60471150

CONTROL DATA
CORPORATION

COMMUNICATIONS CONTROL INTERCOM VERSION 3 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	Initial release at PSR level 473.
(6/13/78)	
B	Cycle 42. Corrective code release. Includes all revisions through PSR level 488.
(2/1/79)	
C	Added accounting package feature and PSRs to level 518.
(5/16/80)	

Publication No.
60471150

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

Address comments concerning this manual to:

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

or use Comment Sheet in the back of this manual.

LIST OF EFFECTIVE PAGES

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

PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV
Cover	-						
Title Page	-						
ii thru iv	C						
v/vi	B						
vii/viii	C						
1-1/1-2	A						
1-3	B						
1-4/1-5	A						
1-6 thru 1-14	C						
1-15	A						
1-16 thru 1-18	C						
2-1/2-2	A						
2-3	B						
2-4 thru 2-9	A						
2-10	B						
2-11 thru 2-18	A						
3-1 thru 3-3	B						
4-1	A						
5-1 thru 5-4	A						
A-1 thru A-6	A						
B-1/B-2	C						
B-3 thru B-6	B						
B-7	C						
C-1 thru C-7	C						
D-1 thru D-3	A						
E-1	A						
F-1	A						
G-1	A						
H-1/H-2	A						
H-3	C						
H-4	A						
H-5 thru H-19	C						
Index-1 thru Index-3	C						
Comment Sheet	-						
Mailer	-						
Back Cover	-						

PREFACE

This manual provides overview information for CONTROL DATA® Communications Control INTERCOM (CCI) version 3.0 and describes the functions that CCI provides for INTERCOM 5. INTERCOM 5 operates under control of the NOS/BE 1.3 operating system for the CDC® CYBER 170 Series; CYBER 70 Models 71, 72, 73, and 74; and 6000 Series Computer Systems. CCI provides the operating system for the CDC® 255x Series of Network Processor Units, called Host Communications Processors.

This manual is a broad reference guide for administrators or system analysts at sites using CCI without modification or a need for in-depth software maintenance information. It is recommended that the reader be familiar with the PASCAL programming language, INTERCOM 5, the NOS/BE operating system, and the CYBER Cross System. Programming experience with data communications software and hardware is assumed.

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, acronyms, keywords, 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 CCI 3 programmer or terminal user, or by the network software as output.

... — The ellipsis indicates omitted entities that 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.

○ — A circle around a terminal key identifier indicates that the key is pressed in conjunction with a control key.

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 7-bit ASCII coded characters.

RELATED MANUALS

Additional information on CDC network processing units, the CCI, and related software can be found in the following documents.

Publication	Publication Number
INTERCOM 5 Reference Manual	60455010
NOS/BE 1 Operator's Guide	60493900
NOS/BE 1 Reference Manual	60493800
CYBER Cross System Version 1 Reference Manual	96836000
CYBER Cross System Version 1 PASCAL Reference Manual	96836100
CYBER Cross System Version 1 Macro Assembler Reference Manual	96836500
CYBER Cross System Version 1 Micro Assembler Reference Manual	96836400
CYBER Cross System Version 1 Link Editor and Library Maintenance Programs Reference Manual	60471200
Network Processor Unit (NPU) Hardware Reference Manual	60472800
Mode 4C Data Communications Control Procedure System Standard	CDC STD 1.10.020

2550-2 (MOS) Host Communications Processor Hardware Reference Manual	74375500
State Programming Language Reference Manual	60472200
MSMP Diagnostics Reference Manual	96700000
ODS Version 2 Reference Manual	96768410

CDC manuals can be ordered from Control Data 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

APPENDIXES

INDEX

FIGURES

TABLES

INTRODUCTION TO CCI AND INTERCOM 5 CONCEPTS 1

Communications Control INTERCOM (CCI) provides the software necessary to process data (messages) through the communication portion of a Control Data network. The network communication function allows the INTERCOM 5 program in the main computer (called the host computer in a network) to process data as if the program communicated directly with an interactive or batch terminal through a host port. Because terminals can be of only two generalized data types, interactive or batch, the host processing becomes essentially independent of terminal communication protocols.

