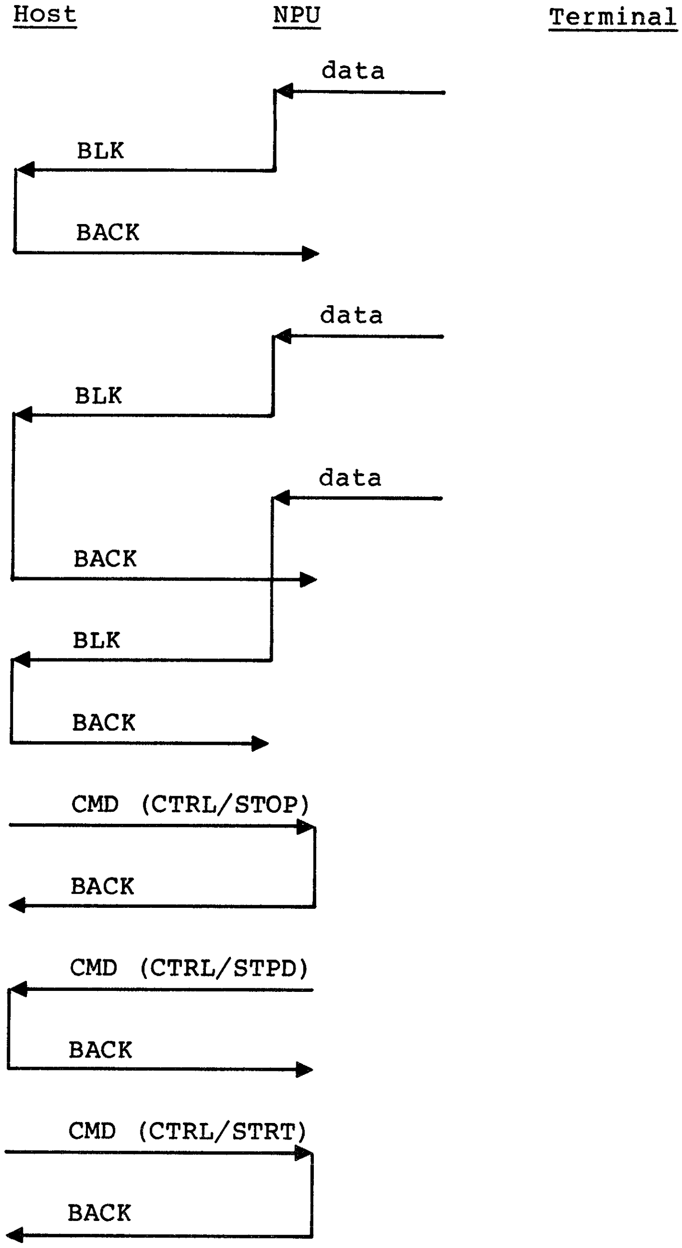
Second block is acknowledged to host.

Host sends third and last block to TIP.

Third block is delivered to terminal.

Third block is acknowledged to host.

Figure D-1. Data Block Protocol Downline



Comments

TIP inputs data from terminal.

TIP sends data block to host.

Host acknowledges block to TIP.

TIP receives input data from terminal.

TIP sends first block to host.

TIP gets more input from terminal.

Host acknowledges first block.

TIP sends second block to host.

Host acknowledges second block.

Host requests stop input.

TIP acknowledges CMD block to host.

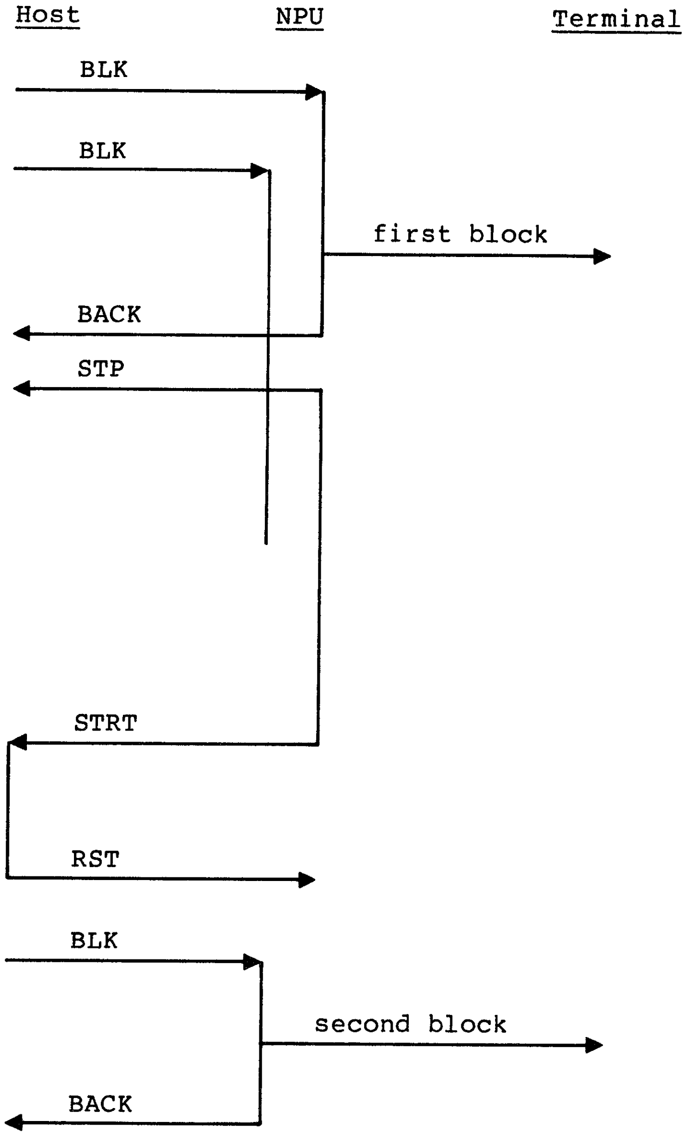
TIP informs host of input stopped.

Host acknowledges CMD block to TIP

Host requests input start.

TIP acknowledges CMD block to host.

Figure D-2. Data Block Protocol Upline



Comments

First block sent from host to TIP.

Second block sent from host to TIP.

Data of first block sent to terminal.

First block acknowledged.

TIP determines an STP condition exists and informs host with an STP containing the BSN of the last acknowledged block.

All unacknowledged data and all new data received by the TIP is discarded until a RST is received in response to a STRT from the TIP.

TIP determines STP condition has been resolved so a STRT is sent.

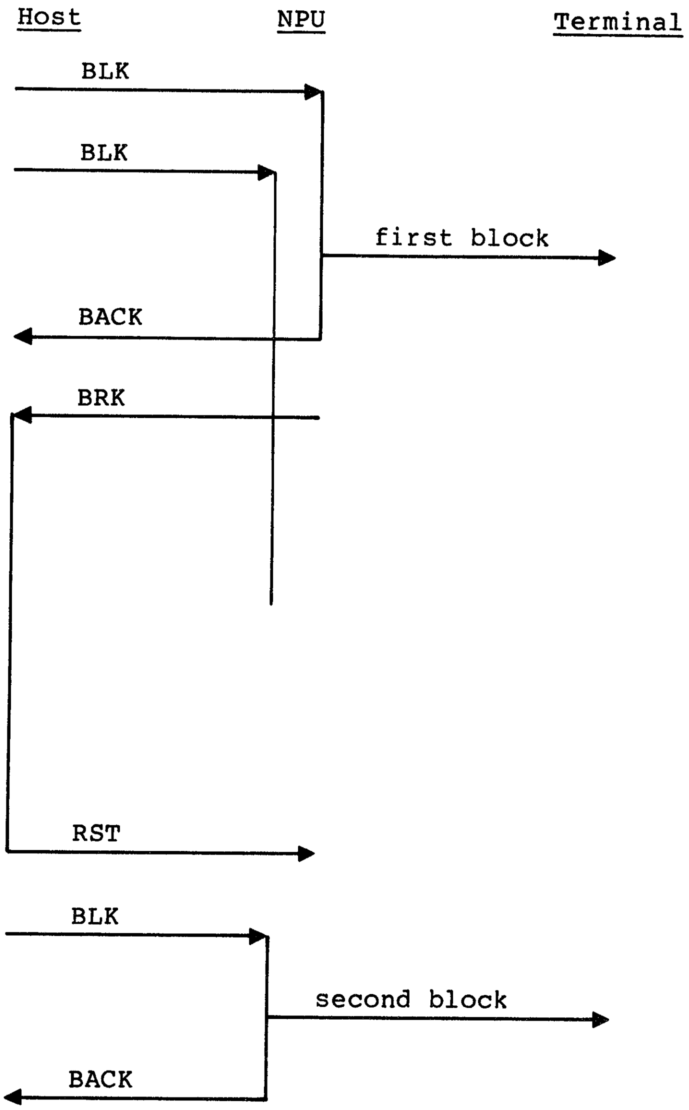
STRT acknowledged by RST from host.

Second block retransmitted to TIP.

Data of second block delivered to terminal.

Second block acknowledged to host.

Figure D-3. Block Flow Downline Control (TIP Controls Restart)



Comments

First block sent from host to TIP

Second block sent from host to TIP

First block delivered to terminal

First block acknowledged to host

TIP determines a BRK condition exists and informs host with a BRK containing the BSN of the last acknowledged block.

All unacknowledged data and all new data received by the TIP is discarded until a RST is received from host.

Host acknowledges BRK with a RST to TIP.

Second block retransmitted to TIP

Second block delivered to terminal

Second block acknowledged to host

Figure D-4. Block Flow Control Downline (Host Controls Restart)

SAMPLE MAIN MEMORY MAP FOR NPU

E

Figure E-1 shows the layout of CCP in the main memory of a 255x network processor unit with 65K words of main memory.

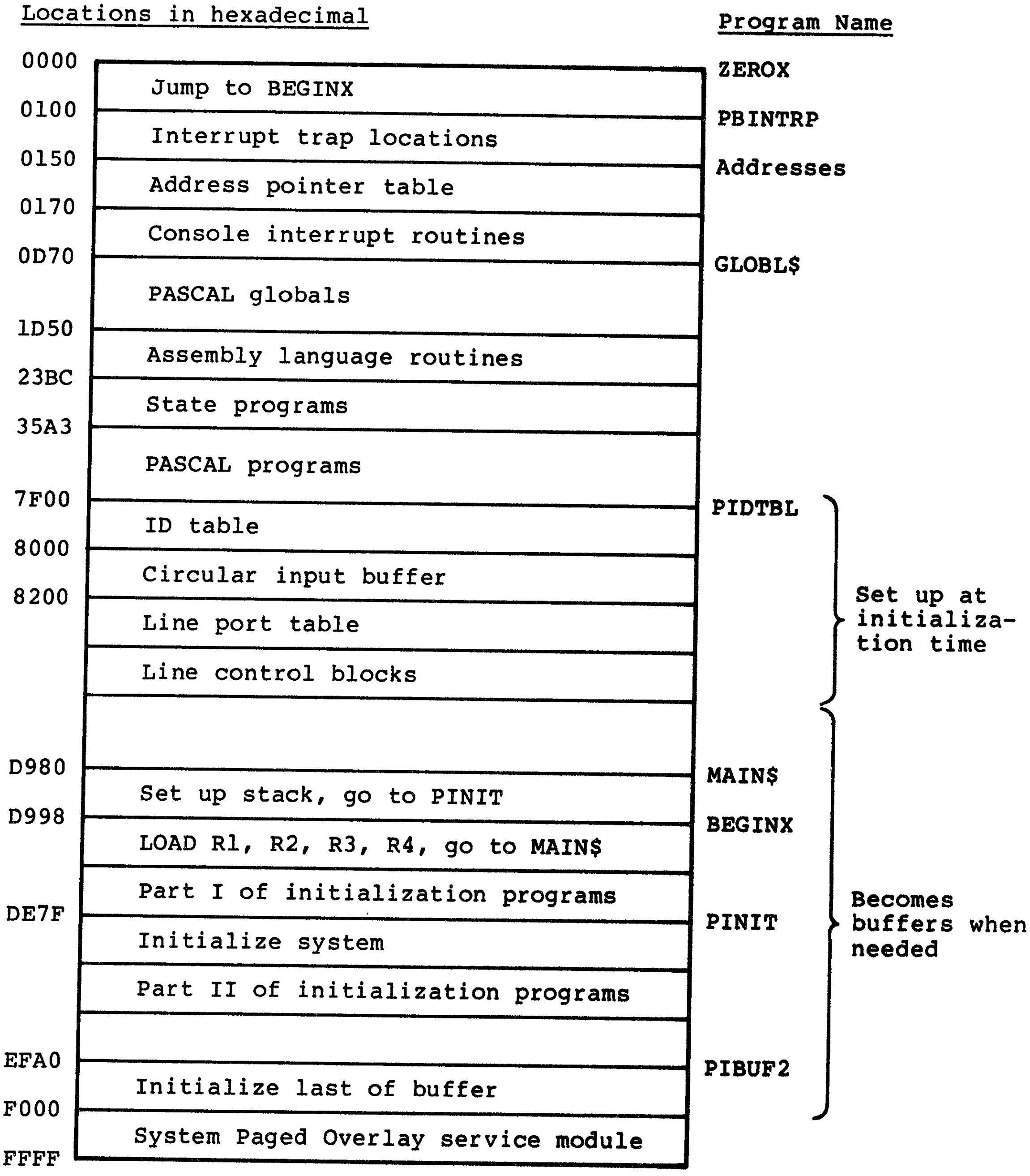


Figure E-1. Sample Main Memory Map

CCP NAMING CONVENTIONS F

The following naming conventions for the CCP PASCAL programs should be regarded as guidelines rather than as strict requirements.

The general format of a label is

PIRRRRSSS

where the usual length is six bytes, but additional bytes can be used.

P values are:	A - O	Global data
	P	Procedure or function
	Q - W	Local data
	X - Z	Non-CDC
I values are:	0	Transparent or not tied down
	1 - 9	Not a structure
	A - Z	A structure

For procedures and functions:

P = P, I =	A	Assurance programs
	B	Base system programs
	D	Diagnostic programs
	M	Multiplex subsystem programs (part of the base system)
	N	Network communications programs
	P	Packets
	T	TIPs, HIP, LIP

For types, variables, and fields:

AO...	OPS-level workcodes
BA...	Overlay
BC...	Physical/logical request packet (PRP/LRP)
BF...	Buffer
BJ...	TIP-type table
BL...	Logical link control block (LLCB)
BS...	Terminal control block (TCB)
BT...	Timing, monitor controlled
BW...	Intermediate array for worklist
BY...	Worklist control block (WLCB)
BZ...	Line control block (LCB)
CM...	Service module
D...	Input/output (I/O)
J...	Logical/physical I/O request packet
JC...	TUP table
LD...	Load or dump
M...	Multiplex subsystem
MM...	Event worklists (multiplex subsystem)
N...	Multiplex subsystem
NA...	Port table
MB...	Line types
MC...	Multiplex LCB (MLCB) or text processing control block (TPCB)
NJ...	Terminal characteristics
NK...	Multiplex command driver inputs (command packet)
NZ...	Diagnostics control block (DCB)
SI...	System interfaces (SIT)

STANDARD TIP AND SVM TREES

G

This apendix consists of four sections, one for each of the standard TIPs (Mode 4, ASYNC, and HASP) and a section for the service module (SVM).

Within each TIP section there are two parts: a one-line description of each routine or subroutine, followed by a tree for the PASCAL-level routines and subroutines comprising the TIP. The trees are laid out so that the OPS work-level entry is on the first sheets and subroutines follow. Following the OPS-level switch and preceding the subroutines are the direct call routines from SVM and multiplex level 2 interrupt routines.

Comparing these trees and TIPs should aid the TIP programmer in finding how other TIP programmers have solved similar problems.

In the illustrations of the trees, external calls are underlined. No effort is made to trace calls from external routines.

MODE 4 TIP PROGRAMS

PTSTACK - provides push down stack for TCB

PTUNSTACK - Pop up part of stacking for TCB

PT4CYCLE - TIP reentry with simulated WC (for shared terminals on line)

PT4RELBUF - Release a buffer chain

PT4DISABLELINE - Processes disable line request

PT4TERMINATETCB - Processes terminate TCB request

PT4OUTPUT - If anything is in the output queue sets flag

PT4GET - Get next downline message (interprets data on stop/start/IVT cards)

PT4TIMECHECK - Checks one second event timer

PT4LASTCHAR - Find last character of message

PT4CMD - Generates and finds upline replies

PT4CSTATE - Change cluster states for batch devices

PT4TEXTPROCESS - Transforms downline data to Mode 4 format. Calls PTTPINF - interface to firmware text processor

PT4PMSG - Generates poll message

PT4LINIO - Initiates I/O for mode 4 line (MLCB setup, start, set timeout value)

PT4RETRY - Checks for unrecoverable errors

PT4TOGGLE - Polls for toggle following write

PT4POLL - Issues poll message

PT4WRT - Issues output data block terminal

PT4EPOLL - Polls for read response

PT4DOUT - Sends message to display

PT4DINP - Polls display for input

PT4PROUT - Sends message to display

PT4PROUT - Sends message to printer

PT4E3WRITE - Generate E3 write for card reader

PT4CRINT - Polls MD4A terminals for data (card reader)

PT4 CONFIGURE - Configure request

PT4AUTORECOGNITION - Polls CA to find code set, for ASCII terminal - configure request for terminal address for MD4A - reports a console, card reader, and line printer

PT4ERRORPROCESS - Disable response to disable request
Line error - send line inop SM
Break - bad downline data
Others - terminal/cluster error

PT4WKALLOCATION - Finds next unit of work for terminal (reports several types of errors)

PT4STRT - Sends stop input message to host

PT4STOP - Send start input message to host

PT4WKPROCESS - Cycles thru TCBs for active line, allocates work on that line

PT4IOCHECK - Processes I/O returns - uses work code

PT4CERR - Processes CE error messages

PT4WCCHECK - Processes OPS level workcodes -

enable line
process queued output
delete TCB
disable line
process cycle reentry
read E1, E2, E3, or autorecognition response
process ACK, REJ, or errors on line
process timeouts

PTMD4TIP - Main OPS-level-worklist entry switch

PT4TCBINIT - Prepares TCBs (direct call from SVM)

OPS LEVEL SWITCH

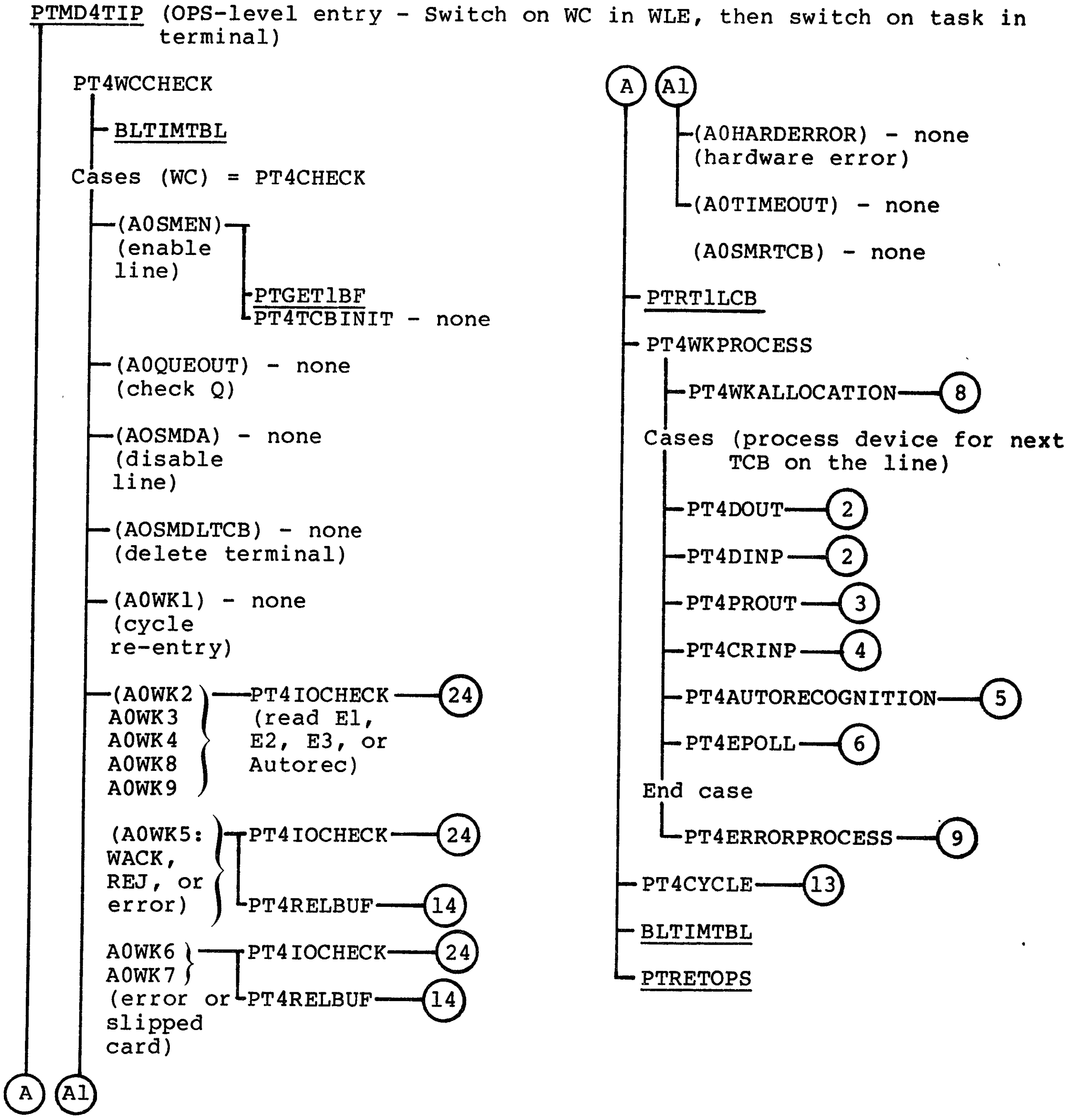


Figure G-1. Mode 4 TIP (sheet 1 of 7)

WORKCODES

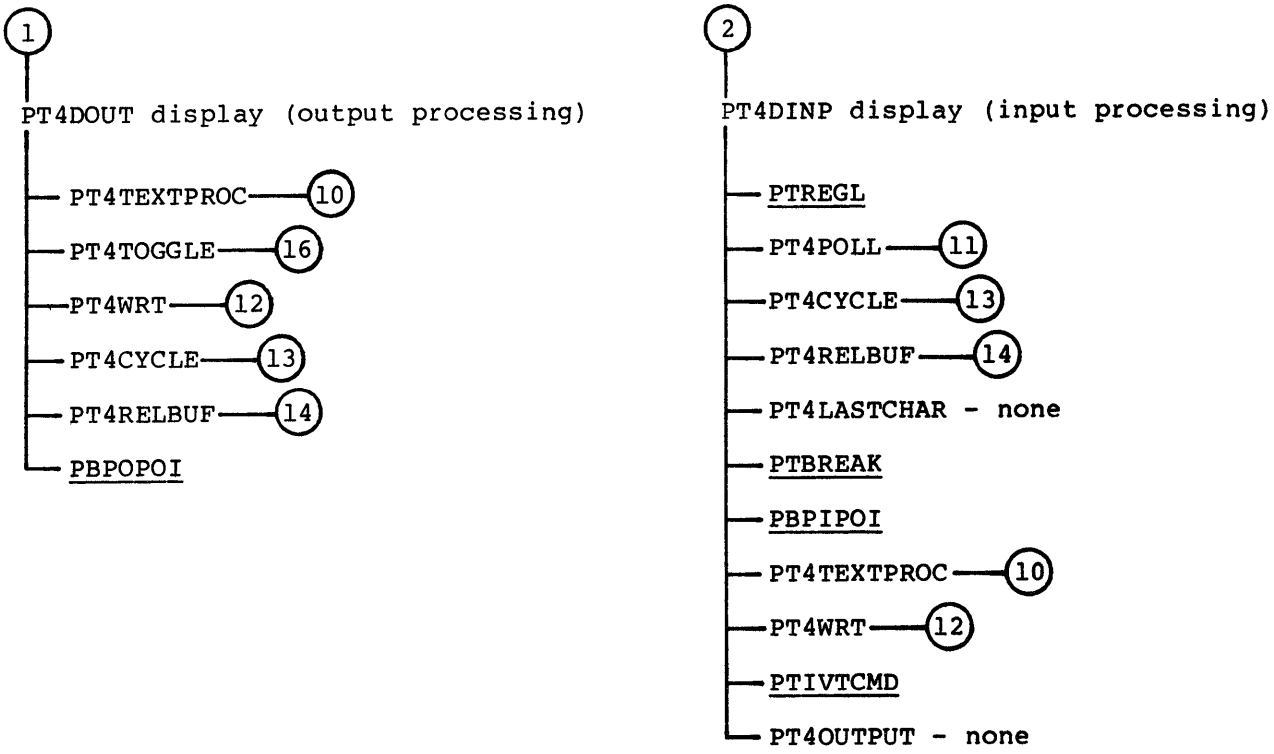


Figure G-1. Mode 4 TIP (sheet 2 of 7)

WORKCODES

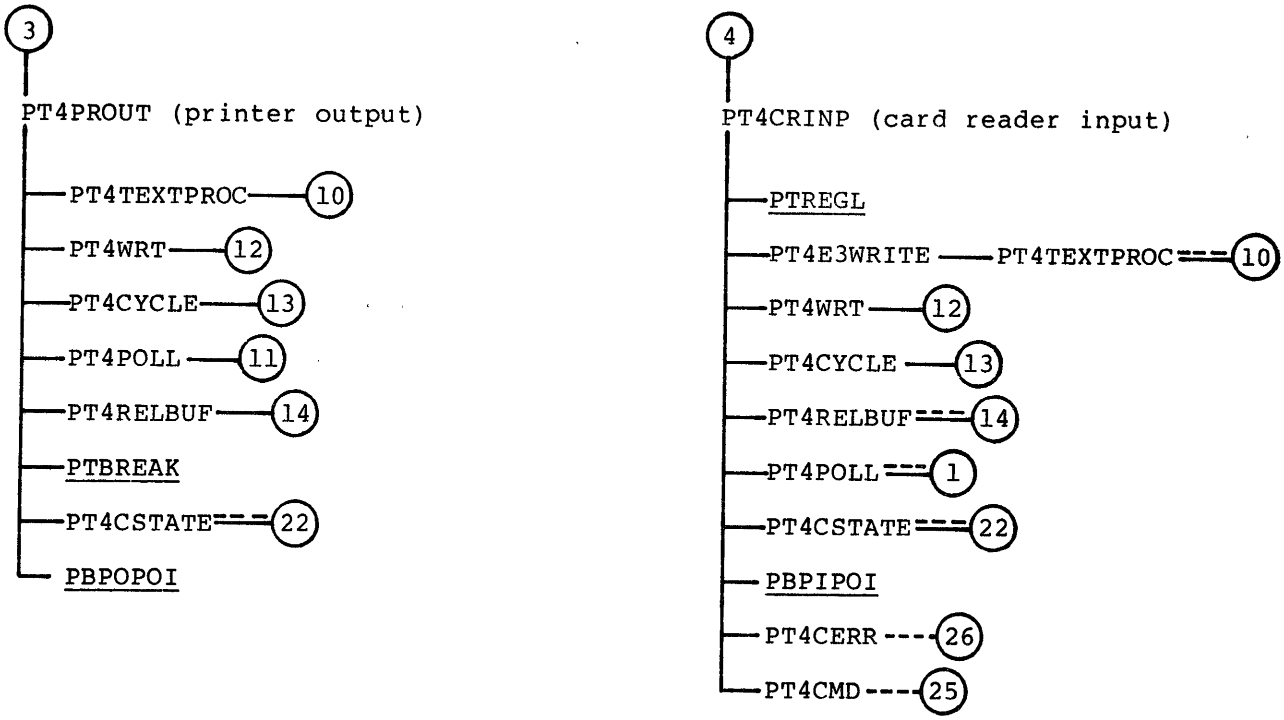


Figure G-1. Mode 4 TIP (sheet 3 of 7)

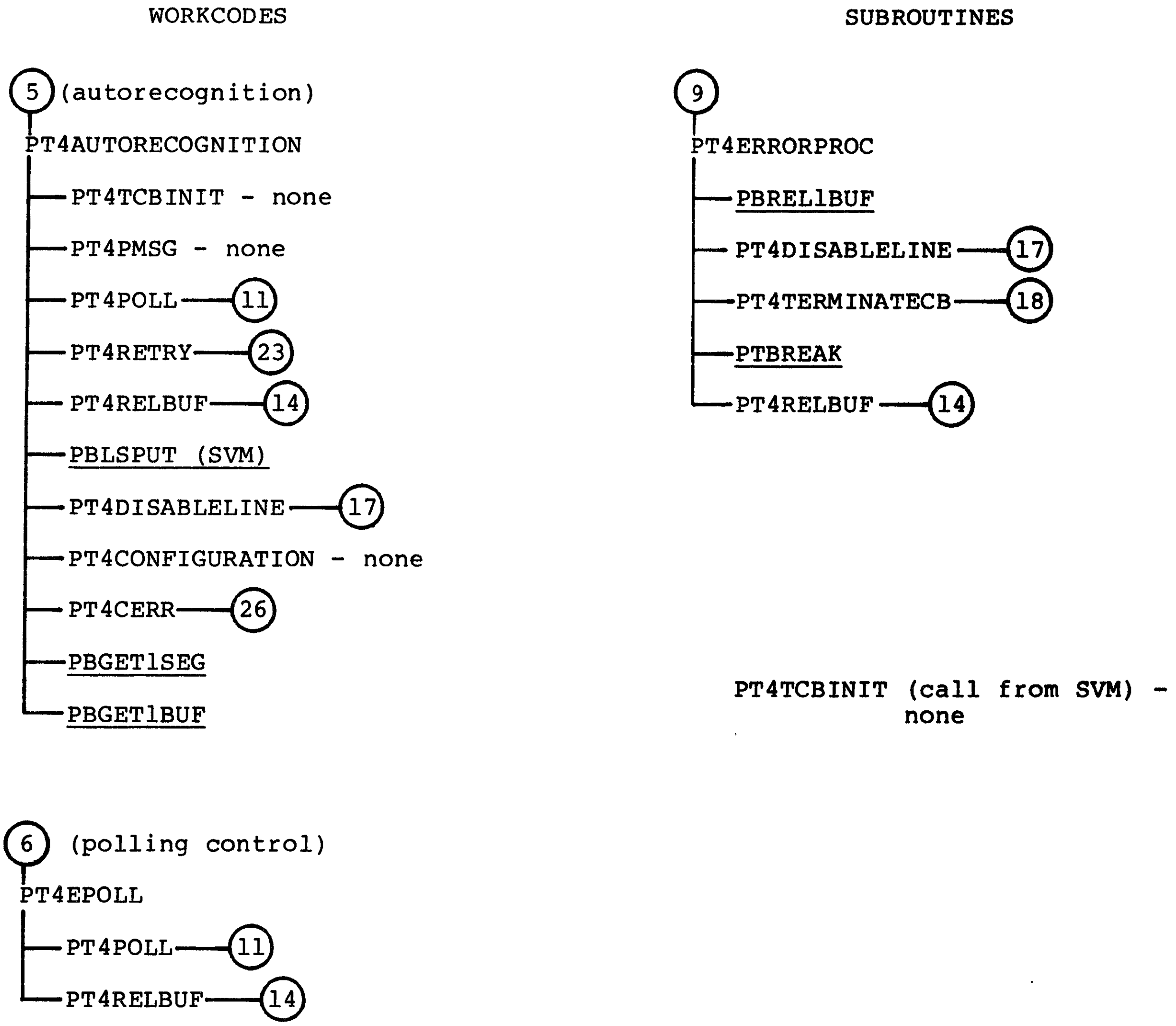


Figure G-1. Mode 4 TIP (sheet 4 of 7)

SUBROUTINES

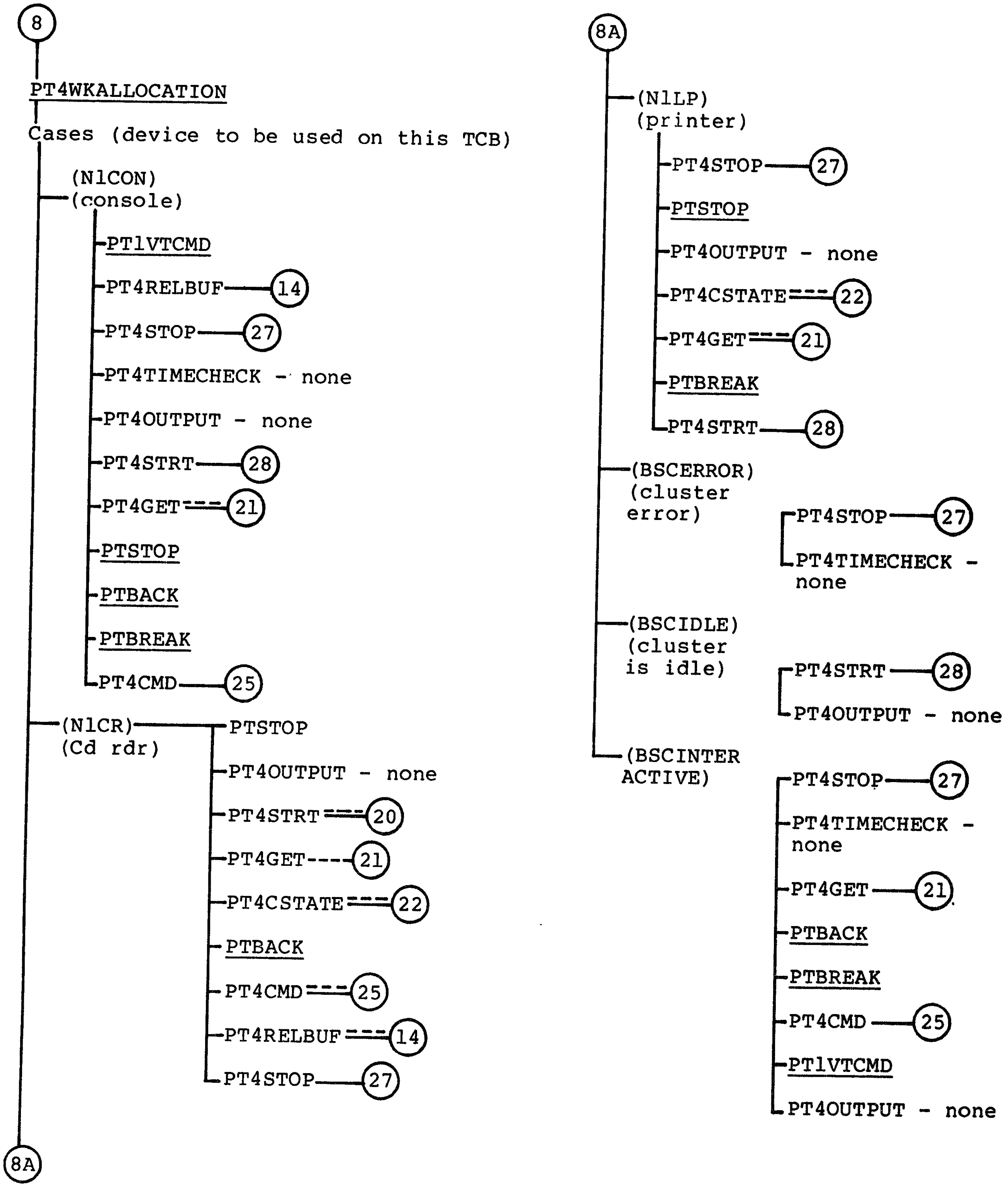


Figure G-1. Mode 4 TIP (sheet 5 of 7)

