60471160

CONTROL DATA CORPORATION

COMMUNICATIONS CONTROL INTERCOM VERSION 3 SYSTEM PROGRAMMERS REFERENCE MANUAL

CDC® COMPUTER SYSTEMS
255X HOST COMMUNICATIONS PROCESSOR
255X NETWORK PROCESSOR UNIT
CDC® HOST NETWORK OPERATING SYSTEMS
NOS/BE 1

REVISION RECORD

REVISION	DESCRIPTION
A 12/31/79	Initial Release

Publication No.
60471160

REVISION LETTERS I, O, Q AND X ARE NOT USED

©1979
by Control Data Corporation

Printed in the United States of America

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
Cover	-
Title Page	-
ii thru xiii	A
1-1 thru 1-19	A
2-1 thru 2-12	A
3-1 thru 3-6	A
4-1 thru 4-30	A
5-1 thru 5-29	A
6-1 thru 6-36	A
7-1 thru 7-26	A
8-1 thru 8-13	A
9-1 thru 9-6	A
10-1 thru 10-17	A
11-1 thru 11-28	A
12-1 thru 12-16	A
Index-1 thru Index-13	A
A-1 thru A-11	A
B-1 thru B-5	A
C-1 thru C-21	A
D-1 thru D-14	A
E-1	A
F-1/F-2	A
G-1 thru G-35	A
H-1 thru H-83	A
I-1 thru I-13	A
Comment Sheet	-
Mailer	-
Back Cover	-

Page	Rev

Page	Rev

PREFACE

This manual describes those externals of the Communications Control Intercom (CCI), Version 3.0, necessary to aid a systems programmer in making minor modifications to standard CCI software. The manual also provides a sufficient basis to understand those standard programs which interface to any new terminal interface program which the user writes for a nonstandard terminal. CCI is used with the CONTROL DATA® 255x Series Network Processing Unit (NPU).

It is assumed that the reader is already familiar with CCI basic functions and the role of CCI in network processing. If the reader does not have this knowledge, he is referred to the CCI 3 reference manual which provides an introduction to CCI functions.

It is recommended that the user be experienced with the PASCAL programming language and the CYBER CROSS support system software. If the user plans to write his own terminal interface program, he 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 host or terminal user, or by the network software as output.

... — 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 underlined.

{ } — Braces enclose entities from which one must be chosen.

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

RELATED MANAULS

Additional information on both the hardware and software elements of the CONTROL DATA 255x Series Computer Systems and the CCI and related software can be found in the following documents:

Publication Title	Publication Number
Network Products UPDATE Reference Manual	60342500
Macro Assembler Reference Manual Mass Storage Operating System	60361900
INTERCOM Version 5 Reference Manual	60455010
Network Products Communciations Control Intercom (CCI) Version 3 Reference Manual	60471150
CYBER CROSS System Version 1 Link Editor and Library Maintenance Programs Reference Manual	60471200
Network Processor Unit Hardware Maintenance Manual	60472000
State Programming Language Reference Manual	60472200
NOS/BE Version 1 Operator's Guide	60493900
NOS/BE Version 1 Installation Handbook	60494300
CYBER CROSS System Version 1 PASCAL Reference Manual	96836100
CYBER CROSS System Version 1 Micro Assembler Reference Manual	96836400
CYBER CROSS System Version 1 Macro Assembler Reference Manual	96836500

These publications can be ordered from Control Data Corporation, Literature and Distribution Services, 308 North Dale Street, St. Paul, Minnesota 55103.

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

1. CCI OVERVIEW 1-1

CCI Design 1-3
Priority Processing at the Interfaces 1-3
OPS-Level Processing 1-4
Downline Message Processing 1-5
Upline Message Processing 1-5
CCI Features 1-8
CCI Modular Structure 1-9
CCI Programming Methods 1-9
Block Protocol 1-9
Block Routing 1-13
Point of Interface (POI) Programs 1-13
Direct and Worklist Calls 1-13
Direct Calls on Firmware Level 1-15
Special Call to Multiplex Subsystem 1-15
Special Call to Firmware Interface 1-15
Communication Using PASCAL Globals (Tables) 1-16
Line Interface Handling 1-16
CCI Programming Languages 1-18

2. INITIALIZING AND CONFIGURING THE NPU 2-1

Initializing the NPU 2-1
Phase I Initialization 2-1
Phase II Initialization 2-2
PINIT 2-2
PIPROTECT 2-2
PIBUF1 2-2
PIWLINIT 2-3
PIINIT 2-3
PIAPPS 2-3
PIMLIA 2-3
PILININIT 2-3
PIBUF2 2-3
Load and Dump NPU 2-4
Configuring the NPU 2-4
Configuring NPU 2-5
Line Configuration 2-5
Configure Line SM 2-6
Configure Line Deletion 2-10
Terminal (TCB) Configuration 2-10
Configure Terminal SM 2-11
TCB Reconfiguration 2-12
TCB Deletion 2-12

3. FAILURE, RECOVERY, AND DIAGNOSTICS 3-1

Host Failure 3-1
NPU Failure 3-1
NPU Recovery 3-2
Halt Codes and Dump Interpretation 3-2
Line Failure 3-2
Line Recovery 3-2
Terminal Failure 3-3
Terminal Recovery 3-3
In-line Diagnostic Aids 3-3
CE Error Messages 3-4
Statistics Messages 3-5

4. BASE SYSTEM SOFTWARE 4-1

System Monitor 4-1
Buffer Handling 4-2
Obtaining a Single Buffer 4-6
Releasing a Buffer 4-7
Releasing a Single Buffer 4-7
Releasing Several Buffers 7-7
Testing Buffer Availability 4-7
Buffer Copying 4-7
Other Buffer Handling Routines 4-8
Timing Services 4-8
Direct Calls 4-9
Worklist Services 4-10
Making a Worklist Entry 4-12
Extracting a Worklist Entry 4-13
Basic Interrupt Processing 4-13
Macrointerrupts 4-13
Interrupt Priority 4-14
PBSMASK - Set Interrupt Mask 4-14
PBAMASK - And Interrupt Mask (and PBLMASK) 4-14
PBOMASK - Or Interrupt Mask 4-14
User Interface 4-15
Microinterrupts 4-16
PASCAL Globals 4-17

Standard Subroutines 4-17
Calling Macroassembly Language Programs from PASCAL Programs 4-17
Defeating Type-Checking in PASCAL Procedure Calls 4-19
Handling Routines 4-19
PBFMAD - Converts from ASCII Decimal to Binary 4-20
PBFMAH - Converts from ASCII Hexadecimal to Binary 4-20
PBMAX - Funds the Larger Maximum of Two Numbers 4-20
PBMEMBER - Test ASCII Set Membership 4-20
PBMIN - Funds the Smaller Minimum of Two Numbers 4-21
PTOAD - Converts Binary to ASCII Decimal 4-21
PBTOAH - Converts Binary to ASCII Hexadecimal 4-22
Maintaining Paging Registers 4-22
PBSTPMODE - Sets Paging Mode 4-22
PBPSWITCH - Performs Page Switching 4-22
PBRDPGE - Reads Dynamic Page Register 4-23
PBPUTPAGE - Write Specified Page Register 4-23
PBGETPAGE - Reads Specified Page Register 4-23
PB18ADD - 18-Bit Addresses 4-23
PB18BITS - 18-Bit Address Functions 4-24
PB18COMP - Compares Two 18-Bit Addresses 4-24
Block Functions 4-24
PBCLR - Clears a Block of Main Memory 4-24
PBCOMP - Compares Two Equal Length Blocks 4-24
Set/Clear Protect Bits 4-25
PBSETPROT - Set Protect Bit 4-25
PBCLRPOT - Clear Protect Bit 4-25
Miscellaneous Subroutines 4-25
PBFILE1 - Load/Display File 1 4-25
PBHALT - Stops the NPU 4-26
PBILL - Illegal Calls 4-26
PBLOAD - Load a User-Defined Message 4-26
Program Execution Timers 4-27
Console Support 4-27
General Peripheral Processing 4-27
Console Support Services 4-28
Console Worklist Entry 4-29
Console Control Messages 4-29

5. MULTIPLEX SUBSYSTEM 5-1

Hardware Components 5-3
Multiplex Loop Interface Adapter 5-3
Loop Multiplexers 5-3
Communications Line Adapters (CLA) 5-3
System and User Interfaces 5-4
System Interfaces 5-4
Multiplex Level 1 (Firmware) 5-4
Multiplex Level 2 (PMWOLP) 5-7
Multiplex Subsystem Firmware Worklist Entries 5-8
Command Driver Worklist Entries 5-8
OPS Level 5-8
User Interfaces 5-9
Command Driver Interface 5-9
Clear Line Command 5-11
Initialize Line Command 5-11
Control Command 5-11
Enable Line Command (KLENBL) 5-12
Input Command (NKINPT) 5-15
Output Command (NKDOUT) 5-15
Input After Output (NKINOUT) 5-18
Terminate Input Command (NKENDIN) 5-19
Terminate Output Command (NKENDOUT) 5-19
Disable Line Command (NKDISL) 5-20
Common Multiplex Subroutines for TIPs 5-21
PMWOLP, Multiplex Worklist Processor 5-21
PTCLAS, CLA Status Analyzer 5-22
CLA Status Overflow Handling 5-24
Modem Response Timeout Handling 5-25
PLINIT, Line Initializer 5-26
PMT1SEC, Output Data Demand Timing Handler 5-29

6. NETWORK COMMUNICATIONS SOFTWARE 6-1

Block Protocol 6-1
Address 6-2
Node 6-2
Connection Number 6-6
BSN/Block Type 6-6
Block Serial Number (BSN) 6-6
Block Types 6-7
BLK (Block) Block 6-7
MSG (Message) Block 6-7
BACK (Block Acknowledgment) Block 6-7
CMD (Command) Block 6-7
Service Channel 6-8
Data Stream Control 6-8
Data Formats 6-8
Interactive Format Data 6-10
Batch Format Data 6-10
Nontransparent Data 6-11
Transparent Data 6-12
Routing 6-12
Directories 6-13
Destinations Node Directory 6-13
Source Node Directory 6-13
Connection Directory 6-13
Routing Process 6-13
Alternating Directories 6-16
Service Messages (SM) 6-16
Internal SM Processing 6-17
Validating and Timing Out SMs 6-17
Generating and Dispatching Service Messages 6-18
Configuring/Enabling/Disabling/Deleting Control Blocks 6-18
Generating and Sending Status SMs 6-19
Line Status Request SM 6-19
Line Count Request SM 6-20
Terminal Status Request SM 6-20
Generating and Sending Statistics SMs 6-20
CE Error Messages 6-21
Common TIP Subroutines 6-21
Point of Interface Routines (POI) 6-21
PBPIPOI, Post Input POI 6-22
PBIOPOI, Internal Output POI 6-22
PBPROPOI - Preoutput POI 6-30
PBPOPOI - Post Output POI 6-30
Standard TIP Subroutines 6-30
Output Queuing (PBQ1BLK and PBQBLKS) 6-30
Removing a Message Segment From Queue PBGT1SET 6-32
Saving and Restoring Registers 6-32
PBBEXIT - Save R1 and R2 6-32
PBAEXIT - Restore R1 and R2 6-33
Interface to Text Processing Firmware, PTTPINF 6-33
Finding Number of Characters to be Processed, PTCTCHR 6-33
Saving and Restoring LCBs, PTSVxLCB, and PTRTxLCB 6-33
Common Return Control Routine, PTRETOPS 6-34
Common TIP Regulation, PTREGL 6-34
Set Logical Link Regulation, PNLLREG 6-34
Set Accept Input/Accept Output Flags, PTINIT 6-35
Discards Non-routable Blocks, PBLOST 6-35
Upline Abort, PBUPABRT 6-35
Downline Abort, PBDNABRT 6-35
Send CMD Block to Host, PTCOMMAND 6-35
Upline PRU Block Routing, PBRTEPRU 6-35
PRU Block Routing, PBRTEIA 6-35
Check to Find if Block is to be Sent, PBBCHCHK 6-35
Generate Banner and Lace Records, PTBANLACE 6-35

7. HOST INTERFACE PACKAGE (HIP) 7-1

Transaction Protocol 7-1
Transfer Functions 7-1
Directives Used 7-2
Transfer Initiation 7-2
Transfer Timing 7-7
Error Processing 7-7
Host/NPU Word Formats 7-7
Coupler Interface Hardware Programming 7-8
Coupler Register Use 7-8
Programming the Coupler By Use of Function Codes 7-10
Host Function Commands 7-10
NPU Function Commands 7-12
HIP Functions 7-12

Single Word Transfers (Control) 7-12
Load/Dump NPU 7-14
Multiple Character Data Transfer (Block Transfer) 7-16
Contention for Coupler Use 7-17
Regulation of Coupler Use 7-18
Host Failure and Recovery 7-18
Error Checking and Timeouts 7-19
Interface Protocol Sequences 7-20
Buffer Format 7-25
HIP States 7-25

8. BINARY SYNCHRONOUS COMMUNICATIONS (BSC) TIP 8-1

Operational Features 8-2
Remote Batch Facilities 8-2
EOR/EOI 8-2
Binary Codes 8-2
026/029 Codes 8-2
Transparent Data 8-3
Carriage Control 8-3
Interactive Carriage Control 8-3
Punch Files 8-3
Compression/Expansion 8-4
Terminal Features 8-5
Operational Characteristics 8-5
2780 Input Nontransparent Terminal Mode 8-5
2780 Input Transparent Terminal Mode 8-6
3780 Input Nontransparent Terminal Mode 8-7
3780 Input Transparent Terminal Mode 8-7
Input Transparent Data Mode, 2780 and 3780 8-7
2780 Output Nontransparent Transmission Mode 8-8
2780 Output Transparent Transmission Mode 8-10
3780 Output Nontransparent Transmission Mode 8-10
Direct Calls to the BSC TIP 8-11
Direct Calls from the BSC TIP 8-12
Error Processing 8-13
Autorecognition 8-13

9. ASYNCHRONOUS (TTY) TIP 9-1

Operating Modes 9-1
Interactive Mode 9-1
Tape Mode 9-2
Carriage Control for Output Messages 9-4
Direct Calls to TTY TIP 9-4
Direct Calls from the TTY TIP 9-5
Error Processing 9-6
Autorecognition 9-6

10. MODE 4 TIP 10-1

Hardware Considerations 10-1
TIP Functions 10-1
Terminal Interface 10-3
Terminal Addressing 10-3
Message Type Indicators 10-3
E Codes 10-3
Code Conversion 10-3
Host Interface 10-7
Interactive Interface 10-7
Cursor Positioning 10-7
Carriage Control 10-8
Upline Breaks 10-8
Contention Resolution 10-9
Card Reader Interface 10-9
Printer Interface 10-9
Error Handling 10-12
Short Term Error Processing 10-12
Long Term Recovery 10-13
Handling of Errors for CDC 711 Terminal 10-13
Duplicating of Write Data on CRT 10-13
Input Regulation 10-13
Autorecognition 10-14
Mode 4 Protocol Features Not Supported 10-16
Direct Calls to the Mode 4 TIP 10-16
Direct Calls from the Mode 4 TIP 10-16

11. HASP TIP 11-1

Hardware Considerations 11-1
Major TIP Functions 11-3
HASP Protocol 11-4
Terminal Operational Procedure 11-6
Multileaving Block Descriptions 11-6
Control Blocks 11-6
Acknowledgment Block (ACK) 11-6
Negative Acknowledge Block (NAK) 11-7
Enquiry Block (ENQ) 11-7

Idle Block (ACK0) 11-8
Control Bytes for Data Blocks 11-8
Block Control Byte (BCB) 11-8
Function Control Sequence (FCS) 11-10
Record Control Byte (RCB) 11-10
Subrecord Control Byte (SRCB) 11-11
String Control Byte (SCB) 11-12
Data Block Description 11-12
Operator Console Blocks 11-13
End-of-File Blocks (EOF) 11-13
FCS Change Blocks 11-13
User Interface 11-14
Workstation Startup and Termination 11-14
Workstation Initialization 11-14
Communication Line Initialization 11-14
Signon Block 11-15
Signoff Block 11-16
Host Interface 11-16
Configuration and Addressing 11-16
Console 11-17
Card Reader 11-18
Card Reader Non-transparent Data Mode 11-19
Card Reader Transparent Data Mode 11-20
Printer 11-20
Printer Nontransparent Data Mode 11-20
Printer Transparent Data Mode 11-21
Command Interface for the Printer 11-21
Punch 11-22
Error Conditions 11-22
CRC-16 Error (Cyclic Redundancy Checking) 11-23
Illegal Block Make-up Error 11-23
Unknown Response Error 11-23
Block Control Byte (BCB) Error 11-23
Regulation and Flow Control 11-25
Autorecognition 11-25
Direct Calls to the HASP TIP 11-26
Direct Calls from the HASP TIP 11-26
HASP Postprint 11-27

12. STATE PROGRAMS 12-1

Execution of State Programs 12-1
Classes 12-2
Components of a State Program 12-4
Functions 12-4
Input State Programs 12-5
Text Processing State Programs 12-6
Firmware Interface to the Output Data Processing 12-7
Modem State Programs 12-8
Firmware Interface to the Modem State Programs 12-9
Multiplex Level Status Handler (PTCLAS) Interface to the Modem State Programs 12-10
Input State Programs Interface to the Modem State Programs 12-10
Macroinstructions 12-10

APPENDIXES

A Glossary A-1
B CCI Mnemonics B-1
C Service and Command Message Summary C-1
D Block Protocol Summary D-1
E Sample Main Memory Map for NPU E-1
F CCI Naming Conventions F-1
G Standard TIP and SVM Trees G-1
H Principal Data Structure H-1
I On-line Debugging Aids I-1

INDEX

FIGURES

1-1 Role of NPU in a Network 1-2
1-2 Priority and Nonpriority Tasks in CCI 1-4
1-3 Downline Message Processing 1-6
1-4 Upline Message Processing 1-7
2-1 NPU Configuration Sequence 2-4
2-2 Line/Terminal Configuration Flowchart 2-7
3-1 Format of CE Error, and Statistics Messages 3-3
4-1 OPS Monitor Table Format 4-4
4-2 Buffer Formats and Stamping 4-5
4-3 Worklist Organization 4-11
5-1 Basic Elements of the Multiplex Subsystem 5-2
5-2 TIP and Multiplex Subsystem Worklist Communications 5-5
5-3 Command Packet General Format 5-10
5-4 Control Command Format 5-12
5-5 Enable Line Command Format 5-13
5-6 Input Command Format 5-16
5-7 Input After Output Command Format 5-18
5-8 Terminate Input Command Format 5-20
5-9 Terminate Output Command Format 5-21
5-10 PTLINIT Relationships With Major CCI Modules 5-27
6-1 Communications Paths for Block Flow Control 6-3
6-2 Data Block Header Formats 6-4
6-3 Use of Routing Directories 6-14
6-4 Simplified Routing Flow Chart, PBSWITCH 6-15
6-5 Important Common TIP Subroutines 6-23
6-6 Structure of a TCB Queue 6-31
7-1 Coupler I/O Transactions 7-3
7-2 I/O Transaction Contention at the Coupler 7-5
7-3 OPS and Interrupt Levels for the HIP 7-6
7-4 Coupler Registers 7-9
7-5 Host Interface Protocol Sequence, NPU Side 7-21
7-6 Host Interface Protocol Sequence, Host Side 7-23
7-7 Standard Data Block Format Used by the HIP 7-25
10-1 Mode 4 Protocol Message Formats 10-4
10-2 MTI Codes for Mode 4 10-5
11-1 Typical HASP Multileaving Data Transmission Block 11-9
11-2 EOF Block 11-13
11-3 FCS Change Block 11-14
11-4 Signon Block Format 11-15
11-5 Format of Block Control Byte (BCB) Error Block 11-24
12-1 Locating a State Process 12-3

TABLES

1-1 CCI Modules 1-10
1-2 Support Programs for TIPs 1-11
1-3 Principal Data Structures 1-17
3-1 Inline Diagnostic Service Messages 3-5
4-1 OPS Monitor Table 4-3
4-2 Interrupt State Definitions (PBINTRAPS) 4-15
4-3 Interrupt Assignments 4-16
4-4 Standard Subroutines 4-18
4-5 NPU Console Control Commands 4-29
5-1 Multiplex Level 2 Worklists 5-6

5-2 TIP/LIP OPS level Worklists 5-7
5-3 Optional Modem/Circuit Functions 5-14
5-4 PTCIAS Worklist Analysis and Action 5-24
5-5 PTLINIT State Transition Table 5-28
6-1 Block Types 6-6
6-2 Command Blocks Used on Nonzero Connections 6-9
7-1 Coupler Status Register Bit Assignment 7-11
7-2 Orderword Register Codes 7-13
7-3 NPU Status Word Codes 7-13
7-4 Address Register Code 7-14
7-5 PPU Function Commands 7-15
7-6 NPU Function Commands 7-16
7-7 HIP States and Transitions 7-27
8-1 Summary of Batch Carriage Control Symbols 8-4
8-2 Summary of Interactive Carriage Control Symbols 8-4
8-3 2780 Batch Carriage Control Action 8-9
8-4 3780 Batch Carriage Control Action 8-11
9-1 TIP State Transitions, Interactive Mode 9-2
9-2 TIP State Transitions, Tape Mode 9-3
9-3 Carriage Control for TTY Output Messages 9-5
10-1 Mode 4 Nomenclature 10-3
10-2 Mode 4 Terminal/Cluster Addresses 10-5
10-3 E-Codes 10-6
10-4 DBC Codes for Carriage Control 10-8
10-5 Break Codes 10-8
10-6 Card Reader Input Stopped CMD Blocks 10-10
10-7 Printer Carriage Control Codes 10-11
10-8 Printer Input Stopped CMD Blocks 10-11
11-1 HASP Workstation Features 11-2
11-2 HASP Protocol Mnemonic Definitions 11-5
11-3 HASP Significant EBCDIC Characters 11-7
11-4 HASP Device Type 11-17
11-5 Card Reader Stream Control CMD Blocks 11-18
11-6 Printer Data Stream Control CMD Blocks 11-20
11-7 HASP Printer Carriage Control Codes 11-21
11-8 Punch Data Stream Control CMD Block 11-22
12-1 State Program Macroinstructions 12-11

CCI OVERVIEW 1

This section describes Communications Control Intercom (CCI) on a conceptual level. The description gives the programmer an overview of how CCI functions in a Network Processor Unit (NPU). For a more complete description of how CCI functions in a network, refer to the CCI reference manual.

CCI 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 the program were 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
- Regulating the volume of message traffic handled

CCI 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 subsections for each major class of terminal
- A multiplex subsystem that provides the hardware and software interface between the NPU and the various types of terminals

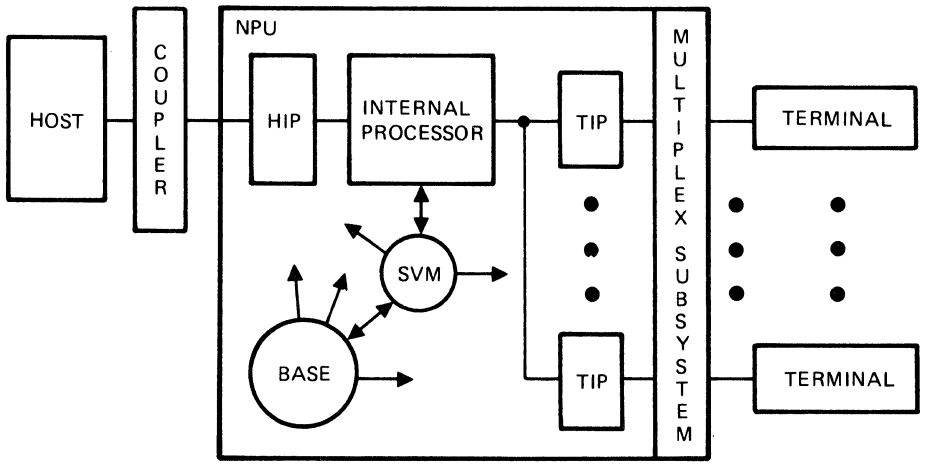


Figure 1-1. Role of NPU in a Network

CCI passes ASCII and display code messages to and from the host in format. CCI 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 host to terminal format and code. Upline messages are normally received from the terminals, converted to host format, and passed to the host.

NOTE

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

CCI DESIGN

CCI 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 (Host Interface Package, HIP, on the upline side; multiplex subsystem on the downline side), hardware and firmware do much of the preparation for a message or command transfer.

PRIORITY PROCESSING AT THE INTERFACES

At the interfaces, CCI 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 preferable 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 CCI 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, Terminal Interface Package (TIP), 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, CCI processes OPS-level tasks. There is an OPS Monitor that 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, 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, CCI 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.

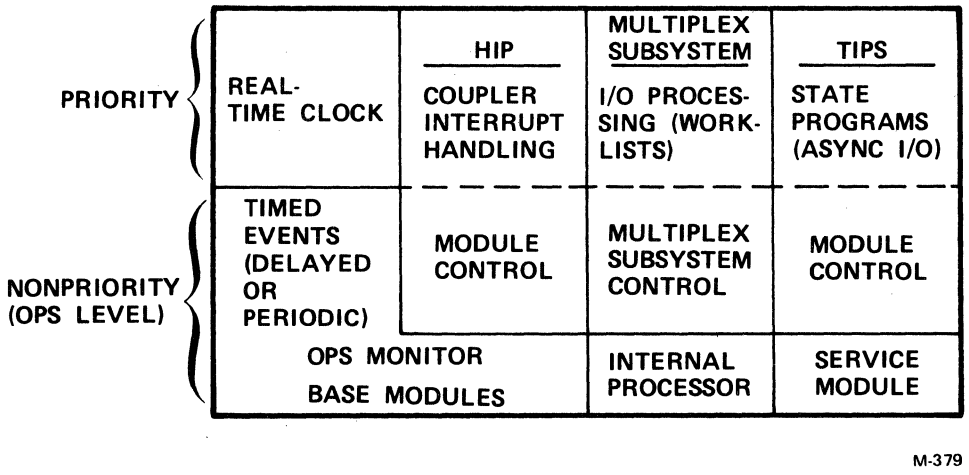


Figure 1-2. Priority and Nonpriority Tasks in CCI

DOWNLINE MESSAGE PROCESSING

Downline interactive messages originate in the host in blocks, each block usually being one output line. It is assumed that the interactive mode is conversational; that is, a line output is followed by a reply of one input line entered from the terminal's interactive device. The interactive device at the terminal is always ready for output unless the connection of the terminal is preempted by batch transfers or by an input interactive message. The output block is passed to the Host Interface Program (HIP) and is handled as a batch mode Physical Record Unit (PRU) block. See description, following.

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 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 host (through the TIP) must have requested the multiplex subsystem to prepare the line connecting the NPU to the terminal for a transmission.

At the multiplex subsystem, the output message block is multiplexed, along with other message blocks 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 communication line controller (consisting of the communications line adapter (CLA)) to the NPU. An output data processor in the multiplex subsystem handles this activity. The host is informed of message transmission progress twice: first, when the complete block is accepted by the NPU; and again after the block is 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 that 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 CCI/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 CCI. The message is converted from terminal code and format to host format. (Note that a transparent mode is also available for messages.) 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 an input Point of Interface (POI) program. That program transforms the batch blocks to PRU blocks and then passes the block to the HIP by way of an input Point of Interface (POI) program. The HIP, 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.

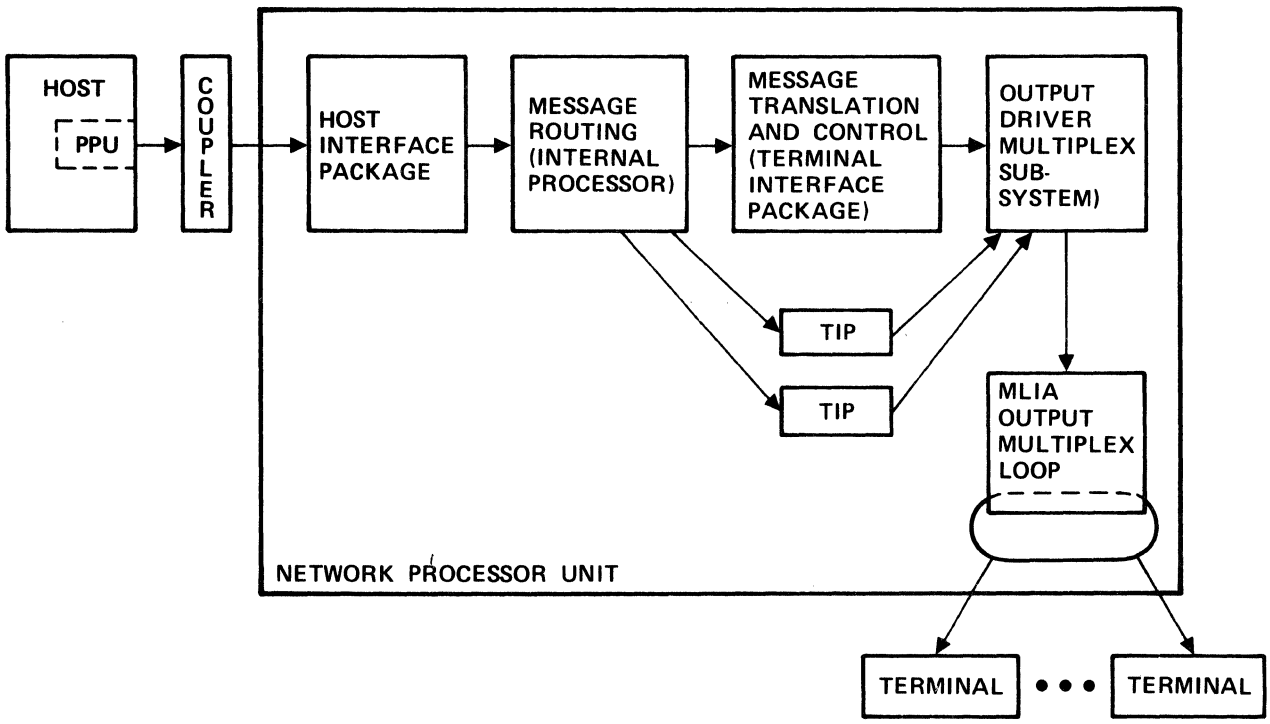


Figure 1-3. Downline Message Processing

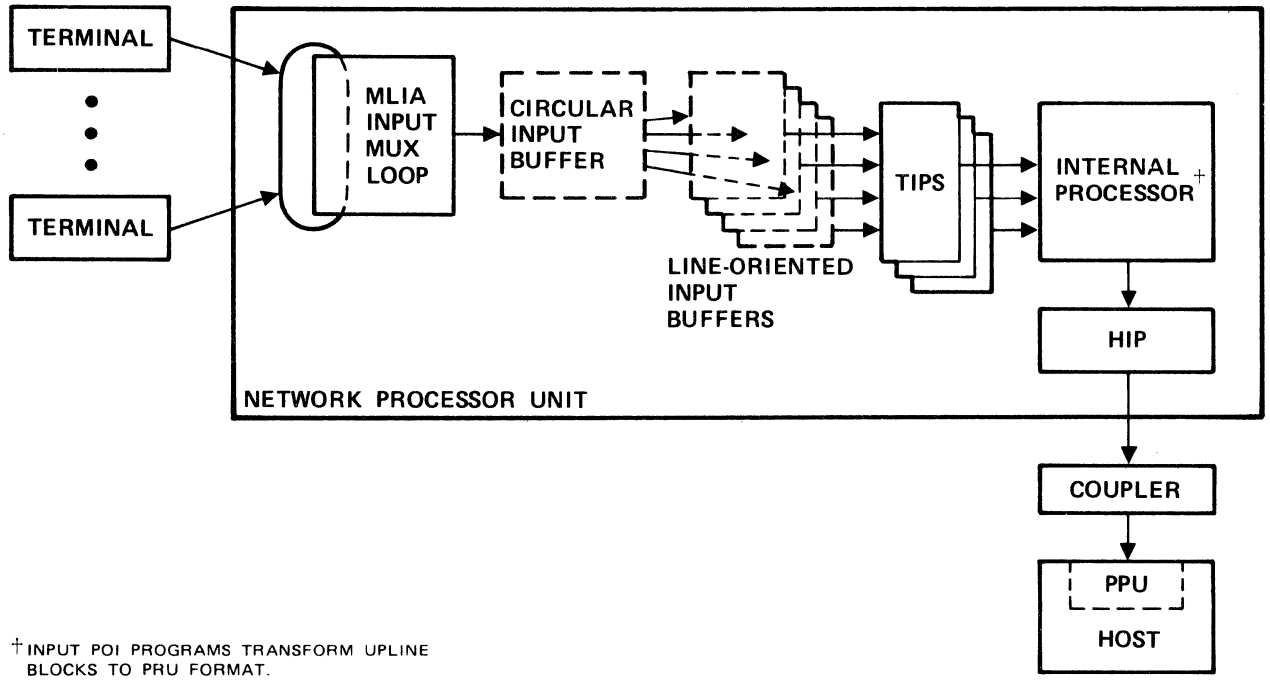


Figure 1-4. Upline Message Processing

CCI FEATURES

CCI provides several message processing features:

- TIPs and Point of Interface (POI) programs relieve host application programs of needing to handle terminal protocols. The TIPs and POIs convert messages to and from host code (display for batch devices) for the host.

- 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 CCI 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 demultiplex data and translate code and format. Every TIP has at least one firmware level program: an upline input state program. Most TIPs also have at least one downline firmware level program: the text processor for translating host code to terminal code and format.

The HIP does not use firmware programs directly. 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.

CCI MODULAR STRUCTURE

CCI can be considered as a group of generalized modules that provide services 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 CCI is shown in tables 1-1 and 1-2.

CCI is always resident in the NPU. It is downline loaded from the host. After loading is complete, additional communications between the host and CCI configure all the tables that hold line and terminal-oriented information. See appendix E for a sample CCI load map.

CCI PROGRAMMING METHODS

CCI 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 2048 bytes. The host is responsible for block size downline; the TIP input state programs and internal process 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, to start or stop a data stream for a specific terminal.

- BACK blocks carry communications protocol information, such as acknowledgment that is sent to the terminal that downline messages have been received from the host, and acknowledgment that upline messages have been received by the host.

Each block header has information relating to routing: source/destination nodes (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.

Data (BLK and MSG) blocks have an additional header, which contains control information and includes a data block clarifier (DBC).

Internal processing handles downline routing by use of the directories. Upline, the originating terminal is known. Using this information, the multiplex subsystem passes the block to the appropriate TIP. The input POI provides destination code information during upline routing, since this data is to be shipped to the host.

TABLE 1-1. CCI MODULES

Module	Major Function	Normal Calls
Terminal-Oriented		
Mode 4 TIP	Handles synchronous Mode 4A/4C terminals.	PT4...
TTY TIP	Handles asynchronous terminals using teletypewriter protocols.	PTTY...
HASP TIP	Handles synchronous HASP workstations.	HS... HASP...
BSC TIP	Handles the bisynchronous protocol used by IBM 2780/3780 terminals.	various
Host-Oriented		
Host Interface Program (HIP)	Handles block protocol between host and NPU; transfers use the host coupler.	PTHIP...

TABLE 1-1. CCI MODULES (Contd)

Module	Major Function	Normal Calls
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). This group of modules also handles upline formatting of blocks to PRU format.	PN... PT...

TABLE 1-2. SUPPORT PROGRAMS FOR TIPS

Programs	Location†	Comments
Host Interface Program (HIP)		
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	

TABLE 1-2. SUPPORT PROGRAMS FOR TIPS (Contd)

Programs	Location†	Comments
Host Interface Program (HIP)		
Internal processor maintenance	B	Building directories
Command driver	M	
Output data processor (ODP)	M	
Input data processor (IDP)	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 command block generator
Inline diagnostics	N	
NPU console services	B	
Initialization programs		Released when initialization is complete

† B = Base system
M = Multiplex subsystem
N = Network communications

All host/NPU transfers are controlled on the NPU side by the HIP. The HIP operates either by coupler interrupts or at OPS-level. The HIP 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 re-receive or retransmit portions of a block.

The service module (SVM) handles most commands between host and NPU other than those to start and stop a data stream. 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 acknowledgment service message to the host. In processing a service message, SVM can call on a TIP or on one or more other support routines.

Commands to start or stop message transmission on a line are sent directly between the host and the appropriate TIP. In this case, CN is not zero.

BLOCK ROUTING

Block switching downline is done by internal processing. Almost all blocks are passed to the receiving program (TIP, or SAVM) using a worklist entry. Invalid blocks are discarded. Upline blocks are routed by internal processing 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 must fulfill 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 checks the block serial number to assure that the block is in sequence. If it is a batch block, the TIP is called directly to convert the PRU block to a block in terminal code/format; then queues the block to the TIP or SVM for further processing.

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

- PBPIPOI - postinput POI. These POIs handle the upline block by building the block header. If it is a batch device block, the block is reformatted to PRU format. This is done by gathering the data buffers together to form a PRU size block (note: the UPs have already converted the data into display code). In all cases, the block is routed upline immediately, or is queued for upline routine.

- 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 PASCAL program. At the OPS level, direct calls are freely made between routines of the same kind (such as SVM routines or TIP routines within the same TIP). Calls are also made freely from the SVM, a TIP, 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 may accompany 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 parm1,...parmn

  MNCALL is at least the first six characters of the entry point name. Parm1...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
- 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 CCI to take appropriate action.

The monitor scans the list of OPS-level programs to find the next event (task) that 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. Information in this packet is 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 the text processing is completed.

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.

COMMUNICATION USING PASCAL GLOBALS (TABLES)

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. CCI 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). A 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 signon from the terminal. Low-speed lines can use autorecognition features (part of the TIP code) to establish line speed and code type.

- Polling synchronous Mode 4 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 and regulation level; one static block per link.	Routing modules, SVM
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, 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, HIP, multiplex subsystem
Command packet (NKINCOM)	Controls information for a multiplex subsystem I/O; builds the MLCB.	Sent from initializer, base or TIP to multiplex subsystem

TABLE 1-3. PRINCIPAL DATA STRUCTURES (Contd)

Structure	Major Functions	Principal Users
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
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.

CCI PROGRAMMING LANGUAGES

Commonly-used base programs, especially those with firmware portions, are written in macroassembly language for speed of execution. These programs should never be altered in the field. 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) that 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. These programs are called in one of three ways:

- A direct call from the internal processor to PTTPINF for a text processing program.
- An event-driven call, 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 also be made.

INITIALIZING AND CONFIGURING THE NPU 2

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

Before the CCI 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 CCI 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 CCI. 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 CCI 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 is available for a quick analysis of the dump. Refer to the CCI 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. CCI 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 0000_{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 the 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 CCI 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; a default size of three is used but no buffer is assigned.

PIINIT

PIINIT sets the NPU console to the write mode so the CCI 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 system 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 CCI.

LOAD AND DUMP NPU

A detailed description of loading and dumping an NPU, whether a local or remote unit, is given in the CCI 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.
- 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.

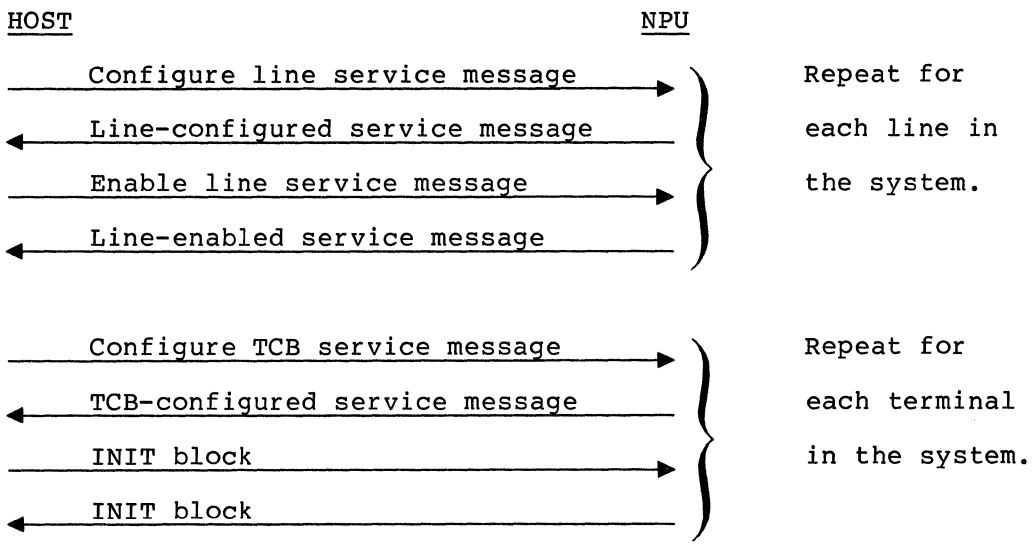


Figure 2-1. NPU Configuration Sequence

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.

CONFIGURING NPU

After the NPU is loaded, the host configures the unit by establishing all logical links and logical connections for that NPU. (Note: In CCI the links are preconfigured.) This is done in the following sequence:

- Lines are configured by building the line control blocks (LCB).
- Terminals are configured by building the terminal control blocks (TCB).

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

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 logical channel connecting the stations. Connection number zero is reserved as a permanent service channel for service messages.

The NPU sends an NPU-initialized service message to the host to notify it that the NPU has entered this active state.

LINE CONFIGURATION

After loading the NPU, the host sends service messages to the NPU to configure the lines between the NPU and the terminals. These configure line service messages are handled by the service module in the receiving NPU. The format of the service message is shown in appendix C.

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, CLA, modem, carrier circuit; answering and turnaround mode; and type of transmission, synchronous or asynchronous.
- Terminal type (TIP/sub-TIP required to process the terminal's data, device type, and terminal class).
- Data necessary to fill the selected fields of the line control block (LCB).

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

A simplified flowchart for line configuration is shown in figure 2-2. 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 of the terminal's associated LCB and TCB are configured. After configuration, the following events occur:

- The host can identify the terminal. The host can also find the proper regulation level to use.
- CCI can identify the protocol necessary for the data transfers and can assign a proper TIP to handle that protocol.
- The hardware in the CLA and modem are prepared for data transfers.

After a 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 service message response message. When the line is reported operational, TCBs are configured. The host starts the line configuration process whenever an NPU has been loaded and all links are configured, or when a network operator has entered a command that generates a specific type of supervisory message in the host.

Configure Line SM

For each line to be configured, the host sends a configure line service message 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. The service module returns a line configured response to the host. The host then sends an enable line service message to the NPU. The service module then attempts to enable the configured line. At the completion of the enable process, the line enabled response SM is returned.

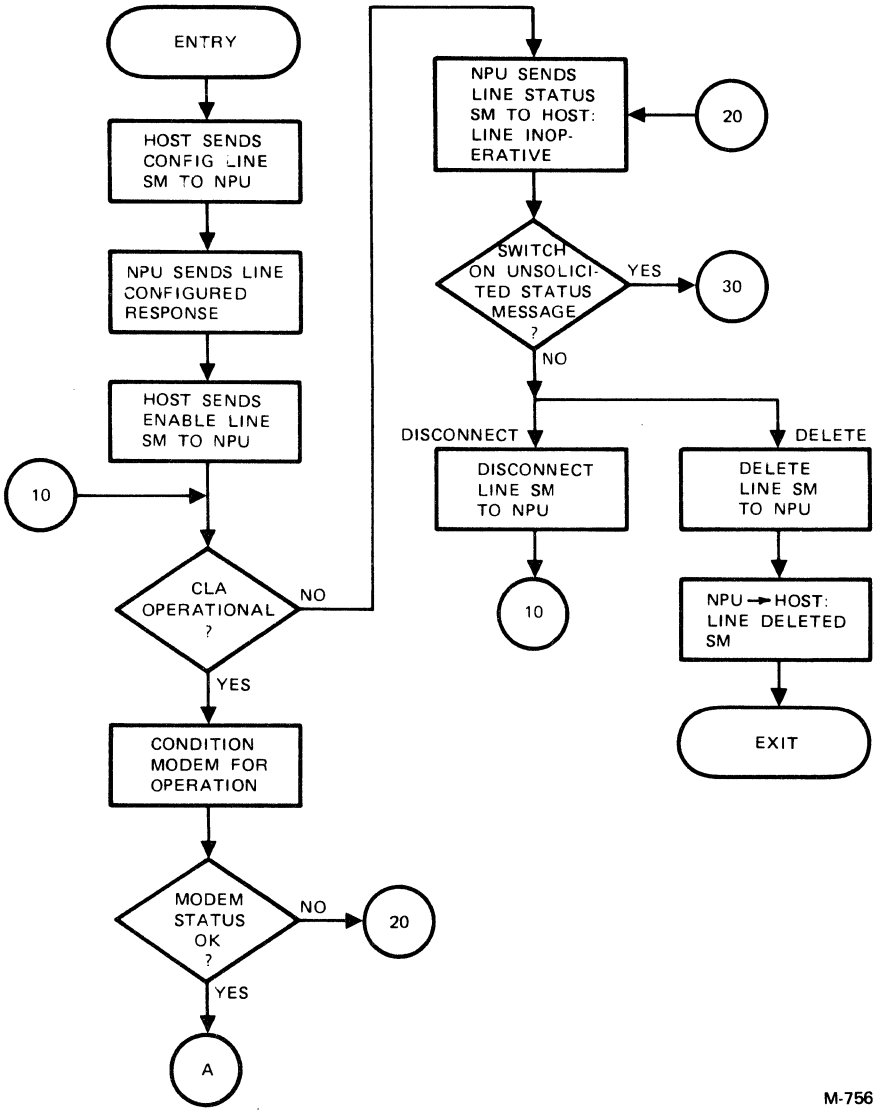


Figure 2-2. Line/Terminal Configuration Flowchart (Sheet 1 of 3)

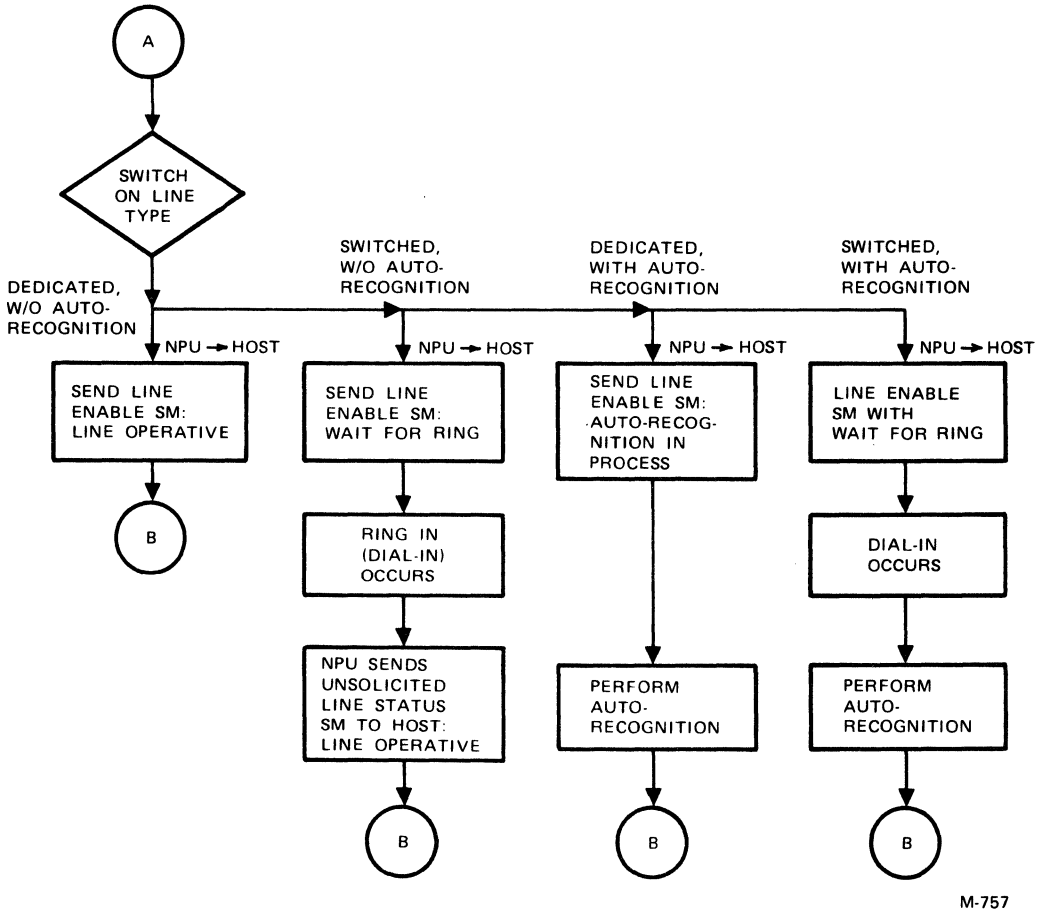


Figure 2-2. Line/Terminal Configuration Flowchart (Sheet 2 of 3)

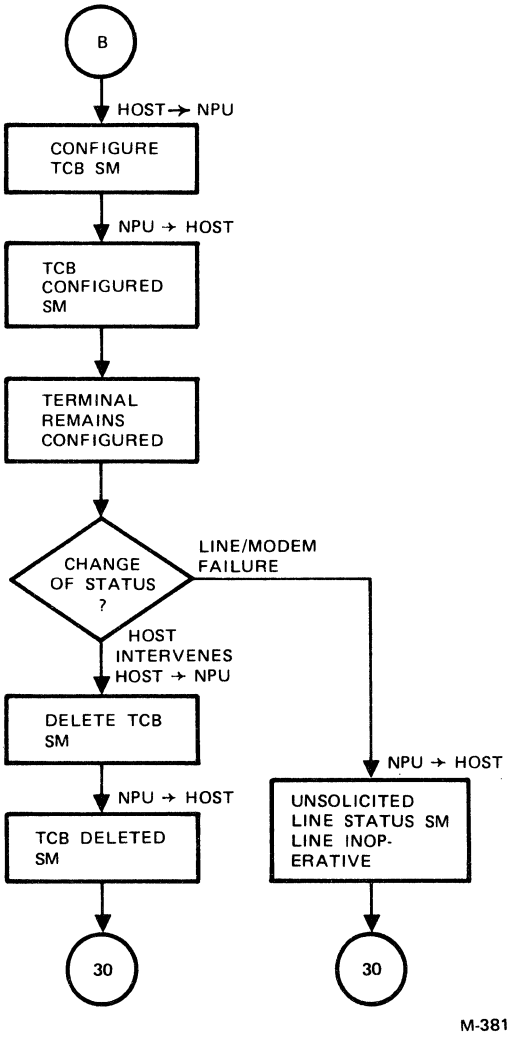


Figure 2-2. Line/Terminal Configuration Flowchart (Sheet 3 of 3)

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 signaled and the NPU returns a data-terminal-ready to answer the call. If, when ring indicator is signaled, the host or logical link is not available, the NPU ignores the dial-in.

Autorecognition/non-autorecognition is the same for switched lines as it is for dedicated lines.

CONFIGURED LINE DELETION

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

TERMINAL (TCB) CONFIGURATION

When the line is operational, the host can configure terminals for the line by issuing one or more configure terminal service messages. CCI responds to the configure terminal SM by generating the TCB. The amount of information in a TCB varies, depending on terminal type.

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

Terminals are identified in service messages by specifying the line, the hardware address, device type, and terminal class. 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
TTY	0	0
HASP	0	0-7
BSC	0	0-1

TA = stream ID of device: console = 1, card reader = 0-7, printers = 0-7, punches = 0-7.

Punch only. All other devices = 0.

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 TTY 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 can use the full range of stream number, but plotters must 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 normally established at the close of the intialization process.

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 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 the host as the result of a line operational SM received and processed by the host. As in the configure line service 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. Ranges for the FVs are given in appendix C.

A response SM is sent to the host 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 the host detects a need to establish or change a connection or modify other values in the TCB.

The format of the reconfigure terminal SM is similar to that 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 response formats are the same as those for the configure terminal SM.

The reconfigure terminal SM provides a general mechanism for the host 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, the host sends a delete terminal SM to delete the TCB and to clean up all table and data space associated with the TCB. CCI removes the connection from the logical connection directory. The service module responds to the host with a TCB-deleted SM. The host is responsible for correctly deleting both ends of a connection.

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

FAILURE, RECOVERY, AND DIAGNOSTICS 3

Failure and recovery of CCI 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.
- 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 CCI, 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 stoppage, and off-line dumps.
- Optional on-line diagnostics - These tests allow checking of circuits to terminals, and 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.) The NPU also sends an informative service message to all connected interactive terminals.

NPU FAILURE

The peripheral processor unit (PPU) of the host has a 10-second deadman timer. If the PPU connected to the 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.

NPU RECOVERY

The host dumps and reloads an NPU after receiving a request for load. The stimulus for this reload comes from the host PPU driver. The reasons for requesting a load are as follows:

- Software failure caused PPU hardware deadman timer to expire.
- Hardware failure caused deadman timer in the PPU to expire.
- Operator initiated a software halt, forcing reloading.
- Operator pressed MASTER CLEAR on the NPU maintenance panel, causing a reload request.

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, the host sets up all logical links for that NPU that the present state of the network allows. The methods of loading and initializing NPUs are described in the CCI 3 reference manual. The host examines its configuration tables for elements that have been affected by the change in status. Then the host configures and enables lines that are supported by the NPU. For any line reported as operational, an examination of the configuration tables reveals those terminals that can be connected. For each such terminal, both the terminal and the host support tables are configured and thereby connected.

HALT CODES AND DUMP INTERPRETATION

Unless the NPU stoppage resulted from a 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 so that it is), a dump is sent to the host. The CCI 3 reference manual describes the mechanics of transmitting the dump. Appendix B of that manual (Diagnostics) describes the format of the dump and its interpretation with or without the use of halt codes.

LINE FAILURE

Line failure is detected by abnormal modem status or by line protocol failure. The change of status is reported to the host using an unsolicited line status reply SM. The host deletes all TCBs supported by the line using the disconnect line service message.

LINE RECOVERY

A line cannot recover from a failure spontaneously. The host must first process the unsolicited status reply (line inoperative) SM by deleting the supported TCBs. The host then disables and reenables the line, using the appropriate service message. At this time, the TIP/HIP commences to check for a change in status. When the line status changes to operational, this is reported to the host with an unsolicited line status reply SM (line operational). When the host receives a message indicating that line status has changed to operational, it 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 the host for network management purposes. An unsolicited terminal status reply (terminal inoperative) SM reports the failure.

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 the host with an unsolicited terminal status reply SM. The host replies with start message to the TIP, allowing transmission for the terminal to begin.

IN-LINE DIAGNOSTIC AIDS

Three types of in-line diagnostic aids are provided with CCI:

- 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 optional 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, a halt message is sent to the NPU console. This 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 dump should be examined. The dump will disclose 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 CCI 3 reference manual. Dump interpretation is described in appendix B. Note that the halt message is delivered using PBQUICKIO; the message does not use an 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.

Byte	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, with PFC = 0A
- 01 - Alarm message
- 00 - NPU statistics
- 01 - Trunk/line statistics
- 02 - Terminal statistics

(with PFC = 07)

DATA - (see table 3-1)

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

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 swapping the NPU or host with error messages when an oscillatory condition arises, an error counter is incremented for 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 CCI 3 reference manual.

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
NPU Statistics	07	00	Error words 1 thru 11† ; 2 bytes/word
Line Statistics	07	01	First: P - port Second: 00 Third: 00 Subsequent: explanation words 1 thru 4; 2 bytes/word†
Terminal Statistics	07	02	First: P - port Second: 00 Third: CA - cluster address Fourth: TA - terminal address Fifth: DT - device type (See appendix C for values.) Sixth: CN - connection number Subsequent: explanation words 1-3, 2 bytes/word

†Refer to appendix B of the CCI 3 reference manual for details.

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 (coupler TCB for NPU, LCB for lines, TCBs terminals) by a call to PNSGATH. The call comes from either a TIP (usually via the post-input or post-output POI) or the HIP.

One stimulus for a statistics report is a request from the timer module PBTIMAL. The period for this timeout is a system build time parameter. PNPSTAT handles the periodic request. Two other stimuli cause PNDSTAT to generate the message: one stimulus arises when any one of the counters that keeps the statistics overflows. 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 PNPSTAT 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 currect 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; then 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. Note also that the HIP (section 7) can be considered as a support program for the TIPs.

The functional grouping of support tasks is as follows:

- Base system - operating system functions (program execution, buffer allocation, interrupt handling), timing support, and 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 (discussed in the CCI 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) which defines the number of worklist entries to be processed by the OPS level program before the pointer is moved to the next program (usual number is 1). If multiple executions are allowed, pointer does not advance until the N allowable worklist entries have been cleared from the worklist, or until there are no more worklist entries in the module's queue. 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.

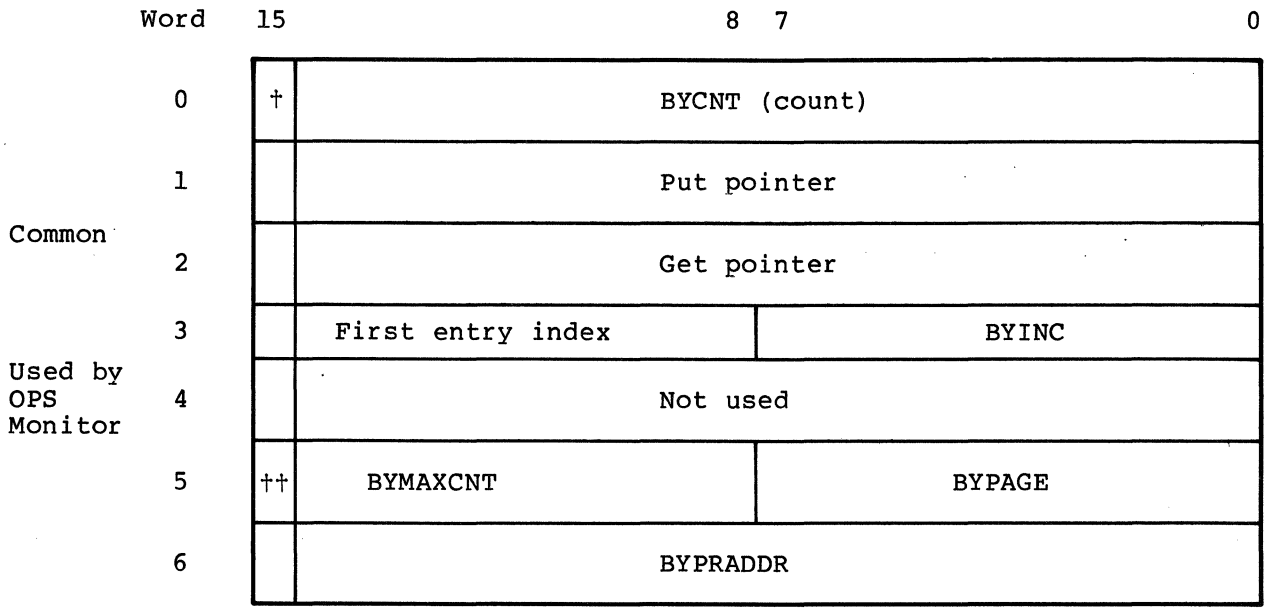
Standard buffers are also assigned for the following:

- Data buffer for special TIP application
- Console format
- Integer overlay
- Buffer chaining overlay
- Terminal control blocks (TCBs)
- Physical I/O request packets
- Active TTY LCB list
- Type 1 table entries
- Type 4 table entries
- Timeout buffers
- Diagnostic control block (DCB)
- Mux line control block (MLCB) and text processing control block (TPCB)

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 the monitor; reserved for generating the 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		13	Reserved			
	B0LIWL	14	Line initializer	1	PTLINIT	3
	B0DGWL	15	(On-line diagnostics)	0	------	-
	B0COWL	16	HIP	1	PTHIPOPS	3
	B0M4WL	18	Mode 4 TIP	1	PTMD4TIP	3
	B0TTYWL	19	TTY TIP (Mode 3)	1	PTTYTIP	3
	B0HASP	20	HASP TIP	1	PTHSOPSTIP	3
Monitor	B027WL	21	2780/3780 TIP	1	PTIP780	3
Pointer	B0HHWL	22	Reserved	0	------	-
recycles...	B0DUMMY	23	Dummy for console; recycles to entry 8	0	------	-

†Number of multiple executions allowed for this program.



See appendix H for the format of entries in a worklist.

†Multi-WLCB flag

††BYWLREQ, worklist required flag

BYCNT - number of WLCBs to process in one pass

BYCNT - number of WLCBs to process in one pass

BYWLINDEX - WLCB index

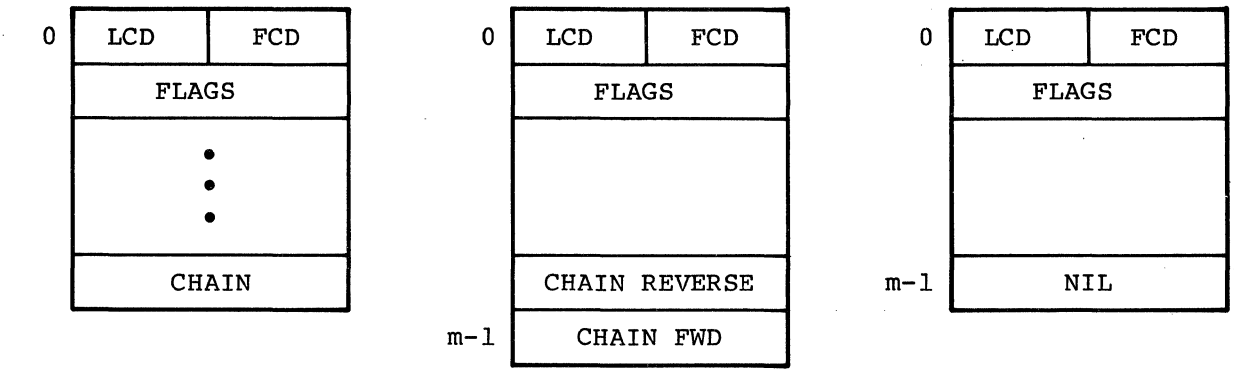
BYMAXCNT - number of WLCBs to process in one pass

BYPAGE - program page address

PYPRADDR - program address

Figure 4-1. OPS Monitor Table Format

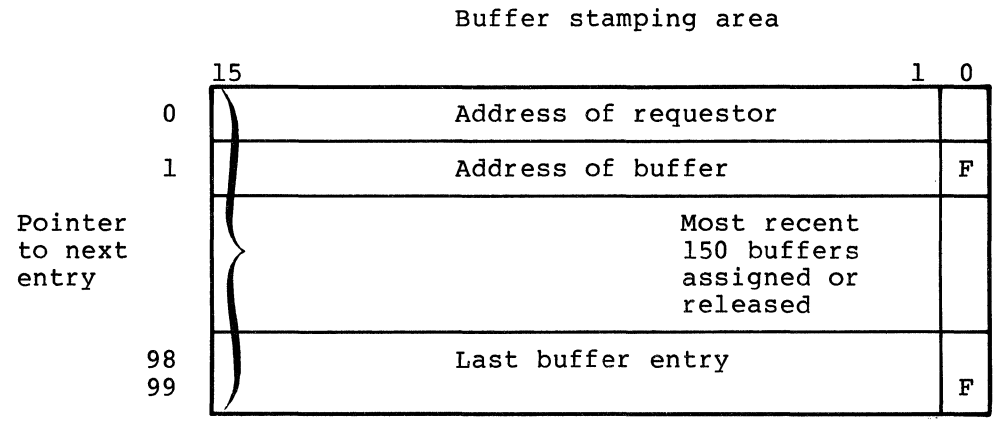
Figure 4-2 indicates the types of buffers assigned. Each buffer type has its own field definitions. The figure also shows the stamping techniques.



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 for chaining reserved.



F status flag
0 = put
1 = get

A circular buffer, two words/entry

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 system halts with a diagnostic halt. 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 2-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 B0BUFPTR 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:

```
PBRELlBF (parml, parm2)
```

Parml is a pointer to any address within any word of the buffer to be released, and parml is the address of the pointer to the buffer control block. Parml is a PASCAL VAR parameter that is altered by the procedure so that, upon completion, parml 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 PBRELCHN (parml, parm2) and PBRELZRO (parml, parm2).

In both cases, parml 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 parml for PBRELZRO is zero, no action is taken.

TESTING BUFFER AVAILABILITY

The calling sequence to test buffer availability is:

```
PBBFAVAIL (parml, 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 use 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 provide 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 = (pgm1...pgmn);
var   table:  array  pgms  of integer:
      index:  pgms;
addr  (  programl  , table  pgm1 );
.
.
.
addr  (  programn  , table  pgmn );
.
.
.
  set up index
PBCALL (table  index ); (call program, no parameters)
```

The PBCALL calling sequence is:

```
PBCALL (addr, parm1,...parmn)
```

addr is the address of the program to be called, and parm1 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 (parm1) switches control directly from one OPS-level program to another. Parm1 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) 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 can 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.

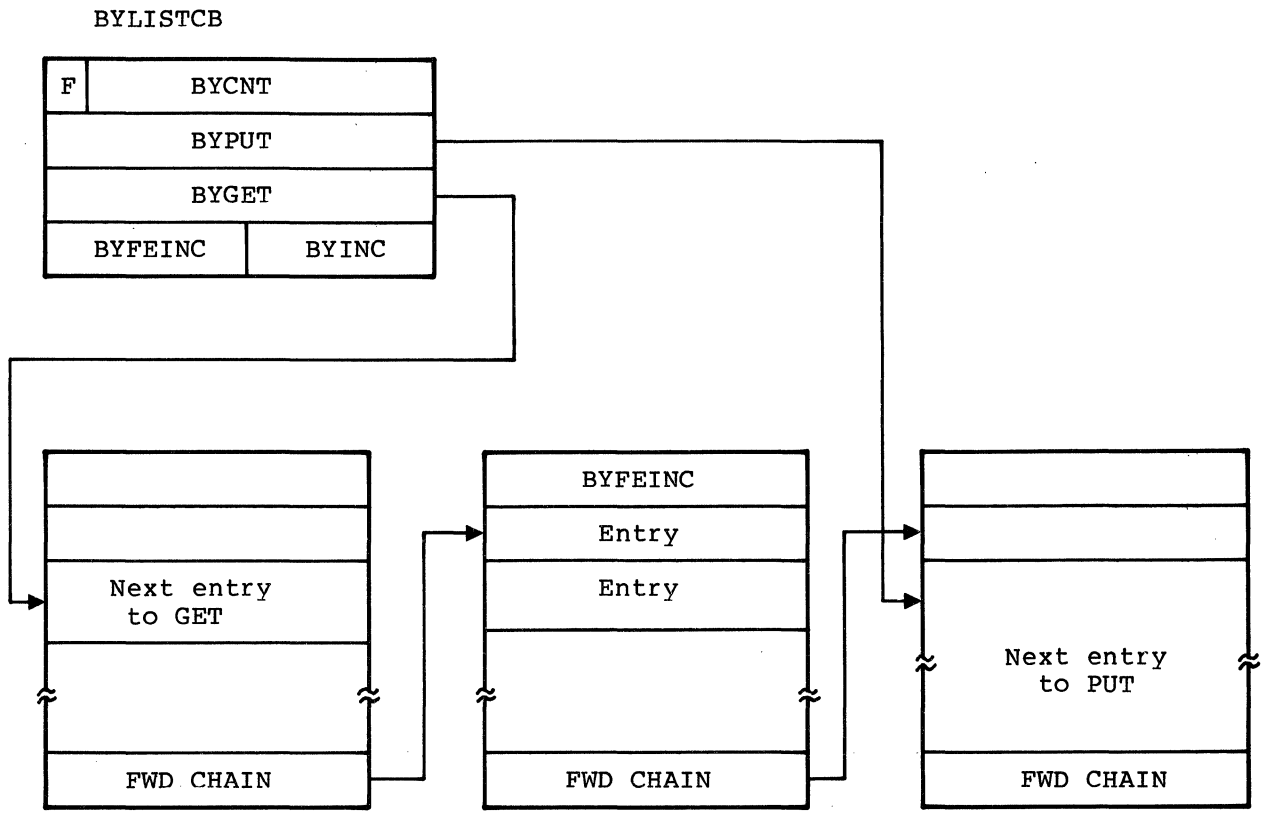
- There is no maximum on the number of entries in a list or on the number of lists serviced.

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 B0WKLSTS. 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 = BYCONTEND - A multiprocessor contention flag for 2552 NPUs
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 (for example, OPS, 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 (parm1, parm2)
```

Parm1 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 (parm1, parm2)
```

Parm1 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 control 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, and so forth. 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 timing services firmware processes the 3.3-millisecond clock interrupt, which is used as the time base for all timed NPU functions.

PASCAL GLOBALS

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

STANDARD SUBROUTINES

Standard subroutines are a miscellaneous group of support routines that 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	Clears block of main memory.	NI	PP	Yes
PBCLRPROT	Clears protect bit.	NI	MA	Yes
PBCOMP	Compares two blocks.	NI	MA	Yes
PBFILE1	Loads/displays file 1.	O	MA	Yes
PBFMAD	Converts from ASCII to binary.	R	PF	No
PBFMAH	Converts from ASCII to binary.	R	PF	No
PBGETPAGE	Reads page register from specified bank.	NI	MA	Yes
PBHALT	System halt.	NI	PP	Yes
PBILL	Illegal call; passes to TIP for CCI variants.	NI	PP	Yes
PBLOAD	Loads a canned message.	R	PP	Yes
PBMAX	Gets max of 2 numbers.	NI	PF	No
PBMEMBER	Tests ASCII set membership.	NI	PF	No
PBMIN	Gets 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
PBRDPGE	Reads dynamic page register.	NI	MA	Yes
PBSETPROT	Sets protect bit.	O	MA	Yes
PBSTPMODE	Sets page mode.	NI	MA	Yes
PBTOAD	Converts to ASCII decimal.	R	PP	No
PBTOAH	Converts 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
TOSTART	Starts program execution timer.	R	PP	No
TOSTOP	Stops program execution timer.	R	PP	No
TOTIME	Programs execution timer.	R	PP	No

†NI = Noninterruptable
O = OPS level only
R = reentrant

††PP = PASCAL 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 type-checking.

HANDLING ROUTINES

These 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 a 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 Maximum of Two Numbers

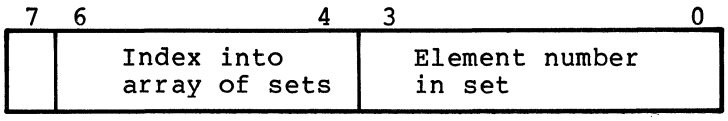
PBMAX is a function that returns the larger (maximum) of two given numbers. The calling sequence is:

```
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 1-word, 16-element sets by accessing an array of 1-word sets. A character is broken up for testing by the following format:



In an array of type JSACIISET, 128 bits are reserved (one for each possible ASCII character), where JSASCIISET = array (0..7) of 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:

```
var    JSHEXSET:    JSACIISET;
value    JSHEXSET =    (0, 0, 0, 3F16,
                                 digits 0-9
                       7E16, 0, 0, 0);
                       characters A-F
```

The calling sequence is:

```
PBMEMBER (parm1, parm2)
```

PARM1 is a value parameter of type B0OVERLAY 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 Minimum of Two Numbers

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

```
PBMIN Pparm1, 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. The 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. The 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. The 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. The 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

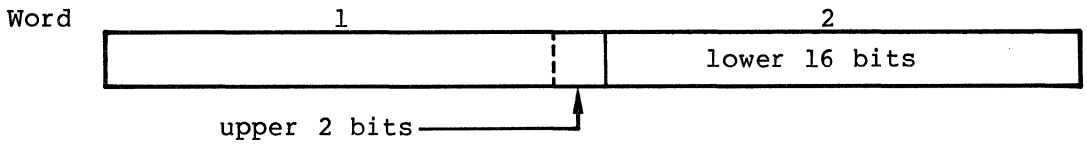
PBGETPAGE reads the contents of the specified page register and returns them to the user. The calling sequence is:

```
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 which is properly loaded by PB18BITS.

PB18BITS — 18-Bit Address Functions

PBl8BITS 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:

```
PBl8BITS (parml, parm2, parm3)
```

Parml 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

PBl8COMP makes a comparison between two 18-bit addresses. The calling sequence is:

```
PBl8COMP (parml, parm2, parm3)
```

Parml is the A address, and parm3 is the B address. Parm2 specifies the type of comparison: A COMP B, where COMP is one of $=$, $\neq$, $>$, $\geq$, $<$, or $\leq$. The output is a BOOLEAN function: true if A COMP Bl; 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. The calling sequence is:

```
PBCLR (parml, parm2)
```

Parml 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; 0, false) is returned to the caller. The calling sequence is:

```
PBCOMP (parml, 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. The 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. The calling sequence is:

```
PBCLRPOT (parm)
```

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

MISCELLANEOUS SUBROUTINES

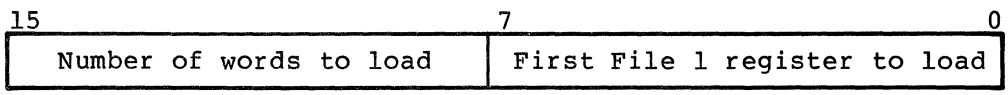
PBFILE1 – Load/Display File 1

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 firmware 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. The calling sequence is:

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

Parm1 is a value parameter, formatted as follows:

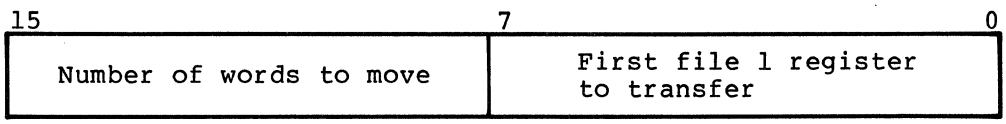


To load all 256 registers, set parm1 to 0. Parm2 is a value parameter specifying the address of the first memory location to transfer.

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

```
PBDF (parm1, parm2)
```

Parm1 is a value parameter formatted as follows:



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.

The 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 CCI system. The 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: PBIOSERV 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. Attempts are made 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 is 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 constitute 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 logical 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 canceled.
/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.

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.

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 multiplex 2 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.

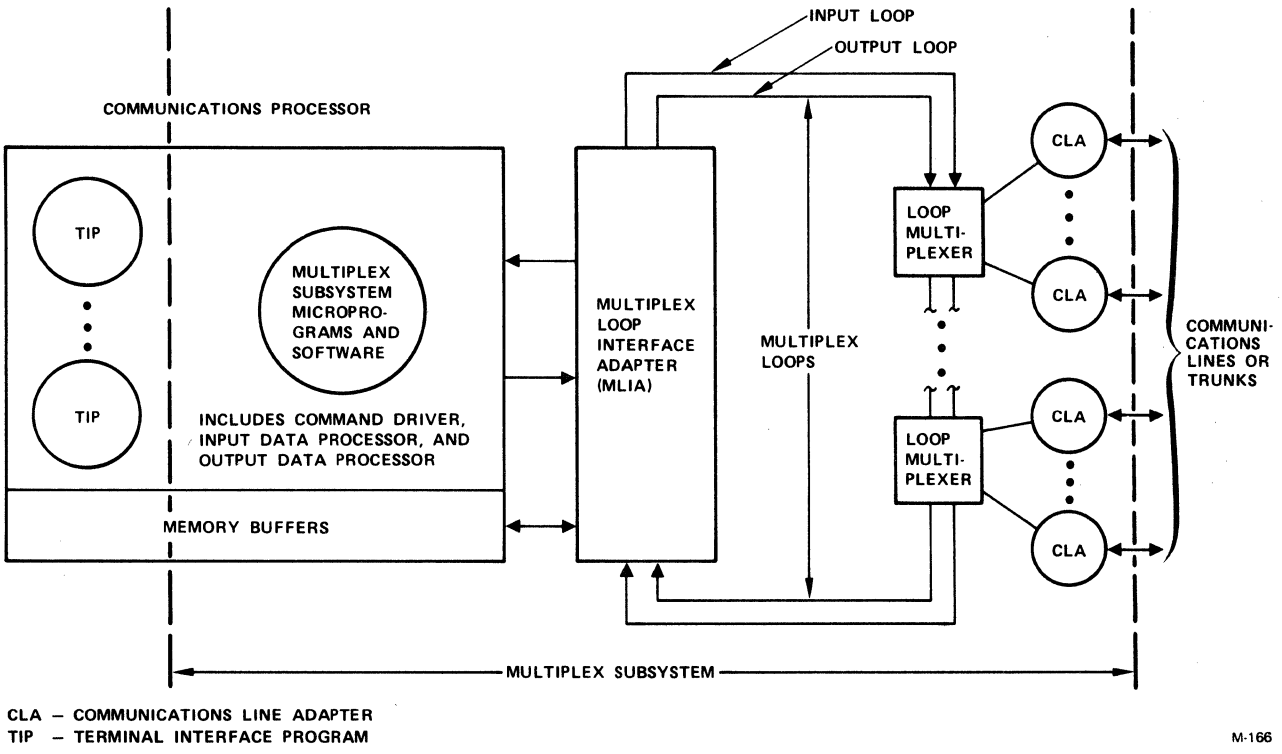


Figure 5-1. Basic Elements of the Multiplex Subsystem

HARDWARE COMPONENTS

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

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

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

SYSTEM INTERFACE

A TIP 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 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 contains 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 TIP 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.

Preliminary handling of CLA status is done by the modem state programs. The two lowest-numbered input states (which receive control from the modem state programs) are reserved to handle the following special status conditions:

- State 0 - is reserved for CLA status such as parity errors and data transfer overruns.
- State 1 - is reserved for data carrier detect (CDC) signal dropped.

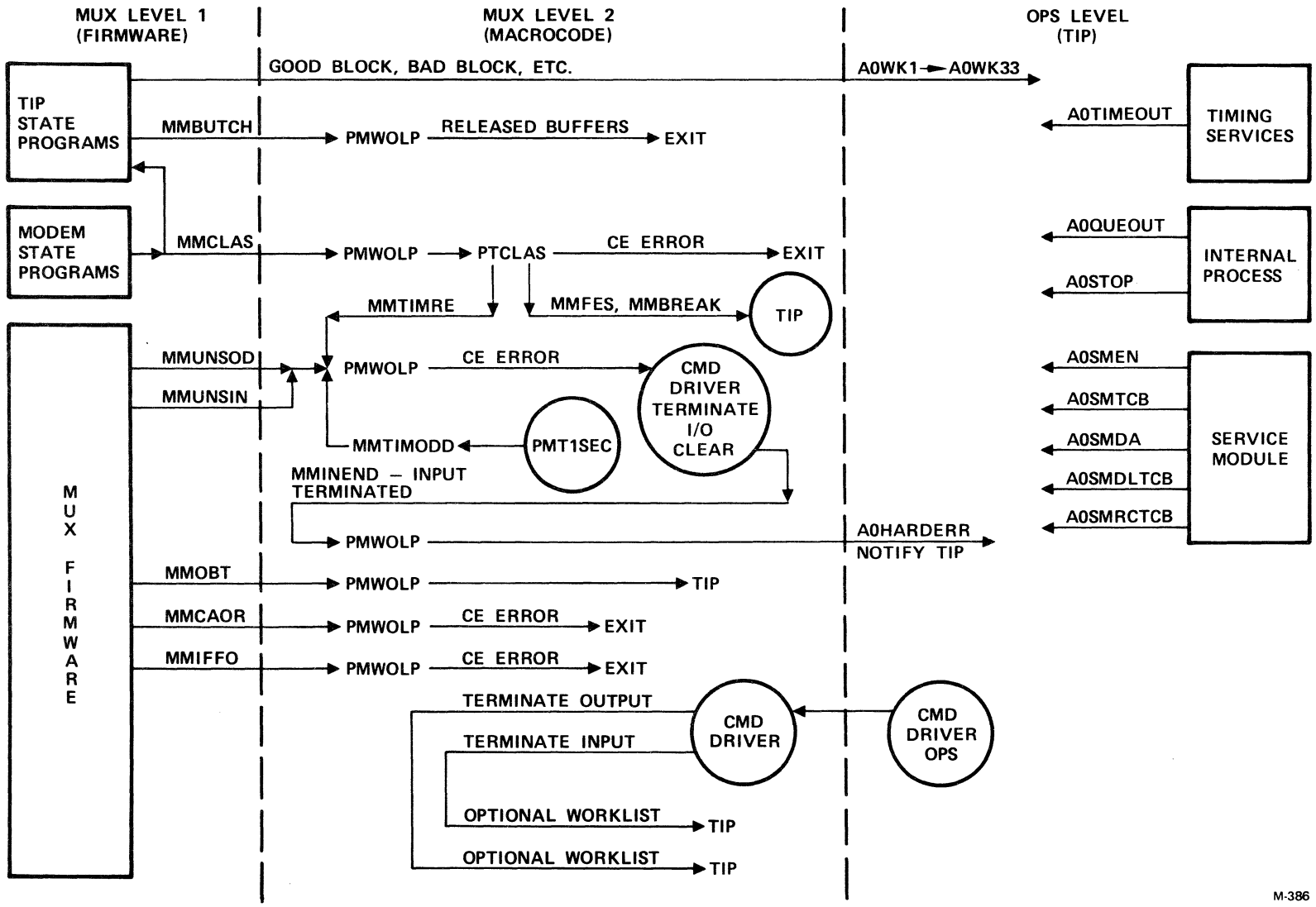


Figure 5-2. TIP and Multiplex Subsystem Worklist Communications

TABLE 5-1. MULTIPLEX LEVEL 2 WORKLISTS

Workcode	Workcode to TIP	Functions
MMCLAS	-	CLA status error, implies line error to TIP.
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	MMHARDER	Modem response timeout, implies hard line error to TIP.
MMOBT	MMOBT	Output block transmitted.
MMBUTCH	MMBUTCH	Multiplex subsystem buffer threshold reached. Buffers are released.
MMCHOUT	MMCHOUT	100-ms timeout.
MMCAOR	-	CLA address out of range - not seen by TIP.
MMIFFO	-	Illegal lineframe format - not seen by TIP.
NMINEND	A0HARDERR to OPS level	Input buffer terminated, response to PMWOLP command for hard errors.
MMFES	-	Framing error status, TIP should cause command driver to send delimiter to line (asynchronous lines).
MMBREAK	-	User break, TIP is called (asynchronous lines).

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	Reconfigure TCB command from service module.

Two additional input states are reserved for buffer handling conditions. These are called by the input data processor if one of the buffer thresholds is exceeded when the multiplex subsystem is trying to store another input character when this requires assigning a new buffer. (Note: the character is discarded.)

- State 2 - Number of input buffers being used by this TIP exceeds the allowable number (ABL threshold).
- State 3 - System buffer threshold reached.

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 (MMOBT, MMBUTCH, MMCHOUT, MMFGS, MMBREAK) plus any of its own that are generated in multiplex level 1. For synchronous TIPs, there is no processing required since the MMOBT entry is optional.

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:

- Worklists resulting from hard errors for unsolicited input or output, and timeouts for output data demand or modem response.
- Worklists to the system indicating 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.
- Worklists resulting from multiplex loop errors indicating 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 TIP. Two optional entries are generated: input terminated and output terminated.

OPS Level

The OPS level portion of the TIP handles all line or terminal servicing, 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 processing 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.

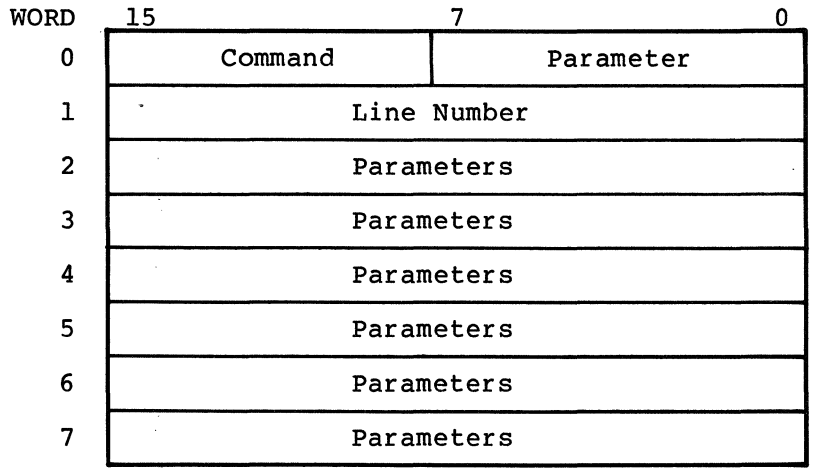


Figure 5-3. Command Packet General Format

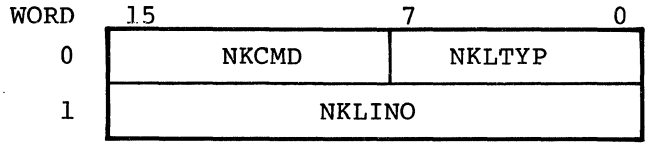
The following commands are available to the user for controlling the flow of data to and from the communications lines:

- 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, 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:



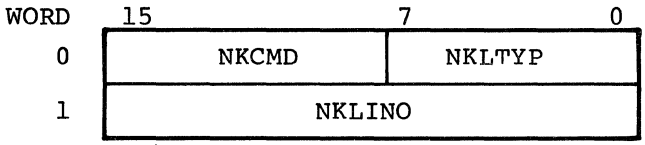
NKCMD - Command code (NKCLRL).

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:



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.

WORD	15 14		7 6	0
0	NKCMD		NKTCKY	
1	NKLINO			
2	F1	NKFUN1	F2	NKFUN2
3	F3	NKFUN3	F4	NKFUN4
4	F5	NKFUN5	NKZERO	

NKCMD - Command code (NKCONTROL).