Minimizing terminal type dependency and removing many of the terminal switching operations from the host computer frees the host to process data more efficiently. As a result of this division of labor, the host can accommodate many more terminals and terminal types. The host also processes INTERCOM 5 functions more rapidly and with greater flexibility, because it is not burdened with input/output functions that are better performed elsewhere.

The network communication function thus removed from the host computer is made resident in the 255x Series Network Processing Unit (NPU). The NPU is a minicomputer system with associated multiplexing and coupling hardware. The following are performed by the NPU:

- Multiplexing data to or from the numerous terminals.
- Demultiplexing data and storing it in buffers for buffered high-speed transfers to or from the host computer.
- Converting numerous terminal protocols into either an interactive or batch data format. The converse operation is performed for output operations to terminals.
- Regulating the volume of traffic handled, as required.

Communications network processing not only relieves the host computer of most input/output overhead; it also relieves INTERCOM 5 of the need to concern itself with terminal characteristics other than the characteristics of the interactive or batch terminal interface.

Since it is necessary to know the basic concepts of network message processing to understand the structure of the CCI software, this introduction includes a description of the network and the distribution of network tasks between the host and the NPUs.

The CCI is functionally divided into these software groups:

- Base system software which includes the NPU operating system monitor, timing and interrupt services, multiplex subsystem, initialization, space allocation, and other general service routines
- Network communications software that performs routing, control and status processing, and hardware configuration control
- Interface programs for the host interface and the standard terminal interfaces

NETWORK CONCEPTS

Network software provides effective access to data-processing services for terminal users. These services consist primarily of application programs written to perform specific functions. The applications programs are executed in the host computer through the interface provided by INTERCOM 5.

Network software is designed to achieve functional separation between the host system services to terminal users, and the communications equipment software required to interconnect the host computer and its terminals. This has led to the concept of viewing the complete network as consisting of two separate networks, a communications network and a computer network, with well-defined hardware and software boundaries between them (figure 1-1).

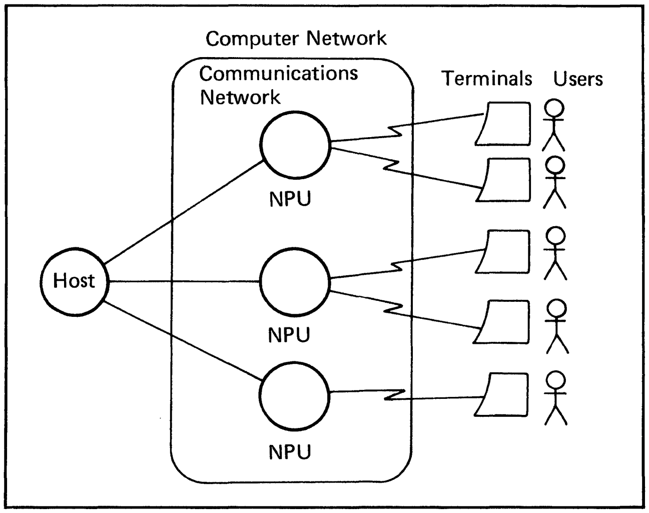


Figure 1-1. CDC Network, Overview of Functions

COMPUTER NETWORK OVERVIEW

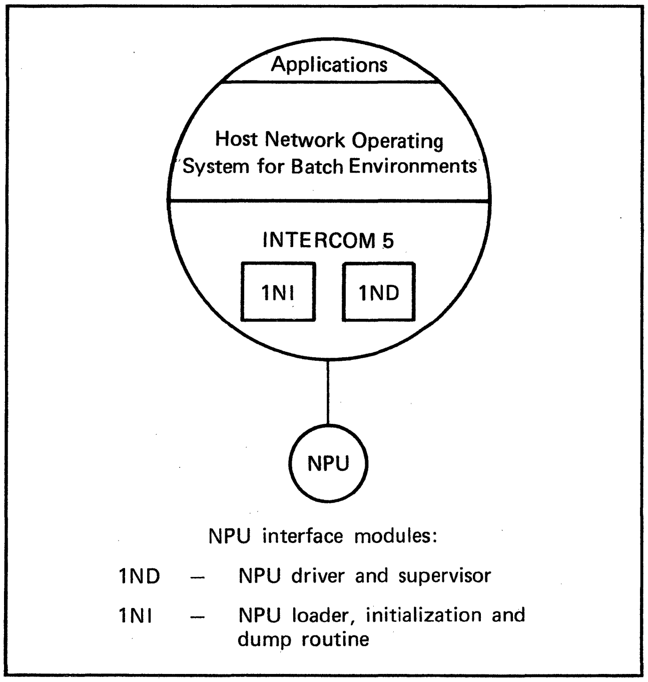
The computer network includes host computers and terminals; the INTERCOM 5 program provides services to the terminal users. The computer network uses the communications network and, in some cases, terminal software to interface between terminal users and INTERCOM 5 in a host computer system.