SUBROUTINES

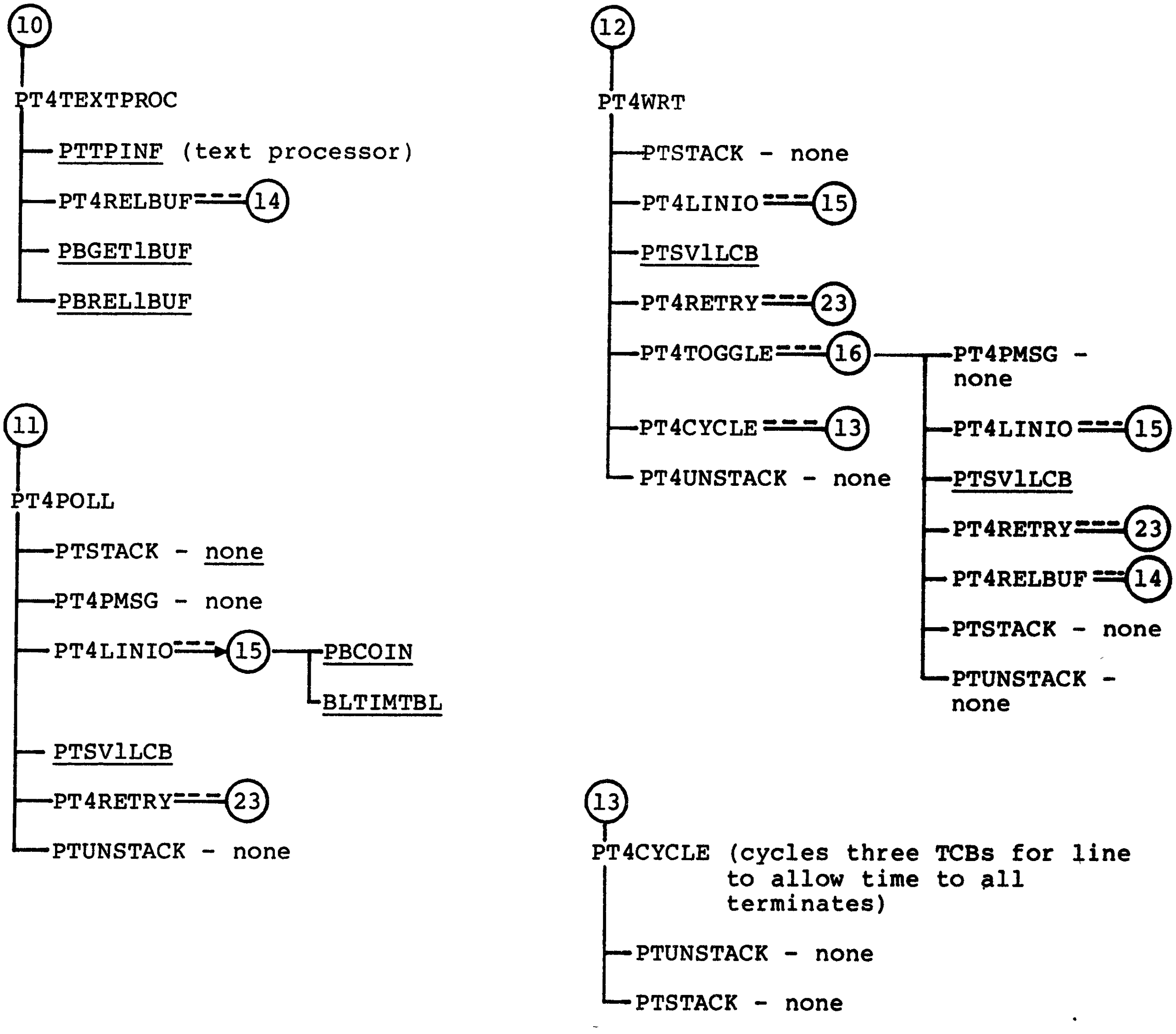


Figure G-1. Mode 4 TIP (sheet 6 of 7)

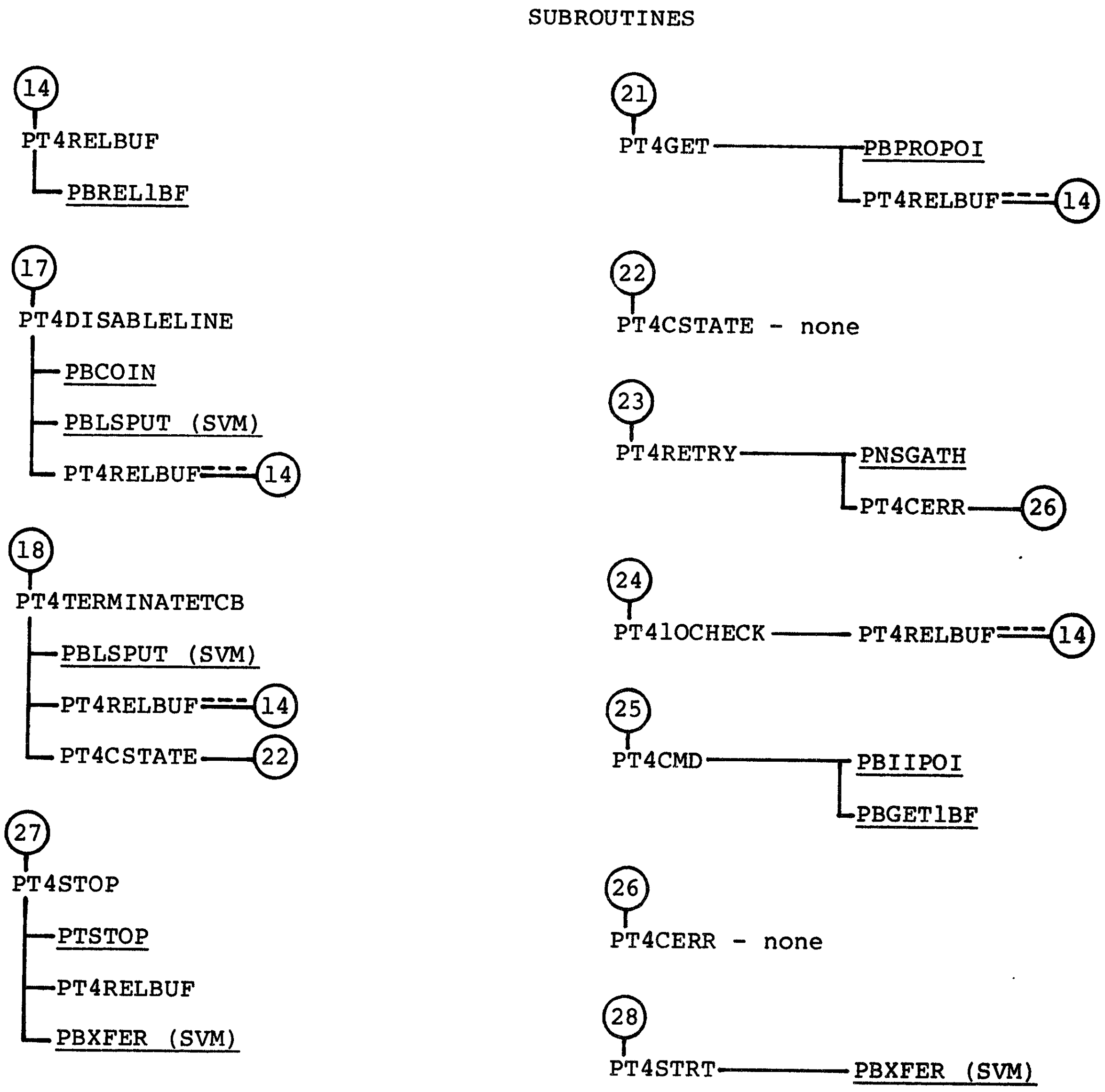


Figure G-1. Mode 4 TIP (sheet 7 of 7)

ASYNC TIP PROGRAMS

PTASNMUX - Multiplex level 2 worklist handler

- APLSPUT - Converts multiplex level 2 interrupt to an OPS-level worklist entry
- APCDRV - Call command driver

PTASNOPS - OPS level entry, worklist main processor

- Output buffer sent, or terminate transmission
- 200ms Timeout handler (WLE from multiplex level 2)
- Regulation and autorecognition timeout
- Buffer threshold reached
- Break received
- XOFF received
- Trailer sequence
- Hard error or bad autorecognition
- Output queued, try to output
- TCB built, try I/O
- Line enabled
- Disable line
- Reconfigure TCB
- Delete TCB
- Input terminated
- Send transparent block to host
- Transparent message timeout
- Transparent block size reached during input; transparent XOFF checked;
- End of logical line; check for commands, echo data; pass data to host
- Input active, turn off output
- Autorecognition

PTAFALASTBUF - Returns address of last buffer in chain

PTAFCMDCHECK - Checks if input block is an IVT command from the terminal or a DATA block

PTAFINOK - Tests if it is OK to input

PTAFNULLMSG - Checks if input block is a null input

PTAFOUTOK - Tests if it is OK to output

AFREGAFTERINPUT - After input is passed to tip, checks if system is in regulation. If so, TIP releases input - notifies terminal that message was discarded.

PTAUTOIN - Autoinput handler

APCMDACTION - Performs action requested by IVT command

APBRINGLINEDOWN - Brings line down due to a disable command or line errors

PTAPBUFREL - Release a buffer chain

APCMDRESP - Responds to command by sending message (or action) to terminal

APENDOFLINE - Sends EOL sequence to terminal

APENOUT - Sends terminate output command to command driver

APEPLX - Change echoplex state at terminal

APGETOUTPUT - Get block for output - prepare it (text process data; interpret if it is a command)

APIOCHECK - Checks for I/O to do

APIVTFORMAT - Puts character string in IVT block format for returning an answer to the terminal

APOUTPUT - Build command packet and call command driver for an output block

APPASSINPUTTOHOST - Pass input block to host - works through Post-Input POI (chains autoinput heading to reply block)

APPREPARETEXT - Text processes output blocks (format for output to terminal)

APRCVST - Sends control D to terminals (2741s) - puts terminal in write mode

APRELOQUE - Purge output queues

APSPECSEQ - Process breaks, abort, cancels characters

APUPBREAK - Process break on output; ends output

PTAREC - Turns on input for autorecognition

PTASETINPUT - Sets up MLBC for IVT interactive input

PTATPTC - Sets up text processor interface (TPCB) by terminal class

PTAPO - ASYNC TIP call to Post-Output POI

PTAPI - ASYNC TIP call to Post-Input POI

APPGPARAM - Sets page width and page length in TCB

APTERMTCB - Terminates a TCB

PTAQOBT - Processes output block from queue

APTCBINIT - Initializes TCB fields

APWTOBTERM - Waits for output buffer terminated

PTAFICCHAR - Checks first input character for controlfunctions

PTABKSPCHECK - Checks backspace character

WORKCODES

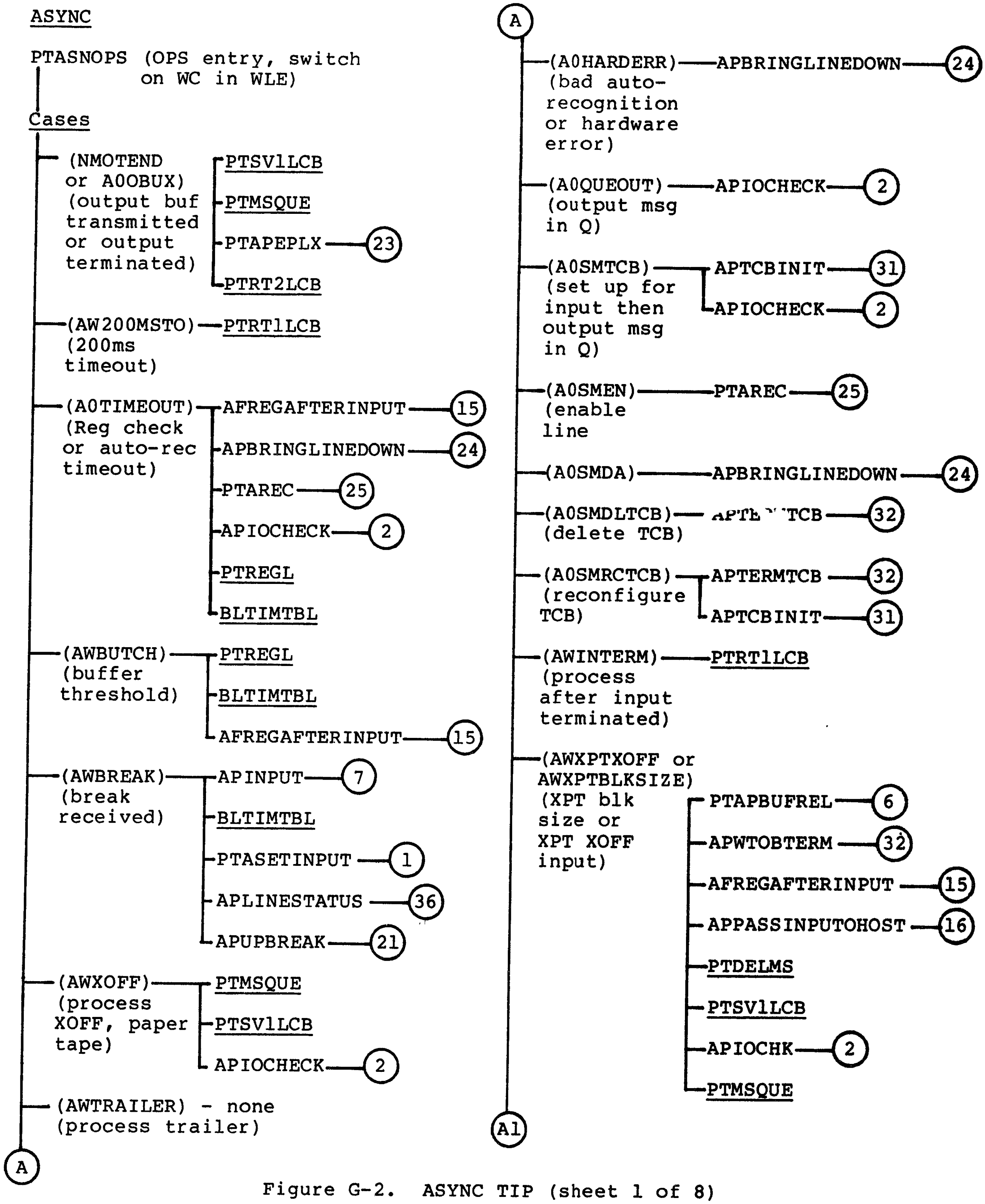


Figure G-2. ASYNC TIP (sheet 1 of 8)

WORKCODES

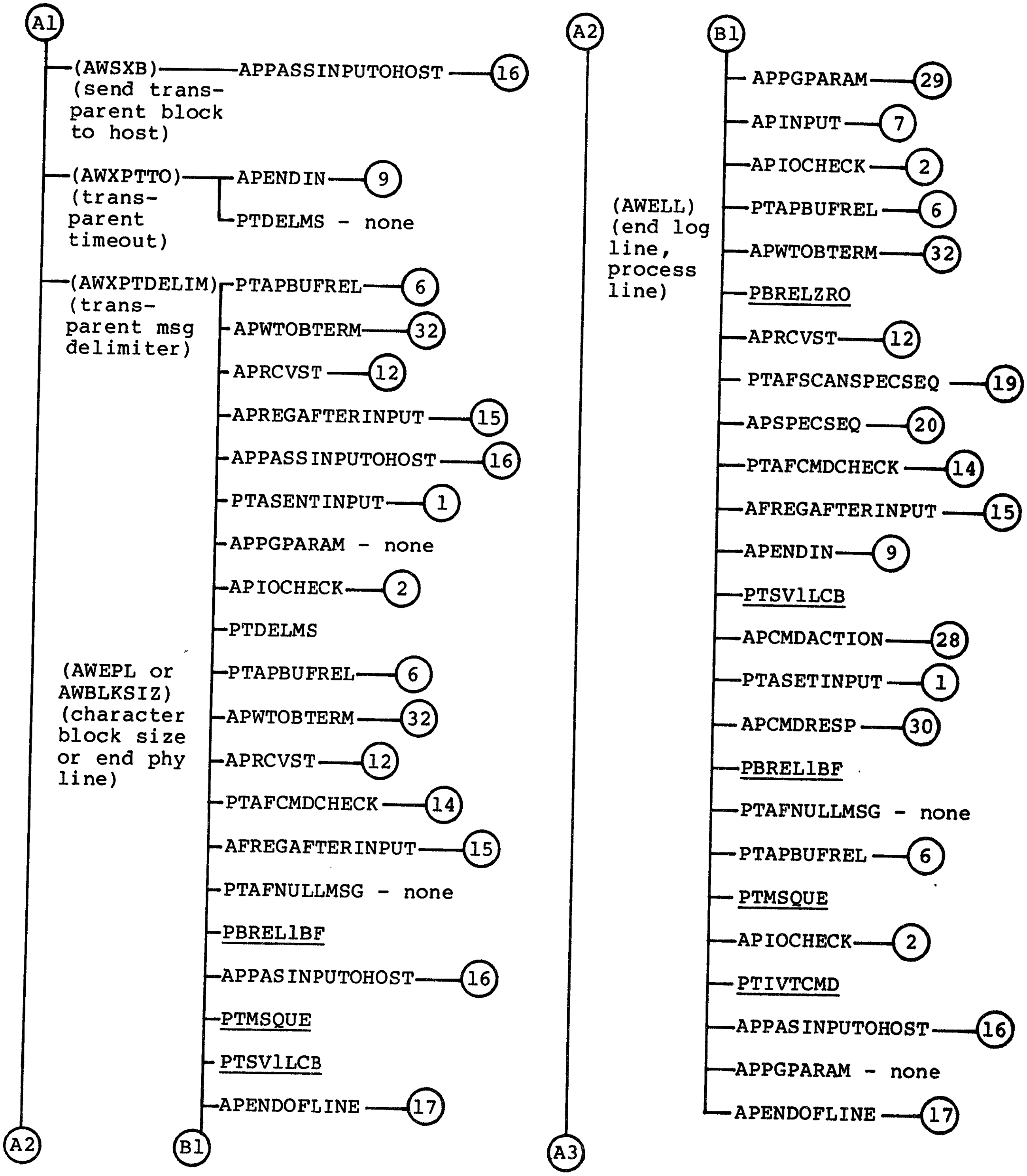


Figure G-2. ASYNC TIP (sheet 2 of 8)

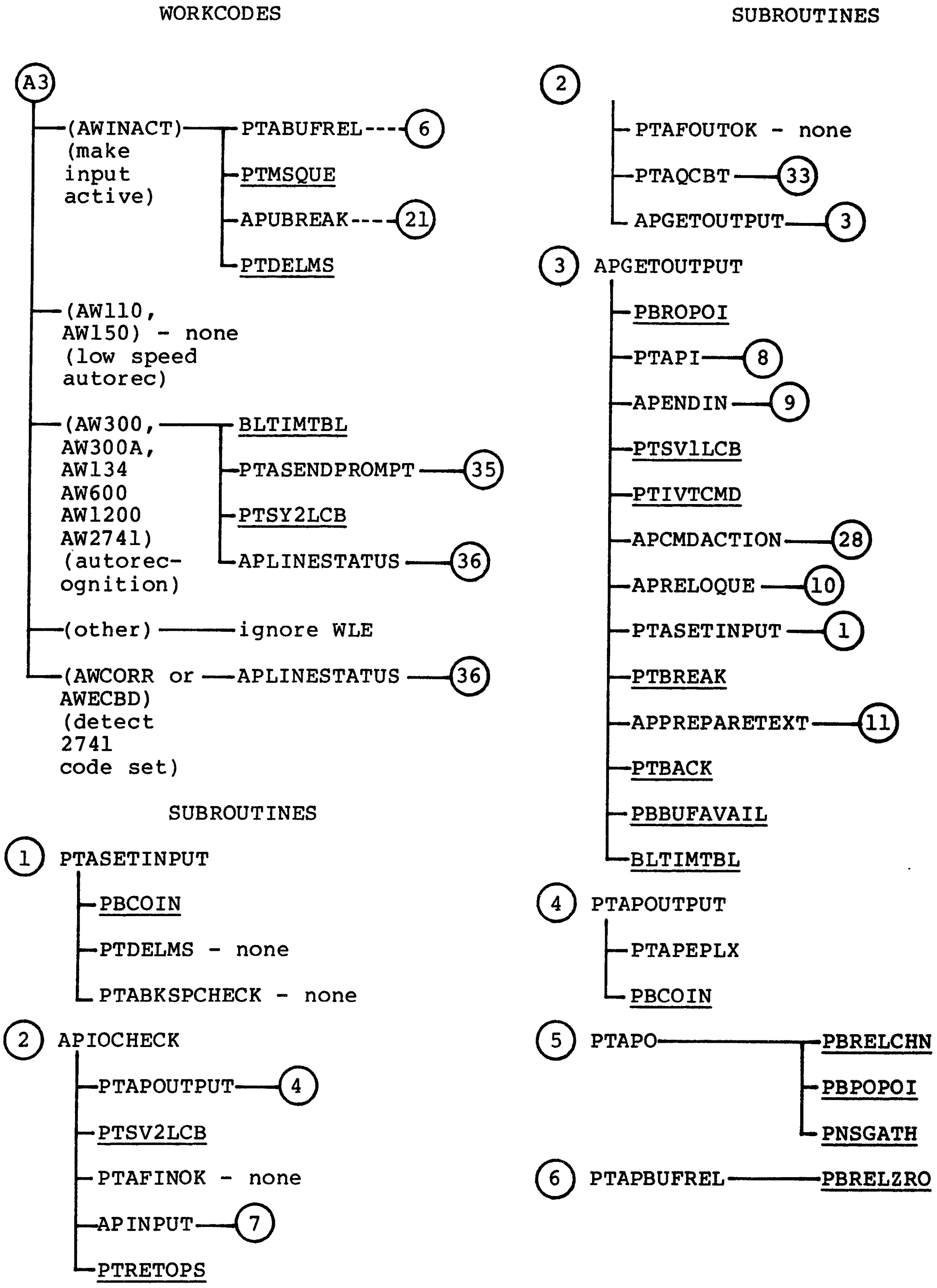


Figure G-2. ASYNC TIP (sheet 3 of 8)

SUBROUTINES

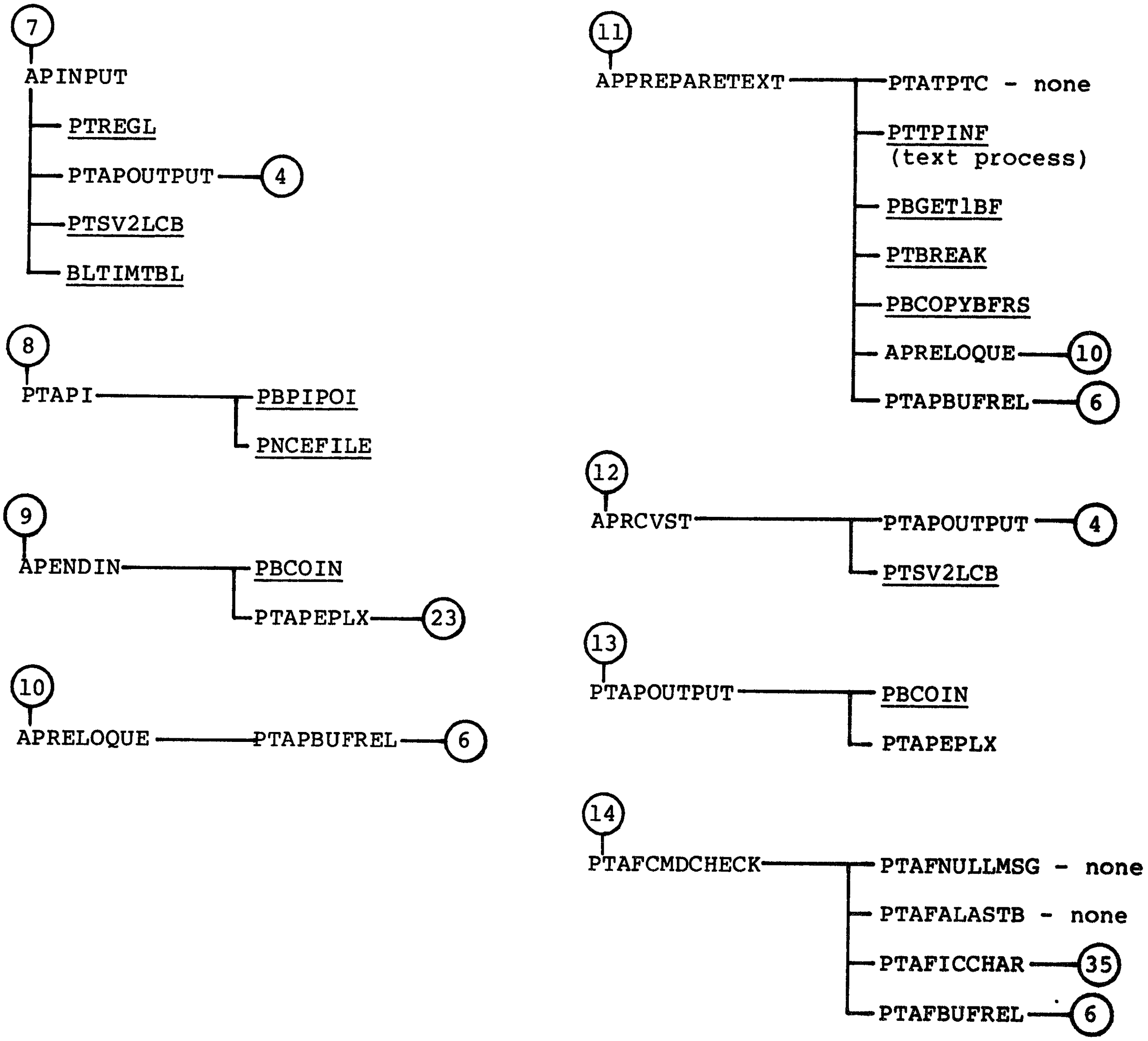


Figure G-2. ASYNC TIP (sheet 4 of 8)

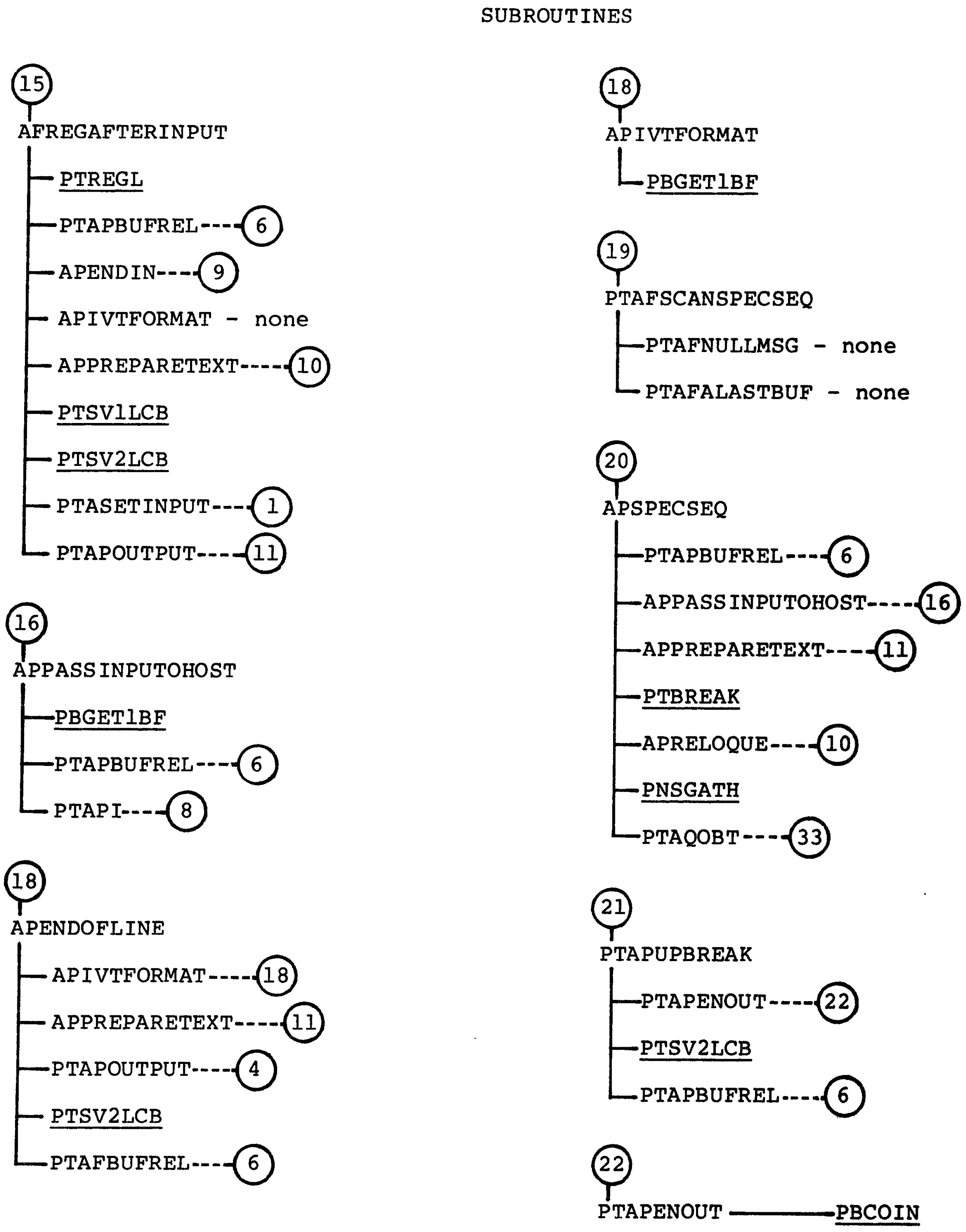


Figure G-2. ASYNC TIP (sheet 5 of 8)

SUBROUTINES

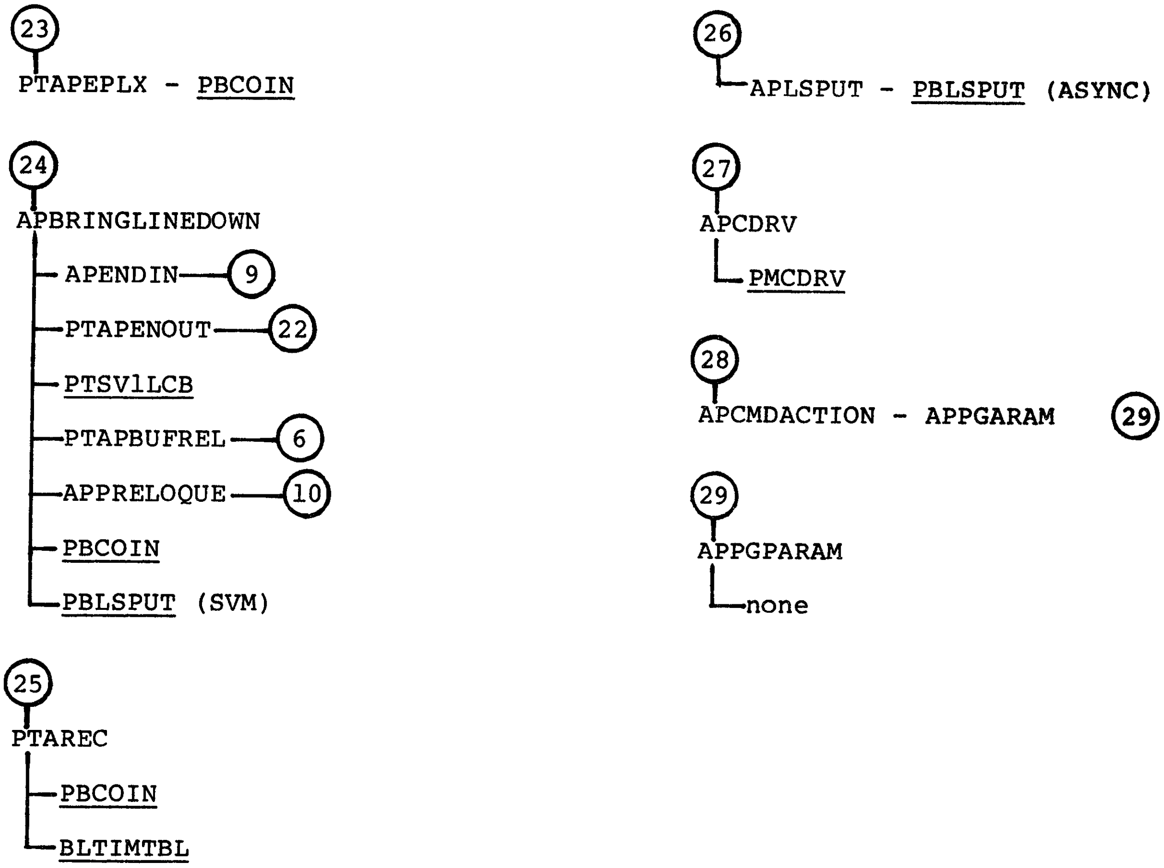


Figure G-2. ASYNC TIP (sheet 6 of 8)

SUBROUTINES

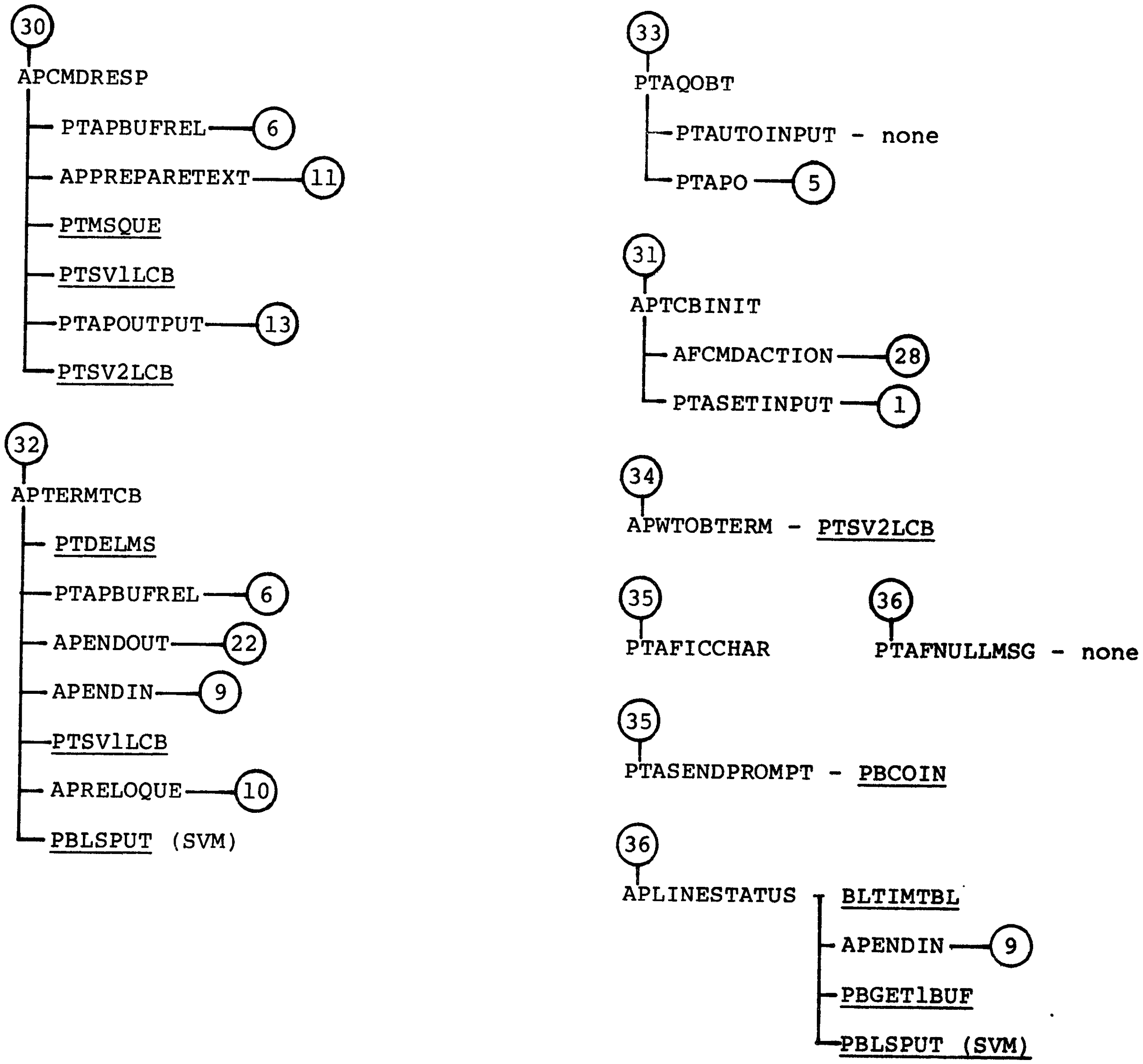


Figure G-2. ASYNC TIP (sheet 7 of 8)