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

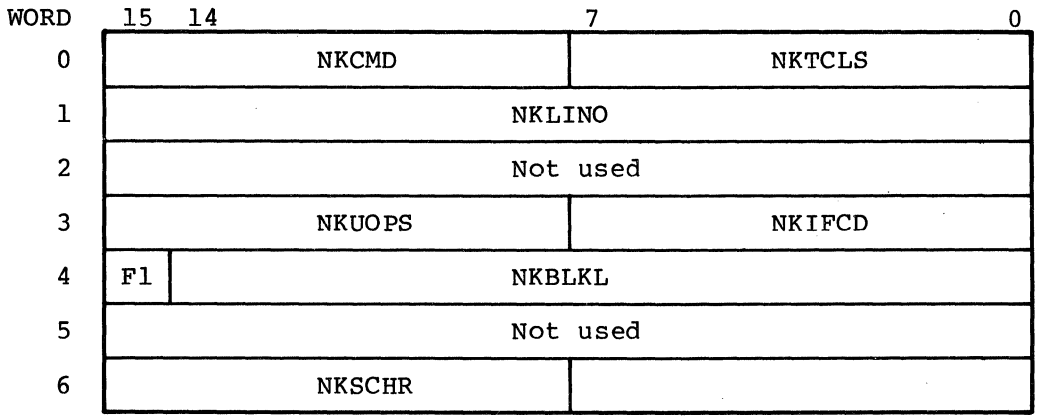
NKLINO - Line number.

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.

NICSRF1 - NKSRF5 is zero, the function is 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



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).

Figure 5-5. Enable Line Command Format (Sheet 1 of 2)

Fl - 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 (Sheet 2 of 2)

TABLE 5-3. OPTIONAL MODEM/CIRCUIT FUNCTIONS

Function Mnemonic	Function Provided	Description
N0ISR	STATUS†	Input status request
N0RTS	RTS	Request to send
N0SRTS	SRTS	Secondary request to send (Supervisory Channel)
N0OM	OM	Originate mode/auxiliary modem control
N0LM	LM	Local mode/auxiliary modem control
N0LT	LT	Local test
N0DTR	DTR	Data terminal ready
N0TB	TB	Terminal busy (line busy out)
N0RSYN	RSYN†	Resynchronize
N0NSYN	NSYN	New sync
N0BREAK	BREAK	Send break
N0DLM	DLM†	Data line monitor
N0ECHO	ECHO	Echoplex mode
N0LBT	LBT	Loopback test
N0ION	ION	Input on
N0OON	OON	Output on

TABLE 5-3. OPTIONAL MODEM/CIRCUIT FUNCTIONS (Contd)

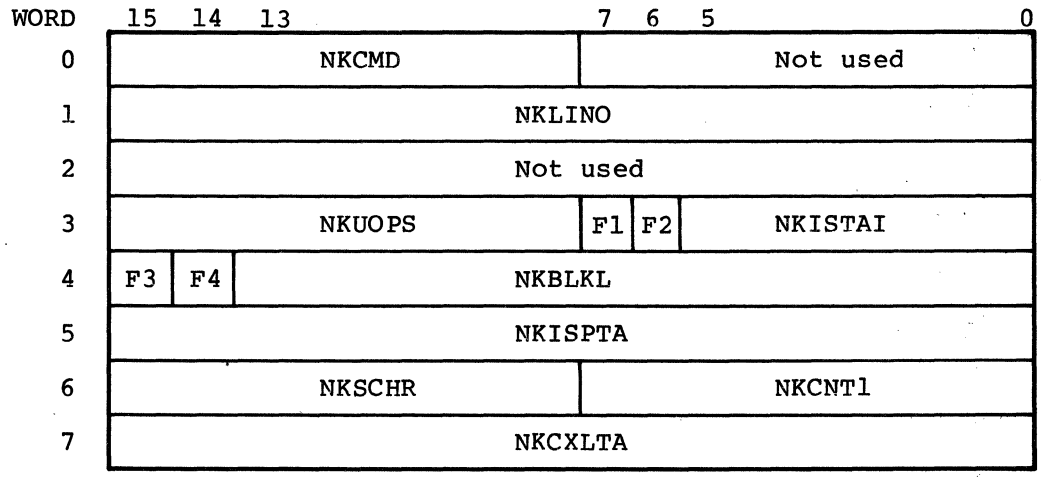
Function Mnemonic	Function Provided	Description
N0ISON	ISON	Input supervision on
N0PON	PON	Parity on
N0PSET	PSET	Parity set (1 = even, 0 = odd)
N0CLLS	CLLS	Character length (LSB)
N0CLMS	CLMS	Character length (MSB)
†Pulsed functions, provide momentary signal and need not be reset.		

INPUT COMMAND (NKINPT)

The input command directs the multiplex subsystem to initiate the processing of data on the specified input line (that is, 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.



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

Figure 5-6. Input Command Format (Sheet 1 of 2)

NKISTAI - Input state program index.

F3 - NKNOXL; code translate flag.

1 = translate

F4 - NKSCENBL; change special character flag.

NKBL - Block length. If this value is 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 (Sheet 2 of 2)

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, the subsystem: 1) 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 the disable command can 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.

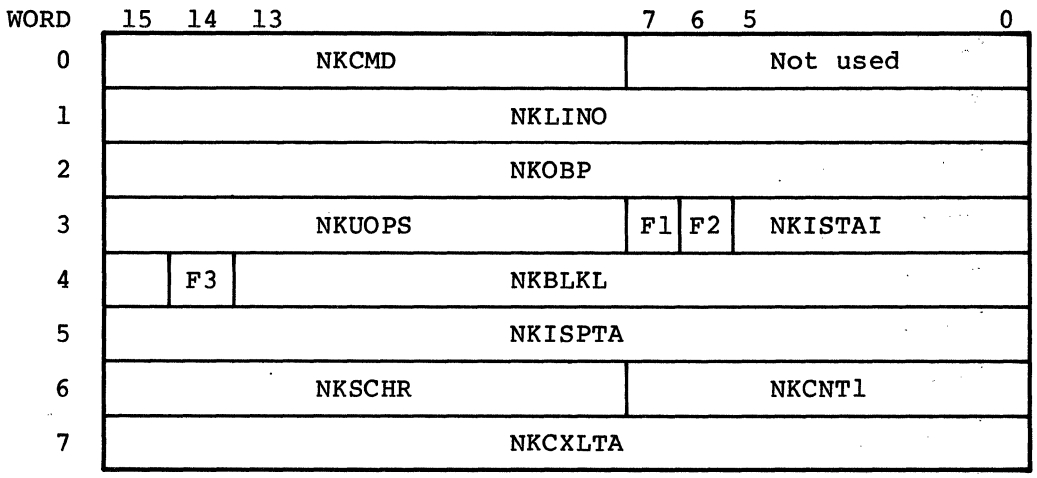
The format of the output command is as follows:

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

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

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.



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.

Figure 5-7. Input After Output Command Format (Sheet 1 of 2)

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 (Sheet 2 of 2)

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.

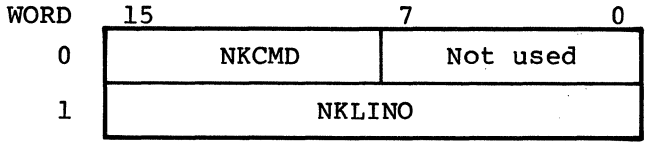
After processing the terminate input command, the subsystem optionally generates a worklist entry to the TIP as specified in the worklist and workcode.

TERMINATE OUTPUT COMMAND (NKENDOUT)

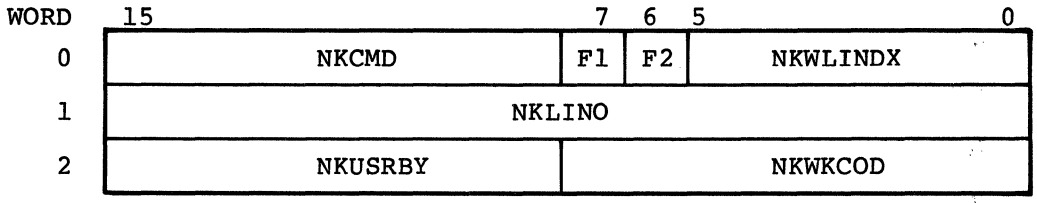
This command enables the TIP to direct the multiplex subsystem to terminate output processing functions on the specified line immediately. After processing the terminate command, an optional worklist entry is generated to the TIP, using the specified worklist and workcode. This command is used when the 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 the line, the system must issue control, initialize, and enable commands, followed by either an input or output command. The format for the disable line command is as follows:



NKCMD - Command code (NKDISL)
NKLINO - Line number



NKCMD - Command code (NKENDIN).

F1 - NKRELBFS; release buffer flag (release buffer if set).

F2 - NKWKFL; send worklist to user (if set).

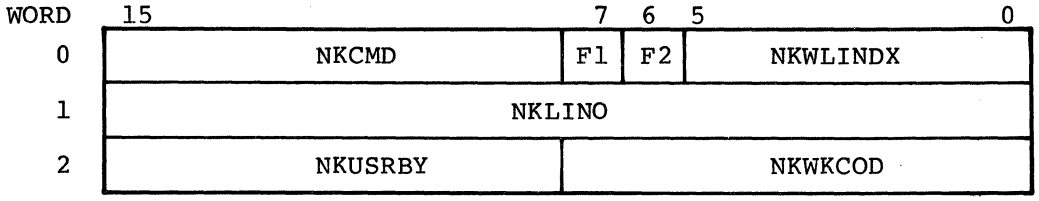
NKWLINDX - Worklist index; used if NKWKFLG is set.

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



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 (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.

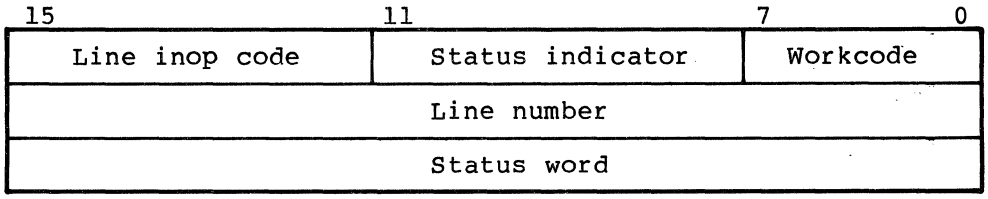
- MMBREAK - User break on synchronous lines. PTCLAS generates this condition after examining the status word. The user break indicates that the user has requested output to be terminated. The condition is reported to the CE error file and control is passed to the responsible TIP at multiplex level 2.

PTCLAS, CLA STATUS ANALYZER

Analyzing CLA status is a joint task of the modem state programs and PTCLAS. All incoming 2-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:



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 M0OVRT 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 M0OVTO 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
M0CLAON (0)	Modem state (MSTLNI)	Line initialized	Any status	Control to line initializer
M0RING (1)	Modem state (MSTLNI)	Ring indicator	RI status	Control to line initializer
M0ENBL (2)	Modem state (MSTENB)	Line enabled	DSR or DSR and DCD status	Control to line initializer
M0HERR (3)	Modem state (MSTCHK)	Hard error	ILE, OLE, INVALID RI, loss of DSR†	Control to TIP (supply INOP code and change workcode)

TABLE 5-4. PTCLAS WORKLIST ANALYSIS AND ACTION (Contd)

Indicator	Reported by	Meaning	Condition Detected	Action
M0SOER (4)	Modem state (MSTOUT)	Soft output error	NCNA status†	Control to TIP (change workcode)
M0SIER (5)	Modem state (MSTINP)	Soft input error	DTO, FES, loss of DCD status†	Control to TIP (change workcode)
M0STRT (6)	Modem state (MSTCHK)	Start modem timeout	Loss of DCD on constant carrier line†	Call PBTOQUE to start 15-second timeout
M0STOP (7)	Modem state (MSTCHK)	Stop modem timeout	DCD status during modem timeout	Cancel timeout
M0OVRF (8)	Firmware	CLA status overflow	Overflow of status counter	
M0OVTO (9)	PBTOQUE (TIMEOUT)	Status overflow timeout	10-second timer expired	
M0MRTO (A)	PBTOQUE (TIMEOUT)	Modem response timeout	15-second timer expired†	Refer to control to TIP (change workcode)
M0BREAK (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 M0STRT 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 M0STOP 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 or not 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.

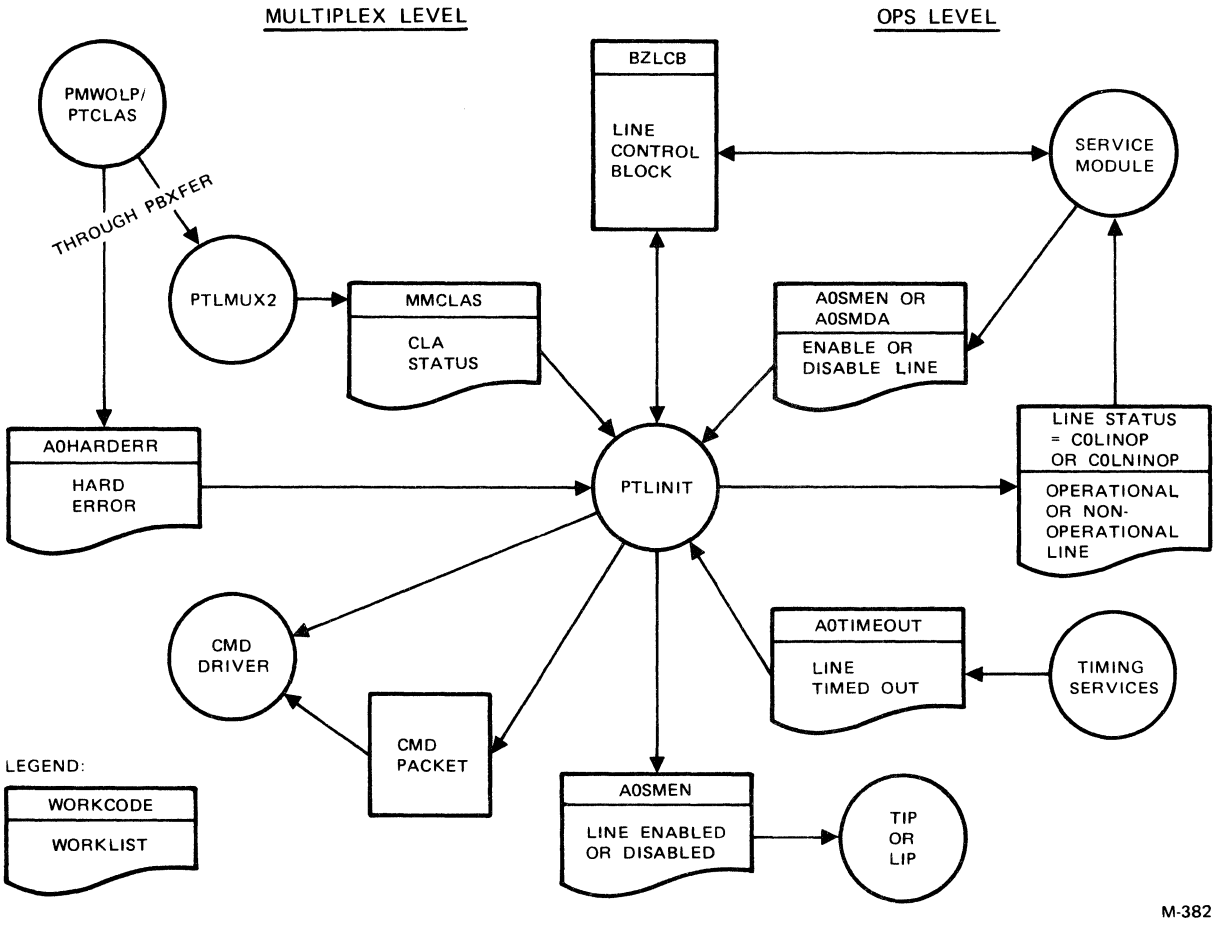


Figure 5-10. PTLINIT Relationships With Major CCI Modules

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. 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

Network communications software programs handle routing of blocks, some command execution (when the service module executes the command), and common TIP subroutines. The block protocol is discussed in this section.

The functions performed by the network communications programs are as follows:

- Defines the types of blocks that are acceptable for data transfer.
- Routes blocks. This includes checking the validity of incoming blocks and attaching the blocks to an NPU program that will continue processing the block.
- Provides and processes a special type of block reserved for command/status/statistics information. All service messages (SM) use this kind of block. The modules that process service messages are collectively called the service module. CE error, statistics, and alarm messages are special classes of service messages.
- Provides formatting for upline blocks so the block set to the HIP is in standard physical record unit (PRU) size and format (if required).
- Provides acknowledgment to assure that waiting batch data is transmitted as rapidly as possible.
- 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

This is the protocol 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 block header.

Block protocol assumes that the logical connection between processes in the host and the NPU is error free (a supportive, lower level protocol provides delivery assurance 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.

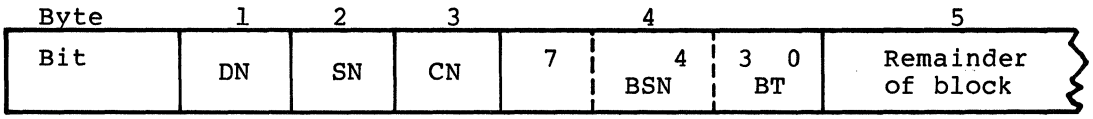
Failure of a process is usually reported by means of a service message. Temporary bottlenecks at a destination process are usually a result of inability to deliver data to an associated terminal or to the host. Block handling provides a standard method for informing the transmitting process of a temporary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

The starting and stopping of a data stream between a host application program and a terminal is handled by a special set of command blocks.

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

- Forward data (FD) functions are performed by BLK and MSG blocks and the command carrying CMD blocks. Two types of command blocks are defined: service messages which are handled by the service module, and other commands which are handled by the TIPs.
- Reverse supervision (RS) functions are performed by the BACK blocks which acknowledge reception of MSG, BLK, and CMD blocks.

The first four bytes of any block constitute the block header. Format of the block header is as follows:



DN - Destination node

SN - Source node

CN - Connection number (00 = service message channel)

BSN - Block sequence number (range 0 - 7)

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

The first three bytes of the block header provide a standard network address. The fourth byte contains block sequence number (BSN) and block type (BT). The content of the remainder of the block, if any, varies with the block type. An additional four bytes are reserved for control information in data (MSG and BLK type) blocks. Data block header information is shown in figure 6-2.

ADDRESS

The address occurs in the first three bytes. It contains the node IDs for the source and destination of the block plus a connection number.

Node

Each NPU has one unique node ID; each interface between host and an NPU has one unique node ID; Node ID = 0 is reserved for the host. The remaining node IDs range between 1 and 255, and are build-time parameters. For example, in a single host, single NPU system, the host ID 0, and the NPU ID is 2. Upline traffic from a terminal would have a destination node of 0 and a source node of 2. A service message going downline to the NPU would have a destination node of 2 and a source node of 0.

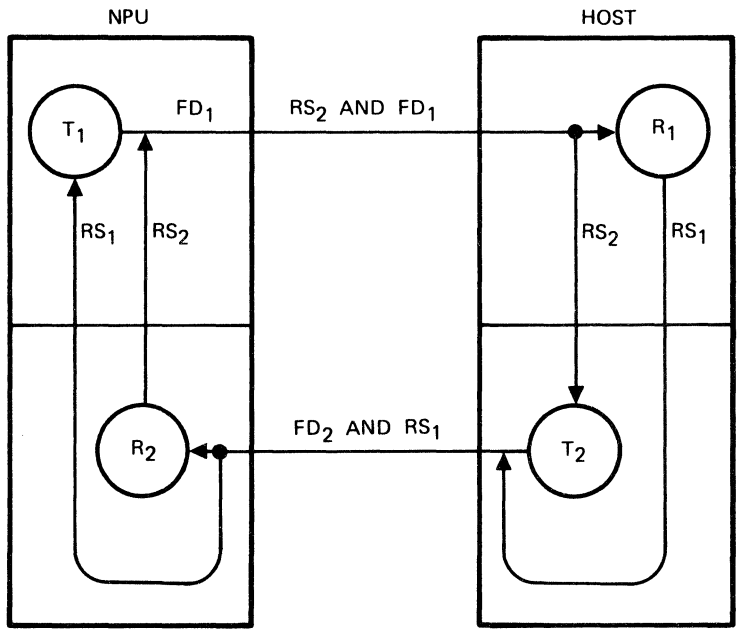


Figure 6-1. Communications Paths for Block Flow Control

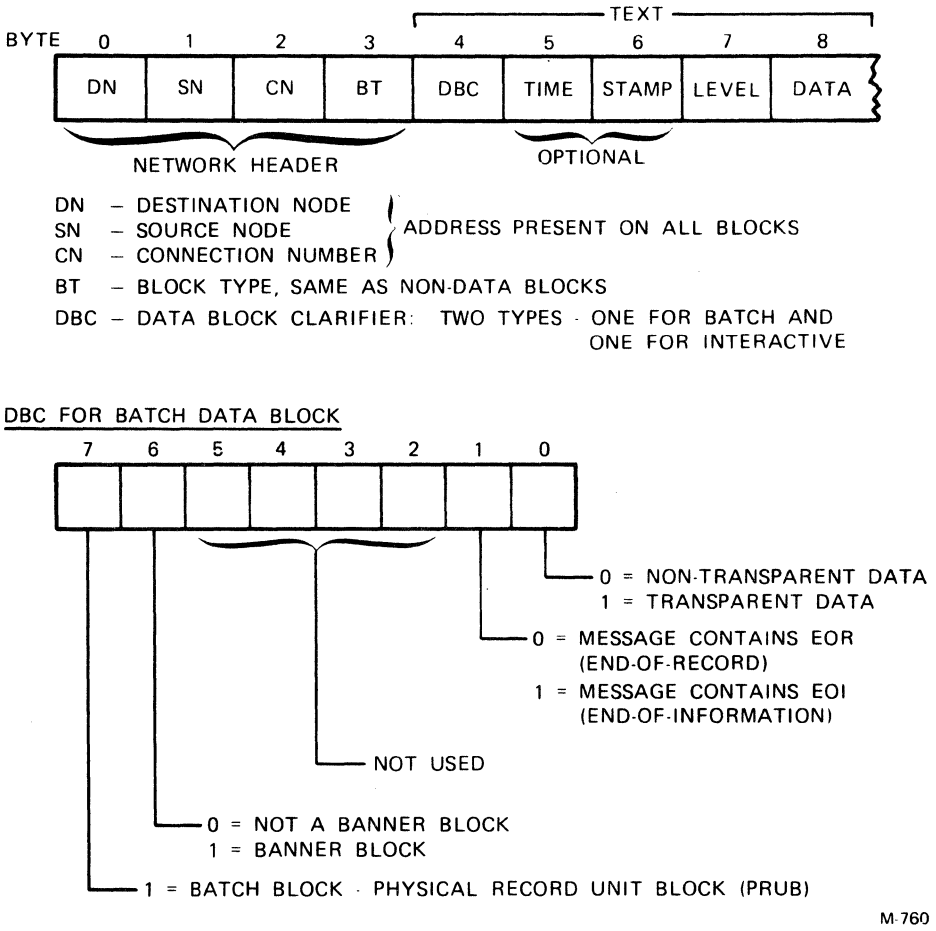
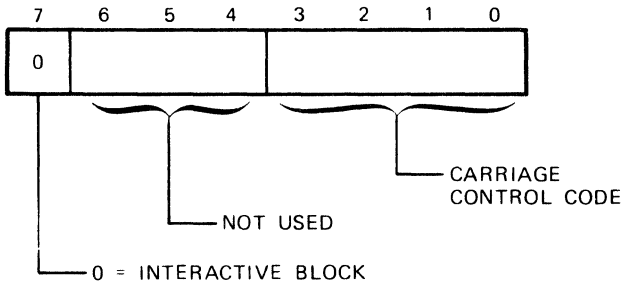


Figure 6-2. Data Block Header Formats (Sheet 1 of 2)

DBC FOR AN INTERACTIVE BLOCK



CARRIAGE CONTROL CODE	GENERAL FUNCTION
0	NEW POSITION
1	NEW PAGE
2	NEW PHYSICAL LINE
3	NEW LOGICAL LINE
4	NO SPACE
5	NO OPERATION
6, 7, 14, 15	INVALID
8 – 13	SAME AS 0 – 5

TIME STAMP – THE OPTIONAL, 2-BYTE TIME STAMP CONTAINS THE TIME THE LAST CHARACTER WAS PLACED IN THE PRUB FOR BATCH BLOCKS. INTERACTIVE BLOCKS DO NOT USE THE TIME STAMP FIELD. THE TIME STAMP IS USED FOR PERFORMANCE ANALYSIS ONLY.

LEVEL NUMBER – THE LEVEL NUMBER FIELD CONTAINS THE FILE LEVEL NUMBER RECEIVED ON THE EOR CARD OF BATCH DATA INPUT. INTERACTIVE BLOCKS DO NOT USE THE LEVEL NUMBER FIELD.

M-761

Figure 6-2. Data Block Header Format (Sheet 2 of 2)

TABLE 6-1. BLOCK TYPES

Mnemonic	Name	Block Type	Traffic Type	General Function
BLK	Block	1	FD	Data block which is a non-end-of-message block of a multi-block message
MSG	Message	2	FD	Data block which is the end-of-message block of multi-block message or an entire single block message
BACK	Block Acknowl-edgment	3	RS	Acknowledgment for block transmitted in opposite direction
CMD	Command	4	FD	Command either a service message (CN = 00) or a command to a line (CN ≠ 00).

Connection Number

A logical connection is the association between a terminal's terminal control block (TCB) and an application program in the host. This unique number therefore fixes the end points of each block transmission in the network. 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 associated with a particular host node. A CMD message with CN = 0 is a service message.

BSN/BLOCK TYPE

The fourth byte contains block serial number and block type information.

Block Serial Number (BSN)

Each block (CMD, BACK, BLK, or MSG) contains a block serial number (BSN) in bits 4 through 7 of the fourth byte. The BSN field is always zero on the service message channel.

BSNs are assigned sequentially by the transmitting process. This is a modulo 16 count that begins with zero when the connection is established. The count continues sequentially until the connection is dissolved. No correlation between upline and downline BSNs can be assumed.

The block receiver checks the BSNs of the received block against the next-expected serial number. If the number is correct, the next expected BSN count is updated. If the NPU detects an out-of-sequence number, the NPU sends an upline stop command to the host.

No specific recovery logic is included to restart a connection when the host receives an out-of-sequence upline block or an upline command from the NPU signifying receipt of an out-of-sequence block.

BLOCK TYPES

The block types are described in detail below.

BLK (Block) 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 2039 bytes of data immediately following the 4-byte block header and the additional 4-byte MSG/BLK block header. The content of the data field is determined arbitrarily by the communicating processes.

MSG (Message) Block

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

If a message is 2039 bytes or less in length, it can be transmitted within a single MSG block. All segments but the last are transmitted within BLK blocks: 1) if a message is longer than 2039 bytes, or 2) if, as is usual, the message is segmented by the terminal, or 3) in order to optimize NPU dynamic space. The last segment is transmitted within a MSG block.

Back (Block Acknowledgment) Block

A BACK block 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 an available block limit (ABL) for each connection. The BACK block, which acknowledges a previously transmitted block, allows the transmitter to maintain an outstanding block count to ensure that the ABL is not exceeded. ABL is established by the connection as a part of the configuration process. Note that no data bytes are associated with a BACK block.

CMD (Command) Block

A CMD block carries a network command and allows connected processes to communicate outside the data stream, but concurrently with it. The command is received by the destination process in the same order sequence to the data stream as existed at the source. For this reason, a set of commands (with CN not 0) is defined for stopping and starting stream. These commands normally originate in, or are processed by, a TIP. These commands are summarized in table 6-2.

The second major group of commands exist where CN is 0. These are called service messages (SMs). SMs normally originate with, and are processed by, the service module. A TIP can be called to perform processing on a service message, as well as call the service module to generate a service message.

Service Channel

The logical channel (CN is 0) for service messages is called the service channel. Unlike other logical connections which can be dynamically created and released, the service channel always exists. Service messages include commands, request for status, error information, statistics information, or replies to one of these 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.

Service messages are described in detail later in this section. The complete summary of service messages is found in appendix C.

Data Steam Control

The following rules are a part of the block protocol between NPU and host:

- Interactive data streams are usually open; therefore, these streams do not require start or stop commands.
- All TIPs start batch input streams by a command from the host, and start output streams by the first output block.
- All INTERCOM commands input at the terminal are passed to the host, and may result in a downline CMD to control the TIP.
- The TIP must notify the host of any terminal condition requiring operator intervention.
- On batch or interactive connections, upline CMD blocks from the NPU to the host must be suspended until the host is expecting a block on that connection. That is, the host has sent a BACK block acknowledging the previous upline block.
- On batch output connections, upline CMD blocks from the NPU to the host cannot be sent unless the host is expecting a BACK block. If the upline CMD is stream stopped, the BACK block that is due is suspended until a restart stream CMD block is received from the host. Then the BACK block is discarded.

DATA FORMATS

The data formats for INTERCOM are interactive format and batch format.

TABLE 6-2. COMMAND BLOCKS USED ON NONZERO CONNECTIONS

Traffic Type	Primary Function	Primary Function Code	Secondary Functions	Secondary Function Code	Use
RS	Start input	1	Initiate, nontransparent	0	Start input from batch device.
			Initiate, transparent	1	Start input from batch device.
			Resume	2	Start input from batch device.
RS	Stop input	2	Terminate	0	Discard data and stop polling.
			Suspend	1	Stop polling and wait.
FD	Input stopped	3	End	0	Normal end.
			Break 1	1	Reason for break defined by TIP.
			Break 2	2	Reason for break defined by TIP.
FD	Input	4	Restart after stop	0	Interactive started resume after break (status only).
RS	Output	5	Break 1	0	Reason for stopped break defined by TIP.
			Break 2	1	Reason for break defined by TIP.
			Break 3	2	Reason for break defined by TIP.
			Break 4	3	Reason for break defined by TIP.
			Break 5	4	Reason for break defined by TIP.
			Break 6	5	Reason for break defined by TIP.
RS	Output started	6	Restart after stop	0	Status only.
FD	Restart output	7	-	0	Resume output stream.
FD	Stop output	8	-	0	Discard data and terminate.
RS	BSN error	9	NPU has detected block sequence number out of order	0	Diagnostic purposes only.

Interactive Format Data

Interactive data (both upline and downline) is transferred between host and NPU over the coupler channel using the 7-bit internal ASCII code set. The NPU translates characters between the code set of the terminal and internal 7-bit ASCII code. All active terminals have at least one interactive connection configured.

Interactive streams are treated independently without concern for interference with batch streams. The TIPs resolve contention between interactive and batch streams. Interactive streams are always open and do not required commands to stop or start. The steams do not notify the host when a connection is stopped under normal conditions. A few abnormal interactive stoppages and resumes are reported to the host for the purpose of updating status.

Interactive input starts after the first output to the device. It continues until the terminal or device fails, or the line or terminal is deleted. If contention exists, interactive input is suspended when preempted by batch input or output to the same terminal. The interactive input automatically resumes whenever possible.

Upline interactive data blocks generally contain a single line of input from the terminal, and are normally followed by a single line of output from the host. The TTY TIP provides block mode and paper tape mode which allow multiple inputs from the terminal before output.

Interactive input data is terminated by special characters, by timeouts, or by the number of input characters, depending on the TIP and mode of operation. The input data is sent either in a BLK or MSG block, depending on the terminator.

Downline interactive data blocks can be terminated by any character. Once the NPU has started output to a device in the interactive mode, BLK blocks are delivered without allowing input until a MSG block has been delivered.

The host defines the desired interactive carriage control by specifying a logical carriage control function in the data block clarifier (DBC) field of each interactive output block. Each TIP translates the DBC codes to an equivalent function for output to the interactive device. In cases where there is not an equivalent function for the output device, the DBC code is either ignored or treated as a new line, depending on the terminal characteristics. DBC codes are shown in figure 6-2.

Batch Format Data

Batch data (both upline and downline) is transferred between the host and NPU over the coupler channel, using physical record unit block (PRUB) format. The PRUB is formatted to be directly compatible with a CYBER physical (disc) record unit (PRU) with the network block header appended to the front. Each data character is 8 bits, as stored in the NPU memory or transferred across the CYBER coupler interface. Data characters are either 6-bit display code (stored right-justified in each 8-bit character) for nontransparent modes, or the code of the terminal device for transparent data modes.

The PRUB can contain one to three PRUs (either 640 characters, 1280 characters, or 1920 characters maximum), depending on preset system option.

The PRU block is terminated by any of three conditions as follows:

- The maximum number of characters has been stored.
- An end-of-record (EOR) has been deleted.
- An end-of-information (EOI) has been deleted.

After the PRUB is terminated, it is forwarded to its destination. Only significant data within the PRUB is transferred across the connection.

PRUBs containing EOR or EOI must be less than the maximum PRUB size. Therefore, a record or file ending on exactly a 640- or 1280-character boundary will cause an additional PRUB to be generated which contains no data characters, but contains a bare EOR or EOI in the header. This applies to both upline and downline blocks.

The block type (BT) field within the network header specifies MSG if the PRUB contains EOR, EOI, or BANNER. The BT field specifies BLK for all other types of PRUB blocks.

NONTRANSPARENT DATA

For upline blocks, nontransparent data within the PRUB is assumed to be card data. For downline blocks, nontransparent data is assumed to be either print or punch data.

Each card input has trailing blanks suppressed and the end-of-card signified by at least two binary zero characters on a modulo 10 character boundary. That is, the NPU inserts 2 to 11 zeros following the last nonblank character on a card so that the total number of characters and zeros is an even multiple of 10.

For downline punch or print data, an 8-bit character of all ones (FF_{16}) signifies the end of each card or print line. The FF_{16} can be preceded by one binary zero character. Zero pad characters normally used to specify end-of-card or line within the PRU are not to be transferred downline to the NPU. The first character of each print line is treated as a carriage control character, using standard INTERCOM conventions.

Where an EOR or EOI card itself is discarded and the appropriate bits are set in DBC field of the PRUB header, EOR or EOI conditions are listed below.

- EOR - 7/8/9 punch in column 1 for Mode 4
- EOI - 6/7/8/9 punch in column 1 for Mode 4
  - /*EOI in columns 1 to 5 for BSC and HASP
- With bisynchronous ETX received from the terminal: ETX is generated when an ETX is punched in the last column of the last card or when the last card is entered with the EOF switch depressed. This is the only method of determining EOI when in the transparent mode for a bisynchronous terminal.

A 1- or 2-digit level number can be specified in columns 2 and 3 of the EOR card. This level number, if present, is converted to an 8-bit binary value and transferred upline to the host in the level number field of the header of any PRUB containing EOR. The level number field is zero if not present in the EOR card.

Downline PRUBs containing EOR or EOI that are directed to a punch device cause an EOR card (7/8/9/ punch) or EOI card (/*EOI) to be punched. The level number contained in the header of an EOR block is also punched in columns 2 and 3 of the EOR card.

TRANSPARENT DATA

Transparent data within the PRUB provides a method of transferring batch data between the terminal and the host files without modification by the CCI software. The block header is identical to that defined for nontransparent data.

Input data received in the transparent mode is stored in the PRUB without code translation, data expansion, or blank suppression. Transparent PRUB blocks are terminated and forwarded to the host when one-half the number of characters specified for the PRUB size is reached (320/640) or the end-of-information is reached. EOR and EOI cards are not recognized in the transparent data mode. Therefore, end-of-information is detected by receiving ETX from the terminal.

Transparent input data is specified by three methods, as follows:

- Optional parameter on the INTERCOM command READ FILE NAME
- TR in columns 79 and 80 of the job card
- TR in columns 79 and 80 of the EOR card

The full 8 bits of each data character are written to disk PRUs for transparent upline PRUB blocks.

Output files can also be specified as transparent. The host marks the header of each PRUB as transparent for the files. Transparent PRUBs are output to the terminal without modification of the data characters (no code translation, data compression, carriage control line folding, etc.). Data characters are, however, blocked into the maximum size transmission blocks specified for the terminal device receiving the data. Transmission blocks are terminated at the last data character of a PRUB block that is marked as an EOR or EOI block.

ROUTING

Routing of blocks is performed by the internal processing, usually called through PBINTPRC. The internal processing 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 which must continue processing the block.

Upline blocks that are completely processed are passed to the HIP for transmission to the host.

Downline blocks that are to be sent to terminals are queued to the TCB, which is associated with the terminal/device that is to receive the message.

A second source of switching uses PNROUTE. At present, only the service module and utilities use this switching method.

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 6-3.

Destination Node Directory

The destination node directory contains an integer value associated with each valid DN address (range: 0-255). 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 Directory

For each logical node there is a connection directory for all terminals, with at least one conection defined. An entry in the connection 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 connection 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 routing; figure 6-4 indicates the steps of the routing search.

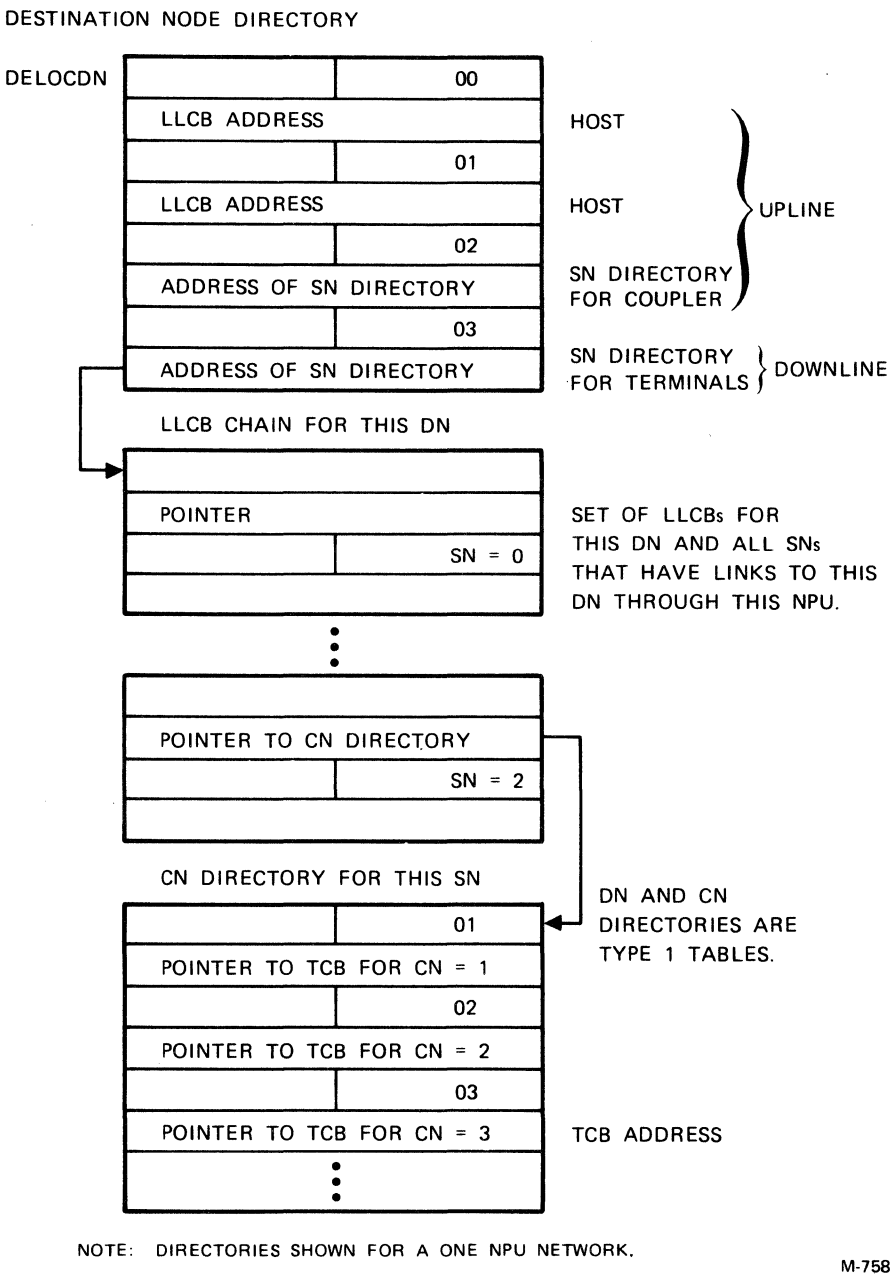


Figure 6-3. Use of Routing Directories

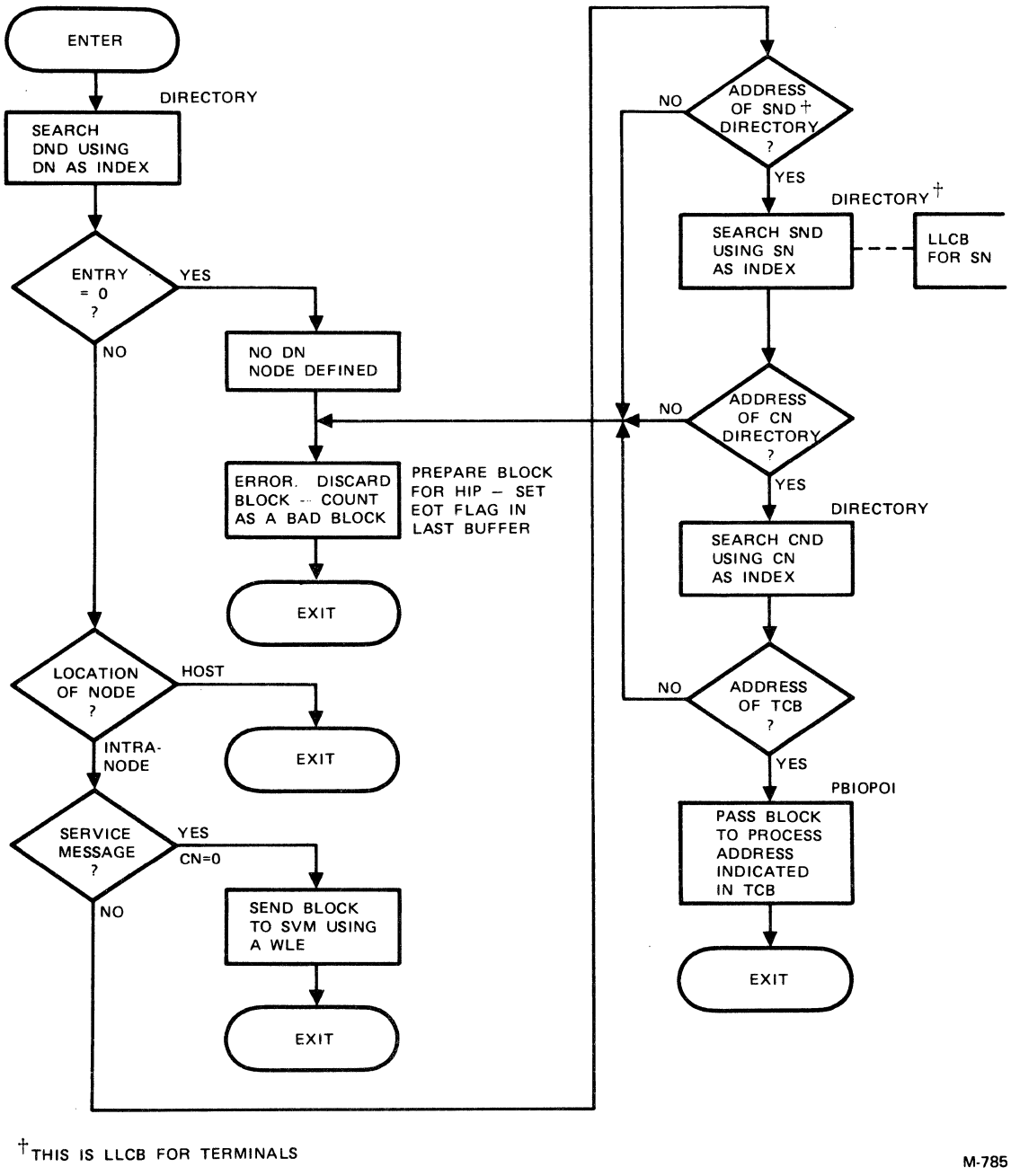


Figure 6-4. Simplified Routing Flow Chart, PBSWITCH

Following the routing flow, DN indexes the destination node directory to obtain an address. If the address obtained is zero, the destination of the block is undefined, and PBSWITCH discards the block without notifying the sender. However, the bad block count is incremented.

Service messages are passed to the service module using a worklist entry.

The LLCB for the terminal load 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 CN's 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, listed as follows:

- The first two are PASCAL values in the range 0 to 255, and represent DN and SN values respectively.
- The third is a PASCAL variable in the range 1 to 255, and represents CN.
- The fourth is a PASCAL variable of the buffer pointer type (range 2 to 65 and 535) that points to a TCB for use in the appropriate directory.

The DN directory can have a new 2-word entry. The CN directory can have new entries as well as new chained segments, if necessary. LLCBs (the SN directory) are pre-established.

PNDIRDLT removes entries from the DN and CN directories. Three input parameters are necessary, listed as follows:

- 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 the index to the SN entry to be removed.
- The third is a PASCAL variable in the range 0 and 255, and is the 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 (SM)

The special group of control messages (SMs) 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/deleting control blocks. These include control blocks for lines (LCB) and terminals (TCB).
- Generating and sending status SMs. These include line and terminal status SMs.
- Generating and sending statistics SMs.
- Generating and sending broadcast one and broadcast all SMs.

INTERNAL SM 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 SMS

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

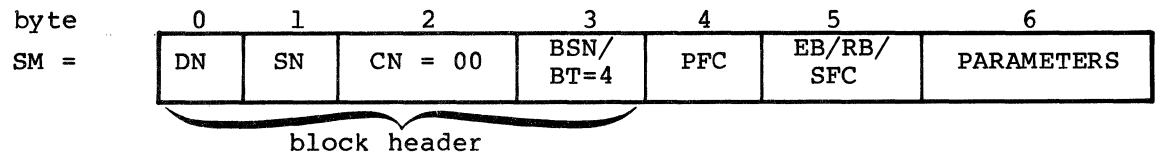
The validation group of modules assures that all SMs have:

- A valid primary function code (PFC) and a 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 as follows:



NOTE

Bytes are number starting at DN byte.

DN - Destination node

SN - Source node

CN - Connection number = 00 for all service messages. The service channel is always assumed to be configured

BSN - Block serial number; bits 7-4 of byte. Always = 0 for SMs

BT - Block type = 4, command block. This is lower 4 bits of byte

PFC - Primary function code.

- $00\text{-}3F_{16}$ - Reserved for network use
- $40\text{-}9F_{16}$ - Reserved for intra-host use (error for CCI 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, which is bit 7 of byte 5
- RB - Normal response SM; EB = 1, which is bit 6 of byte 5
- SPC - Secondary function code; see appendix C, table C-1, bits 5 through 0 of byte 5
- Parameters - Defined in bytes. See appendix C

GENERATING AND DISPATCHING SERVICE MESSAGES

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 test. There is no response to this type of service message.
- Generates PFC and SFC for service messages.
- Dispatches the SM to:

  The HIP if DN designates the local coupler.
  SVM if DN designates a local CCI action.

CONFIGURING/ENABLING/DISABLING/DELETING CONTROL BLOCKS

This set of modules is used for initiation and changing control blocks for lines and terminals. The format and functional effect of these messages are described in detail in the initialization section of the CCI 3 reference manual and in section 2 of this manual.

GENERATING AND SENDING STATUS SMS

This group of modules generates and sends the logical link, line, and terminal status messages. Included in these operations is the ability to count configured lines. The status also indicates whether or not the line is operational.

Line Status Request SM

This 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 this host. 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:

- Port invalid.
- Another line status request is in progress.
- 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 (see unsolicited response, below).

Upon receiving the line status operational SM, the host configures the terminals for the line by sending one or more configure terminal SM(s).

An unsolicited line status request SM is sent whenever the TIP senses conditions causing 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, which are to 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 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; 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, line speed and code type, and the device type. For autorecognition responses, the terminal address and device type are repeated for each terminal that can be detected by the TIP. The TTY TIP reports only one terminal address/device type pair.

Line Count Request SM

The host 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 SM

The host 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 specifies the line to be checked.

The response may be either for a request, or 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
- No terminals configured
- Line inoperative or not enabled
- Another terminal status request SM is in progress
- LCB not configured

GENERATING AND SENDING STATISTICS SMS

The network operator can send a message to one or all terminals. This message text is carried in a service message to the NPU, where it is copied, then sends the terminal-directed messages to the interactive terminals.

The message identifies the cluster and terminal addresses, and the device type of the receiving terminal. The network operator produces the text of the message. The procedures for entering this message from the host console are given in the NOS/BE operator's guide.

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

- Invalid line number
- Invalid device type
- Terminal or line not configured
- Terminal or line inoperative

A broadcast message can be sent to all interactive terminals connected to the NPU. In this case, 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/BE 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. Two types of errors are reported, as follows:

- No logical link established, or this logical link is not established.
- Another broadcast SM is already in progress.

CE ERROR MESSAGES

CE error messages are special SMs that report hardware failures. These messages all include a 1-byte CE error code, and can include additional data. These messages are described in appendix B of the CCI reference manual.

COMMON TIP SUBROUTINES

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

POINT OF INTERFACE ROUTINES (POI)

Four Point of Interface routines are included in the internal processor. These routines handle many of the interfaces enabling the TIPs to begin or to end processing of a message. The programs are as follows:

- PBPIPOI - Post input POI
- PBIOPOI - Internal output POI
- PBPROPOI - Preoutput POI
- PBPOPOI - Post output POI

PBPIPOI, Post Input POI

This POI is called when the TIP has a block to be passed upline to the host. PBPIPOI first calls PNSGATH to gather statistics on the transfer. (See figure 6-5.) It then calls a group of nested subroutines. These subroutines are listed as follows:

- PBIIPOI determines if the block is from a batch or interactive terminal. If it is for an interactive terminal, the block header is completed. The subroutines to convert the data to PRU format are called. In both cases, the block is routed upline to the host, if possible. Otherwise, the block is added to the upline PRU queue.

- ITPRUPOI determines whether the block from a batch device is a command or data. If it is data, the conversion subroutines are called; if it is a command, the header is completed. The output stopped command is sent immediately if an upline BACK block is pending. Other commands are sent immediately unless the terminal is waiting for a downline BACK block.

- BLKTOPRUS processes each buffer in the block to transform the data to PRUB format. This consists of setting the level number, setting the transparent indicators (if necessary), splitting buffers (as necessary), rechaining buffers, and setting character count in the transformed buffers. Also, the EOI must be set.

- ENDPRU performs buffer chaining during the transformation, and handles header and chaining operations.

PBIOPOI, Internal Output POI

This routine is called by the switch to process downline blocks that are routed to TIPs. The POI first checks if the block is in sequence. If it is not, PTCOMMAND is called to generate a sequence error message for this connection. PBIIPOI sends the message to the host. If the block is for an interactive device, PBIOPOI determines the block type as follows:

- Data blocks are regularly queued to the TCB output for all TIPs except the Mode 4 TIP. Data blocks are specially queued for Mode 4 devices.

- CMD blocks are queued to the TCB's output queue.

- BACK blocks cause the outstanding block count to be decremented. If another block is waiting to be routed, PBRTEIA does this.

If the block is for a batch device, the batch downline routing subroutines are called.

- IOPRUPOI checks the block type.

  Data blocks are transformed to TIP input blocks if they are currently in PRU format.

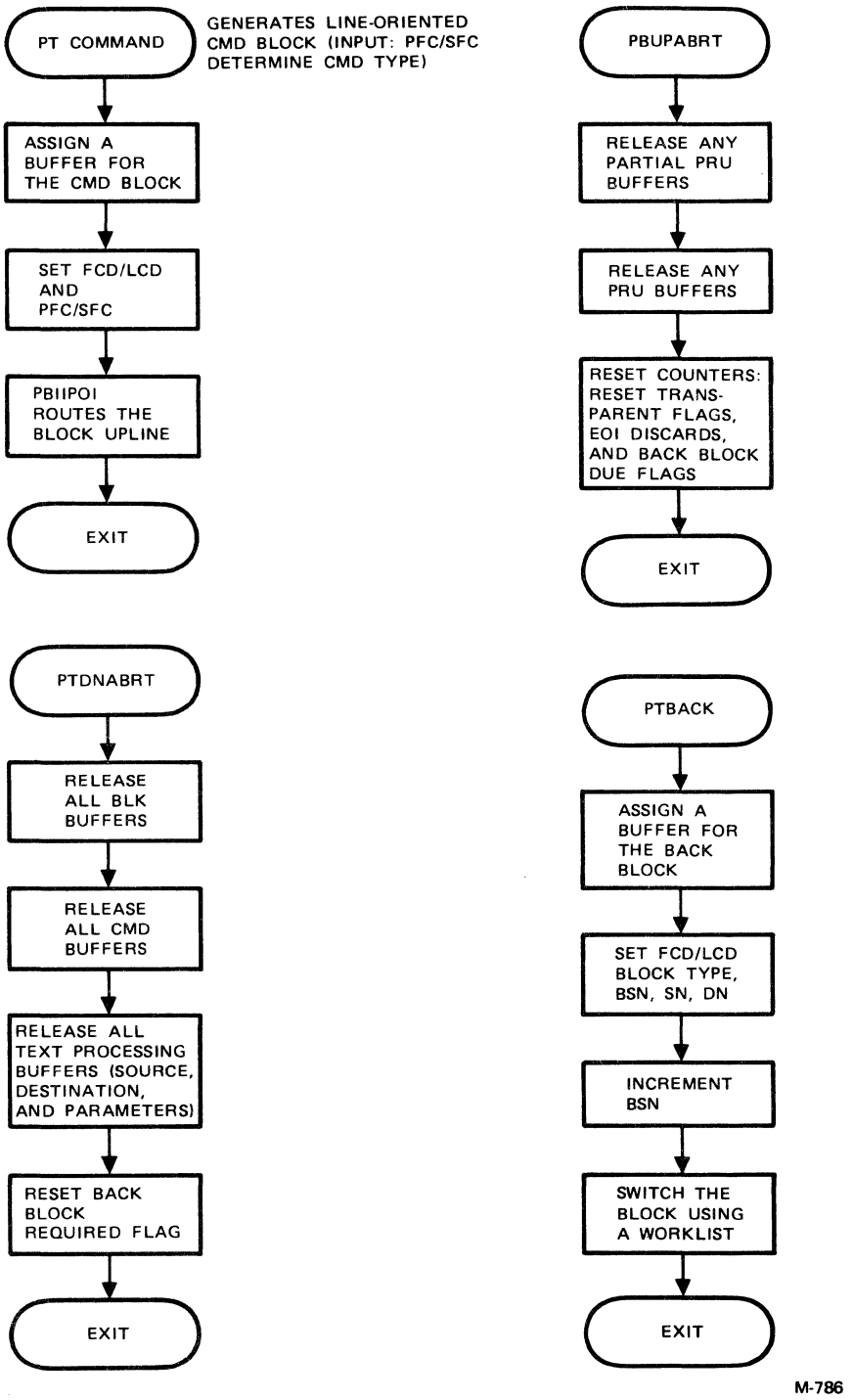


Figure 6-5. Important Common TIP Subroutines (Sheet 1 of 7)

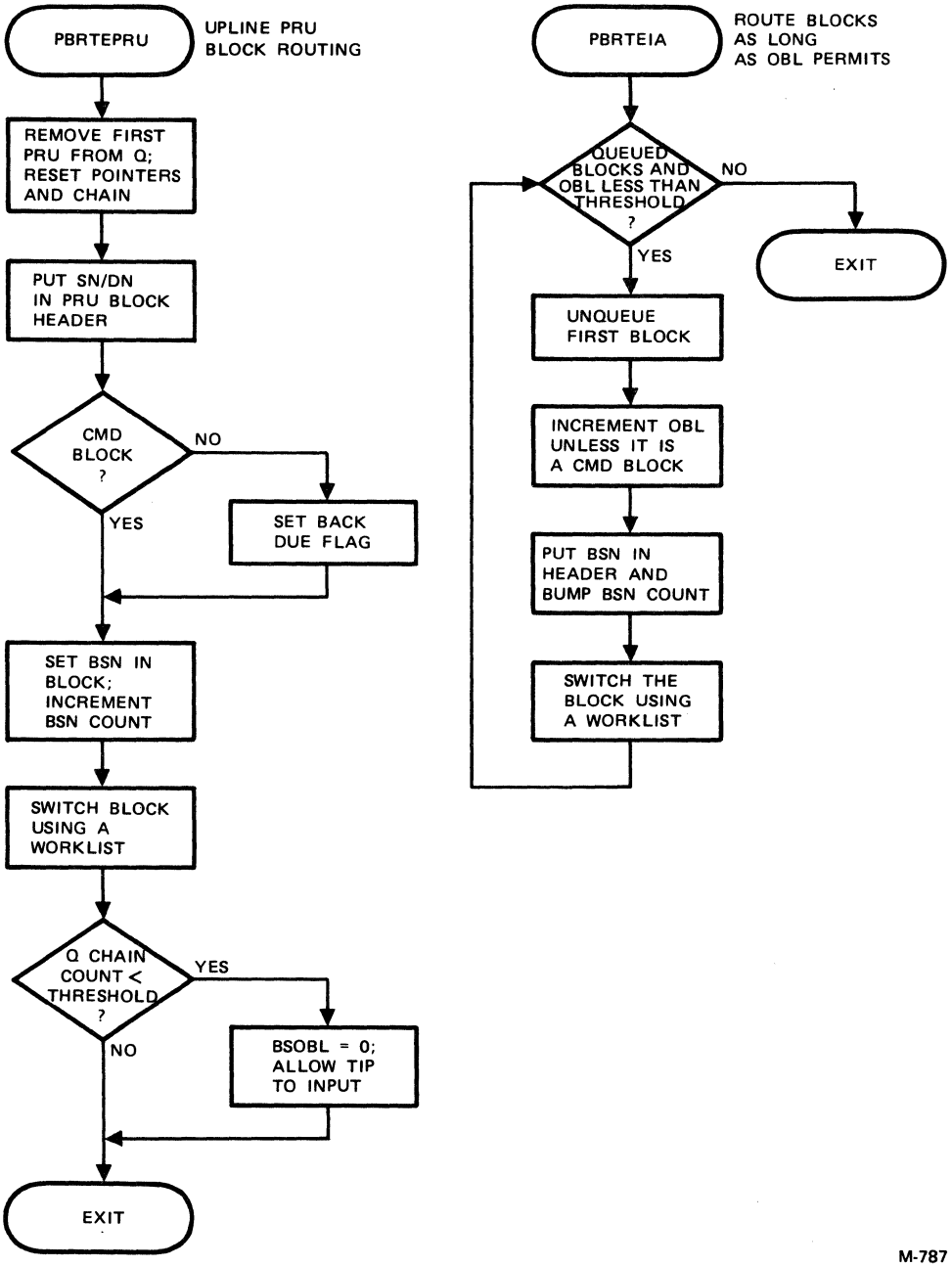


Figure 6-5. Important Common TIP Subroutines (Sheet 2 of 7)

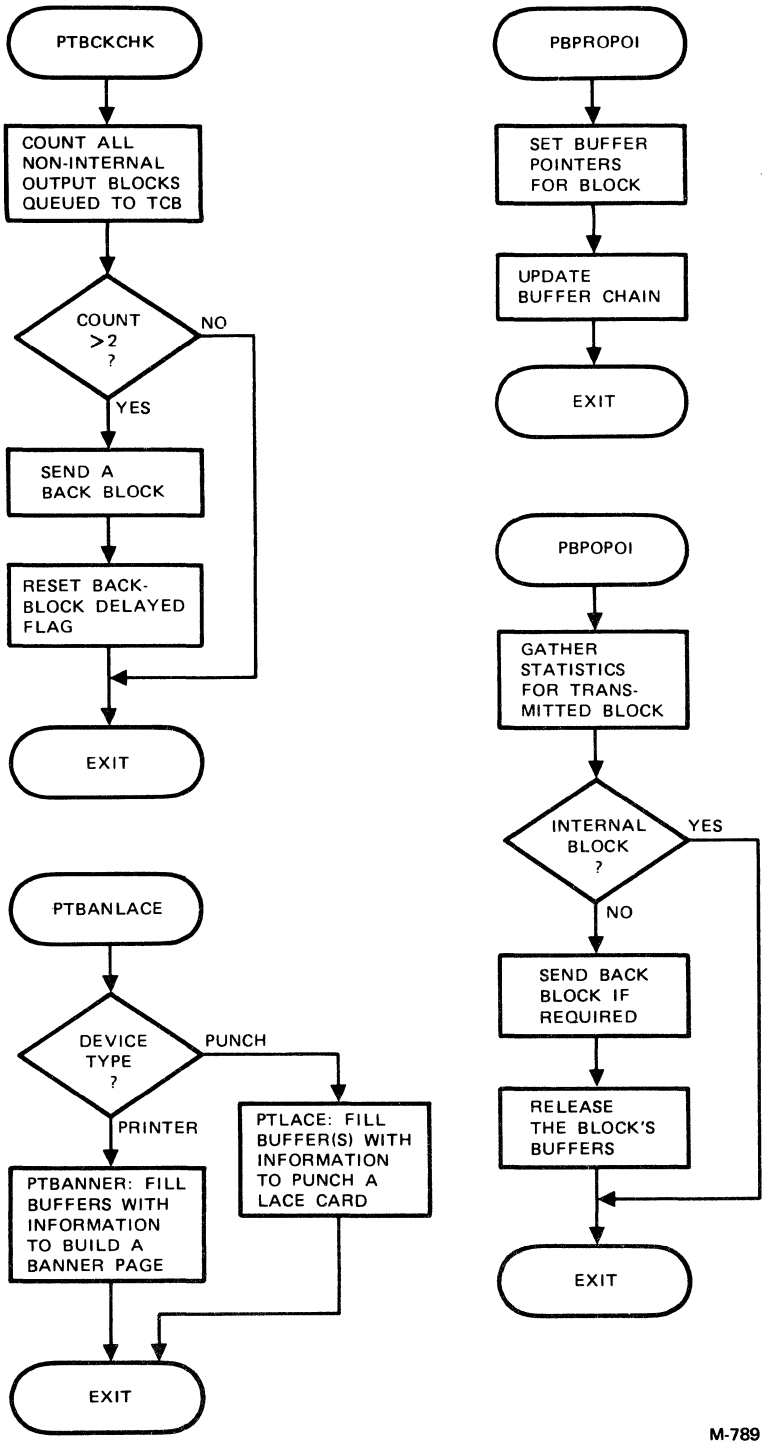


Figure 6-5. Important Common TIP Subroutines (Sheet 3 of 7)

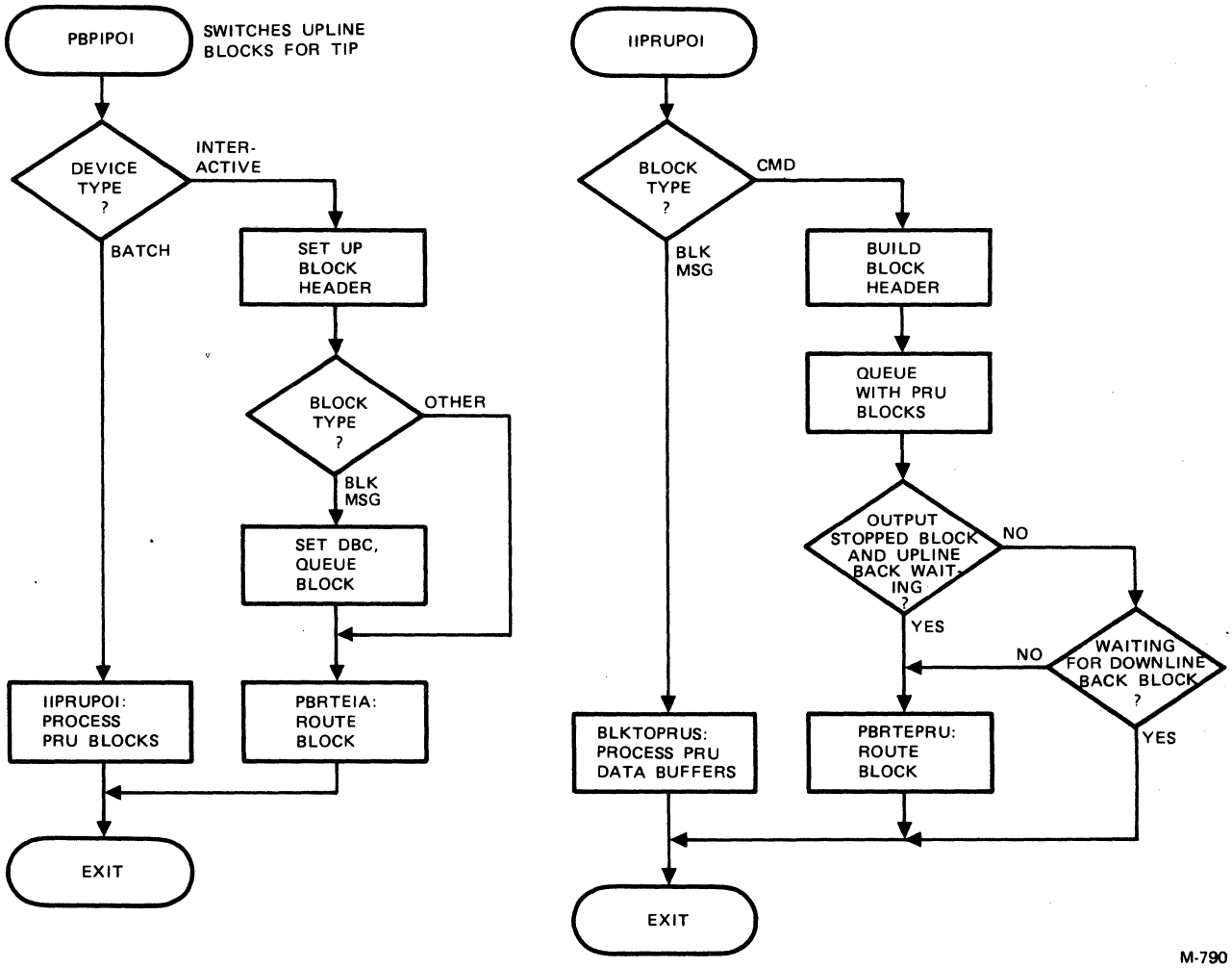


Figure 6-5. Important Common TIP Subroutines (Sheet 4 of 7)

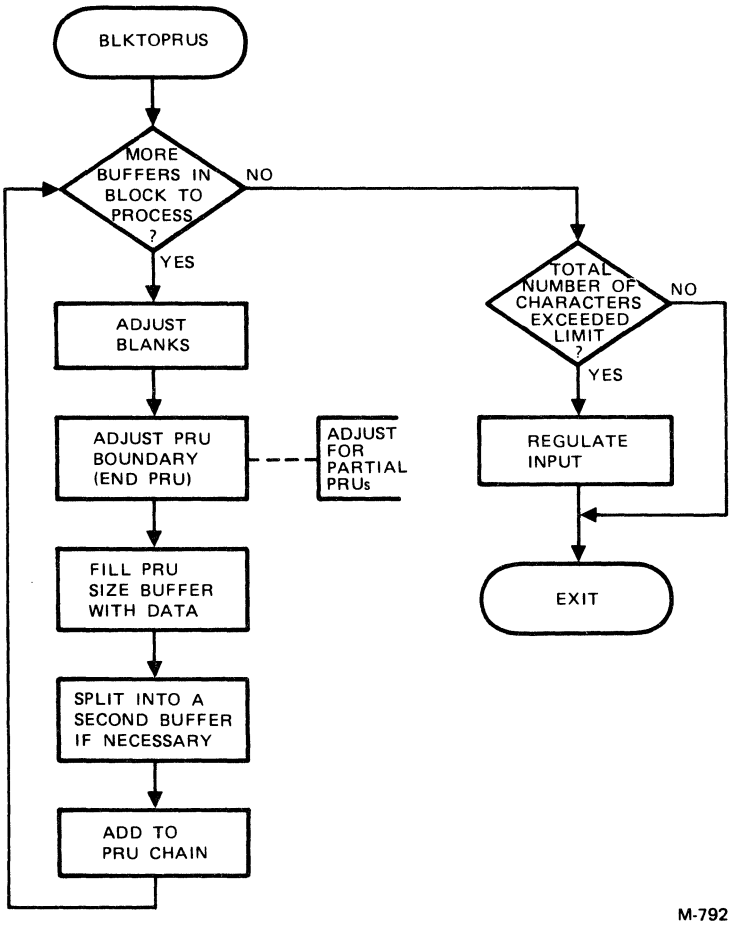


Figure 6-5. Important Common TIP Subroutines (Sheet 5 of 7)

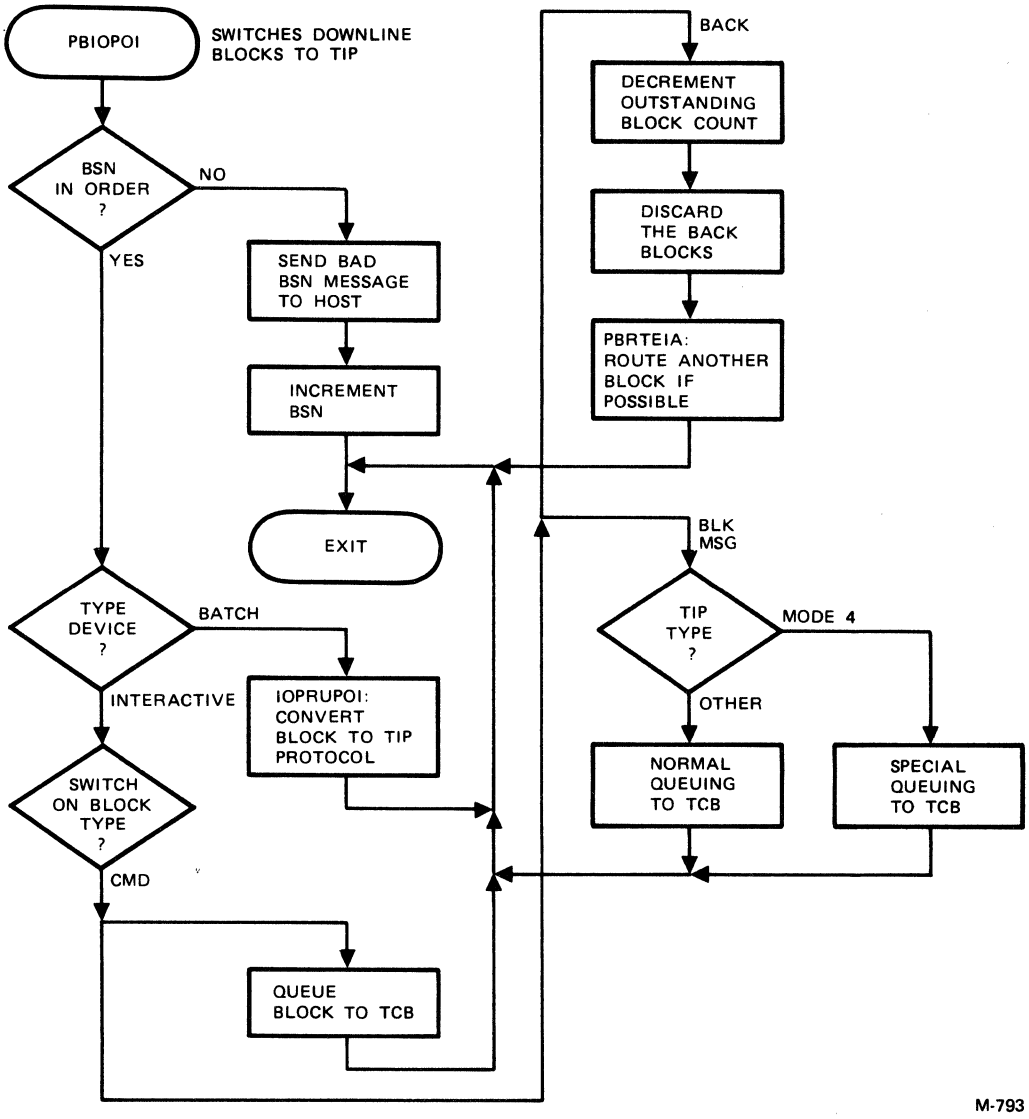


Figure 6-5. Important Common TIP Subroutines (Sheet 6 of 7)

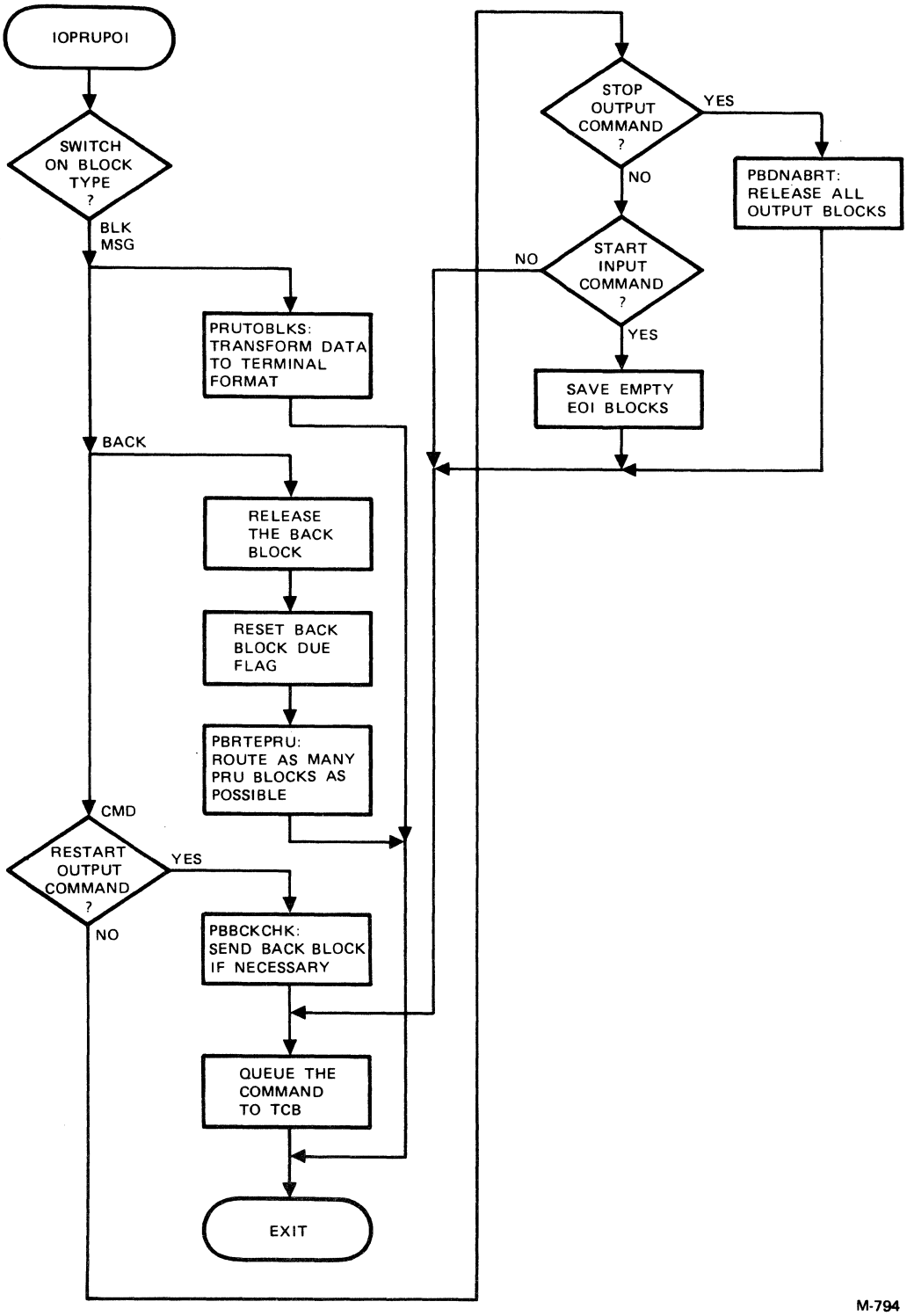


Figure 6-5. Important Common TIP Subroutines (7 of 7)

CMD blocks are of three types. Restart output CMD blocks cause the POI to send a BACK block if the back block threshold has not been reached. Stop output commands cause all blocks queued to this TCB to be released. Start input commands prevent empty EOI blocks from being discarded. In all cases, the CMD block is queued to the TCB's output queue.

BACK blocks cause waiting PRU blocks to be routed upline.

- PRUTOBLKS checks the block use.

  If this is a banner block and banner blocks are to be used, PTBANLACE generates the banner block.

  If this is a text block and if text processing is required, PBXFER calls the TIP's text processing routine to transform the text to terminal format/code at this point. Blocks are queued and BACK blocks are sent, or set for later sending, as a function of the current TIP state.

PBPROPOI — Preoutput POI

This POI is used to update pointers in the output message block that is queued to the TIP.

PBPOPOI — Post Output POI

This POI is called from the TIP's post output routine to generate the statistics for the block (using PNSGATH), and to send a BACK block if one is necessary, assuming the block was not internally generated. (BACK blocks are always generated for interactive downline blocks.) The POI then releases the buffers holding message which the TIP has now finished processing.

STANDARD TIP SUBROUTINES

OUTPUT QUEUING (PBQ1BLK AND PBQBLKS)

Output queues are associated with a specific TCB. That TCB contains a pointer to the first block in the queue, specifically to the first buffer of that block. Figure 6-6 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.

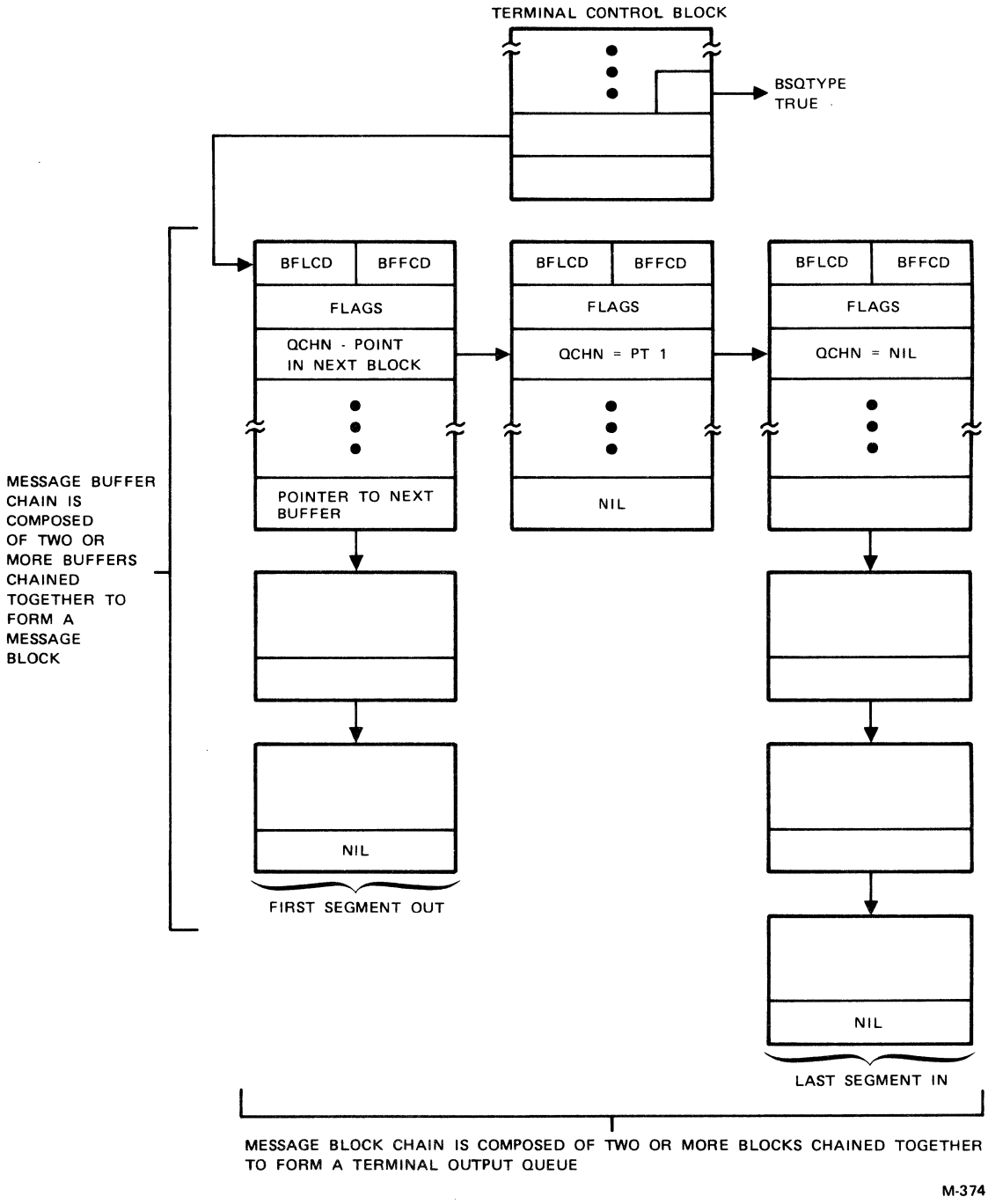


Figure 6-6. Structure of a TCB Queue

- PBQBLKS (parm1, parm2) uses parm1 to find the TCB output queue and parm2 to find the buffers to be added to the chain. If the TCB queue is empty, a worklist entry is made to the TIP which controls the TCB; the TIP can process the queue.

The TIP which must process the message calls PBQ1BLK (or PBTOCONS if the message is for the NPU console). PBQ1BLK is called indirectly using the internal output POI (PBIOPOI).

REMOVING A MESSAGE SEGMENT FROM QUEUE PBGT1SEG

The form of the call used to remove a single buffer from the message is as follows:

PBFT1SEG (parm) where R3SEGPTR contains a pointer to the buffer to be removed from queue. For parameter setup, B1TCB is a global variable containing a pointer to the TCB associated with the queue.

PBGT1SEG is a PASCAL function that returns as follows:

A zero value, if the queue was busy and no buffer was removed from the queue. A value of one, if the queue was not busy, but was empty, and no buffer was removed from the queue.

A value of two, if the queue was neither busy nor empty and a buffer was removed from the queue.

It is assumed in the foregoing that B1TCB does not contain a NIL pointer. It should further be noted that the chain word of a returned segment will not necessarily be NIL.

SAVING AND RESTORING REGISTERS

Two subroutines save and restore the R1 and R2 registers: PBBEXIT & PBAEXIT.

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.

PBBEXIT 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 can 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 2-word save area for R1 and R2.

Interface to Text Processing Firmware, PTTPINF

TIPs call this interface to firmware routine to execute the upline or downline text processing state programs (upline text processing is used only for TIPs that require two-stage input processing, such as the HASP TIP). After escaping to firmware processing, PTTPINF 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, PTTPINF returns control to the calling program. If the text could not be converted, PTTPINF notifies the calling program of the failure.

This module is technically a part of the base system; it is discussed here since it provides a TIP-related service.

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. PTCTCHR is also 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. 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. The former is used for input; the latter is used for output. The TIP will later receive control by a worklist entry to continue processing at this point.

- 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. The former is used for input; the latter is used for output.

These modules are also part of the base system.

Common Return Control Routine, PTRETOPS

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

Common Tip Regulation, PTREGL

The common 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 has been completed on upline data), CCI 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 block.

NOTE

This routine is not called by the mux 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 of checking whether or not 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 mux command driver to turn off the CLA. Until the CLA state is changed, the mux subsystem must continue to accept input data.

The calling format is: PTREGL (parm1, parm2). Parm1 is a pointer to the buffer associated with the proposed input operation. Parm2 is the type of comparison to be made.

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

PTREGL is technically part of the base system.

Set Logical Link Regulation, PNLLREG

This routine is called from the HIP only to set the logical level in the LLCB. The levels are UP or DOWN. In the former case, all messages are accepted; in the latter case, all messages are rejected. The call is

PNLLREG (parm)

where:

parm is the new regulation level, UP or DOWN.

Set Accept Input/Accept Output Flags, PTINIT

This routine is called only from the SVM, but it is used to set the accept input and accept output flags in the TCBs after the TCB has been completely configured. The call is PTINIT.

Discards Non-Routable Blocks, PBLOST

This routine is called from the routing function to handle blocks with a header (DN/SN/CN) that cannot be routed to an existing node or terminal according to the information in the routing directories. The blocks are counted as bad address blocks (for NPU statistics), and the block is released. No acknowledgment of any sort is generated. The call is PBLOST.

Upline Abort, PBUPABRT

This routine is called by a TIP when upline traffic continuity cannot be maintained. Partial and full PRU buffers are released. The TCB fields for character count, block count, and various other flags are cleared. The call is PBUPABRT. This routine is also called by SVM.

Downline Abort, PBDNABRT

This routine is called by SVM when a TCB is deleted, or by a TIP when downline traffic under internal processing control is to be discarded. All BLK and CMD buffers are released, as are all source and destination buffers, for a text processing operation in progress. Various TCB fields are cleared. The call is PBDNABRT.

Send CMD Block to Host, PTCOMMAND

This routine is called to generate the CMD blocks for starting and stopping a data stream. The list of such commands is shown in table 6-2. The TIP calls the program with:

PTCOMMAND (parm1, parm2)

where:

parm1 = PFC. TIP can use these primary function code values:

3 - input stopped
4 - input started
5 - output stopped
6 - output started

parm2 = SFC. The TIP uses the secondary function codes, as shown in table 6-2.

PTCOMMAND uses PBIIPOI to route the block upline to the host.

Upline PRU Block Routing, PBRTEPRU

This routine is called by PBPIPOI. It removes the first upline PRU queued to the designated TCB. The routine completes the header including the block serial number (BSN), and uses a worklist to switch the block. A flag is set which causes the NPU to expect a BACK for MSG, BLK, or CMD blocks. After the block is switched, the routine checks the number of characters queued for this TCB. If it is below the threshold value, BSOBL is set to zero. This allows input to be received from the terminal. The call to the routine is PBRTEPRU.

PRU Block Routing, PBRTEIA

This routine is called for both upline and downline block routing. If there are queued blocks and if the block is either a command block or if the available buffer limit is below threshold value, the first block is unqueued from the TCB. If a data block was unqueued, the BSOBL buffer's counter is incremented. The BSN is put in the header and BSN count in the TCB is incremented. A worklist is used to switch the block. Call to the routine is PBRTEIA.

Check to Find if Back Block is to be Send, PBBCKCHK

CCI uses a scheme of sending BACK blocks so that data flow from the host is optimized. PBBCKCHK is the routine which counts the number of queued non-internal blocks, and sends a BACK block if the number is below a preset value (a build time parameter). When the host receives the BACK block, it can send more downline blocks (if any are waiting) to this terminal. PTBACK sends the BACK block. After sending the BACK block, PBBCKCHK clears the flag which indicates there is a BACK block waiting to be sent upline. Call to the routine is PBBCKCHK.