Host Software

The Network Operating System for Batch Environments (NOS/BE) provides the operational environment and control for the computer network software. INTERCOM 5 provides a standard interface between the communications network and the application programs executing in the host. The INTERCOM 5 PPU programs 1NI and 1ND are responsible for the network coordination and control-oriented activities of the host computer. Figure 1-2 shows this software.

As various events occur in the network, service messages are passed to INTERCOM 5. They may, for example, inform INTERCOM of a new logical connection for a terminal which desires services from it, or of the fact that some failure has occurred. In the same way, INTERCOM uses service messages to communicate with CCI. For example, INTERCOM can disassociate itself from a logically connected terminal.

This use of service messages between CCI and INTERCOM 5 eliminates the need for a defined table structure in the INTERCOM 5 field length. CCI allows INTERCOM to use the table structure that is most efficient for it. Additionally, CCI does not limit the kind of buffering used by INTERCOM. INTERCOM can provide a buffer for each logical connection, or it can perform all its input/output from a single buffer. This allows INTERCOM 5 maximum flexibility of design.



NPU interface modules:

1ND — NPU driver and supervisor

1NI — NPU loader, initialization and dump routine

Figure 1-2. Host Software

Network Definition

The CDC network software defines the complete network:

- The communications network is defined in terms of hosts, nodes, and the physical and logical links between them.
- The computer network is defined in terms of lines and terminals.

Network definition occurs when CCI is built and added to the NOS/BE system library.

Network Supervision

The INTERCOM 5 PPU program 1NI coordinates the activities of the various network processing units (NPUs) in the communications network. The functions of 1NI, as shown in figure 1-3, are as follows:

- 1NI loads software into the NPUs.
- 1NI superimposes a logical network structure on the physical structure of the communications network. Predefined logical links are established between the host and each NPU with which it is allowed to communicate. By this means, the functions of the computer network and the communications network are effectively separated.
- 1NI also receives reports from the NPUs on the status of the network and takes corrective action as required.