CALLS WITHOUT AN OPS LEVEL WORKLIST

PTATPTC - none

PTASNMUX (mux level entry, WLE switch)
- APLSPUT-----㉖
- APCDRV-----㉗

Figure G-2. ASYNC TIP (sheet 8 of 8)

HASPTIP

PTHSMUXTIP - Mux level 2 workcode entry - converts mux level worklist to OPS level HASP worklist

PTHSOPSTIP - OPS-level entry. Processes worklists from OPS-level (main HASP processor)

(A0SMEN) - Enables line (sets LCB fields)

(A0SMDA) - Disables line

(A0SMTCB) - Checks for an ENQ block; process transmission

(A0SMDLTCB) - Terminates and releases TCB, passes terminate command to command driver, notifies host

(MSGCONT) - Prepares RCB/SRCB

/RQP/ - Requests permission to send

/PG/ - Permission granted to send

/BCBERR/ - Bad BCB, brings line down

/CONT/ - Sends control record

/0,3,4,5/ - Purge record

(A0TIMEOUT) - Timeout handler

(A0QUEOUT) - Output handler

(MSGCMPLT - Message completed, return to caller

(ERROR) - Release buffer - return to caller

(ACK/NAK) - Sets good or bad completion value, returns to caller

(NMINDEND) - Ends input, returns to caller

(MMHARDER) - Hardware error, sets inop code and returns to caller

(BUFTHR) - No buffers (threshold reached), drops message

(FALL THROUGH) - End of switch: error

NAKTEST - If NAKs received after I/O, brings line down

FINDTCB - Finds TCB for stream (upline TCB location)

PTHSSENDCMD - Sends upline command to host (multileaving control; input stopped)

STROPN - Checks if workstation device will accept data (wait-a-bit-check); notifies host if it will

DELINK - Unlinks entry from data-list queue (DLQ)

HASPGET - Removes entry from DLQ - i.e., gets buffer of data that is ready to transmit

HASPPUT - Queues entries into DLQ (2 wds/entry)

HASPIO - Calls command driver (PBCOIN)

PUTBCBFCS - Sets up BCB and FCS for output

PTHSBCBFCS - Sets up BCB and FCS

PTTHASP - Output text processing - calls PTTPINF

GENDATA - Sets up buffer prior to PTTPINF call

WRAPUP - Cleans up data transfer to HASP workstation

BRINGLINEDOWN - Terminates a HASP workstation due to errors - sends terminal command to mux, notifies host

ERRCHK - Checks for errors in I/O transfer - mark line down if necessary

CHKCMD - Parses CMD blocks from host for a HASP TCB

PREOUTPUT - Gets next entry in TCB queue and starts processing (downline switch)

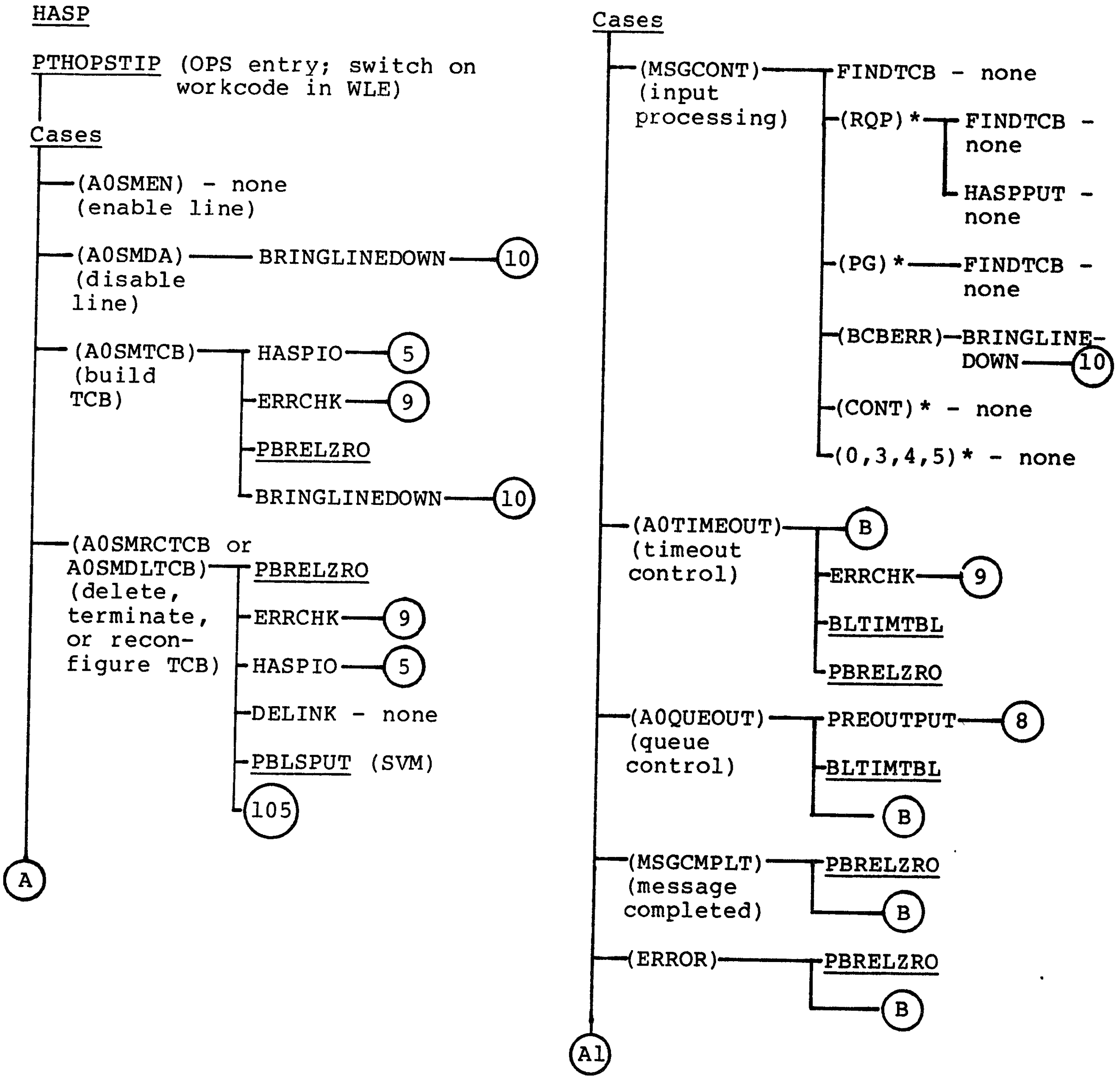
POSTOUTPUT - Cleans up output transmission (PBPOPOI)

HSPR4INP - Input text processing (second pass processing of input data)

HSPOSTINP - Cleans up input transmission (PBPIPOI) for input text processing

HSPTCBUILD - Initializes TIP-dependent TCB fields - direct call from SVM during configuration of terminal

WORKCODES



*RCB Switch

Figure G-3. HASP TIP (sheet 1 of 4)

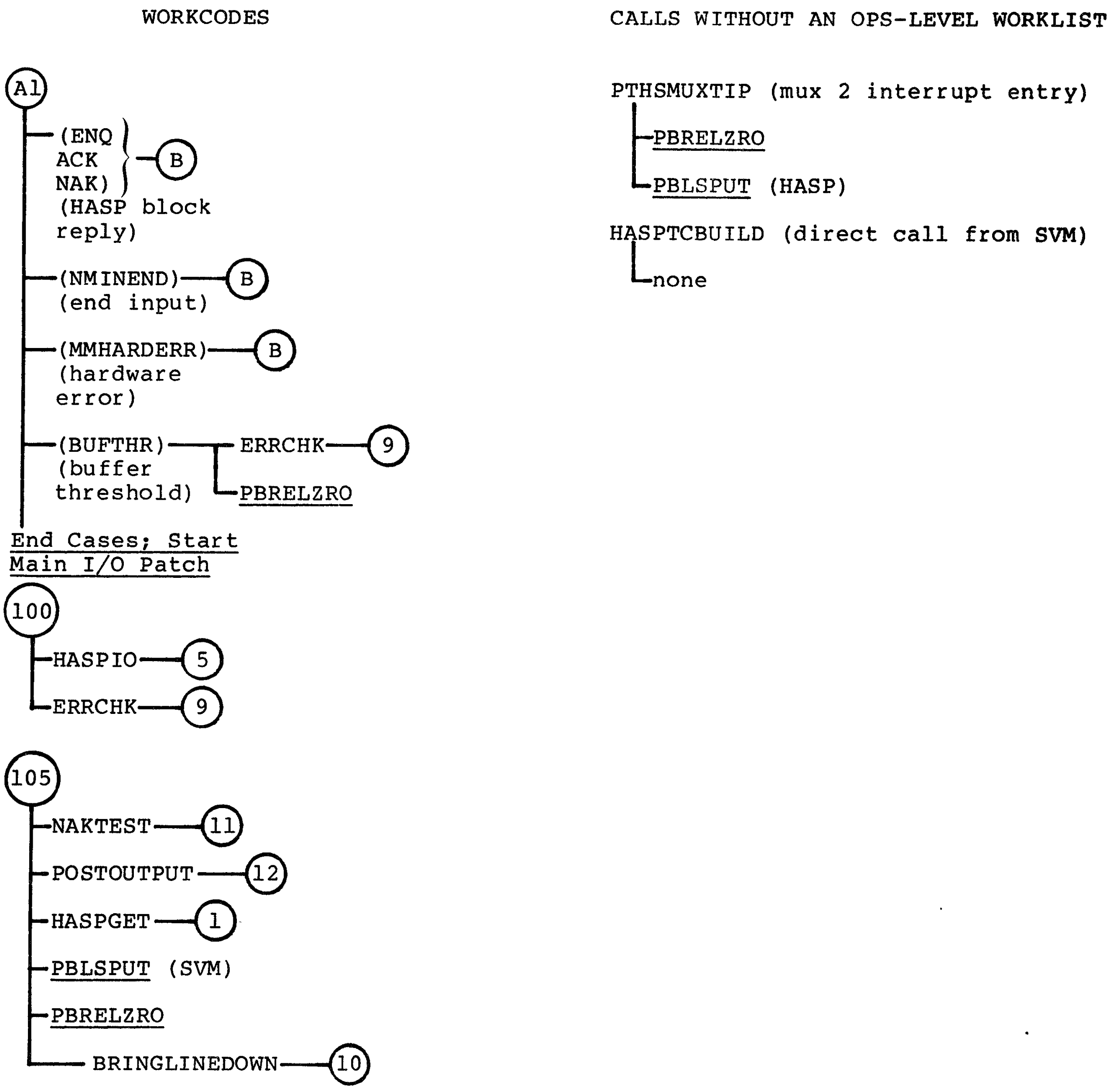


Figure G-3. HASP TIP (sheet 2 of 4)

SUBROUTINES

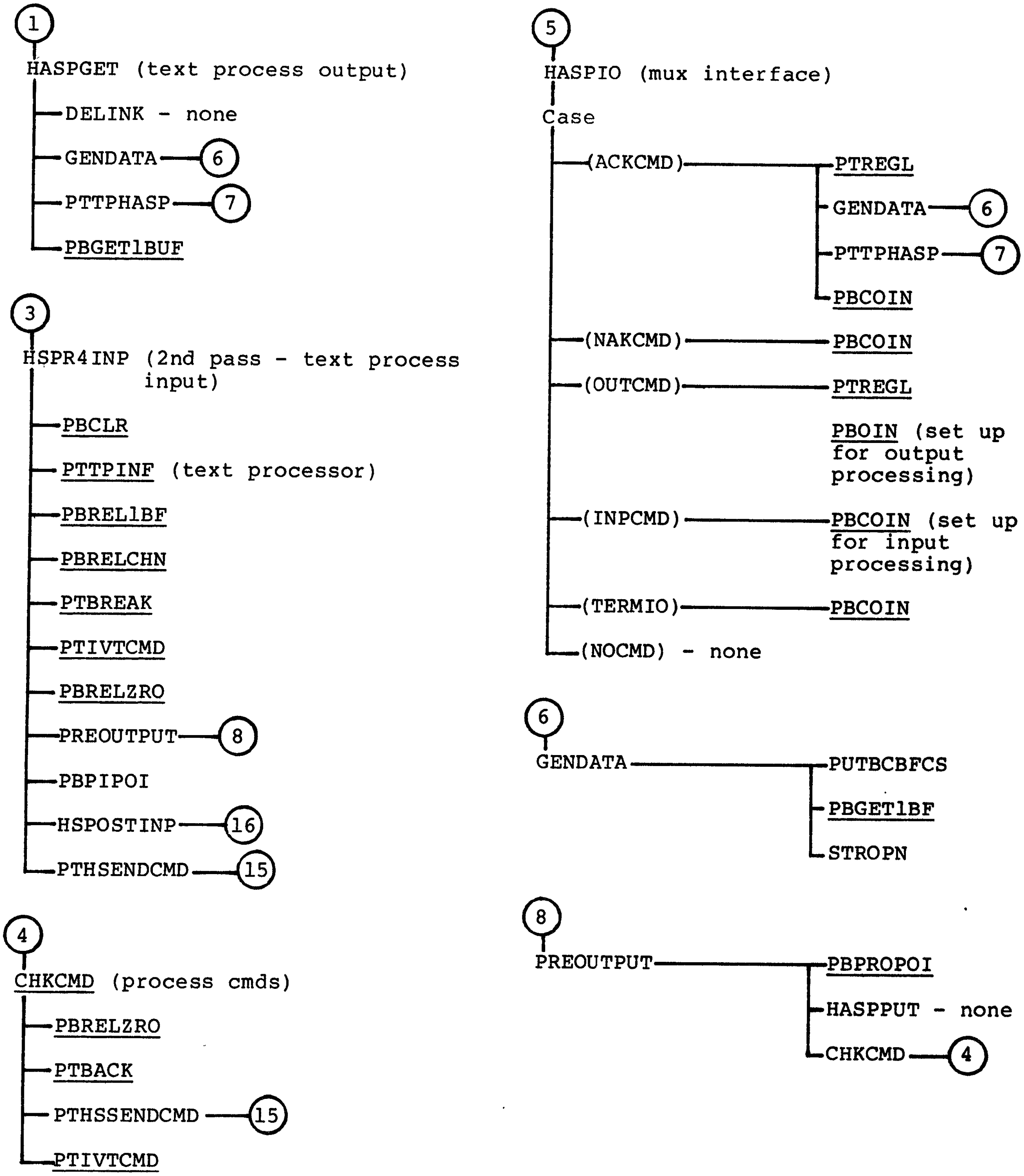


Figure G-3. HASP TIP (sheet 3 of 4)

SUBROUTINES

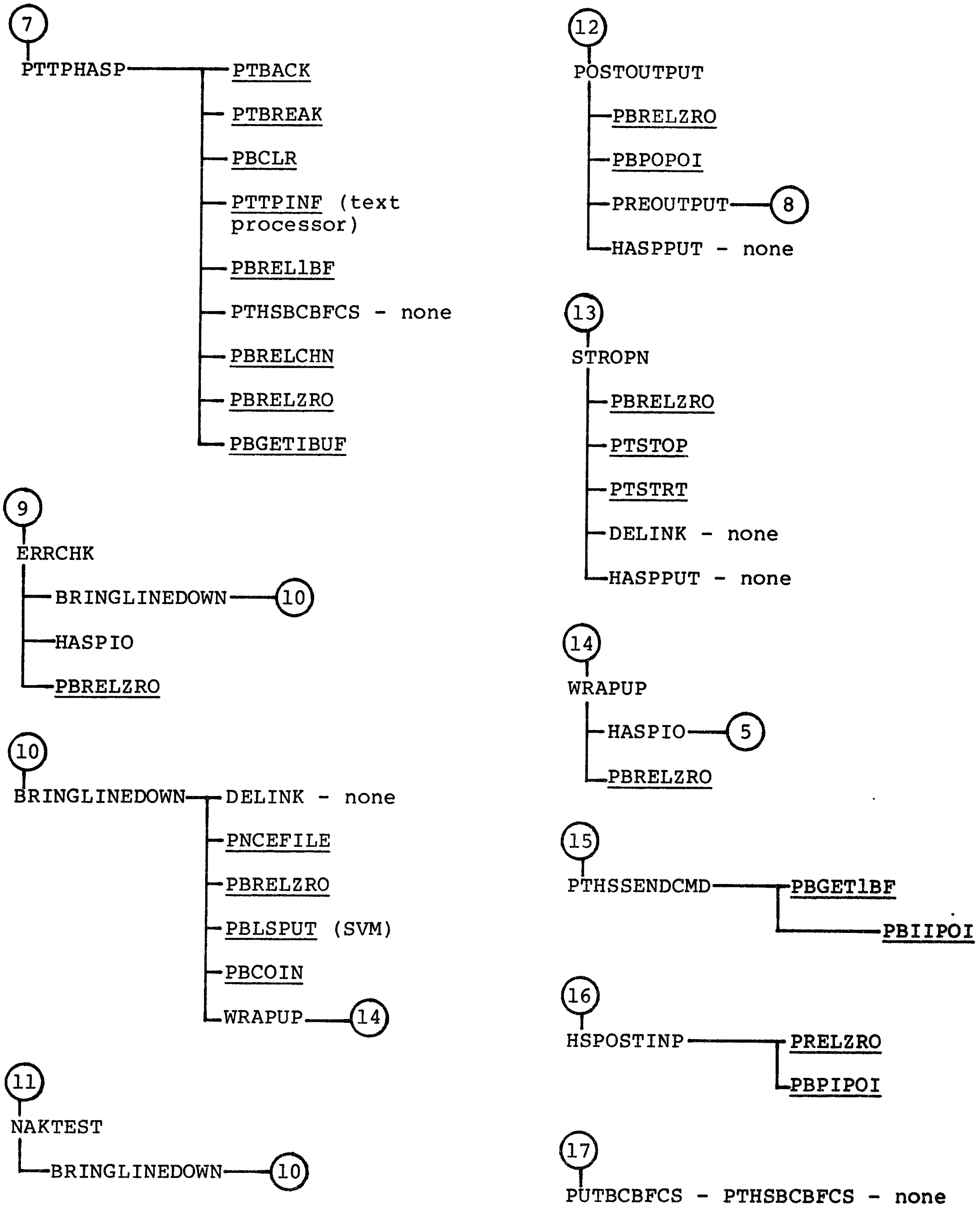


Figure G-3. HASP TIP (sheet 4 of 4)

SVM TREES

The section shows the service module trees. There are two parts: a short description of each SVM routine, and the trees relating the routines.

Note the routines which are service module related, but not a part of the SVM:

PNSGATH - gathers statistics (stores them in TCB or LCB as appropriate)

PTLINIT - initializes the line by setting up the LCB

PNCEFILE - generates the CE error and alarm service messages. The text of the message identifies the line (or other device such as coupler, MLIA, etc.) which failed.

See appendix C for format of individual SMs.

SVM Routines:

PNAWAIT - gives up control while waiting for external event (availability of buffers, or a TIP to perform a specific set-up or deletion)

PNRTN - used to regain control after PNAWAIT is used

PNSMBAD - validates PFC/SFC of SM

PNLNBAD - validates line number (used when enabling and deleting lines)

PNRVRSE - reverses SN and DN to return an SM reply to the host

PNTOCONS - delivers an SM to the NPU console

PNQREL - releases buffers in a queue

PNGTCB - gets a TCB address

PNCRWAIT - terminates a reconfiguration in progress

PNTCBSRCH - uses line number, cluster address, terminal address, and device type to find a TCB

PNDLTCB - deletes a TCB and its queue

PNDISCARD - discards SMs with invalid PFC or SFC

PNSMTO - handles the SVM timeout worklist entries

PNSMTR - removes a WLE in the SVM timer worklist

PNSMWL - WL entry switch for SVM

C0SMIN/C0SMOUT - send/receive SM - this workcode is the subswitch for the SVM handler table (see table C1)

C0SMDISP - calls PNSDISP to send an SM

C0LINOP/C0LNINOP - calls PNLINE to enable or disable a line

COLNDA - handles replies from TIP for line disable requests by the SVM

CODLTCB - handles replies from TIP for delete TCB requests from SVM

COOVLDATA - handles the overlay data SM

COBFT
CODISABLE
COENABLE
COTMLDLT

Call PNRTN to continue processing TIP reply following PNAWAIT release of control

The following routines are called either from the first level (work code) of the main switch (above), or are called from the PFC/SFC decoding of the subcode.

PNSMDISP - sends an SM to the host or remote NPU

PNLLCNF - configures or deletes a logical link

PNCONFIGURE - common subroutine to configure a control block (LCB or TCB)

PNLNCNF - configures a trunk or line

PNTMLCNF - configures or reconfigures a TCB

PNDELETE - deletes a line

PNENABLE - enables a line

PNDISABLE - disables a line

PNLINE - handles line operational or line inoperative work codes

PNTMLDLT - deletes a TCB

PN1LLSTAT - formats the logical link status SM

PNLLSTAT - handles logical link status SM

PNCNTLN - counts trunks or lines

PNLCR - handles the count line request SM

PNSTATE - generates response code for a line status SM

PN1LNSTAT - formats the trunk/line status SM

PN2LSTAT - formats the trunk/line status SM for a single line

PNLNSTAT - handles line status SM

PNTNKSTAT - handles the trunk status SM

PN1TMLSTAT - formats the terminal status SM

PNTMLSTAT - handles the terminal status SM

PNBRDCST - handles broadcast SM (message to all terminals)

PN1BRDCST - handles the broadcast 1 SM (message to one terminal)

PNOVLOAD - processes overlay program loading

PNOVLDATA - processes overlay programs

PNOVLTMT - terminates an overlay program

PNFRCLD - processes the force load SM

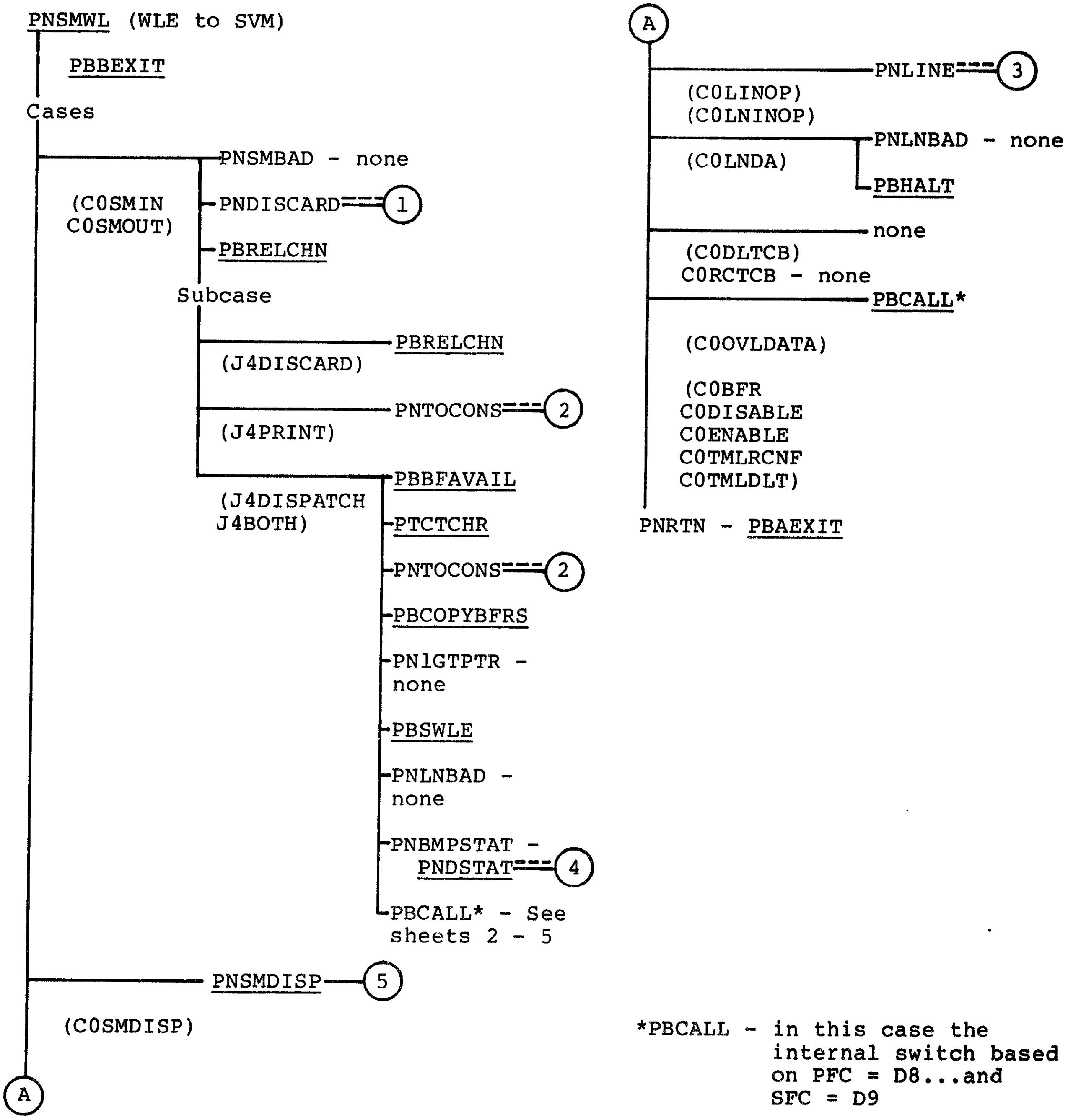
The following programs are called externally as SVM common programs for TIPS, the multiplex subsystem, etc.

PNPSTAT - generates the periodic statistics SM (one statistics block - next in the list)

PNDSTAT - generates the dump statistics SM (the specified statistics block)

PNSMGEN - generates an SM

WORKCODES



*PBCALL - in this case the
internal switch based
on PFC = D8...and
SFC = D9

(See appendix C)

Figure G-4. SVM TREES (sheet 1 of 8)

SWITCH THROUGH PBCALL and PFC/SFC

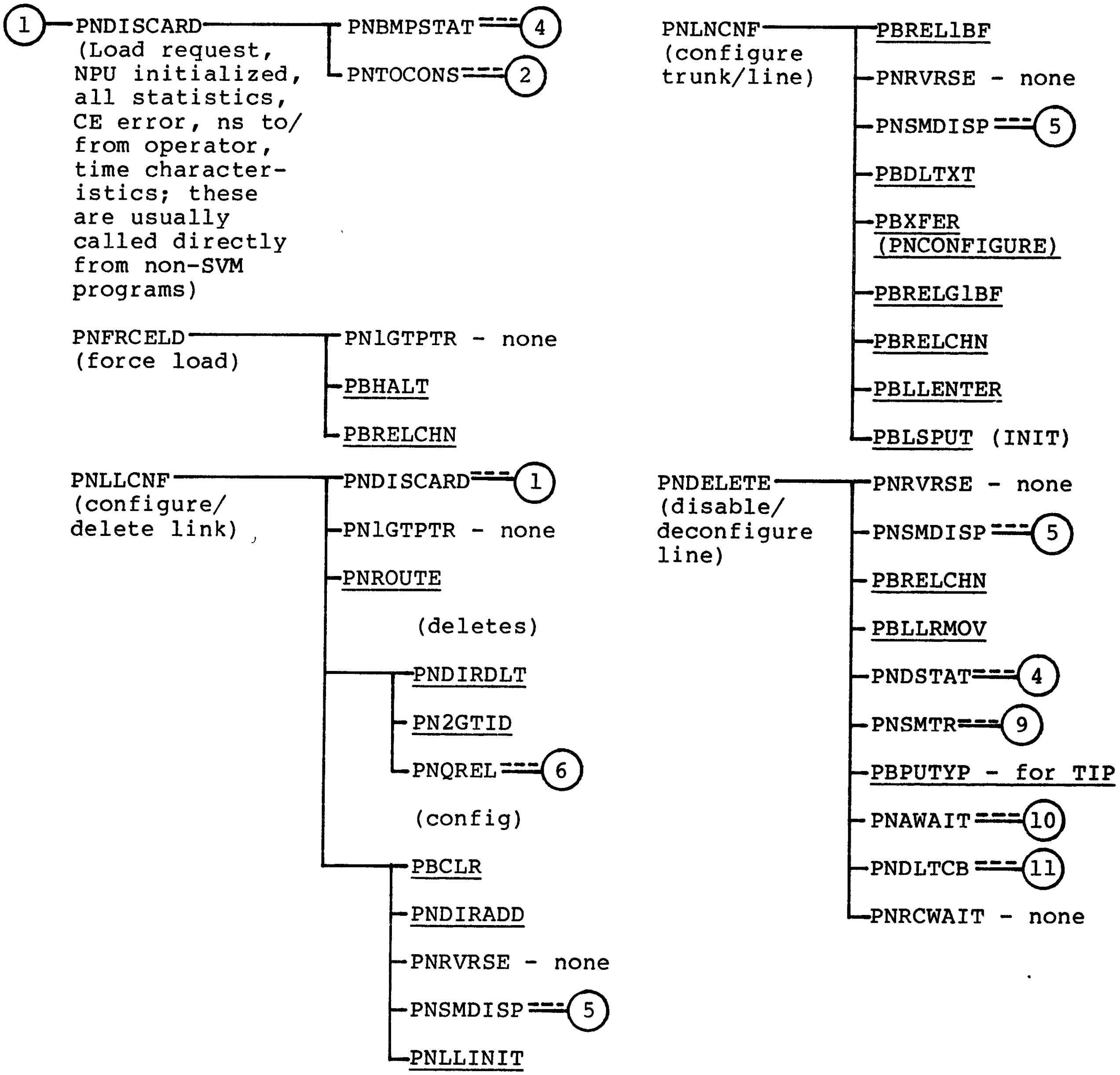


Figure G-4. SVM TREE (sheet 2 of 8)

SWITCH THROUGH PBCALL USING PFC/SFC

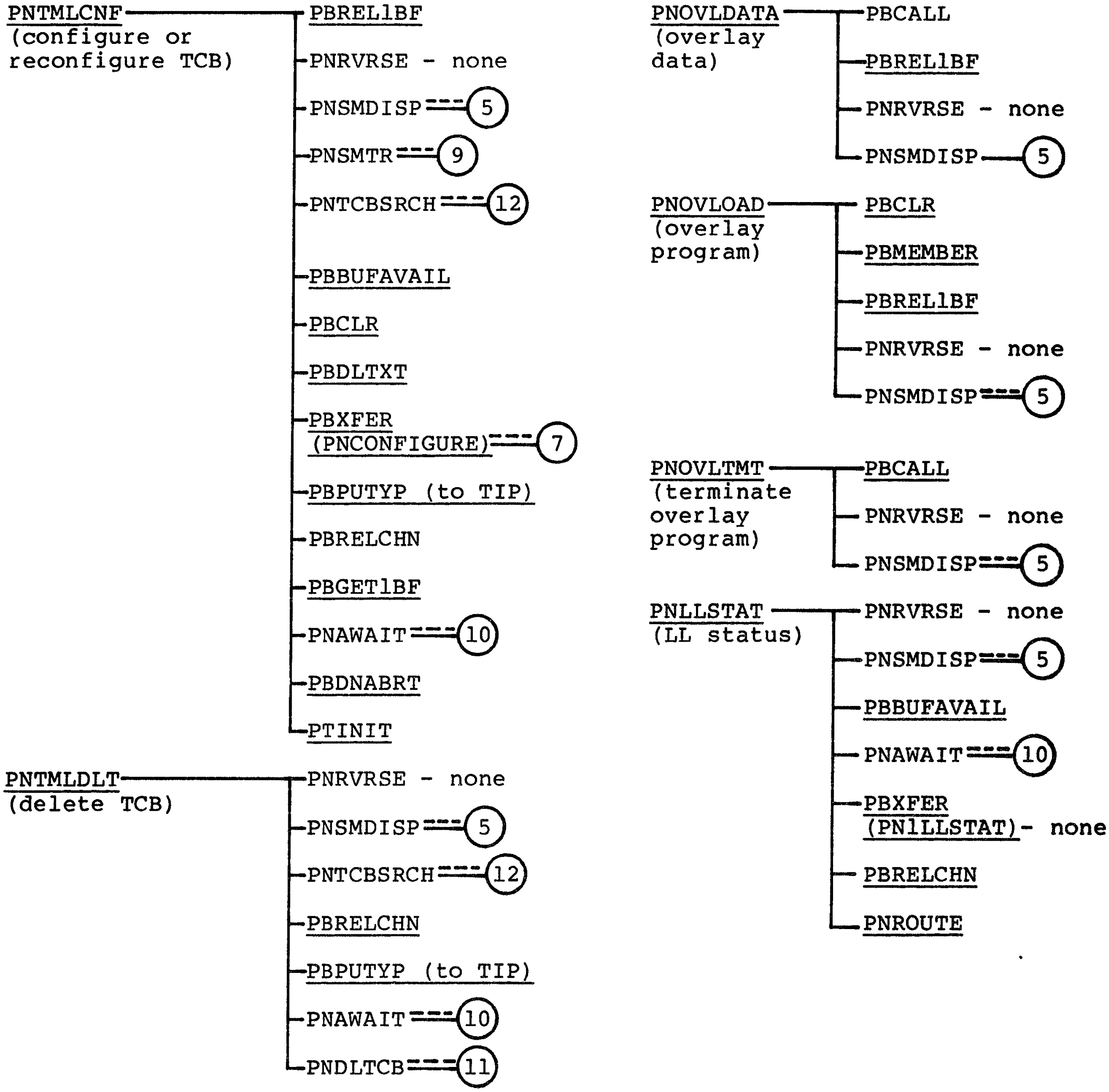


Figure G-4. SVM TREES (sheet 3 of 8)

SWITCH THROUGH PBCALL USING PFC/SFC

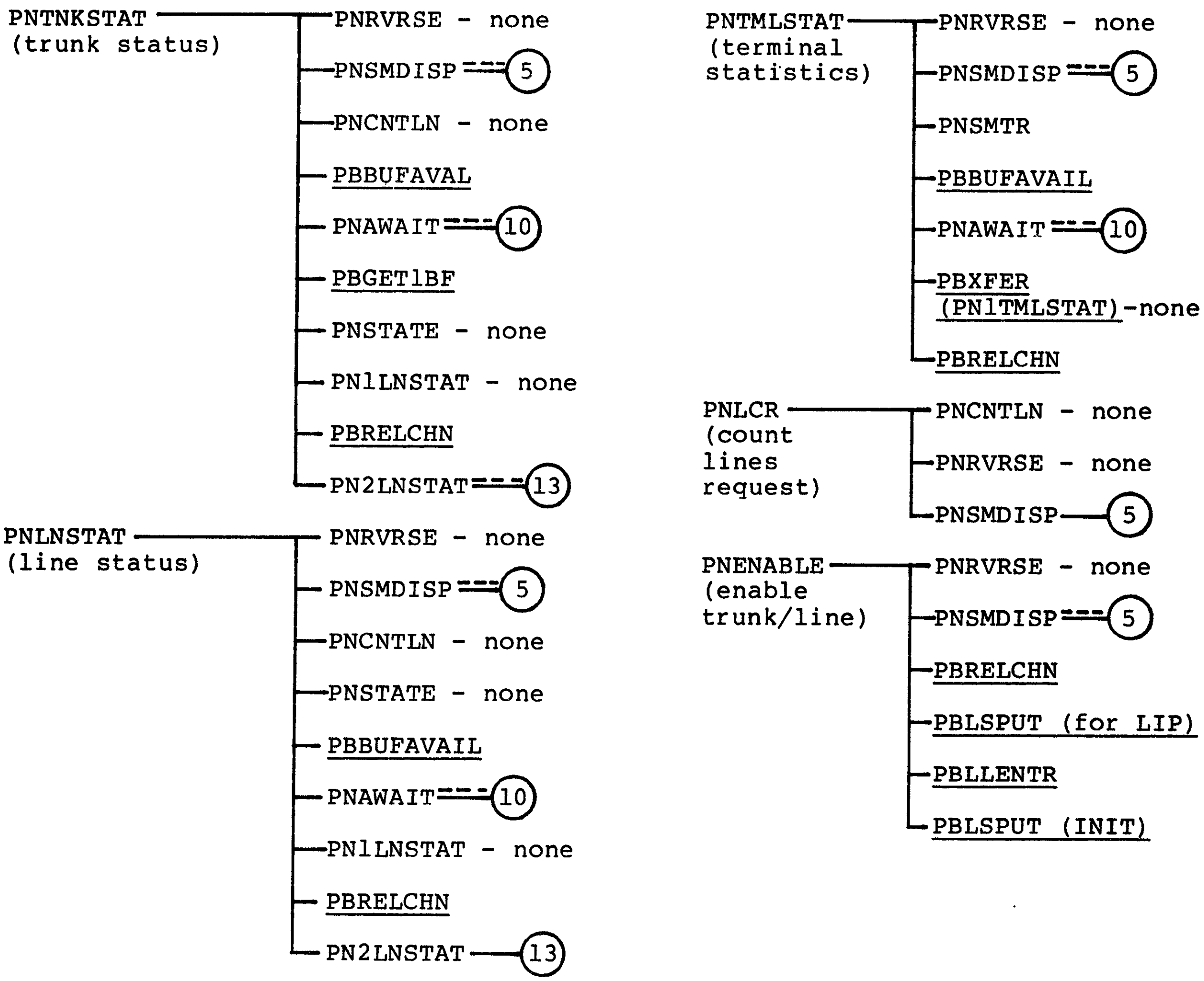


Figure G-4. SVM TREES (sheet 4 of 8)