Generate Banner and Lace Records, PTBANLACE

This routine is called from internal processing when a banner record is needed for a printer or a lace card is needed for a punch. Call to the routine is PTBANLACE, together with an input buffer containing the block header and a job name.

If the device is a punch, PTLACE formats buffer(s) for a lace card of blanks (EOL card has been detected or FF_{16} has been detected), or with the job name. If the device is a printer, PTBANNER builds the banner page buffers with the job name.

HOST INTERFACE PACKAGE (HIP) 7

This section describes the operation of the Host Interface Package (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 is resident in the host; the other is resident in the NPU. The NPU program is called the Host Interface Package (HIP). 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 the host and NPU.

TRANSACTION PROTOCOL

A special protocol is used for transfers between the NPU and the host. The block portion of this protocol is discussed in section 6. The directives that pass the blocks across the coupler are discussed here.

TRANSFER FUNCTIONS

The coupler's transfer path is half-duplex. This means it is bi-directional, but transmission occurs in only one direction at a time. Both the host and 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 can 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.

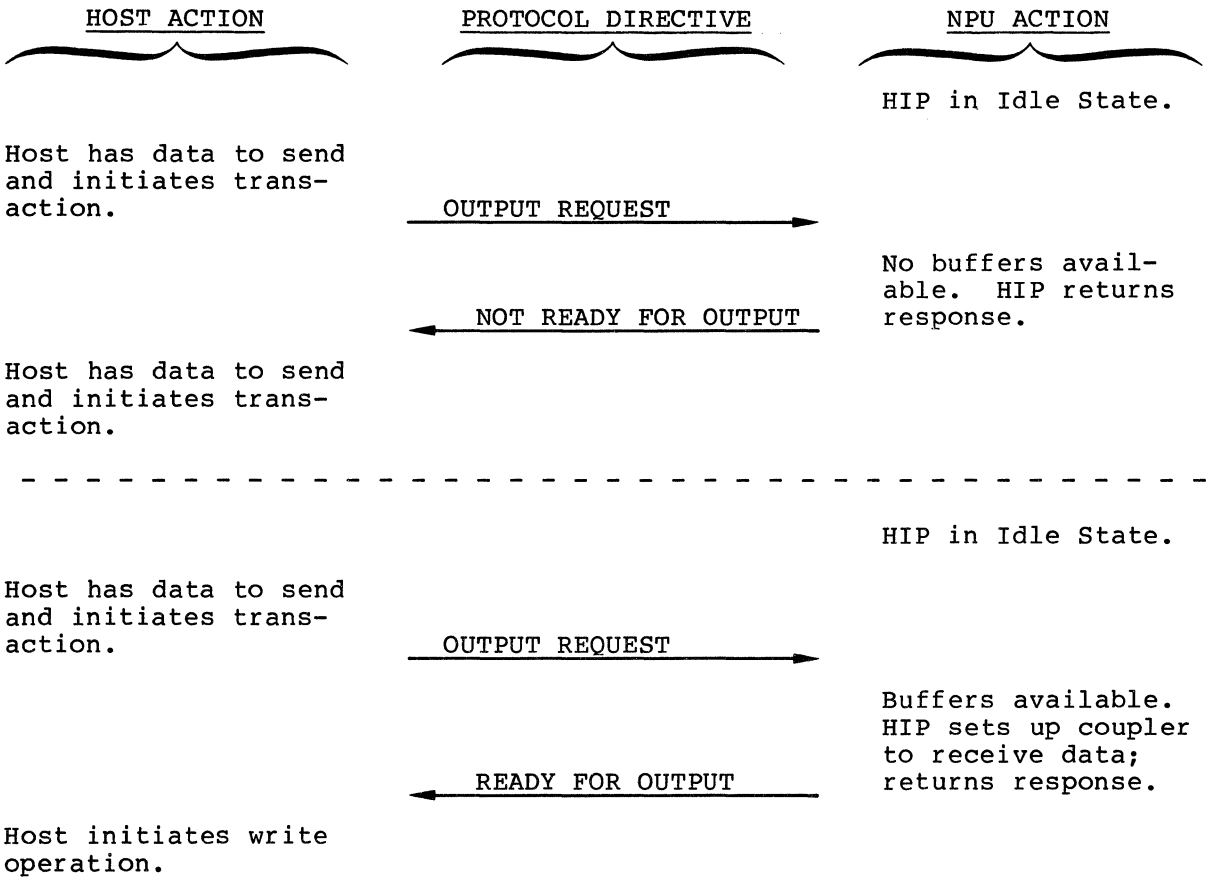
TRANSFER INITIATION

Upline data transfers are initiated by the HIP when the CCI 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



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

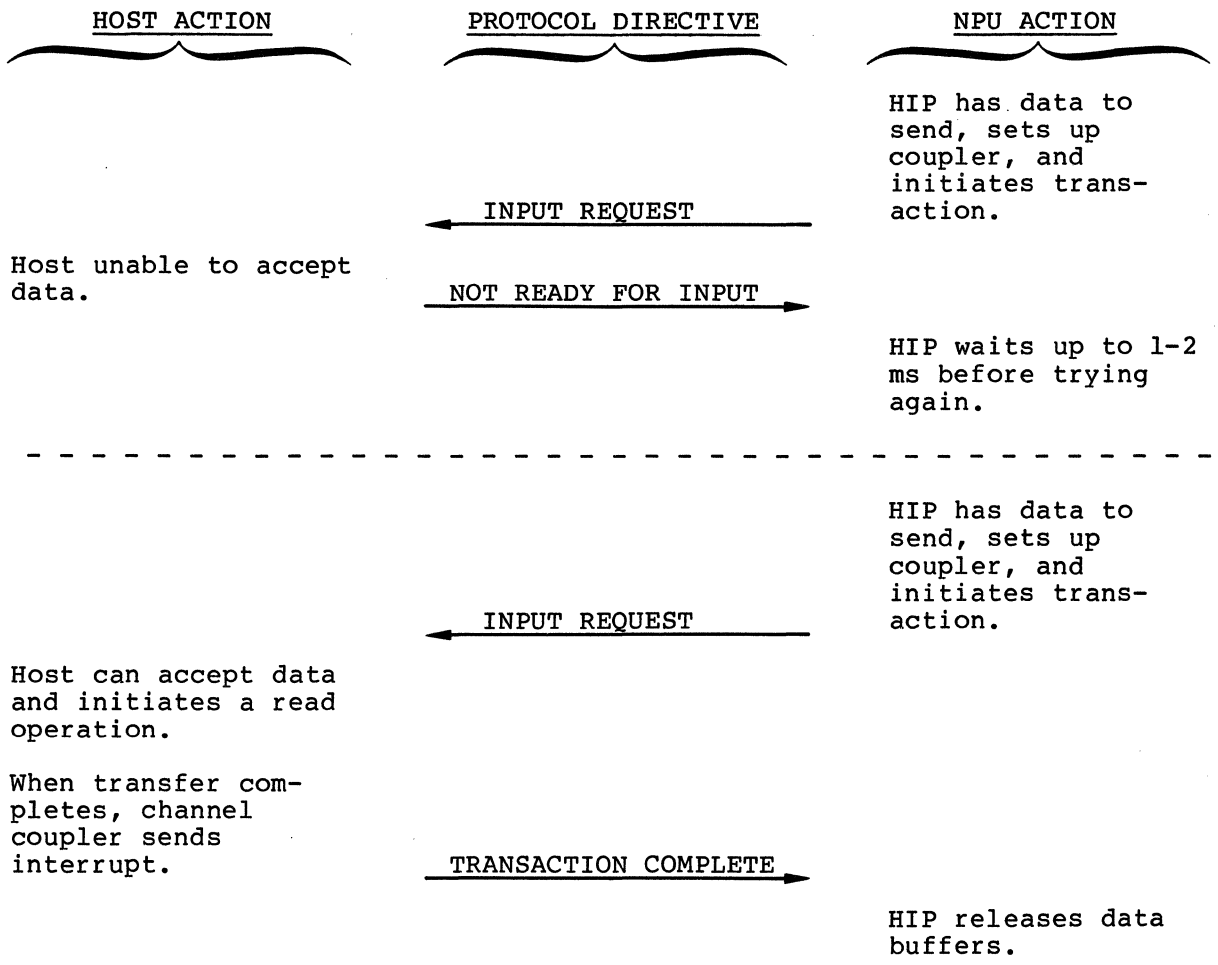


Figure 7-1. Coupler I/O Transactions (Sheet 2 of 2)

INPUT/OUTPUT TRANSACTION CONTENTION

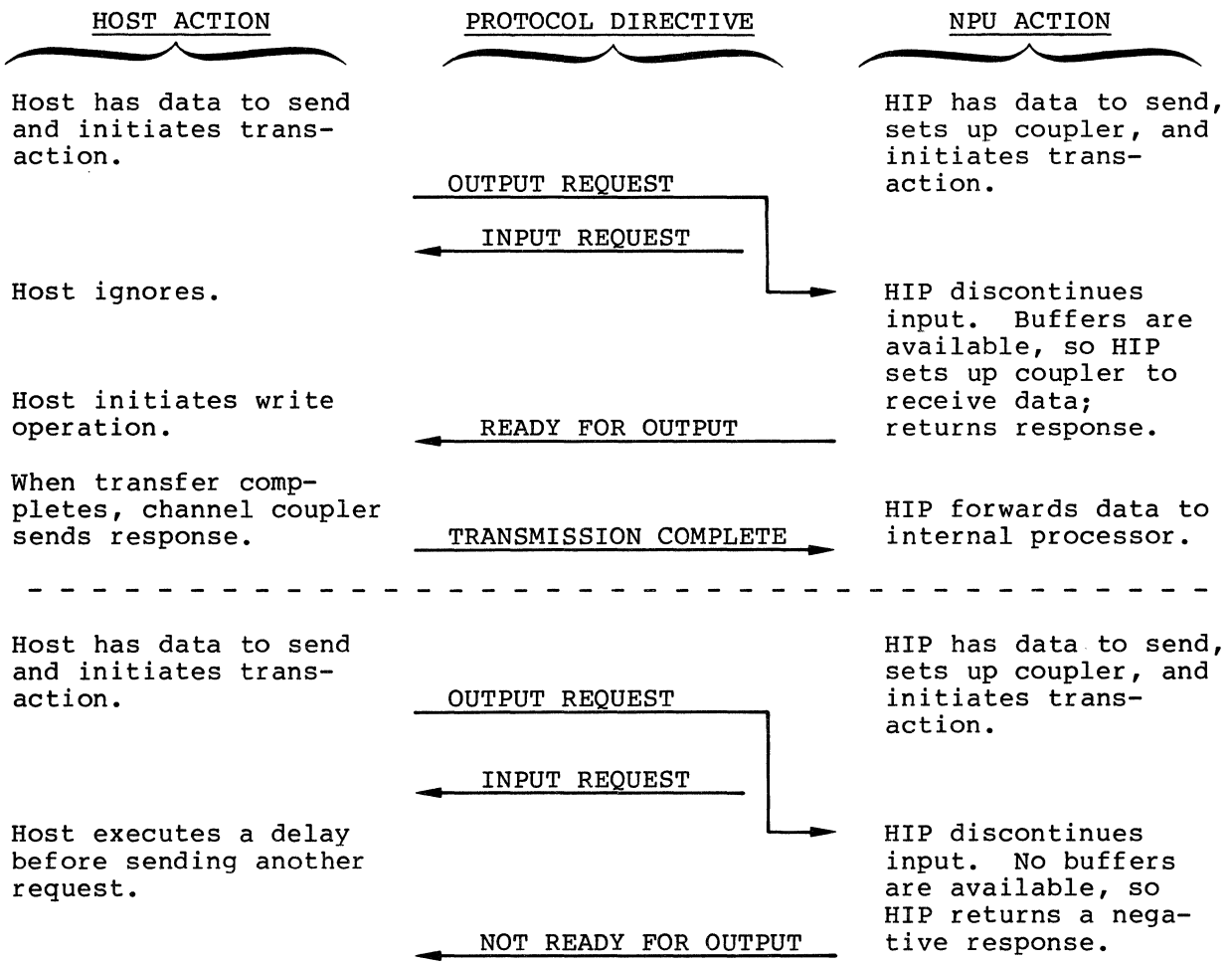


Figure 7-2. I/O Transaction Contention at the Coupler

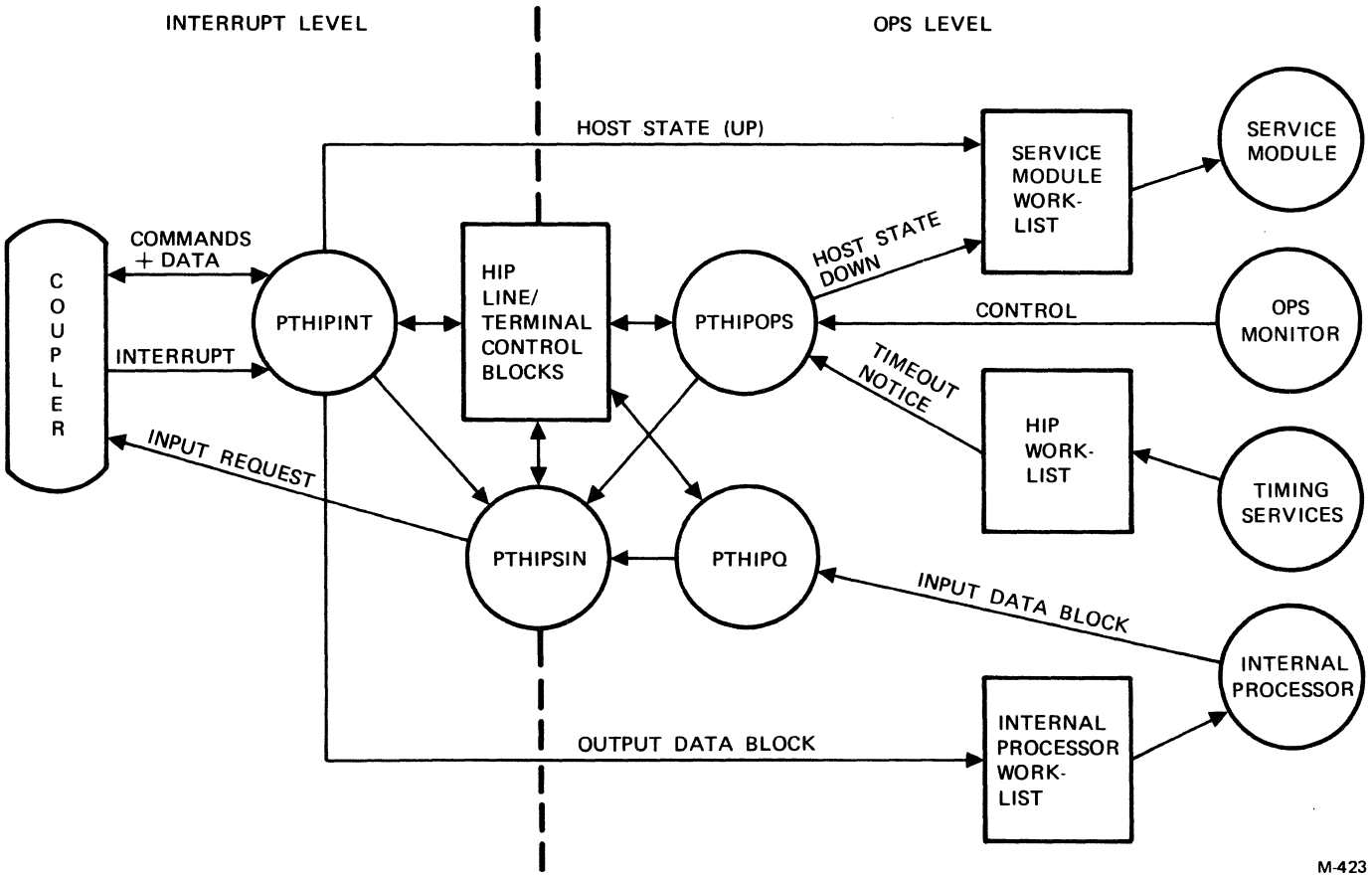


Figure 7-3. OPS and Interrupt Levels for the HIP

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:

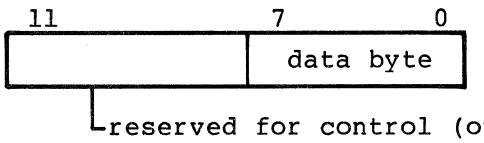
- For recoverable errors, the HIP retries the transfer. The HIP provides an unlimited number of retries to accomplish the transfer. However, in practice the number of retries is limited by the host stopping the transfer or stopping the NPU and reloading the CCI. The recoverable errors are data parity 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 that 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.

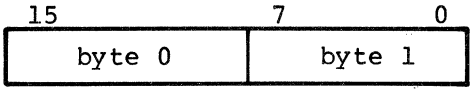
When an error is detected during a downline transfer, the HIP discards the data associated with the transfer, and returns to the idle state.

HOST/NPU WORD 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 8-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 that constitutes the host/NPU interface. A PPU can interface to one or two couplers, but each coupler must connect to a 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 involve 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.

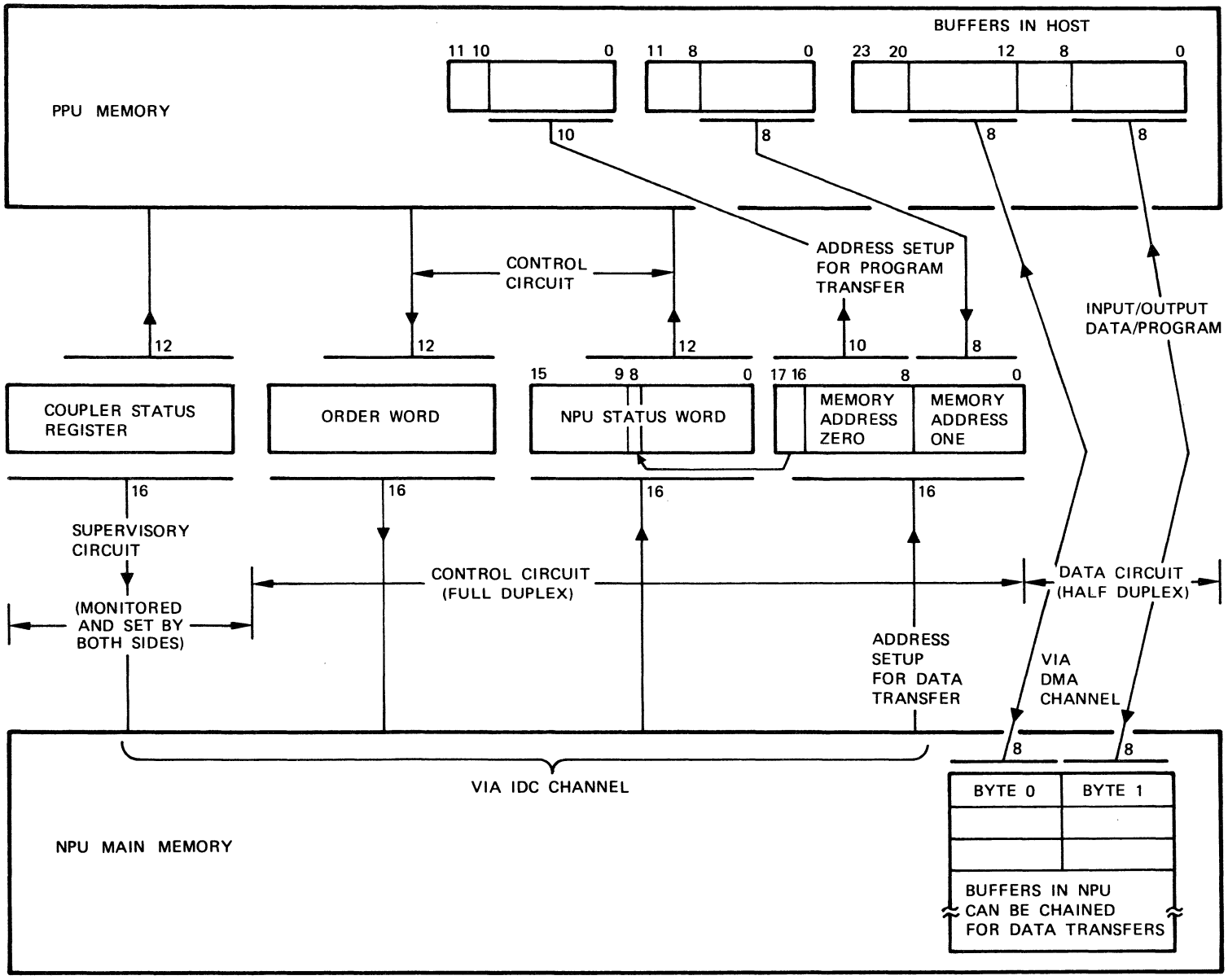


Figure 7-4. Coupler Registers

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 is automatically available in the supervisory circuit.

PROGRAMMING THE COUPLER BY USE OF FUNCTION CODES

The coupler can 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 (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 three 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

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 order-word	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	†
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		

TABLE 7-1. COUPLER STATUS REGISTER BIT ASSIGNMENT (Contd)

Bit Number	I/A	Flag Name	SET Condition	RESET Condition
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 (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:

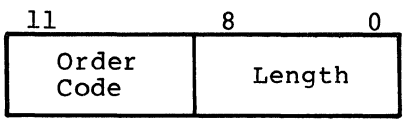
- Processing single word (control/status) function.
- Processing block transfers, for control or message processing purposes.

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.

TABLE 7-2. ORDERWORD REGISTER CODES



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	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
12	Input available: batch block PRUB	Data transfer
13	Input available, 256 bytes (non-PRUB)	Data transfer
14	Input available, 256 bytes (non-PRUB)	Data transfer

TABLE 7-4. ADDRESS REGISTER CODE

Bit 16 (first word)	Bits 15 - 8	Bits 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.

Note that the NPU accesses the orderword and the NPU status word over the internal data channel (IDC).

LOAD/DUMP NPU

Load/dump transfers use the direct memory access channel.

To load or dump the main memory of the NPU, the PPU must first specify a starting location by writing memory address zero and memory address one. The HIP then performs successive data transfers. The first pair of PPU words transferred corresponds to the contents of the specified NPU main memory address. The NPU memory address register is automatically incremented by one, so that successive word pair transfers correspond to the contents of successively higher-numbered NPU main memory locations. The memory load or dump is terminated when the PPU sets the channel inactive. See the initialization section of this manual for a detailed description of dumping and loading an NPU.

The PPU transfers an even number of PPU words. The first word of a pair of words transferred by the PPU corresponds to bits 15 through 8 of the NPU word (byte 0). The low order eight bits of the second word of the pair transferred by the PPU corresponds to bits 7 through 0 of the NPU word (byte 1). The high order four bits of the PPU words are not transferred to the NPU. When transferring from the NPU, the coupler sets the high order four bits of the PPU words to zero.

After loading, the PPU reads back the NPU main memory contents to verify loading of each module prior to issuing the start NPU code. If the load is not verified, the PPU retries loading three times before an alarm message is sent to the network operator.

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.

Load/dump and multiple character data transfer (described below) take place at a maximum instantaneous rate of one PPU word per microsecond. The actual instantaneous rate may be lower as transfers to or from NPU memory may cause direct memory access contention problems; however, such delays are unlikely to exceed one or two microseconds per character, and happen infrequently.

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 can initiate the operation. When both have completed the setup, the transfer takes place.

The PPU sends a function to the coupler, either to input data or to output data. During an output operation the PPU can not directly determine if the NPU has set up its side of the coupler to transfer the data. The determination is accomplished by the preceding comunications 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, 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, PPU 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 flag 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 NPU input data. To limit the frequency of output-request-driven coupler interrupts to the NPU during this data regulation period, a host 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.

Host Failure and Recovery

A special case of regulation occurs when the host fails and when it recovers.

When the NPU software determines that communications 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 communications across the coupler have 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 retransmit 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.

Normally, the NPU accepts all input offered by the PPU. When buffer availability levels drop below predefined 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 at 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.

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 1-second duration generates a periodic idle status, made available to the PPU when no traffic is in progress. The PPU deadman timer provides a 10-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-second 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 explicity 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.

INTERFACE PROTOCOL SEQUENCES

Figures 7-5 and 7-6 show the interface protocol sequences as viewed from the host and from the NPU respectively.

The principal features of the protocol detailed by the flowcharts 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.

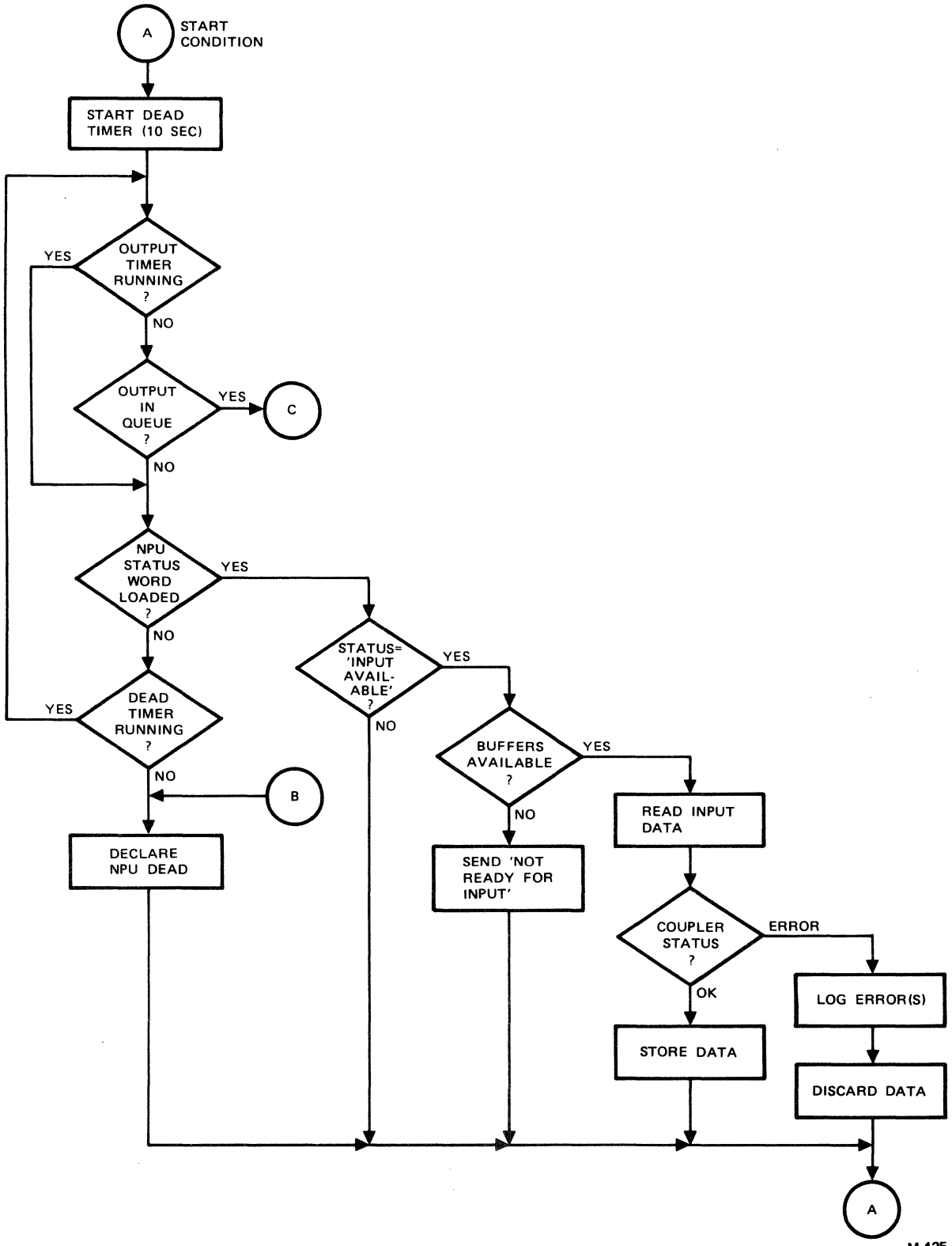


Figure 7-5. Host Interface Protocol Sequence, Host Side (Sheet 1 of 2)

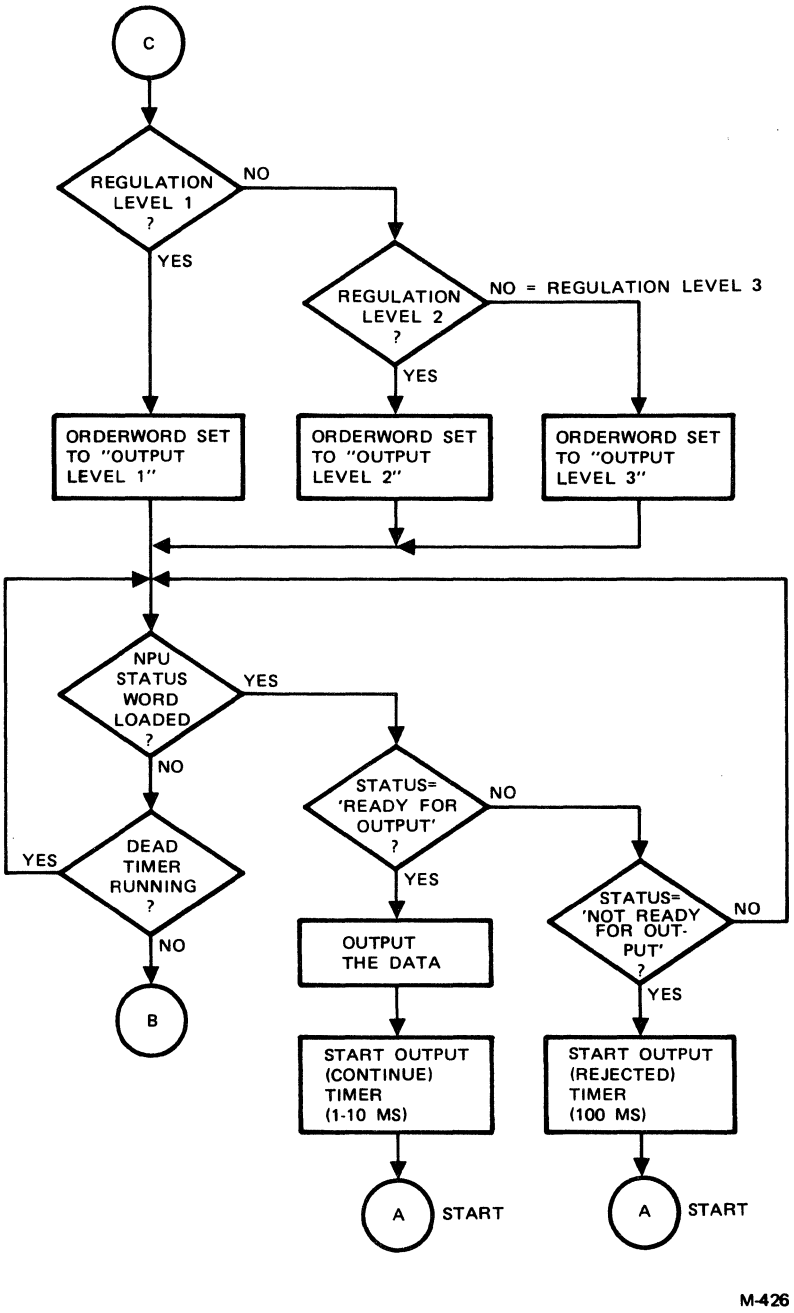


Figure 7-5. Host Interface Protocol Sequence, Host Side (Sheet 2 of 2)

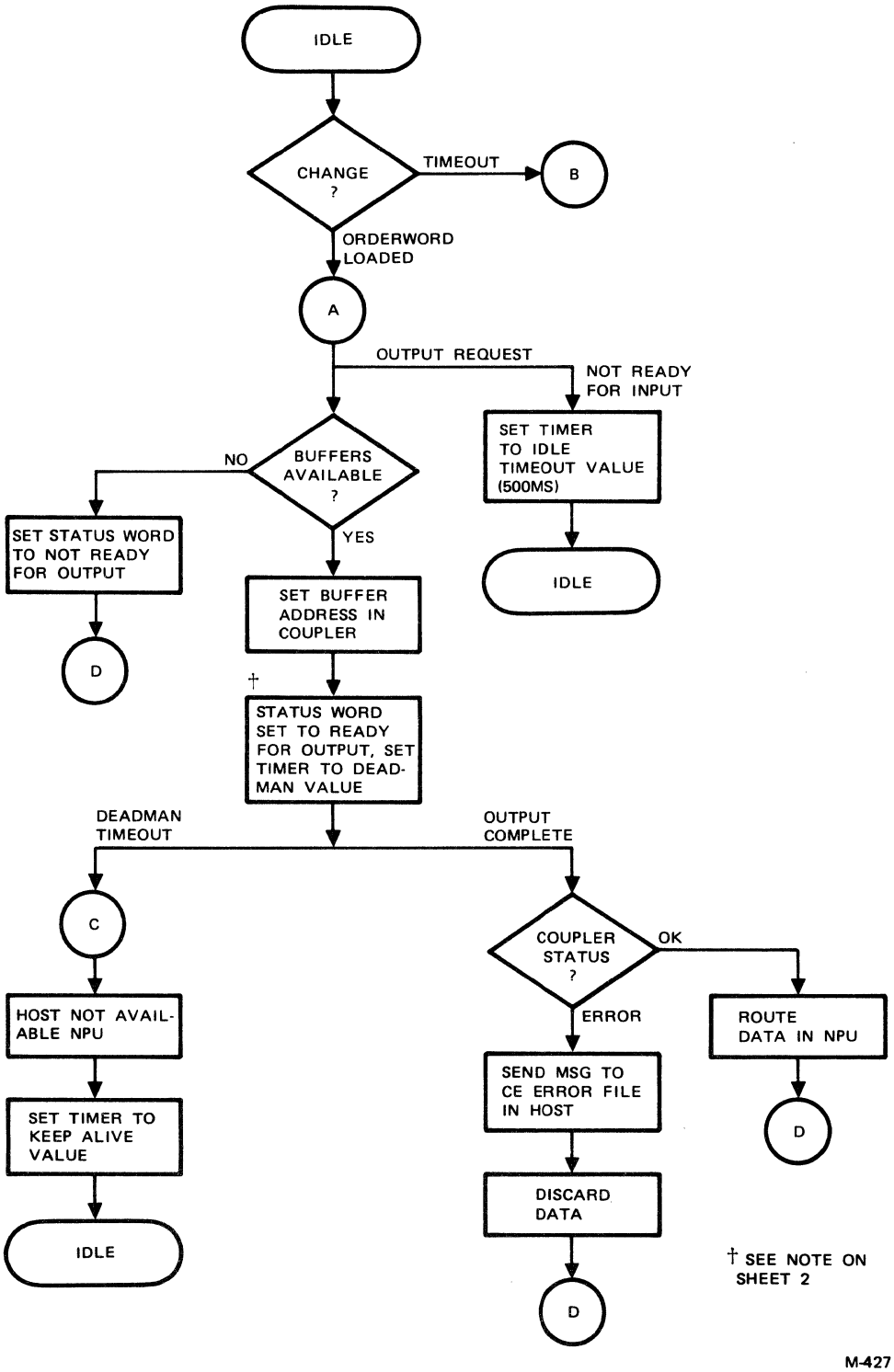


Figure 7-6. Host Interface Protocol Sequence, NPU Side (Sheet 1 of 2)

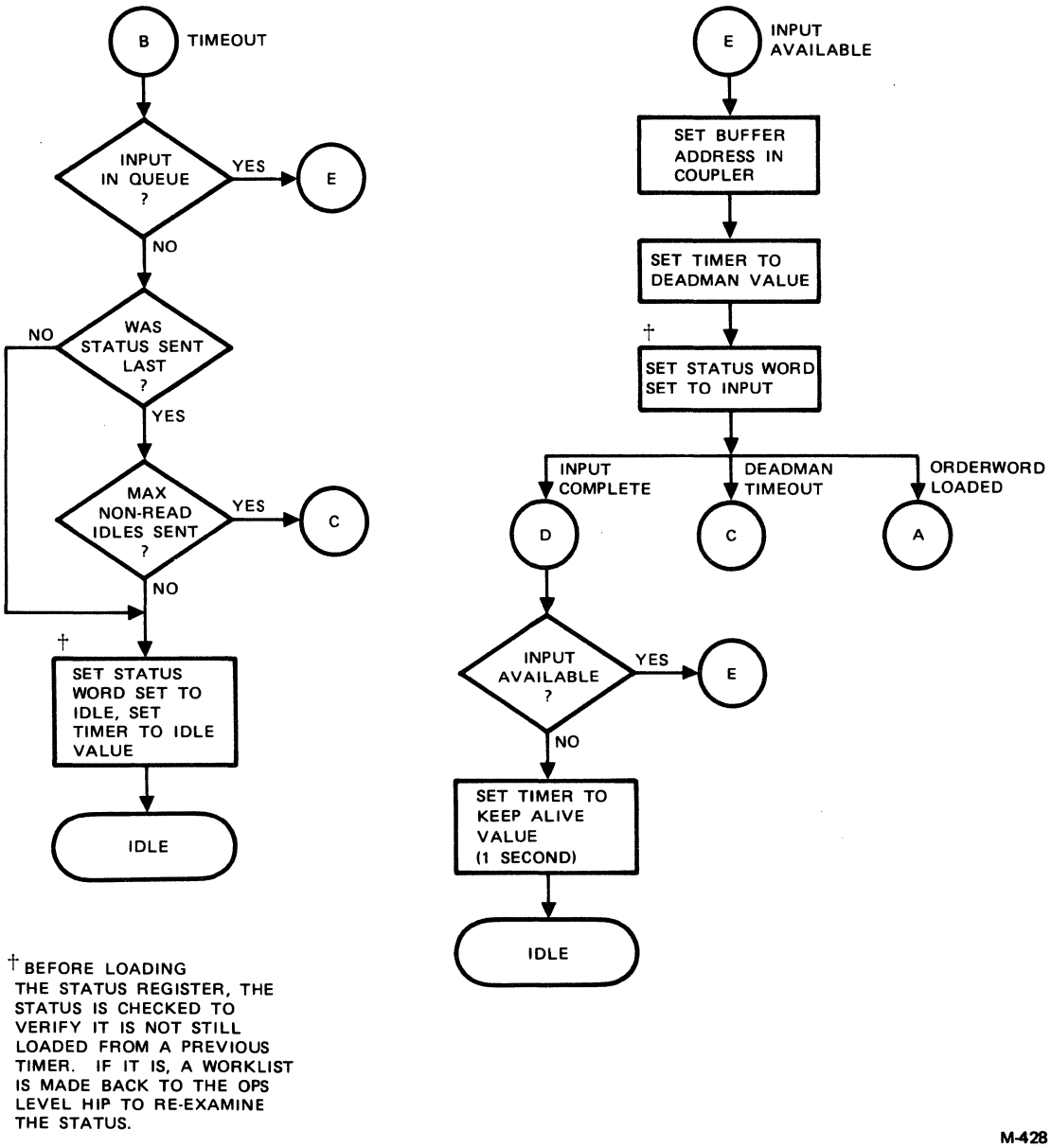


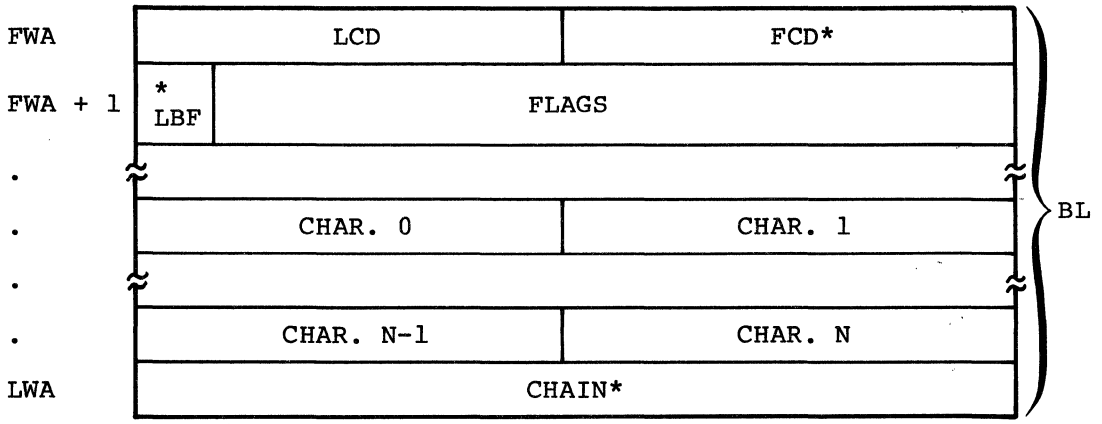
Figure 7-6. Host Interface Protocol Sequence, NPU Side (Sheet 2 of 2)

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 x 7

FCD = First character displacement (relative to FWA) 4 FCD 253

FLAGS = Bit indicators that 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 LCD 253, BL LCD/2 + 1

LWA = Last word address of buffer LWA = FWA + BL - 1

CHAIN = FWA of next data buffer (can 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			Start output		Send idle inquiry
IDLE	CE=Spurious interrupt	CE=Spurious interrupt	(AOPT3) Invalid orderword ⟶ Halt	CE=Spurious interrupt	
AOPT1			Start output or not Ready for input		CK for idle response (deadman timeout)
Idle Inquiry Sent	CE=Spurious interrupt	CE=Spurious interrupt	Invalid orderword ⟶ Halt	CE=Spurious interrupt	Send idle inquiry
AOPT2	NORMAL INPUT COMPLETION	CE=Transfer term by PPU	Terminate input, Start output	CE=Chain address zero, Release input block	Host down to SVC module, Requeue input message
Input Completion		Release input block	(AOPT3) Invalid orderword ⟶ Halt		
AOPT4	NORMAL OUTPUT COMPLETION	CE=Transfer term by PPU	CE=End-of-operation missing	System halt (J0CHAIN)	Host down to SVC module, Release output buffers
Output Completion		Release output buffers	Release output buffers, invalid orderword ⟶ Halt		
AOPT5 AOPT6 Delay	No action	No action	No action	No action	No action

BINARY SYNCHRONOUS COMMUNICATIONS (BSC) TIP

The Binary Synchronous Communications (BSC) TIP provides for the interchange of data between an application program in the host computer and a remote 2780, 3780, or compatible batch terminal. The line protocol used is BSC, operating point-to-point on a dedicated or dial-up line.

Each BSC terminal consists of a card reader and a line printer. A card punch is optional. Remote batch operation with the terminal provides for support of the standard INTERCOM remote batch features and commands, with the minor extension described later.

The operational procedures for submitting remote batch jobs and the return of generated print or punch files follows the rules described in appendix F. The INTERCOM remote batch command set is available to a user at the terminal. The commands are entered from pre-prepared cards at the card reader. One command per card is allowed. Messages to the terminal operator and other unsolicited diagnostic/error messages are directed to the printer. Interactive message output is followed by a form feed to position the message for reading. Interactive message output occurs only at input and output file boundaries.

For 2780 and 3780 terminal input and output data the entire file is transferred before another data transfer can begin; that is, blocks of input and output files cannot be interleaved.

At any time, the TIP must resolve which stream is to be active. Four (or optionally five) streams are managed to or from the two (optionally three) devices on a terminal. The five possible streams and the associated devices are as follows:

Stream	Device
Interactive card input	Card reader
Interactive card input	Line printer
Batch card input	Card reader
Batch card output	Line printer
Batch punch output	Card punch

After the terminal operator dials up the terminal, the host sends an INTERCOM banner message downline, using the interactive output stream. Input messages from the terminal's card reader can then be accepted. These are treated input. For dedicated lines, input is accepted immediately after the terminal is configured (after the banner message is output).

All input is treated as interactive input until a start input CMD block on the batch connection is sent from INTERCOM to the NPU. (INTERCOM generates the start batch input CMD block immediately upon receipt of a READ or READ FILE NAME message from the terminal.) After receipt of the start input CMD block, all subsequent input is assumed to be batch input until EOT is received from the terminal. At this time, the TIP reverts to processing input as interactive input until the next start input CMD block is sent downline from the host.

NOTE

Stream control CMD blocks are defined in section 6.

Interactive output is generated by the host and sent to the TIP. All interactive output messages in queue are delivered to the line printer immediately following the completion of any active batch input or output file (EOT received or transmitted) or immediately following EOT received on the interactive input stream.

After outputting EOT to the terminal, a 3-second delay is initiated by the TIP. This allows any waiting input message to be sent upline before another output file can be started.

Input takes precedence when contention exists between input and output batch data streams. An output printer stream takes precedence when contention exists between two batch output streams. This is the result of the printer's TCB being the first output device TCB in the terminal's TCB chain.

OPERATIONAL FEATURES

It can generally be assumed that INTERCOM interactive and remote batch features are functionally the same as the standard INTERCOM 4.5 release. Special job stream command formats and terminal operations are described below.

REMOTE BATCH FACILITIES

The following remote batch facilities protocols apply.

EOR/EOI

Only 7/8/9 and 6/7/9 end-of-record (EOR) and end-of-information (EOI) cards are required by CDC terminals. Since IBM readers treat multipunches in columns 1 through 7 as errors, EOI cards for type 2780 or 3780 terminals are punched as /*EOI in columns one through five. An EOR card is represented as /*EOR in columns one through five.

A level number placed in columns 2 and 3 of the EOR card is supported. For punch output, the level number is punched in columns 2 and 3 of the EOR card.

Binary Cards

Binary card deck input or output is not supported by any terminal.

026/029 Codes

A 26 or 29 punched in columns 79 and 80 of the JOB CARD or EOR cards changes the input code translation for 2780 and 3780 terminals. Output for punched cards always uses 026 Code.

Transparent Data

Transparent data assumes an 8-bit byte. Transparent input can be defined by either a READ, FILENAME, or MODE command, where mode specifies transparent, or by punching TR in columns 79 and 90 of the EOR card. Transparent mode ends on receipt of logical EOI (last card transmitted). /*EOI is not detected in the transparent mode.

Transparent input data is written to rotating mass storage and is stored 8-bits in 12-bit right-justified characters, with five characters per CYBER word. Neither code translation nor character expansion is performed.

Transparent output is created by using parameters on the ROUTE control card. Transparent output is selected to be deliverd to the terminal using the DEFINE terminal command. (See INTERCOM reference manual.) Transparent output files are delivered to the terminal without performing any character compression, code translation, carriage control conversions, print line, or card blocking.

Carriage Control

Printer carriage control for batch output terminal printers is controlled by the first character of each line. The action taken by the 2780 and 3780 terminals in response to INTERCOM control characters is summarized in table 8-1.

The suppress print file carriage control command is supported. The horizontal and vertical tab features are not supported.

Interactive Carriage Control

Standard INTERCOM carriage control characters are supported. These control characters are translated to equivalents when interactive output is being delivered to a 2780 or 3780 line printer. Control characters and their equivalents are shown in table 8-2.

Punch Files

Punch files are supported for the 2780 and 3780 terminals if a punch device is present. Punch files are specified by setting the forms code. INTERCOM recognizes the forms code and identifies the file to the 255x by the connection number.

Output files to the punch are proceeded by a banner message (generated by INTERCOM) which generates a lace file separator card record with nulls in columns 1 through 70 and the job name in columns 71 through 80. The card record must not contain a carriage control character; it can contain from 1 to 80 characters. Short cards can optionally be punched with the BSC record separator in the last column for 2780 and 3780 terminals, by selection within the INTERCOM DEFINE command.

Transparent mode output files can be sent to the punch. In this case the user is responsible for ensuring that all 80 columns are present.

TABLE 8-1. SUMMARY OF BATCH CARRIAGE CONTROL SYMBOLS

INTERCOM Control Character	Terminal Type	
	2780	3780
1	Space 1	New Page
+	Space 1	No space
0	Space 1	Space 2
-	Space 1	Space 3
⌀	Space 1	Space 1
All others	Space 1	Space 1

TABLE 8-2. SUMMARY OF INTERACTIVE CARRIAGE CONTROL SYMBOLS

INTERCOM Control Character	2780/3780 Terminals	
	Before	After
1	Skip to Top of Page	Space 1
*	Skip to Top of Page	Space 1
+		2780-Space 1 3780-No Space
0	Space 1	
-	Space 2	Space 1
Blank		Space 1

Compression/Expansion

Compressed data from the terminal is expanded to the standard SCOPE file format before the record is written to disc. Conversely, data read from an output file is compressed before transmission to the terminal. Both compression and expansion are performed by the TIP. The methods differ for the two terminal types:

- 2780: Trailing blanks for input nontransparent card data are suppressed. The end-of-card is indicated using the standard SCOPE file format. Trailing blanks are not transmitted to the printer or punch except in the transparent mode.

- 3780: Nontrailing blanks are expanded for input nontransparent card data. Trailing blanks are suppressed and the end-of-card is indicated the standard SCOPE file format. Output nontransparent data has all blanks compressed whenever possible.

TERMINAL FEATURES

The following features of the 2780/3780 devices are supported:

Feature	2780	3780
Character set	EBCDIC	EBCDIC
Horizontal format control	No	No
EBCDIC transparent mode	Yes	Yes
Multiple record feature	Yes	N/A
Space compression/expansion	N/A	Yes
Print line width	80-105	80-105
Punch/component selection	Yes	Yes
Line speeds	2000-9600 bps	2000-9600 bps
Printer character set	EBCDIC 63	EBCDIC 63
Multi-point	No	No
Terminal ID	Accepted but not checked	Accepted but not checked
Conversational mode	Not used	Not used
Processor interrupt	Not used	Not used
Multiple cards in transparent mode	Yes	Yes

OPERATIONAL CHARACTERISTICS

Each terminal can be operated in the nontransparent mode. If the terminal supports the feature, the terminal can also operate in the transparent mode. The operational characteristics of each terminal for transparent and for nontransparent modes are described below.

2780 Input Nontransparent Terminal Mode

Commands are entered one per card. Commands can be stacked in the card reader only if an ETX is punched as the last column of each command. Commands can be input without an ETX punched in the last column only if a single command is placed in the reader and if the EOF toggle switch on the terminal is ON.

The last command entered before an input file or job deck must be either a READ or a READ, FILE NAME. The job deck or input file can be stacked directly behind the READ or READ, FILE NAME command if the ETX is punched in the last column. If the ETX is not included, the command must be entered separately from the input file or job.

The first card of batch input is assumed to be a job card. It is interpreted by the TIP. Batch input can be terminated in one of three ways:

- A /*EOI in columns 1 through 7
- An ETX in the last column of the last input card (can be first column of a separate card)
- Input of the last card with the EOF toggle switch ON.

Note that if ETX is used to terminate a job, another job cannot be stacked directly as the TIP treats the next input after an ETX/EOT as interactive input.

The TIP does not distinguish between batch input initiated by a READ or batch input initiated by a READ, FILE NAME. However, the following rules apply for INTERCOM even though no special checks are made in the TIP. If the data transfer was initiated by a READ command, multiple jobs can be stacked in the reader with each job terminated by /*EOI. Multiple /*EOI cards between jobs are discarded by the TIP. The first non/*EOI card is assumed to be the job card for the subsequent job. If the data transfer was initiated by a READ, FILE NAME command, only one file can be stacked in the reader. Subsequent input must be initiated by a new READ or READ, FILE NAME command.

2780 Input Transparent Terminal Mode

If the 2780 terminal has the transparent option; data can be input with the transparent switch ON.

Each command must be entered separately with the EOF toggle switch ON, as ETX is not recognized in this mode.

Each card input causes a full 80 characters to be transferred to the NPU and each card is transferred upline to the host as a separate transmission block. Operation in this mode is less efficient than in the nontransparent mode.

All other characteristics are the same as in the nontransparent terminal mode.

NOTE

This mode should not be confused with the transparent data feature initiated by a TR optional parameter in the READ, FILE NAME command or by placement of a TR in columns 79 and 80 of the EOR card of a card deck. These modes are described below under the input transparent data mode heading.

3780 Input Nontransparent Terminal Mode

The operational characteristics of the 3780 terminal with respect to card input are the same as those described for the 2780 except:

- EXT can not be punched in a command card or as the last card of an input job or file to terminate input.
- Each command card (interactive input) must be input separately with the EOF toggle switch ON.
- Multiple jobs can be stacked in the reader separated by /*EOI cards, but the last card must be input with the EOF toggle switch ON.

3780 Input Transparent Terminal Mode

If the 3780 terminal has the transparent mode option, data can be input with the transparent switch ON.

The operational characteristics in this mode are identical to the nontransparent terminal mode. However, there are differences in line efficiency and number of blocks transmitted. Each card input causes all 80 characters to be transmitted across the communication line; that is, trailing blanks at the end-of-card are not suppressed, and embedded strings of blanks and zeros are not compressed, as in the nontransparent mode.

Input Transparent Data Mode, 2780 and 3780

The TIP provides another mode of inputting the batch data for both 2780 and 3780 terminals, if the data is being sent to the host without translation. This mode of input is specified only when the terminal is operating in the transparent mode. Three methods are provided to enter the transparent data mode:

- For local input files, an optional parameter TR is included as part of the READ, FILE NAME command.
- TR is included in columns 79 and 80 of the INTERCOM job card for normal input jobs.
- TR is included in columns 79 and 80 of the INTERCOM EOR card.

When TR is specified, none of the following data is translated. The data is stored as 8-bit characters. Since no EOR (7-8-9 punch) is recognized in this mode, a file must be terminated using the EOF toggle switch.

Transparent 8-bit characters are stored in the physical record unit block (PRUB) without marking the record boundary; such as the 80-column card boundary. A BSC transparent transmission block can contain single records that are terminated by DLE ETB or multiple records with each record separated by an DLE ITB (DLE ITB and DLE ETB are not stored in the PRUB). The 3780 also has an optional feature to input four fixed length 80-character records in each transmission block. This feature is not specifically supported by the TIP. If the feature is used, 320 characters are stored in the PRUB for each transmission block received.

If any case, characters are stored as received in the PRUB blocks without regard to record or transmission block boundaries. Transmission blocks are split across PRUB boundaries and input continues to be stored until ETX is received.

In any case, characters are stored as received in the PRUB blocks without regard to record or transmission block boundaries. Transmission blocks are split across PRUB boundaries and input continues to be stored until ETX is received.

2780 Output Nontransparent Transmission Mode

Output streams can be directed to either the line printer or the terminal card punch, where applicable. The host software determines which device is to receive the output. The output to each device is controlled by separate connections.

The TIP accepts PRUB blocks from the host and converts the data from display code to external EBCDIC code and formats print lines into BSC transmission blocks for output. Each transmission block is made up of multiple print lines (records) where the number of lines is either two or seven, depending on the terminal option defined to the system. In any case, the transmission blocks cannot exceed 400 characters. Print lines are never split across transmission block boundaries.

The first character of each print line in the PRUB block is an INTERCOM carriage control character; it is translated to BSC printer carriage control as defined in table 8-1. The algorithm used for the conversion is shown in table 8-3.

Most carriage control operations performed by the TIP can be suppressed by use of the INTERCOM SUP (suppress) command. If this option is used, the host sets the suppress field in the TCB. The first character of each print line is discarded (no specific carriage control characters are transmitted to the terminal; a space 1 after print occurs automatically).

A P carriage control character followed by an M character activates the standard INTERCOM feature for embedding an operator message (print message) within the print file. For the 2780, the print line following the PM is printed on the line printer, preceded by and followed by a skip to top of page carriage control. Output to the printer then stops and the host is notified of the condition by an upline break CMD block. The TIP then allows interactive input from the card reader. Printing will not resume until commanded by the host.

The printer line width can be defined for a terminal as any value from 50 to 150 characters by use of the INTERCOM DEFINE command after login. The host changes the TIP's terminal line width by sending a reconfigure TCB SM with the appropriate FN/FV pair. (See section 3 and table E-7, BSPGWIDTH.) The default line width is set to 144 characters. Characters in a PRUB block print line, in excess of the PRUB block print line length, are printed on the next line. Trailing blanks for short print lines are not transmitted to the terminal.

TABLE 8-3. 2780 BATCH CARRIAGE CONTROL ACTION

Interactive DBC Code†	Control Character	INTERCOM Batch Carriage Action	TIP Action
0 or 1, 8, 9	1	New Page	Transmits null line with skip to channel 1, then transmits the output line with the automatic space after print.
2, 10	⌀	Space 1	No special action - a space 1 after print occurs automatically.
3, 11	+	No space	No special action - a space 1 after print occurs automatically and cannot be suppressed.
4, 12	0	Space 2	Transmits null line which causes an automatic space, then transmits the output line with an automatic space 1 after print.
5, 13	-	Space 3	Transmits null line with space 2 carriage control character, then transmits the output line which automatically spaces 1 after print.
All Others	All Others	Space 1	No special action - a space 1 after print occurs automatically.

†See section 6, DBC.

Transfer of an output data file to a printer or a punch normally continues either until completion or a failure occurs. A method of interrupting an output stream allows for receiving interactive input commands that might change the disposition of the output stream. The method, called "intervene", may differ operationally for type 2780 and 3780 terminals and emulators of these terminals.

In general, making the printer not ready causes a timeout for approximately 30 seconds. Then input from the card reader is allowed. Making the printer ready before the timeout has expired causes continuation of the output file. In some cases, several print lines will be duplicated. The timeout period is a function of the terminal itself. The period is usually extendable by pressing the PRINTER STOP key again.

In any case, an EOT received from the terminal as a response to an output block causes that output stream to be stopped, allowing input from the card reader.

Card punch output data is processed by the TIP in a manner similar to print output data, except that the first character of each card (BSC record) is treated as data instead of carriage control, and each record must be 80 characters or less in length. Characters within the PRUB record in excess of 80 characters or less in length. Characters within the PRUB record in excess of 80 are punched on the next card. An option is provided to punch an EM character as the last character in each card that contains fewer than 80 characters of data. This option can be specified by an INTERCOM DEFINE command, which causes a reconfigure TCB SM to be sent to the TIP. The default value is set for no EM character (BSEM = 0).

2780 Output Transparent Transmission Mode

Output to a terminal is normally transmitted as nontransparent transmission blocks. If the terminal contains the transparent mode optional feature, data is output in the transparent transmission mode only if the data file being output is specified as transparent. Transparent data files are designated to INTERCOM by parameters within INTERCOM ROUTE command. INTERCOM marks the data file as transparent in the DBC field of each PRUB block of the file.

In transparent mode, the TIP processes transparent output blocks differently than nontransparent blocks. PRUB block characters are output as transmission blocks without code conversion and carriage control transforms. Characters are taken from the PRUB to make up transmission blocks without regard to record markers (FF_{16} character) or to the PRUB boundary. Characters are transferred from the PRUB to the transmission block until the transmission block is full or until the last character of an EOR or EOI PRUB has been transferred. An EOI PRUB terminates the file. An EOR PRUB terminates only the transmission block; the following transmission to the terminal continues with the next PRUB block from the host and a new transmission block.

3780 Output Nontransparent Transmission Mode

Output to the 3780 terminal functionally provides the same capabilities as with 2780 terminals. The internal processing differences are as follows:

- The number of print line records or card records is not limited to 2 or 7. The number of records included in a transmission block is limited only by the transmission block size. This is normally 512 characters, but can be configured to any size. Records cannot be split across transmission block boundaries.
- Output to the 3780 follows the rules for data compression for that terminal; that is, multiple blanks are compressed.
- Short records (print lines or cards) are terminated by the EBCDIC IRS character. Trailing blanks are transmitted if contained in the PRUB record.
- Carriage control for 3780 printer output is designed to be directly compatible with Mode 4 terminal carriage control. The algorithm used by the TIP for conversions from INTERCOM carriage control characters to 3780 carriage control is shown in table 8-4.

TABLE 8-4. 3780 BATCH CARRIAGE CONTROL ACTION

Interactive DBC Code	Control Character	INTERCOM Batch Carriage Action	TIP Action
0, 9	1	New Page	Transmits null line with skip to channel 1, then transmits the line with suppress space after print carriage control.
	⁄	Space 1	Transmits a blank line with space 1 after print carriage control followed by the text line with suppress space carriage control.
4, 12	+	No space	Transmits the line with suppress space after print carriage control.
	0	Space 2	Transmits a blank line with space 2 after print carriage control followed by the text line with suppress space carriage control.
	-	Space 3	Transmits a blank line with space 3 after print carriage control followed by the text line with suppress space carriage control.
All Others	All Others	Space	Transmits null line with skip to channel 1, then transmits the line with suppress space after print carriage control.

DIRECT CALLS TO THE BSC TIP

The BSC TIP can be called by:

- Any other program using a 3-word standard TIP worklist. Worklists are queued to PTIP780. The monitor passes control to the TIP with a single worklist attached. PTIP780 is the principal switch for the TIP. (See BSC TIPTREES, appendix G.) The switching procedure is based on the workcode in the worklist (lower half of word 0 of the worklist). The principal users of this call are as follows:

  Internal processing, to process downline data blocks and commands.

  The service module, to set up line and terminal changes, and so forth.

The TIP's own input state programs, to process special conditions in input blocks, and to continue input processing on the OPS-level.

The multiplex subsystem, to have the TIP process special conditions such as terminal failure or CLA status.

- Internal processing, by switching to the page and address of the BSC text processor (PTTP780). Text processing is done at the same time as the output message is converted from PRUB format to BSC terminal format. Control then returns to internal processing which subsequently calls the TIP with a worklist so that the TIP can prepare the text processed block for output transmission to the terminal/device specified.

- SVM, to build the TCB. This is the direct call to PTTCB780 to finish building the TCB and to chain the TCB to the other TCBs in this line's TCB chain (priority: card reader, then printer, then punch - if any).

Note that there is no multiplex level 2 call to this TIP. The input state programs are written to call the OPS-level TIP, if any major error processing is to be performed by the TIP.

DIRECT CALLS FROM THE BSC TIP

The BSC TIP uses the following routines:

- Buffer handlers: PBRELZRO, PBRELCHN, PBREL1BF, and PBGET1BF to release or to assign buffers. PBCLR is used to clear space in assigned buffers.

- PTREGL is called to determine whether input is to be accepted, or whether data from the terminal is to be rejected. All four regulation checks are used. See section 6, PTREGL.

- PTTPINF is used to called the firmware text processing programs. Control returns directly to the TIP after text processing is completed.

- PBPROPOI is called to set up output block parameters where the TIP is searching for the next block to process.

- PBPOPOI is called to acknowledge the block that has been sent to the terminal.

- PBPIPOI is called to prepare the upline block for the host. This relieves the TIP of the conversion to PRUB formatting task.

- PBCOIN is called to cause the command driver to prepare a BSC line for input/output for transmitting a message or for shutting down a line. The BSC protocol requires sending acknowledgment messages to the terminal.

- PTCOMMAND is called to generate and send data stream control CMD blocks to the host. See CMD block formats in section 6.

- BLTIMTBL is called to set up timed functions and to clear timeout counters when the expected event occurs within the allowable period.

- PBLSPUT is called to prepare worklist calls to the service module (for instance, to report that a TCB has been deleted), or to call itself after marking a line down.

ERROR PROCESSING

Lower level error processing takes place on the firmware level. If a problem can be corrected on that level, the OPS-level TIP does not receive notification that a problem error existed. If this cannot be done, a worklist entry is made. The SVM also sends error type entries. The error processing worklists are:

- Disable or disconnect line (from the SVM) or hardware error (from the multiplex subsystem). BRINGLINEDOWN notifies the command driver to deactivate the line, and the service module that the line has been deactivated. Any transmission in progress is aborted.

- Timeouts. BRINGLINEDOWN deactivates the line and the TIP checks for any task that can be started on another line.

- Received bad or error data blocks during input. The host is notified that the input has been stopped and the terminal is sent an end-of-transmission (EOT) message.

- Received NAK or WACK blocks indicated an error condition. A variety of actions are taken depending on the situation. An EOT may be sent to the terminal to abort the transmission.

AUTORECOGNITION

The BSC protocol uses autorecognition as a means of showing that the terminal is operational. A worklist entry is made to SVM indicating this. This routine also handles the preliminary autorecognition for the HASP TIP. A worklist entry is made for the HASP TIP.

ASYNCHRONOUS (TTY) TIP 9

The Terminal Interface Package (TIP) for Mode 3 terminals provides a set of procedures for the interchange of interactive data between the host processor and Mode 3 terminals. The Mode 3 terminals can be Teletype (TTY) or teletypewriter compatible terminals. These terminals customarily use ASCII code.

The TTY TIP supports single terminals switched or dedicated asynchronous lines at speeds of 110, 150, 300, 600, 1200, 2400, 4800, and 9600 baud. The lines are considered to be half duplex; that is, the TIP can be transmitting or receiving on a given line, but not doing both simultaneously. The TIP can be considered to be in transparent mode at all times. No code translation or parity check is performed on data characters that are input from the terminal. Data characters that are output to the terminal are output as received from the host, with the exception of the character parity bit which is complemented, when necessary, so that all output characters have even parity.

All characters (both input and output) are passed between the host and terminal in full 8-bit form, without code translation or parity generation and checking. An input message is sent in one or more blocks. The maximum size of an input BLK or MSG block is controlled by a program build parameter (up to 2047 characters). If the message length exceeds the maximum block size parameter, all but the last block are BLK blocks. The last block is a MSG block. Where message length is less than maximum block size, the single block sent is a MSG block.

OPERATING MODES

The TIP operates in either interactive (keyboard) mode, or tape mode (paper tape reader/punch or magnetic tape cassette).

INTERACTIVE MODE

In interactive mode, the TIP interfaces the network to a Teletype/TTY compatible device for either input or output. The interactive mode processes input from a TTY keyboard or processes block mode characters from a TTY compatible device where the characters are received at line speed.

In interactive mode, the TIP operates in half-duplex fashion with three basic states: idle, input, and output. Table 9-1 shows the events that cause a change of state. Note that input messages are one logical line in length; that is, a carriage return places the TIP in idle state. Output messages can be any length that is acceptable to the terminal. For this reason, stream control commands are not needed for interactive mode messages.

TABLE 9-1. TIP STATE TRANSITIONS, INTERACTIVE MODE

Current State	Stimulus	New State	Other Actions
IDLE	Input of any character except CR, LF, or pad (FF or $7F_{16}$)	INPUT	TIP (using output state programs) accepts the message unless conditions for regulation exist.
IDLE	Input of CR, LF, or pad	IDLE	Character is discarded (LF is echoed for a single CR).
INPUT	End of message, CR	IDLE	TIP sends LF to terminal if no other character is input within 100 ms.
INPUT	LF + 100 ms without another input	IDLE	TIP outputs a CR.
INPUT	Any character but regulation conditions exist	OUTPUT, THEN IDLE	TIP breaks input message, discards characters, notifies the multiplex subsystem.
IDLE	Output message queued to TCB	OUTPUT	
OUTPUT	No more messages queued to any TIP TCB	IDLE	
OUTPUT	Input of any character	INPUT	1. Character is discarded. 2. Current output message is terminated. Message block is requeued at top of TCB's output queue.

TAPE MODE

In tape mode, the TIP interfaces the network to a tape reader (paper tape or tape cassette). Table 9-2 shows the events that cause the TIP to change between input, idle, and output state in tape mode. Note that the tape mode requires two stages. The first stage places the TIP in tape mode so that the TIP can send the X-ON messages to start tape motion on the reader/punch or magnetic tape. The second stage reads the messages. An X-OFF character always stops tape mode. This also places the TIP in idle state. When a block size of 256 is reached, the next CR causes a BLK to be sent to the host. This prevents flooding the NPU with input data.

TABLE 9-2. TIP STATE TRANSITIONS, TAPE MODE

Current State	Stimulus	New State	Other Actions
IDLE	Start input CMD on tape connection	INPUT (tape mode)	1. TIP sends X-ON (11_{16}) to tape reader. 2. CR delimits message blocks.
INPUT (tape mode)	X-OFF	IDLE	TIP now ready for keyboard data.
INPUT (tape mode)	Any character but regulation conditions exist	OUTPUT then IDLE	Message is rejected. Multiplex subsystem is notified of regulation.
IDLE	Output message queued to a punch TCB	OUTPUT	TIP delivers message.
OUTPUT	X-OFF in output block	IDLE	TIP checks for more queued data. Another tape or interaction message queued for output can immediately place TIP in mode again.

Two block stream control messages are used in tape mode (section 6, CMD blocks):

- Start input. This downline message from the host has the format:

DN	SN	CN	BT = 4	PFC = 01	SFC

DN - NPU ID
SN - Host ID
CN - Connection for this TCB
BT - 4 (CMD)
PFC = 01 - Start input, send X-ON to terminal to start tape motion
SFC - Not used

- Input stopped. This upline message results from an X-OFF byte in the incoming message. Format is:

DN	SN	CN	BT = 4	PFC = 03	SFC = 00

DN - Host ID
SN - NPU ID
CN - Connection for this TCB
BT - 4 (CMD)
PFC - 03 - stopped input
SFC - 00 - normal reason: X-OFF detected.

CARRIAGE CONTROL FOR OUTPUT MESSAGES

The carriage control for interactive TTY terminals is defined by the data block clarifier (DBC) character received from the host. The DBC is the fifth byte of a data block. (See section 6.) Only the least significant four bits of the DBC are used by the TIP to determine the carriage control character sequence. The carriage control character sequence is sent to the terminal prior to the first character in the output message block.

The carriage control commands are shown in table 9-3.

The number of line feed characters in the character sequences is decremented by one (described in table 9-3) if the following occurs:

- The line feed count in the sequence is nonzero.
- A line feed was the last character sent to the terminal in response to a carriage return received from the terminal.
- The system parameter GOLFSTRIP is a 1.

DIRECT CALLS TO TTY TIP

The TTY TIP can be called by the following:

- Any other program, using a 3-word, standard TIP worklist. Worklists are queued to PTTYTIP. The monitor passes control to the TIP with a single worklist attached. PTTYTIP is the principal switch for the TTY TIP. (See TTY TIP trees, appendix G.) The switching procedure is based on the workcode in the worklist (lower half to word 0). The principal users of this call are:

  Internal processing to process downline data blocks and commands.

  The service module to set line and terminal changes.

  The TIP's own input state programs to process special conditions in input messages, or to continue processing the input block at OPS-level. These include special calls to process CR, LF, autorecognition, and X-OFF messages.

  The multiplex subsystem to have the TIP process special error conditions.

  The timing system for events that have timed out (processing resumes at the saved address), or for periodic calls to check for input from terminals.

  The multiplex subsystem at mux level 2. PTTTYMUX2 converts this mux level 2 worklist to an OPS-level worklist.

- SVM, to build the TCB. PTTYTCB sets interactive mode.
- Timing related calls handled by PTMSQUE (100 ms timer), PTMSCAN (100 ms timer) for the active TTY LCBs, and PTDELMS to delete periodic timing entries.

TABLE 9-3. CARRIAGE CONTROL FOR TTY OUTPUT MESSAGES

DBC	Character Sequence at Terminal
0	CR, LF
1	CR, 3LF
2	CR, LF
3	CR, LF
4	CR
5	(Nothing)
6	CR, LF
7	CR, LF
8	CR, LF, 4 nulls
9	CR, 3LF, 4 nulls
10	CR, LF, 4 nulls
11	CR, LF, 4 nulls
12	CR 4 nulls
13	4 nulls
14†	CR, LF, 4 nulls
15†	CR, LF, 4 nulls

†CE error message generated by TIP for invalid DBC received from host.

DIRECT CALLS FROM THE TTY TIP

The TTY TIP uses the following routines:

- Buffer handlers: PBGET1BF, PBREL1BF, PBRELCHN, and PBRELZRO to assign and release buffers.
- PTREGL is called to determine whether input is to be accepted from the terminals. Only three of the four possible input checks are used: the available block limit check is not made. (See section 6, PTREGL.) If regulation is in effect, the message is rejected. A break message is sent to the terminal. Later, the terminal is forced into output mode, and still later to idle mode. After two seconds in idle mode, the TIP gains control to end regulation. If a new message is waiting to be input, regulation can be reset immediately if the conditions for regulation have not disappeared.
- PBPROPOI is called to set up output block pointers when the TIP is searching for the next output block to unqueue and process.
- PBPIPOI is called to send the upline block to the host.

- PBCOIN is called to cause the command driver to prepare a TTY line for transmission, to shut down a line, or to transmit a message to a TTY terminal.
- PTSVlLCB is called to suspend processing until an event occurs.
- BLTIMTBL is used to set up timed functions or to clear timeout counters when the expected event occurs within the allowable period.
- PBLSPUT is called to generate worklist calls to the TTY TIP itself or to the service module (notify host of line failure, report autorecognition parameters, etc).
- PTCOMMAND is called to generate the data stream control CMD block (X-OFF) that is sent upline for tape mode.
- PNCEFILE is called to generate a CE error message for the host's engineering file.
- PBRETOPS returns control to the monitor.

ERROR PROCESSING

Lower level error processing takes place on the input state program level. If the error is resolved, the OPS-level TIP does not receive any notification of the problem. Otherwise, an OPS-level worklist comes to the TIP from the following sources:

- Disable line/delete TCB (from SVM) or hardware errors (from the multiplex subsystem). Both types of these cause PTTYHANGUP to abort the current transfer (if any), and cause the line to be marked down.
- Framing errors from the multiplex subsystem cause the TIP to send a delimiter back to the multiplex subsystem.

AUTORECOGNITION

The TIP automatically performs baud rate recognition for lines that have been configured as switched auto-baud by the host. Baud rates are recognized up to 1200 baud (except 600). After the terminal has been dialed in, the operator must enter a carriage return character as the first character to enable the TIP to determine the baud rate.

MODE 4 TIP 10

The Mode 4 terminal interface package (TIP) provides procedures to convert data from synchronous terminals using Mode 4 protocol to data that is compatible with the host's initialize or batch formats. 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 19200 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.

TIP FUNCTIONS

The TIP performs the following major functions:

- It interfaces terminal protocol (some variation of Mode 4 protocol) to host protocol (usually display code/PRUB format for batch devices, ASCII code for interactive devices).

- The TIP simultaneously controls several transfers to terminals on different lines. Each Mode 4 line can have several messages waiting for transfer. Information for controlling a transfer is contained in a worklist entry (WLE). The WLE is attached to the terminal control block (TCB) for the appropriate terminal on that line. The TIP must have an active TCB for each potentially active terminal device on the line (each terminal in a cluster must have at least one TCB). If a terminal device has a task in progress, additional tasks are queued to its TCB in the form of more worklist entries. One terminal cannot interrupt the active transfer on another terminal, but an interactive device on one terminal can interrupt the batch device on the same terminal.

  At the multiplex interface, 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 multiplex subsystem. The TIP specifies the data location on output; on input the mux-level (input state programs) TIP demultiplexes data under the control of the input data processor (part of the multiplex subsystem). The OPS-level TIP must specify the first of the series of state programs to be used to input the data to the command driver. For many output transfers, processing the entire block of data is handled within the firmware level text processing state programs. For both input and output operations, the TIP again gains control to terminate the data transfer or to process an unrecoverable transfer failure.

- 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 display code (or ASCII) where necessary.

- It polls terminals to solicit upline data or to ensure that the terminal is ready to accept downline data. The host requests the polling; the TIP controls actual timing of the polling.

- It resolves all contention between devices and controllers on a line and reports to the host any condition that has stopped an active connection except those caused by host command.

- It processes autorecognition to gather terminal configuration data for the host. Autorecognition cannot be performed on lines that have multicluster terminals.

- It toggles between read and write modes for interactive terminals.

- It processes unrecoverable errors in data transfers and reports the failure to the host. The TIP also processes terminal and line recovery in conjunction with the service module.

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

TERMINAL INTERFACE

A summary of the Mode 4 block format is shown in figure 10-1. The TIP must perform transformations between the Mode 4 formats and the block protocol format used in host/NPU transfers. (See section 6.) The Mode 4 transform used depends on the type of terminal (Mode 4A or Mode 4C). Terminal type is determined by the configuration messages used to set up the TCBs for the terminal/device. Terminal type can also be automatically determined by the TIP and reported to the host by using an upline line status service message (autorecognition).

TERMINAL ADDRESSING

Terminals can have both terminal (TA) and cluster (CA) addresses. The permissible address ranges are given in table 10-2.

MESSAGE TYPE INDICATORS

The message type indicator (MTI) in a Mode 4 transmission block prepares the TIP to accept certain types of blocks in reply. 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. The MTI codes shown in figure 10-2 are in hexadecimal notation, exclusive of parity.

E CODES

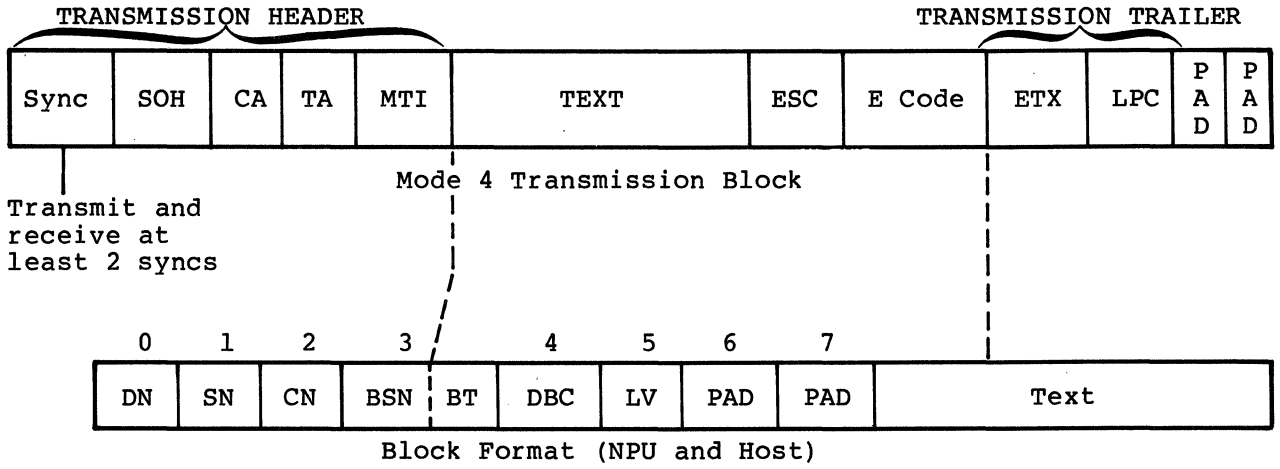
For downline transforms, device selection is performed by E codes. The E code follows the text in the output block and must be appended to the output by the TIP. For upline transforms, E codes coming from the terminal indicate the responding device and also report status. Received E codes are stripped from the input data by the TIP. Table 10-3 shows the E codes, exclusive of parity.

CODE CONVERSION

The TIP uses block protocol to transfer data to and from the host. (See section 6.)

Code conversions between Mode 4 blocks and host blocks are as follows:

DATA BLOCK FORMAT (odd parity):



Block Format (NPU and Host)

NON-DATA BLOCK FORMAT:

Sync	SOH	CA	TA	MTI	ETX	LPC

Mode 4 Transmission Block

Sync - Sync Byte = 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 (table 10-3)

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

BSN/BT - Block serial number/block type. BT for a data block must be 2 or 3

DBC - Data block clarifier

PAD - Byte of all 1's to ensure transmission of LPC by modem

LV - Level - not used

Figure 10-1. MODE 4 Protocol Message Formats

TABLE 10-2. MODE 4 TERMINAL/CLUSTER ADDRESSES

Address Type	Address Values		
	200UT	711	714
CA cluster address	70-7F	20-7F	20-7F
TA terminal address†			61-6F
Cluster controller		60	60
CRT/keyboard		61	
Printer		64	

†Bit 4 of the terminal address is the toggle bit. It is shown here as zero, but it can be a one. When the NPU transmits to the cluster, this bit changes with each succeeding output. If the output was correctly received by the cluster controller, the input response to the output carries the same value in the same bit position. Otherwise, the opposite value is carried in that position.

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	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, ALERT, 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 10-3. E-CODES

E Code	Hexa-decimal equivalent	TRANSMISSION MODE	
		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 is correctly printed. From card reader (text); normal card data.
E4	22	To CRT (text); position to start index.	Not used.

- All data in interactive data blocks to and from the host uses ASCII code. The TIP does character conversion for BCD terminals. The ASCII to external BCD translation includes folding the lowercase letters into uppercase and substituting blanks for any control code (control codes are described later). Those BCD terminals having switch selection for internal or external BCD must have the switch set to external. No translation is performed by the TIP on interactive data to or from ASCII terminals.

- All batch data blocks to and from the host use display code (internal BCD). The TIP does character-for-character conversion between the terminal code (external BCD or ASCII) and display code. For upline traffic from an ASCII terminal that has a 96-code character set, all lowercase characters are folded into uppercase display code characters.

- Each byte, including the longitudinal parity check (LPC), has odd parity in bit 7. LPC is odd parity on bits 0 through 6 of all characters except SYNC. The TIP inserts 7 through 14 SYNC characters between the MTI and TEXT on all output where the MTI is clear write or reset write.

- The TIP is not responsible for conversion to or from PRUB format for batch devices. This is done downline by PBIOPOI and upline by PBPIPOI.

HOST INTERFACE

The NPU transfers data to or from the host using block format. Three major types of transfer are defined, as follows:

- Interactive interface (console keyboard and display) is usually conversational in nature and uses ASCII code.
- Card reader (batch input) interface uses PRUB format upline and display code.
- Printer (batch output) interface uses PRUB format downline and display code.

INTERACTIVE INTERFACE

The interactive interface supports display keyboards attached to Mode 4 synchronous lines. The configuration can be multicluster; each cluster can be multiterminal. The 200UT display is also supported by this interface. Additional logic resolves contention for the common buffer used by batch devices and the display/keyboard.

The display is activated by delivery of the first output to the device. Polling begins following delivery of data to the terminal and continues until the terminal is deleted or fails. Polling is suspended to deliver output to the display, and during Mode 4A batch I/O operations. Polling resumes after completion of these conditions.

Output has priority over input. If there is data in the output queue, the TIP builds a transmission block from multiple BLK blocks received from the host. The transmission block is transmitted to the terminal when either a MSG block is received or the transmission buffer is filled with BLK blocks from the host. If the last block delivered is a MSG block, polling is resumed. If the last block delivered is a BLK block, the TIP waits for more output; it does not resume polling. The TIP inserts a new line character after each BLK block within the transmission block and inserts the appropriate number of blanks at the end of the last BLK or MSG to position the cursor at the beginning of the next line.

Input received while in the interactive mode is sent to the host as a MSG block.

CURSOR POSITIONING

During receipt of input, the TIP calculates the horizontal position of the cursor on the CRT screen. This calculation includes positioning for the following:

- Escape, carriage-return from any Mode 4 CRT.
- New line code from any Mode 4C CRT.

In either case the cursor is forced to the leftmost position on the next line.

After input the TIP generates a write El block containing sufficient blanks to force the cursor to wrap around to the first position of the next line. To do this, the TIP needs the screen width. When the TCB is built, the screen width is initialized to the INTERCOM default value. The screen width parameter can be reset to any value by the downline reconfigure terminal service message. (See appendix C.)

CARRIAGE CONTROL

Interactive carriage control for each output line is based on the data block clarifier (DBC) in each data block. Valid codes for the Mode 4 terminals are shown in table 10-4.

TABLE 10-4. DBC CODES FOR CARRIAGE CONTROL

DBC	Mode 4
0	Clear write
16 (10_{16})	Clear write
All other	Write

NOTE

The TIP will automatically insert a Clear Write on the first output to a Mode 4A display after a batch input or output operation.

UPLINE BREAKS

The reason for break field appears in the upline input stopped CMD block when input is terminated by the TIP. Interactive input is terminated by the TIP for abnormal conditions only. The secondary function break codes that can be generated by the interactive interface are shown in table 10-5.

Whenever an input stopped CMD block is generated on an interactive connection, any output queued for that connection is discarded.

TABLE 10-5. BREAK CODES

(PFC=3: Input Stopped) Secondary Function Codes (SFC)	Meaning
4	No response from terminal
5	Bad response from terminal; unable to select
6	Error response from terminal; unable to deliver

CONTENTION RESOLUTION

For the 200UT, using the display causes the card reader and printer connections to send input stopped CMD blocks to the host. These inter-channel interactions are intended to signal the use of the 200UT transmission buffer which is shared by the display, card reader, and printer.

CARD READER INTERFACE

The card reader is activated by sending a start input CMD block on the card reader connection. The TIP transforms card reader data into PRUB record format. Trailing blanks on each card are suppressed. Each block of data is sent to the host as a BLK block until an EOR (7/8/9 punch in column 1) or an EOI (6/7/8/9 punch in column 1) card is detected. A block containing EOR or EOI is sent to the host as a MSG block with appropriate flags set in the DBC header field. The EOR or EOI card is not included in the MSG block and multiple EOI or blank cards received after EOI are discarded.

The data following the last EOI is considered part of the next message. This allows multiple messages to be stacked in the card reader.

A file level number is taken from columns 2 and 3 of the EOR card, converted to a binary value, and placed in the level number field of the upline PRUB header.

Columns 79 and 80 of the EOR and job card are not tested for either the 026/029 option or TR (transparent), as neither feature applies to the Mode 4 TIP. Even though the 026/029 option is supported by some Mode 4 terminals, the special character conversions are performed by the terminal and are transparent to CCI and INTERCOM support of the terminal.

When card reader empty is detected by the TIP, an input stopped CMD block is forwarded to the host following the last PRUB block. The input stopped CMD block indicates a normal end (if the last card read was an EOI), or a break condition (if the last card was not EOI). Any partial PRUB is saved in CCI buffers. The host must inform the TIP of the desired disposition of the partial PRUB. A start input (resume) CMD block causes card reading input to continue placing data into the same PRUB; a stop input (terminate) CMD block causes the partial PRUB to be discarded. The card reader connection is then returned to the idle state.

If the response to the card reader poll message contains an E code = E1, the TIP sends the host an input stopped CMD block (PFC=3, SFC=2) to indicate a batch input interrupt has occurred. Other possible function codes for the input stopped CMD block are shown in table 10-5.

Flow control for the upline PRUB is regulated by the downline BACK block from the host. The TIP temporarily suspends polling for card reader data if the previous PRUB block has not been acknowledged and one additional transmission block has been received from the terminal. Polling continues when the BACK is received for the outstanding PRUB.