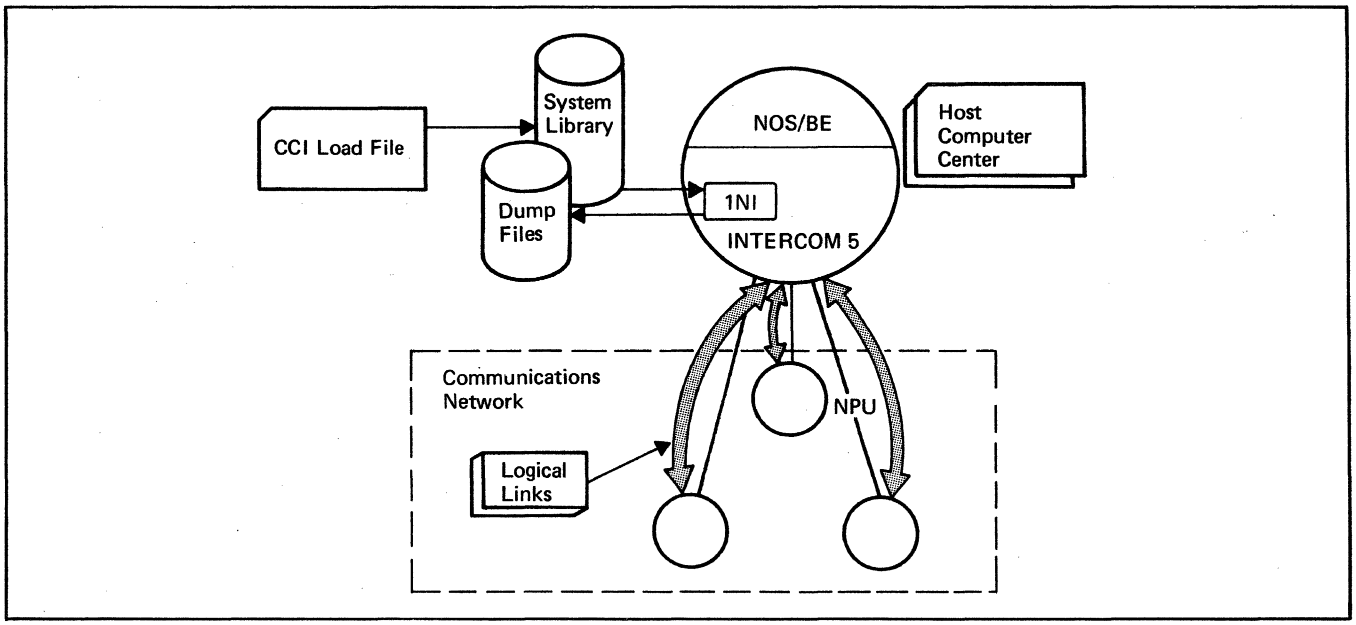


Figure 1-3. Network Supervision Functions

The INTERCOM 5 PPU program 1ND coordinates the network-oriented activities of the host computer. The primary function of 1ND is to isolate the functioning of INTERCOM 5 from the physical configuration of the terminals in the network. As terminals connect to the network, 1ND establishes logical connections between each terminal and INTERCOM 5 as shown in figure 1-4. INTERCOM 5 communicates with terminals using simple connection numbers, regardless of the location of the terminals in the network.

COMMUNICATIONS NETWORK OVERVIEW

The communications network includes a set of network processor units (NPUs). Its purpose is to transport blocks of data between the host computer and terminals. To perform this function, the NPU presents an interface (represented by a set of protocols) to the host computer on one side and to each terminal on the other side. At the host interface, messages are transferred in buffers of data at channel speeds. At the terminal interface, messages are transferred at single character speeds.

Apart from specific channel input/output procedure, the host interface is independent of the host computer architecture. This interface is insensitive to the detailed topology of the communications network, so either a simple network with a single front-end NPU or a complex network with many NPUs can be supported with the same set of host computer software.

Communications Network Hardware and Software

The communications network allows terminals to access the host computer via communications lines. The following network hardware and software are used to implement host computer access:

- The 255x Series Network Processor Unit (NPU) hardware which provides for physical connection between host and terminal.
- Communications Control INTERCOM (CCI) which is the software system in the 255x series NPU.
- The CYBER Cross System software which supports the installation, maintenance, and modification of CCI via the host computer. The CYBER Cross System is a batch-oriented compiler and runtime system.

255x Series Network Processor Unit

The hardware portion of the communications network consists of a 255x series Network Processor Unit, comprising:

- A microprogrammable, 16-bit processor (minicomputer). Main memory contains all the space necessary to execute programs and to provide buffers for network data. Mass storage memory is not used.
- A host channel coupler, which provides the high-speed interface between the minicomputer and the host peripheral processor unit (PPU). Transfers over this channel are buffered.

  Channel buffers within the minicomputer provide enough space to handle an entire message transfer in a single operation.
- A multiplex subsystem, consisting of the following:

  (1) A multiplex loop interface adapter (MLIA) which controls the input and output multiplex loop.