SWITCH THROUGH PBCALL USING PFC/SFC

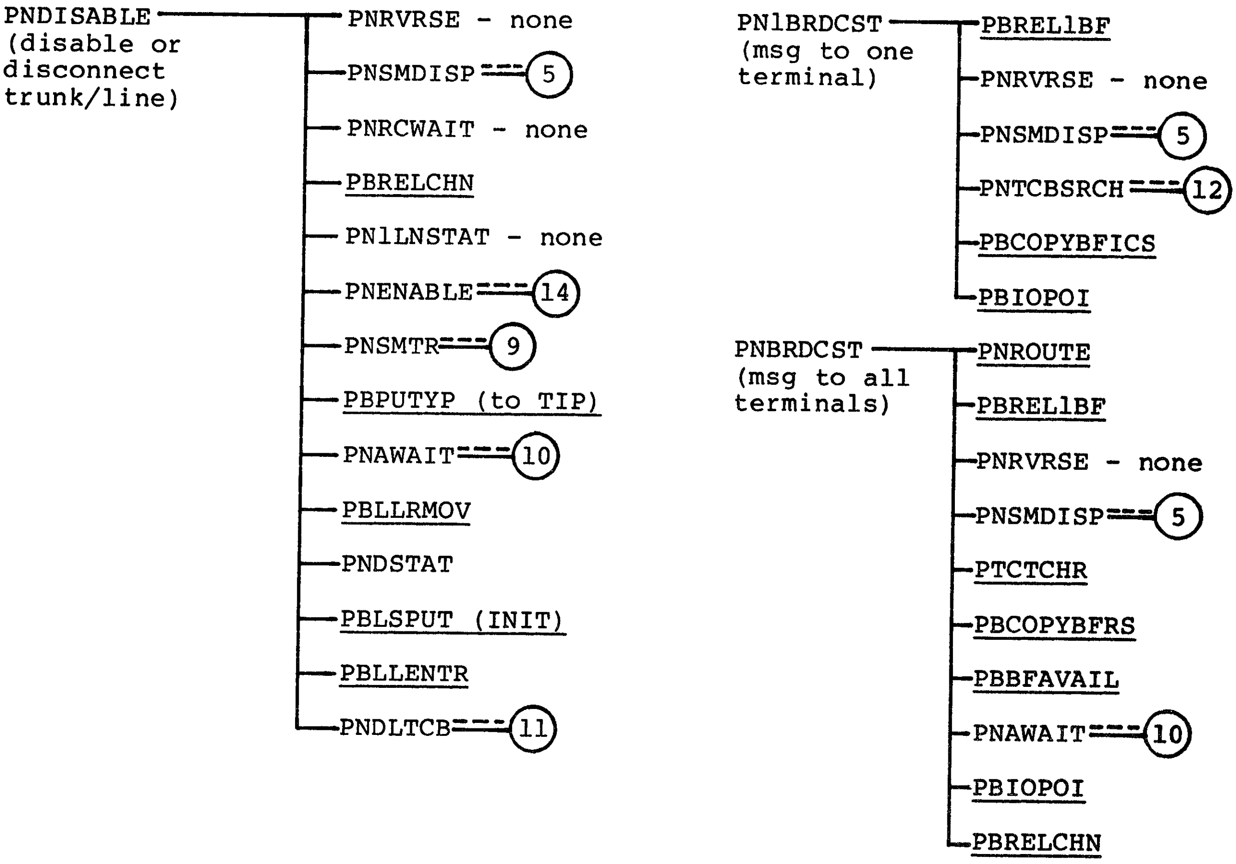


Figure G-4. SVM TREES (sheet 5 of 8)

ROUTINES EXCLUSIVELY FOR DIRECT EXTERNAL CALLS

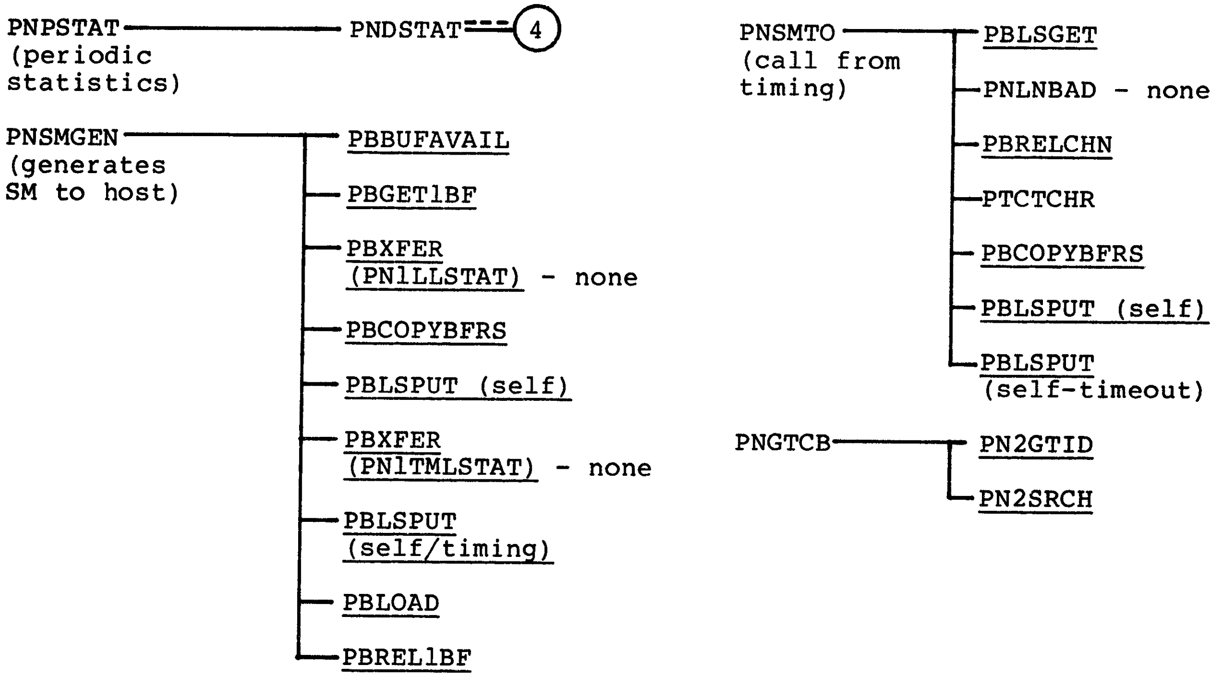


Figure G-4. SVM TREES (sheet 6 of 8)

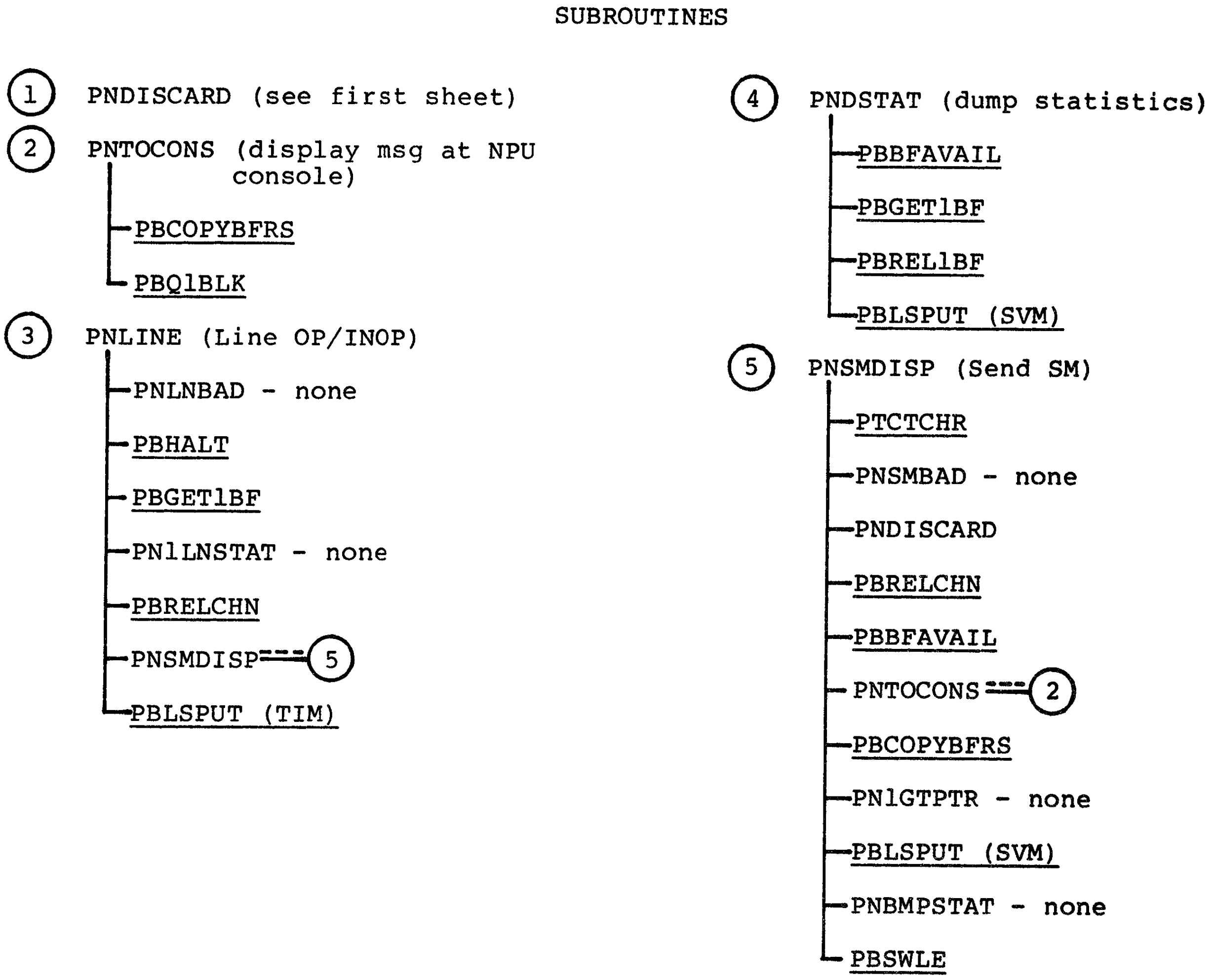


Figure G-4. SVM TREES (sheet 7 of 8)

SUBROUTINES

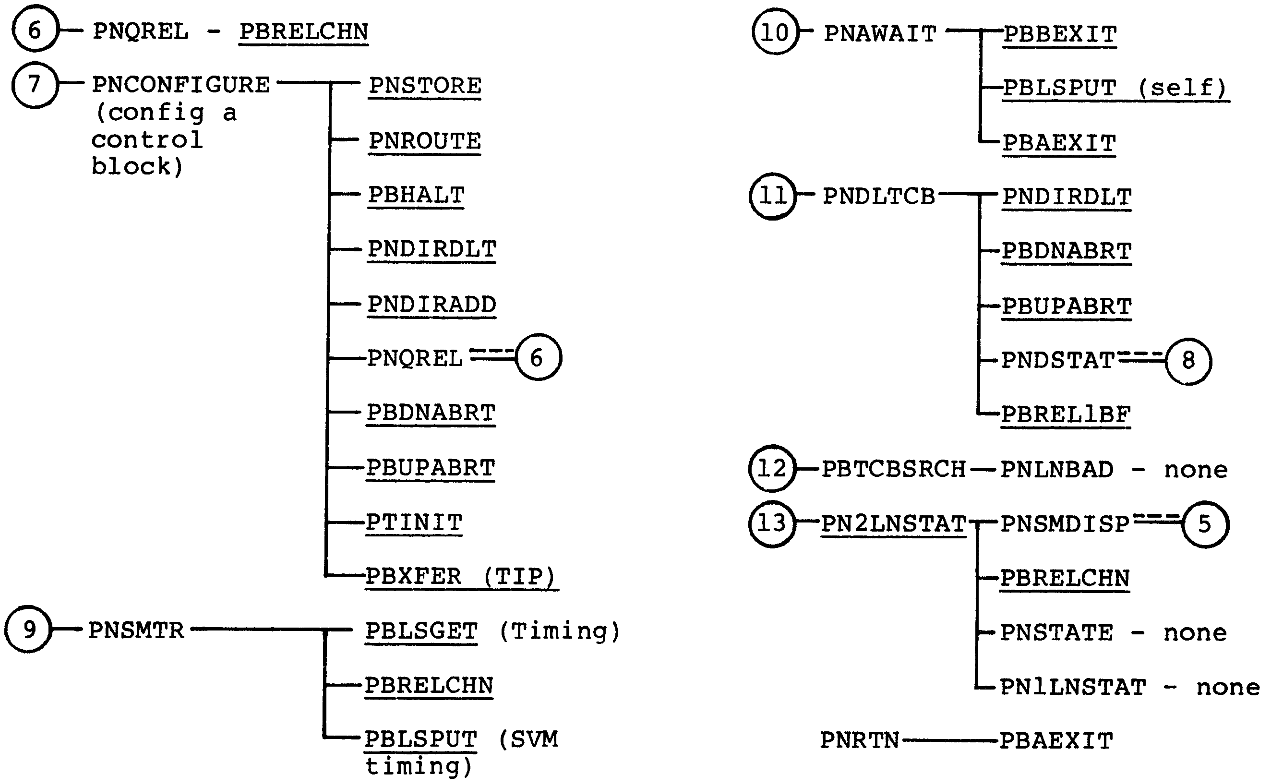


Figure G-4. SVM TREES (sheet 8 of 8)

PRINCIPAL DATA STRUCTURES

H

This appendix lists and describes the principal data structures in CCP. It is intended for use with a link edit on cross-reference listing.

Because PASCAL definitions can occur in three stages (types of structure, variables using these types, and values on constants assigned to type/variable fields), the tables discussed in this section are defined with the type definition. Mnemonics for variables assigned to the same fields are usually similar to the type definition. The listing should be consulted for the correct variable name. Wherever the variable name is frequently used, this name is also given in this appendix.

In some cases (such as service messages) the data structures are already well described elsewhere. In these cases, the reader is referred to another location in this manual or in The CCP Reference Manual.

CONTENTS

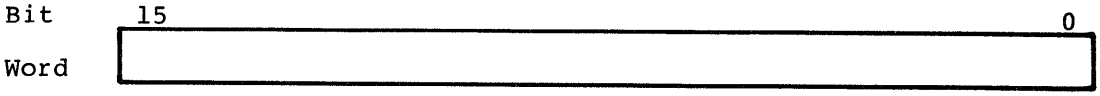
NOTE

For tables with two or more variants, the TIP writer can select fields from any variant so long as incompatible fields in the same word are not used for a single process. (Caution: The writer must know all the programs (within the TIP as well as called directly or indirectly from the TIP) that use the field.) Use of fields from several variants of a table (such as the TCB) is common throughout CCP.

BITS, WORDS AND POINTERS

BIT DEFINITION

The following labels define the bit structure for NPU words.



Mnemonic	Bits	Decimal Range
B01BIT	0	0-1
B02BITS	0-1	0-3
B03BITS	0-2	0-7
B04BITS	0-3	0-15
B05BITS	0-4	0-31
B06BITS	0-5	0-63
B07BITS	0-6	0-127
B08BITS	0-7	0-255
B09BITS	0-8	0-511
B010BITS	0-9	0-1023
B011BITS	0-10	0-2047
B012BITS	0-11	0-4095
B013BITS	0-12	0-8191
B014BITS	0-13	0-16383
B015BITS	0-14	0-32767

The bit elements that make up the 16-bit NPU word are as follows:

```
ELEMENTS = (BIT 0, BIT 1, BIT 2, BIT 3, BIT 4, BIT 5, BIT 6,
            BIT 7, BIT 8, BIT 9, BIT 10, BIT 11, BIT 12,
            BIT 13, BIT 14, BIT 15)
```

Bit 0 is least significant bit; bit 15 is most significant bit.

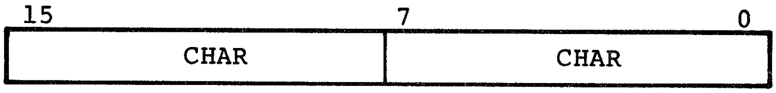
WORD STRUCTURES

Mask Word

```
SETWORD = SET OF ELEMENTS
```

Bit set allows corresponding bit to be inspected (logical AND)

Characters (2/Word)



Array of up to 131K characters

```
BOCHRARAY = PACKED ARRAY (B015BITS) OF CHAR;
```

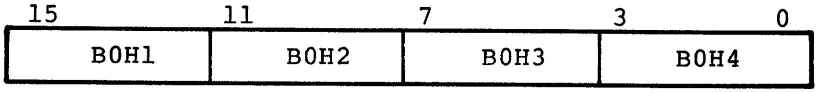
Integers (1/Word)

Word array of 65K words

```
BOINTARAY = ARRAY (BO15BITS) OF INTEGER:
```

Four Hexadecimal Numbers/Word

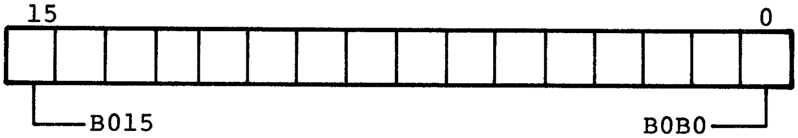
```
BOHEX = PACKED RECORD

        BOH1, BOH2, BOH3, BOH4:  BO4BITS

        END:
```



Flag Word

Sixteen flags are packed in one word.

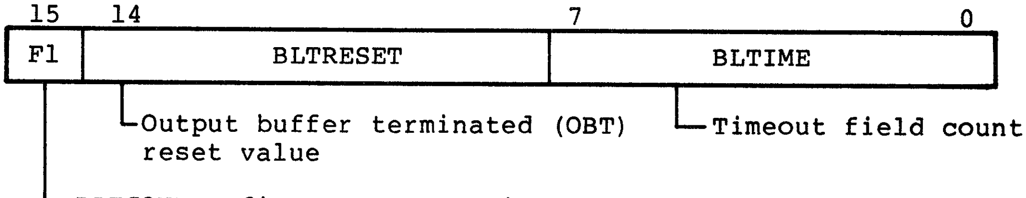
```
BOFKAGS = PACKED RECORD

          BOB15, BOB14, BOB13, BOB12, BOB11, BOB10, BOB9  BOB8, BOB7,
          BOB6, BOB5, BOB4, BOB3, BOB2, BOB1, BOB0:  BOOLEAN
```



Sixteen flags with mnemonic corresponding to bit position of flag in word.

Line Timing

BZLTIME has three values, packed as shown. The count increments are in half seconds.



Output buffer terminated (OBT) reset value

Timeout field count

BLTCONT - firmware contention flag, used for 2552 processing where both multiplex and base sides have a request waiting

MASKS

The principal masks are for single characters and single bits.

Character Masks

Left byte, BY0MSK

15	7 0
All 1's	All 0's

Right byte, BY1MSK

15	7 0
All 0's	All 1's

Bit Masks

Hexadecimal values are 1, 2, 4, 8, 10, 20, 40, 80, 100, 200, 400, 800, 1000, 2000, 4000, 8000.

POINTER DEFINITIONS (B0INTPTR)

Pointers are all one word (INTEGER) type.

Pointer	Control Block or Buffer	Meaning
B0QPTR	B0CBENT	Queue control block pointer (QCB)
B0BUFPTR	B0BUFFER	Buffer pointer for general buffer
B0HEXPTR	B0HEX	Hex pointer (location in hexadecimal)
B0REGPTR	B0REGSAVE	Register save area pointer
N0LCBP	NCLCB	Multiplex LCB (MLCB) pointer
BZLCBP	BZLCB	LCB pointer

VARIABLE WORD DEFINITIONS

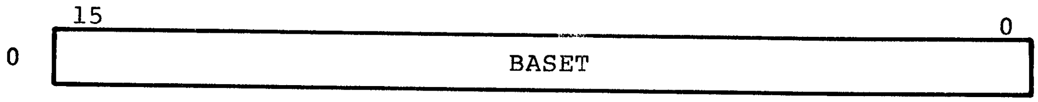
The universal word overlay has many variations. Each variation is of the most frequently used type. Thus, by overlaying the universal overlay over a variable, the variable may be accessed in a variety of formats.

	15	7 0
0	BACHARS CHAR 1	BACHARS CHAR 2

type: CHAR: length 1 to ALFALENG

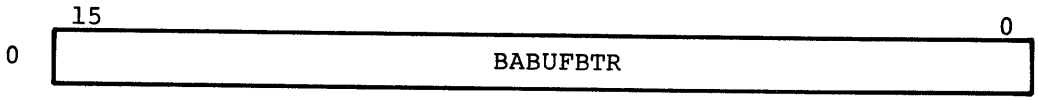
	15 0
0	BABOOL

type: B0FLAGS - up to 16 flags

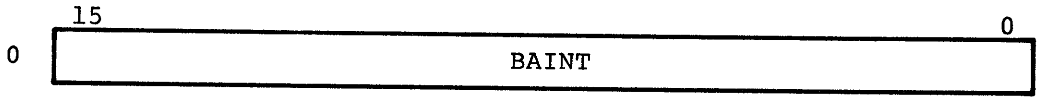


type: SETWORD - mask

type: SET of 0 through F_{16}

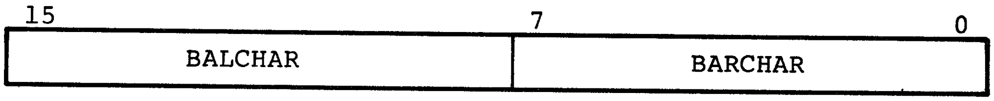


type: B0BUFPTR, buffer pointer



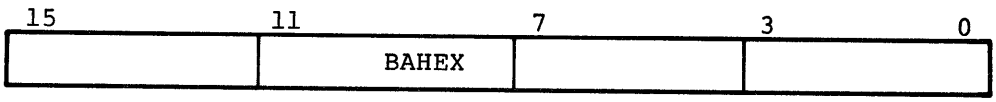
type: INTEGER, full word integer

type: B0WLCODES, worklist code



type: CHAR, left and right characters

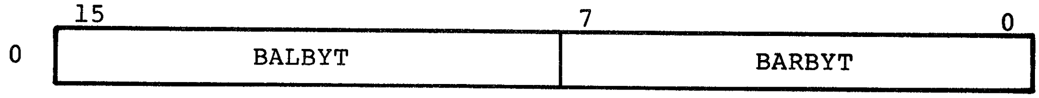
type: B0HEXPTR, hexadecimal pointer



type: B0HEX, 4 hexadecimal digits

type: B00INTPTR, integer pointer

type: B0REGPTR, register pointer



type: B08BITS, integers in left and right bytes

type: BOOLEAN, uses only bit 0

type: B0QPTR, queue pointer

type: B0BUFSIZES. Index to size of buffer (1, 2, 3, 4) for the network. Nominal sizes: 8, 16, 32, 64 correspond to values 1, 2, 3, and 4.

type: B0WKLSTS. Worklist index. Entries in the monitor table as shown in section 5. Uses only bits 0 through 4.

type: N0LTYP. Line type. See table C-3. User only bits 0 through 3.

	15	0
0	BALINO	

type: NOLINO. Line number. Used to index LCBs.

	15	0
0	BALCBP	

type: BALCBP. LCB pointer.

	15	0
0	BATTYP	

type: NOTTYP. Terminal type. See appendix C.

	15	0
0	BA1CHAR	

type: CHAR. Right character. Uses full word with character right justified.

	15	6	3	0
0	BAPAD	BAINDEX	BABITPOS	

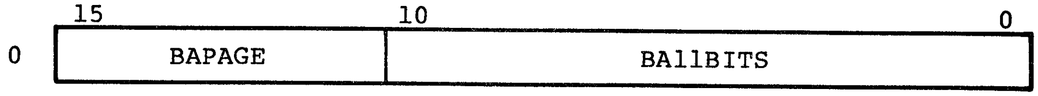
type: Three fields together make a pointer (BACHROVLY) to an ASCII character in the ASCII/binary conversion table. See appendix A of The CCP Reference Manual.

	15	11	8	0
0	BA15T012	BACPOC	BACPLN	

type: Two fields (left most field is spare) called BACPOW. BACPOC is the coupler order word code and BACPLN is block length. Used by the HIP for threshold checks and for computing the number of buffers needed for an input block.

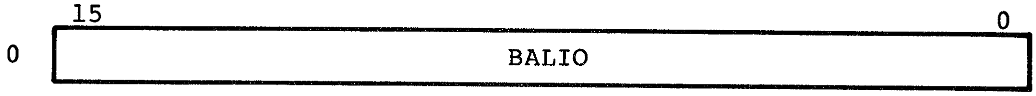
	15	6	0
0	BASP9	BA7BITS	

type: fields: variants 24 and 25 are used together as an 18-bit address. BA7BITS is upper 7 bits, BA11BITS is lower 11 bits of address.



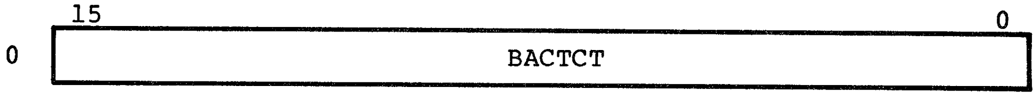
BAPAGE is the page number: range 0 through 31.

type: B0PGM. Used by TUP to index into the OPS monitor table. Uses only bits 0 through 4.



type: J0LIO. Console logical I/O index. Uses only bits 0 through 3.

type: BLKTYPE. Block type (BT) field in the block header. Uses lower 4 bits.



type: NICTCT. Entry in character transmission table (NICTCY).

MULTIWORD ASCII SET

JSASCIISET = ARRAY (303BITS) OF SET OF B04BITS:

This is an eight-column, 16-row array of 8-bit characters. The 8 by 16 array completely defines the full 128-character set (as well as the 96-character subsets) for ASCII. See appendix A of The CCP Reference Manual.

HARDWARE RELATED TABLES

This subsection describes hardware registers and lines which are not handled by the multiplex subsystem.

REGISTER DESIGNATION

This sequence defines the principal 255X hardware registers: R1-R4, Q, A, I, M, overflow. Extra is a dummy register.

```
B0REGISTERS = (B0EXTRA, B0R1, B0R2, B0R3, B0R4, B0Q, B0A,
               B0I, B0M, B0OFLOW)
```

REGISTER SAVE AREA

B0REGSAVE = ARRAY (B0REGISTERS) OF INTEGER

	15	Register Saved 0
Word 0	B0REGSAVE	–
	B0REGSAVE (ARRAY)	R1 ⋮ R4 Q A I M
9	B0REGSAVE ELEMENT 10	Overflow

COUPLER RELATED CONSTANTS

The coupler codes used by the various coupler registers are described in section 7.

Mnemonic	Value	Meaning
Coupler Functions		
	(hexadecimal)	
ACPICS	50	INPUT COUPLER STATUS
ACPIOW	60	INPUT ORDERWORD
ACPONS	48	OUTPUT NPU STATUS
ACPOBL	58	OUTPUT BUFFER LENGTH
ACPCLR	0C	CLEAR COUPLER
ACPOMA	6C	OUTPUT MEMORY ADDRESS
ACPRMA	10	READ MEMORY ADDRESS REGISTER
	(end hexadecimal)	
Data Transfer Status Commands		
AIDLE	1	
AAOUTPT	3	IDLE STATUS
AAREADY	4	OUTPUT DATA AVAILABLE
AANREADY	7	READY TO ACCEPT OUTPUT DATA
AINPSB	13	NOT READY TO ACCEPT OUTPUT DATA
AINPLB	14	INPUT AVAILABLE - SMALL BLK OR MSG
Coupler Condition States		
A0PT0	0	IDLE STATE
A0PT1	1	IDLE INQUIRY SENT
A0PT2	2	INITIATED INPUT
A0PT3	3	INITIATE OUTPUT

Mnemonic	Value	Meaning
A0PT4	4	OUTPUT IN PROGRESS
A0PT5	5	READY FOR OUTPUT DELAY
A0PT6	6	NOT RDY FOR OUTPUT DELAY

Coupler Timeout Values

AIDLETO	3	IDLE TIMEOUT = 1 TO 1 1/2 SECONDS
ADEADTO	60	DEADMAN TIMEOUT = 30 SECONDS

Q and A Register Load Area, NGAQLT

One word is provided for commands (Q register) and two variants are provided for data/subcommands (A register). The console used the A/Q channel for I/O. These are used only for the command driver.

Command (Q)

	15	7 0
0	NGPORT - Port Number	NGCMD - I/O device command

Subcommand (A)

	15	7 0
0	NGLTYP - Line Type	NGCNT - Count of characters being sent

Universal Overlay

	15 0
0	NGINT - Integer

Hardware Lines and Associated Software Priorities

Hardware Line No.	Software Priority	Description of Interrupt
0	P1	Internal (parity and protect, power)
1	P6	Teletype (NPU console)
2	P2	Multiplex loop error
3	P3	Multiplex Level 2
4	P16	1742-30 line printer (for console - not used)
5	P5	Spare

Hardware Line No.	Software Priority	Description of Interrupt
6	P7	Coupler
7	P8	Spare
8	P9	Real-time clock
9	P10	1742 line printer (for console - not used)
10	P11	Spare
11	P12	Spare
12	P13	MLIA ODD (parallel for all NPU ports)
13	P14	MLIA input line frame (parallel for all NPU ports)
14	P15	Spare
15	--	Hardware breakpoint
--	P17	OPS level programs

JKMASK defines the array of 17 priority level masks (B0PR1LEVEL) associated with these interrupts. Priority 1 is highest; priority 17 is not associated with any interrupt driver.

NPU CONSOLE

The NPU console has two levels of data structures.

- The request packet from the user (logical request packet, LRP) establishes the message transfer parameters. The LRP is converted to a physical request packet (PRP) by the console driver so that the user does not need to concern himself with terminal physical characteristics.

- The device controller table provides parameter storage for the A/Q transfer between NPU and console device. One such controller table is provided for each device associated with the NPU console.

In addition, the console driver for the device must:

- recognize the A/Q line responses

- provide the controller functions in the form recognized by the controller (bits set)

- recognize special characters that are used by the console for mode or message control

LOGICAL/PHYSICAL I/O REQUEST PACKET, JCPACKET

These two packets share the same format. The packets are used to pass requests to the NPU console and are the logical equivalent of the LCB/TCB for remote terminals.

<table>
<tr><th></th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>0</td><td colspan="8">JCCOMMAND - I/O command</td><td colspan="8">JCCOMPL - I/O completion code</td></tr>
<tr><td>1</td><td>F1</td><td>F2</td><td>F3</td><td>F4</td><td>F5</td><td>F6</td><td>F7</td><td>F8</td><td>F9</td><td>F10</td><td>F11</td><td>F12</td><td>F13</td><td>F14</td><td>F15</td><td>F16</td></tr>
<tr><td>2</td><td colspan="4">JCLIO-logical I/O function</td><td colspan="12">JCPD - Physical device code (bits 7-0 only)</td></tr>
<tr><td>3</td><td colspan="16">JCUSERWD - User word</td></tr>
<tr><td>4</td><td colspan="16">JCPOINTER - Pointer to tag or first buffer</td></tr>
<tr><td>5</td><td colspan="16">JCBUFSZE - Pointer to buffer control block</td></tr>
<tr><td>6</td><td colspan="4">JCLIO - Worklist code</td><td colspan="6">JCUSRCODE - User program code</td><td>F17</td><td colspan="5">JCRESULT - I/O result code (bits 3-0 only)</td></tr>
<tr><td>7</td><td colspan="4">JCRETRYCNT - Retry count</td><td colspan="4">JCRECDSZE - Record size</td><td colspan="8">JCBLKSZE - block size (bits 3-0 only)</td></tr>
<tr><td>8</td><td colspan="16">JCSTATUS - Physical device status</td></tr>
</table>

F1 - JCRELBUFLG, release output buffers

F2 - JCRELPRFFLG, release physical request packet (PRP)

F3 - JCNOBUFLG, I/O not in buffer

F4 - JCSP1, not used

F5 - JCPRIFLG, priority output

F6 - JCTRANSPFLG, transparent data

F7 - JCGETBUFLG, get buffers for input

F8 - JCRESETFLG, reset wait I/O bit

F9 - JCCHAINFLG, chain messages

F10 - JCSTACKFLG, stack this completion request

F11 - JCENDSTACKFL, end of completion stack

F12 - JCBATCHFLG, batch this request

F13 - JCENDBATCHFLS, last request in batch

F14 - JCSP2, not used

F15 - JCIMMEOFLG, perform immediate output

F16 - JCCOMFLG, call PBDRCOMPL, the console common driver completion routine

F17 - JCOPCODE, worklist OPScode

The following constant values are assigned to the LRP/PRP fields indicated.

Mnemonic	Value	Meaning		Type
J3READ	0	Console read		I/O commands
J3WRITE	1	Console write		
J3NOCOMPL	0	Failed to complete		Completion codes (JCCOMPL)
J3	0	Not used		
J3ACCEPTED	0	LRP accepted		Result codes (JCRESULT)
J3REJECTED	1	LRP rejected		
J3ERR1	2	All retries attempted		
J3ERR2	3	More retries can be attempted		
J3COMPLETE	4	LRP completed		
J1PRIWL	1	Priority worklist	two console queues	Driver worklist priorities
J1REGWL	0	No priority worklist		

Functions (JCLIO field) are:

Mnemonic	Value	Console Mode
J0LIO = (J2TUPOUTPUT,	1	SUPERVISORY INPUT
J2SUPOUT,	2	SUPERVISORY OUTPUT
J2ALM,	3	ALARMS
J2REP,	4	REPORTS
J2ORD,	5	ORDERWIRE
J2DIAG,	6	DIAGNOSTICS
J2TUPINPUT,	7	TUP INPUT
J2TUPOUTPUT,	8	TUP OUTPUT
J2TUPDUMP,	9	TUP DUMP
J2SNP1,	10	SNAPSHOT 1
J2SNP2,	11	DUMP REGISTERS
J2SNP3,	12	PRINT BREAKPOINT ADDRESS
J2SPARE,	13	SPARE
J2QUICK,	14	QUICK I/O
J2WS1,	15	WRAP-SNAP 1
J2LAST);	-	DUMMY

Device Controller Table, JACONTROLLERTABLE

The device controller table is used by the modules comprising the NPU drivers. One controller table is used for each console device.