TABLE 10-6. CARD READER INPUT STOPPED CMD BLOCKS

(PFC=3: Input Stopped) Secondary Function Codes (SFC)	Meaning
00	CR empty, EOI received
01	CR empty, No EOI received
02	Batch interrupt
03	Slipped card
04	No response
05	Bad response
06	Error response

PRINTER INTERFACE

Output to the printer is activated by the host sending the first downline data block on a printer connection. The first block must be a MSG block in PRUB format and must be marked as a banner MSG in the DBC field. If the terminal is configured for the banner off condition, the MSG is discarded; otherwise, the MSG is converted to the file identification banner page.

Each subsequent PRUB is converted to output transmission blocks depending on the terminal code set and line width. If the print line taken from the PRUB is greater than the defined printer line width, the excess characters are automatically printed on the next line.

The first character of each line is normally interpreted as a carriage control character according to table 10-7. The carriage control character is ignored and replaced by single space when the suppress carriage control is activated. This is done by sending a configure/reconfigure TCB service message with the suppress carriage control flag FN/FV set.

The end-of-line character sequence is inserted at the end of each line of any output transmission block except the last. (The E-code sequence is appended to the end of each transmission block and takes the place of the end-of-line sequence.)

The end-of-line control sequences inserted by the TIP are as follows:

Terminal Type	Hexadecimal	Code
BCD	3E50	ESC⌀
ASCII	9B40	ESCa

TABLE 10-7. PRINTER CARRIAGE CONTROL CODES

Function	Display Code		EBCD Code		ASCII Code	
	(hex)	(char)	(hex)	(char)	(hex)	(char)
New page	1C	1	41	1	41	A
New line	2D	⌀	50	⌀	20	B
Space 2	1B	0	4A	0	4A	J
No space	25	+	B0	+	B0	0

The TIP sends a BACK block for each PRUB received. The BACK block is used for flow control only; it is sent upline to solicit the next downline PRUB block. Note that all lines of a PRUB block may not have been delivered to the terminal at the time the BACK is transmitted to the host for that block.

Output to the printer is always sent as a clear write MTI code. A response E-code of E1 or E2, received when polling for the printer, causes the TIP to send an output stopped CMD block to the host. The secondary function code of the CMD is 02 if E1 was received (batch interrupt). The SFC is 01 if E2 was received (printer not ready). Any undelivered PRUB data will remain in CCI buffers until the host issues a restart output CMD block. This causes the file output to be resumed. If the host sends a stop output CMD block, all data is discarded and the printer returns to the idle state.

The last PRUB block of an output file must be a MSG block. This does not cause an output stopped CMD block to be sent upline. This method is used to condition the TIP to expect a banner block as the next block.

The possible secondary function codes for an output stopped CMD block for the printer connection are shown in table 10-8.

TABLE 10-8. PRINTER INPUT STOPPED CMD BLOCKS

(PFC=05: Output Stopped) Secondary Function Codes (SFC)	Meaning
00	
01	Printer not ready
02	Batch interrupt
03	PM message
04	No response
05	Bad response
06	Error response

ERROR HANDLING

The Mode 4 TIP handles three types of errors, as follows:

- Short term errors in which an error counter is incremented and the operation is retired.
- Long term errors in which the short term errors cannot be corrected so an unrecoverable error is declared and the I/O is terminated.

Regulation due to running out of buffers for I/O transfers is discussed in the next subsection.

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 (over-length block or data carrier detected signal drops prematurely). • Character or longitudinal parity error. • Text in block which should not have text.
3	Error response (indicates an error).

Whenever any error occurs, the TIP increments the appropriate counter and retries the output/input sequence. If any counter reaches threshold value (currently set to five) in an attempt to complete a single transaction with the terminal, the TIP performs the long term error handling procedures and the send break subroutine generates a break message. The message is an upline input stopped CMD block or output stopped CMD block depending on the stream direction at the time the error occurred. The secondary function code in the CMD block indicates one of the following reason codes:

Reason for Break (RB	Description
04	No response
05	Bad response
06	Error response

An error condition caused by terminal malfunction normally is reported separately on each active connection.

If a TIP is unable to acquire sufficient buffers for an input block, any partial block is discarded and the terminal is polled again later. If the host is down, the terminal is not polled.

LONG TERM ERROR RECOVERY

After the TIP detects the abnormal terminal operation and sends the reason for break message to the host for all active connections (active indicates TIP is delivering data or polling for data), the TIP begins a failure mode polling cycle. The TIP polls the interactive device at a reduced rate where the rate depends on the number of terminals on a line and the system activity. No output is delivered while the TIP is in the failure mode polling cycle. If the TIP receives a good response to a poll, normal operation is restored. The TIP sends an input started CMD block (PFC=4, SFC=0) on the interactive connection. Any output in queue on the interactive connection is then delivered.

HANDLING OF ERRORS FOR CDC 711 TERMINAL

The toggle bit received from the 711 terminals is always the same as appeared in the previous write or poll message. This makes it impossible to determine whether data was correctly received by the 711 if the ACK or REJECT is garbled by transmission line noise. Therefore, the toggle bit of a poll message (which is ignored by all other Mode 4 terminals) is set to the value opposite to that which the terminal is expected to receive, assuming that the last message was correctly received by the terminal. Thus, if the TIP is polling a 711 for toggle state, and receives an unexpected toggle state, the TIP repeats the write message. This causes a duplicated output on the 711 display. The TIP cannot compensate for the loss of status information; however, no output data is lost. (This procedure is also supplied for the Tektronix 4014 terminal which implements Mode 4 protocol without complying with the standard.)

DUPLICATION OF WRITE DATA ON CRT

Those terminals that do not have separate CRT and transmission buffers (such as the 200 UT) 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 resend the output. Because the cursor is not in the same place as it was when the original write was performed, the output block appears 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.

INPUT REGULATION

The Mode 4 TIP calls PTREGL to check if input should be solicited from the console or the card reader. All four possible regulation criteria are checked. (See section 6.) If none of the regulation causes are present, the console is polled or the card reader is set to receive input.

AUTORECOGNITION

The host can request autorecognition for Mode 4 lines. This activates a procedure for determining the address and terminal that exists on the line. When the host configures the line, the TIP responds with the line enable response. If the line is dedicated, autorecognition begins. If the line is switched, 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 poll is done at cluster address $7D_{16}$ 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 1/2 to 1 second.

Once the cluster address has been determined, the TIP checks for receipt of a read message. The terminal operator must press the send key on at least one of the displays. 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:

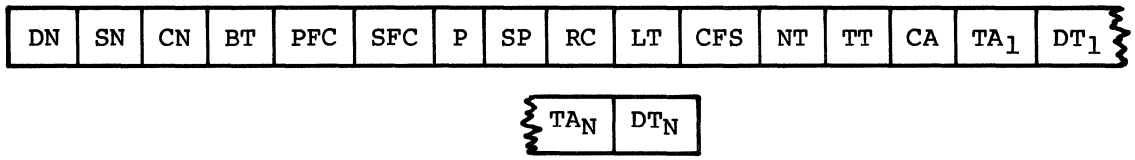
Field Name	Description
TT	Terminal type (table C-2)
CA	Cluster address
TA	Terminal address } for each terminal
DT	Device type } (see appendix E for more details)

For all terminals the appropriate terminal type is reported as one of the following: Mode 4A external BCD, Mode 4A ASCII, or Mode 4C. 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 terminal feature is used for Mode 4C terminals to determine the terminal addresses (TA) and device types (DT). Only the consoles are reported, with addresses ranging from 61_{16} to $6F_{16}$.

The printer device code for a Mode 4C impact printer is 2; the device code for a Mode 4C non-impact printer is 4. The TIP sends the host a line status operational service message for autorecognition. Format of the message is:



DN - Destination mode: 00 for host

SN - Source mode: NPU ID

CN - Connection number = 0 for service message

BT - Block type: 4 for CMD block

PFC - Primary function code: 06 for line status

SFC - Secondary function code: 02 for unsolicited (autorecognition) message

P - Port: line ID

SP - 00

RC - Reason code: 00 = line operational

LT - Line type: 01 - 02 - 03 - } see appendix C

CFS - Configuration state: 06 = line inoperative (no TCBs are configured yet)

NT - Number of terminals configured on the line: 00 (since no terminals are configured yet)

TT - Terminal type: 90 = Mode 4A EBCD
91 = Mode 4A ASCII
92 = Mode 4C

CA - Cluster address: 70_{16} CA $7F_{16}$

Note that only one CA is reported, as multicluster autorecognition is not supported. Multiclusters can, however, be configured on an autorecognition type line after the autorecognition is complete and the line is reported operational.

TA_i - For Mode 4A, this value is 60_{16}. For Mode 4C, value is a sequential value 60_{16} through $6F_{16}$, received from the configuration poll of the cluster address.

DT_i - For Mode 4A, three TA/DT pairs are reported: TA=60_{16}, DT=$0A_{16}$ (console), TA=60_{16}, DT=$2A_{16}$ (card reader), TA=60_{16}, DT=$4A_{16}$ (line printer).

For Mode 4C, up to 16 TA/DT pairs are reported. The TA value can range from 60_{16} through $6F_{16}$, and DT can be any of the following, depending on the cluster configuration.

$0A_{16}$ = console
$4A_{16}$ = impact printer
$8A_{16}$ = non-impact printer

MODE 4 PROTOCOL FEATURES NOT SUPPORTED

The following features of Mode 4 devices are not supported by the TIP.

- Status request
- Alert
- Diagnostic write
- Receipt of initialization

DIRECT CALLS TO THE MODE 4 TIP

The Mode 4 TIP can be called by the following:

- Any other program, using a 3-word standard TIP worklist. Worklists are queued to PTMD4TIP. The monitor passes control to the TIP with a single worklist attached. PTMD4TIP is the principal switch for the Mode 4 TIP. (See Mode 4 TIP trees, appendix G.) The switching procedure is based on the workcode in the worklist (lower half of word 0 of the worklist). The principal users of this call are:

  Internal processing to process downline data blocks and commands.

  The service module to set up line and terminal changes, and so forth.

  The TIP's own input state programs to process special conditions to input blocks, and to continue input processing on the OPS level.

  The mutliplex subsystem to have the TIP process special conditions such as terminal failure.

  Note that the Mode 4 TIP has an important secondary switch, PT4TASKPROCESSOR, which is called from the primary switch.

- Internal processing, by switching to the page and address of the Mode 4 text processor (PTTPMODE4). Text processing is done at the same time that the output message is converted from PRUB format to Mode 4 terminal format. Control then returns to internal processing which subsequently calls the TIP with a worklist so that the TIP can prepare the text processed block for output transmission to the terminal/device specified.

- SVM, to build the TCB. This is the direct call to PT4TCBINIT to finish building the TCB fields with the TIP's special default values.

- At multiplex 2 level by the multiplex subsystem. The state programs have been written to avoid this call. So if it occurs, a serious system error is indicated. PBHALT is called to stop the NPU.

- Directly from internal processing to queue interactive blocks.

DIRECT CALLS FROM THE MODE 4 TIP

The Mode 4 TIP uses the following routines:

- Buffer handlers: PBRELCHN, PBREL1BF, and PBGET1BF to release or to assign buffers.

- PTREGL is called to determine whether input is to be accepted, or whether data from the terminal is to be rejected. All four regulation checks are used. See section 6, PTREGL.

- PTTPINF is used to call the firmware text processing programs. Control returns directly to the TIP after text processing is completed.

- PBPOPOI is called to acknowledge the block that has been sent to the terminal.

- PBPIPOI is called to prepare the upline block for the host. This relieves the TIP of the conversion of PRUB formatting task.

- PBCOIN is called to cause the command driver to prepare a message or command for input or output. The Mode 4 protocol requires acknowledgment of most transmitted blocks (toggle bit serves as a check for some transmission).

- PT COMMAND is called to generate and to send data stream control CMD blocks to the host. See CMD block formats in section 6.

- BLTIMTBL is used to set up timed functions and to clear timeout counters when the expected event occurs within the allowable period.

- PBLSPUT is called to prepare worklist calls to the service module (for instance to report that a TCB has been deleted).

- PBUPABRT is called to send an upline abort message when the card reader fails.

- PNSGATH is called to gather statistics after certain message errors.

- PBGT1SEG is used to unqueue messages from the TCB by the secondary switch.

- PBSV1LCB is used to suspend tasks until an expected event (such as a reply to a poll message) occurs.

- PTRETOPS is the standard return to the OPS-monitor.

- PBHALT is called if a mux-2 level worklist occurs or if the secondary switch cannot find a valid task.

HASP TIP 11

The HASP multileaving 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 for transmission between NPU and HASP workstations; they are: data blocks and control blocks. Data blocks also contain control information. Positive acknowledgment of the receipt of each block is required. These blocks are not to be confused with the blocks used in the host or NPU block protocol. (See section 6.)

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 input/output; so that fast processing devices have most of the transmission time available, yet slow processing devices can have data whenever they are ready to use it. This ability to suspend transmission on one device's data stream, while transmitting data from other devices in a single block, is called multileaving.

Data can be transferred upline and downline in two data formats: transparent or nontransparent.

- Nontransparent data is treated as 6-bit characters formatted for cards on print line images.
- Transparent data is treated as 8-bit characters and is blocked and deblocked between the terminal without regard for card length, carriage control, or print-line width.

The TIP design is insensitive to line speeds, but has been tested at standard synchronous line speeds up to 9600 baud. Lines can be dedicated (with or without a modem/transceiver) or switched (dial up) with a modem. The transmission facilities are used by the TIP in a half duplex manner; that is, the TIP is either transmitting to the line or receiving from the line, but not both simultaneously.

HARDWARE CONSIDERATIONS

Some typical HASP hardware considerations are as follows:

- A typical HASP workstation consists of a keyboard, a CRT display, up to 7 card readers, a processor, and (optionally) an external storage magnetic tape or disk. The processor has computer-like functions, with upline and downline data processing (such as data compression, metering, testing line readiness, constructing data blocks and interpreting them, data storage, etc.)

- The terminal has its own software, 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 on the NPU end and the workstation processor on the terminal end.

- All terminals have interactive devices, and most have batch devices.

- Transmission over the line is bidirectional.

- Line speed is determined by the modem clock.

Other fixed workstation features are given in table 11-1.

TABLE 11-1. HASP WORKSTATION FEATURES

Feature	Supported
Multicard	Yes
Character set	EBCDIC-64
EBCDIC transparency (256 characters)	Yes
Character compression/expansion	All character strings
Console	Mandatory
Printer character set	EBCDIC-64
Card punch line	Yes
Print line width	80-150
Binary cards	No
Plotter	As card/print emulator only
Magnetic tape	
Paper tape	
Autorecognition	Yes

MAJOR TIP FUNCTIONS

The HASP TIP functions as follows:

- Interfaces the host codes and block protocol to a HASP workstation, which uses EBCDIC as its internal code and the HASP protocol.
- Handles tasks by queuing them as worklist entries (WLEs) to the terminal control block (TCB) for the line. 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 other than the ability of the terminal to direct the host to stop sending data to a particular device.
- Supports upline and downline data compression for both interactive and batch devices.
- Supports data flow control to various devices by the use of a function control sequence (FCS).
- Initiates line synchronization when the line has been configured; uses an enquiry/reply protocol to find if line can currently be used for a transfer.
- Provides soft error processing (retransmitting the garbled data block), and hard error processing (declaring a line inoperative when soft error processing fails to transmit data correctly).
- 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.
- Discards the terminal's signon card. A network login is used instead.
- Processes autorecognition only to the extent that this message is used to indicate the workstation is enabled.
- Interfaces to the multiplex subsystem. Downline, nontransparent data is reformatted to the terminal (HASP) protocol by the text processing state programs on a call from PBIOPOI (the state programs are reached through the HASP TIP text processor's call to PTTPINF). The TIP later gains control with a worklist and the converted data, and then calls the multiplex subsystem command driver. The address of the converted block and other message processing information, are placed in a command packet for the command driver (call to PBCOIN). The multiplex subsystem is then responsible for sending the data, character-by-character, over the line 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. Prior to starting the input transfer, the TIP sets up the message processing by passing the transfer parameters to PBCOIN (including the pointer to the first input state program to be used and an input buffer address). 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 host format. Batch data is later converted to PRUB format when the TIP passes control to the internal processing by calling PBPIPOI.

- Transparent data is passed upline and downline without text processing.

HASP PROTOCOL

The multileaving protocol consists of the bidirectional transmission of information blocks between an NPU and a HASP multileaving terminal.

The basic line protocol is standard BSC point to point (one terminal per line) and either transparent or nontransparent modes of BSC transmission are automatically recognized by the TIP on each received data block from the terminal. The TIP then uses the detected transmission mode for subsequent communication with that terminal.

Two types of blocks are defined, as follows:

- Control Blocks - contain binary synchronous communications (BSC) characters only (table 11-2 lists commonly used HASP mnemonics).

- Data Blocks - contain data records that are 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 of 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, all of which can be from the same device. 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 host sends multileaved downline data to the HASP terminal in transparent mode. In nontransparent mode the host must send to the HASP TIP the approximate desirable length of data for each active output stream (device) to make a single data block.

The HASP TIP supports multileaved data from a HASP workstation in both transparent and nontransparent modes. In nontransparent mode, the HASP TIP parses the input stream, relating each physical record to its associated connection (CN), and sends the data to the host, sorted by device. In transparent mode the host must separate the data for the various devices.

TABLE 11-2. HASP PROTOCOL MNEMONIC DEFINITIONS

Mnemonic	Definition	Use
ACK0	Acknowledge block or character	Positive acknowledgment that transmission was received.
BCD	Block control byte	Use 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 line 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 bits 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 character	Additional data record information.
STX	Start of text character	BSC control character.
SYN	Sync control character	Maintains line synchronization.
WLE	Worklist entry	

TERMINAL OPERATIONAL PROCEDURE

The workstation software is loaded and the communications line is initialized. After the signon 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 TCB, 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 TCB is ignored.

When a permission to initiate a function transmission TCB 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 TCB 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 or commands) do not have to follow the request-to-initiate or permission-to-initiate sequence. A console function can be initialized any time that the wait-a-bit in the FCS 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 (ACK)
- Negative acknowledge block (NAK)
- Enquiry block (ENQ)
- Idle block (ACK0)

Table 11-3 lists significant EBCDIC characters associated with these blocks.

Acknowledge Block (ACK)

The acknowledge block (ACK) 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 bits)

The ACK0 back indicates that the previous block was received without error and no data is available for transmission.

TABLE 11-3. HASP SIGNIFICANT EBCDIC CHARACTERS

Char	Hex	Definition
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.

Negative Acknowledge Block (NAK)

The negative acknowledge block (NAK) 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 block consists of the following control characters: SYN, SYN, SYN, SOH, ENG, PAD

SYN Synchronization control character
SOH Start of header control character
ENQ Enquiry control character
PAD Pad control character (all 1 bits)

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 that 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 two 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 appear in the following order:

- Block Control Byte (BCB); used for sequencing block.
- Function Control Sequence (FCS); defines the transmission flow (suspending all data or the data for a device, or restarting data transmission for one or all devices).
- Record Control Byte (RCB); carries status information for the following data and stream identification.
- Subrecord Control Byte (SRCB); carries more status and data control information.
- String Control Byte (SCB); describes the data string (length and nature - whether it is compressed or uncompressed data).

Following the first set of five bytes, additional data subblocks can be preceded by only an SCB, or by a sequence of RCB/SRCB/SCB.

Each control block byte is defined below. Figure 11-1 shows a typical transmission block and its associated control bytes.

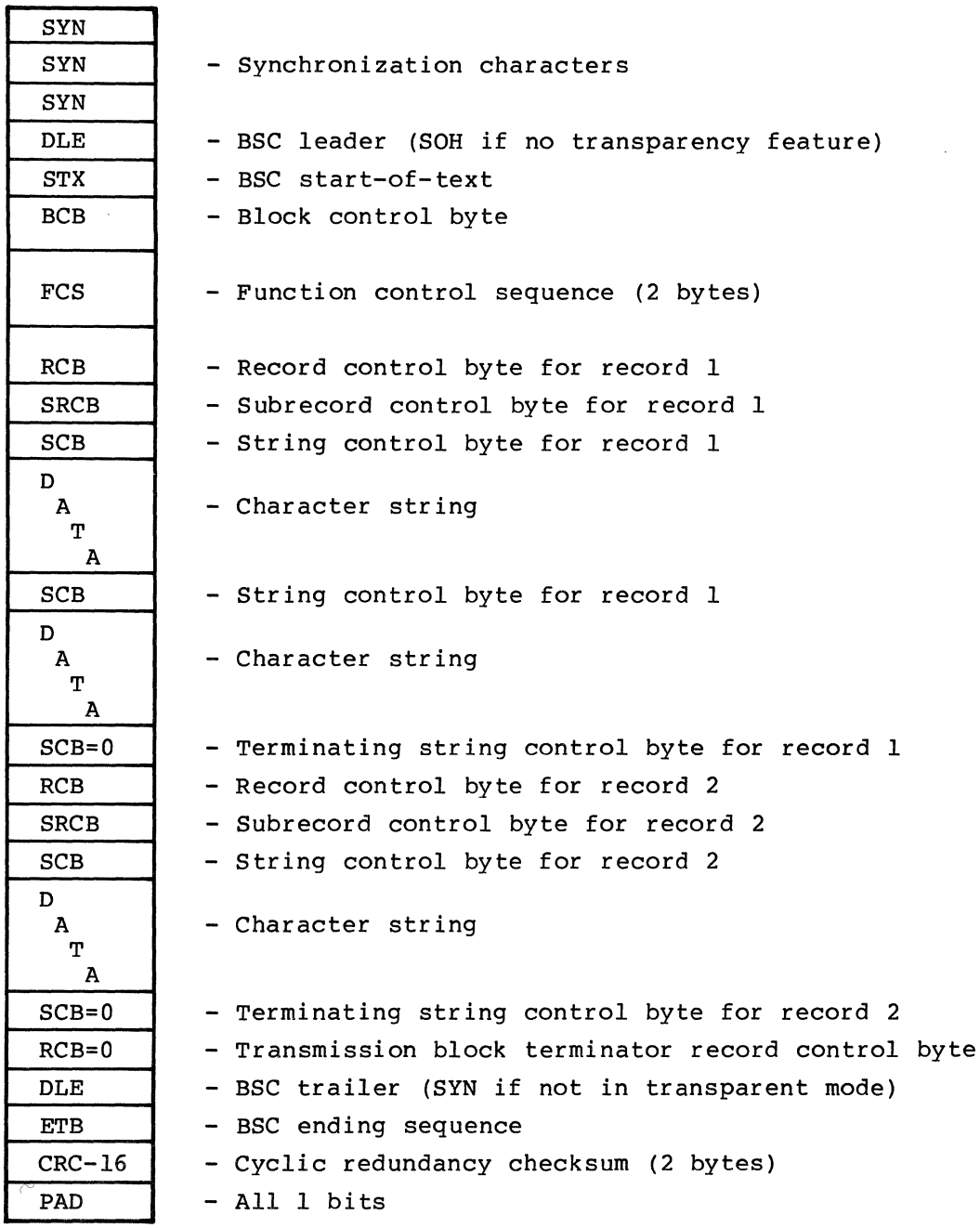
NOTE

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 bit representation is as follows:

```
Bit Number  0       7
           ØXXXCCCC

Ø     =  1  - Must always be 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 to 15
```



The signon blocks are described in the user terminal interface subsection (below). BCB error blocks are described in the error conditions subsection (below).

Figure 11-1. Typical HASP Multileaving Data Transmission Block

Function Control Sequence (FCS)

The function control sequence bit representation is as follows:

```
Bit Number      0       78       F
               ØSRRABCDØTRRWXYZ
```

```
Ø = 1 - Must always be on
S = 1 - Suspend all stream transmission (wait-a-bit)
  = 0 - Normal state
```

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 number	1 = A
Card reader number	2 = B
Card reader number	3 = C
Card reader number	4 = D
Card reader number	5 = W
Card reader number	6 = X
Card reader number	7 = Y
Card reader number	8 = Z

For downline use, the bits identify the punch or printer which will recieve the data:

```
Printer number 1 = A - Punch number 8
Printer number 2 = B - Punch number 7
Printer number 3 = C - Punch number 6
Printer number 4 = D - Punch number 5
Printer number 5 = W - Punch number 4
Printer number 6 = X - Punch number 3
Printer number 7 = Y - Punch number 2
Printer number 8 = Z - Punch number 1
```

Record Control Byte (RCB)

The record control byte bit representation is as follows:

```
Bit Number  0      7
           ØIIITTTT
```

```
      Ø  = 0    - End-of-transmission block (IIITTTT = 0)
      1         - All other RCBs
    III         - Stream identifier if TTTT ≠ 0
                - Control information if TTTT = 0 (control record)
         = 000  - Not used
         = 001  - Request to initiate a function transmission
         = 010  - Permission to initiate a function transmisssion
         = 011  - 101 = Not used
         = 110  - Bad BCD on last block received
         = 111  - General control record†
   TTTT         - Record type identifier
         = 0000 - Control record
         = 0001 - Operator message display request (downline)
         = 0010 - Operator command (upline)
         = 0011 - Card input record
         = 0100 - Print record
         = 0101 - Punch record
         = 0110 - 1111 = Not used
```

Subrecord Control Byte (SRCB)

The bit representation of the subrecord control byte is as follows:

```
Bit Number  0      7
           ØSSSSSSS

      Ø  = 1 (must always be on)
SSSSSSS  = Additional record information dependent upon record type (see
           TCB above)

For general control record:
    SSSSSSS = 100000001 - Initial terminal signon

For request or permission to initiate a function transmission:
    SSSSSSS - Stream identifier and record type identifier as described
              in RCB

For bad BCD on last block received:
    SSSSSS - 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
```

---

†The RCB for these functions is contained in the SRCB.

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 to 11 - Not used
RR = Not used
```

For input record:
SSSSSSS = MMBRRRR

```
 MM = 00       - SCB count unit = 1
      01 to 11 - Not used
  B = 0        - Normal EBCDIC card image
    = 1        - Not used
RRR            - Not used
```

String Control Byte (SCB)

The bit representation of the string control byte is as follows:

```
Bit Number  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 nonblank (character follows
                    SCB)
         CCCCC    - Duplication count
            K = 1 - Nonduplicate character string
        TCCCCC    - Character string length
```

If KTCCCCC = 0 and O = 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 described above, 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 was 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
- Signon blocks
- BCB error blocks

OPERATOR CONSOLE BLOCKS

Blocks that 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 be set in the FCS, but the remote console bit must be set.

END-OF-FILE BLOCKS (EOF)

Blocks that contain the end-of-file indicator do not contain any additional records from the same device stream in the data block following the EOF. Data blocks that are terminated by an EOF contain a final record in the format of figure 11-2 (shown for card reader number 1):

(BSC header)

BCB	
FCS	
RCB	= 10010011 - Card reader stream number 1
SRCB	= 10000000 - SCB count units = 1, EBCDIC card images
SCB	= 00000000 - EOF
RCB	= 00000000 - Transmission block terminator (BSC trailer)

(BSC trailer)

Figure 11-2. EOF Block

In order 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 TCB 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 shown in figure 11-3.

(BSC header)

BCB		
FCS		- Changed FCS
RCB	= 00000000	- Transmission block terminator

(BSC trailer)

Figure 11-3. FCS Change Block

USER INTERFACE

The user is required to load the software into the HASP workstation processor, to execute this initializing software, to signon 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
- Communication 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 signon parameters are used (default signon 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.

COMMUNICATION LINE INITIALIZATION

After the terminal is initialized, the communication 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, communications between the HASP TIP in the NPU, and the HASP workstation, are 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 signon 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 signon 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.

SIGNON BLOCK