	15 0
0	JASTATUS - Physical device status
1	JACRUREQ - Pointer to current I/O request
2	JAIOBUF - Pointer to I/O buffer
3	JAINPROGFLG - I/O in progress flag
4	JABUFXZE - Pointer to I/O buffer control block
	⋮

5	JACHRCNT - I/O character count
6	JATIMER - I/O timer - half seconds
7	JATIMOUT - Timeout count - half seconds - 5 minute overflow
8	JAREJECT - Rejected transfer count
9	JABADINT - Bad interrupts count
10	JARETRY - Retry I/O count
11	JAQVALUE - Q register contents for last I/O transfer
12	(data) JAAVALUE - A register contents for last I/O count
13	JAREADFLG - Last I/O type flag; 1 = read, 0 = write
14	JAMASK - Mask out device for PBSTARTIO
15	JAIOWL - Driver worklists, used for PBLSGET and PBLSPUT
16	JAIOWL (ARRAY) JAIOWL ELEMENT 2 } I/O worklist
17	JAAUTOFLG - Augomatic output flag
18	JAFRSTFLG - First character of message plan
19	JAINTFLG - Message interrupted flag
20	JAMODEFLG - Mode change flag
21	JACHFLG - Console input message flag
22	JACURIBP - Current input buffer pointer
23	JAOLDIBP - First input buffer pointer
24	JAQCHOSEN - Queue chosen
25	JADROPQ - Interactive queue
26	JAERRCNT - Error count

Words 17 through 26 are used only for the display/keyboard.

JAINPROGFLG - Only bit 0 is valid
JAAVALUE - Only bits 7 through 0 are valid
JAREADFLG - Only bit 0 is valid
JAIOWL - Only bits 4 through 0 are valid
JAAUTOFLG - Only bit 0 is valid
JAFRSTFLG - Only bit 0 is valid
JAINTFLG - Only bit 0 is valid
JAMODEFLG - Only bit 0 is valid
JACHFLG - Only bit 0 is valid

I/O Response Codes, J0IORESP

These are hardware responses checked by the console I/O drivers when reading or writing a character.

J0XREJECT	1	EXTERNAL REJECT
J0IREJECT	2	INTERNAL REJECT
J0REPLY)	3	REPLY

Director (Controller) Function Codes for 1713 TTY

BIT 15, 14 - BAUD RATE SELECTOR; 0 = 110, 1 = 300, 2 = 1200, 3 = 9600
BIT 13 - DISCONNECT PRINTER
BIT 12 - 8-BIT WORD
BIT 11 - DE-SELECT PARITY
BIT 10 - CONNECT PRINTER
BIT 9 - SELECT READ MODE
BIT 8 - SELECT WRITE MODE
BIT 7 - NOT USED
BIT 6 - ADT MODE
BIT 5 - NOT USED
BIT 4 - INTERRUPT ON ALARM
BIT 3 - INTERRUPT ON END-OF-OPERATION
BIT 2 - INTERRUPT ON DATA
BIT 1 - CLEAR INTERRUPT
BIT 0 - CLEAR CONTROLLER

Multiple functions are accepted by the controller; they are defined as follows:

TTYCLR, - TTY clear interrupt, controller
TTYREAD, - TTY select read mode, alarm interrupt, data interrupt
TTYRITE, - TTY select write mode, no interrupt
TTYWRITE, - TTY select write mode, alarm interrupt, data interrupt
TTYEOP: CHAR: - Clear interrupt select EOP interrupt

Special TTY (Console Keyboard) Characters

Character Definition	Keyboard Character/Use
J1CP,	CARRIAGE RETURN
J1LF,	LINE FEED
J1CTLH,	CONTROL H - TREATED AS BACKSPACE
J1BCKSPCE,	BACKSPACE
J1TUPCAN,	/TUP MESSAGE EOM
J1TUPCAN,	QUESTION MARK TUP CANCEL INPUT
J1ENTERTUP,	CONTROL A ENTER TUP MODE
J1LVETUP,	CONTROL D LEAVE TUP MODE
J2ENTERMP,	ESCAPE ENTER MAINT PANEL MODE
J2LVEMP,	LEAVE MAINT PANEL MODE
J1ICR,	REPLACE WITH CR CONTROL SHIFT N,
J1ILF,	REPLACE WITH LF CONTROL SHIFT M,
J1IDISCARD,	DISCARD CONSOLE INPUT
J1SYSEOM: CHAR:	CONTROL D SYSTEM EOM

Halt Codes

The NPU halt message is sent to the NPU console. The halt codes are described in The CCP Reference Manual.

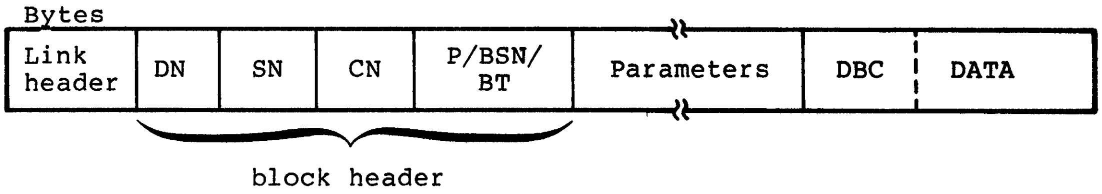
BLOCK PROTOCOL

The block protocol defines the byte structure used to transmit block between the host and the terminal mode of the NPU. Blocks occur in buffer with one or more changed buffers comprising the block. Each buffer in the chain requires a buffer header and (except for the last or only buffer case) a buffer pointer for chaining. All other bytes can be used for the message.

See section 6 for block protocol discussion

BLOCK PROTOCOL CONSTANTS

The block header occurs at the start of the buffer, following the bytes reserved for the buffer header (four bytes) and the two bytes reserved for a link header.



BT - block type; BT uses bits 3-0 only.
DBC - data block clarifier; if present, it is the first data byte.

BLOCK TYPE

Block Type Mnemonic	Value	Block Type Meaning	
BT - bits 3 through 0 of P/BSN/BT byte			
HTBLK	1	Block	data transfer blocks
HTMSG	2	Message	
HTBACK	3	Back - acknowledgment	
HTCMD	4	Command - used for service messages	
HTBREAK	5	Break	Communications start, stop, restart
HTSTOP	6	Stop	
HTSTRT	7	Start	
HTRESET	8	Reset	
HTINIT	9	Initialize	
HTACTL	15	Data assurance (word for LIP only)	
SUBBLOCKS	0		
HTCLR	1	Clear	
HTPRST	2	Protocol reset	
HTREGL	3	Regulation	
HTLINIT	4	Link initialization	
HTLIDLE		Link idle	

BLOCK BYTE SEQUENCE

Byte position assumes buffer header and link header. Bytes are numbered starting at 1 in the upper byte of word 0 of the buffer.

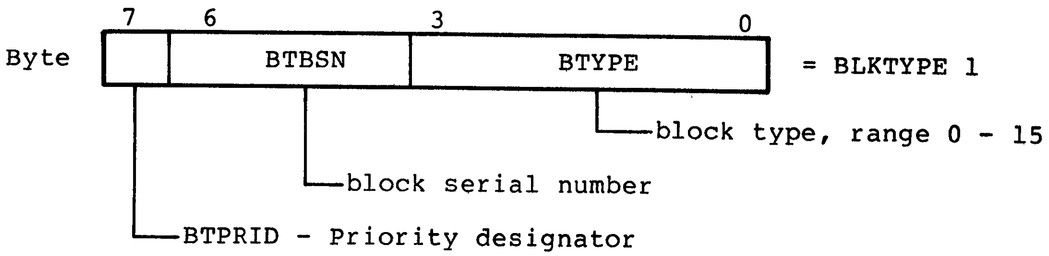
Mnemonic	Byte Position	Byte Use	
DN	6	Destination node	
SN	7	Source node	
CN	8	Connection number	
BTPT	9	Priority/serial number/block type	
P1	10	Parameter 1 (DBC if data instead of parameter)	
P2	11	Parameter 2	Data may start at any parameter position; it must start at the byte following the last parameter used.
P3	12	Parameter 3	
P4	13	Parameter 4	
P5	14	Parameter 5	
P6	15	Parameter 6	
P7	16	Parameter 7	
P8	17	Parameter 8	
P9	18	Parameter 9	
P10	19	Parameter 10	
P11	20	Parameter 11	
P12	21	Parameter 12	
P13	22	Parameter 13	
P14	23	Parameter 14	
P16	25	Parameter 16	
P18	27	Parameter 18	
P20	28	Parameter 20	
P24	33	Parameter 24	
FBYTE	DN	FCD for first byte of data	
BLOCK	DN	FCD of first byte of block header	
DBC	P1	Data block character	
DATA	P1	FCD of first byte of data (may be DBC)	

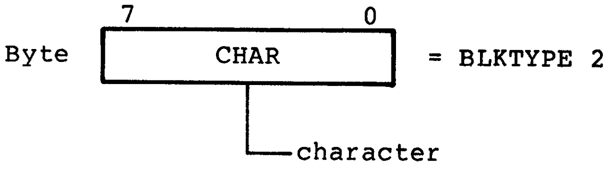
FIELD BIT START POSITION IN BYTE

Mnemonic	Binary Value	Starting Bit
FS0	1	bit 0
FS1	2	bit 1
FS2	4	bit 2
FS3	8	bit 3
FS4	10	bit 4
FS5	20	bit 5
FS6	40	bit 6
FS7	80	bit 7

BLOCK TYPE (BT) TYPE

The fourth byte of the block header can have two forms:





DATA BYTES

Mnemonic	Position of byte in data part of message	Meaning
DBC	P1 = 1	Data block clarifier (control flags for the data which follows)
TIME	P2 = 2	
STMP	P3 = 3	stamp
LVLN	P4 = 4	level numbers
DATA	P5 = 5	data bytes (can begin at positions 1 through 5)
DWORD1	6	
DWORD2	7	
DWORD3	8	
DWORD4	9	
DWORD5	10	
DWORD6	11	

DATA BLOCK CLARIFIER, DBDBC

The DBC is often used as the first byte of data in a message. In the definition, it is right justified in a computer word. Six DBC variants are provided.

Character

	15	7 0
0	DBDM1 - Not used	DBCHAR - Character

Downline DBC

	15	7	6	5	4	3	2	1	0
0	DBDLFILL	F1	F2	F3	F4	F5	F6	F7	F8

F1 - DBDLS1,
F2 - DBDLS2,
F3 - DBDLS3
F4 - DBDLS4, spare
F5 - DBDLFE, format effectors used
F6 - DBDLXPT, transparent data
F7 - DBDLS5, spare
F8 - DBDLAUTO, autoinput block

Upline DBC

	15	7	6	5	4	3	2	1	0
0	DBULFILL	F9	F10	F11	F12	F13	F14	F15	F16

F9 - DBULS1, spare
F10 - DBULS2, spare
F11 - DBULS3, spare
F12 - DBULS4, spare
F13 - DBULS5, spare
F14 - DNULXPT, transparent data
F15 - DBULCAn, cancel data
F16 - DBULPERR, parity error

	15	5	3	2 0
0	DBSP1 - not used	DBCCF	F17	DBDBT

DBCCF - code conversion
F17 - DBBSF, backspace present
DBDBT - data clarifier

Mode 4, transparent data

	15	7	6	5	1	0
0	PDPM1 - not used	F18	F19	PDSP1 - not used	F20	F21

Flags:

F18 - PDPRUB, physical record unit block
F19 - PDBANB, banner block
F20 - PDEOI, message contains an EOI
F21 - PDXPAR, transparent data

HASP TIP

	15	7	6	5	4	3	2	1	0
0	DBF1 - not used	F22	F23	F24	F25	F26	F27	F28	F29

F22 - DBPRUB, physical record unit block
F23 - DBBANNER, banner message
F24 - DBSP2, not used
F25 - DBSP3, not used
F26 - DBSP4, not used
F27 - DBSP5, not used
F28 - DBEOI, block contains EOI or EOR
F29 - DBCXPT, transparent data

DIRECTORIES/INTERNAL PROCESSOR/COMMON TIP ROUTINES

The internal processor includes the POIs and various switching routines. The routing routines use the LCBs as directories; the routines also use directories built in type 1 and type 4 tables. See section 6 for routing and POI descriptions.

TYPE 1 AND TYPE 4 TABLES

Type 1/Type 4 Table Entries, BRDIRCTRY

These are indexed tables with a pointer associated with each index. Two words/entry: word 1 has the index right justified; word 2 has the associated pointer. The routing directories use the following type of table:

BRLFTBYTE	BRID - index	- left byte is optional
BRPTR - pointer		- searching routine returns this pointer to the table user

Type 4 Table List Search Control Block, LSRCHCB

LSCOUNT	- Entry count for this buffer
LSBUFPTR	- Pointer to current buffer

POI INTERFACE VALUES

B1TCB:	B0BUFPTR	POINTER TO A TCB
B1BUFF:	B0BUFPTR	DATA BUFFER POINTER

INPUT STATES POINTER TABLE SIZE: 0.. 80.

One such table exists for each TIP. ACTION TABLES and TIP TYPE/SUBTIP TYPE are discussed under service messages.

TIP TYPE TABLE, TIPTYPE

This table contains one entry of type TIPTYPE for each interface package in a system (TIP, LIP, or HIP). The local console, MLIA, line initializer, and on-line diagnostics are also included. The table fields are unique to each TIP.

Pointer to the table is BJTIPTYPT.

	15	14	13	12	8	6	4	0
0	F1	F2	F3	BJIVTSIZE	BJTCBSIZ	BJQTYPE	BJLISTIX Worklist Monitor Table Index	
1	BJDFTC - Default terminal class when enabling line (see appendix C) bits 0 - 4 only							
2	BJPTIMRTN - TIP TIMAL routine page address							
3	BJETIMRTN - TIP TIMAL routine entry address							
4	BJJFDT - TCB field descriptor table address							
5	BJFDT - LCB field descriptor table address							
6	BJJAT - TCB action table address							
7	BJAT - LCB action table address							
8	BJTPMUX2 - TIP level 2 (multiplex interrupt entry) page address							
9	BJTEMUX2 - TIP level 2 entry address							
10	BJTCBPINIT - TCB initialization routine page address							
11	BJTCBEINIT - TCB initialization routine entry address							
12	BJTXTPAGE - Text processing page address							
13	BJTXTENT - Text processing entry address							

Flags

F1 - BJOBT, generates output buffer terminated (OBT) flag

F2 - BJBZL, resets timer flag when OBT occurs

F3 - BJSP1, not used

BJTCBSIZE - number of words in IVT overlay for TCB/TCT

BJQTYPE - TCB buffer size (0 = 8, 1 = 16, 2 = 32, 3 = 64 in nominal system)

BASE SYSTEM HARDWARE

The base system data structures support the following functions:

- Buffer assignment, release, and copying
- Worklist assignment and control
- Monitor table use
- Finding system interface locations
- Low-core pointers
- Timing
- Masking
- Input regulation
- Control block support (setting up control blocks is a service module/TIP responsibility)
- Multiplex subsystem operators

BUFFERS

The proposed buffer structures are as follows:

- A control block for each pool of free buffers
- Definitions of each type of buffer assigned
- The optional stamping area which contains two words for tracing buffer use
- A copy buffer input parameter list used by the copy buffers routine, PBCOPYBFRS

There are four buffer sizes. In the normal systems, the buffers are assigned as shown:

BOS0 - 8 words
BOS1 - 16 words
BOS2 - 32 words
BOS3 - 64 words

Buffer Maintenance Control Block, BECTRL

This control block contains all the necessary information for allocating and releasing system buffers. There is a control block for each of the four free buffer pools. Each control block is initialized by PIBUF1. Firmware subroutines allocate and release the buffers.

	15	14	7	0
0	F1	BEBAC - Number of buffer currently available for assignment		
1	BENFB - next free buffer location			
2	BELFB - last free buffer location			
3	BEMSK - mask and length - 1			
4	BELCD - LCD of newly assigned buffer		BEFCD - FCD of newly assigned buffer	
5	BETRS1 - Pool's buffer threshold			
6	BECHAIN - Pointer to buffer control block for next largest size buffer			
7	BEDUM2 - not used			

F1 - not used
BECTPTR is pointer to BECTRL

System Buffer, BOBUFFER

System buffers exist in four sizes as defined by BOBUFSIZES. Buffers are used for a variety of purposes as described by the following overlay definitions:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	BFLCD - Last characters displacement								BFFCD - First character displacement							
1	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	BFQCNT		F11	F12	F13	F14
2	BFDATAC CHAR 1								BFDATAC CHAR 2							
	BFDATAC								122 data characters							
62	BFDATAC CHAR 121								BFDATAC CHAR 122							

BFQCNT - queue count

Last word usually reserved for chain to next buffer (see chain variant, below)

Flags:

F1 - BFEOTFLG, end of transmission buffer

F2 - BFSOTT, start of transparent text

F3 - BFSONT, start of nontransparent text

F4 - BFSUPCHAIN, suppress buffer chaining

F5 - BFEOBFLG, end of block buffer

F6 - BFINTBLK, internal block; do not send BACK block

F7 - BFPRTK, buffer protect
F8 - BFPERM, permanent buffer
F9 - BFLNKQ, buffer is part of link queue or frame
F10 - BFSP5, not used
F11 - BFSP7, used by console I/O
F12 - BFSP8, used by console output } reserved for TIP use[†]
F13 - BFSP9, not used
F14 - BFDBSIZE, data buffer size, not used, 0 indicates single data buffer size

---

[†] Use where buffer is assigned for console

OVERLAYS FOR TIP FLAGS

These overlays are for words 0 and 1, to use F10, F11, F12, and F13.

Mode 4 TIP

	15	6		3	2	1 0
0	BFFIL1 - fill for word 0					
1	BFFIL2 - fill for unused bits	F10		F11	F12	F13

F10 - BFFRAG - fragmented line
F11 - BFFE - format effectors present
F12 - PFPARTIAL - MSG block; not complete message
F13 - BFM4D3 - not used

ASYNC TIP

Four types of flag overlay are provided

	15	6		3	2	1 0
0	BFFIL1 - fill for word 0					
1	BFFIL2 - fill for unused bits	F10		F11	F12	F13

Variant 1 - ASYNC

F10 - BFPGWAIT, page wait for this output block
F11 - BFEOM, end of message block - output
F12 - BFEOS, end of source block - output
F13 - BFADM1, not used

Variant 2 - ASYNC

F10 - BFXPT, transparent input
F11 - BFPARITY, parity error in this block
F12/F13 - BFADM2, not used

Variant 3 - ASYNC

F10/F11/F12 - BFFLGS, clumped TIP flags
F13 - BFADM3 - not used

Variant 4

F10 - BFINTIP, internal from TIP
F11/F12/F13 - BFADM4, not used

TIP

	15	6		3	2	1	0
0	BFFIL1 - fill for word 0						
1	BFFIL2 - fill for unused bits	F10		F11	F12	F13	

F10 - BFBCCOK, BCC in and correct (3270)
F11 - BFNOTABRTPKT, input state program terminated
F12 - BFVRCBAD, VRC error in packet
F13 - BF32D3, not used

NPU console (TIP)

	15	6		3	2	1	0
0	BFFIL1 - fill for word 0						
1	BFFIL2 - fill for unused bits	F10		F11	F12	F13	

F10 - BFFORMAT, console format
F11 - BFTFXT, text for console in block
F12/F13 - BFCNSLFIL, not used

General Purpose Integer Buffer (64 words)

	15	0
0	BIINT	
	BIINT (ARRAY)	64 words of integers
63	BIINT ELEMENT 64	

General Purpose Chaining Buffer (64 words)

	15	0
0	BCCHAINS	
	BCCHAINS (ARRAY)	64 words of pointers for chaining (or other) purposes
53	BCCHAINS ELEMENT 64	

TCB buffer (32 word buffer)

	15	0
0		
	BSTCB	See TCB field definitions (above); 32 word maximum
31		

Physical/Logical Request Packet (PRP/LRP) buffer (16 word buffer)

	15	0
0		
	BCPRP	See PRD/LRP field definitions (above); only first 9 words of buffer are assigned
8		

Active TTY LCB List buffer (16 words)

	15		0
0		NELED - Index to last entry	
1	one entry	NELINO - Line number	up to 7 entries per buffer
2	requires 2 words	NELCBP - Pointer to MLCB	
		NEENTRY (ARRAY)	
13 14		NEETRY ELEMENT 7	
15		NECHAIN - Pointers to next active TTY	

Entries for a Type 1 Table

	15	0
0		
	BRTYP1	Two words per entry - see directories section 6
1		

Buffer for type 4 table (16 word buffer)

	15		0
0	CECOUNT - Index to last entry		
1			
		CEENTRY	2 word directory entry - see directories section 6
2	up to 7 directory entries per buffer		
		CEENTRY (ARRAY)	
13 14	CEENTRY ELEMENT 7		

Logical Link Control Block (LLCB) buffer (8 words)

	15	0
0		
	BLLLCB	See LLCB field definition
6		

Timeout buffers (5 words)

Two variants are provided.

	15	14 10	7 0
0	F25	BFTUSR - user bits	BFTWKCOD - work code
1	BFTLINO - Line number		
2	BFTWLINDX Worklist index	BFTSP1 - Not used	
3	BFTOVAL - Timeout count - base = 100 ms		
4	BFTCHAIN - Pointer to next timeout buffer or chain		

Flag:

F25 - BFTREL, release buffer after using

Variant for word 1 of timeout buffer

	15	11	7 0
0	BFTOM1 - Not used	BFTSCI - Status indication	BFTDM2 - Not used

These buffers are assigned as needed for delayed calls and chained together with the last buffer word BFTCHAN. PBTOSRCH searches the chained buffers for timeouts and releases them if BFIREL is set. The adjacent buffers are rechained together if the buffer is released.

Multiplex LCB (MLCB) buffer (32 words). Also used for TPCB.

	15	0
0		
	BGMLCB	See MCLB for field definitions. See also multiplex subsystem, section 5
31		

Mobil systems application flags (8 words)

	15	7	6	5	4	3 0
0	BFMDM1 - Integer					
1	BFMDM2 - Not used	F26	F27	F28	F29	BFMDM3 - Not used

Flags:

F26 - BFEOR, End of record
F27 - BFEOI, End of information
F28 - BFPMMSG, PM message

NPU Statistics buffer (16 words)

See appendix B of CCP Reference Manual for field definitions

	15	0
0	CPFIL0	
	CPFIL0 (ARRAY)	Six words of file for NPU statistics message block header
5	CPCIL0 ELEMENT 6	
6		
15	CPNPU	10 words of NPU statistics (9 words used)
16		

Line Statistics buffer (16 words)

See appendix B of the CCP Reference Manual for field definitions

	15	0
0	CPFIL1	
	CPFIL1 (ARRAY)	8 words of fill message block header and SVM bytes
7	CPFIL1 ELEMENT 8	
8		
	CPLINE	4 words of line statistics
11		

Terminal Statistics buffer (16 words)

See appendix B of the CCP Reference Manual for field definitions

	15	0
0	CPFIL2	
	CPFIL2 (ARRAY)	19 words of fill for message block header and SVM bytes
8	CPFIL2 ELEMENT 9	
9		
	CPTML	3 words of terminal statistics
11		

HASP TIP buffer (8 words)

	15	11	3	2	1	0
0	BFHS1 - Fill for word 0					
1	BFHSTYP - Canned message type	BFHS2 - Not used	F31	F32	F33	F34

1 = request permission to send, 0 = permission to send granted

Flags:

F31 - BFHSTXT, Text processed data
F32 - BFHSCMODE, Transparent data
F33 - BFHSNEW, New record flag
F34 - BFHS3, Not used

LIP

	15	7	6	5	4	3	2	1	0
0	BF1DSUM - file for word 0								
1	BF2DUM - not used	F35	BF3DUM not used		F36	F37	F38	F39	F40
2	BFAFLD - frame A field	BFCFLD - frame C field							
3	BFLFLD - subblock L field	F41	F42	BF4DUM - not used					

F35 - BFLBE, low on buffers encountered
F36 - BFIE, input error
F37 - BFRPD, receive priority sent to neighbor NPU
F38 - BFRPFLG, receive priority flag (this NPU)
F39 - BFREBFG, received end of block flag
F40 - BFTPRFG, transmit priority flag
F41 - BFTEBFG, transmitted end of block flag
F42 - BF4DUM, not used

Buffer Constants

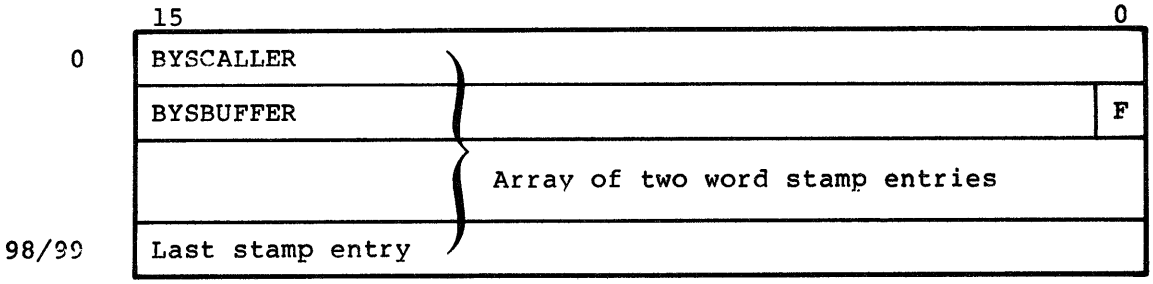
Mnemonic	Value	Meaning
J1FRSTCHAR	4	FCD FOR BUFFER ALLOCATE WHEN NOT IN A NETWORK
J1DATAFRST	4	FIRST CHAR POSITION OF ARRAY BFDATAC IN A BUFFER
J1LST8	13	LAST CHAR OF 8 WORD BUFFER
J1LST16	29	LAST CHAR OF 16 WORD BUFFER
J1LST32	61	LAST CHAR OF 32 WORD BUFFER
J1LST64	125	LAST CHAR OF 64 WORD BUFFER
J2LST128	253	LAST CHAR OF 128 WORD BUFFER
J1LSTCHAR	J1LST64	Maximum LCD in a data buffer (assumes system has selected 8, 16, 32 and 64 word buffers).
J1LCDFCD	0404	Hexadecimal displacements to character positions for LCD, FCD
J2LCDFCD	090A	
J3LCDFCD	1F06	
J4LCDFCD	1706	
J5LCDFCD	1906	
J6LCDFCD	1B06	

Mnemonic	Value	Meaning	
J1BLMAX	64	Maximum buffer length in system with 8, 16, 32, and 64 word buffers	standard system
DBUFLENGTH	64	Data buffer length (largest buffer)	
BYSTSZE	100	Length of circular stamp buffer, one word per buffer	
B1CIBSIZ	512	Size of circular input buffer (CIB)	
QCHN	3	Word 3 of buffer assigned as a block is the chain word	
JQT2SZE	16	Length of type 2 table	buffer assigned as table
JQT4SZE	16	Length of a type 4 table	
D0DNMAX	10	Length of a local directory (DN) table	

Buffer Stamping Area, BYSTAMP

The buffer stamping area provides a circular table of 50 entries to record the usage of the most recently assigned or released buffers in the NPU. As a buffer is assigned or released, the address of the program requesting this action is recorded together with the buffer address. The LSB of the buffer address entry indicates whether the buffer is currently free or assigned. The file 1 microregisters contain information about the buffer stamping:

0095 - stamping status: 0 = not used; ≠ 0 indicates stamping
0069 - base address of stamping area
006A - pointer to next entry to be used in the stamping area
006B - address of last entry in stamping area



F - flag giving the status of the buffer: 0 = put, 1 = get

Copy Buffer Parameters, JTCOPYB

This is the parameter list used when calling PBCOPYBFRS, the buffer copying routine.

	15	14	13	0
0	JTNUM - number of buffers to copy			
1	JTSSIZE - source buffer size			
2	F1	F2	JTRLS - release source buffers flag	

F1 - JTDSIZE, destination buffer size flag
F2 - JTSMIXED, mixed data buffer source chain - not used

JTNUM - only bits 7 through 0 are valid
JTRLS - only bit 0 is valid

Buffer Threshold Levels, BOBUFLEVELS

The following are the buffer threshold levels checked by the various regulation routines when determining whether to assign buffers from the appropriate free buffer pool or to reject input or to move to a lower level of input regulation. In the heirarchy of regulation checks, 9 is the most important, 0 is the least important.

Mnemonic	Value	Meaning	Type
BOT1	0	CONSOLE SNAPSHOT	
BOT2	1	CONSOLE SNAPSHOT	
BOTHDLY	2	COPY TO CONSOLE	
BOTHCT	3	TCB ALLOCATION	
BOTH3LV	4	LOWEST PRIORITY DATA	
BOTH2LV	5	HIGHEST PRIORITY DATA	
BOTH1LV	6	SERVICE MESSAGE DOWNLINE	
BOTHDIS	7	SERVICE MESSAGES UPLINE	
BOTHTIM	8	CLA STATUS HANDLER	
BOTHMUX	9	MUX BUFFER THRESHOLD	

WORKLISTS

Worklists are used on the OPS-level (a variant type of worklist - event worklists - is used in the multiplex subsystem). A worklist is a processing request (task). It is attached to a program. If more than one task is waiting to be executed by an OPS-level program, the worklists for the tasks are queued to the program on a first-in-first-out basis.

Worklists use work codes to describe the task to be done. The called program often uses the work code as a switching index to subprogram entry points.

Each worklist has a control block to point to the locations of the queued worklists.

An intermediate area (BWWORKLIST) is provided which PBLSPUT uses for constructing worklists and PBLSGET uses for handling worklists when a program is called for execution with the next worklist. Several routines define local worklist areas using the BWWORKLIST format.

Intermediate Array Format; BWWORKLIST

BWWORKLIST depicts the different format overlays which the intermediate array can assume. It also depicts the formats of the entries of the different worklists of the system. All fields are word length. The array of entries allows a maximum sized entry for each priority level in the system. The array is located at BWWLENTRY.

	Size (words)		
BWPKTPTY : B0BUFPTR	1	This overlay is for the console drivers worklists (B0TTYP, B0TTYN), and all worklists whose entries are a single pointer word of type B0BUFPTR.	
CATMLEY : INTEGER	1	This overlay is for the timing services worklist (B0BTIWL) and all worklists with single word integer entries.	
B0EWLQ : MMEVENT	5	This overlay is for the multiplex event worklist queue (MMEWLQ) and all worklists whose entries are 5 words long of type MMEVENT. Format is defined below.	
BWTCB, BWBLKPTR : B0BUFPTR	2	This overlay is for the internal processor worklist and all worklists with 2 consecutive pointer words.	
BWIMED : ARRAY (1..J1WLMAX)	1 to 6	This overlay is the general format used by list services for the bulk transfer of entries to and from any worklist.	
CMSMLEY : CMSMWLE	1 to 3	This is the service module worklist overlay	
ACPEVENT : B07BITS		Event code	
ACPBLINO : B0LINO	2	Line number	Coupler overlay
ACPB0BUF : B0BUFPTR		Buffer pointer	

	Size (words)	
BWORD1, BWORD2, BWORD3, BWORD4, BWORD5, BWORD6 : INTEGER		This overlay is for TIP debug and it provides easy access to each word of the intermediate array
BWLIPPARAMS : IELIPPARAMS	3	Overlays for the LIP. Two types of 3 word entries

The largest number of words allowed in any worklist (J1WLMAX) is six.

Multiplex Event Worklist Queue Types, MMEVENT

The event worklist for the multiplex subsystem is five words long. Several types are provided. The worklists can be prepared by users or by multiplex firmware.

VARIANT: Input processing - data

	15	7	5 0
0	MMWTCOUNT wait count in half seconds	MMSP1 (Not used)	MMWKCOD multiplex work code
1	MMLINO - Line number		
2	MMIBP - Input buffer pointer		
3 3	MMDM2 unused		
4	MMDM3 unused		

VARIANT: Output processing - data

	15	7	6 0
0	MMDELAYCNT - delay count	F1	MMSP4 - not used
1 2	MMPORT	MMLOPOR	
2	MMOBP - Output buffer pointer		
3	MMDM5 - not used		
4	MMDM6 - not used		

F1 is a delay completed flag, MMDECMPLT

Note that the TIP use a 3-word variant composed of words 0 and 1 of the first variant and word 2 of the second variant. Downline, this WLE is prepared by PBTWLE.

VARIANT: Universal overlay - user defined word format

	15	0
0	MMWD0	
1	MMWD1	
2	MMWD2	
3	MMWD3	
4	MMWD4	

VARIANT: Error condition

	15	11	7	0
0	MMINOP non-operational code	MMSCI indicator states condition	MMDM8 - spare	
1	MMDM9 - spare			
2	MMCSTS - CLA Status Word - See appendix B of The CCP Reference Manual			

VARIANT: CE error message

	15	14	13	12	11	10	9	8	7	6	5	4	3	0
0	MMDM10 - not used													
1	MMDM11 - not used													
2	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	MMDM12	

F2 - MMLCTS
F3 - MMLDSR
F4 - MMLDCD
F5 - MMLRI
F6 - MMLQM
F7 - MMLSQD
F8 - MMLILE

CLA STATUS BYTE 1

F9 - MMLILE
F10 - MMLPES
F11 - MMLDTO
F12 - MMLFES
F13 - MMLNCNA

CLA STATUS BYTE 2

See appendix B in
The CCP3 Reference Manual

VARIANT: MLIA status

	bits 3 - 0
0	MMDM13 - not used
1	NNDM14 - not used
2	MMLIAST - MLIA status

VARIANT: IELIPPARAMS

Two 3-word variants for LIP

	15	7	6	5 0
0	IECFIELD - C field of input frame	F1	F2	IEWKCODE - work code
1	IELINO - line number			
2	IEFRMPTR - pointer to frame			

F1 - IESPARE, not used
F2 - IELBE, low on buffers condition

15	11	7 0
IEDUM 1 - not used	IESCI - CLA status	IEDUM2 - not used
IELNKPTR - pointer to LLCB		
IEBLKPTR - pointer to block		

Service Module Type Worklist Entry Formats, CMSMWLE

Two principal types of worklists are provided: a class of entries with a work code and one type of entry for timing calls.

Work code class:

- Related to TCB

15	7 0
CMDATA (optional data)	CMWKCODE
CMLINO	Line number
CMPTR	Points to SM or TCB

Code range: 21-3F$_{16}$. See OPS-level workcodes for SVM.

- SM pointer

15 7 0

CMDATA	CMWKCODE	
CMPOINT		Pointer to SM

- Save and Return

15 7 0

CMDATA	CMWKCODE	
CMR1		Save location for R1 and R2 Return address
CMR2		
CMRTN		

- Service message timer

15 7 0

CMTIMER - Timeout in half seconds	CMTIPWC - TIP generated work code for SVM

Worklist Control Block, BYLISTCB

This control block holds information for each worklist. See worklist services portion of section 4.

Variant for multiplex-level worklists

15 14 7 0

0	F1	BYCNT - number of entries in worklist
1	BYPUT - put pointer for next entry	
2	BYGET - get pointer for next entry	
3	BYFEINC - index to first entry in WL buffer	BYINC - size of entry (words)

Normal variant for OPS-level worklists

15 14 10 8 0

0	F1	BYCNT
1	BYPUTMASK - put mask	
2	BYGETMASK - get mask	
⋮	⋮	⋮

<table>
<tr><th></th><th>15</th><th>14</th><th>10</th><th>8</th><th>0</th></tr>
<tr><td>3</td><td colspan="5">BYSPARE - not used</td></tr>
<tr><td>4</td><td colspan="3">BYWLINDEX - worklist index</td><td colspan="2">BYSP2 - not used - can use only bits 7-0</td></tr>
<tr><td>5</td><td colspan="5">BYSP3 - not used</td></tr>
<tr><td>6</td><td>F2</td><td colspan="2">BYMAXCNT - number of work-list to get on this call</td><td colspan="2">BYPAGE - program page address</td></tr>
<tr><td>7</td><td colspan="5">BYPRADDR - program address</td></tr>
</table>

F1 - not used

F2 - BYWLREQ, worklist required flag. Used by PBPAGE to set up intermediate WL array entry if the call was made without a WL.

BYWLTY is the array (BOWKLSTS) of BYLISTCB

Worklist Table, BOWKLSTS

The following ranked worklists determine the indexing of the OPS-monitor table. Values 1 through 7 are not serviced by the OPS-monitor. They are in the index to generate the worklist array. New entries should be added in front of these entries.