```
Column  1      16        25
        /*SIGNON REMOTEnn password
```

NOTE

Record is shown in punched card format; least significant character on the left, most significant character on the right.

nn = a 1- or 2-digit number that can be used to correlate this remote terminal with information about it in the host computer.

Password can be blank.

The signon block format is shown in figure 11-4.

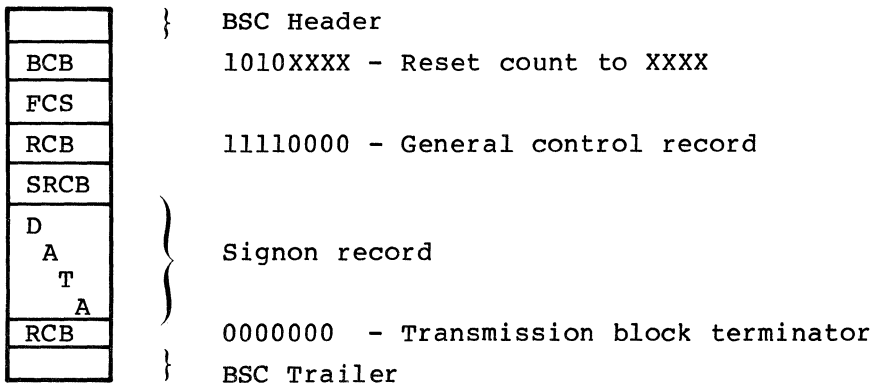


Figure 11-4. Signon Block format

The signon record is not sent to the host, since the host requires a separate logging on procedure at the operator's console.

SIGNOFF BLOCK

The /*SIGNOFF card, when transmitted to the HASP TIP as a record in the data block, has the same effect as an EOF. The HASP TIP converts the signoff record to a 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 signon block to be sent from the the HASP workstation. The signon 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 start input CMD, or downline data block, 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.

After the console is configured, the batch devices are configured. A CMD block from the host causes the HASP TIP to allow the 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. Each such terminal has a stream dedicated to its exclusive use.

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 (unless transparent mode is specified)
- Format conversion (if required), HASP to host format upline, host to HASP format downline
- Flow control, upline and downline
- HASP error recovery procedures
- Input/output streams, to or from a HASP terminal

CONFIGURATION AND ADDRESSING

In addition to the required console, a HASP workstation can have up to seven other devices. Each device has a separate stream ID and is specified in the configuration service messages from the host in the terminal address (TA) field. The cluster address (CA) field is always zero. Stream numbers are identified from 1 to 7 for each device type. Table 11-4 lists possible CA, TA, and DT values for HASP devices.

TABLE 11-4. HASP DEVICE TYPE

CA†	TA†	DT (device/class)	Stream/Device Number	
00	01	09	Console 1	
00	01	29	Card Reader 1	These streams are also used for magnetic tape and paper tape input.
.	.	.	.	
.	.	.	.	
.	.	.	.	
00	07	29	Card Reader 7	
00	01	49	Printer 1	
.	.	.	.	
.	.	.	.	
.	.	.	.	
00	07	49	Printer 7	
00	01	69	Punch 1	These streams are also used for plotter, magnetic tape, and paper tape output.
.	.	.	.	
.	.	.	.	
.	.	.	.	
00	07	69	Punch 7	

†CA = cluster address; TA = terminal address; DT = device type

INTERCOM commands or status to or from the terminal operator use the stream or device number when referring to a particular device. Example: line printer number 1 would be LP 1.

CONSOLE

A console type interactive device is required. The TIP accepts MSG and BLK blocks from the host and passes them to the terminal (the TIP does not distinguish between MSG and BLK blocks). For all blocks, data is converted from ASCII to EBCDIC prior to being sent to the terminal, but no other transformations take place. DBC carriage control values are ignored by the TIP since the HASP workstation does internal carriage control and screen formatting to its interactive device.

Blocks received from the terminal are delivered to the host after conversion from EBCDIC to ASCII code.

The console data stream is always open and does not require any commands to start or stop the stream. If the terminal stops the console data stream (by resetting the console stream FCS bit) for more than 30 seconds, the terminal is assumed to be inoperative. The host is notified by an input stopped CMD block (PFC=3, SFC=4).

NOTE

The data stream control CMD blocks are summarized in section 6.

CARD READER

Several CMD blocks are used to regulate the card reader data stream. Note that this data stream type is also used for magnetic tape and paper tape input. However, to be used in this way, the HASP workstation must have the device emulation controlware installed. A summary of the card reader CMD blocks is given in table 11-5.

TABLE 11-5. CARD READER DATA STREAM CONTROL CMD BLOCKS

PFC	SFC	Name	Definition
1	0	START INPUT, NON-TRANSPARENT	Downline command to start a card input in the nontransparent data mode.
1	1	START INPUT, TRANSPARENT	Downline command to start a card read input stream in the transparent data mode.
1	2	RESUME INPUT	Downline command to resume card read input after a suspended input. This command resets suspension of all workstation streams when issued on any stream.
2	0	STOP INPUT, TERMINATE	Downline command to stop a card read stream and to discard all data received.
2	1	STOP INPUT, SUSPEND	Downline command to stop a card read stream and hold all data received. This command suspends all streams on a workstation when issued on any stream.
3	0	INPUT STOPPED	Upline command signifying a /*EOS has been received on a card reader stream. The command is also sent upline when a stop input, terminate is received to signify all data has been terminated on the connection.

A HASP card reader is activated by the start input CMD block on the card reader connection. When a card reader connection is activated, it runs in a hot card reader mode; that is, the card reader stream is always on, unless terminated by the following:

- Input of a /*EOS (end-of-stream).
- The HASP workstation or the communications line fails and a recovery of the entire line/workstation takes place.
- A stop input CMD block (PFC=2, SFC=0/1) is received from the host.

The card reader stream must be terminated to change from a READ of non-transparent data to a READ FILENAME or a READ FILENAME with transparent data. The host does not act on a mode change request from the interactive console unless the card read stream is currently receiving data. Because of this restriction, the data stream is normally terminated by entering a /*EOS card when changing modes.

If the last input received from the card reader terminated the previous input file (either /*EOI or ETX), the HASP TIP discards any subsequent /*EOI or /*EOS card. A /*EOS card always causes an input stopped CMD block (PFC=3, SFC=0) to be sent to the host. Subsequent data received from the card reader is discarded until the data stream is started again.

Normal termination of an input job with a /*EOI card or ETX from the terminal causes termination of the PRUB data block with EOI marked in the DBC field. No upline command is sent to the host in this case.

Since the card reader not ready status is not reported by the HASP workstation, this condition is not reported to the host.

A stop input CMD block received from the host terminates the input data stream and, in addition, causes the TIP to send an input stopped CMD block (PFC=3, SFC=0) to the host as an acknowledgment. Any subsequent data received from the card reader on that data stream is discarded by the TIP.

Upline data is transformed to PRUB format by PBPOPOI in either transparent or nontransparent modes. The data stream can be placed in the transparent mode by one of three methods:

- For local input files, an optional parameter TR is included as a parameter in the READ FILENAME command.
- TR is included in columns 79 and 80 of the job card.
- TR is included in columns 79 and 80 of an EOR card.

A change to transparent mode requested by the READ FILENAME command causes a CMD block to be sent from the host to the TIP specifying the mode change. When TR is specified, none of the following data is translated. The data is stored as 8-bit characters. No EOR (7/8/9 punch) or /*EOI cards are recognized in transparent mode. The file is read until ETX is received from the terminal.

The HASP TIP also examines columns 79 and 80 of all job cards and EOR cards to determine if the code translation should be the 026 or 029 character set. A 26 in columns 79 and 80 specifies 026 mode and a 29 specifies 029 mode. The code translation default to INTERCOM default code set after a /*EOI card is detected.

Card Reader Nontransparent Data Mode

When in the nontransparent data mode, characters received from the card reader are expanded from the HASP compressed format, translated to display code, and stored in standard PRUB format. Trailing blanks on each card are discarded. The end of each card is marked within the PRUB with 2 to 11 zero characters.

Card Reader Transparent Data Mode

Transparent 8-bit characters are expanded from the HASP compressed format and stored in the PRUB without translation or marking of card boundaries. Records or transmission blocks are stored contiguously within the PRUB and, therefore, can be split across PRUB boundaries. Data is stored until ETX is received. When ETX is detected, the DBC of the last PRUB is marked as EOI and the card read stream is returned to the nontransparent data mode by means of an input stopped CMD block.

PRINTER

Several CMD blocks are used to regulate the printer data stream. Table 11-6 summarizes these commands.

TABLE 11-6. PRINTER DATA STREAM CONTROL CMD BLOCKS

PFC	SFC	Name	Definition
5	1	OUTPUT STOPPED, PRINTER NOT READY	Upline command to notify the host that the print stream has been suspended by the terminal for more than 30 seconds.
5	3	OUTPUT STOPPED, PM MESSAGE	Upline command to notify the host that the print stream has been stopped due to receipt of a PM message from the operator.
7	0	RESUME OUTPUT	Downline command to cause the TIP to restart output after a stop condition.
8	0	STOP OUTPUT, ABORT	Downline command to cause the TIP to discard buffers and stop output to the printer.

Output to a printer is started by the host sending the first downline data block on the printer's connection. The first block received is normally a banner MSG block. If the terminal is configured for the banner off condition, the banner block is discarded; otherwise, the MSG block is converted to two copies of the file identification banner page.

Data blocks originate in the host as PRUBs. The DBC of each PRUB is examined by the TIP (this is the direct call from PBIOPOI) for data mode, transparent or nontransparent, and the data is translated accordingly. Each subsequent PRUB is converted to HASP printer protocol in the same way.

Printer Nontransparent Data Mode

Data within the PRUB is considered to be print lines. The end of each line is detected according to the standard PRUB format: (FF_{16}). This can optionally be preceded by 00_{16}.

If the print line taken from the PRUB is larger than the configured printer line width, excess characters are automatically printed on the next line. Print lines are never split across transmission blocks. The first character of each line is normally interpreted as a carriage control character, and is converted to the corresponding HASP workstation subrecord control byte (SRCB), according to table 11-7. Optionally, if the data stream is configured to suppress carriage control, the first character of each line is ignored and replaced by a single space before printed output.

All characters are converted from display code to EBCDIC code prior to being output.

TABLE 11-7. HASP PRINTER CARRIAGE CONTROL CODES

INTERCOM CODE	Function Before Print	HASP SRCB Value (Hexadecimal)
1	New page	B1
+	No space	80
0	Space 2	A2
-	Space 3	A3
∅	Space 1	A1
Others	Space 1	A1

Printer Transparent Data Mode

For PRUs marked as transparent data, no print lines are detected within the PRUB. Characters are placed in the transmission block without code conversion, carriage control, or end-of-line processing.

If an EOR or EOI block is received, the transmission block is terminated with the last character of that PRUB. Otherwise, transmission blocks are filled to the maximum configured size.

Command Interface for the Printer

Print files are output to the printer without TIP intervention between files. The host is not notified when the last block of a file is output. The host is notified by an output stopped (printer not ready) CMD block (PFC=5, SFC=1) if the data stream is suspended by the terminal for more than 30 seconds. The TIP also notifies the host and stops the printer data stream when a PM message is detected as the first two characters of a print line. The command sent upline for this condition is output stopped, PM (PFC=5, SFC=3). The PM print line is then sent to the console device.

The print stream can be restarted after a stop condition by an INTERCOM command to the host which causes the host to send a downline restart output CMD block (PFC=7, SFC=0). The print stream is aborted by a downline stop output CMD block (PFC=8, SFC=0). This command causes any PRUB queued for the stream to be discarded.

PUNCH

The TIP processes card punch output in a manner similar to printer output. The differences are as follows:

- There is no carriage control function.
- Punch records are 80 characters long.
- A lace card consisting of 70 characters of *, the 7-character job name, and 3 blanks, is punched as the first card of each job file as a separator.
- There is an option to punch an EM ($19) character immediately following the last data character on each card that is less than 80 characters in length. The EM character allows card reading to be more efficient for some terminals.

Note that the punch data stream can be used for plotters, or output magnetic or paper tape, if the workstation contains the required emulation controlware. Files output to the punch can be specified as transparent data in the same manner as print files. In this case, they would be handled exactly as described for transparent data for the line printer except that the lace card would be punched in place of the printer banner page.

A summary of the commands applicable to the punch stream are given in the table 11-8.

TABLE 11-8. PUNCH DATA STREAM CONTROL CMD BLOCK

PFC	SFC	Name	Definition
5	1	OUTPUT STOPPED, DEVICE NOT READY	Upline command to notify the host the print stream has been suspended by the terminal for more than 30 seconds.
7	0	RESUME OUTPUT TRANSPARENT	Downline command to cause the TIP to start output after a stop condition.
8	0	STOP OUTPUT, ABORT	Downline command to cause the TIP to discard buffers and stop output to the punch.

ERROR CONDITIONS

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)

Cyclic redundancy checking only occurs 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 the data block 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 re-transmission 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

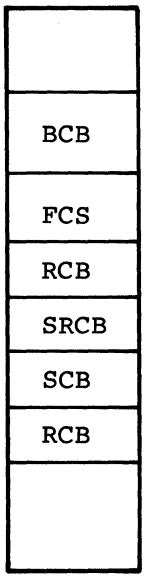
If an unknown response error occurs, the receiving process transmits a NAK block to the transmitting process. This informs the transmitting process that a retransmission of the last block is required. If the retransmitted block is correct, processing continues.

BLOCK CONTROL BYTE (BCB) ERROR

Every data block has a block control byte which 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 equal to the expected block sequence count, a block control byte error occurs.

If a block control byte error occurs and the block sequence count is a duplicate of a block sequence count previously received, (expected block sequence 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 a duplicate block count, as described in the previous paragraph, 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 transmit a reset block control byte. The format of the block control byte error block is shown in figure 11-5.



- BSC header
- 1001XXXX; ignore sequence checking where XXXX = received block sequence count
- 11100000; bad BCB on last block
- 1000YYYY; where YYYY is expected block sequence count
- All zeros; end-of-record
- All zeros; transmission block terminator
- BSC trailer

Figure 11-5. Format of Block Control Byte (BCB) 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/NPU as a function of the busy state of the workstation device, which uses or produces the data.

- Upline Regulation

  In response to the stop input CMD block (section 6), the TIP sends an input stopped CMD block 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 CMD block to the host following the data. Permission to send more data is not granted until a start input CMD block is received from the host.

  To check whether any internal NPU condition exists which should cause rejection of input messages, the TIP calls PTREGL, (1) at the time an acknowledgment is sent, or (2) when an output command is sent to the workstation that could result in an input data block being returned. Only two regulation conditions are checked, as follows:

  Host has reset accept input flag.

  Logical link regulation priority exceeds input priority.

- Downline Data Flow Control

  The function control sequence fields control 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 input stopped CMD block on the related connection after a timeout occurs. In some subsequent upline block from the terminal to the TIP, the function control sequence bit for the specified stream is set to change transmission from the suspend state to the continue state. This causes the TIP to send input resumed CMD blocks upline on the related connection number.

The data stream to the host then continues.

If a request to initiate function transmission sent from the HASP TIP is denied by the terminal, then an output stopped CMD block is sent upline for this device's connection number (CN), after a timeout occurs. If permission is granted, a resume output CMD block is sent.

AUTORECOGNITION

Lines using BSC modes can be configured for autorecognition. Since some program must distinguish between standard TIPs (2780/3780 and HASP terminals) using BSC codes for autorecognition, this function has been placed in the BSC TIP. As soon as the BSC TIP discovers the auto-

recognition message from the terminal is for a HASP workstation, the BSC TIP uses PBLSPUT to build an OPS-level worklist for the HASP TIP. The workcode is set to line enabled. The HASP TIP processes this OPS-level worklist code, by setting up LCB fields (the BSC TIP has previously set up other LCB fields for the HASP TIP).

DIRECT CALLS TO THE HASP TIP

The HASP TIP can be called by the following:

- Any other program, using a 3-word standard TIP worklist. Worklists are queued to PTHOPSTIP. The monitor passes control to the TIP with a single worklist attached. PTHOPSTIP is the principal switch for the HASP TIP. (See HASP TIP trees, appendix G.) The switching procedure is based on the workcode in the worklist (lower half of word 0). The principal users of this call are:

  Internal processing to process downline data blocks and commands.

  The service module to set upline and terminal changes.

  The multiplex subsystem to have the TIP process special conditions such as unrecoverable hardware errors and protocol acknowledgments.

  The TIP's own input state programs to start the TIP into the upline text processing cycle.

  The TIP's own text processing state programs for unrecoverable error conditions.

- Internal processing, by switching to the page and address of the HASP text processor, PTTPHASP. Downline text processing is done at the same time as the output message is converted from PRUB format to terminal format. Control then returns to internal processing which subsequently calls the TIP with a worklist so that the TIP can prepare the text processing block for output transmisssion.

- SVM so that the TIP can finish processing the TCB at configure time. The TIP sets the default values for fields that were not explicitly configured by SVM using an FN/FV pair.

- The multiplex subsystem at mux 2 level. This call is immediately converted to an OPS-level call if the work code indicated buffer threshold has been reached. Otherwise, no action is taken.

DIRECT CALLS FROM THE HASP TIP

The HASP TIP uses the following routines:

- Buffer handlers: PBGETlBF, PBRELlBF, PBRELCHN and PBRELZRO are used to assign and release buffers. PBCLR is used to clear buffer space.

- PTREGL is used to check if input should be rejected. The conditions for regulation were discussed previously.

- PTTPINF is used to call the firmware text processing programs. Control returns directly to the TIP after the upline or downline data block has been text processed.

- PBPOPOI is called to send upline acknowledgments (BACK blocks) when the data block has been sent to the terminal.

- PBPIPOI is called to prepare the upline block for the host. This relieves the TIP of the task of converting the block to PRUB format.

- PBUPABRT is called when the host has sent a stop input (terminate) CMD block to the TIP. All upline PRUBs are released and the TCB fields are set to their no traffic values.

- PTCOMMAND is called to generate the data stream control CMD blocks for the host.

- PBPTlSEG is used for adding entries to the data queue.

- BLTIMTBL is used to prepare timed functions and to clear timeout counters when the expected event occurs within the allowable period.

- PBLSPUT is called to prepare worklist calls to the service module (for instance, to report that a TCB has been deleted), or to call the HASP OPS-level TIP itself when converting the mux 2 level entry to an OPS level entry, and when queuing input messages for later processing.

HASP POSTPRINT

HASP printers vary in the way they perform 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 this format:

[CARRIAGE CONTROL] [DATA]

The terminal action is performed in this order:

(1) Perform the carriage control action.
(2) Print the data.

Initially, CCI treated all printers as preprint terminals.

However, postprint terminals cannot perform these actions as one step. These terminal use the following sequence of actions:

(1) Print the data.
(2) Perform the carriage control actions.

To handle both preprint and postprint terminals with the same data format, CCI divides HASP printer output data into two records:

[CARRIAGE CONTROL] [DATA]

[2 BLANKS] [CARRIAGE CONTROL] and [DATA] [no CC]

where no CC indicates 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 that are used by the TIPs and the multiplex subsystem to speed programming. One set of state programs controls upline transfer (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. (Exception: The TTY TIP does not translate code, up or downline; that TIP lacks text processing programs.) The entire group of state processes comprising 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, and 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 of 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 multiplexer 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 of the state programs and the 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 preempted 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 required unless the output sent by the host is in transparent mode. Normally PBIOPOI calls the text processing programs of the OPS-level TIP directly. This program in turn calls the text processing state program to convert data to terminal format. The TIP makes a direct call to the state programs reformatting and converting to terminal code where necessary. Note that the format has already been changed from PRUB format. This

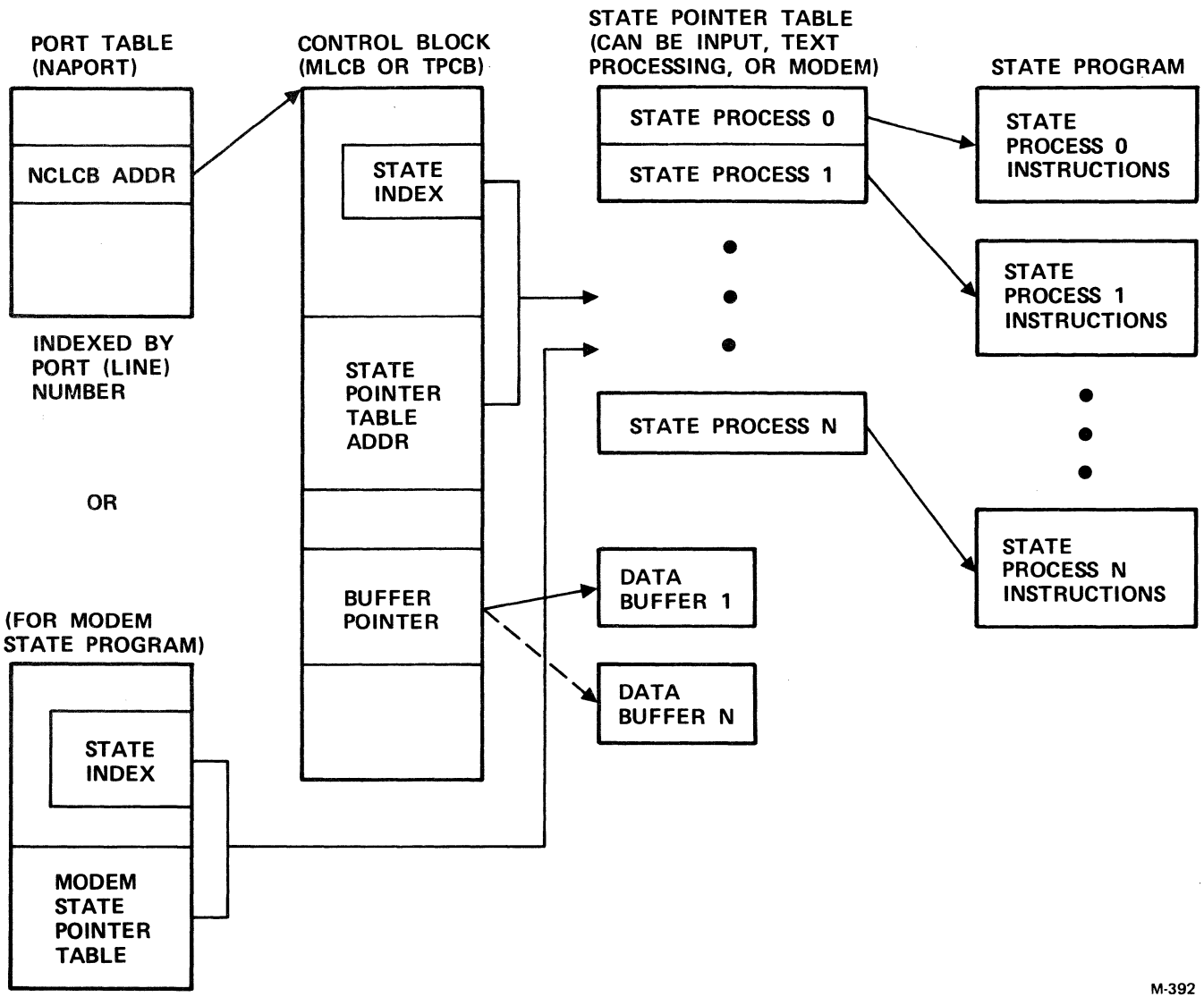


Figure 12-1. Locating a State Process

operation moves the data from the buffers holding the partially transformed output data to buffers holding the data in terminal format. This data conversion must be accomplished before calling the TIP to initiate output on the line.

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 move the data into a line-oriented input buffer. Then the multiplex subsystem calls the OPS-level TIP. The OPS-level TIP calls PTTPINF to convert this block of terminal data to one or more blocks of device-oriented data in host format. Note that conversion to PRUB format is done later by PBPIPOI. 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 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 function 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 converted data are passed to the host via the TIP, PBPIPOI, 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 converted data to the host via PBPIPOI and 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 similiar 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 of 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

  The firmware input data interrupt causes the multiplex subsystem to pass control to the designated input state process for the line/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 characters 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 state 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 (after transformed by PBIOPOI) 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 that performs state-drive 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 text processing state processes.
- Releases unused destination buffers created by the save and restore state instructions 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 text processing 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 multiplexer 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 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 informtion 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 which 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 mux 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 from 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
CLA STATUS HANDLING		
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
FLAG CONTROL		
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)

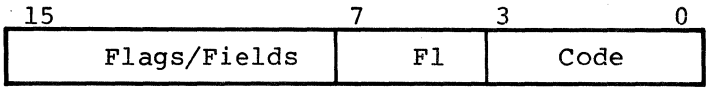
Name	Function	Parameters
WORKLIST HANDLING		
TIBWL	Terminate input buffer, build a worklist entry (WLE) for TIP.	WC, WL, EOT, ACTION, EP
TIBSWC	Terminate input buffer, save workcode (WC) in MLCB.	WC, EOT, ACTION
BLDWL	Build WLE for OPS or multiplex level.	WC, WL, ACTION, EP
BLD01	Generate CLA status WLE for multiplex level 2.	SCI, ACTION
TEXT PROCESSOR OPERATIONS		
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 1-word or a 2-word instruction (parameter list). The usual word-oriented format is as follows:

1-word



Flags - Usually in bits 14 and 15

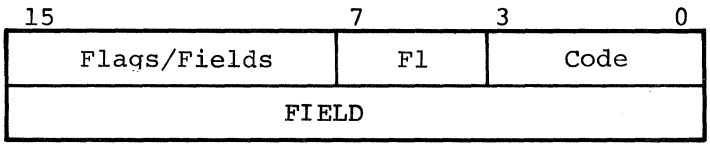
Fl - A set of frequently used parameters, including ACTION, a parameter that specifies the actions to take prior to exiting from the instruction sequence

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.

2-word



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 Language Reference Manual.

GLOSSARY A

ADDRESS - A location of data (as in the NPU main or micromemory) or of a device (as a peripheral device or terminal). The NPU main memory is paged.

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 or status transfers and the Q register for command or addressing information.

ASYNCHRONOUS PROTOCOL - The protocol used by asynchronous, teletypewriter-compatible devices. The NPU/terminal interface is handled by the asynchronous TTY TIP.

AUTORECOGNITION - A capability offered to selected terminals which allows the TIP to discern some device characteristics for the terminal, rather than having the terminal or the host specify the information.

BANDWIDTH - For CCI, bandwidth indicates the transfer rate (in characters per second) between the NPU and the terminal.

BASE SYSTEM SOFTWARE - The set of programs in CCI which supply the monitor, timing, interrupt handling, and multiplexing functions for the NPU. Base software also includes common areas.

BATCH DATA FORMAT - The transmission format used by the block protocol of CCI. Batch data is usually in 6-bit display code, within 8-bit bytes, within PRU-sized blocks.

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. Different 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 system. 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. CCI assigns one size of buffer for data and three 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 matter.

BUFFER THRESHOLD - The minimum number of buffers available for assignment of new tasks. As the buffer level falls below the threshold, new tasks are rejected (regulation).

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 offline diagnostics.

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 CCI, a character is ordinarily in 8-bit ASCII format (7 bits plus an eighth bit reserved for parity) or 6-bit display code.

CIRCULAR INPUT BUFFER (CIB) - The fixed buffer used by the multiplex subsystem to collect all data passing upline from the multiplex. The buffer is controlled by a put pointer for the multiplexer and a pick 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 - Area of main memory dedicated to system and global data. These are usually below address $1D50_{16}$.

COMMUNICATIONS CONTROL INTERCOM (CCI) - A set of modules that perform the tasks delegated to the NPU in the network message processing system.

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 - A terminal devoted to network control processing. There are two such terminals: the host computer system console and the NPU console.

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 can interrupt it share data elements.

CONTROL BLOCK - (1) The type of block used to transmit control information (as opposed to data).

(2) Data structures assigned for special configuration or status purposes in the NPU. The major control blocks are line control blocks (LCB), logical link control blocks (LLCB), terminal control blocks (TCB), worklist control blocks (WLCB), 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 NPU and the host. Transmissions across the coupler use block protocol.

CROSS - The software support system for CCI. This system supports PASCAL coding, and is run on the host computer. One output is a CCI program in 255x machine code ready for execution in the NPU.

CYCLIC REDUNDANCY CHECK (CRC) - A check code transmitted with blocks of data. This code is used by several protocols, including the HASP mode 4, and BSC protocols.

DATA - Information processsed 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 BLOCK CLARIFIER (DBC) - A byte in the header of a data block. The DBC contains data control information.

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 other terminal formats.

DATA SET - A hardware interface that transforms analog data to digital data and vice versa. A data set is used to connect remotely located terminals to the NPU.

DESTINATION NODE (DN) - The network node to which a message is directed; for instance, the DN of an upline message can be INTERCOM 5.

DIAGNOSTICS - Software programs or combinations of programs and tables which aid the troubleshooter in isolating problems.

DIRECT CALL - The method of passing control directly from one program to another. This is the usual transfer mode. Some CCI calls are indirect, through the monitor. Such OPS-level indirect calls pass information to the called program through parameter areas called worklists. See Worklist.

DIRECT MEMORY ACCESS (DMA) - The high-speed input/output channel to the NPU main memory. This channel is used by the coupler for host/NPU buffered transfers and by the multiplex subsystem (MLIA) for line to/from NPU transfers.

DIRECTORY - A table in CCI that contains information used to route blocks to the proper interface and line. There are directories for source and destination node and for connection number. A routed message is attached to the terminal control block for the line over which the message will pass.

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 host to produce a listing of the dumped hexadecimal information.

EXTERNAL BCD - A type of binary coded decimal code used by some TTY and mode 4 terminals.

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.

FORMAT EFFECTOR (FE) - A control symbol used by certain protocols (for instance, the HASP protocol).

FULL DUPLEX (FDX) - A transmission mode allowing data transfer in both directions at the same time. A full-duplex system requires a dual set of data lines, each set dedicated to transmission in one direction only.

FUNCTION CODE - A 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) secondary function code (SFC). See appendix E for definitions of these codes.

GLOBAL VARIABLES - Variables that are defined for use throughout CCI. Contrast global variables with local variables that are identified only within a single NPU or host program.

HALF DUPLEX (HDX) - A transmission mode allowing data transfer in one direction at a time. Normally a single set of data lines carries input, ouput, and part of the control information. Contention for use is possible in half-duplex mode, and must be resolved by the protocol governing line transfers.

HALT CODE - A code generated by the NPU when it executes a soft-stop. These codes (which indicate the cause of the stoppage) are delivered at the NPU console in the form of a halt message.

HASP - Houston Automatic Spooling Process; the 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 code conversions since the host uses ASCII and display code for its processing.

HEADER - A word or set of words at the beginning of a block, record, file, or buffer which contains control information for that unit of data.

HOST - The computer that controls the network and contains INTERCOM 5.

HOST INTERFACE PACKAGE (HIP) - The CCI program that handles block transfers across the host/local NPU interface. The HIP normally uses CCI block protocol.

ID - The identifier for ports, nodes, lines, links, or terminals. Any hardware elements 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 processes have the same IDs for all network terminals, lines, and so forth.

INPUT BUFFER - A data buffer reserved by CCI for receiving an upline message for the host. The input buffer is assigned and released dynamically. Contrast with the circular input buffer on the multiplex subsystem interface.

INTERACTIVE DATA FORMAT - The transmission format used by the block protocol of CCI. Interactive data is in 7-bit ASCII, within 8-bit bytes, within line-sized blocks.

INTERACE (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 internal terminal format handled by a HIP), to the peripheral devices (NPU console protocol handled by base system software), and to the terminals via the multiplex subsystem (various protocols; standard protocols are handled by the mode 4, TTY, 2780/3780, and HASP TIPs).

INTERRUPTS - A set of hardware lines and software programs which allow external events to interrupt NPU processing. Interrupting programs are allowed preferential processing on a priority basis. The lowest priority level is processed by the OPS monitor.

LINE - A connection between an NPU and a terminal.

LINE CONTROL BLOCK (LCB) - A table assigned to each active line in the system. It contains configuration information as well as current processing information.

LOAD - The process of moving programs downline from the host and storing them in the NPU main and micromemory.

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 a network terminal and a host program. Until terminated, the logical connection allows messages to pass between the two entities.

LOGICAL LINK CONTROL BLOCK (LLCB) - A table assigned to each logical link in the system which is directly connected to this NPU. The table contains configuration information as well as current processing information.

LOGICAL REQUEST PACKET (LRP) - A parameter or data packet to or from 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 console device.

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.

MAIN MEMORY - The macromemory of the NPU. This memory 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 three additional bits for parity and program protection. Memory is packaged in 16K and 32K word increments; 48K is the minimum memory size.

MASK REGISTER - A register used in the interrupt subsystem to check if an interrupt is of sufficiently high priority to be processed now. Each bit in the mask register (M) corresponds to an interrupt line. The M register operates under program control.

MESSAGE - A logical unit of information, as processed by a program. When transmitted over a network, a message can consist of one or more physical blocks.

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 contains the emulator microprogram that allows use of assembly language.

MICROPROCESSOR - The portion of the NPU which processes CCI programs.

MODE 4 - A communication line transmission protocol for synchronous terminals. The protocol requires the polling of sources for input to the data communications network. CCI supports mode 4A, mode 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.

NOTE

Considerable differences exist in the terminology associated with Mode 4 devices. The equivalent terms are shown in Table A-1.

TABLE A-1. MODE 4 TERMINOLOGY

Nomenclature in This Manual	Mode 4A Nomenclature	Mode 4C Nomenclature
NPU	Data source	Control station
Cluster address	Site address	Station address
Cluster controller	Equipment controller	Station
Terminal address	Station address	Device address

MODEM - A hardware device for converting analog levels to digital signals and vice versa. Long lines interface to digital equipment via modems. Modem is synonymous with data set.

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 which executes OPS level programs by scanning a table of programs that have pending tasks (worklist entries).

MULTILEAVING - The technique of interleaving several similar data streams in one transmission stream, while preserving the identity of the data stream source or destination.

MUX-LEVEL - A series of priority levels for time dependent tasks such as input or output data processing at the multiplex subsystem interface.

MULTIPLEX LOOP INTERFACE ADAPTER (MLIA) - The hardware portion of the multiplex subsystem which controls the multiplex loops (input and output) as well as the interface between the NPU and the multiplex subsystem.

MULTIPLEX SUBSYSTEM - The portion of the NPU base system software which performs multiplexing tasks for upline and downline data, and also demultiplexes upline data from the CIB and places the data into line-oriented input data buffers.

NETWORK - A connected set of network elements consisting of a host, one or more NPUs, and terminals.

NETWORK LOGICAL ADDRESS - The address used by block protocol to establish routing for the message. The network logical address consists of three parts: DN - the destination node, SN - the source node, and CN - the connection number.

NETWORK PROCESSING UNIT (NPU) - The collection of 255x hardware and peripherals together with the software Communications Control INTERCOM (CCI) modules. These CCI programs buffer and transmit data between terminals and the host computer.

NODE - A network element that creates, absorbs, switches, and/or buffers message blocks. Typical system nodes are INTERCOM in the host, and the coupler node of an NPU.

OFFLINE DIAGNOSTICS - Optional diagnostics for the NPU which require the NPU to be disconnected from the network.

ONLINE DIAGNOSTICS - Optional diagnostics for the NPU which 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 or dialed to an unused CLA address. These diagnostics are provided if the user purchases a maintenance contract.

OPS-LEVEL - The lowest priority level of CCI. All processing that is not time critical is performed at this priority level.

OPS MONITOR - The NPU monitor. See Monitor.

OUTPUT BUFFER - Any buffer that is currently used to output information from the NPU to a peripheral device, or to a terminal via the multiplex subsystem.

PAGING - 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.

PARITY - A bit-oriented data assurance method. Parity in the NPU is word oriented and is ordinarily not controlled by the operator. A parity bit is added when words are stored in main memory, and 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 (LPC - longitudinal parity character).

PASCAL - A high-level programming language used for CCI programs. Most CCI OPS-level programs are written in PASCAL language.

PERIPHERAL DEVICE - An I/O device attached to the NPU A/Q channel. The NPU console is a peripheral device.

PERIPHERAL PROCESSING UNIT (PPU) - The part of the host dedicated to performing input/output transfers. The coupler connects the PPU to an NPU via a data channel.

PHYSICAL RECORD UNIT (PRU) - Under NOS/BE, the amount of information transmitted by a single physical operation of a specified device. The size of a PRU depends on the device, as shown in Table A-2.

TABLE A-2. PRU SIZES

Device	Size in Number of 60-Bit Words
Mass storage	64
Tape in SI format with coded data	128
Tape in SI format with binary data	512
Tape in I format	512
Tape in other format	Undefined

A PRU that is not full of user data is called a short PRU; a PRU that has a level terminator but no user data is called a zero-length PRU.

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 must be converted into a PRP prior to output to the device.

POINT OF INTERFACE (POI) PROGRAMS - A special set of base system programs that interface directly with TIPs. POIs are provided for such standard functions as ending an output operation or ending an input operation.

POLLING - (1) The action of checking ports to find if a terminal is ready to transmit or receive another word of data. The multiplex subsystem performs the polling operation for active lines under the direction of a TIP. (2) The action of soliciting input from certain types of terminals. A poll message is output to the terminal. The response is device status or an indication that no data is 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.

PRIMARY FUNCTION CODE (PFC) - See Function Code.

PRIORITY LEVEL - CCI 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 the format of the data and commands, and the sequence of commands needed to prepare the nodes for sending and receiving data. CCI was block protocol, coupler protocol, and varies terminal protocols.

QUEUE - A sequence of blocks, buffers, messages, and so forth. Most NPU queues are maintained by leaving the queued elements in place and using a combination of tables of pointers to the next queued elements 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 - A data unit defined for the host record manager or for HASP workstations and HASP transmissions. A record contains space for at least one character of data and normally has a header associated with it. Records for HASP can 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 has another 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.

RESPONSE MESSAGES - A subclass of service (network control) messages directed to the host, 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, and CN) is the primary criterion for routing. The NPU directories are used to accomplish routing.

SECONDARY FUNCTION CODE (SFC) - See Function Code.

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.

SOURCE NODE (SN) - The network node originating a message or block of information.

STATE PROGRAMS - Programs in the multiplex subsystem with execution that depends on the current state of the message being transmitted; that is, one state program is executed at the start of the message header processing, another at start of text processing, another at end-of-text processing, and so forth.

STATISTICS SERVICE MESSAGE - A subclass of service messages that contain detailed information about the characteristics and history of a network element such as a line or a terminal.

STATUS - Information relating to the current state of a device, line, and so forth. Service messages are the principal carriers of status information. Statistics are a special subclass of status.

STRING - 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 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.

SWITCHING - The process of routing a message or block to the specified internal program or external destination.

SYSTEM CONFIGURATION - The process of setting tables and variables throughout the network to assign lines, terminals, and so forth, 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 communication line. Terminals supply input messages to, and/or accept output messages from, INTERCOM 5. 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. Most TCBs are dynamically assigned.

TERMINAL INTERFACE PACKAGES (TIPs) - NPU programs that provide the interface between terminal format and host data format. 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 for messages, status, and so forth.

UNSOLICITED SERVICE MESSAGES - Service messages sent to the host which do not respond to a previous service message from the host. Unsolicited service messages report hardware or software failures to the host.

UPLINE - The direction of message travel from a terminal through an NPU to the host.

WORD - The basic storage and processing element of a computer. The NPU uses 16-bit words (main memory) and 32-bit words (internal to the microprocessor only). All interfaces are 16-bit words (DMA and A/Q) or in character format (multiplexer loop interface). Characters are stored in main memory two per word. Hosts 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 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.

WORKLIST PROCESSOR - (1) Any system program that receives and processes worklists. (2) The program within the multiplex subsystem that handles worklist entries within the multiplex subsystem firmware (PMWOLP).

WORKLISTS - Packets of information containing the parameters for a task to be performed. Programs use worklists to communicate information to different operating levels. 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 multiplex (priority) level.

CCI MNEMONICS B

This appendix lists mnemonics used in comment fields within source code listings of CCI, or used as symbolic entities within the code itself. The mnemonics defined in the following columns also appear in the text of other CCI documentation.

ABL	Allowable block limit
ACK	Acknowledge block (HASP mode 4 and BSC protocols)
A/Q	The A/Q internal I/O channel of the NPU
ASCII	American Standard Code for Information Interchange
ASYNC	Asynchronous
BACK	Acknowledgment block
BCB	Block control byte
BFC	Block flow control
BFR	Buffer
BLK	Message block
BSC	Binary synchronous communications (protocol)
BSN	Block serial number (for blocks/SVM)
BT	Block type
CA	Cluster address
CB	Control block
CCI	Communications control INTERCOM in NPU
CE	Customer engineer
CFS	Configuration state (for SVM)
CIB	Circular input buffer
CLA	Communication line adapter
CMD	Command (element of block protocol)
CN	Connection number
CND	Connection number directory
CR	Carriage return
CRC	Cyclic redundancy checksum
DBC	Data block clarifier (for blocks/SVM)
DCB	Data carrier detect
DEL	Delete character

DMA	Direct memory access (in NPU)
DN	Destination node number
DND	Destination node directory
DSR	Data set ready
DT	Device type
DTR	Data terminal ready
EB	Error bit in response service message
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 (6/7/8/9 punch or /*EOI for HASP and 2780/3780 terminals)
EOM	End of medium
EOR	End of record
ETB	End of block (HASP/BSC protocols)
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	Front end
FF	Form feed
FN	Field number (for SVM)
FS	Forward supervision (block protocol)
FV	Field values (for SVM protocol)
HASP	Houston Automatic Spooling Process (protocol)
HCP	Host Communications Processor (alternate name for NPU)
HDLC	High level data link control
HDX	Half duplex
HIP	Host Interface Package
HL	Higher level

ID	Identifier (number or code)
IDC	Internal data channel (in NPU)
I/O	Input/Output
ISO	International Standards Organization
LCB	Line control block in NPU
LCD	Last character displacement (in buffer)
LD	Load/dump
LF	Line feed
LL	Logical link
LLCB	Logical link control block in NPU
LLREG	Logical link regulation
LM	Loop multiplexer
LRP	Logical request packet (I/O)
LT	Line type
M	Mask register
MLCB	Multiplex line control block
MLIA	Multiplex loop interface adapter
MPLINK	The CYBER Cross System linking editor
MSG	Message (element of block protocol)
MTI	Message type indicators (Mode 4 protocol)
M4	Mode 4
MD4	Mode 4
NAK	Negative acknowledgment block (HASP, mode 4, and BSC protocol)
NBL	Network block limit
NL	Number of lines
NPTINTAB	CCP data structure containing initialization status.
NPU	Network processing unit
NT	Number of terminals
ODD	Output data demand (multiplex subsystem)
OPS	Operations (OPS-level = monitor level programs)
OPSMON	Monitor
P	1. Priority 2. Port

PAD	Pading element (synchronous protocols)
PFC	Primary function code (for SVM)
PM	Print message
POI	Point of interface
PPU	Peripheral processing unit (in host)
PRP	Physical request packet
RAM	Random access memory
RB	Response bit in response service message
RC	1. Remote concentrator 2. Reason code
RCB	Record control byte (HASP protocol)
RCV	Receive state
RL	Regulation level
RM	Response message (SM)
RS	Reverse supervision (block protocol)
RT	Record type
RTS	Request to send
SCB	String control byte (HASP protocol)
SCF	System configure file
SFC	Secondary function code (for SVM)
SM	Service message
SN	Source node (for blocks/SVM)
SND	Source node directory
SOH	Start of header
SP	Subport
SRCB	Subrecord control byte (HASP protocol)
SPRM	System Programmer's Reference Manual
STX	Start of text
SVM	Service module for processing service messages
SYNC	Synchronizing character (synchronous protocols)
TA	Terminal address
TC	Terminal class
TCB	Terminal control block in NPU
TDP	Time dependent program
TIP	Terminal Interface Package
TO	Timeout

TOT	Total number of status service messages
TPCB	Text processing control block
TT	Terminal type
TTY	Teletypewriter (asynchronous device)
TUP	Test utility package
VAR	PASCAL keyword for variable statements
WACK	Wait acknowledgment block (synchronization protocol)
WL	Worklist
WLCB	Worklist control block
WLE	Worklist entry
WLP	Worklist processor
X-OFF	Stop tape character (asynchronous protocol)
X-ON	Start tape character (asynchronous protocol)

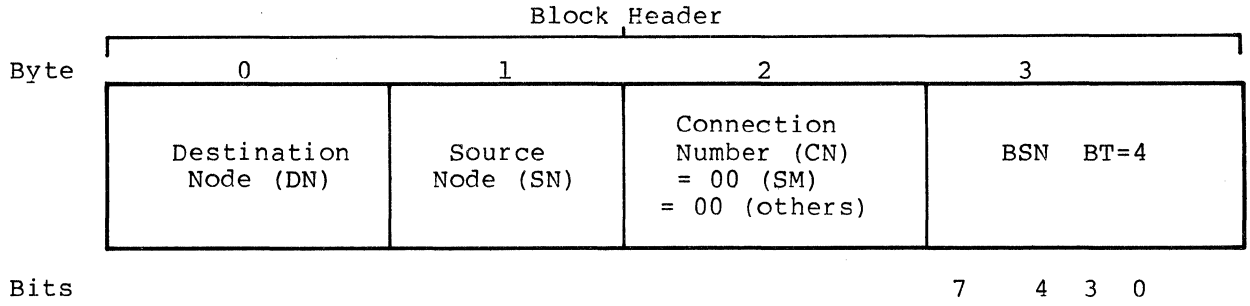
SERVICE AND COMMAND MESSAGE SUMMARY C

This appendix is divided into five parts:

- The general format of 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

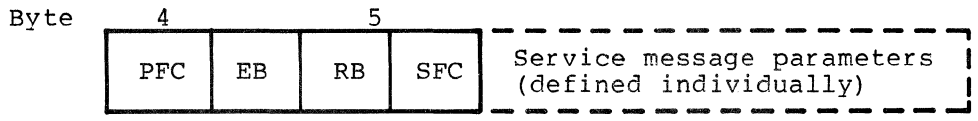
SERVICE AND COMMAND MESSAGE GENERAL FORMAT

All service 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.



BSN - Block serial number
BT - Block type = 04 for service messages/commands. 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.

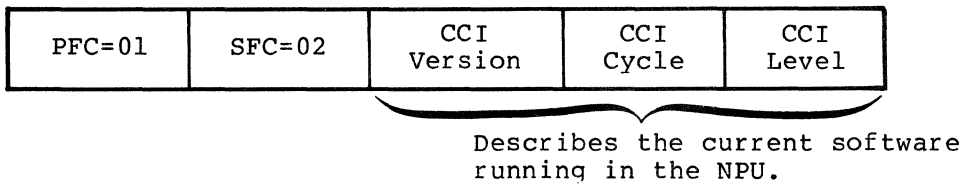


PFC - Primary function code
EB - 1 = Error response service message
RB - 1 = Normal response service message
SFC - Secondary function code

00 - $3F_{16}$ = Reserved for network use
40 - $9F_{16}$ = Reserved for intrahost use
A0 - BF_{16} - Reserved for expansion
C0 - $E0_{16}$ - Reserved for network use
E1 - EF_{16} - Reserved for installations

INDIVIDUAL SERVICE MESSAGES

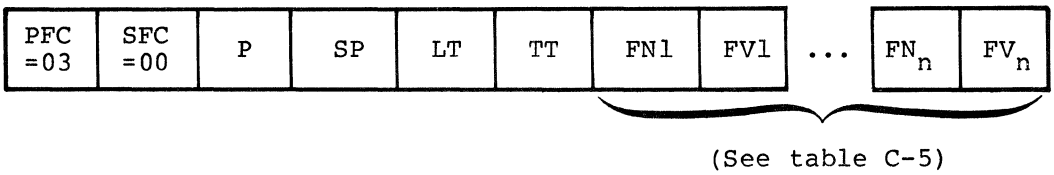
NPU INITIALIZED



Response

NONE

CONFIGURE LINE



LT - See table C-3
TT - See table C-2

TABLE C-1. SERVICE MESSAGE/COMMAND SUMMARY (Sheet 1 of 2)

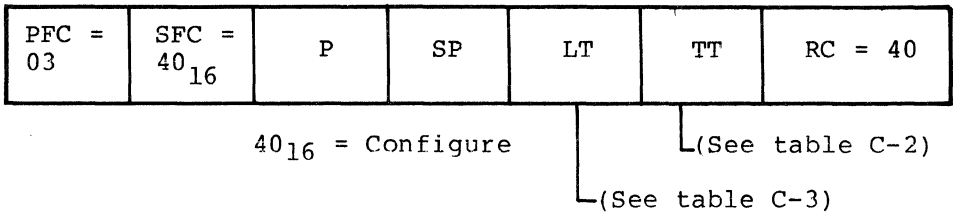
SERVICE MESSAGES

Service Message Name	PFC (hex)	NPU Mnemonic	SFC (hex)	NPU Mnemonic
NPU initialized	01	D8LOAD	1	D9FRC
	02			
Configure line Delete line Configure terminal Reconfigure terminal Delete terminal	03	D8CONFIG	0 1 2 3 4	D9LNCNF D9LNNDLT D9TMLCNF D9TMLRCNF D9TMLDLT
	04			
	05			
Line status request Terminal status request Line count request	06	D8STATUS	2 3 5	D9LNSTAT D9TMLSTAT
NPU statistics Line statistics Terminal statistics	07	D8COUNTS	0 1 2	D9LNSTAT D9CNTLN D9CNTML
Enable line Disable line Disconnect line	08	D8LINE	0 1 2	D9ENABLE D9DISABLE D9DISCONNECT
	09			
CE error	0A	D8EVENT	0	D9CE
Host broadcast one Host broadcast all Operator message Terminal	0C	D8USER	0 1 2 3	D9BRD1 D9BRDCST D9OPMSG D9TMCL

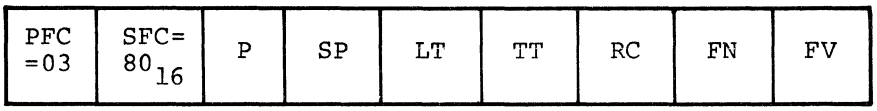
TABLE C-1. SERVICE MESSAGE/COMMAND SUMMARY (Sheet 2 of 2)

Command Message Type	PFC	Command Message Name	SFC	Use
Start input	01	Initiate, nontransparent	0	Start input from batch device
		Initiate, transparent	1	Start input from batch device
		Resume	2	Start input from batch device
Stop input	02	Terminate	0	Discard data and stop polling
		Suspend	1	Stop polling and wait
Input stopped	03	End	0	Normal end
		Break 1	1	Reason for break defined by TIP
		Break 2	2	Reason for break defined by TIP
Input started	04	Restart after stop	0	Interactive resume after break (status only)
Output stopped	05	Break 1	0	Reason for break defined by TIP
		Break 2	1	Reason for break defined by TIP
		Break 3	2	Reason for break defined by TIP
		Break 4	3	Reason for break defined by TIP
		Break 5	4	Reason for break defined by TIP
		Break 6	5	Reason for break defined by TIP
Output started	06	Restart after stop	0	Status only
Restart output	07	-	0	Resume output stream
Stop output	08	-	0	Discard data and terminate
BSN error	09	NPU has detected block sequence number out of order	0	Diagnostic purposes only

Normal Response



Error Response



SFC = 80_{16} - 80_{16} = Configure

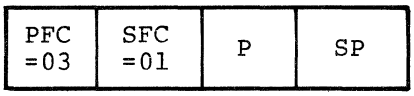
LT - See table C-3

TT - See table C-2

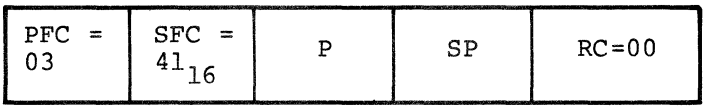
RC - 01 = Invalid FN/FV
02 = Invalid line number
03 = Line control block already configured
04 = Invalid line type
05 = Invalid terminal type

FN/FV - Pair returned if RC = 01

DELETE LINE



Normal Response



Error Response

PFC = 03	SFC = 81_{16}	P	SP	RC =

RC - 02 = Invalid line number
03 = Line control block already deleted

CONFIGURE/RECONFIGURE TERMINAL

PFC =03	SFC =02	P	SP	CA	TA	DT	CN	FN1	FV1	...	FN_n	FV_n

See table C-7

02 = Configure
03 = Reconfigure

DT = See table C-2

The table below shows the valid CA and TA values for each terminal.

	CA (hexadecimal)	TA (hexadecimal)
Mode 4A	70-7F	60
Mode 4C	70-7F	61-6F
TTY	00	00
HASP	00	1-7 (1)
BSC	00	12-13 (2)

(1) Equal to the stream of the device. The interactive console must be 01, card reader(s) 01...07, printer(s) 01...07, punch(es) 01...07.

(2) Punch only, all other devices are zero.

Normal Response

PFC= 03	SFC	P	SP	CA	TA	DT	CN	RC=0

SFC - 42_{16} = Terminal configured
43_{16} = Terminal reconfigured

DT - See table C-2

Error Response

PFC =03	SFC	P	SP	CA	TA	DT	CN	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

05 = Invalid terminal type

06 = Line inoperative or not enabled

08 = Logical link not established

09 = CN in use

010 = Console not configured for a Mode 4 device

} (Configured)

FN/FV - Pair returned if RC = 01 or 09

DELETE TERMINAL

PFC =03	SFC =04	P	SP	CA	TA	DT	CN

DT - See table C-2

Normal Response

PFC =03	SFC= 44_{16}	P	SP	CA	TA	DT	CN	RC=00

DT - See table C-2

Error Response

PFC =03	SFC= 84_{16}	P	AP	CA	TA	DT	CN	RC

DT - See table C-2

RC - 02 = Invalid line number
     03 = Terminal not configured

LINE STATUS REQUEST

PFC= 06	SFC= 02	P	SP

P/SP - If missing, return status on all lines except trunks

Normal Response

PFC= 06	SFC= 42_{16}	P	SP	RC	LT	CFS	NT

LT - See table C-3

CFS - See table C-4

   00 - Line operational
RC=04 - Line inoperative
   05 - No ring indicator or autorecognition in progress
   06 - Stop - CLA not responding, (CE intervention required)

Error Response

PFC= 06	SFC= 82_{16}	P	SP	RC

RC - 01 = Invalid line number or no lines configured
     02 = Line status request in progress
     03 = Illegal state

Unsolicited Response

Only for autorecognition

PFC =06	SFC =02	P	SP	RC	LT	CFS	NT	TT	CA	TA_1	DT_1	--	TA_i	DT_i

RC - Same as other line status responses
LT - See table C-3
CFS - See table C-4
TT - See table C-2

For Autorecognition responses, the TA DT pairs are repeated for each terminal that can be detected by the TIP. The Mode 4 TIP can report up to 15 TA DT pairs with the full range of values as shown in table C-2 for DT.

For autorecognition between BSC and HASP terminals, the TT field is returned as a 03 or 04 if the terminal is configured for TT = 04 with the autorecognition bit set. (See table C-2.) The CA field will be zero and no TA, DT fields will be returned.

TERMINAL STATUS REQUEST

PFC= 06	SFC= 03	P	SP

Normal Response

PFC 06	SFC= 43_{16}	P	SP	CA	TA	DT	RC	DN	SN	CN	TOT

DT - See table C-2

RC - 00 = Terminal operational
04 = Terminal inoperative

Error Response

PFC= 06	SFC= 83_{16}	P	SP	RC

RC - 01 = Invalid line number; no terminals configured belonging to requestor
02 = No terminals configured
03 = Terminal not configured
05 = Terminal status request in progress

Unsolicited Response

NOTE

Normal response can 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

PFC= 07	SFC= 00	Statistics Words

2 bytes/word

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

LINE STATISTICS

PFC= 07	SFC= 01	P	SP	0	00	Statistics Words 1 - 4

2 bytes /word

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)

2 bytes/word

PFC =07	SFC =02	P	SP	CA	TA	DT	CN	Statistics Words 1 - 3

DT - See table C-2

Word 1 - Blocks transmitted
Word 2 - Blocks received
Word 3 - Blocks in error

Response

None

ENABLE LINE

PFC= 08	SFC= 00	P	SP

NORMAL RESPONSE (LINE ENABLED)

PFC= 08	SFC= 40_{16}	P	SP	RC	LT	CFS	NT=0

RC - See line status request response codes
LT - See table C-3
CFS - See table C-4

Error Response (Line Not Enabled)

PFC= 08	SFC= 80_{16}	P	SP	RC

RC - See line status request response codes

DISABLE LINE

PFC= 08	SFC= 01	P	SP

Normal Response (Line Disabled)

PFC= 08	SFC= 41_{16}	P	SP	RC=0	LT	CFS	NT

LT - See table C-3
CFS - See table C-4

Error Response

PFC= 08	SFC= 81_{16}	P	SP	RC

RC - See line status request responses

DISCONNECT LINE

PFC= 08	SFC= 02	P	SP

Normal Response

Normal response is line enabled normal response SM.

Error Response

PFC= 08	SFC= 82_{16}*	P	SP	RC

*SFC = 80_{16} for RC 04

RC - See line status request response codes

CE ERROR

PFC= $0A_{16}$	SFC= 00	EC	1 - 27 bytes of data

EC - See error codes in appendix B of the CCI reference manual

Response

None

HOST BROADCAST ONE

PFC= $0C_{16}$	SFC= 00	P	SP	CA	TA	DT	Text

Normal Response

PFC= $0C_{16}$	SFC= 40_{16}	P	SP	CA	TA	DT	RC=0

Error Response

PFC= $0C_{16}$	SFC= 40_{16}	P	SP	CA	TA	DT	RC

RC - 01 = Invalid line number
02 = Invalid device type
03 = Terminal not configured
04 = Terminal inoperative
05 = Host broadcast in process

HOST BROADCAST

PFC= $0C_{16}$	SFC= 01	ID1= 00	ID2= 00	20_{16}	• • •	

TEXT

20_{16} - Filler space for DBC
If zero, broadcast to interactive terminals

Normal Response

PFC= $0C_{16}$	SFC= 41_{16}	RC= 00

Error Response

PFC= $0C_{16}$	SFC= 81_{16}	RC

RC - 01 = Not used
02 = Broadcast already in progress

SERVICE MESSAGE MNEMONICS

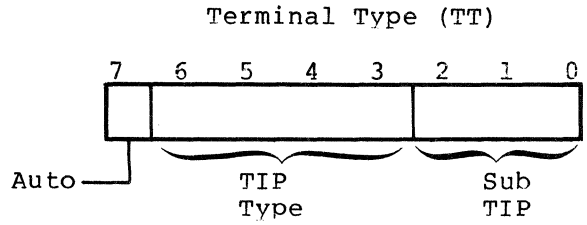
The following table defines abbreviations used in the individual service message descriptions.

Abbreviation	Meaning
ABL	Available Block Limit - the number of blocks allowed to be outstanding for any terminal at any one time.
BSN	Block Serial Number - part of the block protocol.
BT	Block Type - SMs are always of type CMD (BT=4).
CA	Cluster Address - part of a terminal's physical identification.
CD	Code Type.
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 an SM, the CN is always zero. When used as data in an SM, the CN can 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.)

Abbreviation	Meaning
LS	Line Speed Index
LT	Line Type - used to describe the transmission capabilities of the line. (See table C-3.)
NL	Number of Lines - the number of configured lines belonging to the host.
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.
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 (TT)/DEVICE TYPE (DT)



In the Configure Line SM, the TT (Terminal Type) field is defined as shown using the following values:

Auto = 0 No autorecognition.
= 1 Autorecognition performed when line becomes operational.

TIP Type =	0 N/A	1 TTY	2 Mode 4	3 HASP	4 BSC (2780/3780)
Sub TIP = 0	N/A	ASCII - 110		N/A	N/A
= 1			M4A/BCD		
= 2		ASCII - 150	M4A/ASCII		
= 3		ASCII - 300	M4C		
= 4					

Note 1 - Sub TIP is used for upline SMs only.
Note 2 - Use TIP Type 4 for autorecognition on HASP, BSC type lines.

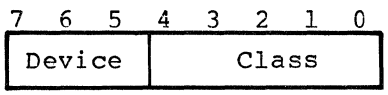


TABLE C-2. TERMINAL TYPE (TT)/DEVICE TYPE (DT) (Contd)

Class	Terminals Supported (By Device)				
	0 Console	1 Card Reader	2 Line Printer	3 Card Punch	4 Non-Impact Printer
1	TTY Comp.				
2					
4					
5					
6					
7	2780	2780	2780	2780	
8	3780	3780	3780	3780	
9	HASP	HASP	HASP	HASP	
10	Mode 4	Mode 4	Mode 4		Mode 4

*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	Transmission Facility	CLA Type	Modem Type	Answer Mode	Carrier Type	Circuit Type	Turn-Around Required	Turn-Around Delayed	Transmission Mode
(1)	HDX	2560-1	RS232-201A/2081 Compatible	Switched	Controlled	2 Wire	YES	NO	Synchronous
(2)	FDX†	2560-1	RS232-201B/208A Compatible	Dedicated	Controlled	4 Wire	YES	NO	Synchronous
(3)	FDX	2560-1	RS232-201B/208A Compatible	Dedicated	Constant	4 Wire	NO	NO	Synchronous
(4)	HDX	2561-1	35.8-1 Transceiver	Dedicated	Controlled	2 Wire	YES	NO	Asynchronous
(5)	HDX	2561-1		Switched	Controlled	2 Wire	YES	NO	Asynchronous
(6)	FDX	2561-1	RS232-103E/113 Compatible	Switched	Constant	2 Wire	NO	NO	Asynchronous
(7)	FDX	2561-1	RS232-103E Compatible	Dedicated	Constant	2 Wire	NO	NO	Asynchronous
(8)	HDX	2561-1	RS232-202S Compatible	Switched	Controlled	2 Wire	YES	YES	Asynchronous
(9)	FDX	2561-1	RS232-103E/113 Compatible	†† Switched	Constant	2 Wire	NO	NO	Asynchronous
(0A)	RESERVED →								
(0B)	RESERVED →								

†Operating with HDX Protocol
††Pseudo dial in - does not require ring, only DCD + DSR.

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	BSC
5	BZOWNER	Node ID of Owning Host	0†	0†	0†	0†
21	BZLNSPD	Line speed index	-	0-8††	-	-

† Required for configuration.
†† Required if autorecognition not specified.

TABLE C-6. LINE SPEED INDEX TABLE

Index	Baud Rate
0	110
1	134.5
2	150
3	300
4	600
5	1200
6	2400
7	4800
8	9600

This field only required if Autorecognition is not specified.

TABLE C-7. TCB FIELD NUMBER (FN)/FIELD VALUE (FV) ASSIGNMENTS

Field Number	NPU Mnemonic Name	Description	Values			
			Mode 4 TIP	Async TIP	HASP	BSC
5	BSTCLASS	Terminal Class†	10	1	9	7-8
12	BSOWNER	Node ID of Host††	0	0	0	0
13	BSCN	Connection Number††	1-255	1-255	1-255	1-255
14	-	Destination Node (NPU)††	1-32	1-32	1-32	1-32
15	-	Source Node (Host)††	0	0	0	0
16	BSNBL	Network Block Limit	1	1	1	1
19	BSIPRI	Input Priority	1-3	1-3	1-3	1-3
28	BSPGWIDTH	Page Width	0-255		0-255	50-150
30	BSXBLKLL	Transmission Block Length Least Significant	0-255		0-255	0-255
31	BSBLKLM	Transmission Block Length Most Significant	0-7		0-7	0-7
32	BS2629	026/029 Code Option	0=29 0=26		0=29 0=26	0=29 0=26
33	BSNUMR	Number of Records per Block				2/7
34	BSSUPCC	Suppress Carriage Control	0=N/S 1=S		0=N/S 1=S	0=N/S 1=S
35	BSBAN	Banner On/Off	0=ON 1=OFF		0=ON 1=OFF	0=ON 1=OFF
36	BSEM	"EM" at end of card for short records			0=No EM 1=EM	0=No EM 1=EM
37	BSCODE	TIP Code	1-3†††			

NS = No Suppress; S = Suppress

† Required for reconfigure SM; cannot use for configure.

†† Required for configure TCB SM (these fields must be ordered 14, 15, 13, 12).

††† Required only if not autorecognition on a configure TCB SM, 1=Mode 4A BCD; 2=Mode 4A ASCII; 3=Mode 4C.

BLOCK PROTOCOL SUMMARY D

BLOCK TYPES

The unit transmission between the host and the NPU is referred to as a block. It is never more than 2047 bytes in length, including block header information. The actual length of a block is a function of the type of source transmitting the data.

The block flow control interface between two logically connected processes can be envisioned as two simultaneously active communications paths. The procedure is fully symmetric. Sequence is maintained on each of the four paths but not between the separate paths.

The types of traffic which exist on each communications path consist of the following:

- Forward data (FD) - Textual information sent from a transmitter directly to a remote receiver. These blocks are either data or command blocks.

- Reverse supervision (RS) - Answer back blocks sent from the receiver in response to receipt of forward data or forward supervision. These blocks can be generated and sent even when not solicited under certain local abnormalities at the receiver.

Every block has a header consisting of four bytes. The first three bytes provide the network address. The last four bits of the last byte indicate the block type; the other four bits in this byte are reserved for network use as a modulo 8 block serial number. The contents of the remainder of the block, if any, vary with the block type. Because the header portion of all blocks has the same format, it is omitted from all block formats shown in this appendix.

The network address consists of a source node number, a destination node number, and a connection number. The node numbers identify the originating and terminal process locations for the block and identify the path direction. These node numbers correspond to the node ID numbers displayed on the NPU console at initialization. The connection number identifies the set of communication paths between the nodes that comprise the logical connection over which the block is transmitted. For data blocks and reverse supervision blocks, the connection number is a nonzero value identifying the terminal control block (TCB); separate connection numbers are assigned to each interactive data stream, upline batch data stream, and downline batch data stream for a terminal.

FD blocks that contain commands are a separate block type from other FD blocks, and are called command blocks. A subset of these command blocks are called service messages. Service message blocks have a connection number of zero in the header, identifying the logical connection called the service channel. Unlike logical connections with nonzero connection numbers, which can be dynamically created and destroyed, the service channel always exists. Blocks traveling via the service channel establish other logical connections, and communicate control, status, and error data in support of the common equipment and software which service the other logical connections. Blocks traveling via the service channel have a block serial number of 0 in their block header.

The relationships of block type values in the block header, traffic type, and block function are shown in table D-1.

TABLE D-1. BLOCK TYPES

Block Type	Mnemonic	Name	Traffic Type	General Function
1	BLK	Block	FD	This block is a data block which is not an end-of-message block of a multi-block message.
2	MSG	Message	FD	This block is a data block which is the end-of-message block of a multi-block message or all of a single block message.
3	BACK	Block acknowledgment	RS	This block is an acknowledgment for a block transmitted in the opposite direction.
4	CMD	Command	FD	This block is a service message on connection number 0 or a command on any other connecton number.

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 0 to 2039 bytes of data immediately following an 8-byte header. The 8-byte header consists of the standard 4-byte header described previously, plus four additional bytes of information describing the subsequent data. The contents of the data field are determined by the communicating processes. In CCI, a BLK block does not contain an EOR or EOI code.

MSG BLOCK

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

If a message is 2039 bytes or less in length, it can be transmitted via a single MSG-type block. If a message is longer than 2039 bytes, or if for some other reason it is desired to segment the message, all segments but the last are transmitted via BLK blocks, and the last segment is transmitted via a MSG block. In CCI, each block containing an EOR or EOI code is a MSG block.

BACK BLOCK

A BACK block is sent from the receiver to the transmitter to allow the transmitter to adjust the rate of issue of data to the delivery rate of the receiver. The transmitter should not issue more than one unacknowledged block for each connection. The BACK block, which acknowledges a previously transmitted block, allows the transmitter to maintain an outstanding block count to ensure that the allowable block limit is not exceeded. The BACK block contains no information other than the block holder.

When the NPU software detects a bad block (any block with fields that contain unexpected or undefined information), the NPU discards the block. If the block is a BLK or MSG, no BACK is sent to the host. For any other block type, no action solicited by the block is taken and it is not acknowledged. An NPU statistics word for a block discarded due to bad address condition is incremented.

CMD BLOCK

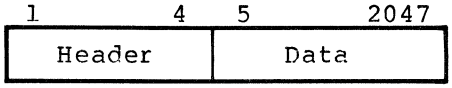
Command blocks allow connected processes to communicate outside of the data stream but synchronous with that stream. Command blocks are received by the destination process in the same ordering sequence to the data stream or other commands as existed at source.

The contents of a CMD block are bilaterally defined by the communicating processes. A CMD with a connection number of 0 has special significance as a service message.

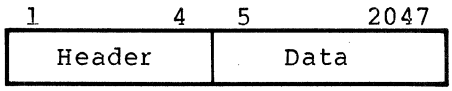
CMD blocks are also used on nonzero connections to control data streams, control terminal nodes, and to report status between the host and the NPU. CMD blocks can use either the reverse supervision or forward data channels of a connection, depending on the type of command. In either case the CMD block flows asynchronously to any data block on the channel and, therefore, cannot be used to mark position within the data stream. Commands received on a connection are not acknowledged by a BACK block in the reverse direction. Figures D-2 through D-10 summarize the general format of CMD blocks used on nonzero connections within CCI. In this table, the primary and secondary function code columns identify the contents of the first and second bytes of the block after the block header.

BLOCK TYPE

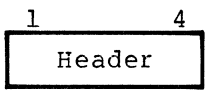
BLK = 1



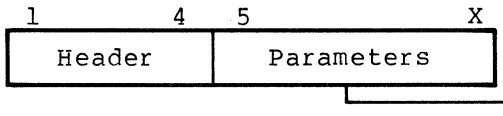
MSG = 2



BACK = 3



CMD = 4



Defined in appendix C.

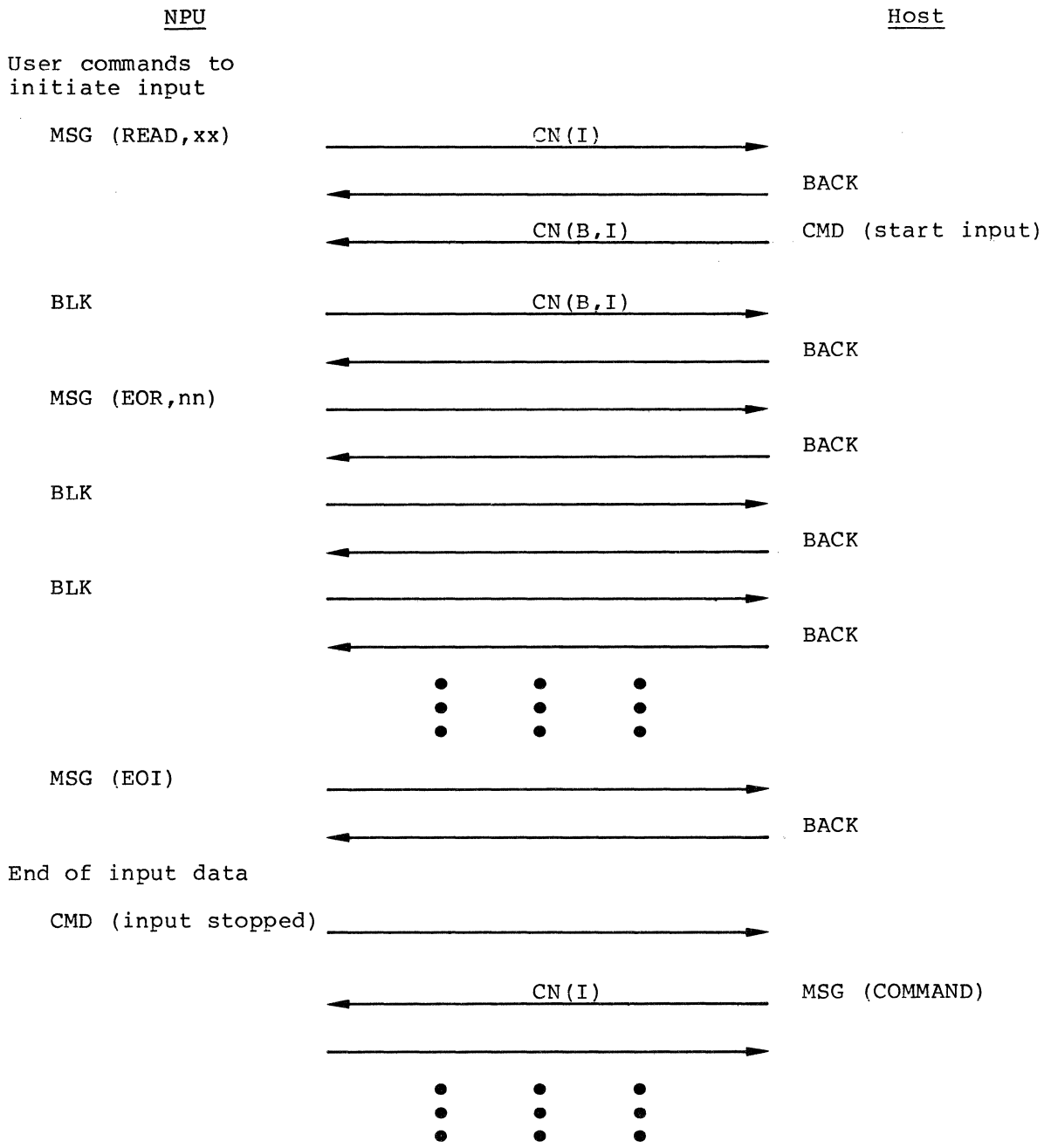


Figure D-1. General Batch Input Flow

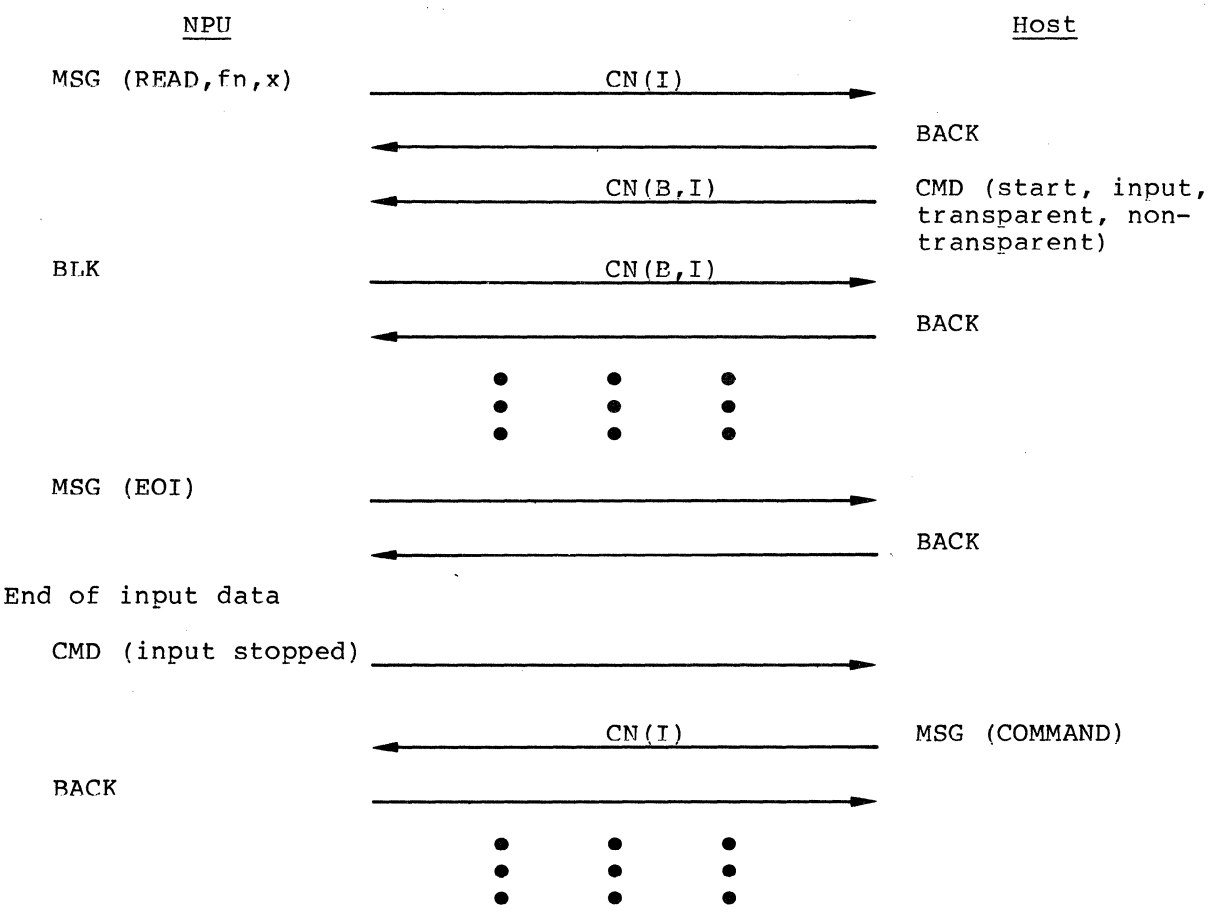


Figure D-2. INTERCOM READ, Filename, Mode Command

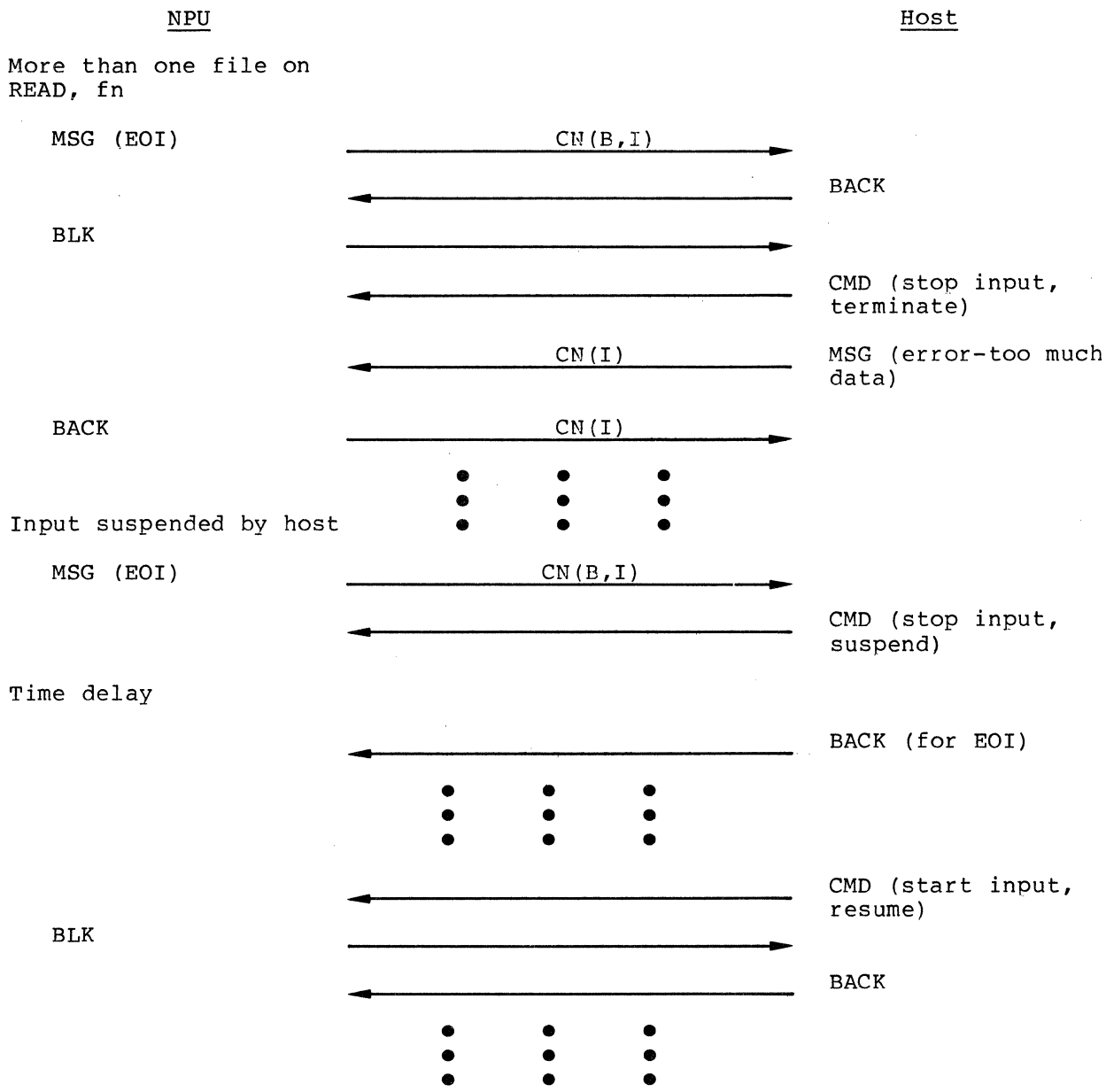


Figure D-3. Input Error Conditions (Sheet 1 of 2)

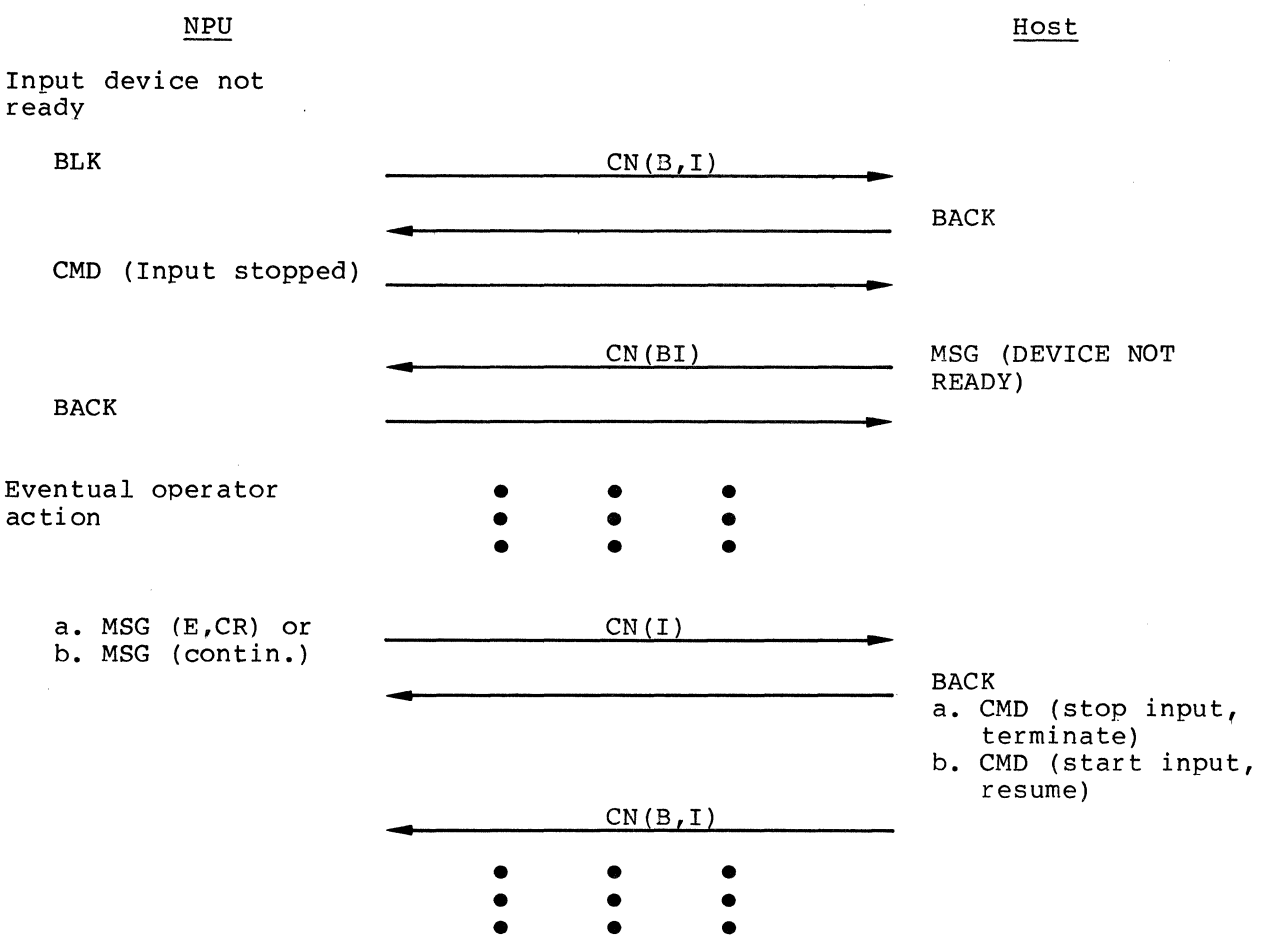


Figure D-3. Input Error Conditions (Sheet 2 of 2)

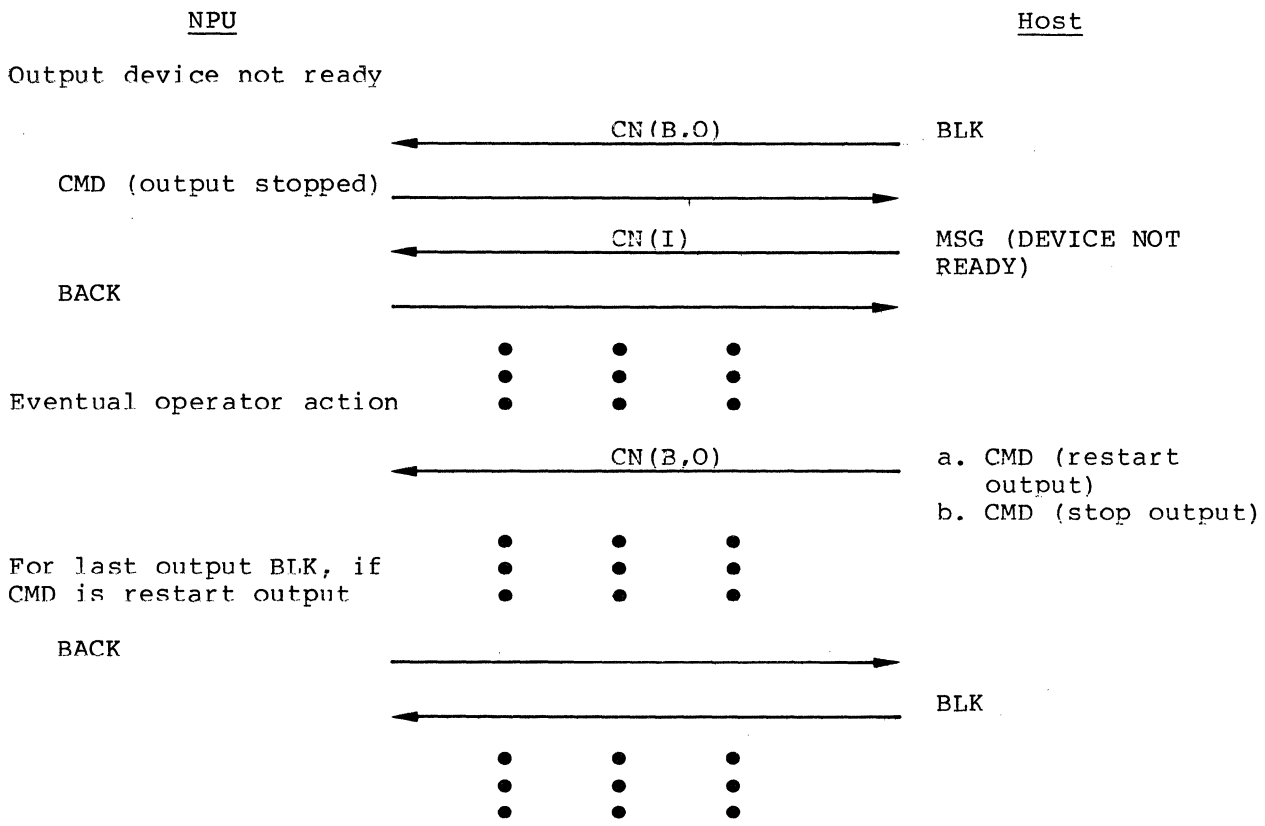


Figure D-4. Output Error Conditions

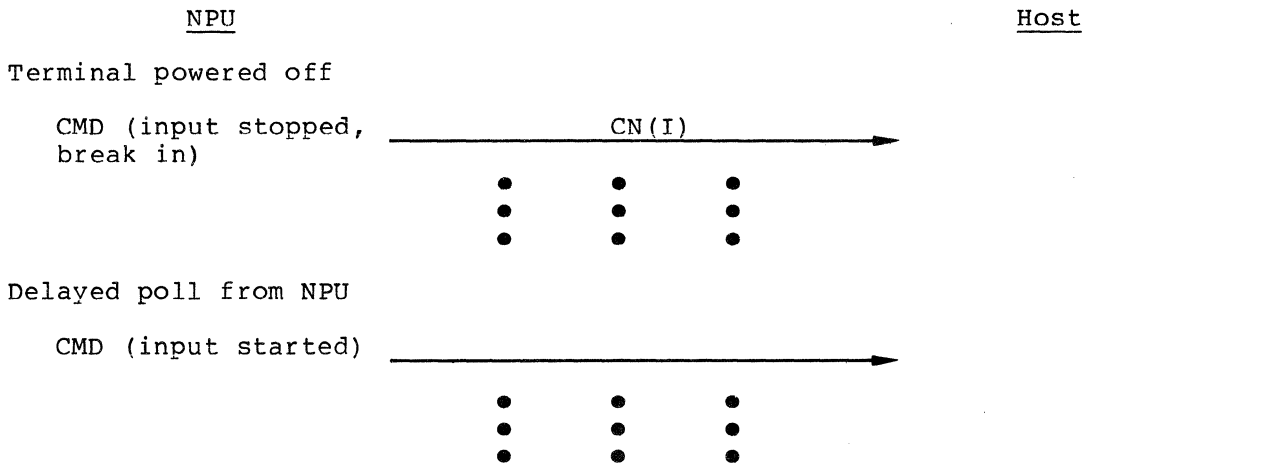


Figure D-5. General Errors

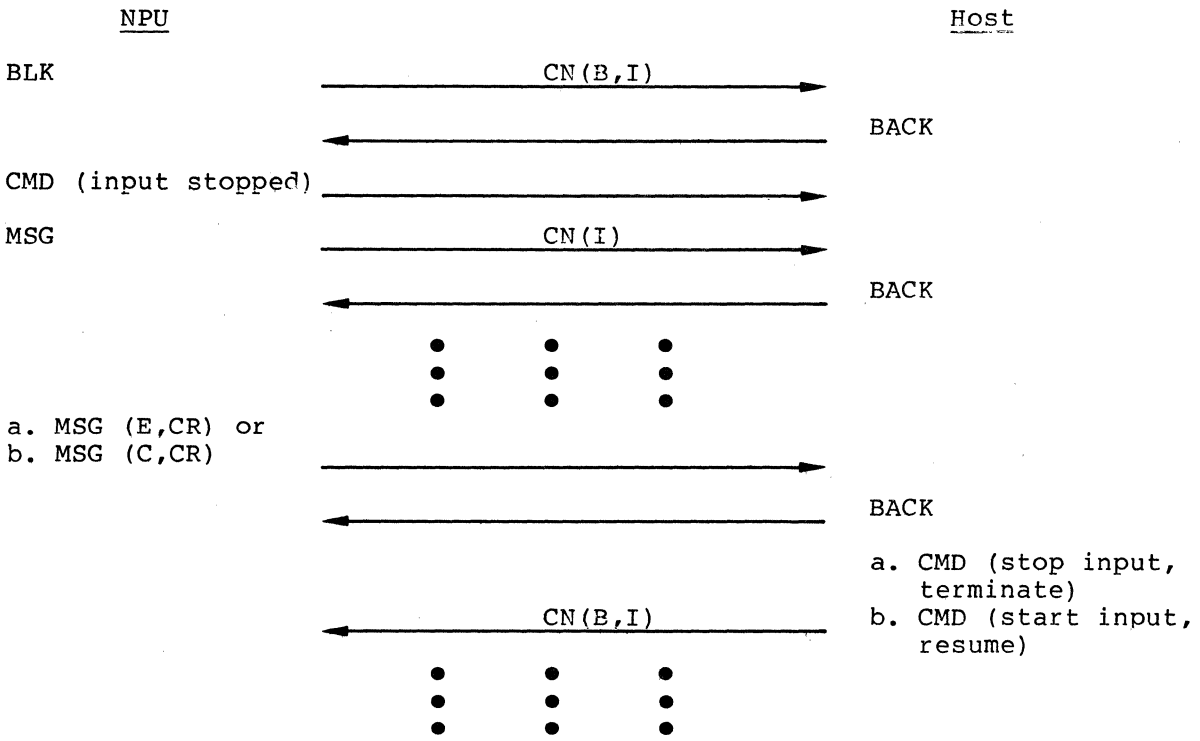


Figure D-6. Input Intervention

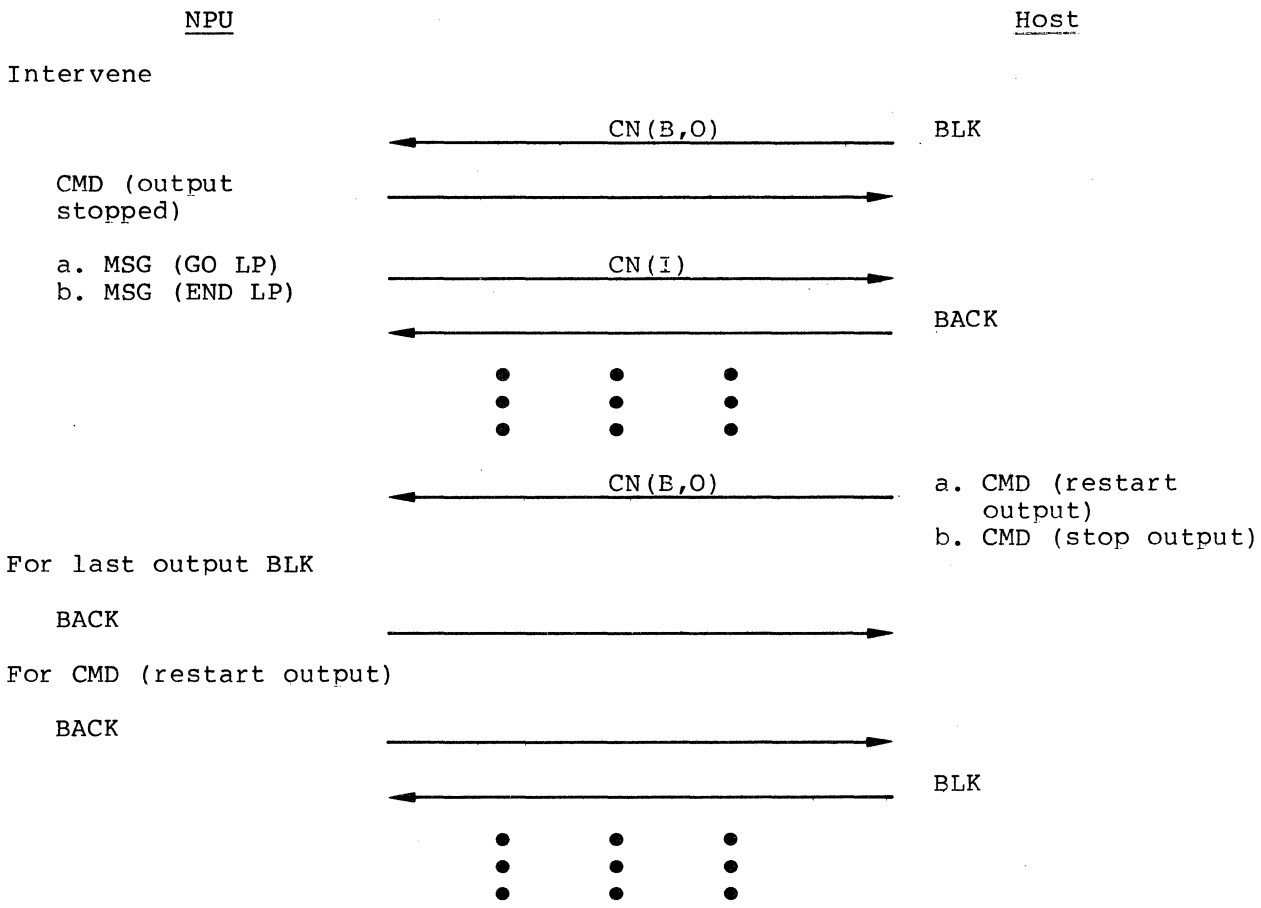


Figure D-7. Output Intervention

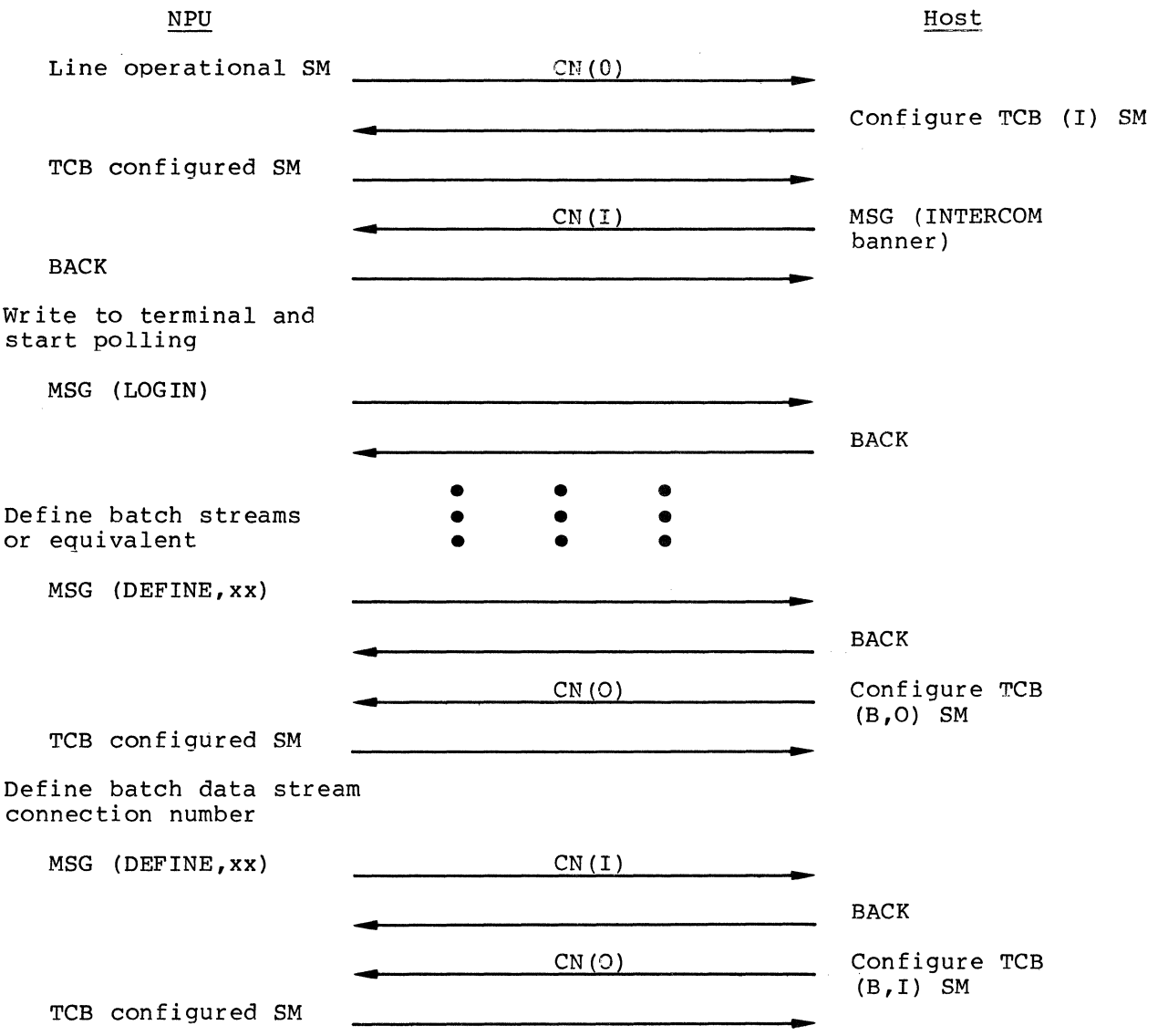


Figure D-8. Initialization

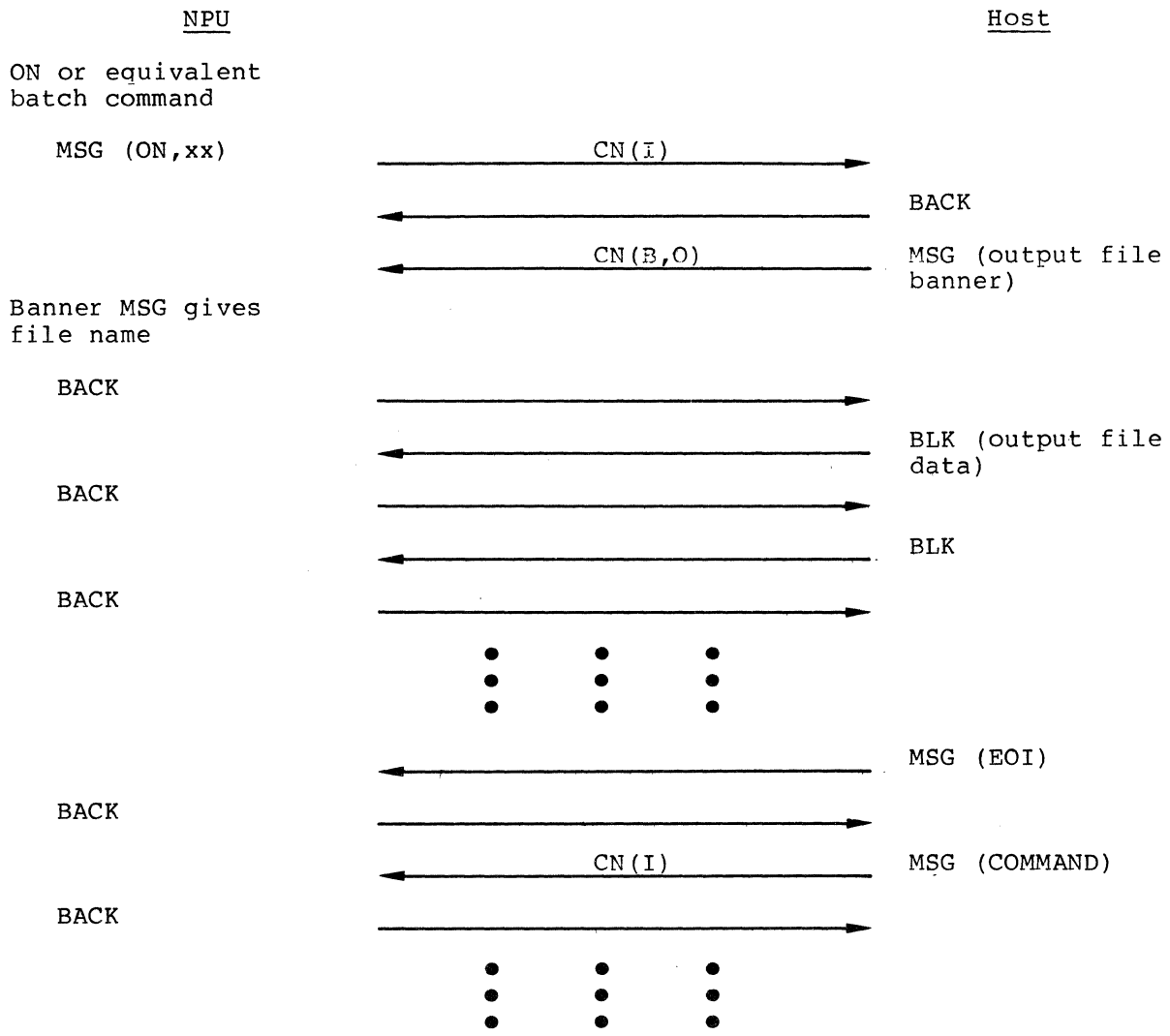


Figure D-9. Batch Output

cn	Binary channel number
en	Binary equipment number of NPU coupler
∅	Display-coded blank fill
h	Display-coded digit of hour, system clock time of dump
.	Display-coded period character
n	Display-coded digit of minute, system clock time of dump
s	Display-coded digit of second, system clock time of dump
m	Display-coded of month, system time of dump
/	Display-coded slash character
d	Display-coded digit of day, system time of dump
y	Display-coded digit of year, system time of dump

File registers format

uhw	upper half of word (character 1)
lhw	lower half of word (character 2)

SAMPLE MAIN MEMORY MAP FOR NPU

Figure E-1 shows the layout of CCI in the main memory of a 255x network processing unit with 65K words of main memory.

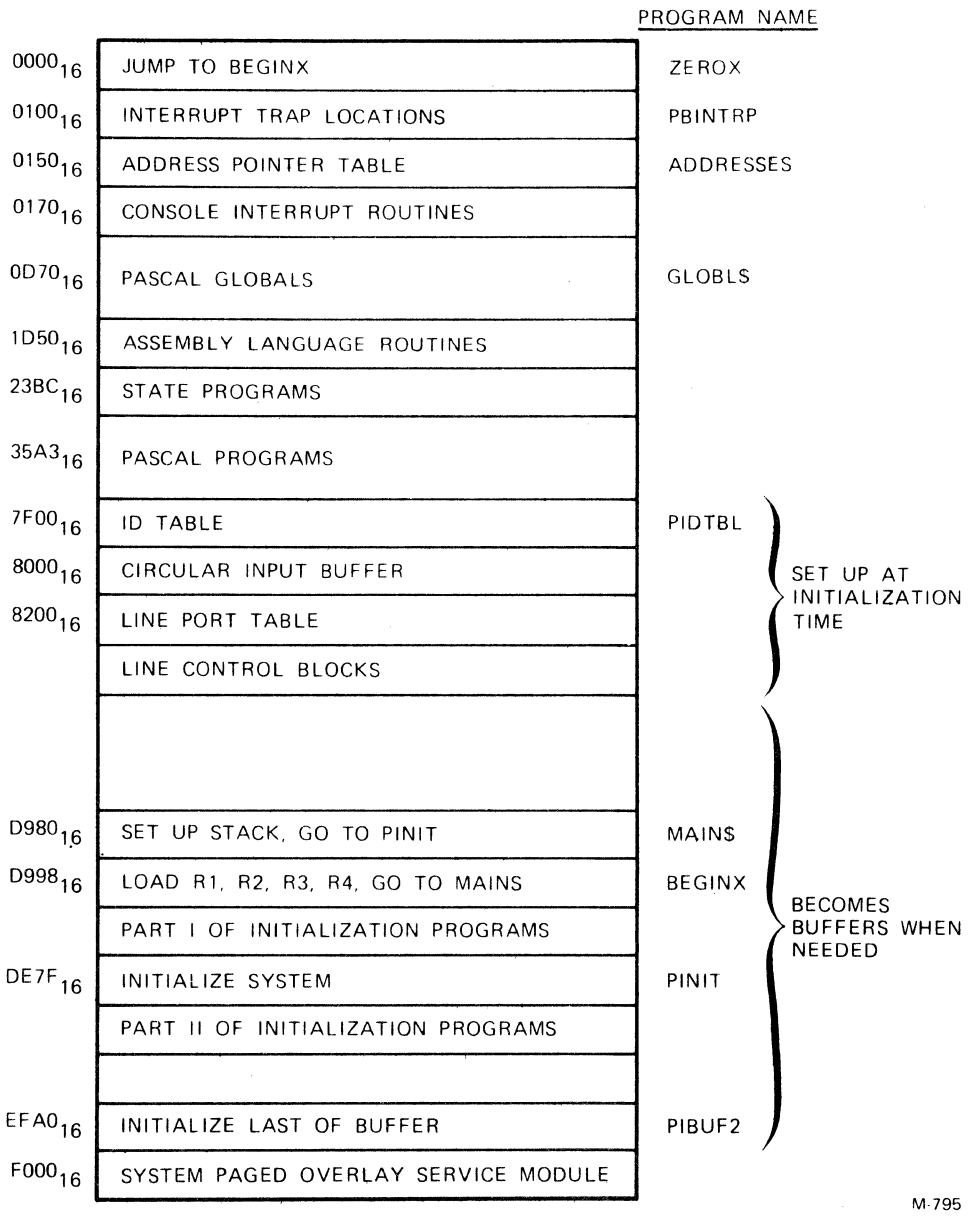


Figure E-1. Sample Main Memory Map

CCI NAMING CONVENTIONS F

The following naming conventions for the CCI 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

For types, variables, fields, and so forth:

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 appendix consists of five sections, one for each of the standard TIPs: Mode 4, TTY, HASP, BSC, 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 making up 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 the mux 2 interrupt routines.

Comparing these trees and TIPs can aid the TIP programmer in finding how other TIP programmers have solved similar problems.

The SVM section follows the TIP sections.

Conventions used are described as follows:

External calls are underlines. No effort is made to trace calls from external routines.

MODE 4 TIP ROUTINES

PTMD4TIP - Main TIP switch

Enable line
TCB build
Output queued
Disable line
Delete TCB
Input status workcodes
Autorecognition work codes
Line timeout workcodes
Hardware errors

PT4TCBINIT - Call from SVM to finish building TCB (also used to build autorecognition TCB).

PTTPMODE4 - Call from PBIOPOI to process downline PRU blocks.

PT4QIA - Call from internal processing to queue downline interactive blocks.

PTMD4MUX - Multiplex level-two entry.

PT4GOTOTASK - Sets next tasks for terminal in TCB.

PT4CONTROL - Determines TCB to get control, and Mode (4A or 4C).

PT4RETRYOVF - Counts errors; determines if more retries should be made.

PT4TOGTA - Retransmits write message if toggle bit problem exists.

PT4WRITE - Builds write E1/E4 message for keyboard or E3 message for CRT.

PTCHOQ - Checks output queue for tasks.

PT4CROFF - Handles card reader off action.

PT4TASKPROCESSOR - Processes next task for current terminal.

Device	Task
--	Output queues
Console	ACK after write
Card reader	Read after poll
Printer	Reject during poll
--	Error retries exhausted

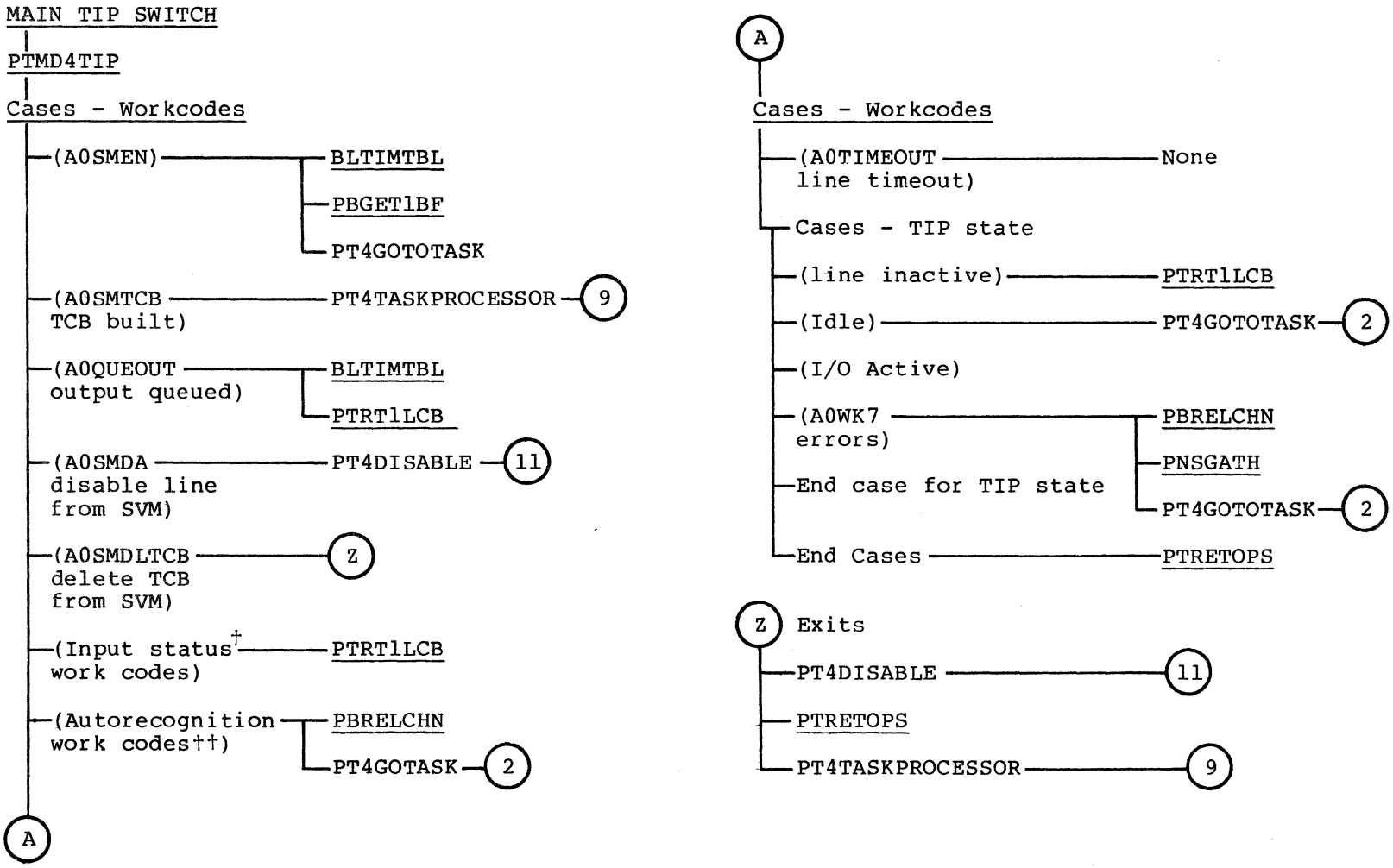
Autorecognition for terminal

I/O related tasks (in addition to device or task)

Output write message
Output write failure
Poll for E-code or toggle
Error responses to E-code or toggle polls
Good responses to toggle polls

PT4IO - I/O processor (calls PBCOIN).

PT4DISABLE - Disables line.



†(A0WK2 - 6 and 8)
††(A0WK9 - 11)

Figure G-1. Mode 4 TIP Trees (Sheet 1 of 6)

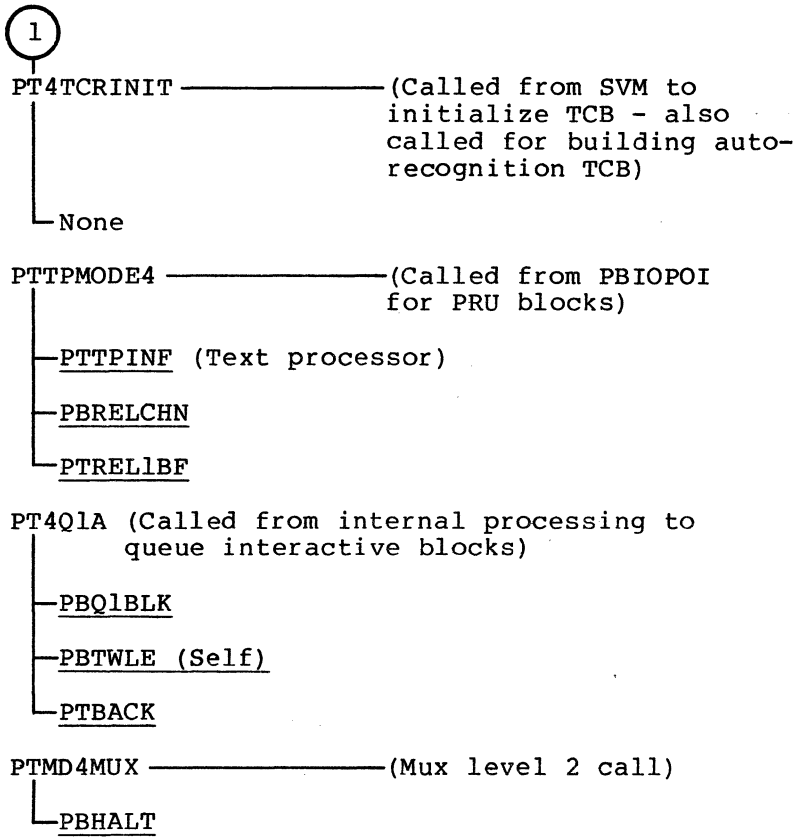


Figure G-1. Mode 4 TIP Trees (Sheet 2 of 6)

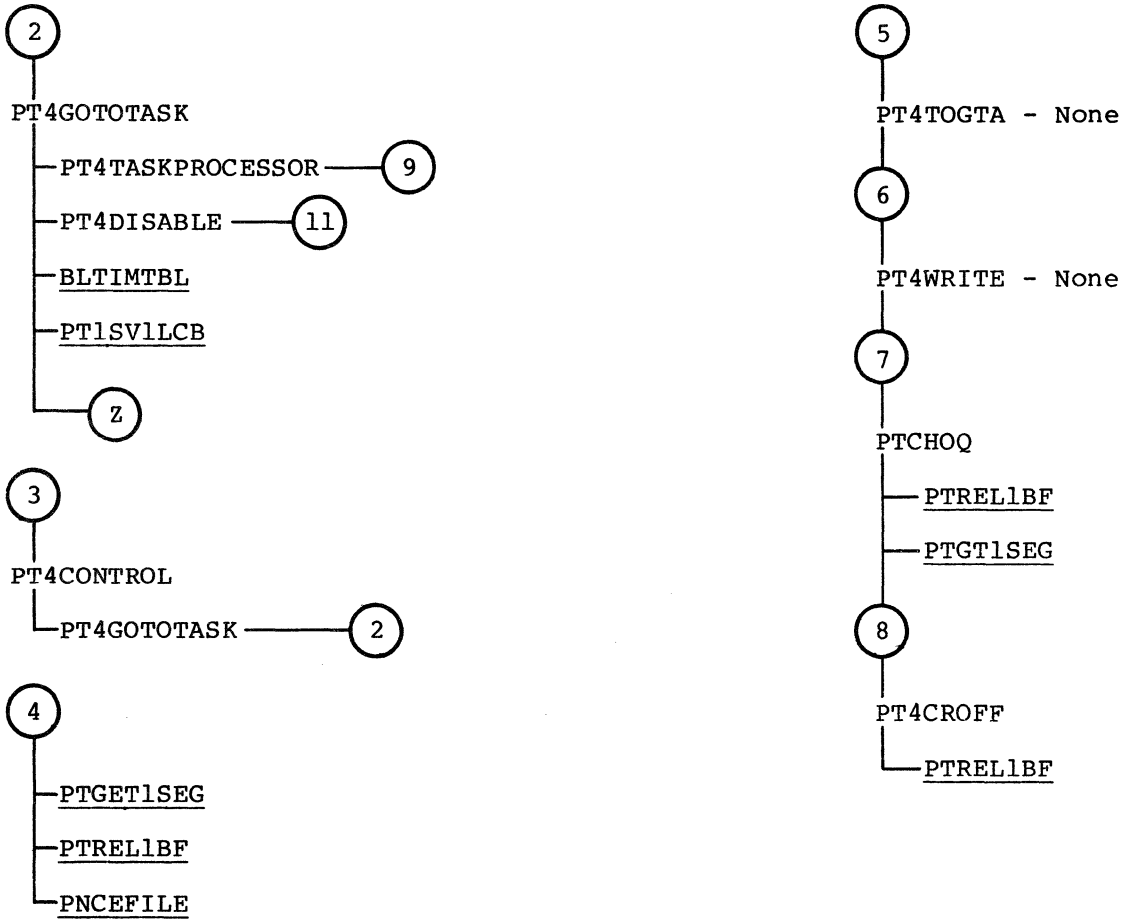


Figure G-1. Mode 4 TIP Trees (Sheet 3 of 6)

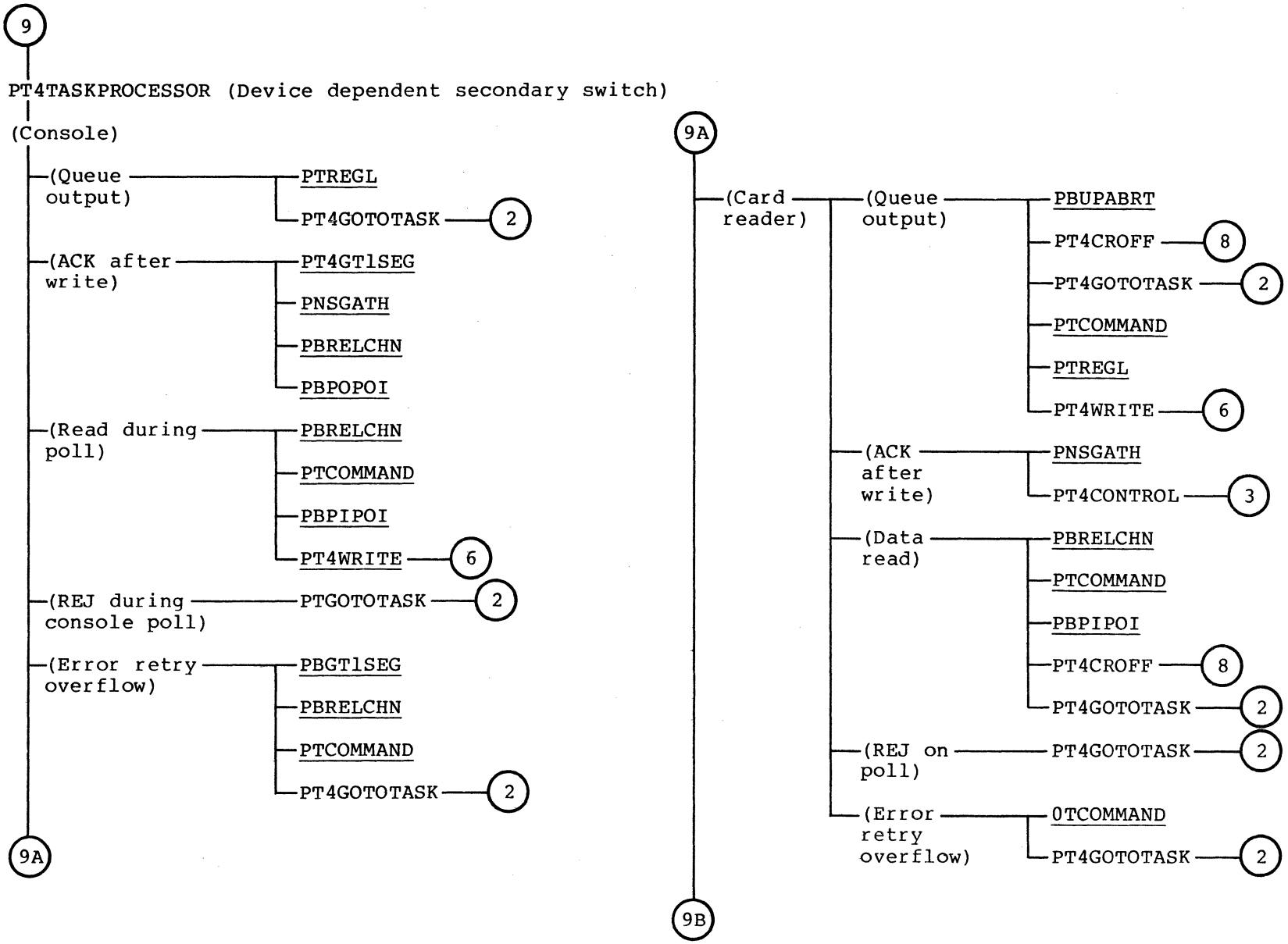


Figure G-1. Mode 4 TIP Trees (Sheet 4 of 6)

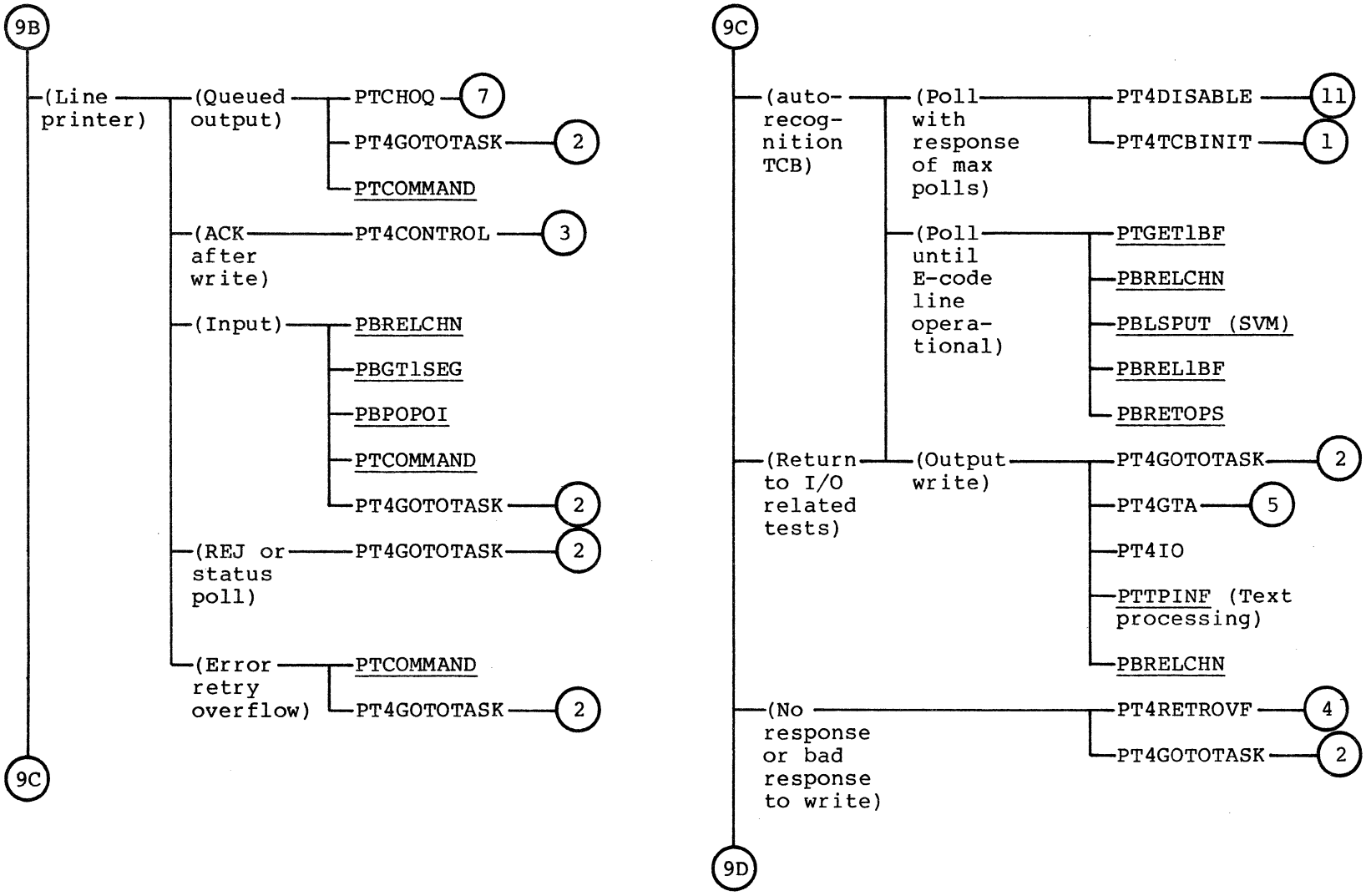


Figure G-1. Mode 4 TIP Trees (Sheet 5 of 6)

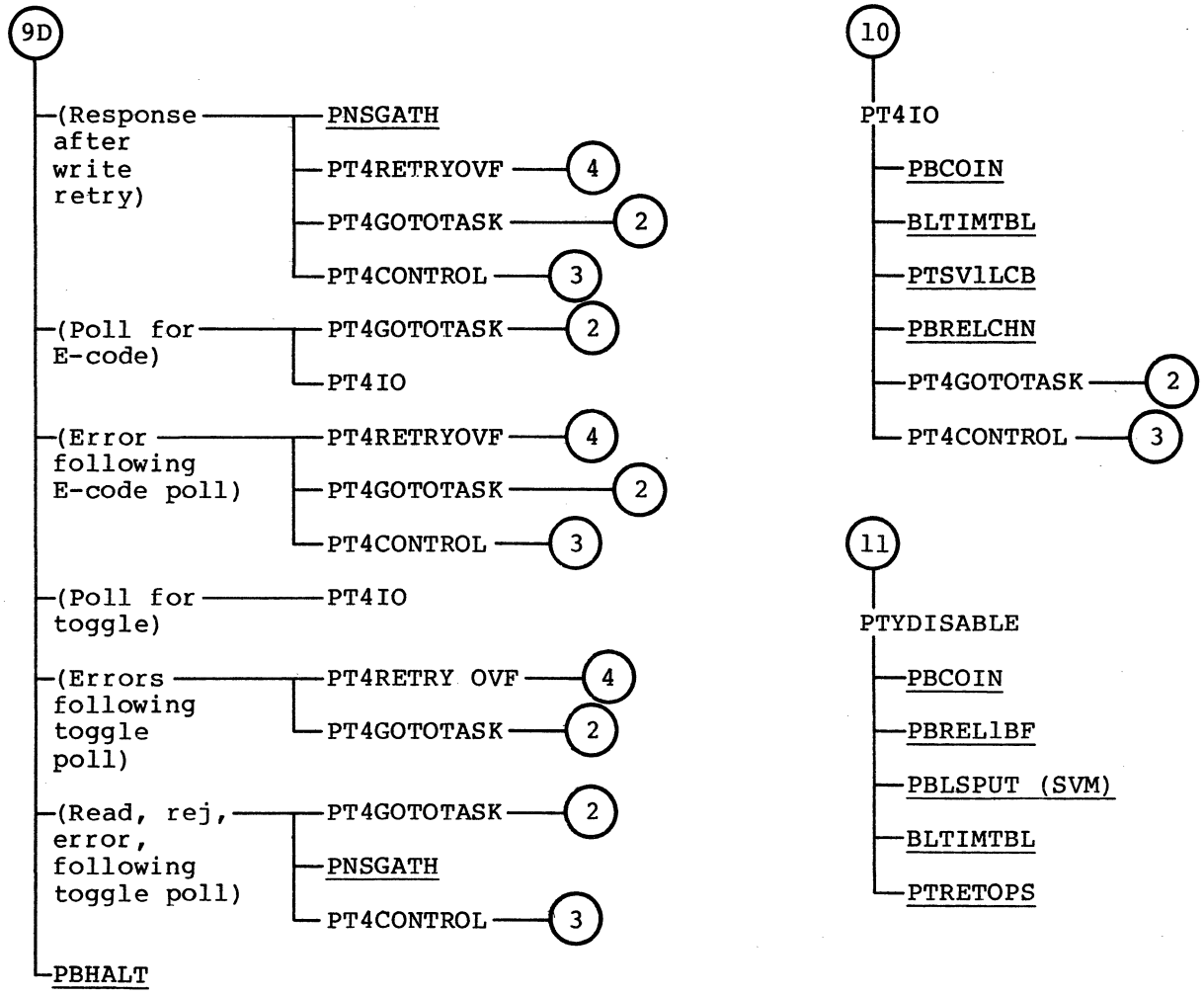


Figure G-1. Mode 4 TIP Trees (Sheet 6 of 6)

HASPTIP

PTSMUXTIP - Mux WC. Converts mux 2 level worklist to OPS level HASP TIP worklist

PTHSOPSTIP - OPS-level entry. Processes worklists from OPS-level (main HASP process).

Workcodes recognized are as follows:

(AOSMEN) - Enables line (sets LCB fields).

(AOSMDA) - Disables line.

(AOSMTCB) - Checks for an ENQ block; processes transmission.

(AOSMDLTCB) - Terminates and releases TCB, passes terminate command to command driver, notifies host.

(MSGCONT) - Sets up 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/ - Purges record.

(AOTIMEOUT) - Timeout handler.

(AOQUEOUT) - Output handler.

(MSGCMPLT) - Message completed, returns to caller.

(ERROR) - Releases buffer; returns to caller.

(ENQACK/NAK) - Sets completion value, returns to caller.

(NMINDEND) - Ends input, returns to caller.

(AOHARDER) - Hardware error, set inop code and return to caller.

(BUFTHR) - No buffers (threshold reached), drops message.

NAKTEST - If NAKs received after I/O, marks line down.

FINDTCB - Finds TCB for stream (upline TCB).

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.

GETBCBFCS - Sets up BCB and FCS for output.

PTTHASP - Text processing; calls PTTPINF.

GENDATA - Sets up buffer prior to PTTPINF call.

HSPREL - Releases data buffers.

WRAPUP - Cleans up data transfers 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; marks 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).

HSPTCBUILD - Initializes TIP dependent TCB fields; directs call from the SVM during configuration of terminal.

POSTINPUT - Prepares to send block to host via PBPIPOI.

HSPTPINP - Prepares for input text processing.

DSTCHAIN - Forms queue chain.

CORBUILD - Prepares EOR block.

EOIBUILD - Prepares EOI block.

PRBLOCKTYPE - Determines block type: EOI, EOR, mag, banner.

TP1BUILD - Prepares a text processing buffer for interactive messages.

TP2NDPASS - Upline text processing for card printer.

TP1STPASS - Downline text processing for punch or printer.

TPINTERACTIVE - Upline/downline text processing for console.

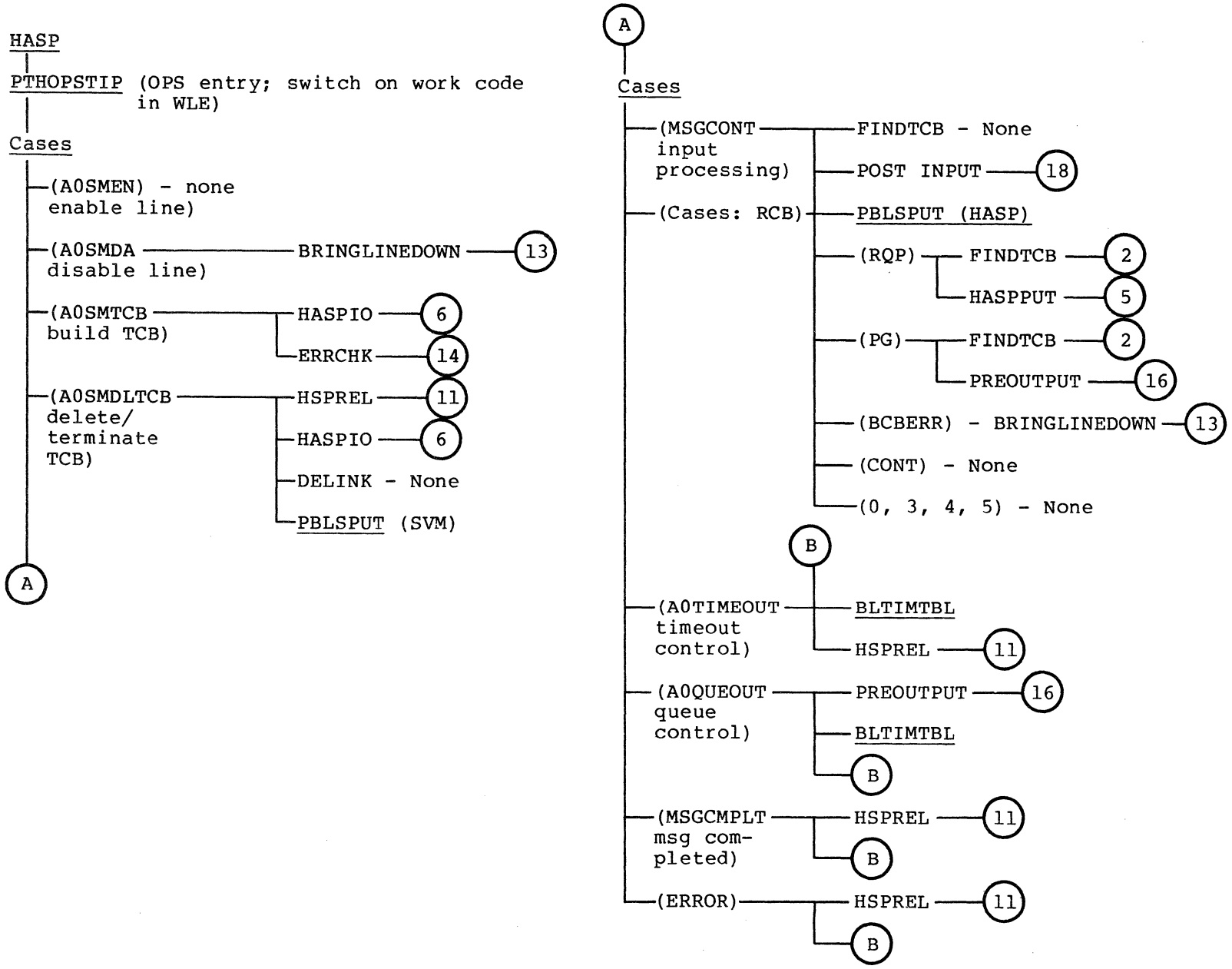
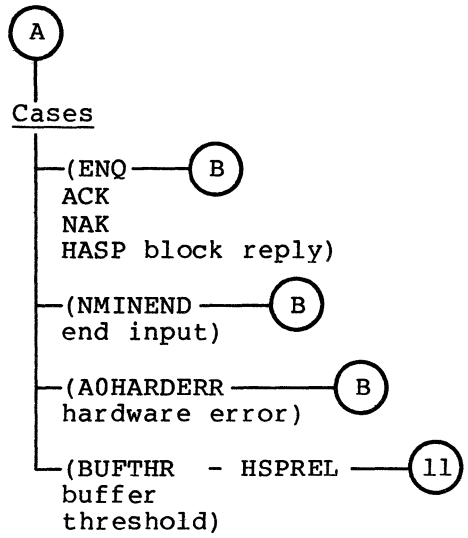
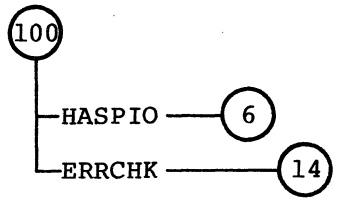
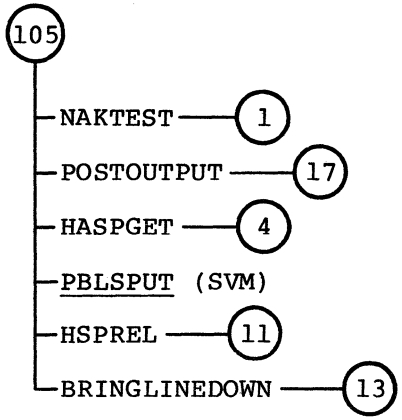


Figure G-2. HASP TIP Main Switches (Sheet 1 of 6)







End

PTHSMUXTIP (Mux 2 interrupt entry)

- PBRELZRO
- PBLSPUT (HASP)

HASPTCBUILD (Direct call from SVM)

- None

PTTPHASP — 28

(Internal call for upline text processing. Also called externally from PBPIOPOI for downline text processing.)

Figure G-2. HASP TIP Main Switches (Sheet 2 of 6)

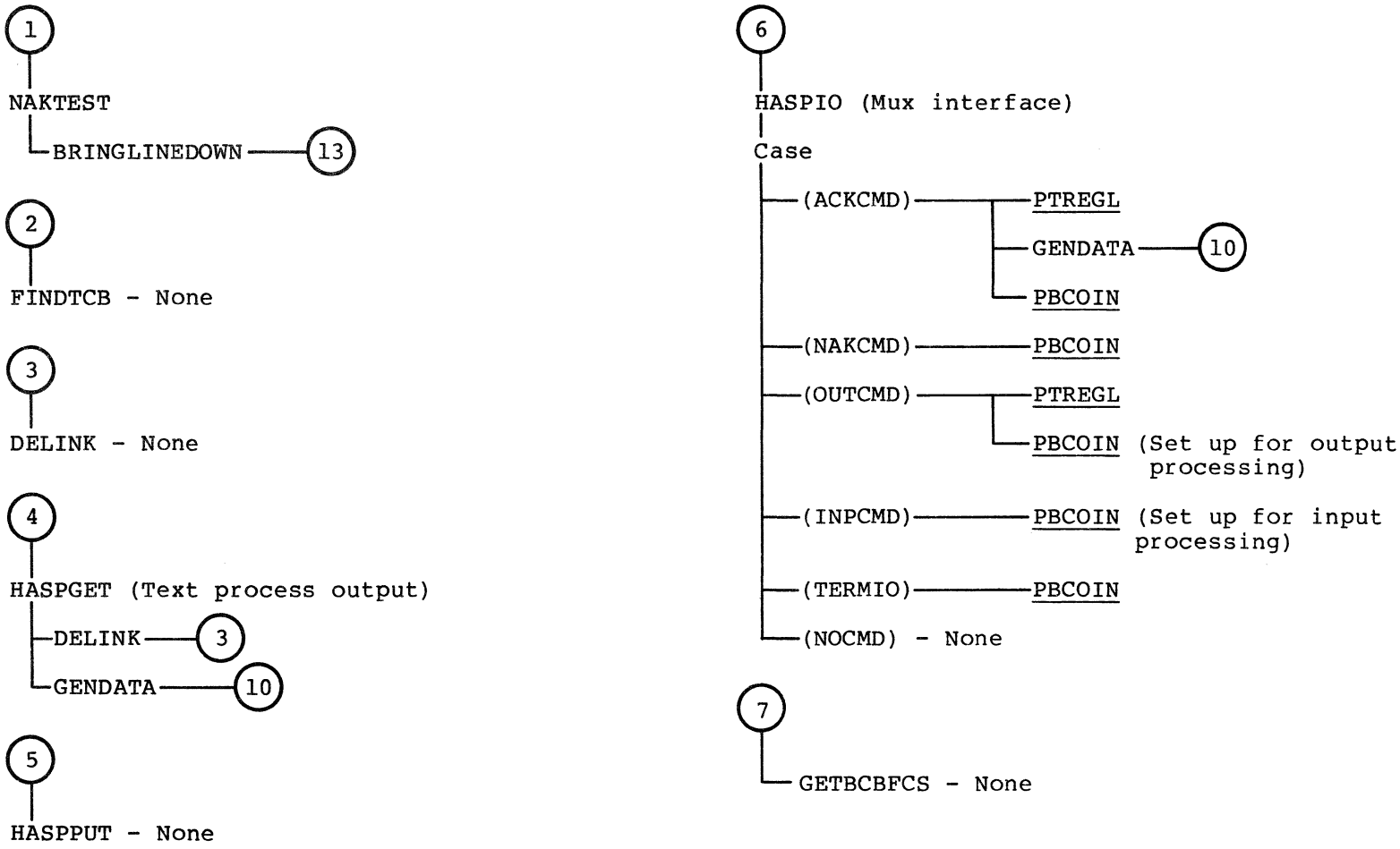


Figure G-2. HASP TIP Main Switches (Sheet 3 of 6)

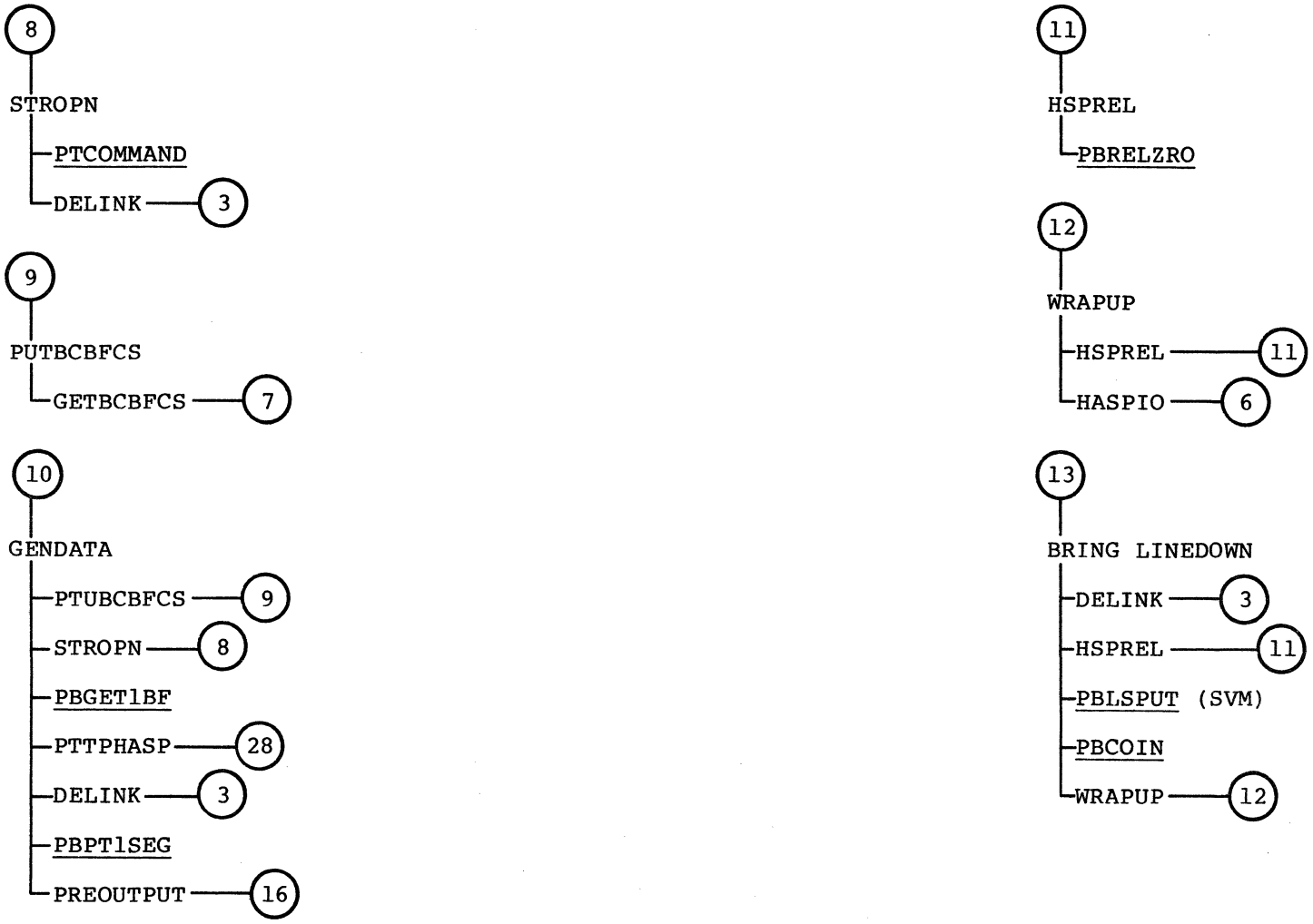


Figure G-2. HASP TIP Main Switches (Sheet 4 of 6)

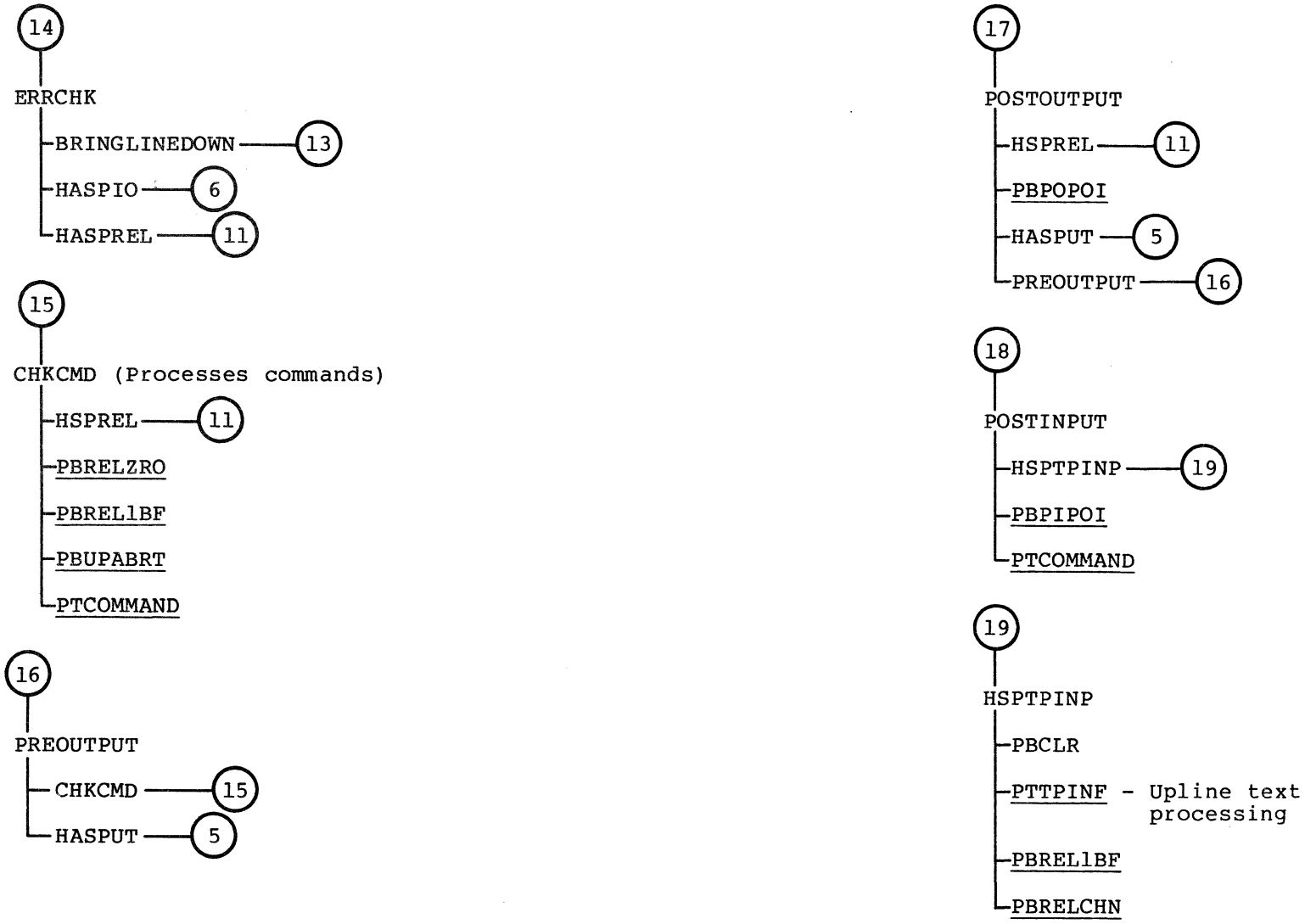


Figure G-2. HASP TIP Main Switches (Sheet 5 of 6)

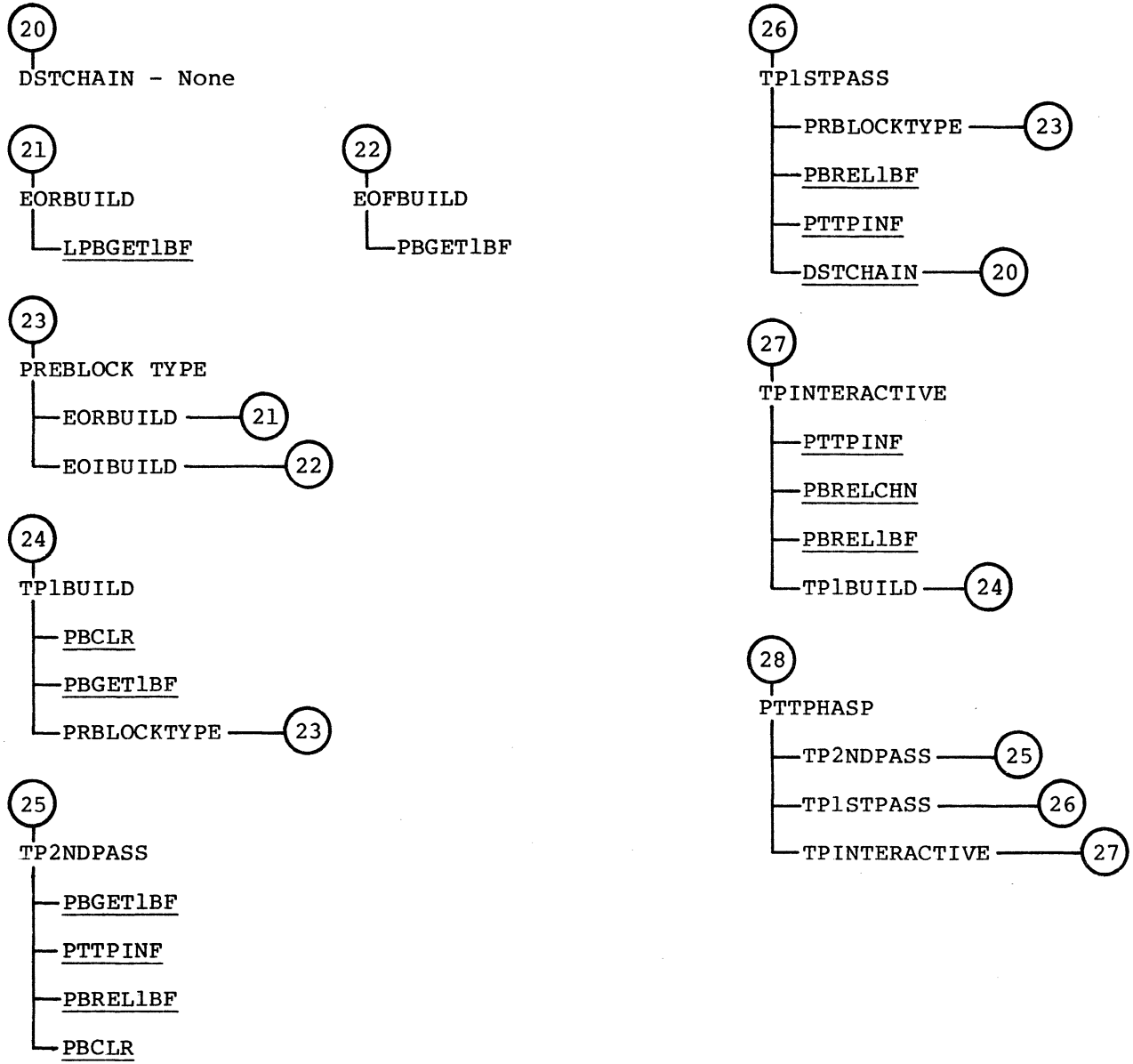


Figure G-2. HASP TIP Main Switches (Sheet 6 of 6)

TTY TIP ROUTINES

PTMSQUE - Adds entries to the 100 ms timeout queue.†
PTMSCAN - Scans LCB timeout queue for expired entries (100 ms).†
PTDELMS - Deletes entry from 100 ms timeout queue.†
PTTMUX2 - Mux level 2 entry; converts to OPS-level entry.
PTTYTIP - Main switch.

Worklists:

- Enable line
- TCB built
- Output queued task
- Disable line
- Delete TCB
- Autorecognition
- Start block
- Framing error
- LF message
- CR message
- Delayed routine returns
- Paper tape message
- Paper tape turned off message
- Overflowed block size
- Mux buffer threshold reached
- Hardware error
- Output block transmitted

PTTYTCB - Calls from SVM to finish building TCB.
PTYCKQ - Checks output queue for tasks to perform.
PTYTERM - Sends terminate I/O command to multiplex subsystem.
PTYPASS - Passes data block to host.
PTYHANGUP - Stops activity on a line.

---

† These can be used by other TIPs also.

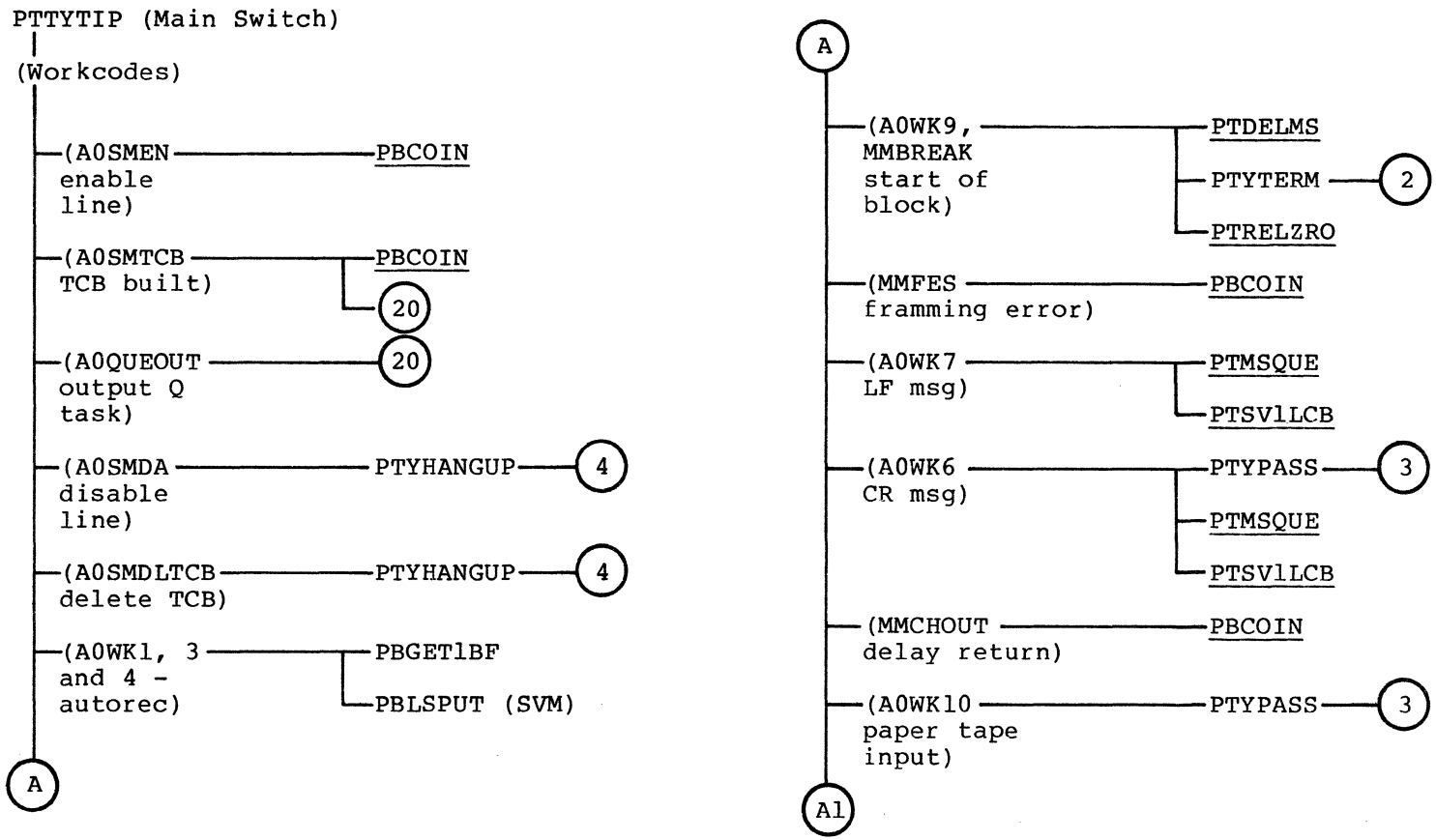
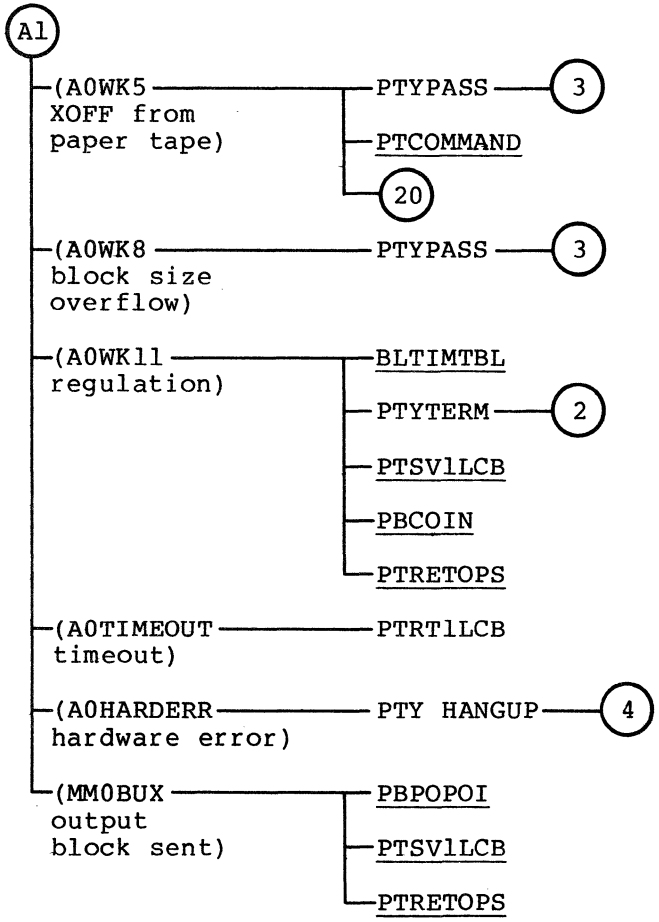
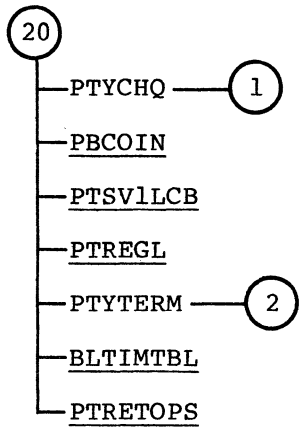


Figure G-3. TTY TIP Trees (Sheet 1 of 3)





PTTYTCB - Call from SVM to build TCB

- None

†PTMSQUE - Call from 100 ms timer

- PBCOIN

†PTMSCAN - Call from active (timed) LCB chain

- PBLSPUT (self)

†PTDELMS - Delete timing entry

- PBCOIN

PTTTYMUX2 - Call from mux 2 level

- PBLSPUT (self)

†Also used by other TIPs.

Figure G-3. TTY TIP Trees (Sheet 2 of 3)

Figure G-3. TTY TIP Trees (Sheet 3 of 3)

BSC TIP ROUTINES

PTIP780 - Entry/switch; main program.
RSTIMER - Set periodic timing.
ISSUECMDPKT - Call CMD driver with command packet.
SENDACK - Sends acknowledgment message.
SENDCMD - Sends command block to host.
SENDSVM - Sends service message to host.
TCBDELETE - Deletes specified TCB.
TERMINATE - Terminates input or output as requested.
ANYOUTPUT - Searches for output to process.
CMDQPROCESS - Processes all queued commands.
GOODACK - Processes good acknowledgment block.
INEOTSENT - Sends EOT to terminal.
INIDLE - Sets input state to idle.
INLINBID - Checks to see if line bids for input.
INTTDSENT - TTD sent.
NTMAXNAKS - Checks for maximum number of NAKs.
NTMAXTIM - Checks for maximum number of timeouts.
NTMAXWACKS - Checks for maximum number of WACKS.
OKTOINPUT - Sets up for input on next available TCB.
POSTINPUT - Post input processor (calls PBPIPOI).
POSTOUTPUT - Post output processor (calls PBPOPOI).
PRLNBID - Prepares line for next type of transfer.
PRNXTBLOCK - Prepares next block to be sent.
RCVABORT - Input message aborted; notify terminal.
RESPAUTOREC - Responds to autorecognition information from terminal.
RESPTOBLOCK - Responds to data or ENQBLOCK from host.
PRBLOCKTYPE - Determines block type.
EOIBUILD - Builds an EOI.
EORBUILD - Builds an EOR.
PTTP780 - Test processor.
PTTCB780 - Finishes building TCB for SVM.

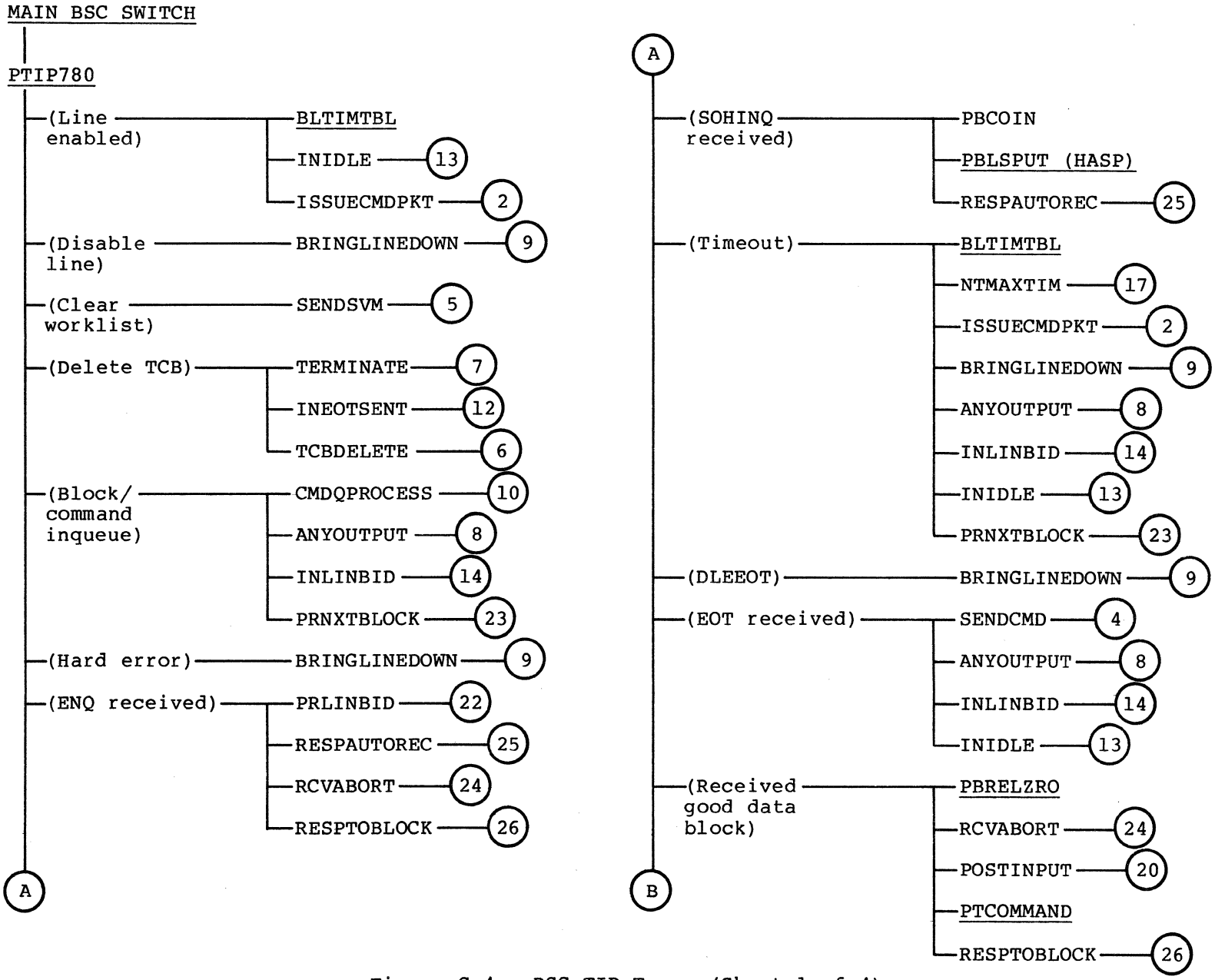


Figure G-4. BSC TIP Trees (Sheet 1 of 4)

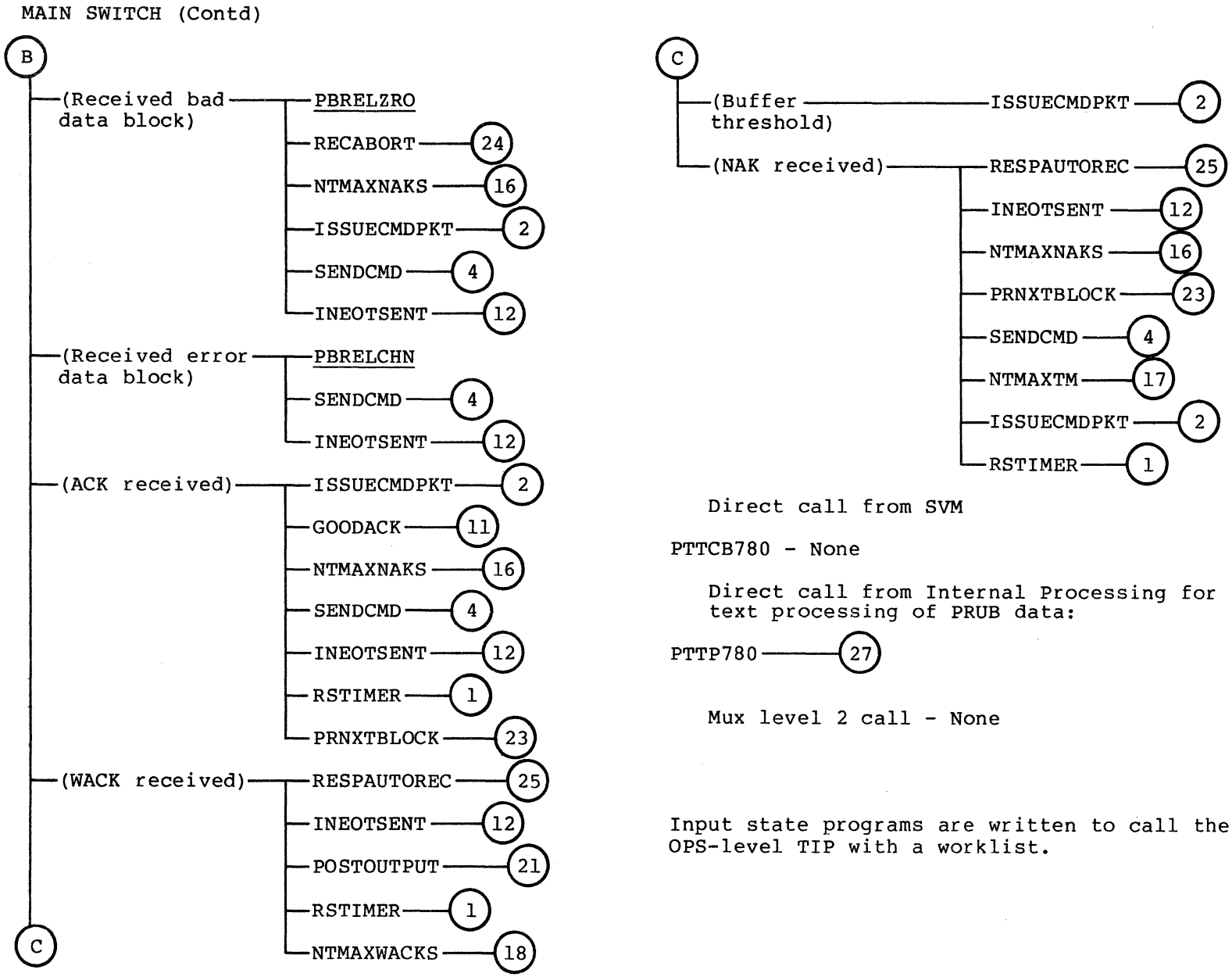


Figure G-4. BSC TIP Trees (Sheet 2 of 4)

Support Routines

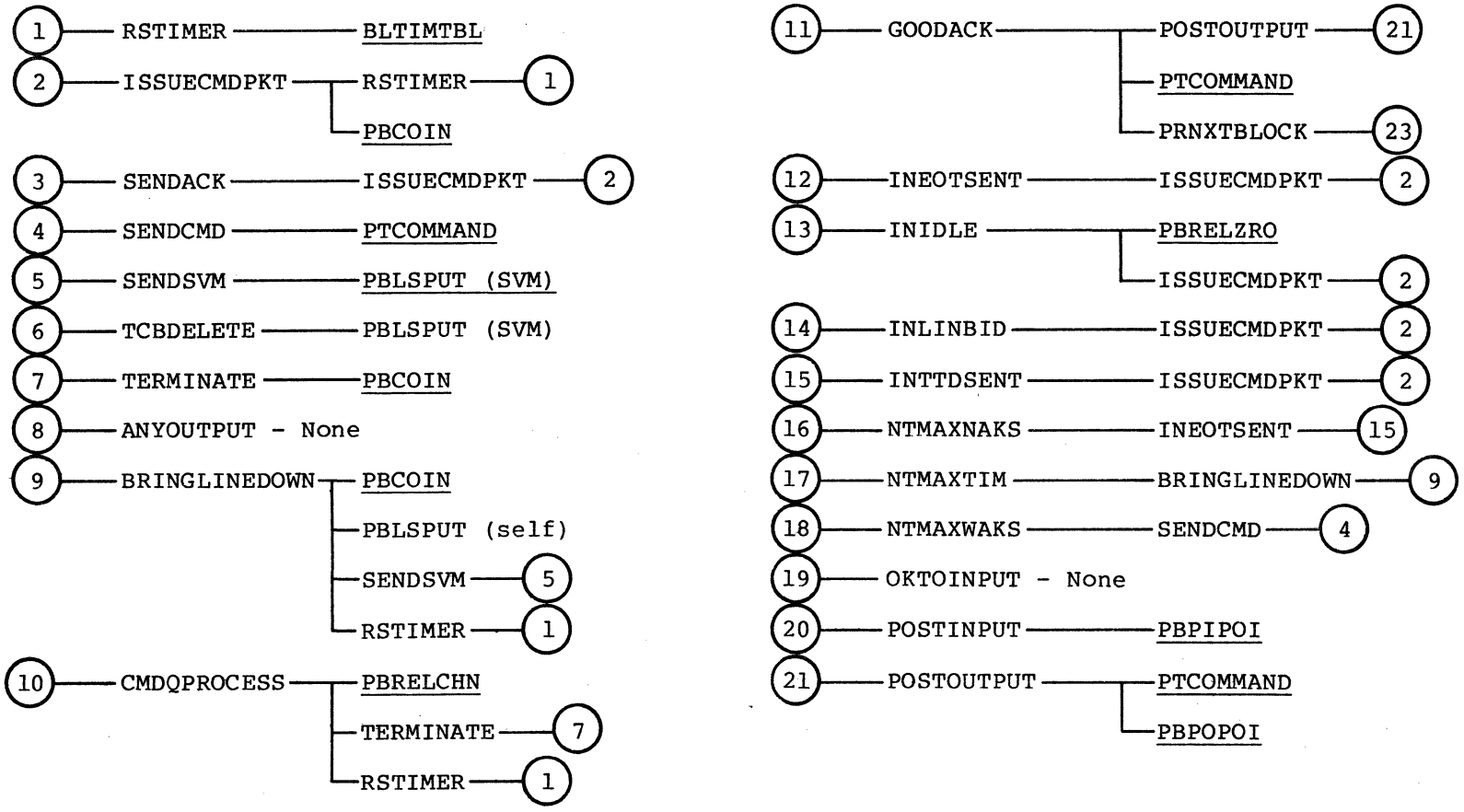


Figure G-4. BSC TIP Trees (Sheet 3 of 4)

Support Routines (Contd)

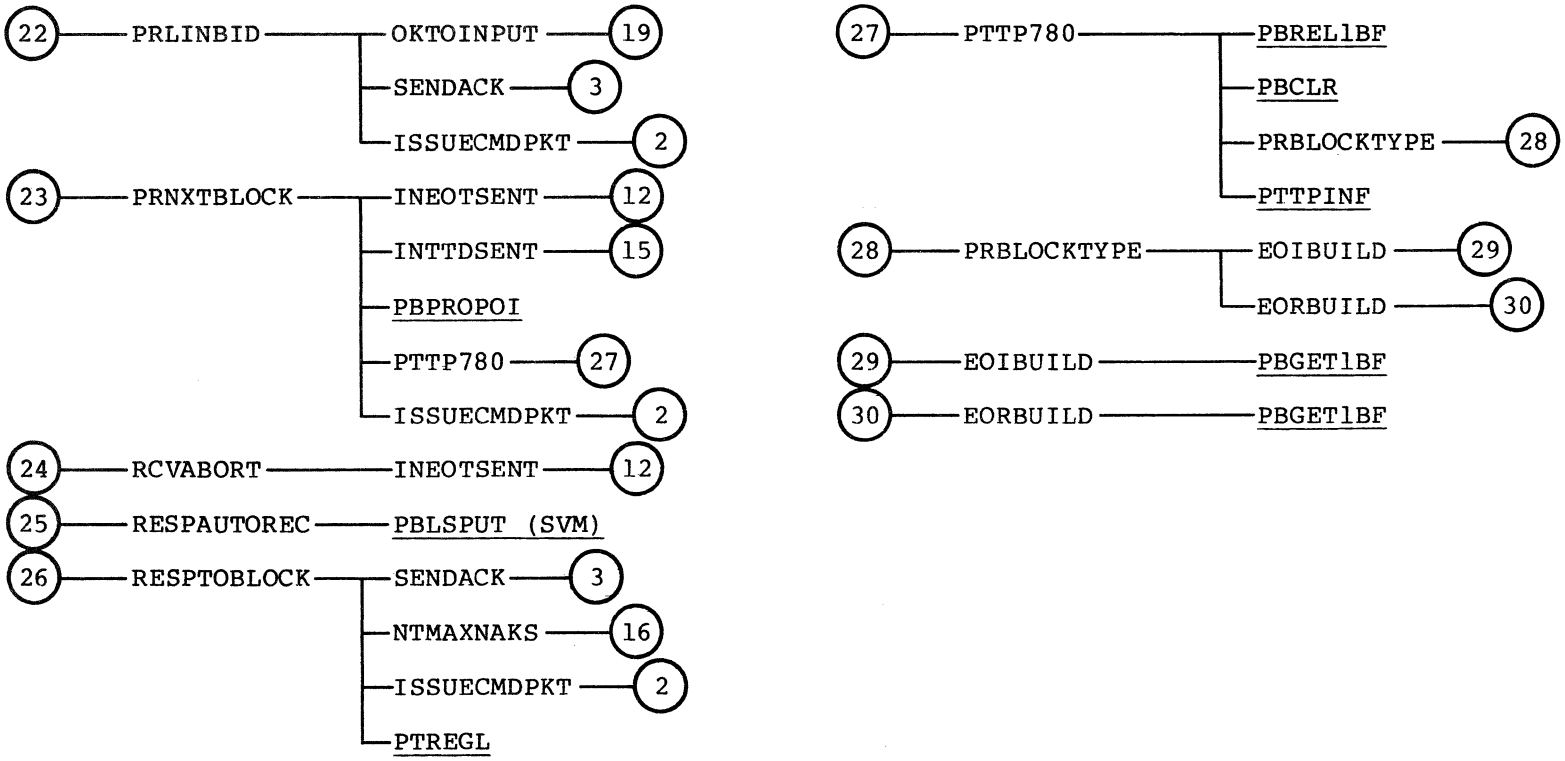


Figure G-4. BSC TIP Trees (Sheet 4 of 4)

SERVICE MODULE 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 that 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.) that failed.

See appendix C for format of individual SMs.

SVM Routines:

PNAWAIT - gives up control for external event.

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.

PNRCWAIT - 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 - sends/receives SM. This work code is the subswitch for the SVM handler table. (See table E-1.)

C02MDISP - calls PNSDISP to send an SM.

C0LINOP/C0LININOP - calls PNLINE to enable or disable a line.

C0LNDA - handles replies from TIP for line disable requests by the SVM.

C0DLTCB - handles replies from TIP for delete TCB requests from SVM.

C0OVLDATA - handles the overlay data SM.

C0BFR
C0DISABLE
C0ENABLE
C0TMLDLT
} 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.

PNCONFIGURE - common subroutine to process LCB or TCB configuration.

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 a 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 line status SM.

PN2LNSTAT - formats the line status SM for a single line.

PNLNSTAT - handles line 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).

The following programs are called externally as SVM common programs for TIPs, the multiplex subsystem, and so forth.

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.

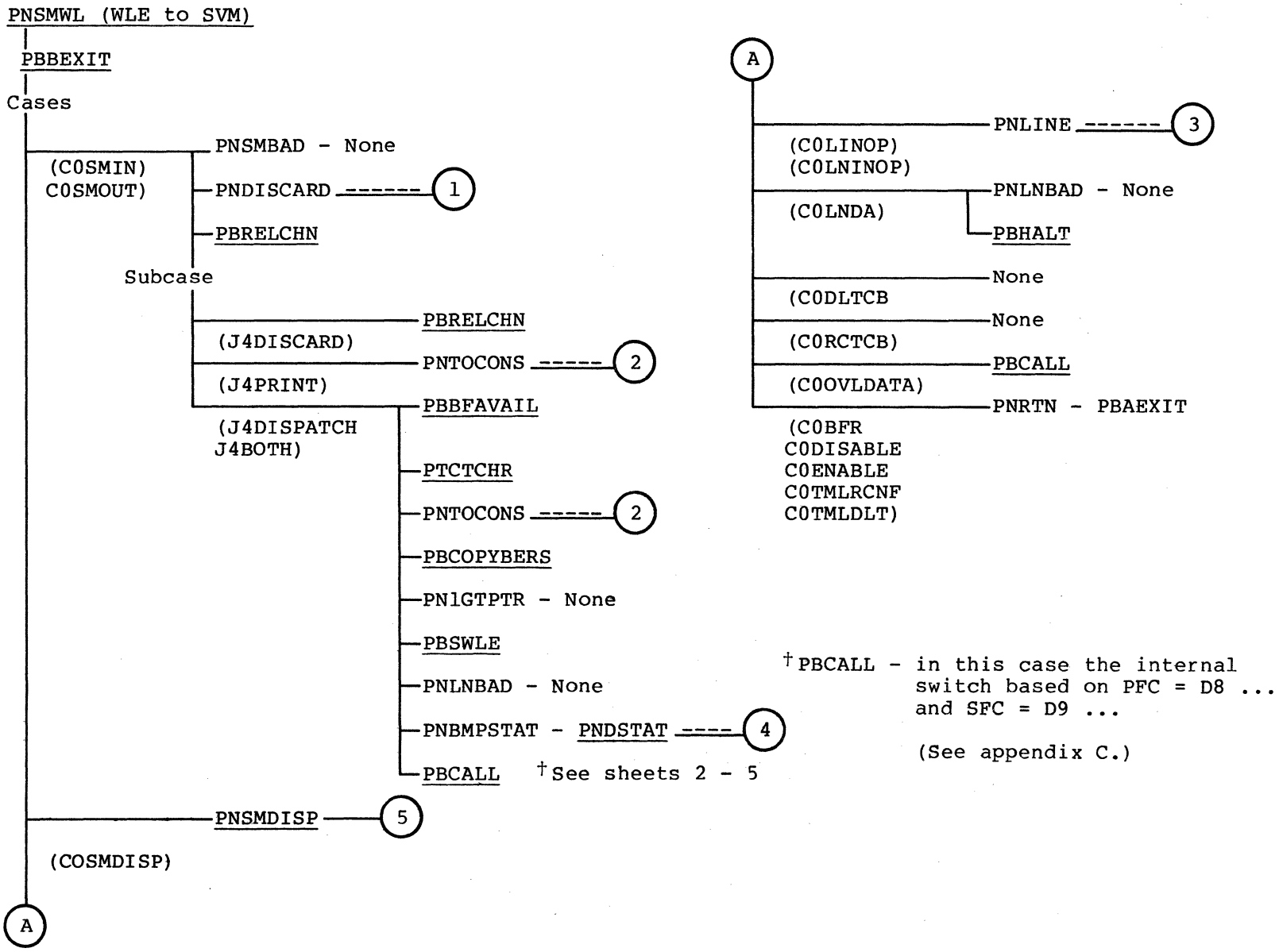


Figure G-5. SVM Trees (Sheet 1 of 8)

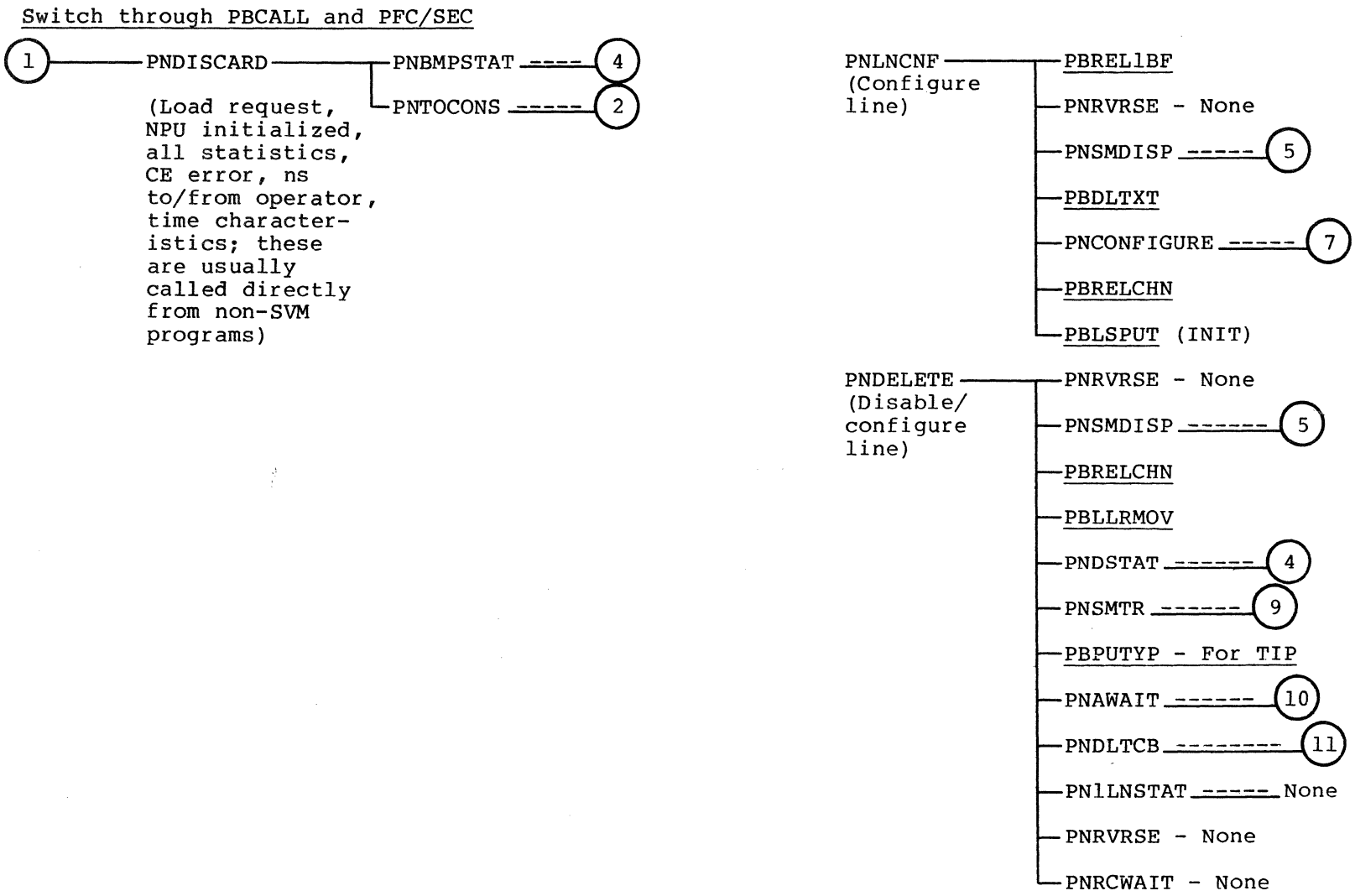


Figure G-5. SVM Trees (Sheet 2 of 8)

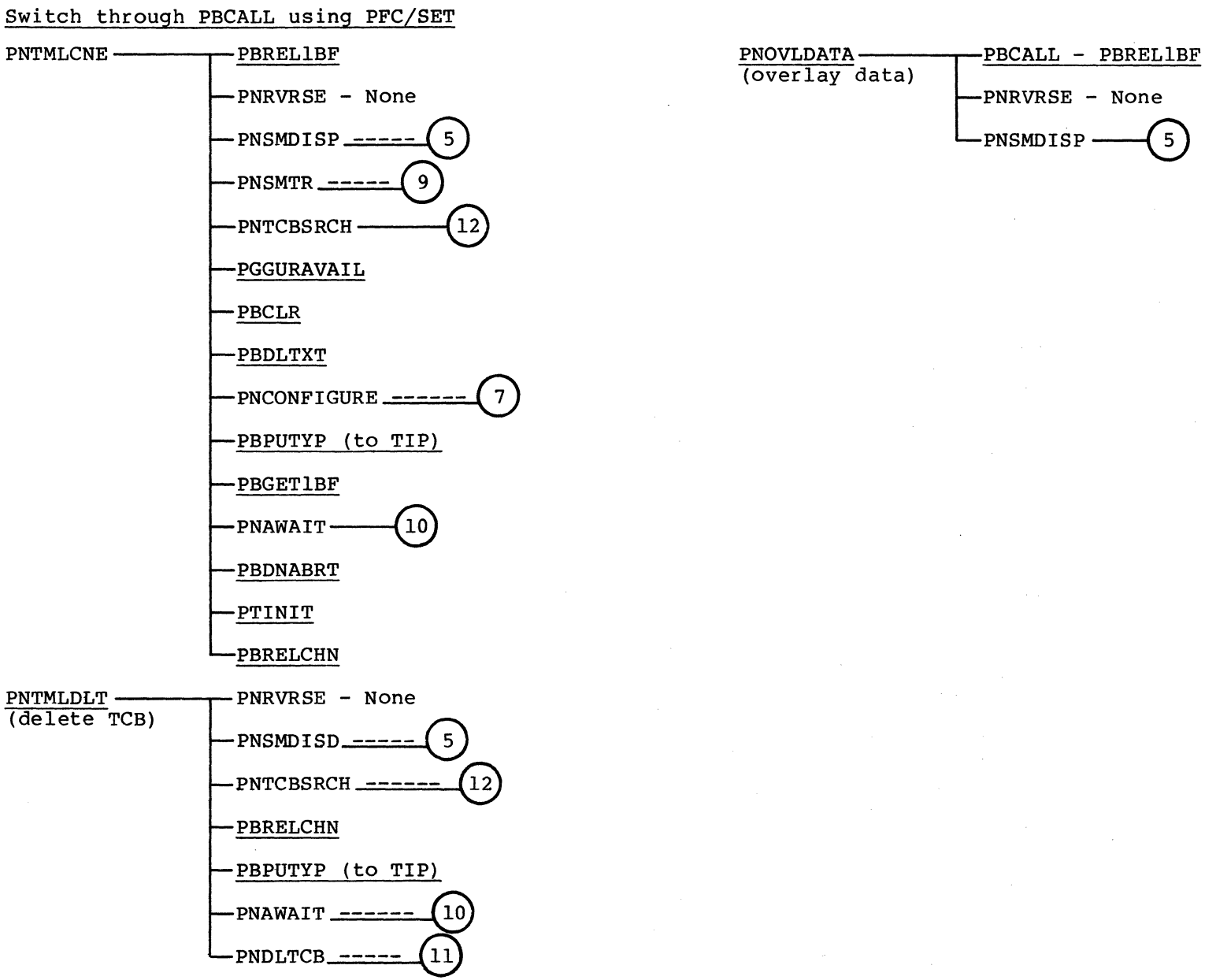


Figure G-5. SVM Trees (Sheet 3 of 8)

Switch through PBCALL using PFC/SFC

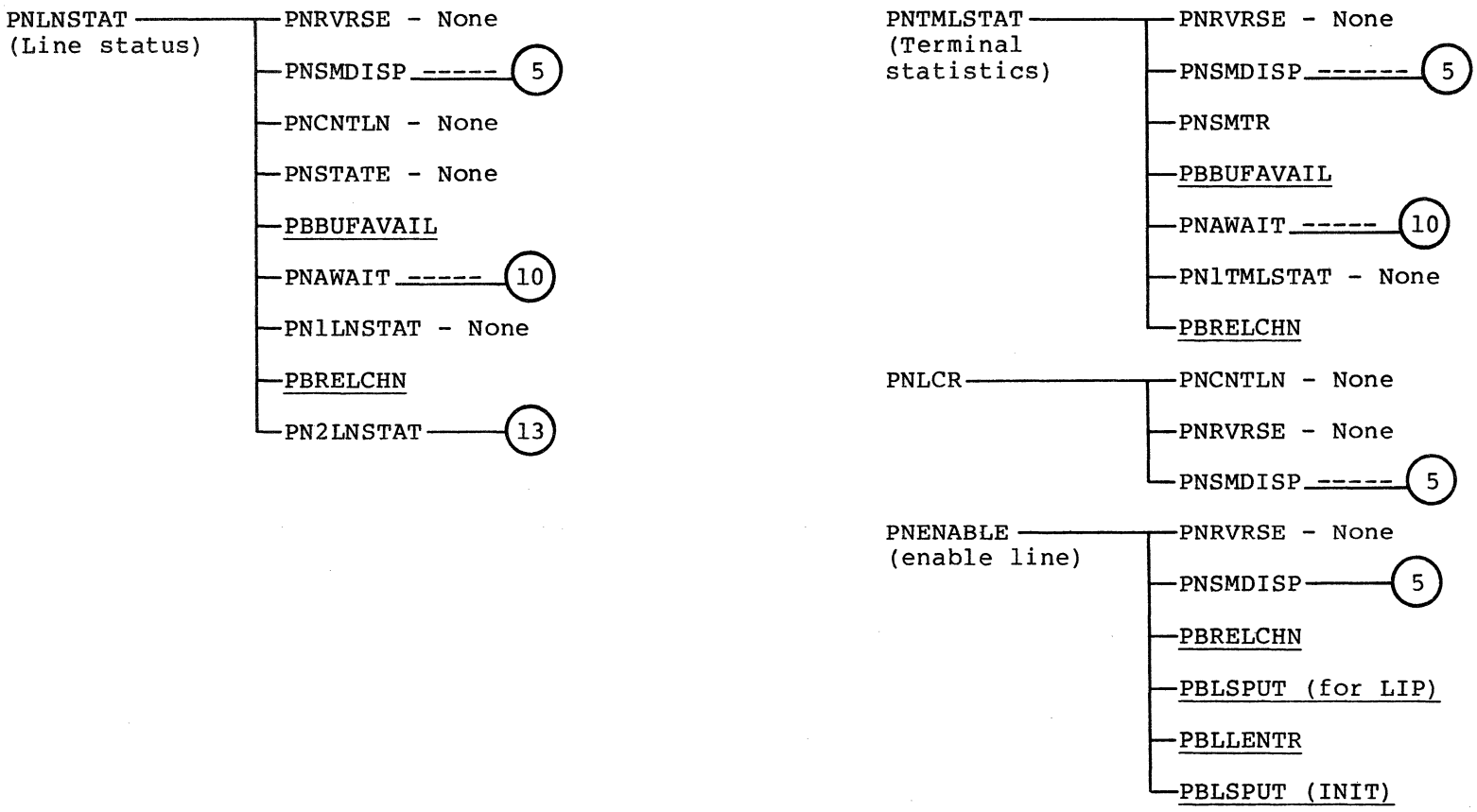


Figure G-5. SVM Trees (Sheet 4 of 8)

Switch through PBCALL using PFC/SFC

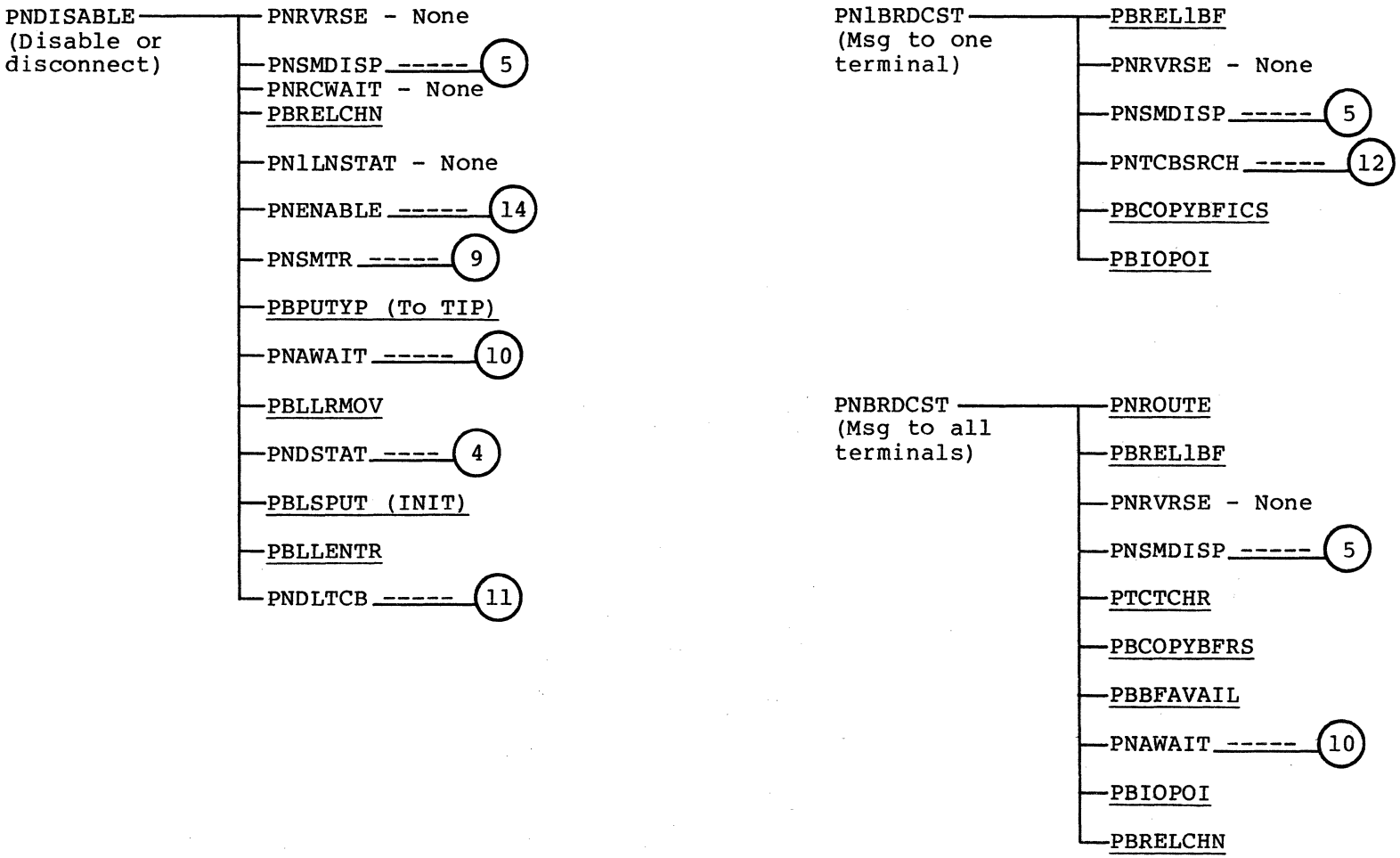


Figure G-5. SVM Trees (Sheet 5 of 8)

SVM routines provide as service routines exclusively for direct external calls

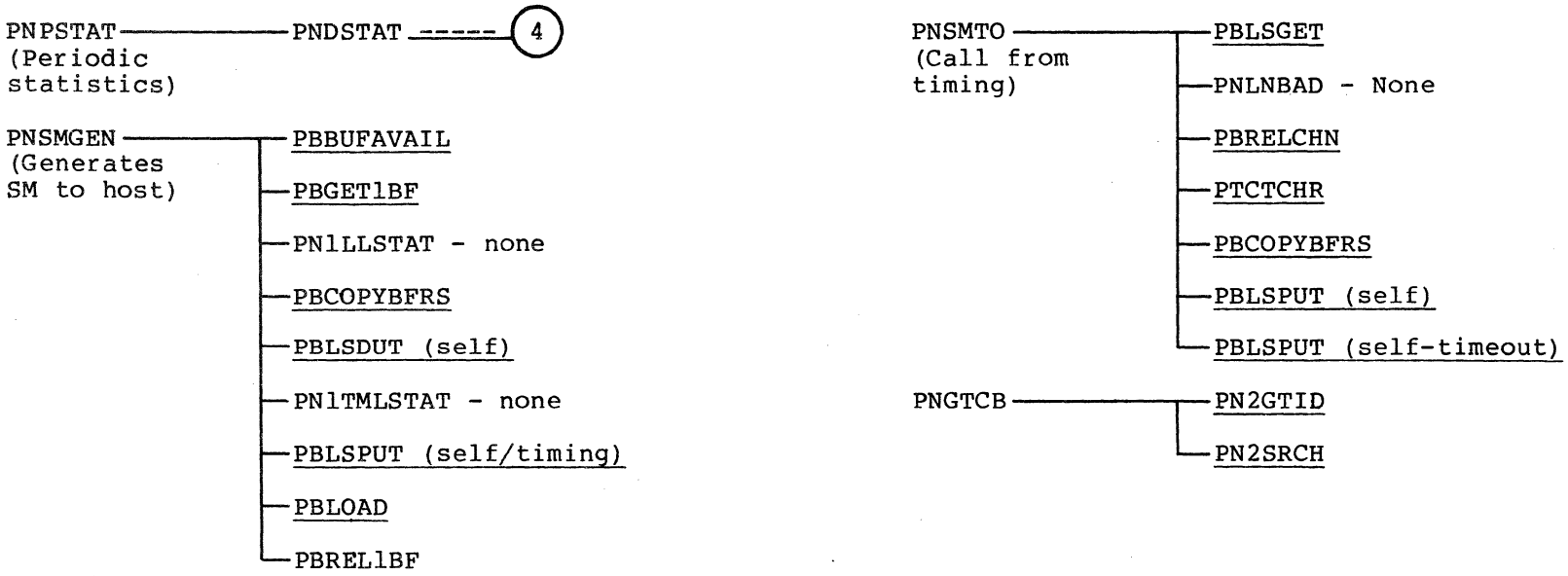


Figure G-5. SVM Trees (Sheet 6 of 8)

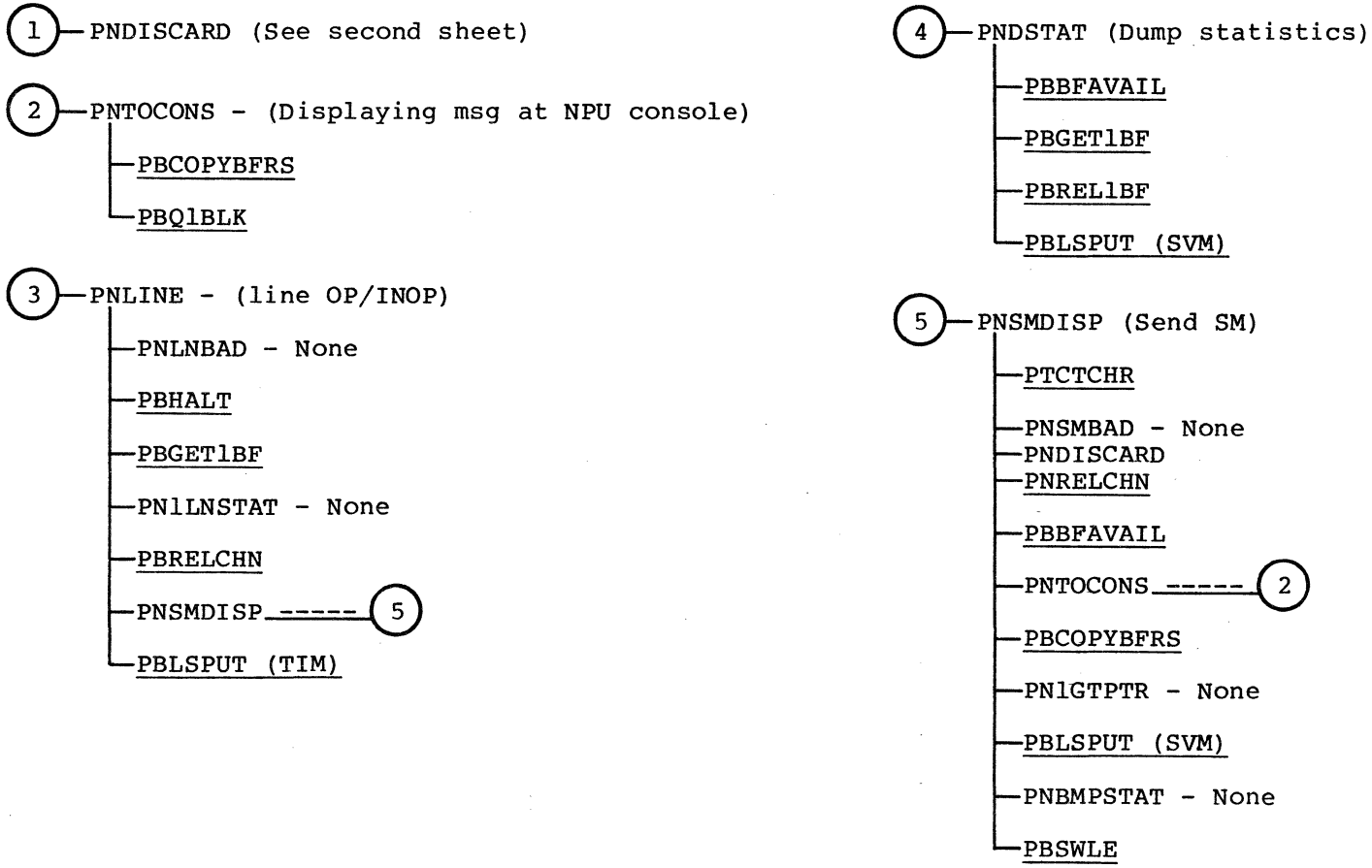


Figure G-5. SVM Trees (Sheet 7 of 8)

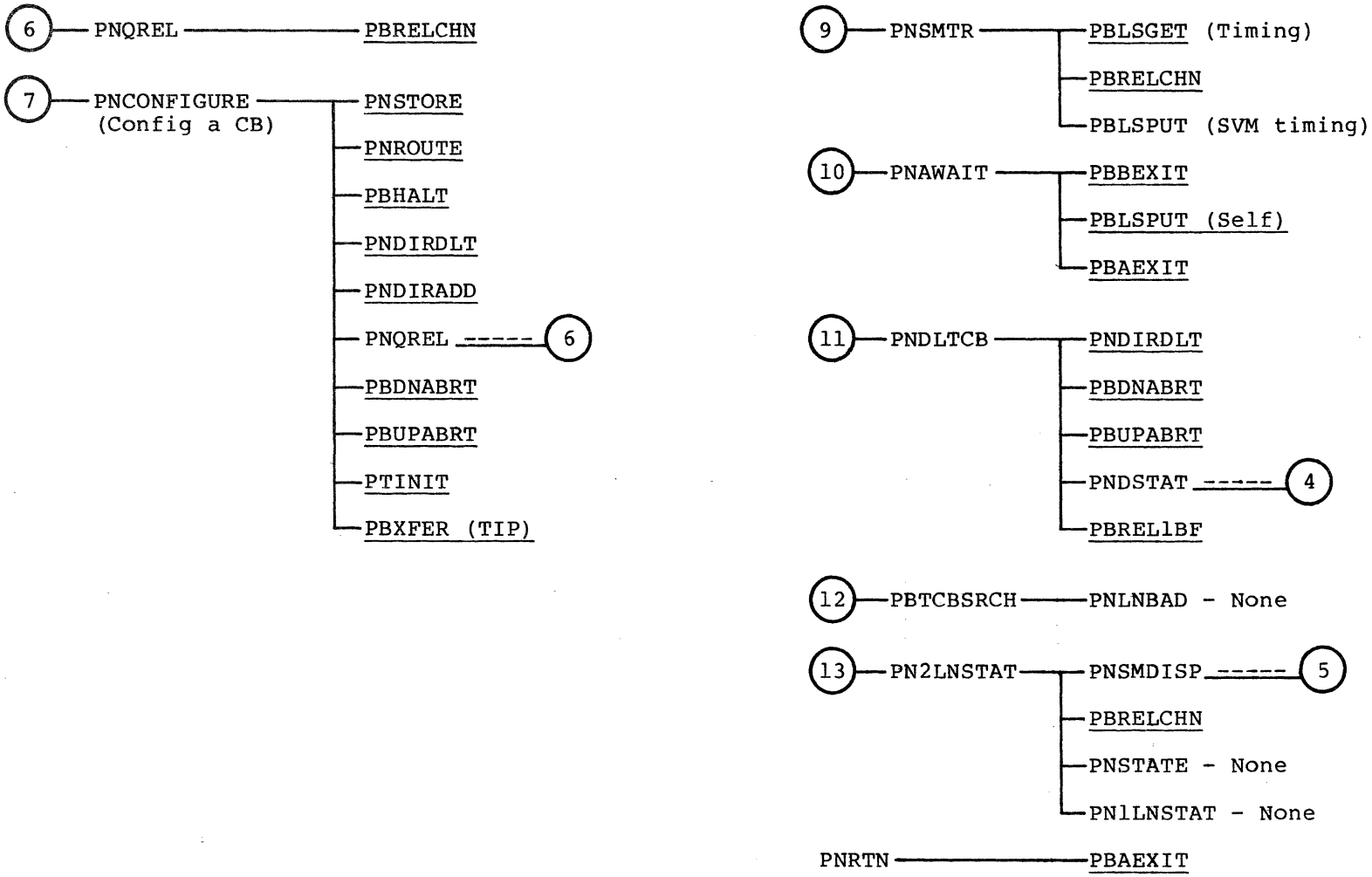


Figure G-5. SVM Trees (Sheet 8 of 8)

PRINCIPAL DATA STRUCTURES H

This appendix lists and describes the principal data structures in CCI. It is intended for use with a link edit or cross-reference listing.

Because PASCAL definitions can occur in three stages (types of structure, variables using these types, and values of 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 vary somewhat. 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 the structure for service messages) the data structures are already described elsewhere. In these cases, the reader is referred to another location in this manual or in the CCI reference manual.

CONTENTS

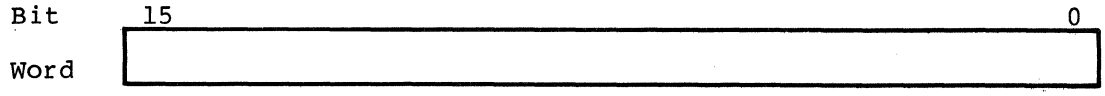
Bits, Words, and Pointers H-4
Bit Definition H-4
Word Structures H-4
Characters (2/word) H-4
Integers (1/word) H-5
Four Hexadecimal Numbers/Words H-5
Flag Word (16 flags/word) H-5
Line Timing H-5
Masks H-6
Character Masks H-6
Bit Masks H-6
Pointer Definitions (B0INTPTR) H-6
Variable Word Definitions H-6
Multiword ASCII Set H-11
Hardware Related Tables H-11
Register Designation H-11
Register Save Area H-11
Coupler Related Constants H-11
Q and A Register Load Area, NGAQLT H-12
Hardware Lines and Associated Software Priorities H-13
NPU Console H-14
Logical/Physical I/O Request Packet, JCPACKET H-14
Device Controller Table, JACONTROLLERTABLE H-16
I/O Response Codes, J0IORESP H-18
Director (Controller) Function Codes for the 1713 TTY H-18
Special TTY (Console Keyboard) Characters H-18
Halt Codes H-19
Block Protocol H-19
Block Protocol Constants H-19
Block Type H-19
Block Byte Sequence H-20
Field Bit Start Position in Byte H-20
Block Type (BT) Byte H-20
Data Bytes H-21
Data Block Clarifier, DBDBC H-21
Character H-22
Downline DBC H-22
Upline DBC H-22
Directories/Internal Processor/Common TIP Routines H-23
Type 1 and Type 4 Tables H-23
Type 1/Type 4 Table Entries, BRDIRCTRY H-23
Type 4 Table List Search Control Block, LSRCHCB H-23
POI Interface Values H-23
Common TIP Routine Structures H-24
TIP Type Table, TIPTYPE H-25
Base System Software H-26
Buffers H-26
Buffer Maintenance Control Block, BECTRL H-26
System Buffer, B0BUFFER H-27
Buffer Constants H-34
Buffer Stamping Area, BYSTAMP H-35
Copy Buffer Parameters, JTCOPYB H-35
Buffer Threshold Levels, B0BUFLEVELS H-35
Worklists H-36
Intermediate Array Format, BWWORKLIST H-36

Multiplex Event Worklist Queue Types, MMEVENT H-37
Service Module Type Worklist Entry Formats, CMSMWLE H-39
Worklist Control Block, BYLISTCB H-40
OPS-Level Worklist, B0WKLSTS H-40
OPS-Level Work Codes, CMWKCODE H-41
Multiplex Event Work Codes H-43
Monitor Tables H-44
PGMSKIP H-44
BYPGMS H-44
SMONT H-44
CBSYTMT H-44
Miscellaneous H-44
System Interfaces H-44
System Interface Table, SITTBL H-44
Firmware Entry Points H-46
Low-Core Pointers H-46
Timing Tables H-47
RTC/Autodata Transfer Table, CICLKADT H-47
One-Second Clock, CASECNTR H-47
Line Timing Control Table, BLTIMTBL H-47
Periodically Executed Programs, CBTIMTBL H-48
Time of Day Tables, CADATE H-49
Loop Forever Instruction H-49
Regulation H-49
Input Regulation Option for PTREGL, REGLTYPES H-50
Control Blocks H-50
Static Logical Link Control Block (LLCB), B0SLLCB H-50
Line Control Block (LCB), BZLCB H-51
Terminal Control Block (TCB), BSTCBLK H-55
Multiplex Subsystem H-61
Multiplex Command Driver Packet, NKINCOM H-62
Multiplex Line Control Block (MLCB), NCLCB, Text Processing Control Block (TPCB) H-64
Port Table (NAPORT) H-69
Line Tables H-70
Multiplex Line Type Table, NBLTYT H-70
Line Types, N0LTYP H-71
Asynchronous Line Speeds H-72
Line Number Field, B0LINO H-72
Multiplex Character Transmit Characteristics Table, NICTCT H-72
CLA/Modem Tables H-73
Modem/CLA Relationships H-73
CLA Types H-74
CLA Commands and Status H-74
Control Command Sequence Word, NDSEQE H-74
Multiplex CLA Command Status Table Entries, NFCCSE H-74
CLA Status Condition Indicators, M0SCTYP H-78
Modem Control States H-78
Modem State Programs H-78
Terminal Tables H-78
Terminal Characteristics Table, NJTECT H-78
Terminal Classes H-80
Terminal and Device Types (TT/DT) H-80
Service Messages H-82
FN/FV Data Structures H-82
Field Description Table, DDFDTRECORD H-82
Action Table Entries, DFATENTRY H-82
Halt Codes H-83

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.

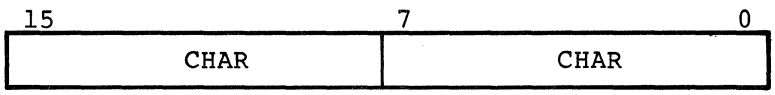
WORD STRUCTURES

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

```
B0CHRARAY = PACKED ARRAY (B015BITS) OF CHAR;
```

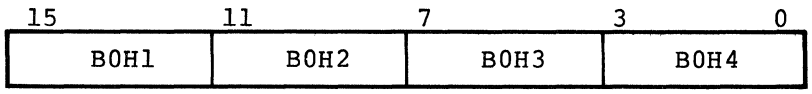
Integers (1/Word)

Word array of 65K words

```
B0INTARAY = ARRAY (B015BITS) OF INTEGER:
```

Four Hexadecimal Numbers/Word

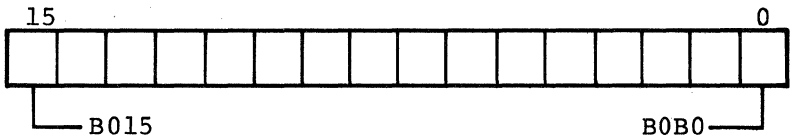
```
B0HEX = PACKED RECORD

        B0H1, B0H2, B0H3, B0H4:  B04BITS

        END:
```



Flag Word

Sixteen flags are packed in one word.

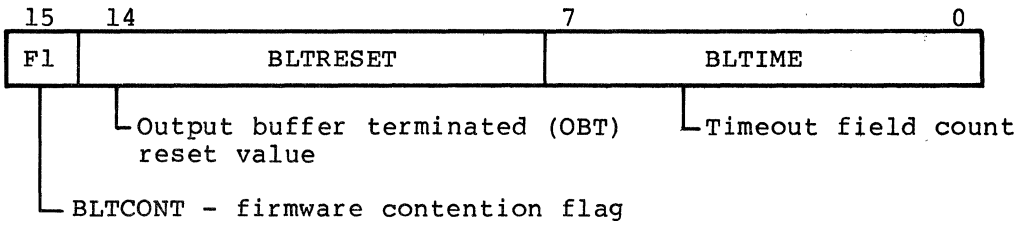
```
B0FKAGS = PACKED RECORD

          B0B15, B0B14, B0B13, B0B12, B0B11,
          B0B10, B0B9  B0B8, B0B7, B0B6,
          B0B5, B0B4, B0B3, B0B2, B0B1, B0B0:  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.

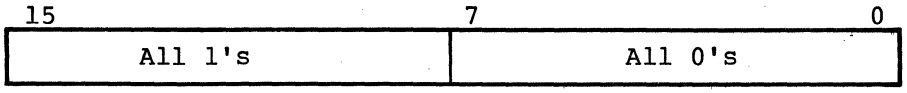


MASKS

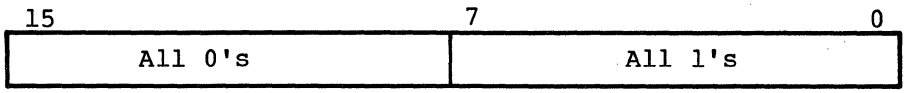
The principal masks are for single characters and single bits.

Character Masks

Left byte, BY0MSK



Right byte, BY1MSK



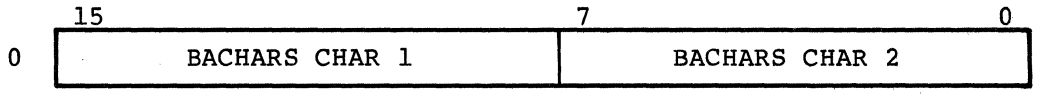
POINTER DEFINITIONS (B0INTPTR)

Pointers are all one word (INTEGER) type.

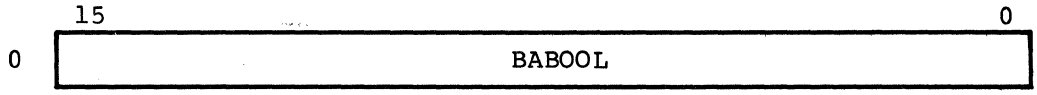
Pointer	Control Block or Buffer	Meaning
B0INTPR	INTEGER	Integer pointer
B0QPTR	B0CBENT	Queue control block pointer (QCB)
B0BUFPTR	B0BUFFER	Buffer pointer for general buffer
B0HEXPTR	B0HEX	Hex pointer
B0REGPTR	B0REGSAVE	Register save area pointer
N0LCBP	NCLCB	Multiplex LCB (MLCB) pointer
BZLCBP	BZLCB	LCB pointer
B0DCBP	NZDCB	Diagnostic control blk PTR.

VARIABLE WORD DEFINITIONS

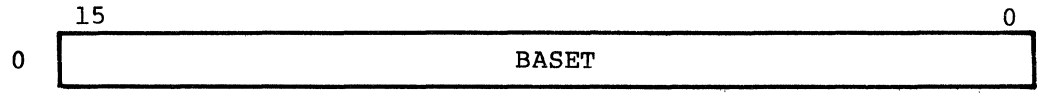
The universal word overlay has many variations. Each variation is of the most frequently used type. Thus, by overlaying the universal overlay over a variable, the variable can be accessed in a variety of formats.



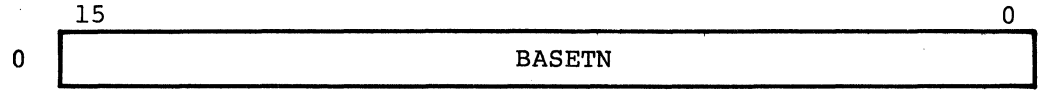
type: CHAR: length 1 to ALFALENG



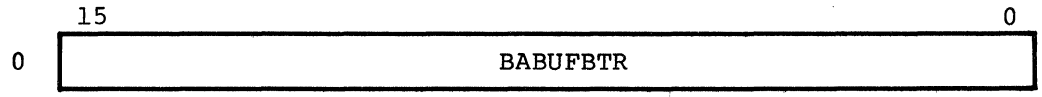
type: B0FLAGS - up to 16 flags



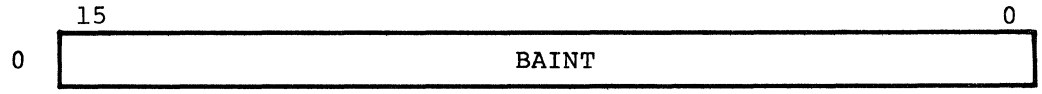
type: SETWORD - mask



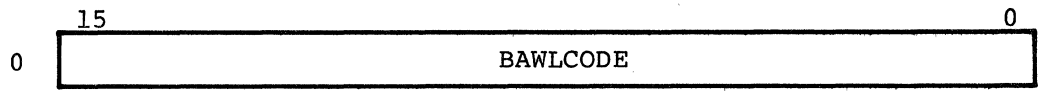
type: SET of 0 through F_{16}



type: B0BUFPTR, buffer pointer

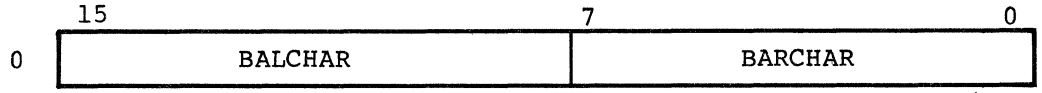


type: INTEGER, full word integer



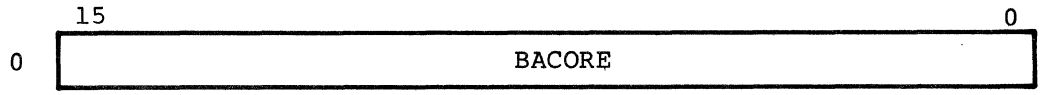
type: B0WLCODES, worklist code

VARIANT:



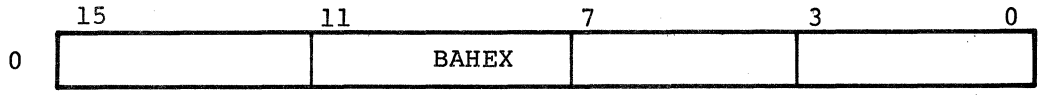
type: CHAR, left and right characters

VARIANT:



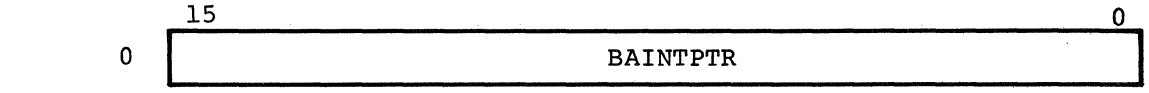
type: B0HEXPTR, hexadecimal pointer

VARIANT:



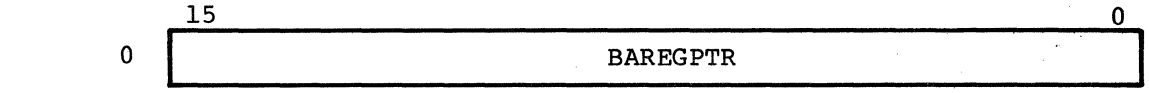
type: B0HEX, 4 hexadecimal digits

VARIANT:



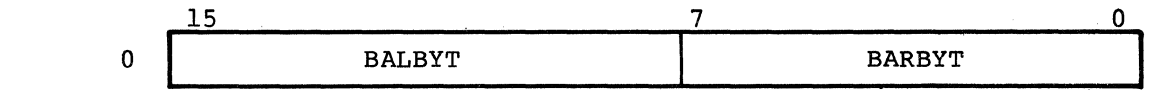
type: B00INTPTR, integer pointer

VARIANT:



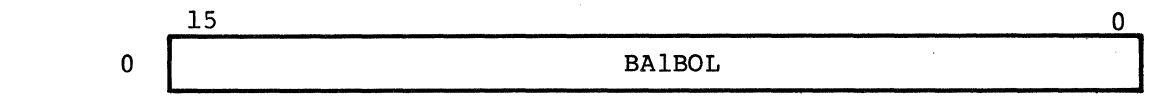
type: B0REGPTR, register pointer

VARIANT:



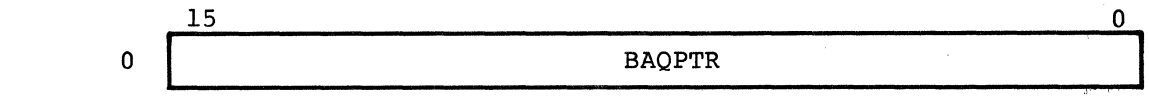
type: B08BITS, integers in left and right bytes

VARIANT:



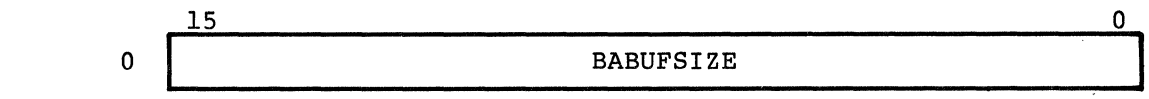
type: BOOLEAN, uses only bit 0

VARIANT:



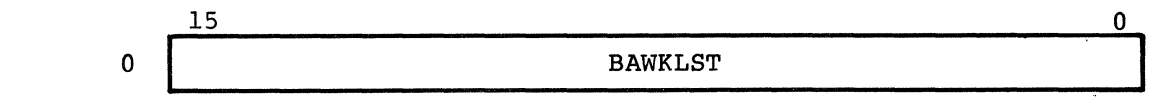
type: B0QPTR, queue pointer

VARIANT:



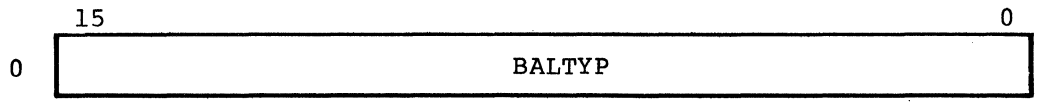
type: B0BUFSIZES. Index to size of buffer (1, 2, 3, 4) for the network. Nominal sizes: 8, 16, 32, 64 correspond to values 1, 2, 3, and 4.

VARIANT:



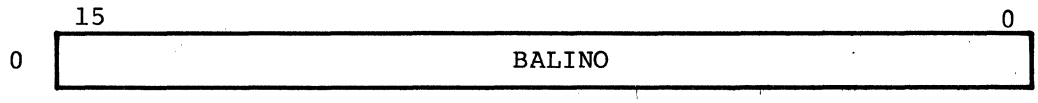
type: B0WKLSTS. Worklist index. Entries in the monitor table as shown in section 5. Uses only bits 0 through 4.

VARIANT:



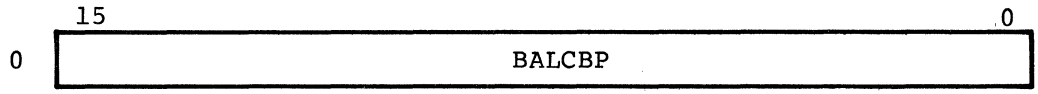
type: N0LTYP. Line type. See table C-3. Uses only bits 0 through 3.

VARIANT:



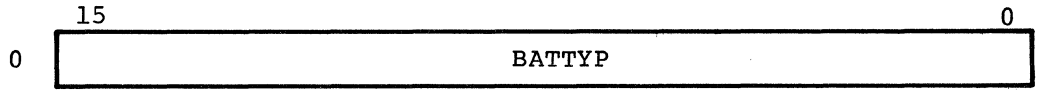
type: N0LINO. Line number. Used to index LCBs.

VARIANT:



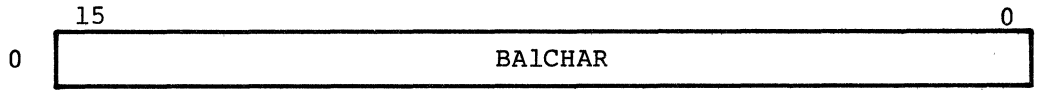
type: BZLCBP. LCB pointer.

VARIANT:



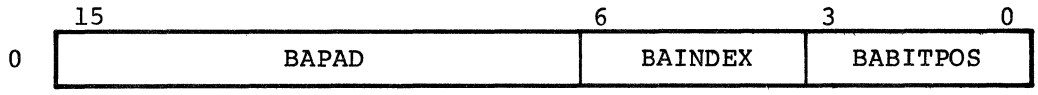
type: N0TTYP. Terminal type. See appendix C.

VARIANT:



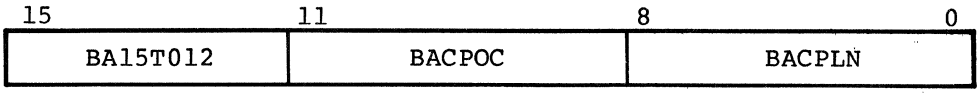
type: CHAR. Right character. Uses full word with character right-justified.

VARIANT: BACHROVLY



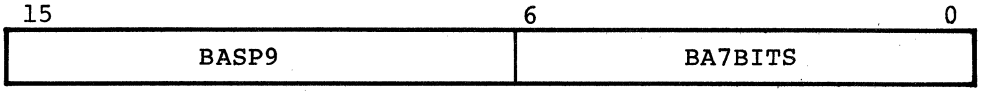
type: Three fields together make a pointer to an ASCII character in the ASCII/binary conversion table. See appendix A. Used in firmware code conversion tables, as shown in assembly listings.

VARIANT:



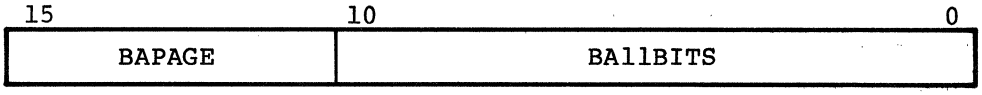
type: Two fields (leftmost field is spare). BACPOC is the coupler orderword code and BACPLN is block length, used by the HIP for threshold checks and for computing the number of buffers needed for an input block.

VARIANT:



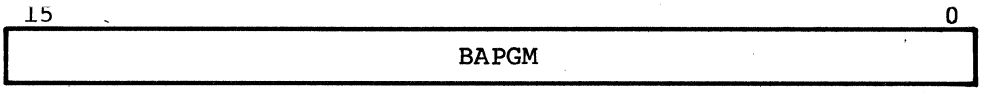
type: fields: variants 24 and 25 are used together as an 18-bit address. BA7BITS is upper 7 bits, BA11BITS is lower 11 bits of address.

VARIANT:



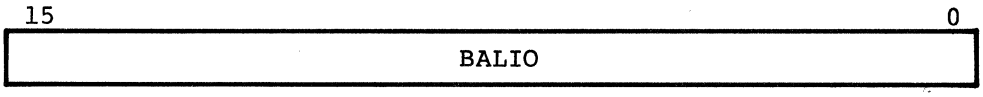
BAPAGE is the page number: range 0 through 31.