The remaining worklists (8 through end) are serviced by the OPS-monitor program. They are also part of the worklist array. New entries must be added at the end, but in front of BODUMMY. The last entry must be BODUMMY which is equal to the last TIP worklist value and causes the monitor scan pointer to return to value 8.

Mnemonic	Value	Meaning
BOFSWL	1	FIRST WORKLIST = MMEWLQ
MMEWLQ	1	MUX EVENT WORKLIST QUEUE
BOHIPDLQ	2	HIP DATA LIST QUEUE
BOSMTO	3	SERVICE MODULE TIMEOUT LIST
BOT200	4	CRITICAL 200 MS TIMEOUT
BOTTYP	5	TTY CONSOLE DRIVER - PRIORITY
BOTTYN	6	TTY CONSOLE DRIVER - NON PRIORITY
BOLPWL	7	LINE PRINTER DRIVER
BOCHWL	8	CONSOLE PROGRAM
BOINWL	9	INTERNAL PROCESSOR (IP)
BOMLWL	10	MLIA INTERRUPT HANDLER
BOSMWL	11	SERVICE MODULE (SVM)
BOTIWL	12	TIMING SERVICES
BOTYWD	13	TIP DEBUG (PTTIPDBG) - OPTIONAL
BOLIWL	14	LINE INITIALIZER (LINIT)
BODGWL	15	ONLINE DIAGNOSTICS - OPTIONAL
BODOWL	16	HOST INTERFACE PACKAGE (HIP)
BOHDLC	17	LINK INTERFACE PROGRAM
BOM4WL	18	MODE 4 TIP - BVT
BOTTYWL	19	ASYNC TIP

Mnemonic	Value	Meaning
B0HASP	20	HASP TIP
B027WL	21	2780/3780 TIP - not used
B0HHWL	22	HASP/360 HIP - not used
B0DUMMY	23	DUMMY FOR CONSOLE

The subtables scanned by the OPS monitor is called B0PGMS. It extends from B0CHWL to B0DUMMY. The value assigned in the array is B0WLCODES.

OPS-Level Workcodes, CMWKCODE

The work codes are used in the worklist entry to indicate the type of task a called module is to perform. These are also called TIP workcodes.

Mnemonic	Value (hex)	Meaning		
System work codes for HIP, LIP or TIPs				
A0HARDERR	0F	Hard error	From multiplex subsystem	
A0TIMEOUT	10	Line timer expired	From LCB scan timing	
A0QUEOUT	11	Output in queue	From internal processor	
A0SMEN	12	Enable line	From line initializer	
A0SMDA	13	Disable line	From SVM	
A0SMTCB	14	TCB built		
A0SMDLTCB	15	Delete TCB		
A0SMRCTCB	16	Reconfigure TCB		
Miscellaneous				
A0BREAK	19	Downline break	From SVM	
A0DBUX	1A	Output buffer XMIT	From TIP to itself	
A0SMLN	1B	Line status protect	From SVM	
A0SMNPUINIT	1C	NPU init protect	From SVM	
A0SMMPCCINIT	1D	MPCC Init protect	From SVM	Not used
A0SMFAIL	1E	Force load MPCC	From SVM	
SVM Workcodes				
C0LINOP	20	LINE OPERATIONAL	From TIP, LIP, or line initializer (LINIT)	
C0LNINOP	21	LINE INOPERATIVE		
C0LNDA	22	LINE DISABLED		

Mnemonic	Value (hex)		Meaning
C0DLTCB	23	TCB DELETED	From TIP or LIP
C0SMIN	24	SM IN	From SVM to itself
C0SMOUT	25	SM OUT	
C0SMDISP	26	DISPATCH SM	
C0OVLDATA	27	OVERLAY DATA	
C0BFR	28	MISCL. BFR EVENT	
C0ENABLE	29	ENABLE LINE EVENT	
C0DISABLE	2A	DISABLE LINE EVENT	
C0TMLDLT	2B	DELETE TERM. EVENT	
C0RCTCB	2C	Terminal reconfigured	From TIP or LIP
C0TMLRCNF	2D	Reconfigure terminal event	From SVM to itself

Generated by input state programs to OPS-level TIP or LIP (note that multiplex macros must equate this A0WK1 to its own A0WK1 with the same value)

Mnemonic	Value (hex)	Meaning
A0WK1	21	TIP/LIP WORK CODE 1
A0WK2	22	TIP/LIP WORK CODE 2
A0WK3	23	TIP/LIP WORK CODE 3
A0WK4	24	TIP/LIP WORK CODE 4
A0WK5	25	TIP/LIP WORK CODE 5
A0WK6	26	TIP/LIP WORK CODE 6
A0WK7	27	TIP/LIP WORK CODE 7
A0WK8	28	TIP/LIP WORK CODE 8
A0WK9	29	TIP/LIP WORK CODE 9
A0WK10	2A	TIP/LIP WORK CODE 10
A0WK11	2B	TIP/LIP WORK CODE 11
A0WK12	2C	TIP/LIP WORK CODE 12
A0WK13	2D	TIP/LIP WORK CODE 13
A0WK14	2E	TIP/LIP WORK CODE 14
A0WK15	2F	TIP/LIP WORK CODE 15
A0WK16	30	TIP/LIP WORK CODE 16
A0WK17	31	TIP/LIP WORK CODE 17
A0WK18	32	TIP/LIP WORK CODE 18
A0WK19	33	TIP/LIP WORK CODE 19
A0WK20	34	TIP/LIP WORK CODE 20
A0WK21	35	TIP/LIP WORK CODE 21
A0WK22	36	TIP/LIP WORK CODE 22
A0WK23	37	TIP/LIP WORK CODE 23
A0WK24	38	TIP/LIP WORK CODE 24

Mnemonic	Value (hex)	Meaning
A0WK25	39	TIP/LIP WORK CODE 25
A0WK26	3A	TIP/LIP WORK CODE 26
A0WK27	3B	TIP/LIP WORK CODE 27
A0WK28	3C	TIP/LIP WORK CODE 28
A0WK29	3D	TIP/LIP WORK CODE 29
A0WK30	3E	TIP/LIP WORK CODE 30
A0WK31	3F	TIP/LIP WORK CODE 31
A0STOP	A0WK1	Stop transmission code

Multiplex Event Work Codes

These work codes appear in the work code field of the event packet returned to the multiplex event worklist queue. The codes specify the nature of the information contained in the packet. Code values of 01 through $01E_{16}$ are reserved for multiplexer use.

Mnemonic	Value (hex)	Meaning
MMCLAS	1	CLA status received
MMOBUX	2	Output buffer transmitted
MMBUTCH	3	Buffer threshold changed
MMUNSOD	4	Unsolicited ODD
MMCAOR	5	CLA address out of range
MMIFFO	6	Illegal frame format (multiplex)
MMUNSIN	7	Unsolicited input
MMFES	8	Framing error status (multiplex subsystem frames)
MMCHOUT	9	Character timeout
MMTIMOD	A	ODD timeout
MMTIMRE	B	Modem response timeout
MMINEND	C	Input terminated
MMOTEND	D	Output terminated
MMBREAK	E	ASYNC terminal break detected
MMHARDERR	F	Hardware error

MONITOR TABLES

The main monitor tables is the OPS-level worklist array described above. That table's use is described in section 4. Other monitor tables are defined below.

PGMSKIP = (Run, Skip)

Run skip flag

BYPGMS

Three cases:

- BYIPGM - BOPGMS type
- BYWKLS - worklists type
- BYINT - integer type

SMONT

Used by timing services for timed programs (half-second time base)

15 0

BTTIMER - timer count
BTCURSP - BTCURPD - } non used pointers
BTMRIX - loop end check index

DOOVLSTATE

Overlay state table, scalar definition.

Four scalers:

- DOLDING - overlay loading
- DOLDED - overlay loaded
- DORNING - overlay running
- DOAVAIL - overlay space available

CBSYTMT

Used for OPS-level, time-dependent programs.

MISCELLANEOUS

System Interfaces

A system interface table (SIT) is defined in the form of a pointer array. Pointers define the locations of individual entries in this group of tables which are frequently used. In addition to the formally defined tables at the top of the SIT, the last group of entries are pointers to frequently used base programs.

System Interface Table, SITTBL

Mnemonic		Common Name
SIENTY	POINTER TO BWWLENTRY	OPS monitor
SITMTB	POINTER TO CBTIMTBL	Timing (PBTIMAL)
SIWLCB	POINTER TO BYWLCB	Worklist CB
SIDBSIZE	POINTER TO BEDBSIZE	Data buffer sizes
SITPSIZE	POINTER TO BETPSIZE	Not used
SINJTEC	POINTER TO NJTECT	Terminal characteristics
SITIMTBL	POINTER TO BLTIMTBL	Line timing
SITIPTYP	POINTER TO BJTIPTYPT	TIP type
SIOVLBLK	POINTER TO SYOVLCB	Overlay control
SILCBP	POINTER TO HALCBP	Sub TIP
SILLRMOV	ADDRESS OF PBLLRMOV	LLCB remove
SILLENTB	ADDRESS OF PBLLENTB	LLCB enter
SILCBS	ADDRESS OF PBLCBF	LCB
SICOIN	ADDRESS OF PBCOIN	Command down
SIGT1BF	ADDRESS OF PBGET1BF	Get buffer
SIRL1BF	ADDRESS OF PBREL1BF	Release buffer
SIBFAVL	ADDRESS OF PBBFAVAIL	Buffer availability
SIRT1LCB	ADDRESS OF PTRT1LCB	Return to TIP entry
SISV1LCB	ADDRESS OF PTSV1LCB	Save TIP entry
SILSPUT	ADDRESS OF PBLSPUT	Make a worklist
SIRELCHN	ADDRESS OF PBRELCHN	Release buffers
SIRELZRO	ADDRESS OF SIRELZRO	Release and zero buffers
SILOAD	ADDRESS OF PBLOAD	Load NPU
SI1BADD	ADDRESS OF PB18ADD	18-bit address final
SI18COMP	ADDRESS OF PB18COMP	18-bit address computer
SITOAH	ADDRESS OF PBTOAH	Convert of hex in ASCII format

Extent of the entries which point to other tables are:

Name	Description	Pointer	Number of Entries
SYLCBP	LCBs	BZLCBP	HLRANGE
SYLINO	Line number	B0LINO	HLRANGE
SYENTY	Interrupt worklist	BWWORKLIST	B0PRILEVEL
SYLTYT	Line type	NBLTYE	N0LTYP, 1...NKCONTROL
SYPRTT	Port	NAPORY	N0PORTS

Name	Description	Pointer	Number of Entries
SYCTCT	Multiplex character transmit characteristics	NICTCY	NOLNSPDS
SYTMTB	OPS - level periodic programs	CBSYTMT	COTDPGMS
SYTECT	Terminal characteristics	NJTECY	NOTCLASS
SYTIMTBL	Line timing	BZLTIME	O...C4LCBS
SYTIPTYPT	TIP type	TIPTYPE	NOTIPTY
SYOVLCB	Overlay control block	SYOVLCB	1 - see below

Overlay Control Block, SYOVLCB

This control block is used during overlay operations (remote NPU load/dump) some diagnostics and console overlays.

15	11	0
DCOVID - overlay ID		
DCOVST		
	DOOVLSTATE flags - see monitor table above	
DCLBN - last block number		
DCBN	array of block numbers loaded in ASCII/binary	

Firmware Entry Points

The following words (integer type) are the entry points for frequently used firmware routines.

Mnemonic	Address (hex)	Function Performed by Firmware
PFLSGET	607	Gets a worklist entry
PFLSPUT	608	Builds a worklist entry and queues if it's necessary
PFBURLS	606	Release a buffer
PFBUGET	605	Assign a buffer of the size requested
PFBUEXT	609	Extract a buffer

Mnemonic	Address (hex)	Function Performed by Firmware
N1FIRMAD	600	Output to CLA sequence
N2P3INTAD	601	Generate a multiplex - level 2 interrupt
N3P3INTAD	602	Reset multiplex - level 2 interrupt
PFLINTO	60A	Decrement line timeout count
PFSR2SM	60E	Set/reset status bits. Used for program execution timing (requires external hardware measuring device)

Low-Core Pointers

The low-core pointers (also called the address table) is a sequence of address extending from location 0150_{16} to location $016A_{16}$. Refer to appendix B of The CCP3 Reference Manual.

8K Page Locator Table

This table is used to jump to a routine which is not located on the same 8K page of memory. CCP uses memory up to the 96K word boundary JUMPTEL in an array of two-word entries.

	15	0
0	JPAGEVAL - Page index for a routine	
1	JENTADDR - Entry address with the page	

TIMING TABLES

The principal timing tables are:

- RTC (real-time clock) table used to count 3.3 ms increments to generate the 100 ms RTC interrupt
- One-second clock counter
- Line timing table for timing out I/O events
- Array of programs which are run periodically
- Time of day tables
- Timeout buffers - See Buffer subsection

RTC/Autodata Transfer Table, CICLKADT

CICOUNT is incremented by firmware every 3.3 ms. When CICOUNT = CILIMIT = 30 (100 ms), the timer is reset and PBTIMER generates the 100-ms interrupt.

15	0
CIWORD1 - Constant = $80F0_{16}$	
CICOUNT - Counter - incremented every 3.3 ms	
CILIMIT - Interrupt count = 30. Compared to CICOUNT	
CISPARE - Not used	

One-Second Clock, CASECNTR

This clock is used by PBTIMEOFDAY for time of day calculations. The count is used modulo 60 - by the minute counter modulo, 60 x 60 by the hour counter, modulo 60 x 60 x 24 by the day counter, and modulo 60 x 60 x 24 x month (days) by the month counter.

15	0
CASECNTR - One-second clock	

Line Timing Control Table, BLTIMTBL

This table is used for timing out the output buffer (OBT) for each line. Entries are accessed by line number. Entries use a half-second time base.

BLTIMTBL uses SYTIMTBL type table and BZLTIME entry (one word).

	15	14	7 0
0	F1	BLTRESET	BLTIME
last line			

F1 - BLTCONT, not used
BLTRESET - timeout value for the line, used to set BLTIME
BLTIME - Set by line user; decremented each half-second PBTIMAL

Periodically Executed Programs, CBTIMTBL

This array of timing entries (type CBSYTMT) is used to time out the period between program executions. The table is scanned every half second by PBTIMAL and each program's count is decremented.

If count = 0, the associated periodic program is called, and the timing counter returns to the full period value.

	15 0
0	CBTIMER - Timing remaining
1	CBINTVAL - Period - in half seconds - used to reset periodic program calls
	CBPADDR - Page address of program to be called
3	CBADDR - Address of program to be called
	CBTIMTBL (ARRAY) of CBSYTMT four word entries
18 49 50 51	CBTIMBL ELEMENT 13

The period is set for each program at build time. The programs in the normal system and this place in the table are shown below:

Mnemonic	Value	Meaning/Program	
COLCETMSON	0	ACTIME LCB LIST SCAN PBLCBTMSCAN	
COADJUST	1	BUFFER ADJUSTMENT PBADJUST	
COTUP	2	TEST UTILITY PROGRAM (TUP), PBTUP	
COTIMEOFDAY	3	TIME OF DAY AND DATE/PBTIME OF DAY	
COT1SEC	4	MUX TTY TIMER, PMT1SEC	
COPSTAT	5	PERIODIC STATISTICS DUMP, PNDSTAT	
COIOTMR	6	I/O TIMAL APPENDAGE, PBTUSRCH	
COCECNT	7	RESET CE ERROR COUNT, PB	
COVAFQTAT	8	Variable frequency turnaround	
COOVLY	9	Overlay	
COHLIP	10	LIP, PLIPTC	PNLLLI, PNLLLO, PNLLTC
COLLPR	11	Logic link protocol	
COSPARE	12	Spare - for debugging use	

Time of Day Tables, CADATE

The table is checked every second and incremented. An overflow in one word causes that word to be zeroed and the next word to be incremented.

	15 0
0	CASEC - seconds (0-59)
1	CAMIN - minutes (0-59)
2	CAHOUR - hour (0-23)
3	CADAY - days (0-31)
4	CAMONTH - months (1-12)

Overlay for conversion

	15	0
0	CATIMV	
	CATIMV (ARRAY)	
4	CATIMV ELEMENT 5	

Loop Lower Instruction

LOOPFOREVER has a value of $18FF_{16}$. Executing this instruction places the NPU in a closed, continuous loop.

REGULATION

Input Regulation Option for PTREGL, REGLTYPES

These options define the type for regulation check for the link.

RELOGLNK	1	LOGICAL LINK REGULATION
RELOCAL,	2	LOCAL BUFFER LEVELS
REABL,	3	ALLOWABLE BLOCK LIMIT
REACPINP	4	ACCEPT INPUT

The set of REGLTYPES = REGSET

CONTROL BLOCKS

The system structures provide these principal control blocks for network elements:

- LLCBs for logical links
- LLCBs for each line

} state assignment

- TCBs for each terminal - dynamic assignment at enable time

Static Logical Link Control Block (LLCB), B0SLLCB

A static LLCB is required for each logical link connected through this NPU (that is, this NPU has at least one of the nodes forming this logical link). The number of LLCBs is a build time parameter and LLCBs are initialized at load time. Two variants are provided for word 6. These are a maximum of 5 (J0MAXLLCB) LLCB in the system.

<table>
<tr><th></th><th>15</th><th>14</th><th>13 7</th><th>5</th><th>2 0</th></tr>
<tr><td>0</td><td>F1</td><td>F2</td><td>BLDMY - not used</td><td>BLTREG</td><td>BLREG</td></tr>
<tr><td>1</td><td colspan="5">BLCONDIR - Connection directory or coupler TCB</td></tr>
<tr><td>2</td><td colspan="3">BLDN - destination node</td><td colspan="2">BLSN - source node</td></tr>
<tr><td>3</td><td colspan="5">BLCHAIN - chain to next LLCB</td></tr>
<tr><td>4</td><td colspan="3">BLHO - host ordinal</td><td colspan="2">BLSTATE - configuration state</td></tr>
<tr><td>5</td><td colspan="5">BLTE - LL state expiration time</td></tr>
<tr><td>6A</td><td colspan="4"></td><td>F3</td></tr>
<tr><td>6B</td><td colspan="5">BLSTE - LL state</td></tr>
</table>

F1 - BLCDS, connection directory flag
F2 - BLINIT, initial LL status SM sent to host
BLTREG - last transmitted regulation level at this end
BLREG - regulation level at this end

Logical link states for BLTREG and BLREG

	Value	
LL0	0	Not configured
LL1	2	Waiting for clear
LL11	4	Waiting for linkage
LL12	6	Waiting for PRST (protocol reset)
LL2	1	Operational
LL3	8	Down

When used as a directory, the chain of blocks can be searched using either BLDN or BLSN as an index. BLCONDIR points to the connection directory for this link (looking toward multiplex lines) or to the coupler TCB (looking toward host).

Line Control Block (LCB), BZLCB

One line control block is provided for each line (port) connected to the NPU. The LCB contains the line dependent information used primarily by OPS level interface packages to:

- Define and control line protocol
- Define and interface with external line managers (such as the service module)

Words 0 through 14 are common to all LCBs. A series of overlays is provided for various TIP and subport types, starting at word 15. The line control block array is composed of successive 24 word LCBs. A maximum of 33 array elements are permitted for a total of 792 words.

C0LCBD = ARRAY (0..C4LCB5) of BZLCB

	15	14	13	12	11	7	0
0	BZLINO - line number						
1	BZTMRCHN - active LCB timer chain						
2	BZWTCOUNT - wait count - half-second base					BZOWNER - Node ID of CS which owns line	
3	Save locations for				BZRET1ADDR - input routine return address		
4	suspended TIP processing				BZRET2ADDR - Output routine return address		
5	F1	F2	F3	F4	Line type BZLTYP see appendix C	BZHO - host ordinal	
6	BZCNFST - current configuration state				BZLNSPD - line speed; see appendix C	BZTCBONT - number of TCBs currently attached to this line	
7	F5	F6	F7	F8	BZSTATE - line state (note 1)	BZWKCODE - last work code received	F9
8	BZTIPTYPE - TIP type - see appendix C				BZSUBTIP - sub TIP type - see appendix C	BZSVTIPTYPE - save area for TIP type during initialization (uses only bits 3 - 0).	
9 12	BZSTIC - line statistics block. A four integer record called BZLNCNTS. Words (in order 9 - 12) are: BZBTRANS - number of blocks transmitted; BZBRCV - number of blocks received; BZCTRANS - number of characters transmitted; and BZCRCV - number of characters received						
13	BZTCBPTR - pointer to first TCB attached to this line						
14	BZLBTOMUX - pointer to last buffer given to multiplex subsystem						
15	BZALCT - Alarm/message counter for this line						

Flags:

F1 - BZTAPEX, TIMAL appendage exists for this line; that is, PBTIMAL scans this block for an I/O timeout (active LCB)

F2 - BZCHECKQS, checks when output queued

F3 - BZSMRESP, SM response received

F4 - BZSMTO, SM is being timed out

F5 - BZTOUTPUT, terminate output

F6 - BZTINPUT, terminate input

F7 - BZDIS, line disabled (used by SVM only)

F8 - BZDIAG, online diagnostic test in progress

F9 - BZAUTO, autorecognition required on this line

NOTE 1: These states are local constants in the line initializer program: PTLINIT. See that routine for values assigned to line states.

VARIATIONS: Words 16 and higher.

Subline control block

	15	7 0
16	BZSUB1PTR - pointer to first attached subport	
17	BZSP5 - not used	BZNUMSUBS - number of subports

Mode 4 TIP

	15	14	13	5 0
	BZCURTCB - TCB currently being serviced by TIP			
16	F10	F11	BZ4R1 - not used	BZMAXRETRY - maximum number of retries for this line

F10 - BZDELAYLINE
F11 - BZMULTIDROP, multidrop line

HASP TIP

Two variations are provided as follows:

	15	14	13	12	11	7	6	5	0
16	BZHSWFCS - workstation function control sequence (FCS)								
17	BZHSTFCS - TIP function control sequence								
18	BZHSHEAD - Pointer to head of data list queue								
19	BZHSTAIL - Pointer to tail of data list queue								
20	BZHSCONSOLE - Pointer to address of console TCB								
21	BZHSOTCB - Pointer to current TCB address								
22	BZHSCCB - Pointer to current continue buffer								
23	BZHSIBCB - Input BCB count				BZHSOBCB - Output BCB count	F13	F14	BZHSETO - Retry count - errors	F15
24	F16	F17	F18	F19	BZHSNAK Retry count - NAKS	BZHSRRBITS - Read request bits			F20

Flags:

F13 - BZHSGNON, Sign on card seen
F14 - BZHSENQSEEN, Enquiry block seen
F15 - BZHSWOQ, Waiting for output
F16 - BZHSICREG, Suspend card reader command
F17 - BZHSIPREG, Transparent mode
F18 - BZHSXPT, Transparent mode
F19 - BZHSRSBCB, Reset BCB needs to be sent flag
F20 - BZHSLINERR, Line error occurred

HASP TIP Records

	15	0
16	BZHSC	8 word array for clean up purposes
	BZHSC (ARRAY)	
23	BZHSC ELEMENT 8	
	BZHSRQP - Input stream requests - 16 bits, one per device (0 = must request permission, 1 = permission granted)	
	BZHSPNED - Output stream requests - 16 bits - one per device (0 = must request permission, 1 = permission granted)	

ASYNC TIP

	15	11	8	7	6	5	4	3	2 0
16	BZMSCNT - 100 ns counter	BZMSRST - 100 ns reset value	F21	F22	F23	F24	F25	F26	BZADM1 - not used
17	BZMSCHN - LCB chain for active LCBs with 100 ms timeout								

F21 - BZMSCART, character timeout
F22 - BZEPDLY, Echoplex - delay control
F23 - Input terminated flag
F24 - BZEPLREC, End of physical line received
F25 - BZXOFFREC, Transparent X-OFF received
F26 - BZARWK, Possible low-speed autorecognition

Terminal Control Block (TCB), BSTCBLK

The terminal control block defines terminal-dependent information. One TCB is provided for each terminal in the system. There can be several TCBs for a single line. The first 12 words are basic to all TCBs, static or dynamic (static TCBs are assigned for the MLIS and the coupler). Overlays are then provided. There is a common overlay for all dynamic TCBs, called the IVT overlay. All of the TIP TCBs use at least the first three words of this overlay; the remaining words for each TIP require a unique overlay for that TIP. The statis TCBs occupy words dynamic TCBs and are released by a line disabled condition.

<table>
<tr><th></th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>0</td><td colspan="16">BSCHAIN - pointer to next TCB for this line</td></tr>
<tr><td>1</td><td colspan="16">BSLCBP - pointer to LCB for this line</td></tr>
<tr><td>2</td><td colspan="8">BSCA cluster address</td><td colspan="8">BSTA - terminal address</td></tr>
<tr><td>3</td><td>F1</td><td colspan="3">Code set for BSCODE terminal (1)</td><td colspan="4">BSHO Host ordinal</td><td colspan="3">BSDEVTYPE device type. See appendix C</td><td colspan="5">BSTCLASS Terminal class. See appendix C</td></tr>
<tr><td>4</td><td colspan="16">BSQPTR - pointer to TCB queue (OQ indication)</td></tr>
<tr><td>5</td><td colspan="6">BSOWNER - Node ID of CS</td><td colspan="10">BSCN - Connection number (CN)</td></tr>
<tr><td>6</td><td colspan="16">BSLLCB - Pointer to LLCB for this line</td></tr>
<tr><td>7</td><td>F2</td><td colspan="3">BSABL - Available block limit</td><td>F3</td><td colspan="3">BSOBL - Outstanding block count (2)</td><td colspan="5">BSLBTPROC - type of last block processed (3)</td><td>F4</td><td>F5</td><td>BSIPRI Input priority(4)</td></tr>
<tr><td>8</td><td>F5A</td><td>F5B</td><td colspan="3">BSBSNLAST BSN of last back block (5)</td><td colspan="3">BSBSNCRNT BSN of CRNT output block</td><td>F6</td><td colspan="2">BSPARIT (6)</td><td colspan="2">BSCHLEN (7)</td><td>F7</td><td>F8</td><td>F9</td></tr>
<tr><td>9</td><td colspan="16" rowspan="3">BSSTIC } Terminal statistics block. This is a record of three integer words named BSTMLCNTS. The statistics words (in order 9 - 11) are:
BSBTRANS - number of blocks transmitted
BSBRCN - number of blocks received
BSBBAD - number of bad blocks transmitted and received</td></tr>
<tr><td></td></tr>
<tr><td>11</td></tr>
</table>

Notes:

1. SBCODE, see subTIP type table in appendix C

2. BSOBL, number of blocks queued to this TCB awaiting processing

3. BSLSTPROC, see block types in section 6

4. BSIPRI, see PTREGL priority level: 1 = 0, 2 = 1

5. BSBSNLAST, } see block protocol in section 6
   BSBSNCRNT

6. BSPARITY, type parity

   0 = zero
   1 = odd
   2 = even
   3 = none

7. BSCHLEN, character length

0 = 5 bits
1 = 6 bits
2 = 7 bits
3 = 8 bits

8. BSBCKLTR - upline back postponed flag (uses only bit 0)

Flags:

F1 - BSSTOP, Data stopped to this terminal
F2 - BSINOP, Terminal inoperative
F3 - BSTBRCONF, Terminal to be reconfigured
F4 - BSACPINP, Terminal accepts input for host (AI)
F5 - BSACPOUT, Terminal accepts output from host (AO)
F5A - BSRES1, Not used
F5B - BSWAIT, Waiting for initialization
F6 - BSTBTERM, TCB is to be deleted
F7 - BSPGWAIT, In page wait mode
F8 - ESXPARENT, Terminal data in transparent mode
F9 - BSHOTOGL, Host ordinal toggle bit

MLIA Handler (statis TCB)

	15 0
12	F18 BSWRKCO - Process state work code (bits 0 and 1 only)
13	BSCONB - Condition B counter (input drop errors)
14	BSCONC - Condition C counter (last data errors)
15	BSCOND - Condition D counter (Input error ODD first-in, first-out error)

Coupler TCB (static TCB)

This is the TCB used by the HIP for transfers to/from the host.

	15 14 7 0
12	BSCPAVPTR - Pointer to first available output buffer
13	BSCPLAST - Pointer to last available output buffer
14	BSCPINPUT - Input buffer address
15	BSBUFOTT - Memory access loaded address
16	BSCPSTATUS - Coupler Status
17	BSCPDATA - Order word storage
18	BSCPCMD - Last NPU status word sent to host
	⋮

⋮

19	BSCPBUFAV - Number of available buffers	BSCOBZSTA - Previous state
20	BSCPAMASK - Coupler interrupt mask	
21	F42	BSCPIDLT - Idle timeout counter
22	BSCPCONN - Coupler connection number	

Flag:

F42 - BSCPHST, host status

1 = host available
0 = host down

IVT Overlay - Used by HASP, ASYNC, and MODE 4 TIPs

	15	14	13	12	11	9	7	0
12	BSPGWIDTH - page width						BSPGLENGTH - page length	
13	BSCANCHAR - cancel input character						BSCNTRLCHAR - control character	
14	BSUSR1 - user break 1						BSUSR2 - user break 2	
15	F33	F34	F 34A	F 34B	BSXCNT - character count in transparent mode			
16	F35	F36	F37	F38	BSOUTDE (1)	BSAPL (2)	BSXCHAR - transparent mode delimiter character	
17	BSBSCHAR - backspace character						BSABTLINE - abort output line character	
18	BSCRIDLES - count of idles after CR						BSLFIDLES - count of idles after LF	

Flag:

F33 - BSXTO, transparent transmission delimited by time

F34 - BSXCHRON, transparent transmission delimited by character (BSXCHAR)

F34A - BSFIRST, first 2741 upline message

F34B - BSRES3, not used

F35 - BSCRCALC, calculate the CR idle count

F36 - BSLFCALC, calculate the LF idle count

F37 - BSECHOPLX, Echoplex mode

F38 - BSINDEV, Input device

0 = keyboard
1 = paper tape

(1) - BSOUTDEV, output device

0 = printer
1 = display
2 = PT
3 = illegal

(2) - BSAPL, APL node

0 = no
1 = yes
2 = special APL node

Mode 4 TIP - The first three words are reserved for the first three words of the IVT overlay.

<table>
<tr><th></th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>1 0</th></tr>
<tr><td>12</td><td colspan="12">BSM4IVT</td></tr>
<tr><td>13</td><td colspan="5">BSM4IVT</td><td colspan="7">3-word array for IVT overlay parameters</td></tr>
<tr><td>14</td><td colspan="12">BSM4IVT ELEMENT 3</td></tr>
<tr><td>15</td><td colspan="12">BSIBUFF - input buffer pointer</td></tr>
<tr><td>16</td><td colspan="12">BSOBUFF - output buffer pointer</td></tr>
<tr><td>17</td><td colspan="3">BSCSTATE</td><td colspan="2">BSTSTATE</td><td>F39</td><td>F40</td><td>F41</td><td>F42</td><td>F43</td><td>BSRESS - not used</td><td>F43A</td></tr>
<tr><td>18</td><td>F44</td><td>F45</td><td>F46</td><td>F47</td><td>BSRES6 not used</td><td colspan="4">BSERRTYPE - type of CE error</td><td colspan="3">not used</td></tr>
<tr><td>19</td><td colspan="7">BSCLC - current line count</td><td colspan="4">BSERRSP - error response error count</td><td>not used</td></tr>
<tr><td>20</td><td colspan="12">BSTIMER - Event timer (Event is polling, etc.)</td></tr>
<tr><td>21</td><td colspan="12">BSSTACK</td></tr>
<tr><td>22</td><td colspan="5">BSSTACK</td><td colspan="7">return address stack</td></tr>
<tr><td>23</td><td colspan="12">BSTACK ELEMENT 3</td></tr>
<tr><td>24</td><td colspan="7">BSPOLL CHAR 1</td><td colspan="5">BSPOLL CHAR 2</td></tr>
<tr><td>25
⋮</td><td colspan="5">BSPOLL</td><td colspan="7">(2 character/word) array of 14 characters for polling</td></tr>
<tr><td></td><td colspan="5">⋮</td><td colspan="7">⋮</td></tr>
</table>

	15	7 0
30	BSPOLL CHAR 13	BSPOLL CHAR 14
31	BSCLSPTR - pointer to cluster TCB	

BSCSTATE - Cluster state

0 = idle
1 = interactive
2 = batch/card reader
3 = batch/printer
4 = batch/card reader and printer
5 = cluster error
6-7 = not used

BSTSTATE - Terminal states

0 = idle
1 = active
2 = degraded
3 = terminal error
4 = autorecognition
5-7 = not used

Flags:

F39 - BSCRESERVED, reserved for cluster use

F40 - BSTRESERVED, reserved for terminal use

F41 - BSTOGEXPECTED, toggle expected (toggles for configure and reconfigure)

F42 - BSTOGRECEIVED, toggle received

F43 - BSTOBERCF, reconfiguration expected

F43A - BSOUTINPRGES, output in progress

F44 - BSAUTOINPUT, autoinput

F45 - BSINREQ, input required

F46 - BSCYCLE, cycle request outstanding - check TCB for next cluster device

F47 - BSJOB, message in progress

ASYNC TIP - first seven words are reserved for IVT overlay.

	15 14 13 12 11 10	9 8 7 6 5 4 3 2 1 0
12	BSASYIVT	
13	BSASYIVT	7 word array reserved for IVT parameters
17	BSASYIVT ELEMENT 7	

<table>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="8">BSLNCNT - text processor line count</td><td colspan="8">BSPGCNT - text processor page count</td></tr>
<tr><td colspan="16">BSAUTOINPUT - pointer to auto-input buffer</td></tr>
<tr><td colspan="16">BSFIBP - pointer to input buffer</td></tr>
<tr><td colspan="16">BSSOURCE - pointer to output source buffer</td></tr>
<tr><td colspan="16">BSCURSOURCE - pointer to current logical line source</td></tr>
<tr><td colspan="16">BSNXTSOURCE - pointer to next logical line source</td></tr>
<tr><td colspan="16">BSOBPTR - pointer to text processed output buffer</td></tr>
<tr><td>F48</td><td>F49</td><td>F50</td><td>F51</td><td>F52</td><td>F53</td><td>F54</td><td>F55</td><td>F56</td><td>F57</td><td>F58</td><td>F59</td><td>F60</td><td>F61</td><td colspan="2">BSPARTY</td></tr>
<tr><td colspan="6">BSTPSTATE (2)</td><td colspan="8">BSFESAVE - save format effector</td><td>F62</td><td>F63</td></tr>
<tr><td colspan="16">BSRETADR - save area for return address</td></tr>
<tr><td colspan="8">BSCURLCD - current logical line LCD</td><td colspan="8">LCDBSNXTLCD - next logical line LCD</td></tr>
<tr><td colspan="8">BSCURCHR - current character in right byte</td><td colspan="8">BSNXTCHR - next character in right byte</td></tr>
<tr><td colspan="16">BSCMDQUE - pointer to IVT command queue</td></tr>
</table>

Flags:

F48 - BSALWIN, allow input; no input stop

F49 - BSXOFF, X-OFF character detected

F50 - BSBREAK, break character detected

F51 - BSINACT, input active

F52 - BSOUTACT, output active

F53 - BSMXREG, send message indicating multiplex buffer regulation is in effect

F54 - BSWTFORMSG, waiting for downline message

F55 - BSPGTURN, wait for page turn input

F56 - BS1SLF, output paper tape LF detected

F57 - BSTRAILER, trailer detected

F58 - BSPWFE, save format effect or after page wait

F59 - BSCURRIGHT, current logical line byte (0 = left, 1 = right)

F60 - BSNXTRIGHT, next logical line byte (0 = left, 1 = right)

F61 - BSLASTWLE, last input line type (0 = logical line, 1 = physical line)

F62 - BSNXTEOBS, next text processing source - flag

F63 - BCCUREORS, current text processing source - flag

(1) - BSPARTYPE, terminal parity (0 = zero, 1 = odd, 2 = even, 3 = none)

(2) - BSTPSTATE, saved index to TP states pointer table

HASP TIP - first three words are reserved for IVT overlay.

	15	12	11	10	9	8	7	6	5	4	3	2	1	0
12	BSHSPIVT													
13	BSHPIVT			3-word array reserved for IVT parameters										
14	BSHSPIVT ELEMENT 3													
15	BSHSOBUFF - Pointer to current output buffer													
16	BSHSHEAD - Pointer to head of data list queue													
17	BSHSTAIL - Pointer to tail of data list queue													
18	BSHSQBF - Pointer to data list queue buffer													
19	BSHSSTMR - Suspend transmission timer													
20	BSHSFCSM - Stream mask for function control sequence (FCS)													
21	BSHSDBP - Pointer to text processing destination buffer													
22	BSHSFDBA - Pointer to text processing source buffer													
23	BSHSAUTOPTR - Pointer to auto-input buffer													
24	BSHSXLTA - Pointer to code translation table													
25	BSHSIMD	F64	F65	F66	F67	F68	F69	F70	F71	F72	F73	F74	F75	F76

BSHSIMD - Card read mode - 029 card default in each case, 1 = 026 card. (Bit 15 - transparent mode; bit 14 - nontransparent mode, bit 13 - EBCDIC mode)

Flags:

F64 - BSHSJOB, Job expected flag
F65 - BSHSRSNT, Request for permission to send signal sent
F66 - BSHSPNED, Request for permission to send signal needs to be sent
F67 - BSHSOIP, Output in progress
F68 - BSHSSUSP, Output stopped message sent to host
F69 - BSHSSF, Countdown for output stopped
F70 - BSHSITPCB, Input TPCB built
F71 - BSHSDCRB, Destination character in right byte
F72 - BSHSENDOFINF, EOI on input detected

F73 - BSHSIVTCERR, Upline IVT command error
F74 - BSHSSATAUTO, Auto-input satisfied
F75 - BSHSIOK, Received start of input
F76 - BSHSOKIVT, Send good IVT command response

LIP

	15	7	0
12	BSI		
		Array of 8 words for frame retention queue	
19	BSI - Element 8		
20	BSLNKQPTR		
21		Array of two pointers to link queues	
22	BSTEXTQPTR		
23		Array of two pointers to text queues	
24	BSRBRF	Array of two pointers to first receive	
25		block reassembly buffers	
26	BSRBRL	Array of two pointers to last receive	
27		block reassembly buffers	
28	BSRBRF	Array of two pointers to first transmit	
29		block reassembly buffer	
30	BSTBRL	Array of two pointers to last transmit	
31		block reassembly buffer	
32	BSLINO - Line number		
33	BSNID - ID of neighbor NPU node	BSITSS - Initial trunk states sent (Bit 0 only)	
34	BSUI - Pointer to UI block		
35	BSLDPTR - Pointer to load/dump block		
36	BSFRMPTR - Pointer to frame		
37	BSCMDCFLD - Command C-field	BSRSPCFLD - Response C-field	
⋮			⋮

<table>
<tr><td></td><td colspan="9">15 14 13 12 11 10 6 4 2 0</td></tr>
<tr><td>38</td><td colspan="4" rowspan="6">ALL COUNTERS ARE SET TO MAXIMUM VALUES AND ARE DECREMENTED. SEE LOCAL CONSTANT IN PLIP</td><td colspan="5">BST1ET - T1 expiration time (in seconds)</td></tr>
<tr><td>39</td><td colspan="5">BSHCTE - HDLC control time expiration time counter (in seconds)</td></tr>
<tr><td>40</td><td colspan="5">BSLCTRE - Transmit frame expiration time counter (in seconds)</td></tr>
<tr><td>41</td><td colspan="5">BSLCTSE - Trunk status SM expiration time (in seconds)</td></tr>
<tr><td>42</td><td colspan="5">BSLMTE - Line expiration time counter (in seconds)</td></tr>
<tr><td>43</td><td colspan="5">BSTOI - Trunk initialization expiration time counter (in seconds)</td></tr>
<tr><td>44</td><td colspan="3">BSHCSTE</td><td colspan="2">BSLCSTE</td><td>BSLMSTE</td><td>BSNSO</td><td>*</td><td>BSNSR</td></tr>
<tr><td>45</td><td colspan="3">BSNSU</td><td colspan="2">BSSECSTE</td><td colspan="2">not used</td><td colspan="2">BSPRISTE</td></tr>
<tr><td>46</td><td colspan="9">BSWD1XMT - FCD, LCD and flags for</td></tr>
<tr><td>47</td><td colspan="9">BSWD2XMT - Short TCC transmit frame</td></tr>
<tr><td>48</td><td colspan="9">BSXMTAFLD BSXMTCFLD
A and C fields for a short TCC transmit frame</td></tr>
<tr><td>49</td><td colspan="9">BSTCCOB - Point to frame being output currently</td></tr>
<tr><td>50</td><td>F76</td><td>F77</td><td>F78</td><td>F79</td><td>F80</td><td colspan="4">BSRC - Retry counter</td></tr>
</table>

*Not used

BSCMDFLD - See control byte - figure 8-2 in LIP

BSRSPCFLD - Description for values

BSHCSTE - CDCCP control state

0 = HC0
1 = HC11 - Awaiting response to SARM
2 = HC12 - Timing transmission
3 = HC2 - Sending
4 = HC3 - Loading
5 = HC4 - Waiting for load block

BSLCSTE - Link control state

0 = LC0
1 = LC1 - Awaiting send
2 = LC2 - Awaiting LINIT
3 = LC3 - Link control transmit (operating)

BSLMSTE - Line control state

0 = LM0
1 = LM1 - timing
2 = LM2 - awaiting CLA on
3 = LM3 - idle
4 = LM4 - transmitting
5 = LM5 - awaiting enabled state

BSNSO - Indicates next frame to transmit

BRNSR - Indicates next frame to receive

BSNSU - Index to next unacknowledgment frame

BSSECSTE - secondary states

0 = S0 - basic state
1 = S1 - receive
2 = S2 - reject
3 = S3 - busy

BSPRISTE - primary states

0 = P0 - basic state
1 = P1 - link set up
2 = P2 - transmitting
3 = P20 - idle trunk
4 = P21 - modules limit
5 = P3 - timeout-recovery
6 = P4 - busy-recovery
7 = P5
8 = P6 - initialization mode
9 = P7
10 = P8 - loading

Flags:

F76 - BSTCCREL, release buffers
F77 - BSF, set P/F flag to F
F78 - BSP, P/F flag to P
F79 - BSLSTXMTPRI, last transmit was primary
F80 - BSXMTING, transmission in progress

MULTIPLEX SYSTEM

The multiplex subsystem data structures are of two types: those that interface the multiplex subsystem to the other NPU software (such as TIPs) and those that concern the physical characteristics of lines, terminals, CLAs, modems, and hardware controllers for the lines.

The data structures in the system interface category are:

- MLCB - The format for this table is also used for the TPCB. In either case it contains information used for state programs.
- The multiplex command driver packet (command packet) which sets up the data transfer parameters.

The data structures in the hardware characteristics category are:

- Multiplex Port table (NAPORT) which has an entry for each line
- Line type tables
- CLA related tables
- Modem related tables
- Terminal related tables
- Device related tables

MULTIPLEX COMMAND DRIVER PACKET, NKINCOM

The command packet provides the interface between TIPs, LIP service module, etc., and the command driver, PBCOIN. This parameter list provides the necessary information for the multiplex subsystem to prepare the line for a transmission. Six standard formats are provided.

Set up commands - see section 5, multiplex command driver.

	15	7 0
0	NKCMD - command	NKLTYP - line type
1	NKPORT - I/O port	NKLOPOR - not used
2	NKCARY CHAR 1	NKCARY CHAR 2
	NKCARY	An 8-character array holding the command parameters
5	NKCARY CHAR 7	NKCARY CHAR 8

Function commands

	15	14	7	6 0
0	NKDM1 - not used		NKTCLS - default terminal class	
1	NKLINO - line number			
2	F1	NKFUN1	F2	NKFUN2
3	F3	NKFUN3	F4	NKFUN4
4	F5	NKFUN5	NKZERO - end of function	

NKFUN1-5 are function bytes

F1 - NKSRF1
F2 - NKSRF2
F3 - NKSRF3
F4 - NKSRF4
F5 - NKSRF5

Function selected flags

ASYNC TIP for IVT input

	15	14	13	12	11	10	9	8	7 0
0	NKDM2 - not used								
1	NKDM3 - not used								
2	NKIBP - Input buffer address								
3	F6	F7	F8	F9	F10	F11	F12	F13	Optional FCD of I/O buffer NKIFCD
4	F14	F15	NKDM9		NKBLKL - block length (words)				

F6 - NKUOP1
F7 - NKUOP2
F8 - NKUOP3
F9 - NKUOP4
F10 - NKUOP5
F11 - NKUOP6
F12 - NKUOP7
F13 - NKUOP8

User option flags 1-8. Can also be used as a single field, NKUOPS

Multiplex bit 15
.
.
.
.
.
.
Multiplex bit 8

in the MLCB, field NCUOPS

F14 - NKNOXL, Translate code flag; 1 = translate
F15 - NKSCENBL, Move special character flag

NKDM9 - not used

Set up for input processing (call from TIP or LIP)

	15	7	6	5 0
0	NKWD0	Word 0 and 1 of universal overlay		
1	NKWD1			
2	NKOBP - pointer to output buffer			
3	NKUOPS - user bits	F16	F17	NKISTAI - program index to input state
4	NKDM6 - not used			
5	NKISPTA - Pointer to input state table			
6	NKSCHR - special character	NKCNT1 - character counter 1 value for input state programs		
7	NKCXLTA - Translate table address			

F16 - NKMVB, move user bits to LCB flag
F17 - NKRPRT, strip parity flag

Universal input

	15	0
0	NKDM7 - not used	
1	NKDM8 - not used	
2	NKWD2	Universal overlay words
3	NKWD3	
4	NKWD4	
5	NKWD5	
6	NKWD6	
7	NKWD7	

Terminate I/O command

	15	7	6	5 0
0	NKDM10 - not used	F18	F19	NKWLINDX - worklist index
1	NKDM11 - not used			
2	NKJSRDY - user parameter for worklist	NKWKCOD - user work code if worklist requested		

F18 - NKRELBFS, release input buffer flag
F19 - NKWKFLG, make worklist for caller flag

NKWLINDX: Only bits 4 through 0 are valid

Values for NKCMD (first variant) are shown below. See section 5 for description of parameters list for each command.

Mnemonic	Value (hex)	Meaning
NKTURN	3	TURN LINE AROUND
NKINIL	4	INITIALIZE LINE
NKENBL	5	ENABLE LINE
NKINPT	6	INPUT
NKDOUT	7	DIRECT OUTPUT
NKOBT	8	OUTPUT BUFFER TRANSMITTED
NKINOUT	9	INPUT AFTER OUTPUT
NKENDIN	A	TERMINATE INPUT
NKENDOUT	B	TERMINATE OUTPUT
NKDISL	C	DISABLE LINE
NKCLRL	D	CLEAR LINE
NKCONTROL	E	CONTROL
NKSPECIAL	10	UPDATE MUX TABLE

Multiplex Line Control Block (MLCB), NCLCB
Text Processing Control Block (TPCB)

The MLCB is a dynamically allocated buffer obtained and released as a result of requests issued by the TIPs. The MLCB defines the processing functions to be provided by the multiplex subsystem. For a given communications line, there is one line control block for each enabled line.

Seven variants of the MLCB are provided. Some of these are TPCBs.

Usual TIP I/O data transfer request

<table>
<tr><th></th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>0</th></tr>
<tr><td>0</td><td>F1</td><td>F2</td><td>F3</td><td>F4</td><td>F5</td><td>F6</td><td>F7</td><td>F8</td><td colspan="5">NCOCHR - Next output character</td></tr>
<tr><td>1</td><td>F9</td><td>F10</td><td>F11</td><td colspan="5">NCTIME - Mux timer</td><td colspan="5">NCOBLCD - LCD of output buffer</td></tr>
<tr><td>2</td><td colspan="13">NCOBP - Pointer to output buffer</td></tr>
<tr><td>3</td><td>F12</td><td>F13</td><td>F14</td><td>F15</td><td>F16</td><td>F17</td><td>F18</td><td>F19</td><td>F20</td><td>F21</td><td colspan="3">NCISTAI - Input state program index</td></tr>
<tr><td>4</td><td colspan="8">NCCNTL - Character count limit</td><td colspan="5">NCCNT1 - Character counter 1</td></tr>
<tr><td>5</td><td colspan="13">NCISPTA - Pointer to input state program table</td></tr>
<tr><td>6</td><td colspan="13">NCIBP - Pointer to input buffer</td></tr>
<tr><td>7</td><td>F22</td><td>F23</td><td>F24</td><td>F25</td><td>F26</td><td>F27</td><td>F28</td><td>F29</td><td>F30</td><td>F31</td><td>F32</td><td colspan="2">NCCRCP - CRC polynomial</td></tr>
<tr><td></td><td colspan="8">NCUOPS</td><td colspan="5"></td></tr>
</table>

<table>
<tr><th></th><th>15</th><th>12</th><th>7</th><th>0</th></tr>
<tr><td>8</td><td colspan="2">NCSCHR - Special character</td><td colspan="2">NCIBFCK - FCD of input buffer</td></tr>
<tr><td>9</td><td colspan="4">NCCRCS - CRC accumulation</td></tr>
<tr><td>10</td><td>NCZER1 - Zero</td><td colspan="3">NCCNT2 - Character counter 2</td></tr>
<tr><td>11</td><td>NCZER2 - Zero</td><td colspan="3">NCBLKL - Block length (records)</td></tr>
<tr><td>12</td><td colspan="4">NCCXLTA - Pointer to code translate table</td></tr>
<tr><td>13</td><td colspan="4">NCSCBA - Pointer to first buffer in block</td></tr>
<tr><td>14</td><td colspan="2">NCBLCNT - Number of buffers allocated</td><td colspan="2">NCSVWL - Saved worklist</td></tr>
</table>

Word 15 is not used in this basic format

Flags:

F1 - NCEOBL, end of block
F2 - NCNXOCA, next output character available
F3 - NCLCT, last character transmitted (CDCCP)
F4 - NCBCREQ, buffer chaining required
F5 - NCOMPRO, output message in progress

F6 - NCINOUT, input after output expected
F7 - NCODDIN, ODD received
F8 - NCODBS, output data buffer size (not used)
F9 - NCSUPCHAIN, suppress buffer chaining
F10 - NCOBT, generate output buffer terminated (OBT)
F11 - NCBZL, reset timer
F12 - NCRINCH, input character in right byte
F13 - NCCAREC, character received
F14 - NCRIGHTC, left/right source flag (1 = right)
F15 - NCINPRO, input message in progress
F16 - NCNOXL, code translation active
F17 - NCRPRT, strips parity bit
F18 - NCSCF, suppress chain flag
F19 - NCLASTCH, LCD of source buffer reached
F20 - NCEOSR, end of source buffer reached
F21 - NCSP3, not used
F22 - NCUOP1
F23 - NCUOP2
F24 - NCUOP3
F25 - NCUOP4
F26 - NCUOP5
F27 - NCUOP6
F28 - NCUOP7
F29 - NCUOP8

optional user flags; can also be addressed as a single field NCUOPS

F30 - NCETX, delay ETX worklist generation
F31 - NCMRTO, modem response timed out
F32 - NCCARR, line carrier type (1 = controlled, 0 = constant)

Sixteen integer words

	15	0
0	NCCARRY	Integer
	NCCARRY (ARRAY)	
15	NCCARRY ELEMENT 16	

Eight user option words - includes half a word of flags

	15	7	0
0	NCWD0	user option words	
1	NCWD1		
2	NCWD2		
3	NCWD3		
4	NCWD4		
5	NCWD5		
6	NCWD6		
7	NCUOPS - User option flags	NCDMOM2 - Not used	

TPCB

15 14 7 5 0

0	NCLCDFCD - Source buffer LCD/FCD		
1	NCDUM2 - See MLCB (flags F9-F11, NCTIME and NCOBLCD)		
2	NCSBP - Source buffer pointers		
3	F33	NCFFLGS - Text processing firmware firmware (see F13 - F21 of MLCB) source flags mask	NCSTAI - Index to state programs
4	NCDUM4 - See MLCB (NCCNTL, NCCNT1)		
5	NCSPTA - Pointer to state programs table		
6	NCDBP - Pointer to destination buffer		
7	NCDUM5 - See MLCB (flags F22-F32)		
8	NCDUM6 - See MLCB (NCSCHR)	NCBFCD - FCD of buffer	
9	NCDUM7 - Not used		
10	NCDUM8 - See MLCB (NCCNT2)		
11	NCDUM9 - See MLCB (NCBLKL)		
12	NCDUMA - See MLCB (NCCXLTA)		
13	NCFDBA - Pointer to first destination buffer		
14	NCDUMB - See MLCB (NC8LCNT and NCSVWL)		
15	NCDUMC		
16	NCDUMD	Used only on CCI - MLCB	
17	NCDUME		
18	NCFSBA - First storage buffer address		

F33 - NCDCRB, character in right byte

Integer/File Register 1 TPCB - This MLCB has 16 words of INTEGER and 16 words for saving the first 16 file 1 registers (firmware level).

15 0

0	NCTPML	
	NCTPML (ARRAY)	16 integers
15	NCTPML ELEMENT 16	
⋮		

	15	0
16	NCTPF1	Spare for 16 file 1 registers
	NCTPF1 (ARRAY)	
31	NCTPF1 ELEMENT 16	

HASP TIP - Overlay for output text processing (TPCB)

	15	14	13	12	11	10	9	8	7	6	0
0	NPAD1				7 word array for standard TPCB						
6	NPAP1 - element 7										
7	F34	F35	F36	F37	F38	F39	F40	F41	F42	NPAD2 - not used	
8	NPAD 3				11 word array of integers						
18	NPAD3 - element 11										
19	NCCBLIMIT										03
20	NCCNBLIMIT										04
21	NCNCLIMIT										05
22	NCCMPINIT										06
23	NCCRB										07
24	NCNCINIT			Save space for file 1 registers							08
25	NCPAGEWDTH										09
26	NCLINE										0A
27	NCCOUNT										0B
28	NCCLIMIT										0C
29	NCXLTA										0D
30	NCSVCH										0E
31	NCPADH										0F

F34 - NCERROR - NCUOP1
F35 - NCDATA - NCUOP2
F36 - NCPLOT - NCUOP3
F37 - NCPUNCH - NCUOP4
F38 - NCMSG - NCUOP5
F39 - NCFORMAT - NCUOP6
F40 - NCWXPT - NCUOP7
F41 - NCTRPL - NCUOP8
F42 - NCBLNK - NCUOP9

User flags

HASP TIP - Overlay for input (two-pass) text processing

	15	14	13	12	11	10	9	0
0	NCFIL1							
							7 word array for standard TPCB	
6	NCFIL1 - element 7							
7	F43	F44	F45	F46	F47	F48	NCFIL2 - not used	
8	NCFIL3							
							11 word array of integers	
18	NCFIL3 - element 11							
19	NCSAVE							03
20	NCCMPBLK							04
21	NCNCMP							05
22	NCCARD						Save space for file 1 registers	06
23	NCCRECORD							07
24	NCLRCB							08
25	NCCMPNBL							09
26	NCFIL4							
27	NCFIL5							
28	NCCLAST						Save space for file 1 registers	0C
29	NCNXTL							0D
30	NCFIL6							
31	NCFIL7							

F43 - NCIA - NCUOP1
F44 - NCJOB - NCUOP2
F45 - NCENDSRC - NCUOP3
F46 - NCEOI - NCUOP4
F47 - NCEOR - NCUOP5
F48 - NCEOF - NCUOP6

user flags

Mode 4 TIP TPCB

	15	7 0
0	NC4PML	Array of 16 words reserved for standard TPCB
≈		
15	NC4PML - element 16	
16	NC4R1S - not used	NC4CHR - current character
17	NC4R2S - not used	NC4RCHR - right character
18	NC4SBA - pointer to first source buffer	
19	NC4ERR	
20	NC4CA - cluster address	NC4TA - terminal address
21	NC4CRCS - temporary CRC count	
22	NC4R3S	Array of 10 words
≈		
31	NC4R3S - element 10	

ASYNC TIP TPCB

	15	14	13	12	11	10	9	8	7	6	5	4 0
0	NCLCD - LCD								NCAFL1 - not used			
1	NCAFL7								Array of 6 words reserved for standard TPCB			
≈												
6	NCAFL7 - element 6											
7	F49	F50	F51	F52	F53	F54	F55	F56	F57	F58	F59	NCAFL2 - not used
8	NCAFL3								Array of 8 words			
≈												
15	NCAFL3 - element 8											
16	NCAFL4 - not used								NCCRCHR - current character			
17	NCAFL5 - not used								NCRTCHR - right character			
18	NCAFL6 - not used											
19	NCLFILCHR - LF and idle character											

⋮

	15	7 0
20	NCLNCNT - line character count	
21	NCPGWIDTH - line size	
22	NCPGCNT - page line counter	
23	NCPGLNGTH - page size	
24	NCONE - constant value = 1	
25	NCFESIZ - number of LFs in a formal effector	
26	NCFESAVE - FE save area for post-print	
27	NC10CHRCNT - 10 character counter (2741)	
28	NC255 - constant value = 255	
29	NCCRIDLES - CR idle value	NCLFIDLES - LF idle value
30	NCOCC - output character count (M1240)	
31	NCAPL - APL mode: 0 = no, 1 = yes (bit 0 only)	

F49 - NCTAPE, user option - paper tape
F50 - NCGRFC, user option - M1240 graphics
F51 - NCPL, user option: PL = 0 is page length - 1
F52 - NCFE, format effectors present (0 = none)
F53 - NCCALC, idle calculation needed for 2741/M1240
F54 - NCPWFE, format effector after page wait
F55 - NC1LSLF, optional line feed
F56 - NCNLFE, N line feeds exit
F57 - NCCORR, 2741 correlation flag
F58 - NCNOLF, no line feed on this terminal
F59 - NCEOP, end of page

ASYNC TIP - alternate TPCB

	15	14	7 6	0
0	NCAFL8			
≈			6-word array reserved for standard TPCB	
6	NCAFL8 - element 7			
7	F60	NCAFL9 - not used	F61	NAFL10 - not used

Flags:

NCUC - Upper case selected
NCPGWAIT - Page wait active

Port Table (NAPORT)

A Multiplex port table entry (NAPORT) defines information relating to each line. Entries are ordered by line number and an entry is provided for each port in the system. The Multiplex port table is the starting point of line orientation to the Multiplex subsystem. The Multiplex subsystem accessed the Multiplex port table to obtain modem and circuit related parameters necessary to establish the proper communication interface between the multiplex subsystem and a user communication line. The port table entry points to the MLCB which in turn points to the state programs which process data for the multiplex subsystem. Four variants are provided.

Normal Port Table

	15	14	13	12	11	10	7	6	5	4	3	0
0	F1	F2	F3	F4	F5	NALTYP - Line type - see appendix C		F6	NASPILL - CLA status count			
1	NALCBP - Pointer to MLCB											
2	NAOBTCMD - CLA turn around command						F7	F8	F9	F10	NAMSI - Index to state pointer table	
3	NAMSPTA - Pointer to modem state pointer table											
4	NAFCCST - CLA command status											
5												
6	NASTAT	not used										
7	NASPARE											

F1 - NAION, Input on
F2 - NAOON, Output on
F3 - NAISON, Input supervision
F4 - NALCBUP, LCB assigned
F5 - NAISR, CLA status pending
F6 - NAHARDER, Hard error in progress
F7 - NANDCD, Data carrier delete signal (DCD) dropped
F8 - NAMTO, Modem timeout in progress
F9 - NAWAIT, Timeout flag for first status overlay worklist
F10 - NAOVFE, First status overflow worklist received

Clearing Port Table Variant

	15	0
0	NAARY	NAARY (ARRAY)
≈		
7	NAARY ELEMENT 8	

Pointer/Flags Variant

This table allows the MLCB and the word 2 flags to be overlaid

	15	6	3 0
0	NADM3 - Not used		
1	NABFPTR - Buffer pointer		
2	NADM4 - Not used	NAFLAGS	NADM5 - not used

NAFLAFS - Overlay for flags F8, F9, and F10

General Overlay

	15	0
0	NAOVERLAY	
≈		8-word overlay for general use
7	NAOVERLAY - element 8	

Line Tables

MULTIPLEX LINE TYPE TABLE, NBLTYT

The line type table is an array of entries of type NBLTYE. Each entry corresponds to a line type in the system. See appendix C. The line type table entry defines the physical characteristics of a given port, modem circuit. Four variants are provided.

Normal Entry

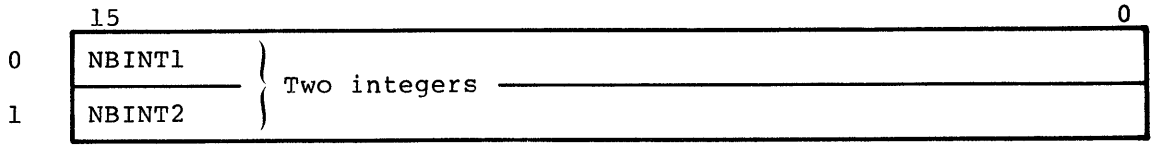
	15 14	13	12	11	10	9	8	4 0
0	NBSP1	F1	F2	F3	F4	F5	NBMODCLS - modem class; see below	NBOTYP - CLA type - see CLA constants above
1	NBAND - Mask							

NBSP1 - Not used

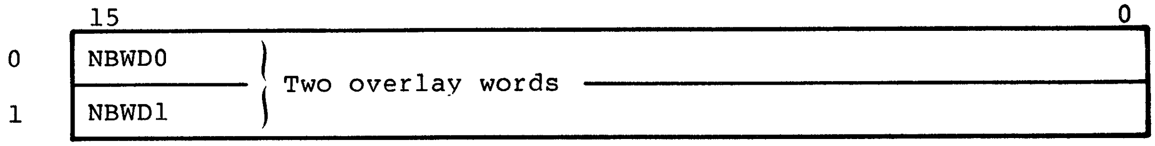
Flags:

F1 - NBTURN, Line turnaround required
F2 - NBDELAY, Delay the line turnaround
F3 - NBANSMOD, Answer mode: 0 = autorecognition, 1 = dedicated
F4 - NBCARR, Carrier type: 0 = constant, 1 = controlled
F5 - NBCIRTYP, Circuit type: 0 = 2 wire, 1 = 4 wire

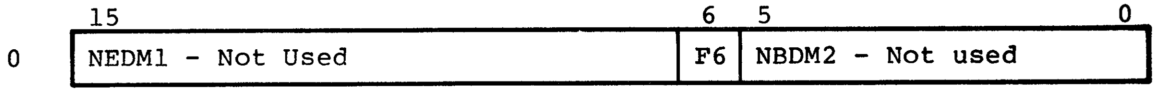
Integer Entry



Universal Overlay Entry



Overlay for Input Status Flag Word 0 Overlay



F6 - NBISR, Input status request

LINE TYPES, NOLTYP

This is the line type entry for the LCB. The sequence of line types is included in the SIT. SW indicates a switched (dial up) line; DE indicates a dedicated line. See appendix C.

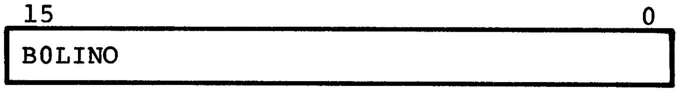
Mnemonic	Value (hex)	Meaning
NOLDIAG	0	RESERVED FOR ON-LINE DIAGNOSTICS
NOL1	1	2560-1 201A SW HDX CONTR 2WIRE
NOL2	2	2560-1 201B DE FDX CONTR 4WIRE (HDX MODE)
NOL3	3	2560-1 201B DE FDX CONST 4WIRE
NOL4	4	2560-1 208A DE FDX CONST 4WIRE
NOL5	5	2560-1 208A SW HDX CONTR 2WIRE
NOL6	6	2561-1 103E SW FDX CONST 2WIRE
NOL7	7	2561-1 103E DE FDX CONST 2WIRE
NOL8	8	2561-1 202S SW HDX CONTR 2WIRE REVERSE CHANNEL
NOL9	9	SPARE (UNDEFINED)
NOLA	A	2563-1 201B DE FDX CONST 4WIRE (SDLC)
NOLS	B	SPARE (UNDEFINED)
NOLAST	B	LAST LINE TYPE

ASYNCHRONOUS LINE SPEEDS

Mnemonic (Index)	Value	Baud Rate
N0B00	0	800
N0110	1	110
N0134	2	134, 5
N0150	3	150
N0300	4	300
N0600	5	600
N01200	6	1200
N02400	7	2400
N04800	8	4800
N09600	9	9600
N0DIAG	10	DIAGNOSTICS CLASS

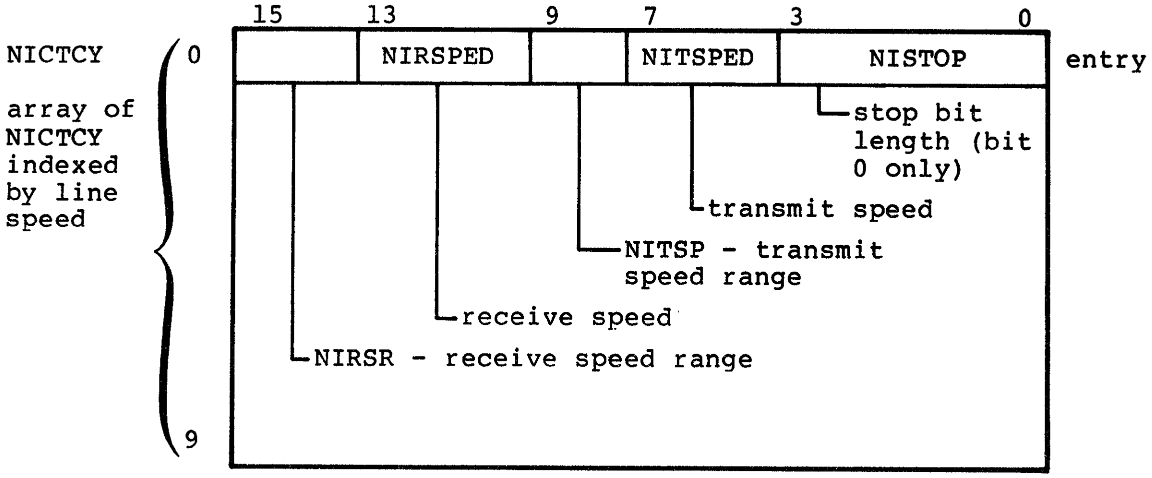
LINE NUMBER FIELD, B0LINO

This is the usual field used by the system to reference line number. It is used in the LCB, and line number fields compose the line array part of the SIT. Several routines define their own line number variable using B0LINO type as a basis.



MULTIPLEX CHARACTER TRANSMIT CHARACTERISTIC TABLE, NICTCT

The character transmission characteristics table is an array of 1 word entries (type NICTCY) indexed by the line speed index. Each entry specifies the speed range, speed, and number of output stop bits for transmitting and receiving to/from asynchronous terminals. An array of these entries is a part of the SIT.



NIRSP - Not used

NIRSPED - See asynchronous line speeds

NITSP - Not used

NITSPED - See asynchronous line speeds

NISTOP - 0 = 1 stop bit } for character delimiting
- 1 = 2 stop bits }

CLA/Modem Tables

MODEM/CLA RELATIONSHIPS

CLA Type	Maximum Modem Speed	Modem Class (hexadecimal)	Modems (The modems listed are only a sampling of modems available)
All	Not Applicable	0	None
2560-1 2560-2 2560-3 2563-1	Not Applicable	1	201B, 201A, 201C, 201D 208A, 208B 358-2
2561-1 Async	100	2	
	110	3	
	120	4	
	134.5	5	
	150	6	
	300	7	103 series, 113A, 113B, VA3405 A thru G
	600	8	VA3405 A thru G
	800	9	
	1050	A	
	1200	B	
	1600	D	
	2400	F	
	4800	10	
	9600	12	358-1

CLA TYPES

Mnemonic	Value	Meaning
N0SYNC	0	Synhronous CLA 2560-1
N0ASYNC	1	Asynchronous CLA 2561-1
N0NORS232	2	High-speed synchronous CLA 2560-3, 2560-4
N0SDLC	3	Trunk data line control for LIP protocol - CLA 2563-1

CLA COMMANDS AND STATUS

A control command sequence word (NDSEQE) is used by the multiplex level command driver, PMCDRV, to send commands to the CLAs. These commands are indexed as shown below. Four CLA Status words (8 byte) make up the two NPU/CLA status words (NRCCSE) and use a bit set method of checking the commands currently in effect for a given CLA/modem.

CONTROL COMMAND SEQUENCE WORD, NDSEQE

Used for multiplex commands to modem or circuit hardware. Three variants are provided.

Normal Entry

	15	7	6 0
0	NDDM1 - Not used	F1	NDCASE - Index to modem/circuit command case

F1 - Set function flag (0 = reset)

NDCASE is defined in the table below

Character Overlay

	15	7 0
0	NDDM2 - Not used	NDCHAR - Character

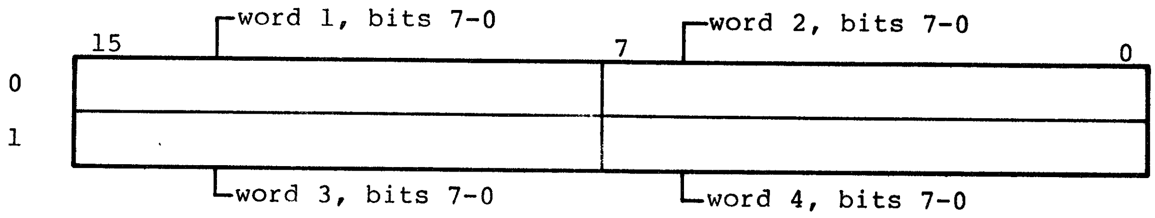
Universal Overlay

	15 0
0	NDWORD - Universal word

MULTIPLEX CLA COMMAND STATUS TABLE ENTRIES, NFCCSE

The CLA command status table reflects the current command status of each CLA. It contains the cumulative history of all physical commands sent to each CLA. Five variants are provided.

Bit Assignment Entry - This variant provides four eight-bit words with a name assigned to each bit. It is used to set and clear bits in the CLAs.



The individual bits are named as shown

NFW1B7
.
.
.
NFW1B0 } Word 1, bits 7 through 0

NFW2B7
.
.
.
NFW2B0 } Word 2, bits 7 through 0

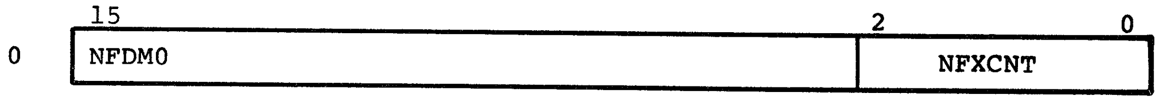
NFW3B7
.
.
.
NFW3B0 } Word 3, bits 7 through 0

NFW4B7
.
.
.
NFW4B0 } Word 4, bits 7 through 0

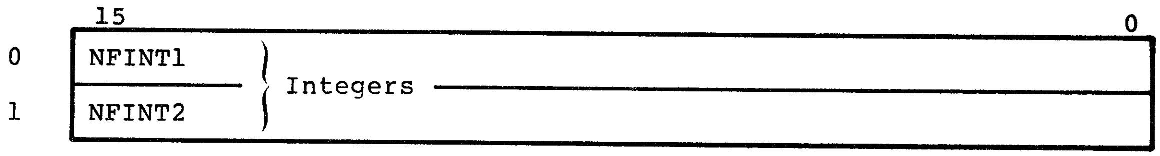
The assignment of bits is given in the table at the end of this paragraph.

SDLC CLA Entry

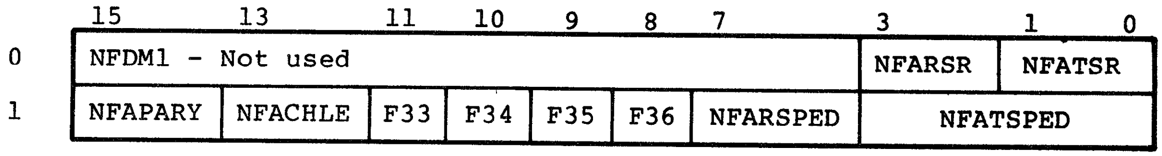
Defines SDLC CLA bit assignment.