VARIANT:



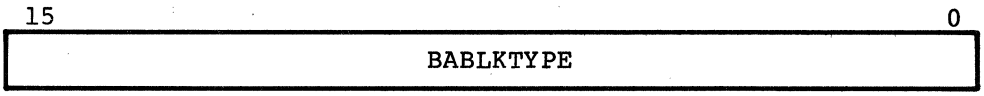
type: B0PGM. Used by TUP to index into the OPS monitor table. Uses only bits 0 through 4.

VARIANT:



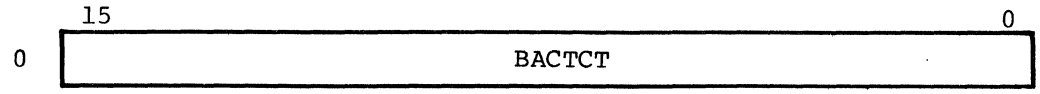
type: J0LIO. Console logical I/O index. Uses only bits 0 through 3.

VARIANT:



type: BLKTYPE. Block type (BT) field in the block header. Uses lower 4 bits.

VARIANT:



type: NICTCT. Entry in character transmission table (NICTCY).

MULTIWORD ASCII SET

JSASCIISET = ARRAY (303BITS) OF SET OF B04BITS:

This is an 8-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 CCI reference manual.

HARDWARE RELATED TABLES

This subsection describes hardware registers and lines which are not handled by the multiplex subsystem.

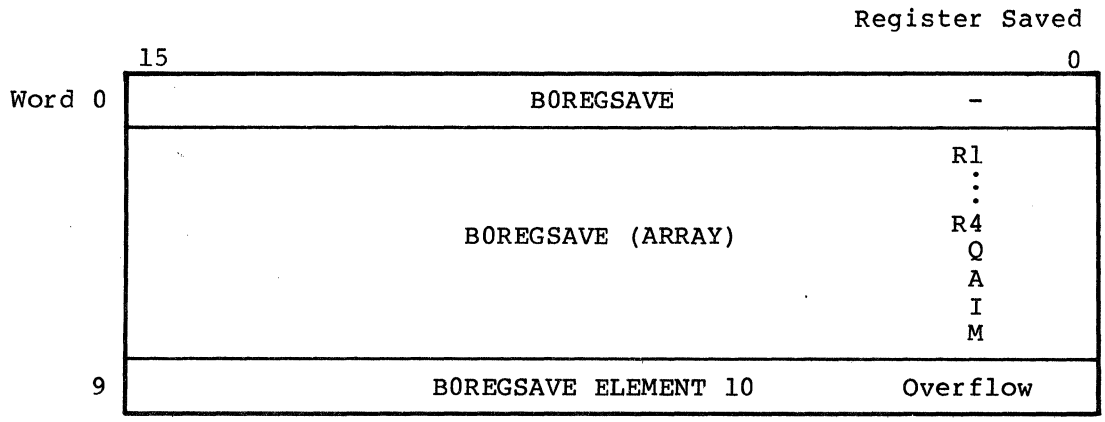
REGISTER DESIGNATION

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



COUPLER RELATED CONSTANTS

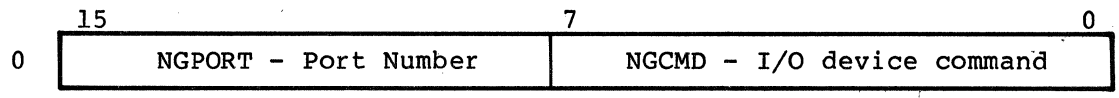
The coupler codes used by the various coupler registers are described in section 7.

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
AINPPU	12	INPUT AVAILABLE - LARGE BLK OR MSG
		INPUT AVAILABLE - PRU BLOCK
Coupler Condition States		
A0PT0	0	IDLE STATE
A0PT1	1	IDLE INQUIRY SENT
A0PT2	2	INITIATED INPUT
A0PT3	3	INITIATE OUTPUT
A0PT4	4	OUTPUT IN PROGRESS
A0PT5	5	READY FOR OUTPUT DELAY
A0PT6	6	NOT RDY FOR OUTPUT DELAY
Coupler Timeout Values		
AIDLET0	3	IDLE TIMEOUT = 1 TO 1 1/2 SECONDS
ADEADTO	60	DEADMAN TIMEOUT = 30 SECONDS

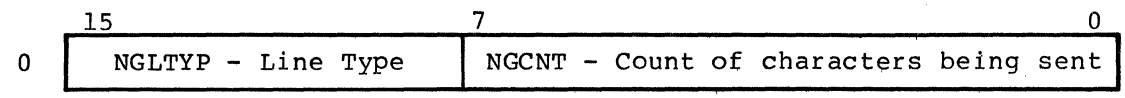
Q and A Register Load Area, NGAQLT

One word is provided for commands (Q register) and two variants are provided for data/subcommands (A register). The NPU console uses the A/Q channel for I/O. These are used only for the command driver.

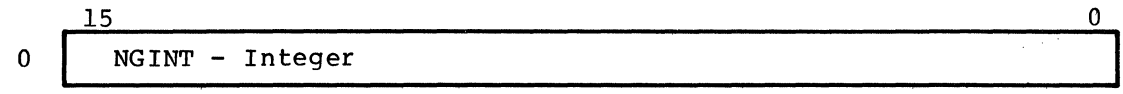
Command (Q)



Subcommand (A)



Universal Overlay



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
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.

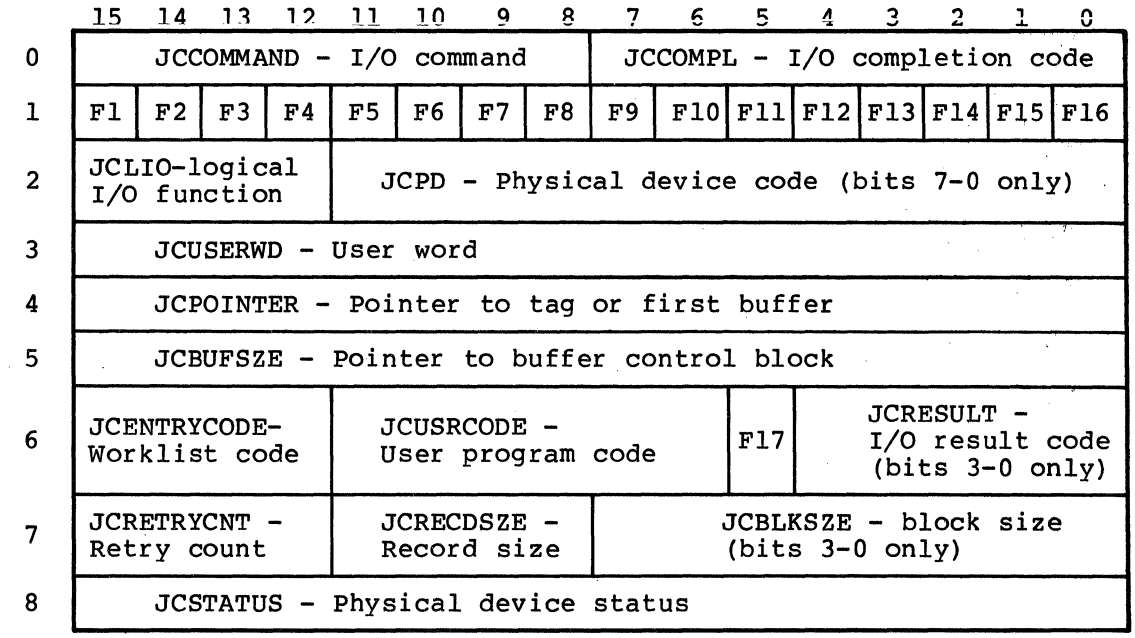
- The request packet from the user (logical request packet, LRP) establishes the message transfer parameters. The LRP is converted to a physical request packet (PRP) by the console driver so that the user does not need to concern himself with terminal physical characteristics.
- The device controller 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.



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 OPS code

The following constant values are assigned to the LRP/PRP fields indicated.

Mnenomic	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 (two console queues)	

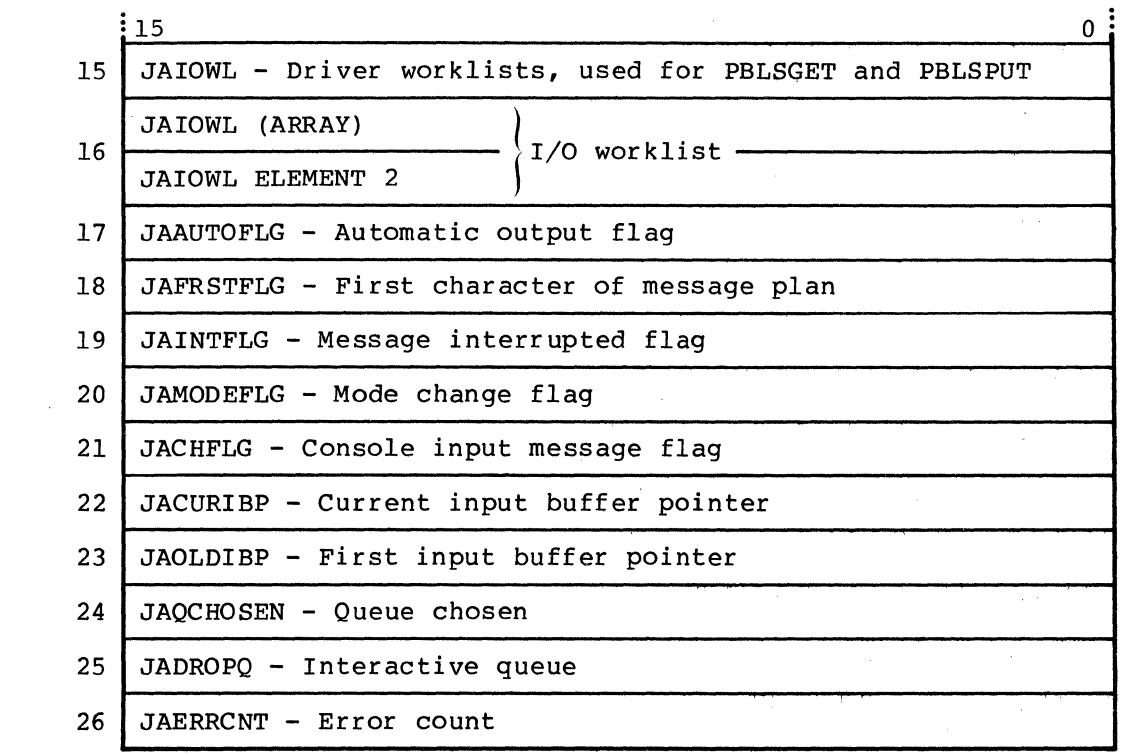
Functions (JCLIO field) are:

	Mnemonic	Value	Console Mode
J0LIO = (	J2SJPIN,	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 (i.e., TTY).

	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	JAAVALUE - A register contents for last I/O count (data)
13	JAREADFLG - Last I/O type flag; 1 = read, 0 = write
14	JAMASK - Mask out device for PBSTARTIO
⋮	⋮



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, clear 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 (CHAR)	Keyboard Character/Use
J1CR,	CARRIAGE RETURN
J1LF,	LINE FEED
J1CTLH,	CONTROL H - TREATED AS BACKSPACE
J1BCKSPCE,	BACKSPACE
J1TUPEOM,	/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	CONTROL D SYSTEM EOM

Halt Codes

The NPU halt message is sent to the NPU console. The halt codes are described in The CCI reference manual.

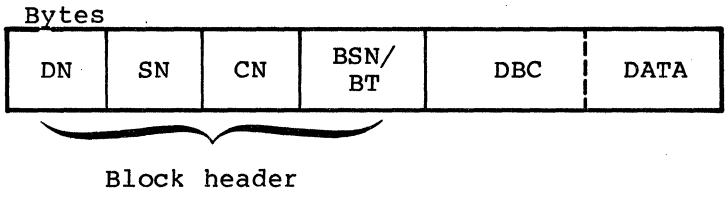
BLOCK PROTOCOL

The block protocol defines the byte structure used to transmit blocks between the host and the terminal mode of the NPU. Blocks are composed of buffers with one or more chained buffers. 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). An additional four bytes are reserved for data block headers, starting with DBC.



BT - block type; BT uses bits 3-0 only.
DBC - data block clarifier; if present, it is the first data byte.

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 and data/ stream control

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.

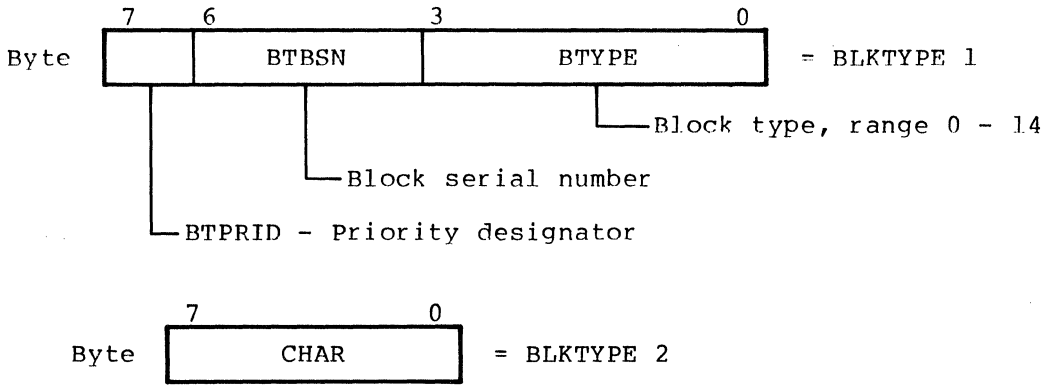
Mnemonic	Byte Position	Byte Use	
DN	6	Destination node	
SN	7	Source node	
CN	8	Connection number	
BTPT	9	Block type/packet type	
P1	10	Parameter 1 (DBC if data instead of parameter)	
P2	11	Parameter 2	Data can start at any parameter position; it must start at the byte following the last parameter used.
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	

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:



Character

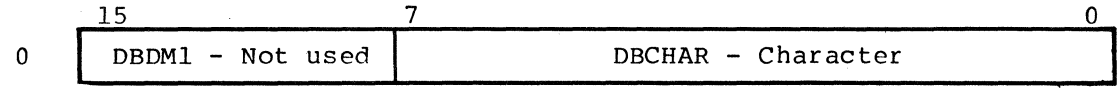
DATA BYTES

Mnemonic	Position of Byte in Data Part of Message	Meaning
DBC	P1 = 1	Data block clarifier (control flags for the data that 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	

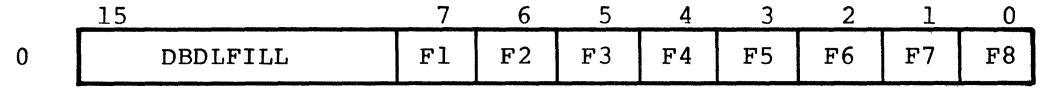
DATA BLOCK CLARIFIER, DBDBC

The DBC is often used as the first byte of the data header in a message. In the definition, it is right-justified in a computer word. Six DBC variants are provided.

Character

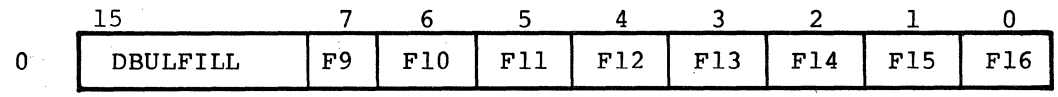


Downline DBC

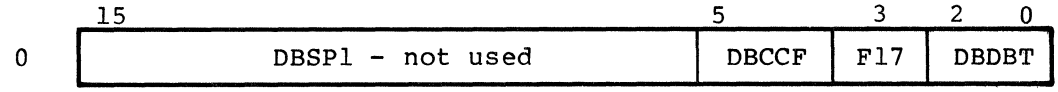


F1 - DBDLS1, spares
F2 - DBDLS2, spares
F3 - DBDLS3, spares
F4 - DBDLS4, spares
F5 - DBDLFE, format effectors used
F6 - DBDLXPT, transparent data
F7 - DBDLS5, spare
F8 - DBDLAUTO, autoinput block

Upline DBC

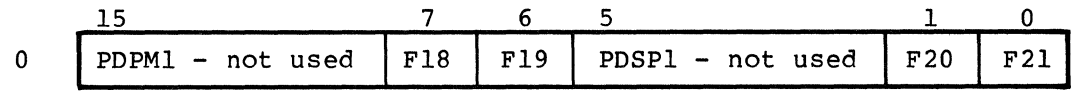


F9 - DBULS1, spare
F10 - DBULS2, spare
F11 - DBULS3, spare
F12 - DBULS4, spare
F13 - DBULS5, spare
F14 - DNULXPT, transparent data
F15 - DBULCAn, cancel data
F16 - DBULPERR, parity error



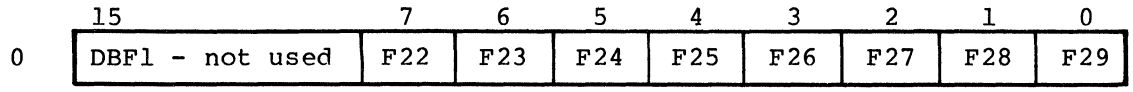
DBCCF - Code conversion
F17 - DBBSF, backspace present
DBDBT - Data clarifier, see section 6, block description

Batch data



F18 - PDPRUB, physical record unit block
F19 - PDBANB, banner block
F20 - PDEOI, message contains an EOI
F21 - PDXPAR, transparent data

Batch data



	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, 0: block contains an EOR
F29 - DBCXPT, transparent data

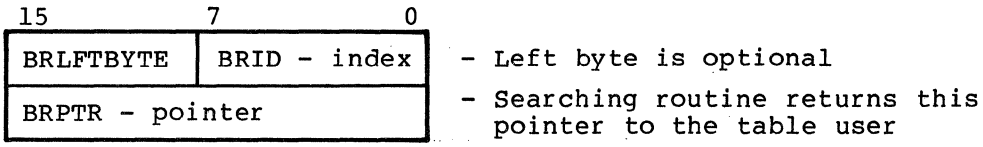
DIRECTORIES/INTERNAL PROCESSOR/COMMON TIP ROUTINES

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

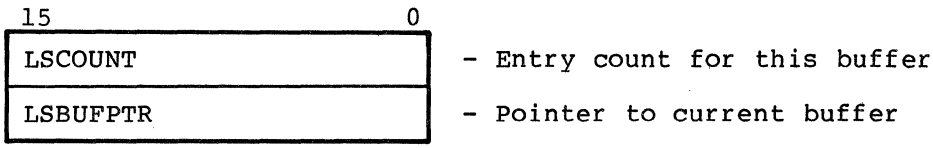
Type 1/Type 4 Table Entries, BRDIRCTRY

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:



15	7	0	
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



15	0	
LSCOUNT		- Entry count for this buffer
LSBUFFPTR		- Pointer to current buffer

POI INTERFACE VALUES

```
B1TCB:  B0BUFFPTR    POINTER TO A TCB
B1BUFF: B0BUFFPTR    DATA BUFFER POINTER
```

COMMON TIP ROUTINE STRUCTURES

TIP CONSTANTS

Mnemonic	Value	Meaning
Mode 4 TIP		
C9M4LCA	20_{16}	Cluster address (CA) — Lower limit
C9M4UCA	$7F_{16}$	Cluster address (CA) — Upper limit
C9M4LTA	60_{16}	Terminal address (CA) — Lower limit
C9M4TUA	$6F_{16}$	Terminal address (CA) — Upper limit
TTY TIP		
G0KEYBRD	4	INPUT STATE - KEYBOARD
G0AUTO	8	INPUT STATE - AUTO REC
G0TAPE	9	INPUT STATE - PAPERTAPE
G0OUTPUT	13	OUTPUT BREAK DETECTION
G0IOUT	3	INITIAL OUTPUT

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 each interface package in a system (TIP, 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 routine entry address						

Flags

F1 - BJOBT, generates output buffer terminated (OBT) flag

F2 - BJBZL, resets timer flag when OBT occurs

F3 - BJSP1, not used

BJQTYPE - TCB buffer size (0 = 8, 1 = 16, 2 = 32, 8 = 64 in nominal system)

BASE SYSTEM SOFTWARE

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 principal buffer structures are as follows:

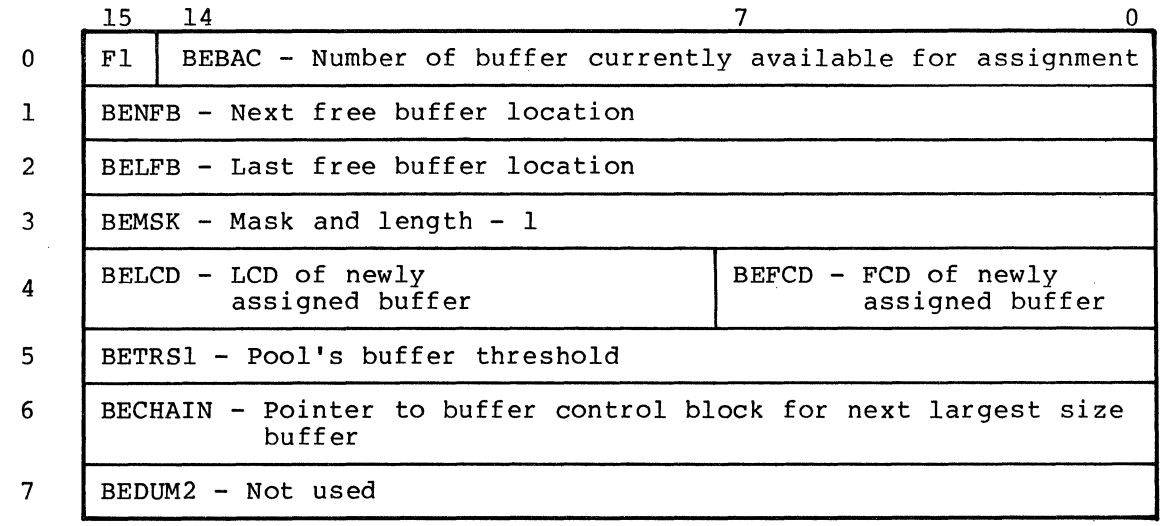
- A control block for each pool of free buffers
- Definitions of each type of buffer assigned
- The optional stamping area which contains two words for tracing buffer use
- A copy buffer input parameter list used by the copy buffers routine, PBCOPYBFRS

There are four buffer sizes. In the normal systems, the buffers are assigned as shown:

B0S0 - 8 words
B0S1 - 16 words
B0S2 - 32 words
B0S3 - 64 words

Buffer Maintenance Control Block, BECTRL

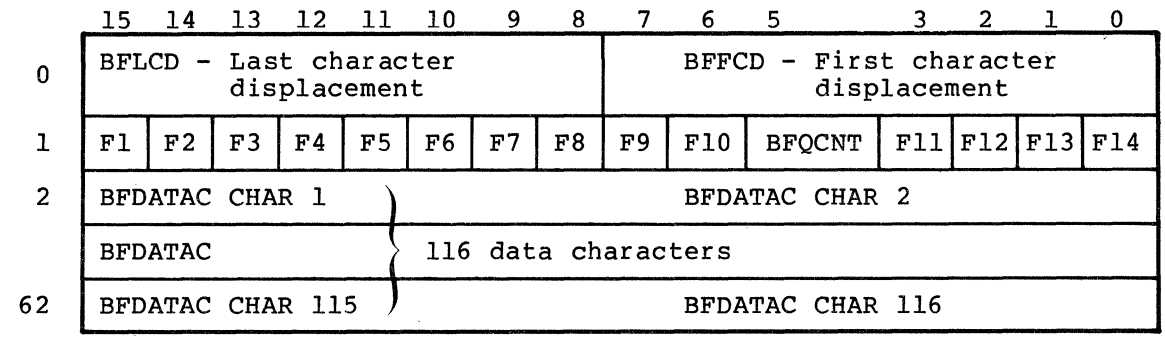
This control block contains all the necessary information for allocating and releasing system buffers. There is a control block for each of the four free buffer pools. Each control block is initialized by PIBUF1. Firmware subroutines allocate and release the buffers.



F1 - Not used
BECTPTR is pointer to BECTRL

System Buffer, B0BUFFER

System buffers exist in four sizes as defined by B0BUFSIZES. Buffers are used for a variety of purposes as described by the following overlay definitions:



BFQCNT - queue count

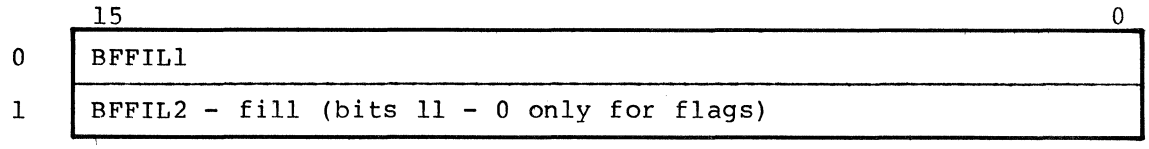
Last word usually reserved for chain to next buffer (see chain variant, below)

Flags:

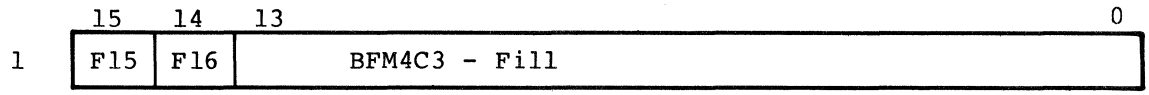
F1 - BFEOTFLG, end of transmission buffer
F2 - BFSOTT, start of transparent text
F3 - BFSONT, start of nontransparent text
F4 - BFSUPCHAIN, suppress buffer chaining
F5 - BFEOBFLG, end of block buffer
F6 - BFINTBLK, internal block; do not send BACK block
F7 - BFPRTK, buffer protect
F8 - BFPERM, permanent 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 user
F13 - BFSP9, not used
F14 - BFDBSIZE, data buffer size, not used (always 64 words in nominal system

Overlays for TIP flags (word one)

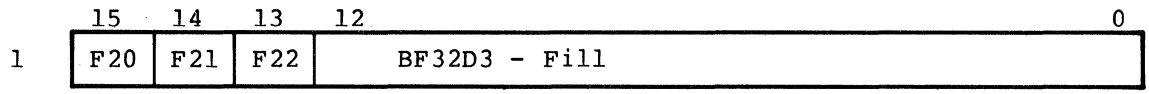


Mode 4 TIP flags



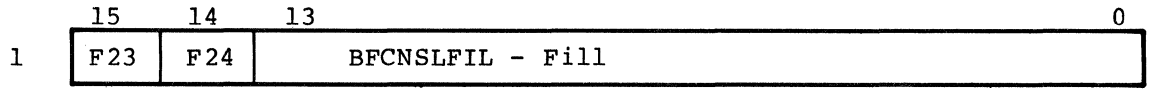
F15 - BFPREPARED, text prepared by text processor
F16 - BFTOGGLE, toggle bit contained in block

BSC TIP



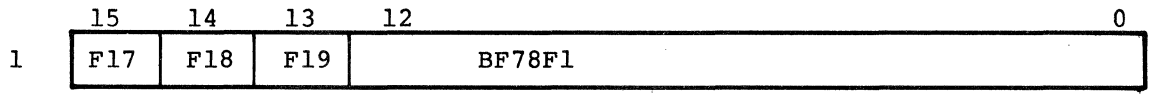
F20 - BFBCCOK, BCC in and OK (3270)
F21 - BFNOTABRTPRT, input state program terminated
F22 - BFVRCBAD, VRC error in packet

Console TIP



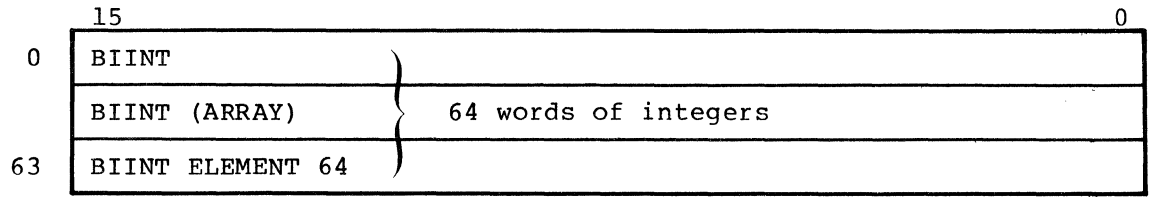
F23 - BFFORMAT, message in console (1=true 0=false)
F24 - BFTEXT, console text to be delivered (1=true 0=false)

Any batch TIP

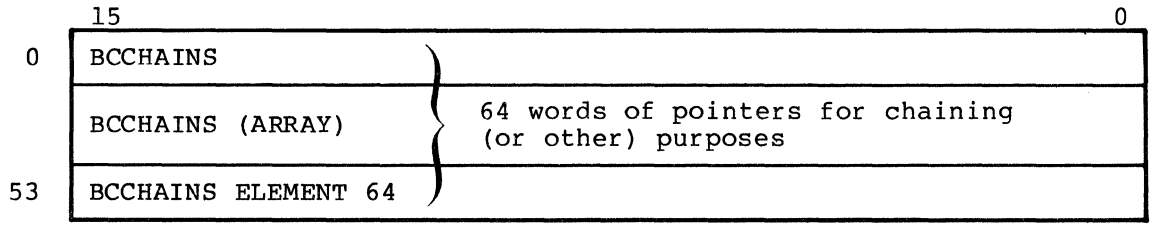


F17 - BFJOB, job card expected next
F18 - BFCXLTA, 026 code translation (0=029, 1=026) by control card type
F19 - BFTR, transparent input

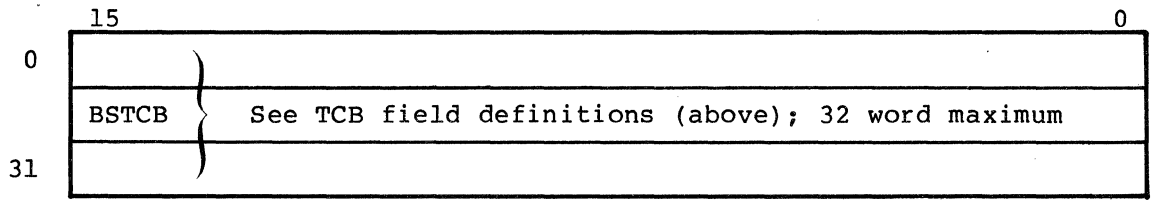
General Purpose Integer Buffer (64 words)



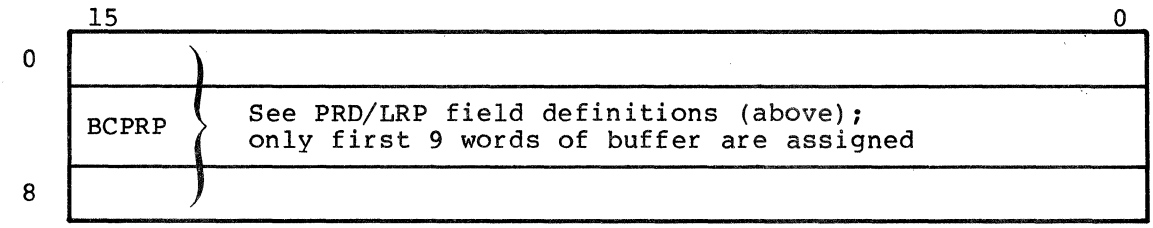
General Purpose Chaining Buffer (64 words)



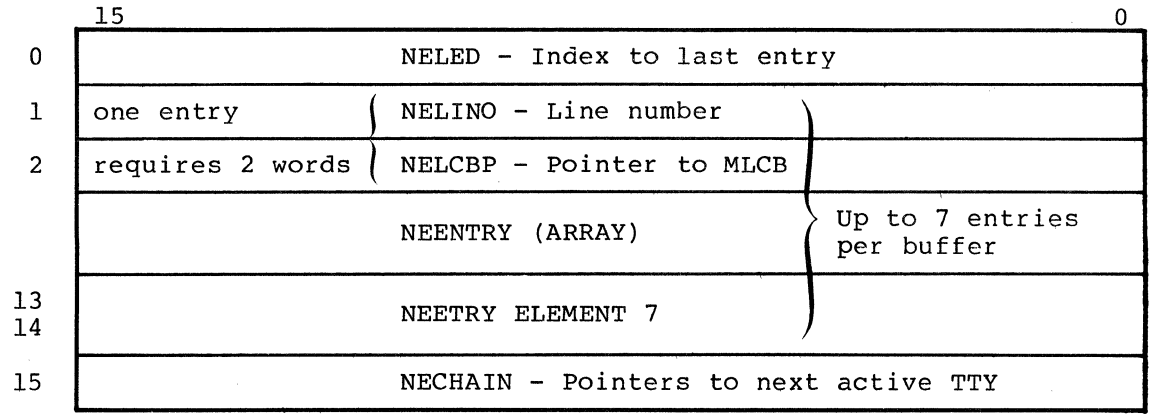
TCB buffer (32 word buffer)



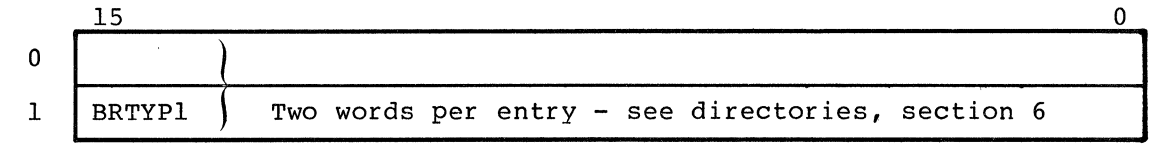
Physical/Logical Request Packet (PRP/LRP) buffer (16 word buffer)



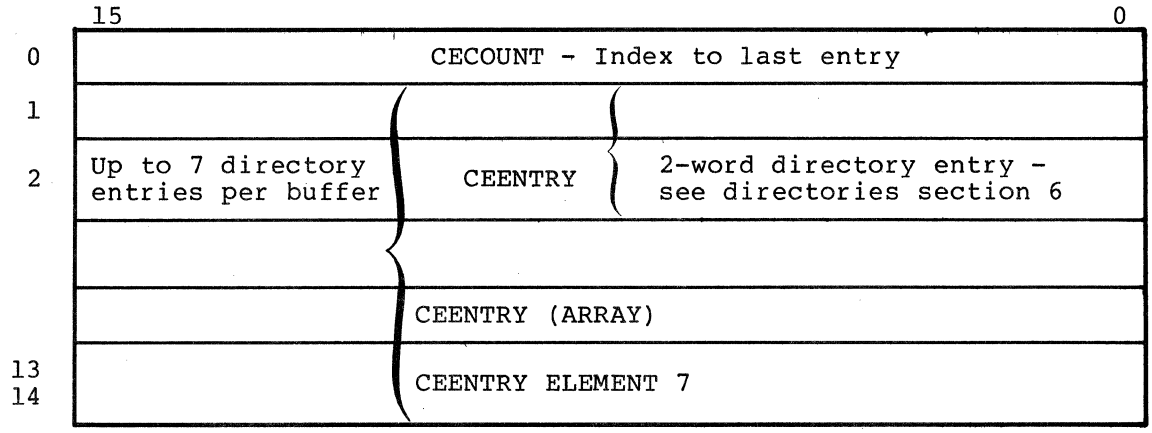
Active TTY LCB List buffer (16 words)



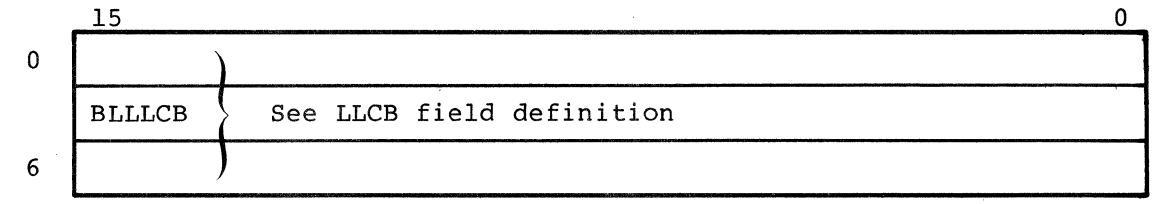
Entries for a Type 1 Table



Buffer for type 4 table (16 word buffer)

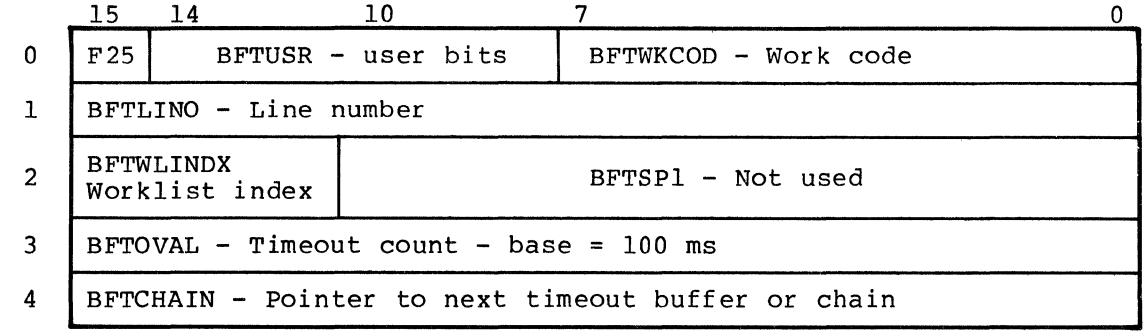


Logical Link Control Block (LLCB) buffer (8 words)

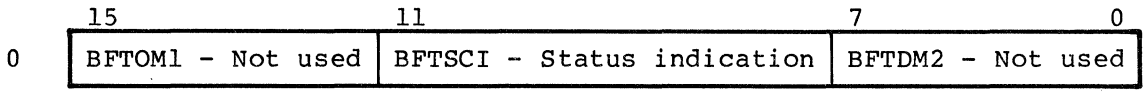


Timeout buffers (8 words)

Two variants are provided.



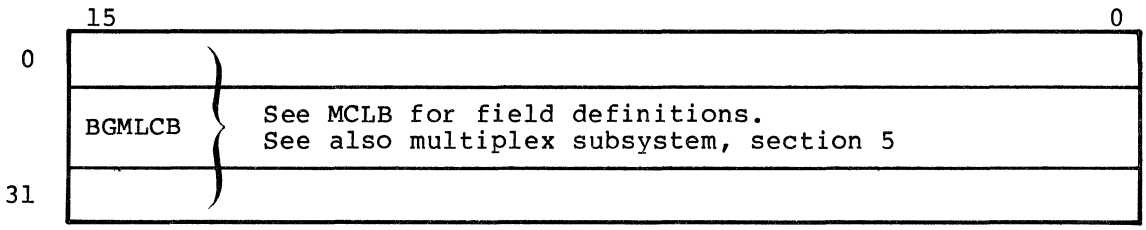
Variant for word 1 of timeout buffer



Flags:

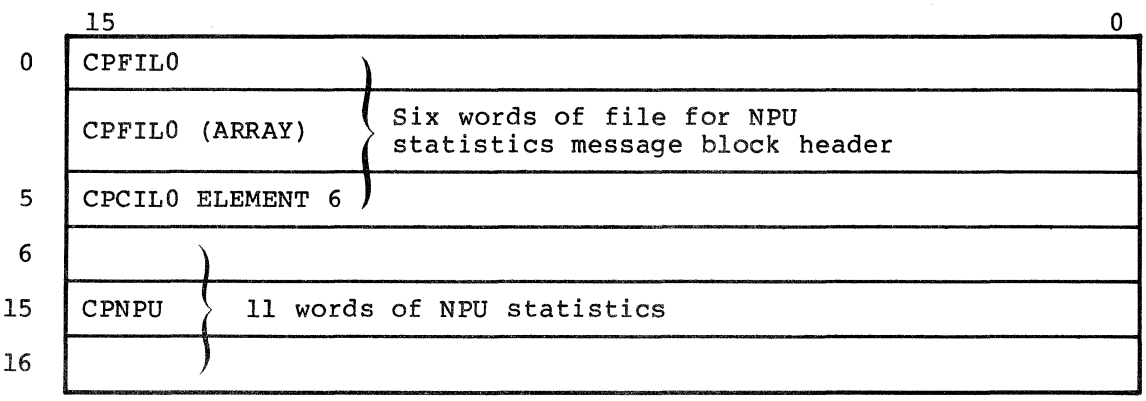
F25 - Buffer release flag

Multiplex LCB (MLCB) buffer (32 words). Also used for TPCB.



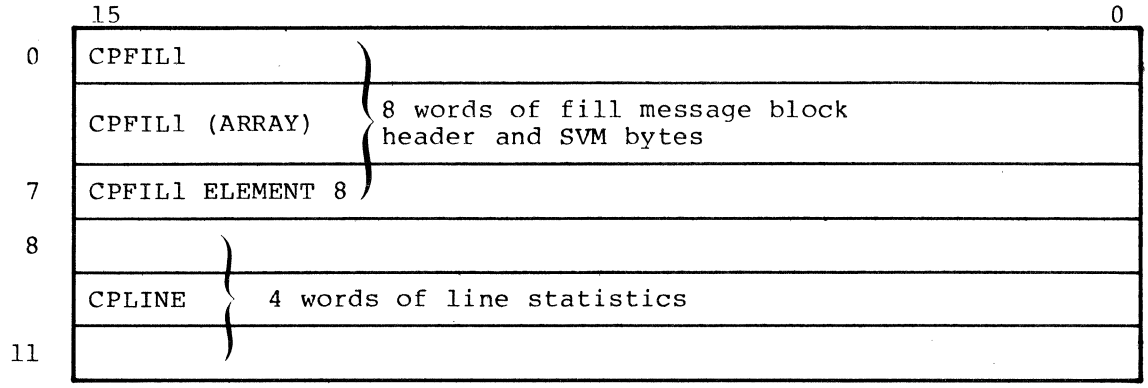
NPU statistics buffer (16 words)

See appendix B of CCI reference manual for field definitions.



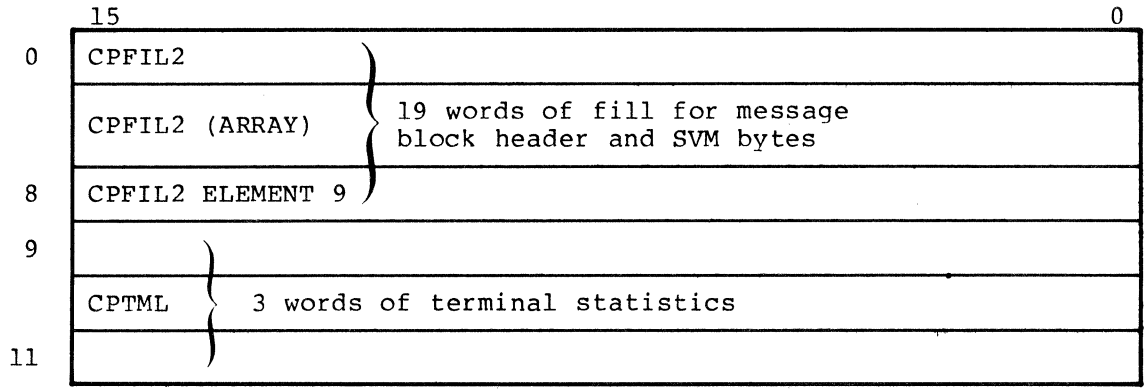
Line statistics buffer (16 words)

See appendix B of the CCI reference manual for field definitions.

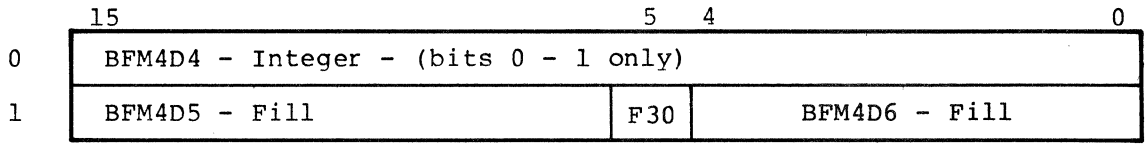


Terminal statistics buffer (16 words)

See appendix B of the CCI reference manual for field definitions.

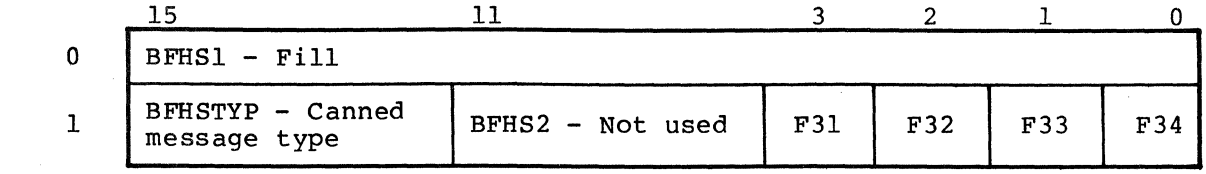


Mode 4 buffer (8 words)



F30 - BFMD4EOJ - extra EOI flag

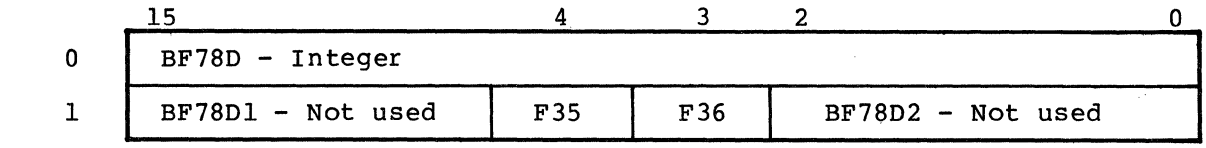
HASP TIP buffer (8 words)



Flags:

F31 - BFHSTXT, text processed data
F32 - BFHSCMODE, transparent data
F33 - BFHSNEW, new record flag
F34 - BFHS3, not used

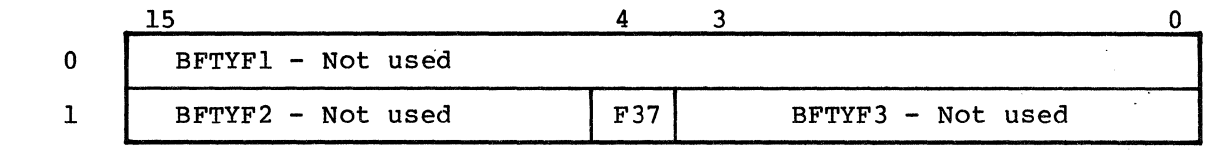
BSC TIP buffer (8 words)



Flags

F35 - BFETXRCVD, ETX received
F36 - BFTPROCESSED, text processed data

TTY TIP buffer (8 words)



Flags

F37 - PFTYPREPARED, Data block clarifier prepared by TTY TIP

Buffer Constants

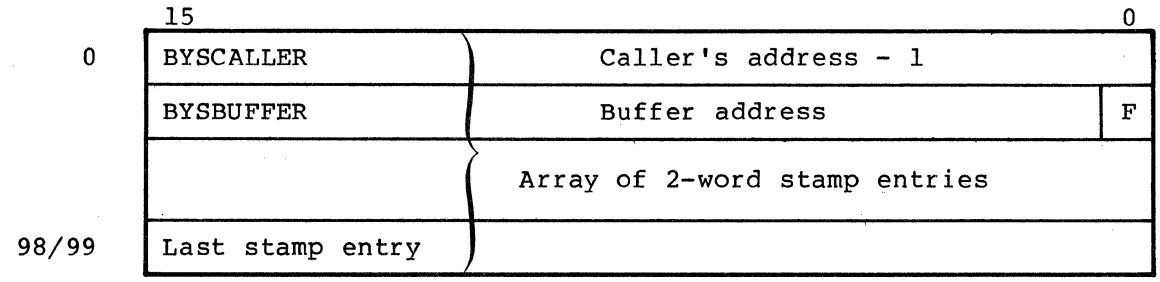
Mnemonic	Value	Meaning
J1FRSTCHAR	4	FCD FOR BUFFER ALLOCATE WHEN NOT IN A NETWORK
J1DATAFRST	4	FIRST CHAR POSITION OF ARRAY BFDATAC IN A BUFFER
J1LST8	13	LAST CHAR OF 8-WORD BUFFER
J1LST16	29	LAST CHAR OF 16-WORD BUFFER
J1LST32	61	LAST CHAR OF 32-WORD BUFFER
J1LST64	125	LAST CHAR OF 64-WORD BUFFER
J2LST128	253	LAST CHAR OF 128-WORD BUFFER
J1LSTCHAR	J1LST64	Maximum LCD in a data buffer
J1LCDFCD	0404	Hexadecimal displacements to character positions for LCD, FCD
J2LCDFCD	090A	
J3LCDFCD	1F06	
J4LCDFCD	1706	
J5LCDFCD	1906	
J6LCDFCD	1B06	
J1BLMAX	64	Maximum buffer length in system with 8-, 16-, 32-, and 64-word buffers
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 (Buffer assigned as 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 entry indicates whether the buffer is currently free or assigned. The file 1 microregisters contain information about the buffer stamping:

File 1 displacement

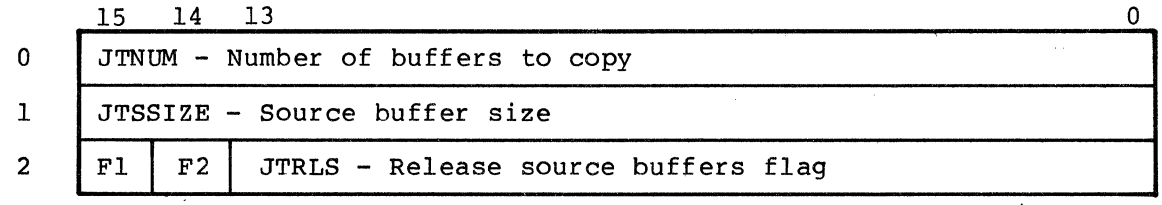
0095 - Stamping status: 0 = not used; ≠ 0 indicates stamping
0069 - Base address of stamping area
006A - Pointer to next entry to be used in the stamping area
006B - Address of last entry in stamping area



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.



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, B0BUFLEVELS

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 hierarchy of regulation checks, 9 is the most important, 0 is the least important.

Mnemonic	Value	Meaning	Type
B0T1	0	CONSOLE SNAPSHOT	
B0T2	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	MULTIPLEX SUBSYSTEM BUFFER THRESHOLD	

WORKLISTS

Worklists can be used on any level, but are principally 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.

BWPKTPTY : B0BUFPTR	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	This overlay is for the timing services worklist (B0BTIWL) and all worklists with single word integer entries.
B0EWLQ : MMEVENT	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	This overlay is for the internal processor worklist and all worklists with 2 consecutive pointer words.
BWIMED : ARRAY (1..J1WLMAX)	This overlay is the general format used by list services for the bulk transfer of entries to and from any worklist.
CMSMLEY : CMSMWLE	This is the service module worklist overlay.
ACPEVENT : B07BITS ACPBLINO : B0LINO ACPB0BUF : B0BUFPTR	Event code Line number Buffer pointer } Coupler overlay
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.

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 subsystem firmware.

VARIANT: Input processing - data

	15 — 7	5 — 0
0	MMWTCOUNT wait count in half seconds	MMSP1 (Not used) | MMWKCOD multiplex work code (given later in this subsection)
1	MMLINO - Line number	
2	MMIBP - Input buffer pointer	
3 3	MMDM2 - Not used	
4	MMDM3 - Not used	

VARIANT: Output processing - data

	15 — 7	6 — 0
0	MMDELAYCNT - Delay count	F1 | MMSP4 - not used
1 2	MMPORT - Physical port number	MMLOPOR

	15 7 6 0
2	MMOBP - Output buffer pointer
3	MMDM5 - Not used
4	MMDM6 - Not used

F1 is a delay completed flag, MMDECMPLT

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 CCI reference manual		

VARIANT: Defines CLA status flags

	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 CCI reference manual

VARIANT: MLIA status

	bits 3 - 0
0	MMDM13 - Not used
1	NNDM14 - Not used
2	MMLIAST - MLIA status

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	
0	CMDATA (optional data) | CMWKCODE	Code range: $21\text{-}3F_{16}$. See OPS-level work-codes for SVM.
1	CMLINO | Line number	
2	CMPTR | Points to SM or TCB	

- SM pointer

	15 7 0	
0	CMDATA | CMWKCODE	
1	CMPOINT	Pointer to SM

- Save and return

	15 7 0	
0	CMDATA | CMWKCODE	
1	CMR1	Save location for R1 and R2 return address
2	CMR2	
3	CMRTN	

- Service message timer

	15 7 0	
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 4C.

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		
3	BYSPARE - Not used		
4	BYWLINDEX - Worklist index	BYSP2 - not used; can use only bits 7-0	
5	BYSP3 - Not used		
6	F2	BYMAXCNT - Number of worklist to get on this call	BYPAGE - Program page address
7	BYPRADDR - Program address		

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 (B0WKLSTS) of BYLISTCB.

OPS-Level Worklist, B0WKLSTS

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 B0DUMMY. The last entry must be B0DUMMY which is equal to the last TIP worklist value and causes the monitor scan pointer to return to value 8.

Mnemonic	Value	Meaning
B0FSWL	1	FIRST WORKLIST = MMEWLQ
MMEWLQ	1	MUX EVENT WORKLIST QUEUE
B0HIPDLQ	2	HIP DATA LIST QUEUE
B0SMTO	3	SERVICE MODULE TIMEOUT LIST
B0T200	4	CRITICAL 200 MS TIMEOUT
B0TTYP	5	TTY CONSOLE DRIVER - PRIORITY
B0TTYN	6	TTY CONSOLE DRIVER - NON PRIORITY
B0LPWL	7	LINE PRINTER DRIVER
B0CHWL	8	CONSOLE PROGRAM
B0INWL	9	INTERNAL PROCESSOR (IP)
B0MLWL	10	MLIA INTERRUPT HANDLER
B0SMWL	11	SERVICE MODULE (SVM)
B0TIWL	12	TIMING SERVICES
B0TYWD	13	TIP DEBUG (PTTIPDBG) - OPTIONAL
B0LIWL	14	LINE INITIALIZER (LINIT)
B0DGWL	15	ONLINE DIAGNOSTICS - OPTIONAL
B0DOWL	16	HOST INTERFACE PACKAGE (HIP)
B0HDLC	17	NOT USE
B0M4WL	18	MODE 4 TIP
B0TTYWL	19	MODE 3 TIP - TTY
B0HASP	20	HASP TIP
B027WL	21	2780/3780 TIP
B0HHWL	22	HASP/360 HIP
B0DUMMY	23	DUMMY FOR CONSOLE

The subtable scanned by the OPS monitor is called B0PGMS. It extends from B0CHWL to B0DUMMY. The value assigned in the array is B0WLCODES.

OPS-Level Work Codes, 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, or TIPs			
A0HARDERR	0F	Hardware 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	From SVM
A0SMDLTCB	15	Delete TCB	From SVM
A0BREAK	16	Downline break	

Mnemonic	Value (hex)	Meaning	Origin
Miscellaneous			
A0DBUX	17	Output buffer XMIT	From TIP to itself
A0SMLN	18	Line status protect	From SVM
A0SMNPUINIT	19	NPU init protect	From SVM
A0SMMPCCINIT	1A	MPCC Init protect	From SVM, not used
A0SMFAIL	1B	Force load MPCC	From SVM, not used
C0LINOP	20	LINE OPERATIONAL	From TIP, or
C0LNINOP	21	LINE INOPERATIVE	line initializer
C0LNDA	22	LINE DISABLED	(LINIT)
C0DLTCB	23	TCB DELETED	From TIP
C0SMIN	24	SM IN	
C0SMOUT	25	SM OUT	
C0SMDISP	26	DISPATCH SM	
C0OVLDATA	27	OVERLAY DATA (not used)	
C0BFR	28	MISCL. BFR EVENT	From SVM to itself
C0ENABLE	29	ENABLE LINE EVENT	
C0DISABLE	2A	DISABLE LINE EVENT	
C0TMLDLT	2B	DELETE TERM. EVENT	

Generated by input state programs to OPS-level TIP (note that multiplex macros must equate this A0WK1 to its own A0WK1 with the same value).

Mnemonic	Value	User	Work Code ID
A0WK1	21	TIP	WORK CODE 1
A0WK2	22	TIP	WORK CODE 2
A0WK3	23	TIP	WORK CODE 3
A0WK4	24	TIP	WORK CODE 4
A0WK5	25	TIP	WORK CODE 5
A0WK6	26	TIP	WORK CODE 6
A0WK7	27	TIP	WORK CODE 7
A0WK8	28	TIP	WORK CODE 8
A0WK9	29	TIP	WORK CODE 9
A0WK10	2A	TIP	WORK CODE 10
A0WK11	2B	TIP	WORK CODE 11
A0WK12	2C	TIP	WORK CODE 12
A0WK13	2D	TIP	WORK CODE 13
A0WK14	2E	TIP	WORK CODE 14
A0WK15	2F	TIP	WORK CODE 15
A0WK16	30	TIP	WORK CODE 16
A0WK17	31	TIP	WORK CODE 17
A0WK18	32	TIP	WORK CODE 18
A0WK19	33	TIP	WORK CODE 19
A0WK20	34	TIP	WORK CODE 20
A0WK21	35	TIP	WORK CODE 21

Mnemonic	Value	User	Work Code ID
A0WK22	36	TIP	WORK CODE 22
A0WK23	37	TIP	WORK CODE 23
A0WK24	38	TIP	WORK CODE 24
A0WK25	39	TIP	WORK CODE 25
A0WK26	3A	TIP	WORK CODE 26
A0WK27	3B	TIP	WORK CODE 27
A0WK28	3C	TIP	WORK CODE 28
A0WK29	3D	TIP	WORK CODE 29
A0WK30	3E	TIP	WORK CODE 30
A0WK31	3F	TIP	WORK CODE 31
A0STOP	A0WK1	Stop transmission code	

Multiplex Event Work Code

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	TTY 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.

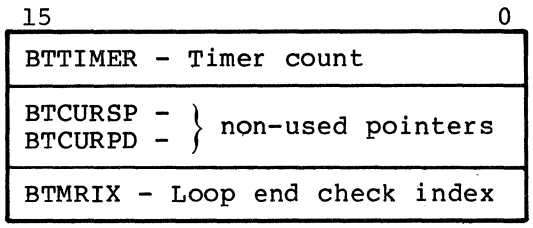
BYPGMS

Three cases:

- BYIPGM - B0PGMS type
- BYWKLS - Worklists type
- BYINT - Integer type

SMONT

Used by timing services for timed programs (half-second time base)



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	Definition	Common Name
SIENTY	POINTER TO BWWLENTRY	OPS monitor
SITMTB	POINTER TO CBTIMTBL	Timing (PBTIMAL)
SILCBS	POINTER TO CGLCBS	LCB
SIWLCB	POINTER TO BYWLCB	Worklist CB

Mnemonic	Definition	Common Name
SIDBSIZE	POINTER TO BEDBSIZE	Data buffer sizes
SINJTEC	POINTER TO NJTECT	Terminal characteristics
SITIMTBL	POINTER TO BLTIMTBL	Line timing
SITIPTYP	POINTER TO BJTIPTYPT	TIP type
SIOVLBLK	POINTER TO SYOVLCB	Overlay control (not used)
SILCBP	POINTER TO HALCBP	Sub TIP
SILLRMOV	ADDRESS OF PBLLRMOV	LLCB remove
SILLENTB	ADDRESS OF PBLLENTB	LLCB enter
SICOIN	ADDRESS OF PBCOIN	Command driver
SIGTlBF	ADDRESS OF PBGETlBF	Get buffer
SIRLlBF	ADDRESS OF PBRELlBF	Release buffer
SIBFAVL	ADDRESS OF PBBFAVAIL	Buffer availability check
SIRTlLCB	ADDRESS OF PTRTlLCB	Return to TIP entry after event
SISVlLCB	ADDRESS OF PTSVlLCB	Save TIP entry until event occurs
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
SIlBADD	ADDRESS OF PBl8ADD	18-bit address final
SIl8COMP	ADDRESS OF PBl8COMP	18-bit address computer
SITOAH	ADDRESS OF PBTOAH	Convert hex in ASCII format

Extent of the entries that 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
SYCTCT	Multiplex character transmit characteristics	NICTCY	N0LNSPDS
SYTMTB	OPS - level periodic programs	CBSYTMT	C0TDPGMS
SYTECT	Terminal characteristics	NJTECY	N0TCLASS
SYTIMTBL	Line timing	BZLTIME	0...C4LCBS
SYTIPTYPT	TIP type	TIPTYPE	N0TIPTY
SYOVLCB	Overlay control block	SYOVLCB	Not used

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 necessary.
PFBURLS	606	Releases a buffer.
PFBUGET	605	Assigns a buffer of the size requested.
PFBUEXT	609	Extracts a buffer.
N1FIRMAD	600	Outputs to CLA sequence.
N2P3INTAD	601	Generates a multiplex - level 2 interrupt.
N3P3INTAD	602	Resets multiplex - level 2 interrupt.
PFLINTO	60A	Decrements line timeout count.
PFSR2SM	60E	Sets/resets status bits. Used to load/dump /start the multiplex side of a 2552 program execution timing (requires external hardware measuring device).

Low-Core Pointer

The low-core pointer (also called the address table) is a sequence of addresses extending from location 0150_{16} to location $016A_{16}$. It is shown in appendix B of the CCI reference manual.

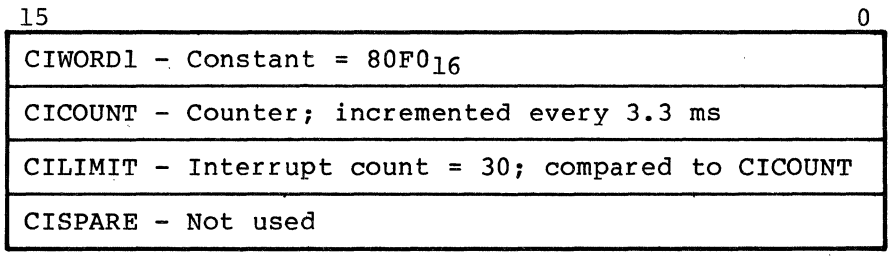
TIMING TABLES

The principal timing tables are:

- RTC (real-time clock) table used to count 3.3 ms increments that are used to generate the 100 ms RTC interrupt
- One-second clock counter
- Line timing table for timing out I/O events
- Array of programs that are run periodically
- Time of day tables

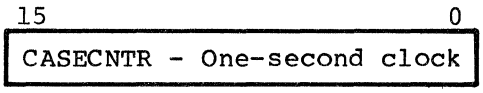
RTC/Autodata Transfer Table, CICLKADT

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.



One-Second Clock, CASECNTR

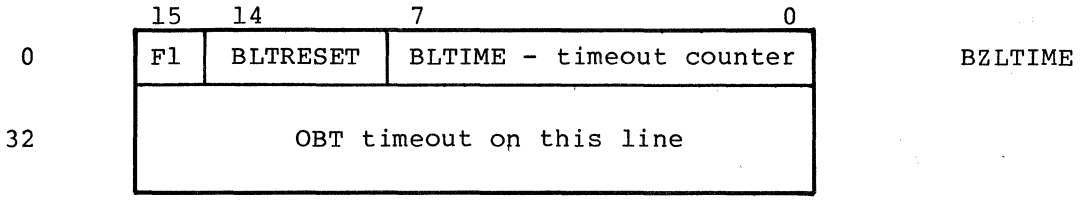
This clock is used by PBTIMEOFDAY for time of day calculations. The count is used modulo 60 by the minute counter, modulo 60 x 60 by the hour counter, modulo 60 x 60 x 24 by the day counter, and modulo 60 x 60 x 24 x month (days) by the month counter.



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).

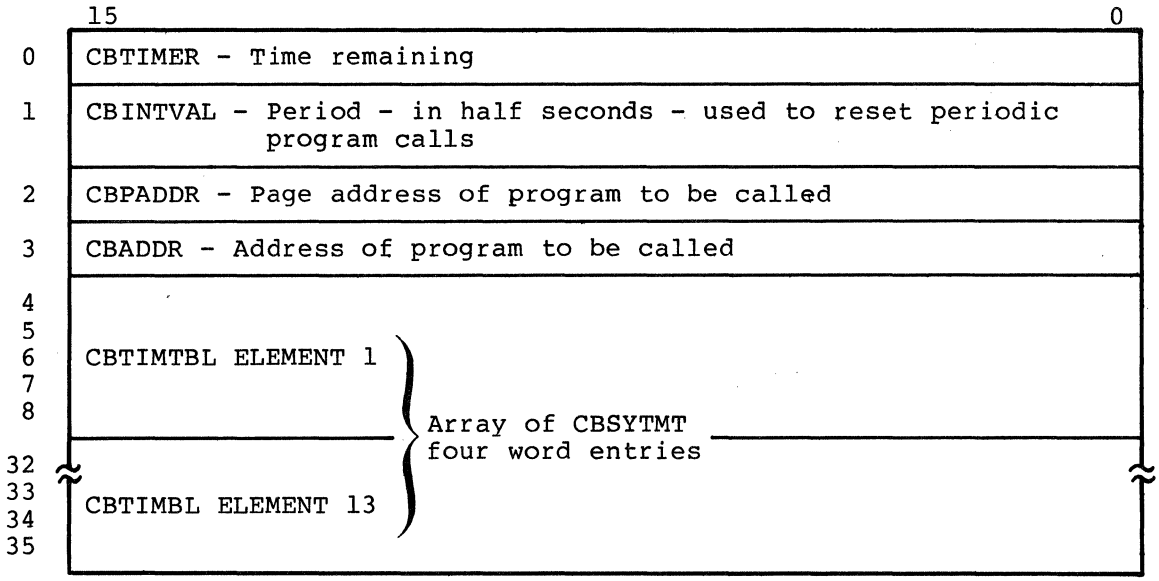


Fl - Not used
BLTRESET - OBT timeout value for the line
BLTIME - Set by line user; decremented each half-second by PBTIMER

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.

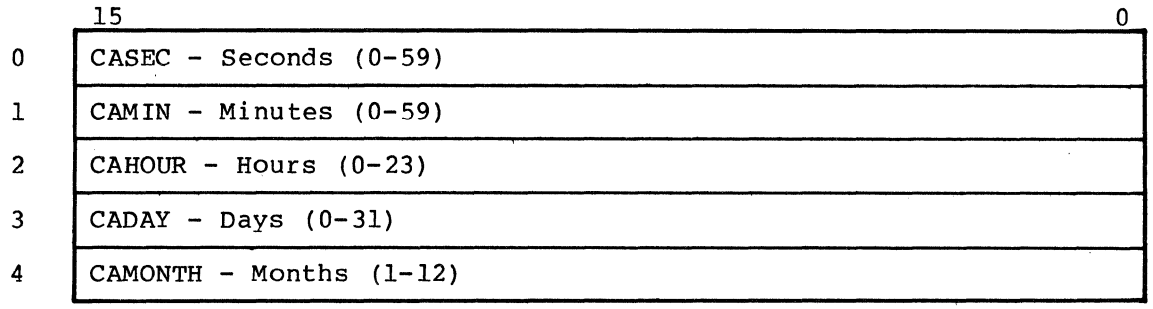


The period is set for each program at build time. The programs in the normal system and their place in the table are shown below:

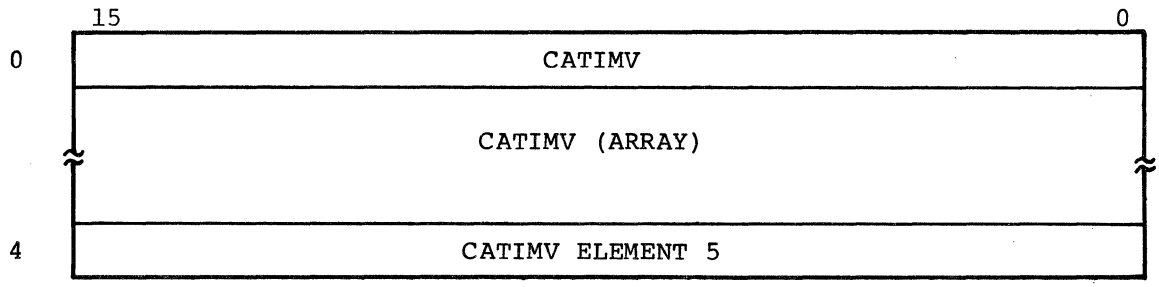
Mnemonic	Element	Meaning/Program
C0LCETMSON	1	ACTIME LCB LIST SCAN, PBLCBTMSCAN
C0ADJUST	2	BUFFER ADJUSTMENT, PBADJUST
C0TUP	3	TEST UTILITY PROGRAM (TUP), PBTUP
C0TIMEOFDAY	4	TIME OF DAY AND DATE/PBTIMEOFDAY
C0T1SEC	5	MUX TTY TIMER, PMT1SEC
C0PSTAT	6	PERIODIC STATISTICS DUMP, PNDSTAT
C0IOTMR	7	I/O TIMAL APPENDAGE, PBTOSRCH
C0CECNT	8	RESET CE ERROR COUNT,
COSPARE	9	SPARE - FOR DEBUG PURPOSE

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.



Overlay for conversion



Loop Forever 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 four types for regulation conditions which PTREGL can check.

RELOGLNK	1	LOGICAL LINK REGULATION level higher than input priority
RELOCAL,	2	LOCAL BUFFER LEVELS sufficient for input
REABL,	3	ALLOWABLE BLOCK LIMIT greater than outstanding block count
REACPINP	4	ACCEPT INPUT flag set

The set of REGLTYPES = REGSET

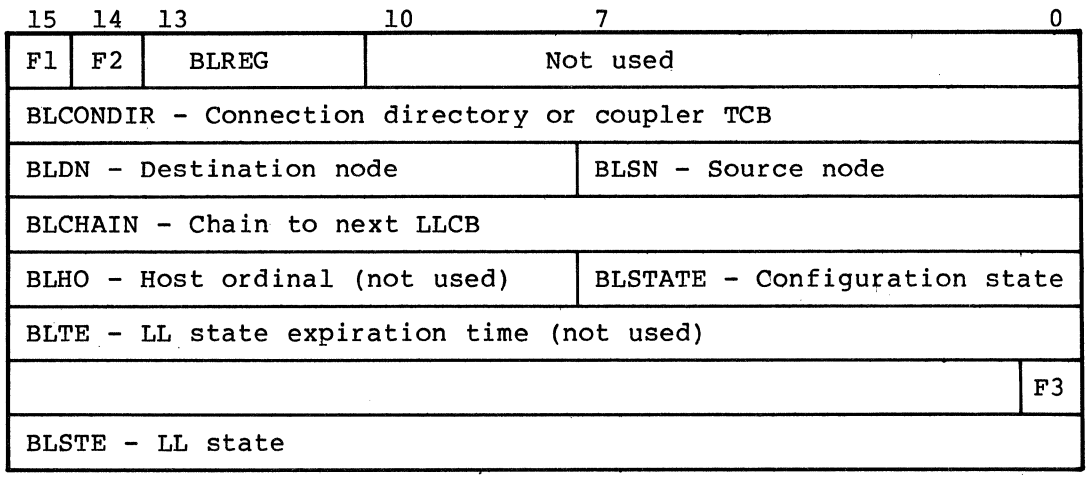
CONTROL BLOCKS

The system structures provide these principal control blocks for network elements:

- LLCBs for logical links } State assignment
- LLCBs for each line }
- 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.



F1 - BLCDS, connection directory flag
F2 - BLINIT, initial LL status SM sent to host
F3 - LL operational
BLREG - Regulation level at this end of link (range 0-7) { 0 = down, 4 = up }

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 subsystem lines) or to the coupler TCB (looking toward host).

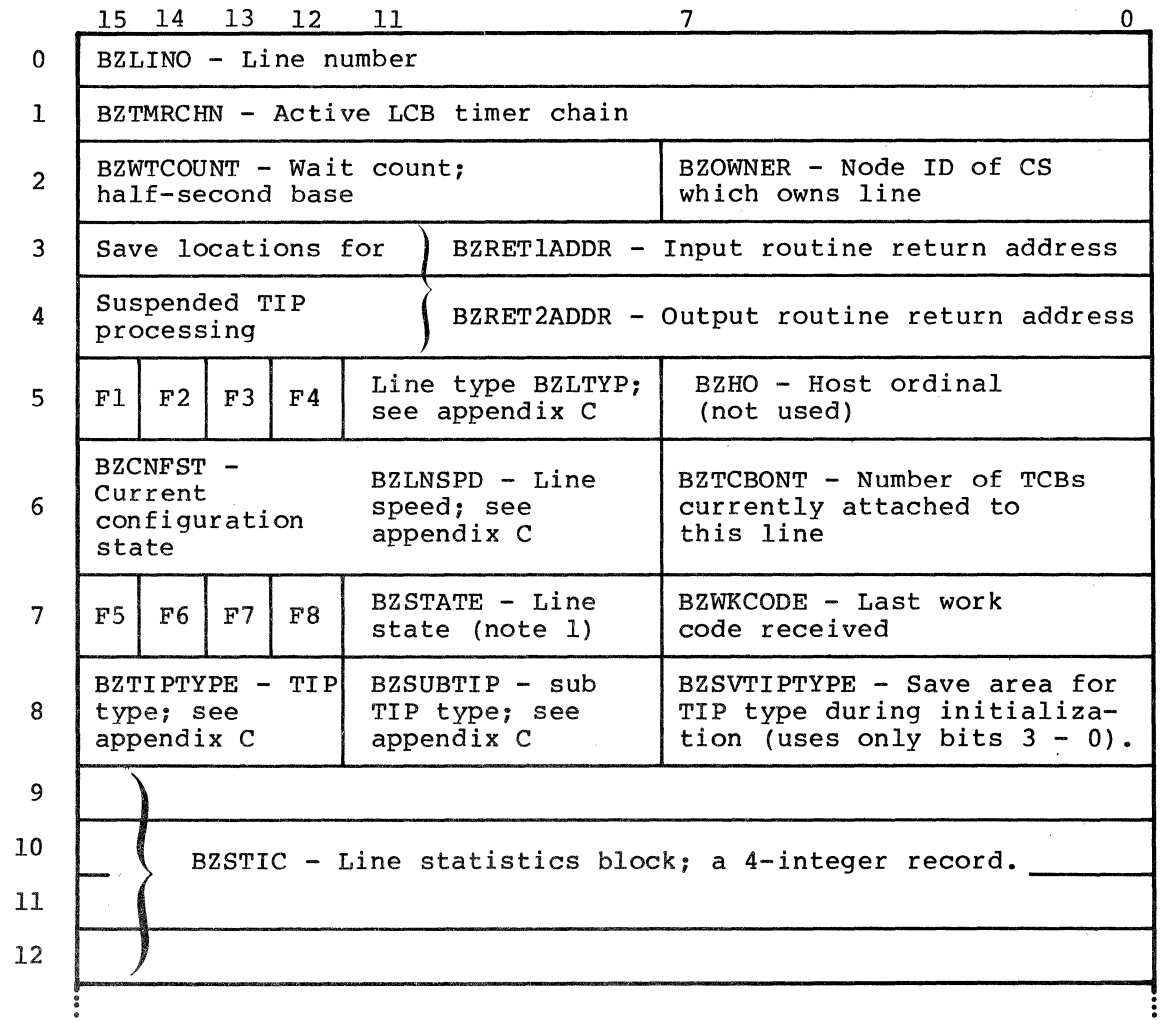
LINE CONTROL BLOCK (LCB), BZLCB

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
13	BZTCBPTR - Pointer to first TCB attached to this line		
14	BZLBTOMUX - Pointer to last buffer given to multiplex subsystem		

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 15 and higher.

Subline control block

	15	7	0
15	BZSUB1PTR - Pointer to first attached subport		
16	BZSP5 - Not used	BZNUMSUBS - Number of subports	

Mode 4 TIP

	15	13	12	7	6	5	0
	BZCURTCB - TCB currently being serviced by TIP						
16	FA	F10	BZKARTASK - Last autorecognition task	F11	F12	BZMAXRETRY - Maximum number of retries for this line	

FA - BZDISABLED, line disabled
BZENABLED, line enable
BZAUTOREC, autorecognition on line
} initialization flags BZINITSEQ

F10 - BZMLTCLS, multiple clusters on line
F11 - BZDELAYLINE, delay service on this line
F12 - BZTBDISABLED, disable line requested

HASP TIP

Two variations are provided as follows:

	15 14 13 12 11 7 6 5 0
15	BZHSWFCS - Workstation function control sequence (FCS)
16	BZHSTFCS - TIP function control sequence
17	BZHSHEAD - Pointer to head of data list queue
18	BZHSTAIL - Pointer to tail of data list queue
19	BZHSCONSOLE - Pointer to address of console TCB
20	BZHSOTCB - Pointer to current TCB address
21	BZHSCCB - Pointer to current continue buffer
22	BZHSIBCB - Input BCB count | BZHSOBCB - Output BCB count | F13 | F14 | BZHSETO - Retry count - errors | F15
23	F16 | F17 | F18 | F19 | BZHSNAK Retry count - NAKS | BZHSRRBITS - Read request bits | F20

Flags:

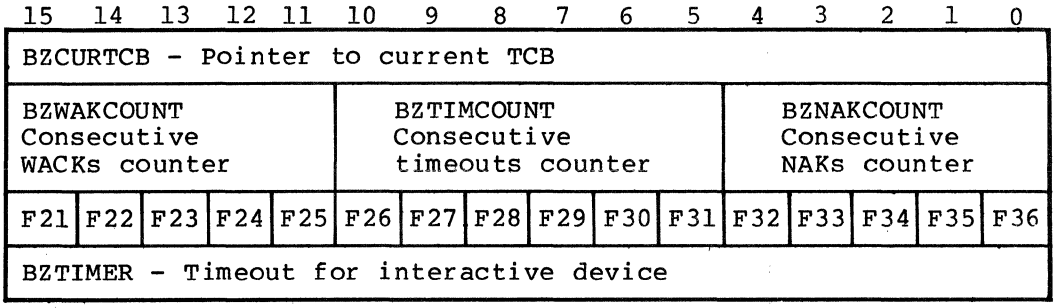
F13 - BZHSGNON, Sign on card seen
F14 - BZHSENQSEEN, Enquiry block seen
F15 - BZHSWOQ, Waiting for output
F16 - BZHSICREG, Suspend card reader command
F17 - BZHSIPREG, Suspend input - buffer regulation
F18 - BZHSXPT, Transparent mode
F19 - BZHSRSBCB, Reset BCB needs to be sent flag
F20 - BZHSLINERR, Line error occurred

HASP TIP Records

	15 0
15	BZHSC
	≈BZHSC (ARRAY) ≈ | 8-word array for clean up purposes
22	BZHSC ELEMENT 8
23	BZHSRQP - Input stream request flags; 16 bits, one per device (0 = must request permission, 1 = permission granted)
	BZHSPNED - Output stream request flags; one per device

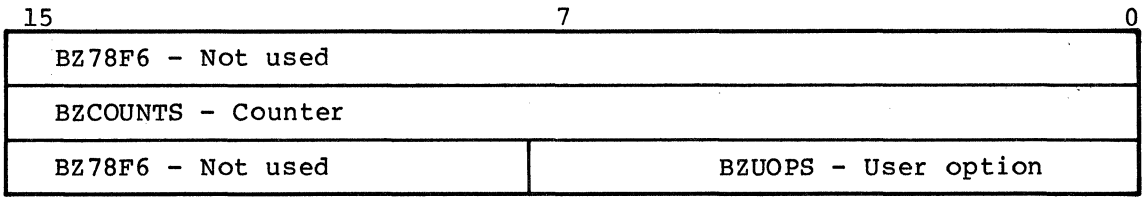
2780/3780 TIP

Two variations are provided:

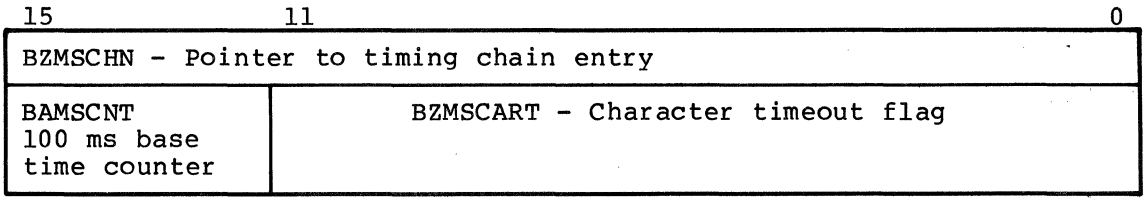


F21 - BZETXRCVD, ETX block received
F22 - BZACKSENSE, toggle bit to indicate if ACK received on last block
F23 - BZBADBLOCK, last received block was bad
F24 - BZETXSENT, ETX block sent
F25 - BZ2629, code translation (026=1, 029=0)
F26 - BZRTIMER, timer running
F27 - BZNOTIFIED, host notified of timeout
F28 - BZ78F2, not used
F29 - BZUOP1, not used
F30 - BZUJOB, not used
F31 - BZUOP3, not used
F32 - BZUOP4, not used
F33 - BZUXLTA, current code translation (026=1, 029=0)
F34 - BZ2780, 2780 terminal (0=3780 terminal)
F35 - BZUOP7, not used
F36 - BZUTR, transparent mode

2nd 2780/3780 TIP



TTY TIP - timed entry



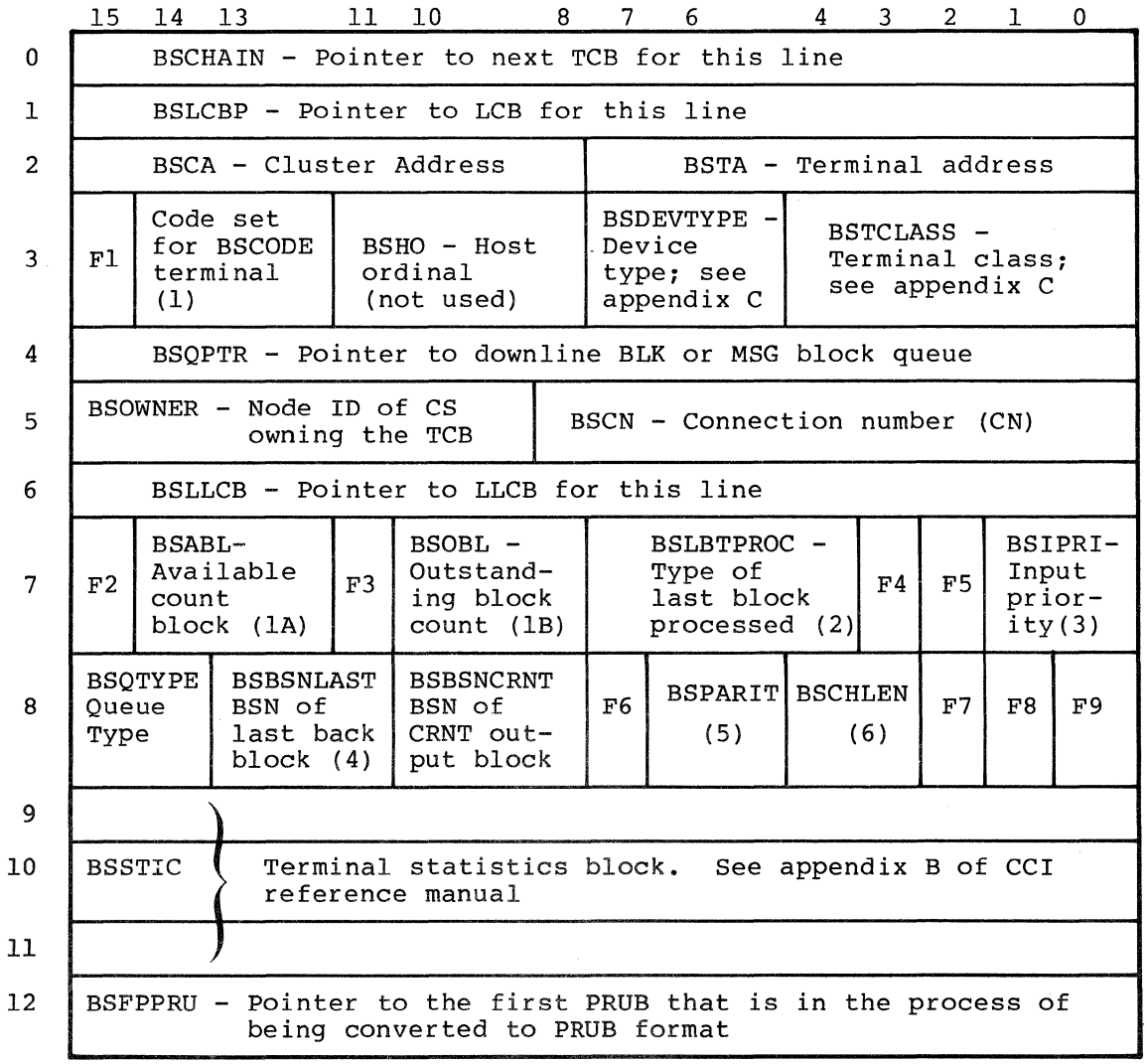
BZMSCART - Uses bit zero only

TERMINAL CONTROL BLOCK (TCB), BSTCBLK

The terminal control block defines terminal-dependent information. One TCB is provided for each terminal in the system. Some terminal devices also have independent TCBs. The first 20 words of the TCB contain common terminal information. The remaining words are used for TIP dependent variations.

Most TCBs are dynamically allocated. However, there is an array of five fixed TCBs (For the MLIA, coupler, etc.). This array of static TCBs (CGTCBs) occupies 160 words.

Allocatable TCBs are released by a line disable condition or by a delete TCB command.



	15	14	13	12	11	7	4	3	2	1	0
13	BSCCPRRU - Character count in partial PRUB						F10	F11	F12	F13	F14
14	BSPRUF - Pointer to upline PRUB queue (ready for sending to HIP)										
15	BSCCPRUS - Character count in a PRUB queue								F15	BSBCKLTR (7)	
16	BSCPTR - Pointer to downline CMD block queue										
17	BSOTPP - Pointer to text processing parameters										
18	BSPGWIDTH - Page width							BSPGLENTH - Page length			
19	F16	F17	BSNUMR (8)	BSXBLKLENTH - Transmission block length							

Notes:

1. SBCODE, see subTIP type table in appendix C.

2. BSABL is the largest number of interactive blocks that can be sent upline without a BACK block acknowledging them. In the standard system BSABL = 1.

3. BSOBL is the number of blocks that have currently been sent upline without being acknowledged with a BACK block.

4. BSIPRI - Input priority

5. BSBSNLAST,
   See block protocol in section 6, $0 \leq BSN \leq 7$
   BSBSNCRNT
   See block protocol in section 6, $0 \leq BSN \leq 7$

6. BSPARITY - Parity type:

   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, must send a BACK block for this batch PRUB (the PRUB has already been text processed. Flag is set by PBIOPOI. It is reset by PBDNABRT when discarding a block, or by PBBLKCHK when a BACK block is sent.

9. BSNUMR - Number of records in a block

Flags:

F1 - BSSTOP, data stream to this terminal stopped
F2 - BSINOP, terminal inoperative
F3 - BSRES1, not used
F4 - BSACPINP, terminal accepts input for host
F5 - BSACPOUT, terminal accepts output from host
F6 - BSTBTERM, TCB is to be deleted
F7 - BSPGWAIT, in page wait mode
F8 - BSXPARENT, input data in transparent mode
F9 - BSHOTOGL, host ordinal toggle bit (not used)
F10 - BSBTCH, batch (PRU) terminal
F11 - BS2629, 026/029 code; 1=026, 0=029
F12 - BSEM, EM punch required for a short record
F13 - BSDROPEDI, discard repeated EOI block
F14 - BSXPTOEOI, transparent to EOI
F15 - BSBCKDUE, awaiting downline BACK block
F16 - BSSUPCC, suppress carriage control
F17 - BSBAN, banner off (PRU records)

The following queues are controlled by the TCB:

UPLINE:

- The PRUB queue, located through pointer BSPRUF. Blocks in this queue are ready to be routed by PBRTEPRU. Associated field for this queue is BSCCPRUS (total character count for all blocks in the queue). Note that both batch and interactive blocks use the queue. The TCB itself can be for either a batch or an interactive device.

- A partial PRUB queue, used during PBPIPOI's conversion of a block to PRUB format. When the conversion is complete, the block is moved into the PRUB queue. Pointer to this queue is BSFPPRU. Field associated with this queue is BSCCPRRU (character count of all blocks in the partial PRUB queue).

DOWNLINE:

- Output command queue:

  - For both batch and interactive devices, CMD blocks (which regulate the data stream) are queued to a special command queue located through pointer BSCPTR. When the TIP processes this queue, all entries are processed during a single TIP pass.

  - For interactive devices, BACK blocks are handled entirely by PBIOPOI. These blocks are not queued to the TCB; therefore, the blocks cannot be processed by the TIP.

  - For batch devices, some BACK blocks are partially processed by PBIOPOI. All BACK blocks are queued to the output command queue.

- Output data (MSG and BLK) block queue (located through pointer BSQPTR)

  - Interactive Mode 4 blocks are queued here by PT4QIA (which is called through PBIOPOI) if they are BLK blocks. The TIP is not notified that there is data to be processed. The TIP is notified when all the blocks of the message are queued; that is, when the MSG block marking the end of the message is ready to be processed.

- Interactive blocks for all other TIPs is queued here. PBQ1BLK queues the block on call from PBIOPOI; then PBQ1BLK notifies the TIP that the block is available for processing unless other blocks are already in queue waiting to be processed.

- Batch blocks: PBIOPOI calls PRUTOBLKS to text process the blocks, then PBQ1BLK queues the block. The TIP is notified that output is available, unless other blocks are already in the queue waiting to be processed.

Associated field: BSBLKLTR (the send-a-BACK-block-later flag) must be set in all cases.

MLIA Handler (static TCB)

	15 14 ... 0
20	F18 | BSWRKCO - Process state work code (bits 0 and 1 only) 0=SBTIMOUT 1=SBRD STATUS 2=SBHALT 3= ---
21	BSCONB - Condition B counter (input drop errors)
22	BSCONC - Condition C counter (last data errors)
23	BSCOND - Condition D counter (Input error ODD first in, first out error)

Flag

F18 = BSCEMI - CE Error SM issued

MODE 4 TIP

There are two overlays for the Mode 4 TIP.

	15	14	13	12	11	10	9	8	7	6	5 ... 0
20	F19	F20	BSERRODE - Number of bad (error) responses			BSTASK, task index (1)				BSERRICOUNT - Number of no response errors	
21	F21	F22	F23	F24	F25	F26	F27	F28	F29	F30	BSERR2COUNT - Number of bad responses received
22	F31	F32	F33	F34	F35	F36	F37	F38	F39	F40	BSM4COUNT - General purpose counter
23	BSM4POLL										
	BSM4POLL (ARRAY)										Poll Message
30	BSM4POLL ELEMENT 8										
31	BSCLSPTR - Pointer to cluster TCB										

(1) BSTASK task index 0-31. These tasks are defined in the local data area for the Mode 4 TIP (PTMD4TIP) flags

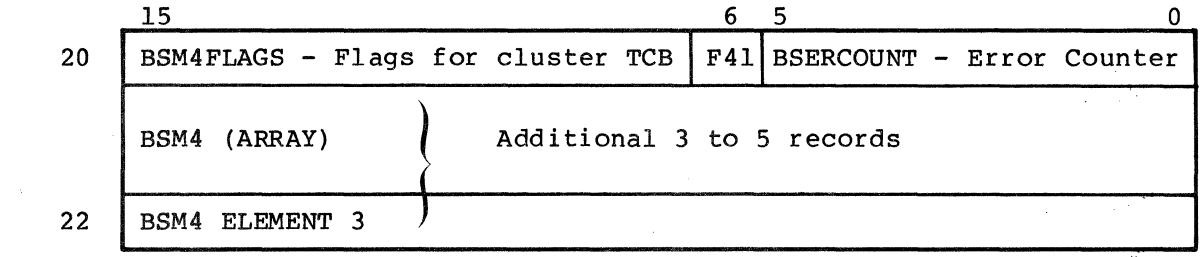
Flags:

F19 - BSM4CRPS, configuration poll message sent
F20 - BSHOG, hog control flag
F21 - BSCRON, card reader on
F22 - BSPRON, printer on
F23 - BSBATCH, batch interrupt
F24 - BSLASTDEV, last device used for I/O (1=batch, 0=interactive)
F25 - BSTOGKNOWN, toggle is known
F26 - BSTOGSTATEXP, expected toggle state
F27 - BSCLSDOWN, cluster is down
F28 - BSHOLD, TCB control in hog mode
F29 - BSEOICRO, skip blanks during EOI card reading
F30 - BSM4S2, not used
F31 - BSM4FAIL, terminal in failure mode
F32 - BSM4WRQ, mode 4 queue flag (write request message, blank fill message, or print message (PM) is queued)
F33 - BSPFECODE, escape to poll for E-code
F34 - BSACKOUT, acknowledge last write request is still outstanding
F35 - BSLIO, last I/O state (0=write, 1=poll)
F36 - BSWAIT, wait for resume
F37 - BSCLRCNT, clear error counters
F38 - BSINPUT, input accepted (A1)
F39 - BSTOGRECEIVED, toggle state last received
F40 - BSTOGCORRECT, toggle state is correct

} Toggle for read/write made at console

Second Mode 4 TIP overlay

Four to six records as shown in first word



F41 - BSM4D2, not used

Coupler TCB (static TCB)

This is the TCB used by the HIP for transfers to/from the host.

	15 14 ... 7 ... 0
20	BSCPAVPTR - Pointer to first available output buffer
21	BSCPLAST - Pointer to last available output buffer
22	BSCPINPUT - Input buffer address
23	BSBUFOTT - Memory address loaded address
24	BSCPSTATUS - Coupler status
25	BSCPDATA - Orderword storage
26	BSCPCMD - Last NPU status word sent to host
27	BSCPBUFAV - Number of available buffers | BSCOBZSTA - Previous state
28	BSCPAMASK - Coupler interrupt mask
29	F42 | BSCPIDLT - Idle timeout counter
30	BSCPCONN - Coupler connection number

Flag:

F42 - BSCPHST, host status

1 = host available
0 = host down

HASP TIP

	15 14 13 12 11 10 9 8 7 ... 0
20	BSHSOBUFF - Pointer to current output buffer
21	BSHSHEAD - Pointer to head of data list queue
22	BSHSTAIL - Pointer to head of data list queue
23	BSHSQBF - Pointer to buffer for data list queue
24	BSHSSTMR - Suspend transfer timer
25	BSHSFCSM - Stream mask for function control sequence (FCS)
26	BSHSIMD | F43 | F44 | F45 | F46 | F47 | F48 | BSHSIOK - Start input received

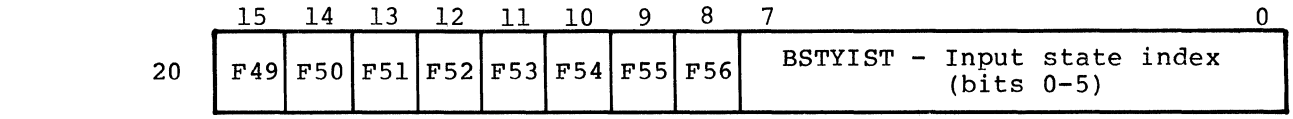
BSHSIMD, Card reader mode

3 = transparent
2 = nontransparent
1 = EBCDIC - 026 card
0 = EBCDIC - 029 card

Flags:

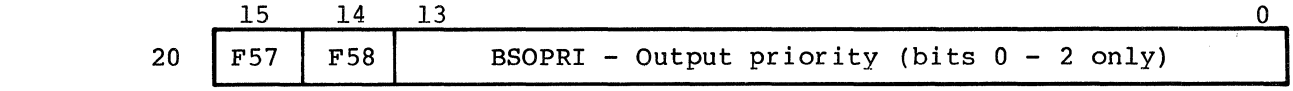
F43 - BSHSEOISNT, EOI BVT record sent
F44 - BSHSRSNT, request permission to transmit sent
F45 - BSHSPNED, request permission is needed to start transfer
F46 - BSHSOIP, output in progress
F47 - BSHSSUSP, data stream output stopped message sent to host
F48 - BSHSSF, countdown for output stopped condition

TTY TIP



F49 - BSTYLF, echo line feed
F50 - BSTYCR, echo carriage return
F51 - BSTYTAPE, paper tape mode
F52 - BSTYRGL, regulation in effect
F53 - BSTRXON, send an X-ON to paper tape
F54 - BSTYHOUT, hold output
F55 - BSTYBLK, block mode
F56 - BSTYKEY, keyboard mode

2780/3780 TIP



Flags:

F57 - BSSUSPIO, suspend I/O
F58 - BSINPOK, OK to input

MULTIPLEX SUBSYSTEM

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) that 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 communication from either the TIPs or service module to 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	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

Variation: Input operation

	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	NKIFCD- Optional FCD of I/O buffer
4	F14	F15	NKDM9		NKBLKL - Block length (words)				

Flags:
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 to be changed

NKDM9 - not used

Set up for input processing (call from TIP)

	15 — 7	6	5 — 0
0	NKWD0 - Word 0 and 1 of universal overlay		
1	NKWD1 - Word 0 and 1 of universal overlay		
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		

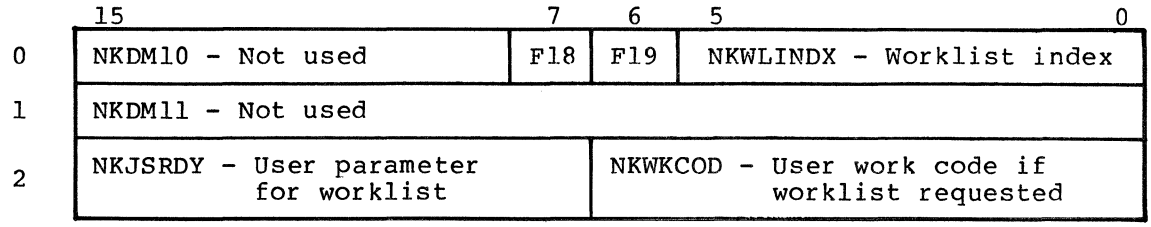
F16 - NKMVB, Move user bits to LCB
F17 - NKRPRT, Strip parity flag

Universal input

	15 — 0
0	NKDM7 - Not used
1	NKDM8 - Not used
2	NKWD2
3	NKWD3
4	NKWD4
5	NKWD5
6	NKWD6
7	NKWD7

} Universal overlay words

Terminate I/O command



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.

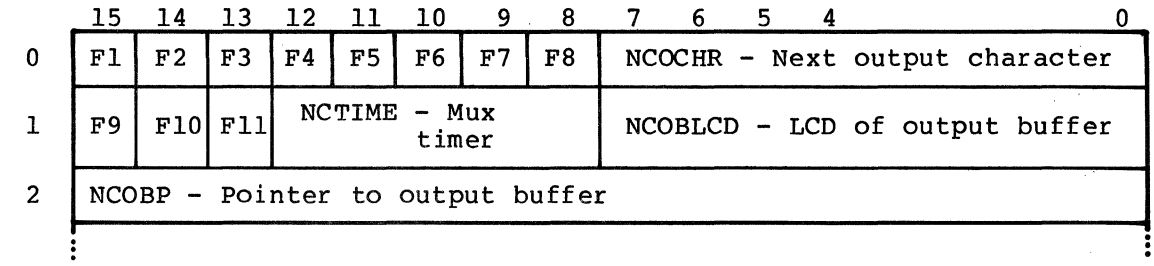
Mnemonic	Value (hex)	Meaning
NKTURN	3	TURN LINE AROUND
NKINIL	4	INITIALIZE LINE
NKENBL	5	ENABLE LINE
NKINPT	6	INPUT
NKDOUT	7	DIRECT OUTPUT
NKOBT	8	OUTPUT BUFFER TRANSMITTED
NKINOUT	9	INPUT AFTER OUTPUT
NKENDIN	A	TERMINATE INPUT
NKENDOUT	B	TERMINATE OUTPUT
NKDISL	C	DISABLE LINE
NKCLRL	D	CLEAR LINE
NKCONTROL	E	CONTROL
NKSPECIAL	10	UPDATE MUX TABLE

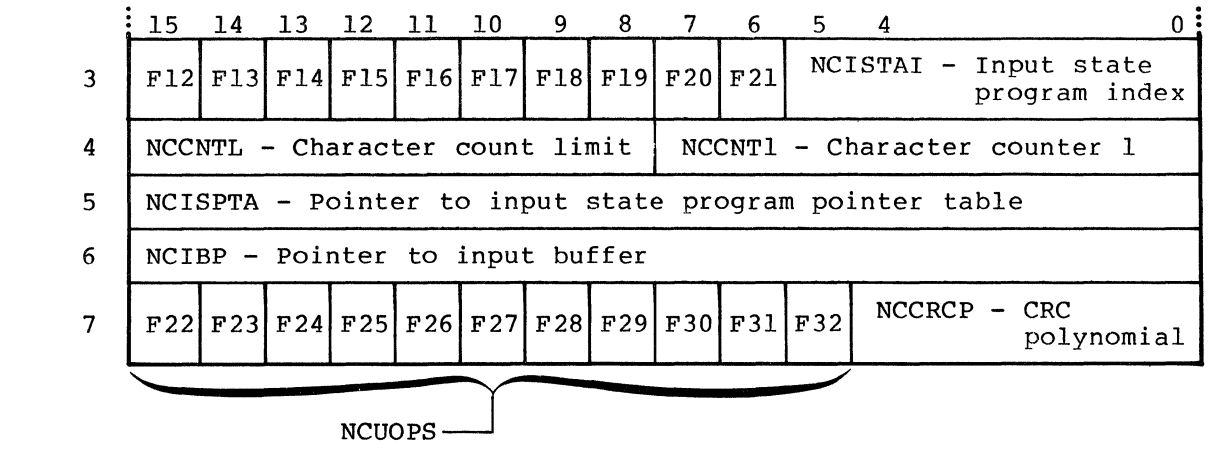
MULTIPLEX LINE CONTROL BLOCK (MLCB), NCLCB TEXT PROCESSING CONTROL BLOCK (TPCB)

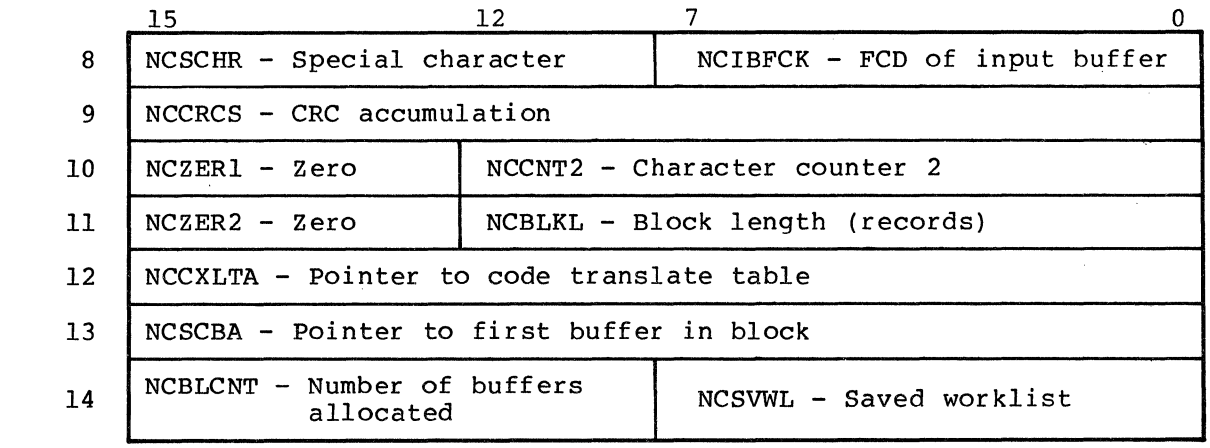
The MLCB is a dynamically allocated buffer obtained and released as a result of requests issued by the TIPs. The MLCB defines the processing functions to be provided by the multiplex subsystem. For a given communications line, there is one MLCB for each enabled line.

Seven variants of the MLCB are provided.

Usual TIP I/O data transfer request







Flags:

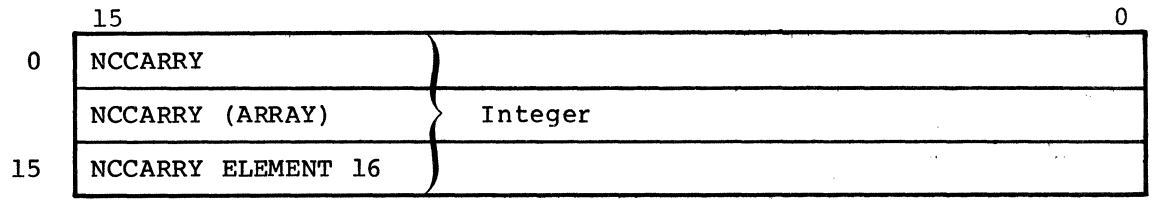
F1 - NCEOBL, end of block
F2 - NCNXOCA, next output character available
F3 - NCLCT, last character transmitted (CDCCP)
F4 - NCBCREQ, buffer chaining required
F5 - NCOMPRO, output message in progress
F6 - NCSP1, spare
F7 - NCODDIN, ODD received
F8 - NCSP1, spare
F9 - NCSUPCHAIN, suppress buffer chaining
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

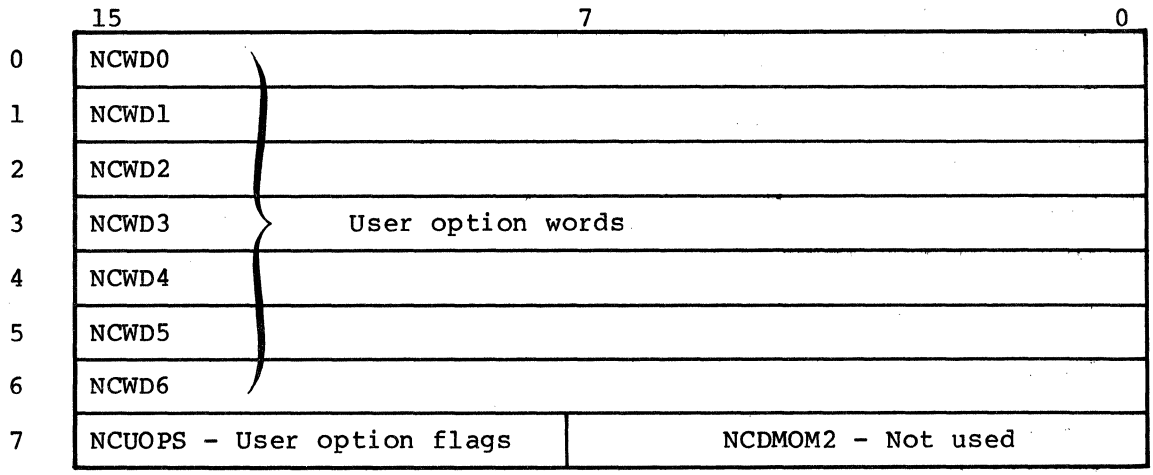
Optional user flags; can also be addressed as a single field NCUOPS

F30 - NCETX, delay ETX worklist generation
F31 - NCMRTO, modem response timed out
F32 - NCCARR, line carrier type
1 = controlled
0 = constant)

Sixteen integer words



Eight user option words includes half a word of flags



TPCB

	15 14 7 5 0
0	NCLCDFCD Integers
1	NCDUM2 -
2	NCSBP - Source buffer pointers
3	F33 | NCFFLGS - Text processing firmware (see F13 - F21 of MLCB) source flags mask | NCSTAI - Index to state programs
4	NCDUM4 - Not used
5	NCSPTA - Pointer to state programs table
6	NCDBP - Pointer to destination buffer
7	NCDUM5 - Not used
8	NCDUM6 - Not used | NCBFCD - FCD of buffer
9	NCDUM7 - Not used
10	NCDUM8 - Not used
11	NCDUM9 - Not used
12	NCDUMA - Not used
13	NCFDBA - Pointer to first destination buffer
14	NCDUMB - Not used
15	NCDUMC - Not used
16	NCDUMD - Not used
17	NCDUME - Not used
18	NCFSBA - First source buffer address

Flag:

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	
16	NCTPF1	
	NCTPF1 (ARRAY)	Spare for 16 file 1 registers
31	NCTPF1 ELEMENT 16	

Batch TIPs - Variant for data compression on batch device (TPCB)

	15	0
0	NPAD1	
	NCPAD1 (ARRAY)	19 integers
18	NCPAD1 ELEMENT 19	
19	NCDBLC - Destination block count	
20	NCCOUNT - Counter	
21	NCCLIMIT - Count limit	
22	NCBINIT - Blank initial count	
23	NCBLIMIT - Blank limit	
24	NCDINIT - Duplicate initial count	
25	NCDLIMIT - Duplicate limit	
26	NCUINIT - Unlike initial count	
27	NCULIMIT - Unlike limit	
28	NCSVCH - Save character for compression	
29	NCRCB - Record control byte	
30	NCPAD2 - Not used	
31	NCPAD3 - Not used	

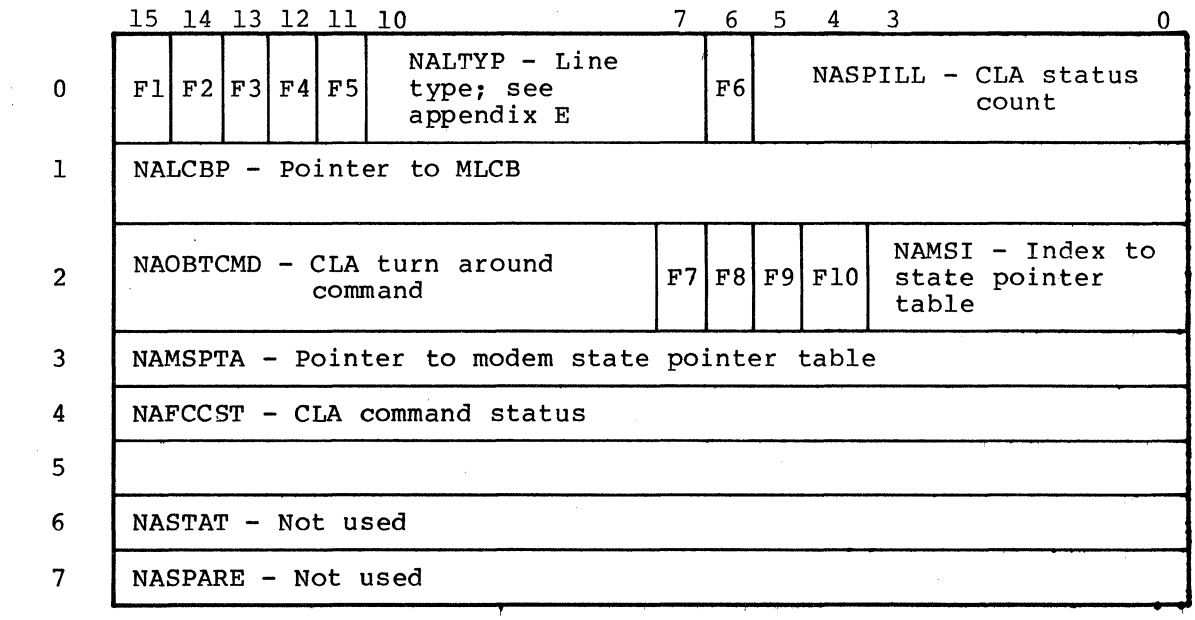
2780/3780 TIP TPCB

	15	0
0	NC78D - Integer	

PORT TABLE (NAPORT)

A multiplex subsystem 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 input state programs which process data for the multiplex subsystem. The port table points to the modem state program pointers directly. Four variants are provided.

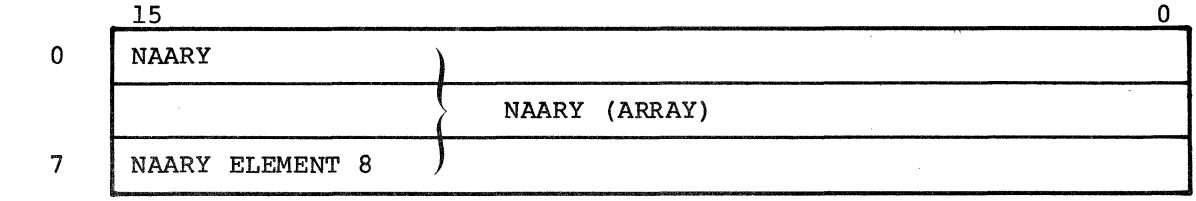
Normal Port Table



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 detected signal (DCD) dropped
F8 - NAMTO, modem timeout in progress
F9 - NAWAIT, timeout flag for first overflow
F10 - NAOVFE, first status overflow worklist received

Clearing Port Table Variant

This table of 8 integer entries can be used to clear all of the port table.



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

Overlay Array

15 — 0	
B0OVERLAY	
	NAOVERAY - 8 words
B0OVERLAY	

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

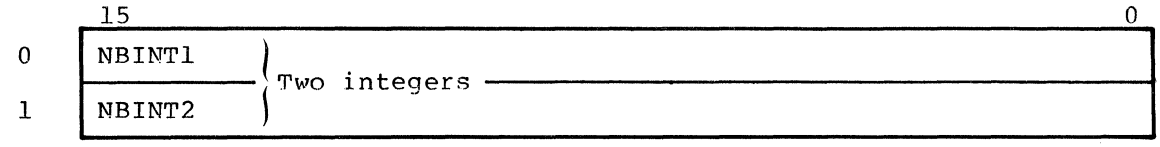
	15	13	12	11	10	9	8 — 4	0
0	NBSP1	F1	F2	F3	F4	F5	NBMODCLS - Modem class; see below	NBOTYP - CLA type; see CLA constants above
1	NBAND - Mask							

NBSP1 - Not used

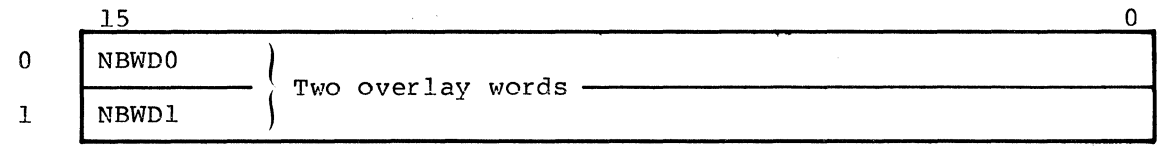
Flags:

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

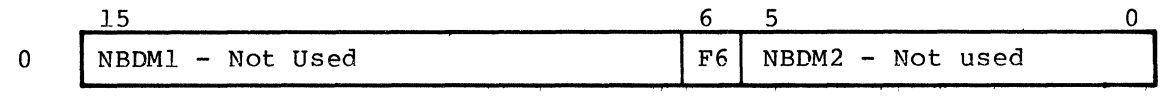
Integer Entry



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.

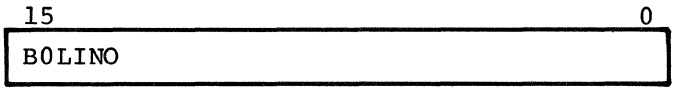
Mnemonic	Value (hex)	Meaning
NOLDIAG	0	RESERVED FOR ON-LINE DIAGNOSTICS
NOL1	1	2560-1 201A SW HDX CONTR 2WIRE
NOL2	2	2560-1 201B DE FDX CONTR 4WIRE (HDX MODE)
NOL3	3	2560-1 201B DE FDX CONST 4WIRE
NOL4	4	2560-1 208A DE FDX CONST 4WIRE
NOL5	5	2560-1 208A SW HDX CONTR 2WIRE
NOL6	6	2561-1 103E SW FDX CONST 2WIRE
NOL7	7	2561-1 103E DE FDX CONST 2WIRE
NOL8	8	2561-1 202S RS 232 103E/113 SW HDX CONTR 2 WIRE
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

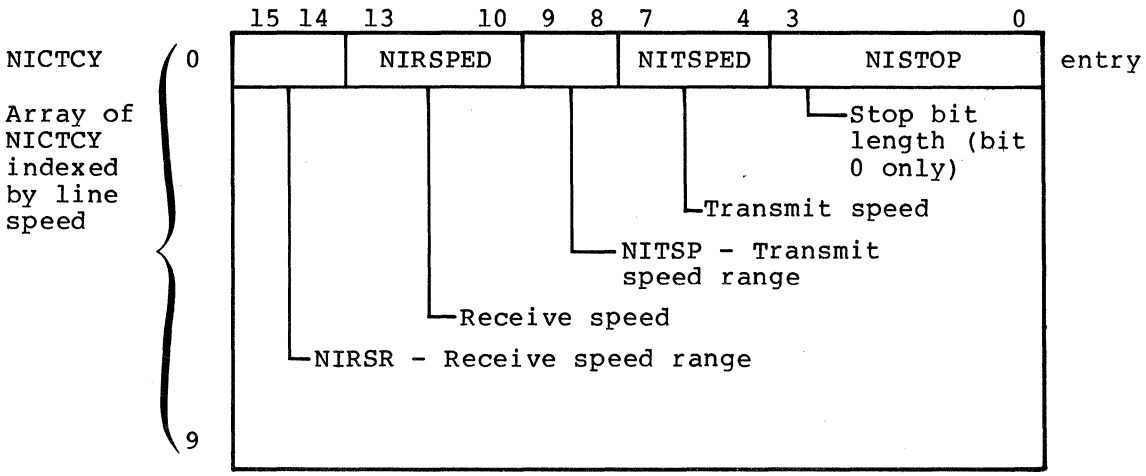
Line Number Field, B0LINO

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 Characteristics 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 0 = 110 1 = 134.5 2 = 150 3 = 300 baud

NIRSPED - See appendix C

NITSP 0 = 110 1 = 134.5 2 = 150 3 = 300 baud

NITSPED - See appendix C

NISTOP - 0 = 1 stop bit
- 1 = 2 stop bits
} for character delimiting

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	Synchronous 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

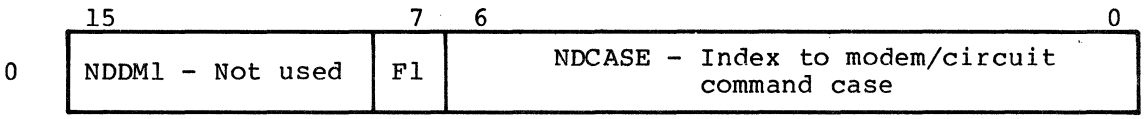
CLA Commands and Status

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.

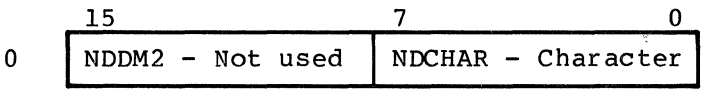
Normal Entry



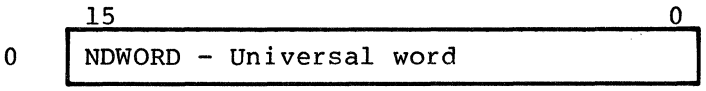
F1 - Set function flag (0 = reset)

NDCASE is defined in the table below.

Character Overlay



Universal Overlay

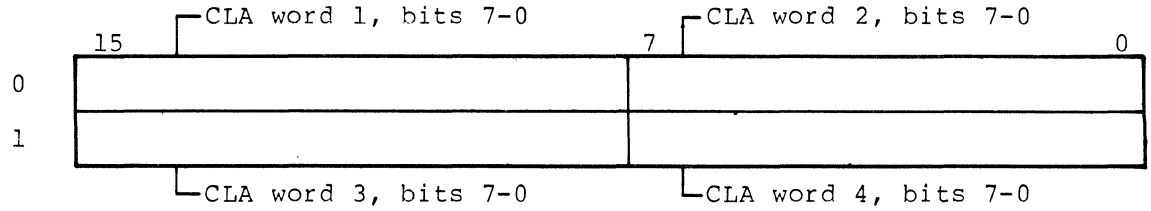


Multiplex CLA Command Status Table Entries, NFCCSE

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 CLA 8-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 — CLA word 1, bits 7 through 0

NFW2B7 ... NFW2B0 — CLA word 2, bits 7 through 0

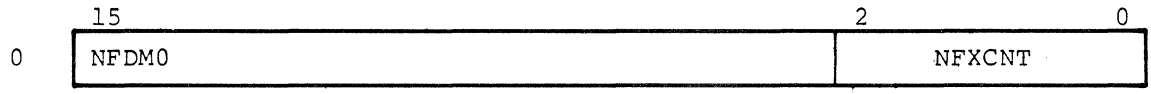
NFW3B7 ... NFW3B0 — CLA word 3, bits 7 through 0

NFW4B7 ... NFW4B0 — CLA word 4, bits 7 through 0

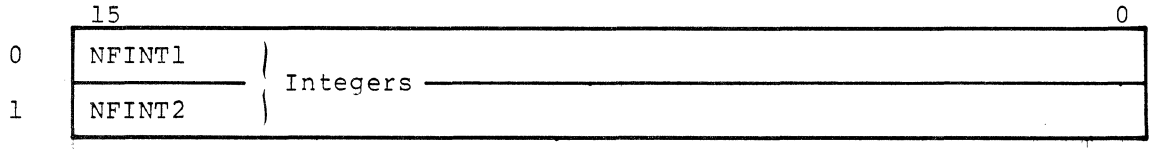
See table that follows for flag usage.

SDLC CLA ENTRY

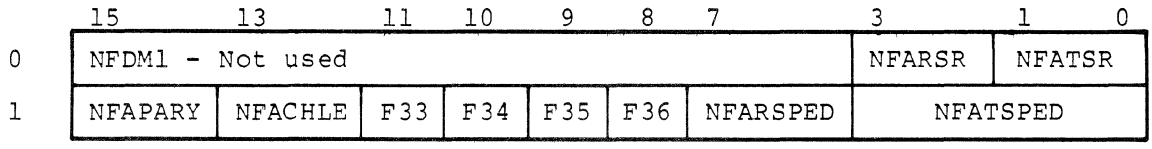
Defines SDLC CLA bit assignment.



WHOLE WORD VARIATION



ASYNC CLA ENTRY



NFARSR - Receive speed range (baud)

0 = 110
1 = 134.5
2 = 150
3 = 300

NFATSR - Transmit speed range (baud)
0 = 110
1 = 134.5
2 = 150
3 = 300

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

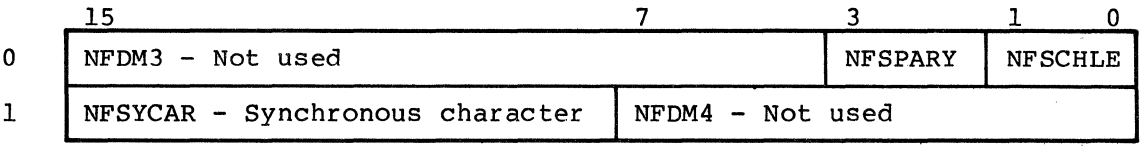
F35 - NFECHO, echoplex mode

F36 - NFLBT, currently in one-line diagnostic loopback test

NFARSPED - Receive speed (baud)
0 = 110
1 = 134.5
2 = 150
3 = 300

NFATSPED - Transmit speed (baud)
0 = 110
1 = 134.5
2 = 150
3 = 500

Synchronous CLA Entry



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, M0SCTYP

The status indicators are used in the worklist entry.

```
M0CLAON,     0    CLA ON DETECTED
M0RING,      1    RING INDICATOR DETECTED
M0ENBL,      2    LINE ENABLED
M0HERR,      3    HARD ERRORS DETECTED
M0SOER,      4    SOFT OUTPUT ERRORS DETECTED

M0SIER,      5    SOFT INPUT ERRORS DETECTED (unsolicited input)
M0STRT,      6    START MODEM TIMEOUT
M0STOP,      7    STOP MODEM TIMEOUT

M0OVRF,      8    CLA STATUS OVERFLOW (unsolicited output)
M0OVTO,      9    CLA STATUS OVERFLOW TIMEOUT
M0MRTO,      A    MODEM RESPONSE TIMEOUT
M0BREAK;     B    BREAK FROM FRAMING ERROR STATUS
```

Modem Control States

These states 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

Modem State Prorams

N0MSPT 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 that 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 interactive terminal parameters are used.

	15	14	13	12	11	9	8	7	3	0
0	NJISPTA - Address of input state programs pointer table									
1	NJCXLTA - Address of code translate table									
2	NJCNT1 - Input character count 1								NJSYNC - Sync character	
3	NJCRCP - CRC polynomial 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 (bits 0 -7)									
7	NJPGLENGTH - Page length (bits 0 - 7)									
8	NJCANCHAR - Character for cancel input line (bits 0 - 7)									
9	NJCNTRLCHAR - Control character (bits 0 - 7)									
10	NJUSF1 - Character for user break 1 (bits 0 - 7)									
11	NJUSR2 - Character for user break 2									
12	F3	F4	F5	F6	NJXCNT - Counter for transparent character					
13	F7	F8	F9	F10	NJOUTDE- output device	NJAPL- APL mode	NJXCHAR - Character that delimits transparent text			
14	NJBSCHAR - Backspace character						NJABTLINE - Character to abort output line			
15	NJCRIDLES - Count of idles following a CR						NJFIDLES - Count of idles following an LF			

IVT parameters are in words 6 through 11 IVT variant.

IVT variant

	15	0
0	NJARRY	
	NJARRY (ARRAY)	Overlay for interactive terminal parameters
14	NJARRY ELEMENT 15	

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 - NJXCCON, expected delimiter is reaching transparent character count
F5 - NJXCHRON, delimiter is transparent character
F6 - NJOM1, 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 Classes

This is the BZTIPTYPE field of the LCB. Further information on terminal class is found in appendix C.

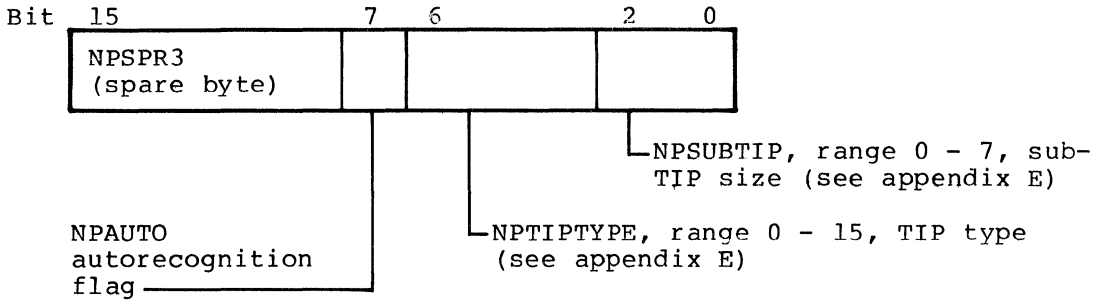
Mnemonic	Value	Meaning
N0TMLIA	0	MLIA (Multiplex interface adapter)
N0M33	1	Async - M33, M35, M37, M38 (TTY terminals)
N02780	7	2780
N03780	8	3780
N0HASP	9	HASP
N020OUT	10	Mode
N0COUPLER	11	Coupler
N0TCONSOLE	12	Console
N0TDIAG	13	Diagnostics

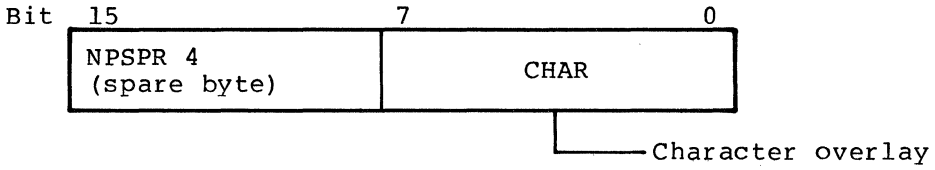
Terminal and Device Types (TT/DT)

These data structures are used to find TCBs, check devices for deliverability of messages, and so forth. See appendix C.

TERMINAL TYPE, NPTT

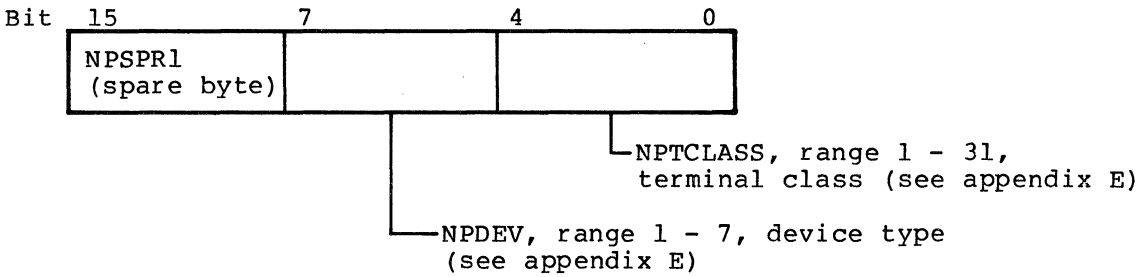
Three cases are possible:

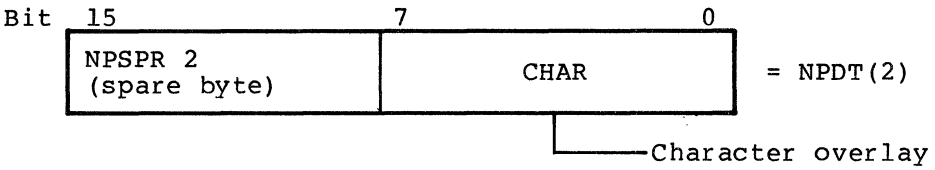




DEVICE TYPE, NPDT

Two cases are possible:





= NPDT(2)

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.

Other service message information is at the following location.

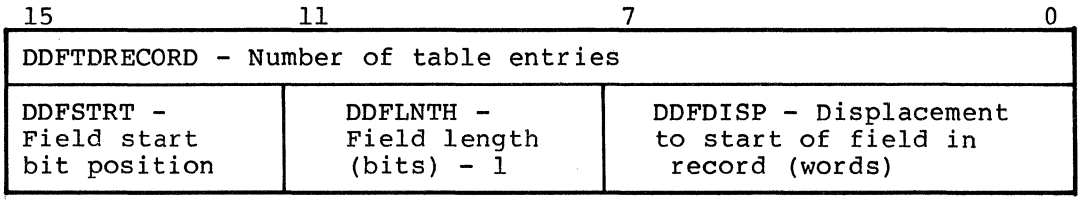
Definition	Fields	Location, table	
TIP/Sub-TIP	N0...	Appendix C, C2	
Line type		Appendix C, C3	
Configuration states	C7...	Appendix C, C4	
CE error messages	CN...	Appendix C, C2	
Statistics messages			
(NPW)	CP...	Appendix B, B4	CCI reference manual
(tr/ln)	BZ...	Appendix B, B4	
(term)	BS...	Appendix B, B4	

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 1-word entries (DDFDTENTRY) follow.



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/FV pair in the configure/reconfigure service message. Normal values for the entries can be found in the Link Edit listing normal table entry.

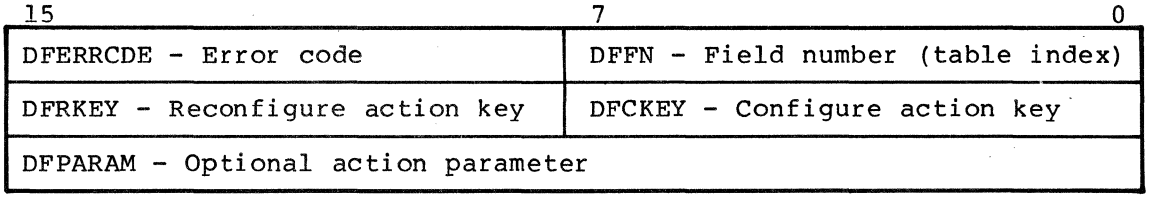
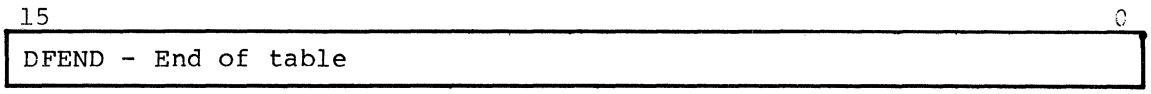


Table end



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 routine.

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 (not used)
D2INK	7	GET INDEX INTO LINK TABLE FROM LRN
D2QCB	9	GET A QUEUE CONTROL AND SEND INIT (not used)
D2PARITY	10	PROCESS PARITY
D2INIT	15	EMPTY OUTPUT QUEUE AND SEND INIT (not used)
D2TCBINIT	16	SET UP VARIANT TCB

Configure action error codes - If the action specified by the action table cannot be completed, a 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/reconfigure SM reply (normal or error) to the host.

DFERRCDE

D3AC	0	Action complete
D3FNFVERR	1	Field number of 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
D3INVTT	5	Invalid terminal type
D3INVDT	5	Invalid device type
D3NOTENABLED	6	Line not enabled
D3NOL	8	Logical link not established
D3CNINUSE	9	Connection number already in use

HALT CODES

The halt codes delivered to the NPU console are shown in appendix B, of the CCI reference manual.

ON-LINE DEBUGGING AIDS I

The on-line debugging aids for CCI 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 CCI 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{C/L},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{P/L}, 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{B/L},line number/
Dump LLCB, TCB	TC{B/L},DN, SN, CN/
Search for TCB	TS{B/L},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 always 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 reentering 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 an 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 user 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 are 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-FF16)/
* 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 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
XA16       Register snapshot
XB16       Breakpoint return address snapshot
XC16       Spare breakpoint
XD16       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 prebuilt 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 prebuilt 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.

```
{LCB}
{LCL}, 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 parameters 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 different priority levels to be breakpointed at a specific priority level or levels.

```
BL, priority level (0..11₁₆)/
*
```

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..11₁₆)/
```

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 1-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 debugging 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 debugging 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

Abort
  Downline 6-35
  Upline 6-35
Accept
  Input Flag, PTINIT 6-35
  Output Flag, PTINIT 6-35
ACK 11-6
ACK0 11-8
Acknowledgment Block, HASP 11-6, 11-7
Address 4-23, 6-2
  Functions 4-24
  HASP 11-16
  Mode 4 10-5
  Mode 4 Terminal 10-3
  Register Code 7-14
  18-Bit 4-24
Alternating Directories 6-16
Analyzer, CLA Status 5-22
ASCII
  Decimal Conversions 4-20, 4-21
  Hexadecimal Conversions 4-20, 4-22
  Set Membership 4-20
Assignments, Interrupt 4-16
Asynchronous (TTY) TIP 9-1
Autorecognition
  BSC 8-13
  HASP 11-25
  Mode 4 10-14
  TTY 9-6
Availability, Buffer 4-7

BACK Block 6-7
Base System Software 4-1
Basic Interrupt Processing 4-13
Batch Carriage Control
  Action, 2780 8-9
  Action, 3780 8-11
  Symbols 8-4
Batch Format Data 6-10
BCB 11-8
  Error 11-23
  Error Block 11-24
Binary 4-20
Binary Codes, BSC 8-2
Binary Synchronous Communications TIP 8-1
Binary Conversions
  to ASCII Decimal 4-21
  to ASCII Hexadecimal 4-22
Bit Assignment, Coupler Status Register 7-11
BLK (Block) Block 6-7
Block 6-35
  ACK 11-6
  Acknowledgment 6-7
  BACK 6-7
  BLK 6-7
  BSN Type 6-6
  CMD 6-7, 6-35, 10-10
  Card Reader CMD 11-18
  Command 6-9
  Control Byte (BCB) Error 11-24
  Control Byte Error, HASP 11-23
  Control Byte, HASP 11-8
  Data 6-4
  Discard Non-routable 6-35
  FCS Change 11-13
  Flow Control 6-3
  Format, Data 7-25
  Format, HASP Signon 11-15
  Functions 4-24
  HASP
    Control 11-6
    Data 11-8, 11-12
    EOF 11-13
    End-of-File 11-13
    Enquiry 11-7
    FCS Change 11-14
    Idle 11-8
    Multileaving 11-6
    Operator Console 11-13
    Signoff 11-16
    Signon 11-15
  Length, Compare 4-24
  Make-up Error 11-23
  Main Memory, Clear 4-24
  Printer CMD 10-11
  Printer Data CMD 11-20
  Protocol 1-9, 6-1
  Protocol Summary D-1
  Punch Data CMD 11-22
  Routing 1-13
  Serial Number (BSN) 6-6
  Transfer 7-16
  Types 6-6, 6-7
Break Codes 10-8
Breaks, Upline 10-8
BSC 026/029 Codes 8-2
BSC Autorecognition 8-13
BSC Binary Codes 8-2

BSC Carriage Control 8-3
BSC Error Processing 8-13
BSC Interactive Carriage Control 8-3
BSC Operational Characteristics 8-5
BSC Operational Features 8-2
BSC TIP 8-1
BSC TIP, Direct Calls from 8-12
BSC TIP, Direct Calls to 8-11
BSC Terminal Features 8-5
BSC Transparent Data 8-3
BSN 6-6
BSN/Block Type 6-6
Buffer
  Availability 4-7
  Copying 4-7
  Formats 4-5, 7-25
  Handling 4-2, 4-7
  Stamping 4-5
  Releasing 4-7
  Releasing Several 7-7
  Single 4-6, 4-7

Call
  BSC TIP 8-11
  Direct 4-9
  Direct and Worklist 1-13
  Firmware Interface 1-15
  Firmware Level 1-15
  Illegal 4-26
  Macroassembly Programs from PASCAL 4-17
  Multiplex Subsystem 1-15
  PASCAL 4-19
Card Reader
  HASP 11-18
  Input Stopped CMD Blocks 10-10
  Interface, Mode 4 10-9
  Nontransparent Data, HASP 11-19
  Stream Control CMD Blocks 11-18
  Transparent HASP Data, HASP 11-20
Carriage Control
  Action, 2780 Batch 8-9
  Action, 3780 Batch 8-11
  BSC 8-3
  BSC Interactive 8-3
  Codes, HASP Printer 11-21
  Codes, Printer 10-11
  DBC Codes for 10-8
  Mode 4 10-8
  Symbols, Batch 8-4
  Symbols, Interactive 8-4
  TTY Output Messages 9-5
CCI
  Design 1-3
  Features 1-8
  Mnemonics B-1
  Modular Structure 1-9
  Modules 1-10
  Modules Relationships With PTLINIT 5-27
  Naming Conventions F-1
  Overview 1-1
  Priority and Nonpriority Tasks 1-4
  Programming Languages 1-18
  Programming Methods 1-9
CDC 711 Terminal Error Processing 10-13
CE Error Messages 3-3, 3-4, 6-21
Change Block, HASP FCS 11-14
Channel, Service 6-8
Characteristics, BSC 8-5
Characters, EBCDIC 11-7
Check if Block is to be Sent, PBBCHCHK 6-35
CLA 5-3
  Status Analyzer 5-22
  Status Overflow Handling 5-24
Classes 12-2
Clear
  Block of Main Memory 4-24
  Line Command 5-11
  Protect Bits 4-25
Cluster Addresses, Mode 4 10-5
CMD Block 6-7
  Card Reader Input Stopped 10-10
  Card Reader Stream Control 11-18
  Printer Data Stream Control 11-20
  Printer Input Stopped 10-11
  Punch Data Stream Control 11-22
  to Host 6-35
Code Conversion, Mode 4 10-3
Codes, Halt 3-2
Command
  Clear Line 5-11
  Control 5-11
  Disable Line 5-20
  Enable Line 5-12
  Host Function 7-10
  Initialize Line 5-11
  Input 5-15
  Input After Output 5-18
  NPU Console Control 4-29
  NPU Function 7-12, 7-16
  Output 5-15
  PPU Function 7-15
  Terminate Input 5-19
  Terminate Output 5-19
Command Block 6-7
  Used on Nonzero Connections 6-9
Command Driver
  Interface 5-9
  Worklist Entries 5-8

Command Format
  Enable Line 5-13
  Input 5-16
  Input After Output 5-18
  Terminate Input 5-20
  Terminate Output 5-21
Command Interface for Printer, HASP 11-21
Command Message Summary C-1
Command Packet Format 5-10
Common
  Multiplex Subroutines for TIPs 5-21
  Return Control Routine, PTRETOPS 6-34
  TIP Regulation, PTREGL 6-34
  TIP Subroutines 6-21, 6-23
Communication Line Adapters (CLA) 5-3
Communication
  Line Initialization, HASP 11-14
  Network 6-1
  Paths for Block Flow Control 6-3
  Using PASCAL Globals 1-16
  Worklist 5-5
Compare
  Equal Length Blocks 4-24
  Two 18-Bit Addresses 4-24
Components
  Hardware 5-3
  State Program 12-4
Compression 8-4
Configuration
  Control Blocks 6-18
  HASP 11-16
  Line 2-5
  Line Deletion 2-10
  Line Service Message 2-6
  Line/Terminal 2-7
  NPU 2-1, 2-4, 2-5
  Sequence, NPU 2-4
  Terminal (TCB) 2-10
  Terminal Service Message 2-11
Connection
  Directory 6-13
  Nonzero 6-9
  Number 6-6
Console
  Blocks, HASP 11-13
  Control Commands 4-29
  Control Messages 4-29
  HASP 11-17
  Support 4-27
  Support Services 4-28
  Worklist Entry 4-29
Contention, Coupler 7-5, 7-17
Contention Resolution, Mode 4 10-9
Control
  Blocks
    Configuring 6-18
    Deleting 6-18
    Disabling 6-18
    Enabling 6-18
    HASP 11-6
  Byte
    Data Blocks, HASP 11-8
    HASP Block 11-8
    HASP Record 11-10
    HASP String 11-12
    HASP Subrecord 11-11
  Codes
    HASP Printer Carriage 11-21
    Printer Carriage 10-11
  Command 5-11
    Format 5-12
    NPU Console 4-29
    BSC Carriage 8-3
    BSC Interactive Carriage 8-3
    Carriage for TTY Output
Messages 9-5
    DBC Codes 10-8
    Data Stream 6-8
    Flow 11-25
    Mode 4 Carriage 10-8
    Routine, PTRETOPS 6-34
    Sequence, HASP Function 11-10
    Single Word Transfers 7-12
    Symbols, Batch Carriage 8-4
    Symbols, Interactive
Carriage 8-4
    TTY Carriage 9-4
Convert ASCII
  Decimal to Binary 4-20
  Hexadecimal to Binary 4-20
Convert Binary
  to ASCII Decimal 4-21
  to ASCII Hexadecimal 4-22
Copying Buffer 4-7
Count, Line 6-20
Coupler
  Contention 7-17
  Function Codes, Programming 7-10
  I/O Transaction Contention 7-5
  I/O Transactions 7-3
  Interface Hardware Programming 7-8
  Interface Protocol Sequences 7-20
  Registers 7-8, 7-9
  Regulation 7-18
  Status Register Bit Assignment 7-11
CRC-16 Error (Cyclic Redundancy Check) 11-23

CRT, Duplicating of Write Data 10-13
Cursor Positioning, Mode 4 10-7
Cyclic Redundancy Check 11-23

Data
  Batch Format 6-10
  Block
    Format Used by the HIP 7-25
    Header Formats 6-4
    HASP 11-8, 11-12
  Formats 6-8
  Interactive Format 6-10
  Nontransparent 6-11
  Mode
    Nontransparent 11-19, 11-20
    Transparent 11-20, 11-21
  Processing, Output 12-7
  Stream Control 6-8
  Structures 1-17, H-1
  Transfer, Multiple Character (Block) 7-16
  Transmission, HASP 11-9
  Transparent 6-12, 8-3
DBC Codes for Carriage Control 10-8
Debugging Aids, On-line I-1
Deletion
  Configure Line 2-10
  Control Blocks 6-18
  TCB 2-12
Design, CCI 1-3
Destination Node Directory 6-13
Device Type, HASP 11-17
Diagnostics 3-1
  Aids, In-line 3-3
  Service Messages 3-5
Direct Calls 1-13, 4-9
  from the BSC TIP 8-12
  from the Mode 4 TIP 10-16
  from the TTY TIP 9-5
  on Firmware Level 1-15
  to TTY TIP 9-4
  to the BSC TIP 8-11
  to the HASP TIP 11-26
  to the Mode 4 TIP 10-16
Directives, HIP 7-2
Directories 6-13
  Alternating 6-16
  Connection 6-13
  Destination Node 6-13
  Routing 6-14
  Source Node 6-13
Disable Line Command - NKDISL 5-20
Disabling Control Blocks 6-18
Discard Non-routable Blocks, PBLOST 6-35
Dispatching Service Messages 6-18
Display File 1 4-25
Downline
  Abort, PBDNABRT 6-35
  Message Processing 1-5, 1-6
Dump
  Interpretation 3-2
  NPU 2-4, 7-14
Duplicating of Write Data on CRT, Mode 4 10-13
Dynamic Page Register 4-23

E-Codes 10-3, 10-6
EBCDIC Characters, HASP 11-7
Elements of Multiplex Subsystem 5-2
Enable Line Command
  Format 5-13
  NKLENBL 5-12
Enabling Control Blocks 6-18
End-of-File Blocks, HASP 11-13
ENQ 11-7
Enquiry Block, HASP 11-7
Entries, Worklist 5-8
Entry
  Console Worklist 4-29
  Worklist 4-12, 4-13
EOF 11-13
  Block, HASP 11-13
EOI 8-2
EOR 8-2
Error
  Block, BCB 11-24
  CRC-16 11-23
  Checking, HIP 7-19
  Conditions, HASP 11-22
  HASP Block Control Byte 11-23
  HASP Unknown Response 11-23
  Handling, Mode 4 10-12
  Illegal Block Make-up 11-23
  Messages 6-21
  Processing
    BSC 8-13
    CDC 711 Terminal 10-13
    HIP 7-7
    Short Term 10-12
    TTY 9-6
Execution Timers 4-27
Execution of State Programs 12-1
Expansion 8-4
Extracting a Worklist Entry 4-13

Failure 3-1
  Host 3-1, 7-18
  Line 3-2
  NPU 3-1
  Terminal 3-3
FCS 11-10
  Change Block 11-13, 11-14

Features
BSC 8-2
BSC Terminal 8-5
CCI 1-8
HASP Workstation 11-2
File 1, Load/Display 4-25
Files, Punch 8-3
Finding Number of Characters to be Processed 6-33
Firmware
Interface 1-15
Interface to Modem State Programs 12-9
Interface to Output Data Processing 12-7
Level, Direct Calls 1-15
Multiplex Level 1 5-4
Text Processing 6-33
Worklist Entries 5-8
Flags, Set Accept Input/Accept Output 6-35
Flow Control 6-3
HASP 11-25
Format
Batch Data 6-10
Buffer 4-5, 7-25
CE Error and Statistics Messages 3-3
Command Packet 5-10
Control Command 5-12
Data 6-8
Data Block 7-25
Data Block Header 6-4
Enable Line Command 5-13
HASP Signon Block 11-15
Host/NPU Word 7-7
Input After Output Command 5-18
Input Command 5-16
Interactive Data 6-10
Terminate Input Command 5-20
Terminate Output Command 5-21
Function 12-4
Block 4-24
Codes, Coupler 7-10
Commands, NPU 7-16
Commands, PPU 7-15
Control Sequence, HASP 11-10
HASP TIP 11-3
HIP 7-12
Host 7-10
Mode 4 TIP 10-1
Modem/Circuit 5-14
NPU 7-12
Transfer 7-1
18-Bit Address 4-24

General Peripheral Processing 4-27

Generate
Banner Records, PTBANLACE 6-35
Lace Records, PTBANLACE 6-35
Service Messages 6-18
Statistics Service Messages 6-20
Status Service Messages 6-19
Globals, PASCAL 1-16, 4-17
Glossary A-1

Halt Codes 3-2
Handling
Buffer 4-2, 4-7
CLA Status Overflow 5-24
Line Interface 1-16
Modem Response Timeout 5-25
Routines 4-19
Hardware
Components 5-3
Considerations, Mode 4 10-1
Considerations, HASP 11-1
Programming, Coupler Interface 7-8
HASP
Acknowledgment Block (ACK) 11-6
Addressing 11-16
Autorecognition 11-25
Block Control Byte (BCB) 11-8, 11-23
Card Reader 11-18
Card Reader Nontransparent Data Mode 11-19
Card Reader Transparent Data Mode 11-20
Command Interface for Printer 11-21
Communication Line Initialization 11-14
Configuration 11-16
Console 11-17
Control Blocks 11-6
Control Bytes for Data Blocks 11-8
Data Block Description 11-12
Device Type 11-17
EOF Block 11-13
End-of-File Blocks (EOF) 11-13
Enquiry Block (ENQ) 11-7
Error Conditions 11-22
FCS Change Block 11-13, 11-14
Flow Control 11-25
Function Control Sequence (FCS) 11-10
Hardware Considerations 11-1
Host Interface 11-16
Idle Block (ACK0) 11-8
Illegal Block Make-up Error 11-23

Multileaving Block Description 11-6, 11-9
Negative Acknowledgment Block (NAK) 11-7
Operator Console Blocks 11-13
Postprint 11-27
Printer 11-20
Carriage Control Codes 11-21
Nontransparent Data Mode 11-20
Transparent Data Mode 11-21
Protocol 11-4, 11-5
Punch 11-22
Record Control Byte (RCB) 11-10
Regulation 11-25
Significant EBCDIC Characters 11-7
Signoff Block 11-16
Signon Block 11-15
Signon Block Format 11-15
String Control Byte (SCB) 11-12
Subrecord Control Byte (SRCB) 11-11
TIP 11-1
TIP Functions 11-3
TIP, Direct Calls to 11-26
Terminal Operational Procedure 11-6
Unknown Response Error 11-23
User Interface 11-14
Workstation
Features 11-2
Initialization 11-14
Startup 11-14
Termination 11-14
Header Formats, Data Block 6-4
HIP 7-1
Data Block Format 7-25
Directives 7-2
Error Processing 7-7, 7-19
Functions 7-12
OPS and Interrupt Levels 7-6
States 7-25, 7-27
Timeouts 7-19
Transfer Initiation 7-2
Transfer Timing 7-7
Transitions 7-27
Host
Failure 3-1, 7-18
Function Commands 7-10
Interface Package (HIP) 7-1
Interface Protocol Sequence
NPU Side 7-21
Host Side 7-23
Interface
HASP 11-16
Mode 4 10-7
NPU Word Formats 7-7
Recovery 7-18
Send CMD Block to 6-35
I/O Transactions
Contention at the Coupler 7-5
Coupler 7-3
Idle Block, HASP 11-8
Illegal Block Make-up Error, HASP 11-23
Illegal Calls 4-26
In-line Diagnostic Aids 3-3
Indicators, Mode 4 Message Type 10-3
Initialization
HASP Communication Line 11-14
HASP Workstation 11-14
Line 5-26
NPU 2-1
Phase I 2-1
Phase II 2-2
Initialize Line Command 5-11
Initiating HIP Transfer 7-2
Inline Diagnostic Service Messages 3-5
Input After Output Command - NKINOUT 5-18
Input Command - NKINPT 5-15, 5-16
Input Nontransparent Terminal Mode
2780 8-5
3780 8-7
Input Regulation, Mode 4 10-13
Input State Programs 12-5
Input State/Modem State Programs Interface 12-10
Input Stopped, Card Reader 10-10
Input Transparent
Data Mode, 2780 and 3780 8-7
Terminal Mode, 2780 8-6
Terminal Mode, 3780 8-7
Input, Accept 6-35
Interactive Carriage Control
BSC 8-3
Symbols 8-4
Interactive
Format Data 6-10
Interface, Mode 4 10-7
Mode TIP State Transitions 9-2
Mode, TTY 9-1
Interface
Command Driver 5-9
Coupler 7-8, 7-20
Firmware 12-7, 12-9
HASP Command for Printer 11-21
HASP User 11-14
HASP/Host 11-16
Input State/Modem State Programs 12-10
Line 1-16
Mode 4
Card Reader 10-9
Printer 10-9
Terminal 10-3

PTCLAS/Modem State Programs 12-10
Package, Host 7-1
Priority Processing 1-3
Protocol Sequence
Host/NPU 7-21
NPU/Host 7-23
Special Call to Firmware 1-15
System and User 5-4
Text Processing Firmware 6-33
User 4-15, 5-9
Internal Output POI 6-22
Internal Service Message Processing 6-17
Interpretation, Dump 3-2
Interrupt
Assignments 4-16
Levels for the HIP 7-6
Mask
AND 4-14
OR 4-14
Set 4-14
Priority 4-14
Processing 4-13
State Definitions (PBINTRAPS) 4-15

Languages, CCI Programming 1-18
LCBs, Saving and Restoring 6-33
Level 1, Multiplex 5-4
Level 2
Multiplex 5-7
Worklists 5-6
Levels, OPS and Interrupt 7-6
Line
Clear Command 5-11
Configuration 2-5, 2-6, 2-7, 2-10
Count Request Service Message 6-20
Enable Command 5-12, 5-13
Failure 3-2
Initialization, HASP 11-14
Initialize Command 5-11
Initializer 5-26
Interface Handling 1-16
Recovery 3-2
Status Request Service Message 6-19
Link, Logical 6-34
LIP/TIP OPS level Worklists 5-7
Load
File 1 4-25
NPU 2-4, 7-14
User-Defined Message 4-26
Locating a State Process 12-3
Logical Link Regulation 6-34
Long Term Recovery, Mode 4 10-13
Loop Multiplexers 5-3

Macroassembly Programs 4-17
Macroinstructions 12-10
State Program 12-11
Macrointerrupts 4-13
Main Memory
Clear 4-24
Map for NPU E-1
Maintaining Paging Registers 4-22
Making a Worklist Entry 4-12
Mask
AND Interrupt 4-14
OR Interrupt 4-14
Set Interrupt 4-14
Maximum, Two Numbers 4-20
Memory Map for NPU E-1
Message
Block, MSG 6-7
CE Error 3-4, 6-21
Command C-1
Console Control 4-29
Downline Processing 1-5
Formats
CE Error and Statistics 3-3
Mode 4 Protocol 10-4
Output 9-4
Processing
Downline 1-6
Upline 1-5, 1-7
Segment 6-32
Service 6-16, C-1
Statistics 3-5
Type Indicators, Mode 4 10-3
User-Defined 4-26
Methods, CCI Programming 1-9
Microinterrupts 4-16
Minimum, Two Numbers 4-21
Miscellaneous Subroutines 4-25
MLIA 5-3
Mnemonics
CCI B-1
HASP Protocol 11-5
Mode 4
Autorecognition 10-14
Card Reader Interface 10-9
Carriage Control 10-8
Code Conversion 10-3
Contention Resolution 10-9
Cursor Positioning 10-7
Duplicating of Write Data on CRT 10-13
Error Handling 10-12
Hardware Considerations 10-1
Host Interface 10-7
Input Regulation 10-13
Interactive Interface 10-7
Long Term Recovery 10-13
MTI Codes 10-5
Message Type Indicators 10-3
Nomenclature 10-3

Printer Interface 10-9
Protocol Features Not Supported 10-16
Protocol Message Formats 10-4
Short Term Error Processing 10-12
TIP 10-1
TIP, Direct Calls 10-16
Terminal Addressing 10-3
Terminal Interface 10-3
Terminal/Cluster Addresses 10-5
Upline Breaks 10-8
Modem Response Timeout Handling 5-25
Modem State Programs 12-8
Firmware Interface 12-9
PTCLAS Interface 12-10
Modem State/Input State Programs Interface 12-10
Modem/Circuit Functions 5-14
Modular Structure, CCI 1-9
Modules, CCI 1-10
Monitor Table 4-3, 4-4
Monitor, System 4-1
MSG (Message) Block 6-7
MTI Codes for Mode 4 10-5
Multileaving Transmission Block, HASP 11-6, 11-9
Multiple Character Data Transfer 7-16
Multiplex Subsystem 5-1, 5-2
Firmware Worklist Entries 5-8
Level 1 - Firmware 5-4
Level 2 - PMWOLP 5-7
Level 2 Worklists 5-6
Level Status Handler Interface 12-10
Loop Interface Adapter 5-3
Special Call 1-15
Subroutines for TIPs, Common 5-21
Worklist Communications 5-5
Worklist Processor 5-21
Multiplexers, Loop 5-3

NAK 11-7
Naming Conventions, CCI F-1
Negative Acknowledgment Block, HASP 11-7
Network 1-2
Communication Software 6-1
NKDISL 5-20
NKDOUT 5-15
NKENDIN 5-19
NKENDOUT 5-19
NKINOUT 5-18
NKINPT 5-15
NKLENBL 5-12
Node 6-2
Destination 6-13
Source 6-13
Nomenclature, Mode 4 10-3
Non-routable Blocks 6-35
Nonpriority Tasks in CCI 1-4
Nontransparent Data Mode 6-11
HASP Card Reader 11-19
HASP Printer 11-20
Nontransparent Terminal Mode
2780 8-5
3780 8-7
Nontransparent Transmission Mode
2780 Output 8-8
3780 Output 8-10
Nonzero Connection Command Blocks 6-9
NPU
Configuring 2-4 2-5
Console Control Commands 4-29
Failure 3-1
Function Commands 7-12, 7-16
Host Word Formats 7-7
in a Network 1-2
Initializing and Configuring 2-1
Load and Dump 2-4, 7-14
Memory Map E-1
Recovery 3-2
Side, Host Interface Protocol Sequence 7-21
Status Word Codes 7-13
Stop 4-26
Number of Characters to be Processed 6-33
Numbers 4-20, 4-21

Obtaining a Single Buffer 4-6
ODD Handler 5-29
On-line Debugging Aids I-1
Operating Modes, TTY 9-1
Operational Characteristics, BSC 8-5
Operational Procedure, HASP Terminal 11-6
Operator Console Blocks, HASP 11-13
OPS Level 5-8
HIP 7-6
Processing 1-4
Worklists, TIP/LIP 5-7
OPS Monitor Table 4-3, 4-4
Optional Modem/Circuit Functions 5-14
OR Interrupt Mask 4-14
Orderword Register Codes 7-13
Organization, Worklist 4-11
Output

Accept 6-35
Command - NKDOUT 5-15
Data Demand Timing Handler 5-29
Data Processing, Firmware Interface 12-7
Messages, TTY Carriage Control 9-4, 9-5
Nontransparent Transmission Mode
2780 8-8
3780 8-10
Queuing (PBQ1BLK and PBQBLKS) 6-30
Transparent Transmission Mode, 2780 8-10
Overflow Handling, CLA Status 5-24
Overview, CCI 1-1

Page Mode 4-22
Page Register 4-22, 4-23
Page Switching 4-22
PASCAL
Globals 1-16, 4-17
Procedure Calls, Type-Checking 4-19
Programs 4-17
Paths, Communications 6-3
PB18ADD - 18-Bit Addresses 4-23
PB18BITS - 18-Bit Address Functions 4-24
PB18COMP - Compares Two 18-Bit Addresses 4-24
PBAEXIT, Restore R1 and R2 6-33
PBAMASK - AND Interrupt Mask 4-14
PBBCHCHK 6-35
PBBEXIT, Save R1 and R2 6-32
PBCLR - Clears a Block of Main Memory 4-24
PBCLRPOT - Clear Protect Bit 4-25
PBCOMP - Compares Equal Length Blocks 4-24
PBDNABRT 6-35
PBFILE1 - Load/Display File 1 4-25
PBFMAD - Converts ASCII Decimal to Binary 4-20
PBFMAH - Converts ASCII Hex to Binary 4-20
PBGETPAGE - Reads Specified Page Register 4-23
PBGT1SET 6-32
PBHALT - Stops the NPU 4-26
PBILL - Illegal Calls 4-26
PBINTRAPS 4-15
PBIOPOI - Internal Output POI 6-22
PBLMASK 4-14
PBLOAD - Load a User-Defined Message 4-26
PBLOST 6-35
PBMAX - Finds the Maximum of Two Numbers 4-20
PBMEMBER - Test ASCII Set Membership 4-20
PBMIN - Finds the Minimum of Two Numbers 4-21
PBOMASK - OR Interrupt Mask 4-14
PBPIPOI - Post Input POI 6-22
PBPOPOI - Post Output POI 6-30
PBPROPOI - Preoutput POI 6-30
PBPSWITCH - Performs Page Switching 4-22
PBPUTPAGE - Write Specified Page Register 4-23
PBQ1BLK 6-30
PBQBLKS 6-30
PBRDPGE - Reads Dynamic Page Register 4-23
PBRTEIA 6-35
PBRTEPRU 6-35
PBSETPROT - Set Protect Bit 4-25
PBSMASK - Set Interrupt Mask 4-14
PBSTPMODE - Sets Paging Mode 4-22
PBSWITCH 6-15
PBTOAD - Converts Binary to ASCII Decimal 4-21
PBTOAH - Converts Binary to ASCII Hex 4-22
PBUPABRT 6-35
Perform Page Switching 4-22
Peripheral Processing 4-27
Phase I Initialization 2-1
Phase II Initialization 2-2
PIAPPS 2-3
PIBUF1 2-2
PIBUF2 2-3
PIINIT 2-3
PILININIT 2-3
PIMLIA 2-3
PINIT 2-2
PIPROTECT 2-2
PIWLINIT 2-3
PLINIT - Line Initializer 5-26
PMT1SEC - Output Data Demand Timing Handler 5-29
PMWOLP 5-7
PMWOLP - Multiplex Worklist Processor 5-21
POI Programs 1-13, 6-21
Internal Output 6-22
Post Input 6-22
Post Output 6-30
Preoutput 6-30
Point of Interface Programs 1-13, 6-21
Positioning Cursor 10-7
Post Input POI 6-22
Post Output POI 6-30

Postprint, HASP 11-27
PPU Function Commands 7-15
Preoutput POI 6-30
Principal Data Structures 1-17, H-1
Printer
  Carriage Control Codes 10-11
    HASP 11-21
  Data Stream Control CMD Blocks 11-20
  HASP 11-20
  HASP Command Interface 11-21
  Input Stopped CMD Blocks 10-11
  Interface, Mode 4 10-9
  Nontransparent Data Mode, HASP 11-20
  Transparent Data Mode, HASP 11-21
Priority
  Interrupt 4-14
  Processing at the Interfaces 1-3
  Tasks in CCI 1-4
Procedure Calls, PASCAL 4-19
Procedure, HASP Terminal 11-6
Processing
  BSC Error 8-13
  Basic Interrupt 4-13
  CDC 711 Terminal Error 10-13
  Downline Message 1-5, 1-6
  General Peripheral 4-27
  HIP Error 7-7
  Mode 4 Short Term Error 10-12
  Number of Characters 6-33
  OPS-Level 1-4
  Routing 6-13
  Priority 1-3
  Service Message 6-17
  TTY Error 9-6
  Upline Message 1-5, 1-7
Program Execution Timers 4-27
Programming Coupler
  By Use of Function Codes 7-10
  Interface Hardware 7-8
Programming Languages, CCI 1-18
Programming Methods 1-9
Programs
  Macroassembly 4-17
  PASCAL 4-17
  Point of Interface 1-13
  State 12-1
  Support 1-11
Protect Bit
  Clear 4-25
  Set 4-25
Protocol
  Block 1-9, 6-1, 11-4, D-1
  Features Not Supported, Mode 4 10-16
  Mnemonic Definitions, HASP 11-5
  Mode 4 10-4
  Sequence, Host Interface 7-21, 7-23
  Sequence, Coupler Interface 7-20
  Transaction 7-1
PRU Block Routing, PBRTEIA 6-35
PTBANLACE 6-35
PTCLAS - CLA Status Analyzer 5-22
  Interface to Modem State Programs 12-10
  Worklist 5-24
PTCOMMAND 6-35
PTCTCHR 6-33
PTINIT 6-35
  Relationships With CCI Modules 5-27
  State Transition Table 5-28
PTREGL 6-34
PTRETOPS 6-34
PTRTxLCB 6-33
PTSVxLCB 6-33
PTTPINF 6-33
Punch Data Stream Control CMD Block 11-22
Punch Files 8-3
Punch, HASP 11-22

Queuing
  Output 6-30
  Removing a Message Segment from 6-32
  TCB 6-31

RBF 8-2
RCB 11-10
Read
  Dynamic Page Register 4-23
  Specified Page Register 4-23
Reconfiguration, TCB 2-12
Record Control Byte, HASP 11-10
Records, Banner and Lace 6-35
Recovery 3-1
  Host 7-18
  Line 3-2
  Mode 4 10-13
  NPU 3-2
  Terminal 3-3
Register
  Address 7-14
  Coupler 7-9
  Orderword 7-13
  Paging 4-22, 4-23
  Saving and Restoring 6-32
  Use, Coupler 7-8
Regulation
  Coupler Use 7-18
  HASP 11-25

Logical Link 6-34
Mode 4 Input 10-13
TIP 6-34
Relationships, PTLINIT/CCI
Modules 5-27
Releasing
Buffer 4-7, 4-7
Several Buffers 7-7
Remote Batch Facilities (RBF) 8-2
Removing a Message Segment from
Queue 6-32
Restoring
LCBs 6-33
Registers 6-32
R1 and R2 6-33
Return Control Routine, PTRETOPS
6-34
Routines 4-8, 4-19
Point of Interface 6-21
Routing 6-12
Block 1-13
Directories 6-14
Flow Chart, PBSWITCH 6-15
PRU Block 6-35
Process 6-13
Upline PRU Block 6-35

Saving
LCBs 6-33
Registers 6-32
R1 and R2 6-32
SCB 11-12
Sending
CMD Block to Host, PTCOMMAND
6-35
Statistics Service Messages
6-20
Status Service Messages 6-19
Service Channel 6-8
Service Message 6-16
Configure Line 2-6
Configure Terminal 2-11
Dispatching 6-18
Generating 6-18
Inline Diagnostic 3-5
Internal Processing 6-17
Line Count Request 6-20
Line Status Request 6-19
Statistics 6-20
Status 6-19
Summary C-1
Terminal Status Request 6-20
Timing Out 6-17
Validating 6-17
Services
Console Support 4-28
Timing 4-8
Worklist 4-10
Set
Interrupt Mask 4-14
Logical Link Regulation,
LNLLREG 6-34
Membership 4-20
Paging Mode 4-22
Protect Bits 4-25
Short Term Error Processing, Mode
4 10-12
Signoff Block, HASP 11-16
Signon Block, HASP 11-15
Single Word Transfers Control
7-12
Software, Base System 4-1
Source Node Directory 6-13
Special Call
to Firmware Interface 1-15
to Multiplex Subsystem 1-15
SRCB 11-11
Stamping, Buffer 4-5
Standard
Subroutines 4-17, 4-18
TIP Subroutines 6-30
TIP Trees G-1
Startup, HASP Workstation 11-14
State Process 12-3
State Program 12-1
Components 12-4
Execution 12-1
Input 12-5
Interface 12-10
Macroinstructions 12-11
Modem 12-8, 12-9
Text Processing 12-6
State Transitions
Table, PTLINIT 5-28
TIP Interactive Mode 9-2
TIP Tape Mode 9-3
States, HIP 7-25, 7-27
Statistics Messages 3-3, 3-5,
6-20
Status
Analyzer, CLA 5-22
Handler, Multiplex Level
Interface 12-10
Line 6-19
Overflow Handling, CLA 5-24
Register, Coupler 7-11
Service Messages 6-19
Terminal 6-20
Word, NPU 7-13
Stop NPU 4-26
Stream Control CMD Block
Card Reader 11-18
Printer Data 11-20
Punch Data 11-22
String Control Byte, HASP 11-12
Structures, Data 1-17

Subrecord Control Byte, HASP 11-11
Subroutines
  Common TIP 6-21
  for TIPs, Common Multiplex 5-21
  Miscellaneous 4-25
  Standard 4-17, 4-18
  Standard TIP 6-30
Subsystem, Multiplex 5-1
Support Programs for TIPs 1-11
Support Services, Console 4-27, 4-28
SVM Trees G-1
Switching, Page 4-22
System Interfaces 5-4
System Monitor 4-1

Table
  OPS Monitor 4-3, 4-4
  PTLINIT State Transition 5-28
Tape Mode
  TIP State Transitions 9-3
  TTY 9-2
Tasks, Priority and Nonpriority 1-4
TCB
  Configuration 2-10
  Deletion 2-12
  Queue 6-31
  Reconfiguration 2-12
Terminal
  Addressing, Mode 4 10-3, 10-5
  Configuration 2-7, 2-10, 2-11
  Error Processing, CDC 711 10-13
  Failure 3-3
  Features, BSC 8-5
  Interface, Mode 4 10-3
  Mode, 2780 8-5, 8-6
  Mode, 3780 8-7
  Operational Procedure, HASP 11-6
  Recovery 3-3
  Status Request Service Message 6-20
Terminate Input Command - NKENDIN 5-19, 5-20
Terminate Output Command - NKENDOUT 5-19, 5-21
Termination, HASP Workstation 11-14
Test
  ASCII Set Membership 4-20
  Buffer Availability 4-7
Text Processing
  Firmware Interface 6-33
  State Programs 12-6
Timeout
  Handling, Modem Response 5-25
  HIP 7-19
  Output Data Demand 5-29
  Service Messages 6-17
Timers, Program Execution 4-27
Timing Services 4-8
Timing, HIP Transfer 7-7
TIP
  BSC 8-1
  Common Multiplex Subroutines 5-21
  HASP 11-1, 11-3
  LIP OPS level Worklists 5-7
  Mode 4 10-1
  Regulation, PTREGL 6-34
  State Transitions
    Interactive Mode 9-2
    Tape Mode 9-3
  Subroutines 6-21, 6-23, 6-30
  Support Programs 1-11
  TTY 9-1, 9-4, 9-5
  Worklist Communications 5-5
Transaction
  Contention at the Coupler 7-5
  Protocol 7-1
Transactions, Coupler I/O 7-3
Transfer
  Block 7-16
  Functions 7-1
  Initiation, HIP 7-2
  Single Word 7-12
  Timing, HIP 7-7
Transitions, HIP 7-27
Transmission Mode
  2780 Output Nontransparent 8-8
  2780 Output Transparent 8-10
  3780 Output Nontransparent 8-10
Transparent Data Mode 6-12
  BSC 8-3
  HASP Card Reader 11-20
  HASP Printer 11-21
  2780 and 3780 8-7
Transparent Terminal Mode
  2780 8-6
  3780 8-7
Transparent Transmission Mode, 2780 Output 8-10
TTY
  Autorecognition 9-6
  Carriage Control for Output Messages 9-4
  Error Processing 9-6
  Interactive Mode 9-1
  Operating Modes 9-1
  Output Messages Carriage Control 9-5
  TIP 9-1
  TIP Direct Calls 9-4, 9-5
  Tape Mode 9-2
Type, BSN/Block 6-6

Type Checking in PASCAL Procedure Calls 4-19
Types, Block 6-6, 6-7

Unknown Response Error, HASP 11-23
Upline
  Abort, PBUPABRT 6-35
  Breaks, Mode 4 10-8
  Message Processing 1-5, 1-7
  PRU Block Routing, PBRTEPRU 6-35
User Interface 4-15, 5-4, 5-9
  HASP 11-14
User Defined Message 4-26

Validating Service Messages 6-17

Word Formats, Host/NPU 7-7
Word Transfers Control 7-12
Worklist
  Calls 1-13
  Communications, TIP and Mux Subsystem 5-5
  Entries
    Command Driver 5-8
    Console 4-29
    Extracting a 4-13
    Making 4-12
    Multiplex Subsystem Firmware 5-8
  Multiplex Level 2 5-6
  Organization 4-11
  PTCLAS 5-24
  Processor, Multiplex 5-21
  Services 4-10
  TIP/LIP OPS level 5-7
Workstation, HASP 11-2, 11-14
  Initialization 11-14
Write Data, Duplicating on CRT 10-13
Write Specified Page Register 4-23

026/029 Codes, BSC 8-2
18-Bit Address Functions 4-24
18-Bit Addresses 4-23
18-Bit Addresses, Compare 4-24
2780 Batch Carriage Control Action 8-9
2780 Input Nontransparent Terminal Mode 8-5
2780 Input Transparent Data Mode 8-7
2780 Input Transparent Terminal Mode 8-6
2780 Output Nontransparent Transmission Mode 8-8
2780 Output Transparent Transmission Mode 8-10
3780 Batch Carriage Control Action 8-11
3780 Input Nontransparent Terminal Mode 8-7
3780 Input Transparent Data Mode 8-7
3780 Input Transparent Terminal Mode 8-7
3780 Output Nontransparent Transmission Mode 8-10

COMMENT SHEET

MANUAL TITLE CCI, VERSION 3 SYSTEM PROGRAMMERS REFERENCE MANUAL

PUBLICATION NO. 60471160 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

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