Whole Word Variation



ASYNC CLA Entry



NFARSR - receive speed range (baud)

NFATSR - transmit speed range (baud)

NFAPARY - Parity:

0 = zero
1 = odd
2 = even
3 = none

NFACHLE - Character length (bits)

0 = 5
1 = 6
2 = 7
3 = 8

F33 - NFSTOP, stop bit

F34 - NFDM2, not used

F35 - NFECHO, Echoplex mode

F36 - NFLBT, currently in one-line diagnostic loopback test

NFARSPED - Receive speed (baud), see asynchronous line speeds

NFATSPED - Transmit speed (baud), see asynchronous line speeds

Synchronous CLA Entry

<table>
<tr><th></th><th>15</th><th>7</th><th>3</th><th>1 0</th></tr>
<tr><td>0</td><td colspan="2">NFDM3 - Not used</td><td>NFSPARY</td><td>NFSCHLE</td></tr>
<tr><td>1</td><td>NFSYCAR - Synchronous character</td><td colspan="3">NFDM4 - Not used</td></tr>
</table>

NFSPARY - parity

0 = zero
1 = odd
2 = even
3 = none

NFSCHLE - character length (bits)

0 = 5
1 = 6
2 = 7
3 = 8

The following table (not a data structure) correlates command index to command status.

Mnemonic for NDCASE	Value NDCASE (hex)	NFCCEE (Word/bit)	Meaning	Sync/ Async or general
N0RTS	1	(W1B7)	(RTS) Request to send	-
N0SRTS	2	(W1B6)	(SRTS) Secondary request to send	A
N0RSYN	2	(W1B6)	(RSYN) Resync	-
N0OM	3	(W1B5)	(OM) Originate mode/auxiliary	A
N0LM	4	(W1B4)	(LM) Local mode/auxiliary	A
N0NSYN	4	(W1B4)	(NXYN) New sync	S
N0LT	4	(W1B4)	(LT) Local test (2560-3)	-
N0DTR	5	(W1B3)	(DTR) Data terminal ready	-
N0TB	6	(W1B2)	(TB) Terminal busy	A
N0ION	7	(W1B1)	(ION) Input on	-
N0OON	8	(W1B0)	(OCN) Output on	-
N0BREAK	9	(W2B7)	(BREAK) Break mode	A
N0ISR	A	(W2B6)	(ISR) Input status request	-
N0ISON	B	(W2B5)	(ISON) Input supervision on	-
N0DLM	C	(W2B4)	(DLY) Data line monitor	A
N0ECHO	D	(W3B1)	(ECHO) Echoplex mode	A
N0LBT	E	(W3B0)	(LIT) Loopback test	A
N0LBT	E	(W2B4)	(LIT) Loopback test	S
N0LBT	E	(W2B4)	(LIT) Loopback test	†
N0LBT	E	(W2B4)	(LIT) Loopback test	SDLC
N0PON	F	(W3B6)	(PON) Parity on	A
N0PON	F	(W2B2)	(PON) Parity on	S
N0PON	F	(W2B2)	(PON Parity on	†
N0PSET	10	(W3B7)	(PSET) Parity set, 1 = even	A
N0PSET	10	(W2B3)	(PSET) Parity set, 0 = odd	S
N0PSET	10	(W2B3)	(PSET) Character length - LSB	†
N0CLLS	11	(W3B4)	(CLLS) Character length - LSB	A
N0CLLS	11	(W2B0)	(CLLS) Character length - LSB	S
N0CLLS	11	(W2B0)	(CLLS) Character length - LSB	†
C0CLMS	12	(W3B5)	(CLMS) Character length - MSB	A
N0CLMS	12	(W2B1)	(CLMS) Character length - MSB	S
N0CLMS	12	(W2B1)	(CLMS) Character length - MSB	†

---

† Not RS-232

CLA STATUS CONDITION INDICATORS, MOSCTYP

The status indicators are used in the worklist entry.

MOCLAON,	0	CLA ON DETECTED
MORING,	1	RING INDICATOR DETECTED
MOENBL,	2	LINE ENABLED
MOHERR,	3	HARD ERRORS DETECTED
MOSOER,	4	SOFT OUTPUT ERRORS DETECTED
MOSIER,	5	SOFT INPUT ERRORS DETECTED (unsolicited input)
MOSTRT,	6	START MODEM TIMEOUT
MOSTOP,	7	STOP MODEM TIMEOUT
MOOVRF,	8	CLA STATUS OVERFLOW (unsolicited output)
MOOVTO,	9	CLA STATUS OVERFLOW TIMEOUT
MOMRTO,	A	MODEM RESPONSE TIMEOUT
MOBREAK);	B	BREAK FROM FRAMING ERROR STATUS
MONSDCD	C	SDCD dropped on output - possible break
MOSDCD	D	SDCD on - ready for output
MOSDTO	E	SDCD timeout - break
MOSD2TO	F	SDCD timeout - disconnect

MODEM CONTROL STATES

These status are used in the command packet to PBCOIN to set up the modem's state of operation.

Mnemonic	Value	Meaning
MSTCHK	0	STATE 0
MSTERR	1	STATE 1 LINE CLEARED
MSTLNI	2	STATE 2 LINE INITIALIZED
MSTENB	3	STATE 3 LINE ENABLED
MSTIDL	4	STATE 4 LINE IDLED
MSTOUT	5	STATE 5 OUTPUT ON
MSTINP	6	STATE 6 INPUT ON
MSTRFO	7	STATE 7 READY FOR OUTPUT ON REVERSE CHANNEL

MODEM STATE PROGRAMS

NOMSPT has range 0..40; this is the size of modem states pointer table. One table exists for multiplex modem state pointers subsystem.

Terminal Tables

TERMINAL CHARACTERISTICS TABLE, NJTECT

The terminal characteristics table entry (NITECY) contains parameters which define the special processing characteristics of a given terminal type. It is used to set up the MLCB and to configure the system (SVM use). The variant is accessed when the IVT parameters are used.

15 14 13 12 11 9 8 7 3 0

0	NJISPTA - address of input status pointer table						
1	NJCXLTA - address of code translate table						
2	NJCNT1 - Input character count 1	NJSYNC - Sync character					
3	NJORCP - CRX polynormal index	NJIBFCD - FCD of first buffer (bits 7-0 only)					
4	NJBLKL - block size (words)	NJTIPTY - TIP type - see appendix C					
5	NJPARIT -Parity	NJCHLEN- Character length	NJPARTY- ASYNC TIP parity	F1	F2	NJSP1 - Not used	
6	NJPSWIDTH - Page width	NJPGLENGTH - Page length					
7	NJCANCHAR - Character for cancel input line	NJCNTRLCHAR - Control character					
8	NJUSF1 - Character for user break 1	NJUSR2 - Character for user break 2					
9	F3	F4	F5	F6	NJXCNT - Counter for transparent character		
10	F7	F8	F9	F10	NJOUTDE output device	NJAPL- APL mode	NJXCHAR - Character which delimits transparent text
11	NJBSCHAR - Backspace character	NJABTLINE - Character to abort output line					
12	NJORIOLES - Count of idles following a CR	NJFIDLES - Count of idles following an LF					

IVT parameters are in words 6 through 12 IVT variant

IVT variant

15 0

0	NJARRY	
	NJARRY (ARRAY)	overlay for IVT parameters
14	NJARRY ELEMENT 15	

Terminal Classes

This is the BZTIPTYPE field of the LCB. Further information on terminal class is found in appendix C.

Mnemonic	Value	Meaning
N0TMLIA	0	MLIA
N0M33	1	ASYNC - M33, M35, M37, M38
N0713	2	- CDC 713
N02741	4	- IBM 2741
N0M40	5	- M40
N0H2000	6	- HAZELTINE 2000
N0751	7	- CDC 751-1
N0T4014	8	- TEKTRONIX 4014
N0HASP	9	HASP
N0200UT	10	MODE 4 - 200UT
N0214	11	- 214
N0711	12	- 711-10
N0714	13	- 714
N0731	14	- 731
N0734	15	- 734
N0TCOUPLER	16	COUPLER
N0TCONSOLE	17	CONSOLE
N0THDLC	18	HDLC LIP
N0TDIAG	19	DIAGNOSTICS

Values

NJPARIT	0 = zero	1 = odd	2 = even	3 = none
NJCHLEN	0 = 5 bits	1 = 6 bits	2 = 7 bits	3 = 8 bits
NJPARTY	0 = zero	1 = odd	2 = even	3 = none
NJOUTDE	0 = printer	1 = display	2 = paper tape	3 = not used
NJAPL	0 = No	1 = Yes	2 = Special APL mode	3 = not used

Flags:

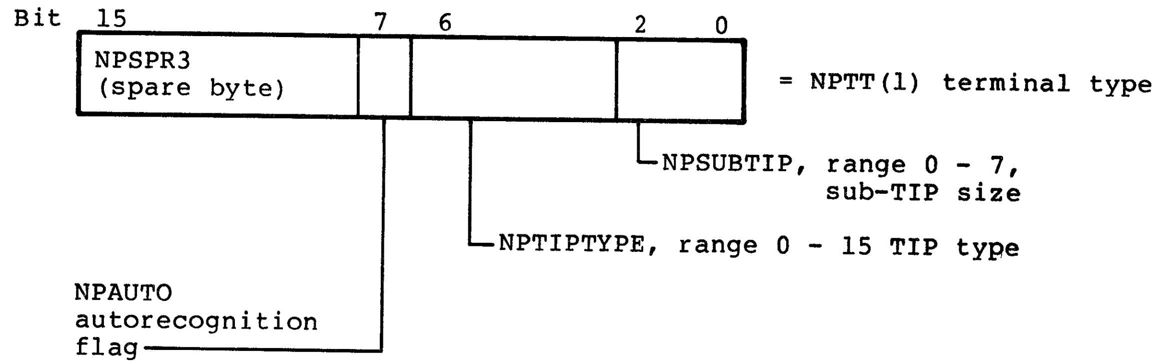
F1 - NJPGWAIT, Page wait mode
F2 - NJXPARENT, Input transparent mode
F3 - NJXTO, Expected delimiter is a timeout
F4 - NJXCHRON, Expected transparent delimiter is a delimiting character
F5 - NJDUM2, Not used
F6 - NJDUM1, Not used
F7 - NJCRCALC, Calculate CR idle count
F8 - NJLFCALC, Calculate LF idle count
F9 - NJECHOPLX, Echoplex mode
F10 - NJINDEV, Input device (0 = keyboard, 1 = paper tape)

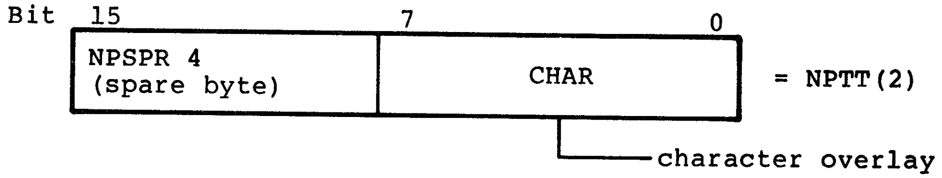
Terminal and Device Types (TT/DT)

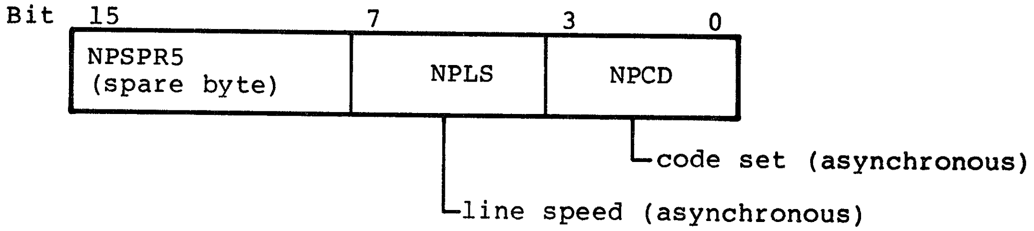
These data structures are used to find TCBs, check devices for deliverability of messages, etc. See appendix C.

TERMINAL TYPE

Three cases are possible:

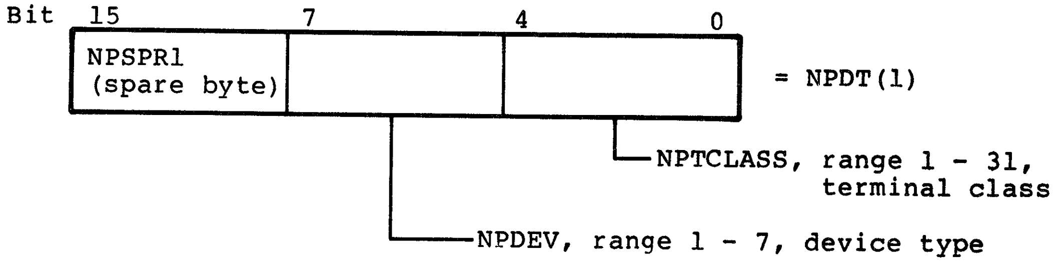


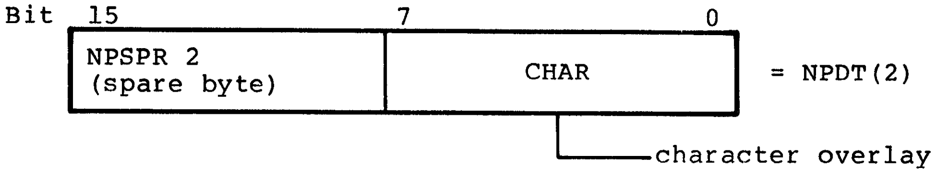




DEVICE TYPE

Two cases are possible:





DEVICE TYPES

These mnemonics are used by programs to determine if device type is proper for delivery of message, generating status, and so forth.

Mnemonic	Value	Meaning
N1CON	0	Console
N1CR	1	Card reader
N1LP	2	Line printer
N1CP	3	Card punch
N1PLOT	4	Plotter
N1INTDEV	7	Internal device

SERVICE MESSAGES

Appendix C defines most of the service message data structures. Table C-1 defines the function and subfunction codes used to switch processing within the SVM to the indicated SVM routines. Appendix B of The CCP3 Reference Manual defines the error-related service message structures (CE error and statistics messages).

Definition	Fields	Location, table	
TIP/Sub-TIP	N0...	Appendix C	
Line type		Appendix C	
Configuration states	C7...	Appendix C	
CE error messages	CN...	Appendix B	CCP3 Reference Manual
Statistics messages			
(NPW)	CP...	Appendix B	
(tr/ln)	BZ...	Appendix B	
(term)	BS...	Appendix B	

FN/FV DATA STRUCTURES

These data structures are used when taking FN/FV parameters from the configure service messages and entering them in the appropriate place (usually in the TCB).

Field Description Table, DDFDTRECORD

The field descriptor table size is given by DDFTDRECORD in the first word. A series of one-word entries (DDFDTENTRY) follow.

DDFTDENTRY

15	11	7 0
DDFTDRECORD - Number of table entries		
DDFSTRT - field	DDFLNTH - field	DDFDISP - displacement to start of field in record
ARRAY OF DDFTDENTRYs		

Pointer to table is DDFTDPTR

Action Table Entries, DFATENTRY

The action table is used for configuring lines and terminals. There can be an entry in the table associated with the field number (FN) of each possible FN/FU pair in the configure service message. Normal values for the entries can be found in the link-edit listing normal table entry.

15	7 0
DFERRCDE - Error code	DFFN - Field number (table index)
DFRKEY - Reconfigure action key	DFCKEY - Configure action key
DFPARAM - Optional action parameter	

Table end

15 0
DFEND - End of table

Pointer to the table is DFATPTR.

Configure Action Codes - Each TIP has associated with it an action table which is set up in a link edit operation. After storing the field value (FV) in the TCB, PNCONFIGURE checks the TIPs action table using the action code as an index, and takes the action specified by the PNCONFIGURE subroutine.

DFCKEY or DFRKEY

D2NA	0	NO ACTION
D2VUL	1	VERIFY UPPER AND LOWER VALUE
D2VU	2	VERIFY UPPER VALUE
D2VL	3	VERIFY LOWER VALUE
D2ACN	4	PROCESS CONNECTION NUMBER
D2LLCB	5	TRANSFORM ON, SN IN LLCB ADDRESS
D2TCPCHN	6	CHAIN TCB
D2INK	7	GET INDEX INTO LINK TABLE FROM LRN (TRUNKS ONLY)
D2TCB	9	MOVE IN DEFAULT TCB VALUES
D2PARITY	10	RESTORE CR IDLE COUNT
D2TCBDFLT	11	RESTORE LF IDLE COUNT
D2CRIDLE	12	VALIDATE J100 APL CODE
D2LFIDLE	13	RESTORE LF IDLE COUNT
D2APL	14	VALIDATE APL CODE
D2INIT	15	EMPTY OUTPUT QUEUE AND SEND INIT
D2TCBINIT	16	SET UP VARIANT TCB
D2SKP	17	SKIP TO NEXT ACTION CODE
D2VM	18	VERIFY MIDDLE VALUE
D2CODECK	19	CHECK AND SET CODE TYPE

Configure action error codes - If the action specified by the action table cannot be completed, the acting PNCONFIGURE subroutine sets an error code (DEFERRCDE) in the action table entry which commanded the action. Other SVM routines use this code to generate the configure SM reply (normal or error) to the host.

DFERRCDE

D3AC	0	ACTION COMPLETE
D3FNFVERR	1	FIELD NUMBER OR FIELD VALUE OUT-OF-RANGE
D3INVCB	2	INVALID CONTROL BLOCK ID
D3CNFERR	3	CONTROL BLOCK ALREADY CONFIGURED (CONFIGURE SM) CONTROL BLOCK NOT CONFIGURED (RECONFIGURE SM)
D3NOBFR	4	NO BUFFER FOR TCB
D3INVLT	4	INVALID LINE TYPE
D3PRNDL	4	LINE PRINTER OR CARD READER STILL CONFIGURED
D3INVTT	5	INVALID TERMINAL TYPE
D3INVDT	5	INVALID DEVICE TYPE
D3NOTENABLED	6	LINE NOT ENABLED
D3HOTGERR	7	HOST ORDINAL TOGGLE ERROR
D3NOL	8	LOGICAL LINK NOT ESTABLISHED
D3CNINUSE	9	CONNECTION NUMBER, ALREADY IN USE
D3CONNCFG	10	CONSOLE NOT YET CONFIGURED
D3NOTCNF	11	LINE IS NOT CONFIGURED

ON-LINE DEBUGGING AIDS

The on-line debugging aids for CCP include the Test Utility Package (TUP) and other aids. These debugging aids offer a variety of interactive commands useful to the programmer who is altering CCP code or adding a new TIP to the system. Several breakpoint commands are available.

NOTE

> These on-line debugging aids are not a supported product. The descriptions are given here because of their usefulness. However, the user should be cautious about any analysis based on the use of these debugging aids.

CONSOLE COMMANDS

Commands for on-line debugging are entered through the NPU console. A special character (control A) places the console in debug mode. In this mode, the console is an interactive device. In addition to the standard machine language debugging features, there are aids based on the internal structure of the software (such as dumping a line control block (LCB) or making a worklist entry). Various machine language level breakpoints are also available. These debugging aids allow one or more breakpoints per machine instruction.

INSTALLING DEBUGGING AIDS

The on-line debugging aids are an optional feature. They are made available by using the Update command

```
*   DEFINE DBUGALL
```

during the build process. During the MPEDIT phase, the global to console must be set to true.

GENERAL COMMAND FORMAT

Once the debugging system is activated, it accepts any of the commands listed in table I-1. Rules for entering the commands are as follows:

- Control A allows the user to enter debug mode. The control A must be recognized as the first character of the input message.
- Control D allows the user to leave the debug mode.
- Each command can include up to eight parameters. Each parameter field includes one to five hexadecimal characters (18-bit addressing is supported).

- Commas or blanks delimit the parameters. These symbols are interchangeable.
- A slash (/) delimits the end of a command or the end of a command line.
- Control C or question mark (?) cancels a partially entered debugging command.
- Shift O or control H are used for backspacing.
- An error message (*ERR) is printed in response to an invalid input. The usual invalid inputs are a bad command mnemonic, the wrong number of parameters, or a parameter containing nonhexadecimal characters.

TABLE I-1. DEBUGGING AID COMMANDS

Command	Syntax
OPS Halt	OH/
OPS Restart	OR/
Dump Memory	DP$\begin{Bmatrix}C\\L\end{Bmatrix}$,start, stop, base/
Load Memory	LHX, start, base/C, word 1, ... word 8/
Display Register	DR/
Enter Register	E R / where R is 1, 2, 3, 4, Q, A, I, or M
Display File 1	DR, file 1 register (0 .. X'FF)/
Enter File 1	EF, file 1 register 0 .. X'FF)/
Get a Worklist	BG, buffer size (0 .. 3)/
Release a Buffer	BR, buffer address, buffer size (0 .. 3)/
Get a Worklist	LG, worklist number/
Put a Worklist	LP, worklist number, word 1, ... word 6/
Device Assignment	DA, LIP, PD/
Dump OPS Program	DM$\begin{Bmatrix}P\\L\end{Bmatrix}$, start, end, OPS worklist number/
Load OPS Program	LDX, start, OPS worklist number/C, word 1, ... word S.
Read Page Register	RP, page number + X'8000*bank/
Dump LCB	LC$\begin{Bmatrix}B\\L\end{Bmatrix}$,line number/
Dump LLCB, TCB	TC$\begin{Bmatrix}B\\L\end{Bmatrix}$,DN, SN, CN/
Search for TCB	TS$\begin{Bmatrix}B\\L\end{Bmatrix}$,line number, CA, TA, DT/
Enter Breakpoint	EB, inst. start, inst. stop, BP code, optional parameters/
Remove Breakpoint	RB, inst. start, inst. stop, BP code/
Enable Software BP	BL, software priority level (0 .. X'11)/
Disable Software BP	DL, software priority level (0 .. X'11)/
Breakpoint Restart	RS/

COMMAND FORMATS

Each command is described individually in this subsection. The normal response to the command is also given. Two types of responses occur:

- Debug asks for more parameters (such as where a Load Hexadecimal command is used). These additional parameters alway use a C command in the form

  ```
  C, word 1, ... , word 8/
  ```

  Word is a hexadecimal value (00000-$FFFFF_{16}$) (5-character hexadecimal should be used only for addresses above ($FFFFF_{16}$)

  , or   is the delimiter
  / ends the input.

- Debug returns results or a comment. The return always begins with *.

In the following, the syntax of the input is given on the first line and the format of the normal response is given on subsequent lines.

OPS Halt

The OPS halt command stops OPS-level processing in the system. All other debug commands can be entered while the system is in this mode.

```
OH/
*
* OPS HLT
```

The error response *ERR SYS HLT is returned if the OPS level is already halted.

OPS Restart

This command returns control to the OPS level after an OPS halt.

```
OR/
*
```

The error response *ERRR SYS HLT prints if the OPS level is not halted.

Dump Memory

```
{DPC}
{DPL}, start address, stop address, base address/

        * dump address    word 1 ... word 8
        * dump address +8 word 9 ... word 16
                    etc.
```

The DPC command displays the memory contents within the specified range on the local console. The DPL command dumps memory to the assigned dump device.

The base address is optional and is used for relative addressing. If only the start address is entered, one word of memory is dumped.

An error response is returned if the user attempts to dump outside the memory range.

A DR/command can be repeated without re-entering the command by pressing the manual interrupt (control G) key.

Load Memory

```
LHX    start address, base address/
*
C, new word 1, ... new word 8/
* load address  old word 1 ... old word 8
```

The LHX command sets up the load address. The C command loads from one to eight words into memory. The load address is incremented for each word loaded. Thus, mulitple C commands load contiguous memory. Other debug commands (except a LHX command) can be executed between C commands without disturbing the load address. The previous contents of the loaded memory locations are displayed in response to a C command. If the debugger tries to load an out-of-range location, dashes print following the contents of the last in-range location.

Display Registers

The contents of macro registers R1, R2, R3, R4, Q, A, I, and M are displayed. The command gives valid information only if the system is in the OPS halt, breakpoint halt or system halt mode.

```
DR/
*1 = contents of R1 ... M = contents of M
```

Enter Register

The specified register is loaded. This command is accepted only in the OPS halt or breakpoint halt modes.

```
E{R}, value/  where R is 1, 2, 3, 4, Q, A, I, or M
* previous register contents
```

Display File 1

The contents of the specified micro file 1 register is displayed. A series of file 1 registers can be displayed quickly by using the manual interrupt (control G) key. After the initial display file 1 command, the next file 1 register is displayed by pressing manual interrupt.

DF, file 1 register (0-FF_{16})/
* register contents

An error response is displayed if the file 1 register number is too large.

Enter File 1 Register

A specified file 1 register is loaded with a given value.

```
EF, file 1 register (0 .. FF16), value/
* previous file 1 register contents
```

An error response is displayed if the file 1 register number is too large.

Get a Buffer

A buffer of a given size is obtained.

```
BG, buffer size (0..3)/
* buffer address
```

An error response is displayed if the buffer size is too large.

Release a Buffer

A given buffer is returned to the free buffer pool.

NOTE

> No real error checking is performed by the Release a Buffer command. Incorrect use of this command can cause a system halt.

```
BR, buffer address, buffer size (0..3)/
*
```

An error response is displayed if the buffer size is too large.

Get a Worklist Entry

The next entry from the specified worklist is removed and printed. If the worklist is currently empty, *LIST EMPTY is printed.

```
LG, worklist number/
* worklist entry word 1 ... worklist entry word 6/
```

An error response is displayed if the worklist number is too large.

Put a Worklist Entry

The given worklist entry (zero to six words) is placed into the specified worklist. OPS-level programs can be exercised with the command. First, halt OPS level scheduling via the OPS halt command. Next, place the desired worklist entry or entries into the desired OPS-level worklist(s). Finally, return control to OPS scheduling using the OPS Restart command. The queued worklist entries are worked off and results can be verified.

```
LP, worklist number, word 1, ... word 6/
*
```

An error response is displayed if the worklist number is too large.

Device Assignment

This command allows the user to dynamically assign logical input/output functions (LIO) to physical devices (PD). The available PD codes are as follows:

0	Null device
1	Local console
2	Line printer

The currently defined LIO codes are as follows:

8	Dump device
9	Memory snapshot
XA_{16}	Register snapshot
XB_{16}	Breakpoint return address snapshot
XC_{16}	Spare breakpoint
XD_{16}	Quick output

The default for all LIO codes except the dump device is the local console. The dump device is the local line printer if the line printer software is built into the system.

```
DA, LIO, PD/
*
```

An error response is displayed if either parameter is too large.

Dump OPS Program Locations

This command is similar to the DP command, which uses the base address feature. Instead of a base address, however, the user enters the desired OPS program worklist number. The correct OPS program base address is obtained from a pre-built table.

```
{DMP}
{DML}, start address, end address, OPS wl number/
 * dump address    word 1 ... word 8
 * dump address +8 word 9 ... word 16
              etc.
```

The DMP command dumps to the local console. The DML command dumps to the assigned dump device. All three parameters are mandatory. An error response is printed if the OPS worklist number is too large.

NOTE

> When the OPS programs are paged above 64K (FFF_{16}), the necessary paging is automatically performed.

Load OPS Program Location

This command is similar to the LHX command, which uses the base address feature. Instead of a base address, however, the user enters the desired OPS program worklist number. The correct OPS program base address is obtained from a pre-built table.

```
LDX, start address, OPS wl number/
*
C, new word 1, ... new word 8/
* load address  old word 1 ... old word 8/
```

NOTE

When the OPS programs are paged above 64K, the necessary paging is automatically performed.

Read Page Register

In NPUs with the paging feature, page registers in either bank can be displayed. Writing a page register while the system is on-line is quite hazardous and is not allowed. The leftmost bit of the page number parameter determines which bank to read: $0..1F_{16}$ for bank 0 and $8000..801F_{16}$ for bank 1.

```
RP, page number/
* page contents.
```

An error response is displayed if the page number is out of range.

Dump Line Control Block

Given a line number, the corresponding line control block (LCB) is dumped. The line number is a 16-bit quantity containing the port (left 8 bits) and subport (right bits), subport = 00.

$\begin{Bmatrix} \text{LCB} \\ \text{LCL} \end{Bmatrix}$, line number/

```
*LCB start address    word 1 ... word 8
*LCB start address +8 word 9 ... word 16
                  etc.
```

LCB dumps to the local console. LCL dumps to the assigned dump device. An error response is displayed if either the port or subport is too large for the configured system.

Dump Terminal Control Block or Logical Link Control Block by DN, SN, and CN

If the CN is zero, the logical link control block (LLCB) is dumped. Otherwise, the terminal control block (TCB) is dumped. The DN and SN (and CN) form the logical network address and a search through the routing directory is performed to find the proper control block.

```
⎰TCB⎱
⎱TCL⎰, DN, SN, CN/
* control block start address    word 1 ... word 8
* control block start address +8 word 9 ... word 16
                                 etc.
```

TCB dumps to the local console. TCL dumps to the assigned dump device. An error response is displayed if the control block is not found in the routing directory.

Dump Terminal Control Block by Line Number, CA and TA

The line number, cluster address and terminal address form the physical network address of the terminal control block (TCB) and a search through the active line control blocks is performed to find the TCB.

```
⎰TSB⎱
⎱TSL⎰, line number, CA, TA, DT/
* TCB start address    word 1 ... word 8
* TCB start address +8 word 9 ... word 16
                       etc.
```

TSB dumps to the local console. TSL dumps to the assigned device. An error response is displayed if the TCB is not found.

Enter Breakpoint

This command places an entry into the software breakpoint table (JEBPTABLE). The entry consists of the starting and ending addresses of the instruction to breakpoint, the breakpoint code specifying which breakpoint to execute and any optional paraemters required by the breakpoint. A maximum of five optional parameters are allowed.

```
EB, instruction start, instruction stop, breakpoint code, parameter
1, ... parameter 5/
*
```

The following conditions cause an error response to be displayed:

- Breakpoint table full
- Start address, end address and/or breakpoint code missing
- Start or end address out-of-range

Breakpoint codes are discussed below.

Remove Breakpoint

This command removes a specified entry from the breakpoint table. Only matches with the instruction start and end addresses and the breakpoint code are searched for in the breakpoint table. An error response is displayed if the wrong number of parameters are entered or if the entry is not found.

```
RB, instruction start, instruction end, breakpoint code/
*
```

Enable Software Breakpoint by Priority Level

This command allows software breakpoints to occur at a specific software priority level. This allows reentrant code which is executed at difference priority levels to be breakpointed at a specific priority level or levels.

BL, priority level $(0..11_{16})$/
*

An error response is displayed if the priority level is too large.

Disable Software Breakpoint by Priority Level

This command disables software breakpoints on a specific priority level.

DL, priority level $(0..11_{16})$/

An error response is displayed if the priority level is too large.

SOFTWARE BREAKPOINTS

Software breakpoints on the NPU are generated through the hardware program protect system. When the system is initialized, all of memory except the dynamic buffer area is protected. That is, the program protect bits are set on each nonbuffer memory location. When a breakpoint is set on an instruction, the program protect bits are reset for that instruction. When the protected instruction following the unprotected (breakpoint) instruction is executed, a program protect interrupt (line 0) is generated, provided the program protect system is activated. The instruction generating the interrupt executes as a one-word NOP. The line 0 interrupt handler passes control to the breakpoint handler. The breakpoint interrupt handler searches the breakpoint table using the interrupt return address for line 0. If an entry in the breakpoint table is not found, a true program protect fault has occurred and the system is halted. Otherwise, control is passed to the proper breakpoint handler for each entry found in the breakpoint table, provided software breakpoints are enabled for the interrupting priority level. Note that more than one breakpoint entry per instruction is allowed.

A basic knowledge of the macro assembly language is necessary when using software breakpoints.

Certain restrictions must be observed when using software breakpoints.

Instructions that write into nonbuffer memory, jump, return jump or skip, or are priviledged (disable and enable interrupts, set and clear protect bit, and interregister instructions with the interrupt mask register as the destination register) cannot have breakpoints. The enter breakpoint command is:

```
EB, start global area, end global area, 0/
```

This clears the protect bits on all global variables, allowing the user to breakpoint instructions that write into the global area.

Two consecutive instructions cannot have breakpoints. Noninterruptable code cannot have breakpoints.

Note that both the proper software priority and the program protect system must be active before a breakpoint interrupt can occur. The program protect system is activated by entering J28: on the NPU maintenance panel. Entering J20: deactivates the program protect system.

The global constant J1BREAKMAX specifies the number of entries in the breakpoint table JEBPTABLE. Currently, J1BREAKMAX is 10.

BREAKPOINT HANDLERS

Currently, there are seven breakpoints handlers available:

- Enter debug mode
- Memory snapshot
- Register snapshot
- Instruction address snapshot
- Quick output
- Wraparound snapshot
- User-defined snapshot

The enter debug mode breakpoint enters a loop after the breakpoint instruction executes. In this loop, all priority levels at and below the breakpoint priority level are suspended until the loop is exited using the breakpoint restart debug command. All debug commands can be entered while in the breakpoint loop.

The memory snapshot formats a specified memory range into system buffers and queues them to a specified local peripheral.

The register snapshot formats the contents of macro registers R1, R2, R3, R4, Q, A, I, and M into a system buffer and queues it to a specified local peripheral.

The instruction address snapshot places the address of the breakpoint instruction into a system buffer and queues it to the memory snapshot local peripheral.

Quick output writes the contents of one buffer of ASCII characters to a specified local peripheral.

The wraparound snapshot places the contents of a specified memory range into a user supplied circular save area.

The user-defined snapshot consists of 20 NOPs available to contain user-written breakpoint code.

The local peripheral for the above snapshots is specified by the device assignment debug command.

Combinations of the above snapshots can be entered for a single breakpointed instruction. Table I-2 defines the optional parameters for the Enter Breakpoint Debug command. The execution count is the maximum number of times the snapshot is to be executed.

OPS SCHEDULED DEBUG AID

A special OPS scheduled program (PBTIPDBG) is available to execute user-supplied debug code. PBTIPDBG is entered by making a worklist entry from source code or through the List Put Debug command (LP, parameters, see table I-1). The first word of the worklist entry is a code defining which user code to execute. The next four words are optional and are used to pass parameters to the user code. Code 0 is reserved and contains 20 NOPs available for on-line patching.

TABLE I-2. BREAKPOINT PARAMETERS

Breakpoint Code (Hex)	Breakpoint	Parameter Number and Description
7	Enter debug mode	No parameters
9	Memory snapshot	1 - Snapshot start address 2 - Snapshot end address 3 - Execution count
A	Register snapshot	1 - Execution count
B	Instruction address snapshot	1 - Execution count snapshot
C	User-defined snapshot	1 - Execution count
D	Quick output	1 - Address of buffer to output 2 - Execution count
E	Wraparound snapshot	1 - Start address of snap area 2 - End address of snap area 3 - Start address of save area 4 - End address of save area 5 - Execution count

INDEX

COMMENT SHEET

MANUAL TITLE CCP Version 3 System Programming Reference Manual

PUBLICATION NO. 60474500 REVISION A

FROM: NAME:

BUSINESS

ADDRESS:

COMMENTS:
This form is not intended to be used as an order blank. Your evaluation of this manual will be welcomed by Control Data Corporation. Any errors, suggested additions or deletions, or general comments may be made below. Please include page number references and fill in publication revision level as shown by the last entry on the Revision Record page at the front of the manual. Customer engineers are urged to use the TAR.

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

Fold on Dotted Lines and Tape

TAPE TAPE

FOLD FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 8241 MINNEAPOLIS, MINN.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
Publications and Graphics Division
P. O. Box 4380-P
Anaheim, California 92803

CUT ALONG LINE

FOLD FOLD

CORPORATE HEADQUARTERS, P.O. BOX 0, MINNEAPOLIS, MINN 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD

LITHO IN U S.A

CONTROL DATA CORPORATION