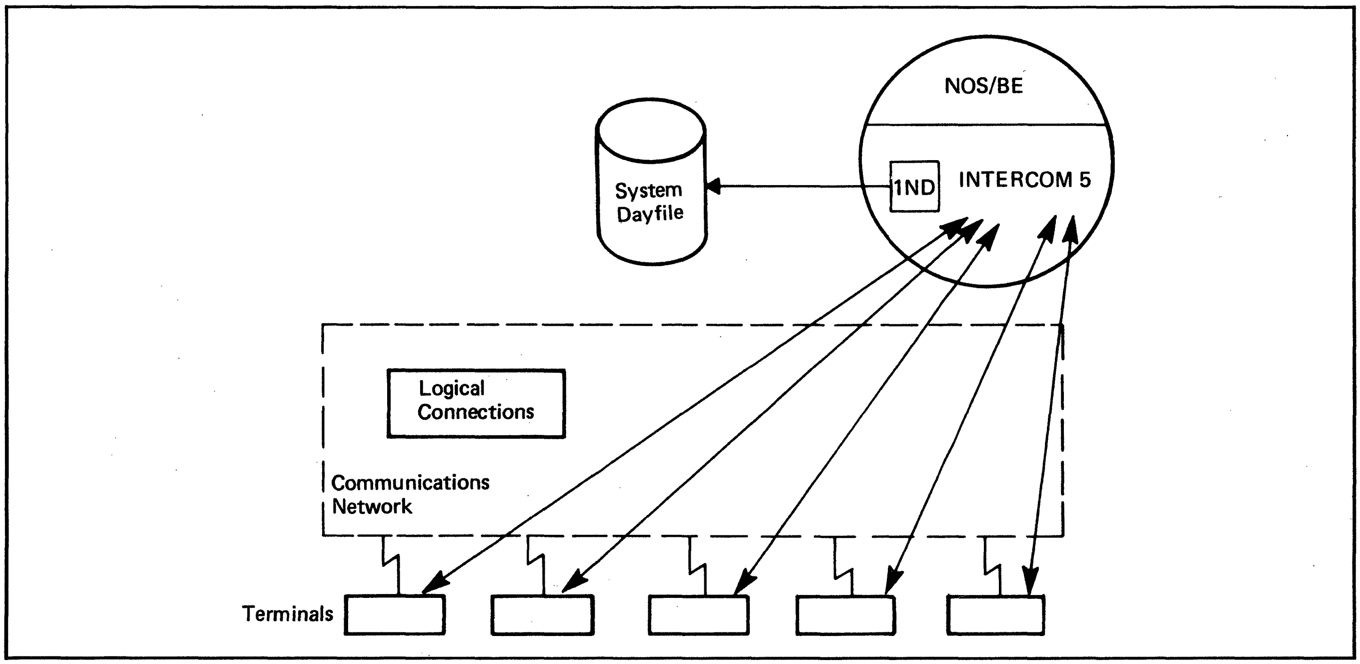


Figure 1-4. Communications Supervision Functions

(2) Individual loop multiplexers which attach to the input and output multiplex loop on one side and to individual communication line adapters (CLAs) on the other. The CLAs provide line-by-line interface compatibility with the modems attached to terminals. (Terminals can also be connected by direct wire or transceivers.)

NOTE

The interface of the NPU to the modems passes data to and from the terminals in a format (protocol) compatible to the terminal; that is, characters 5 to 8 bits in length accompanied by the necessary control and status information.

- A local NPU communications console, which can be used to run diagnostics and/or to initiate and receive status and error information about the NPU operation.

Communications Control INTERCOM

Communications Control INTERCOM (CCI) has three major parts:

- The base system software, which includes the operating system (OPS) monitor, interrupt handlers, timing services, multiplex subsystem software and firmware, initialization routines (the NPU is downline loaded from the host), space allocation routines, program-to-program call and data transfer features, standard subroutines, NPU console control, text processing, and error checking.

- Network communications software, which provides configuration control, routing, and handling of control and status messages/statistics, and terminal interface support.

NOTE

Inline diagnostics are provided as part of CCI. If the customer elects to purchase a CDC maintenance contract, online diagnostics are provided.

- Interface software, which provides the following standard types of interface programs:

  (1) Host Interface Program (HIP). This program supports the high-speed, buffered interface to the host computer. Data is normally assumed to be in internal terminal format.

  (2) Terminal Interface Programs (TIPs). Four standard TIPs handle transfers for terminals that operate using asynchronous Mode 3, synchronous Mode 4, 2780/3780, and HASP multileaving protocols, respectively.

CCI Coding Languages

For ease in programming the NPU, the programmer codes source language routines in CYBER Cross System PASCAL, a language similar to ALGOL. The CYBER Cross System programs are run on the host, and the principal output is a load file in NPU machine language, which resides on host mass storage. This load file contains all of the CCI modules in NPU image format, and is used to load the NPU following an NPU or host failure. The CYBER Cross System software is described at the end of this section.

A few common programs are coded in the CYBER Cross System macro assembler language. A special subset of this language called the state programs use a set of predefined macro commands to process messages at the microcode level. Most Terminal Interface Programs contain message conversion routines written using this state programming language. All such programs, regardless of source language, are included in the CCI load file.

MESSAGE MOVEMENT IN A NETWORK

The basic concepts and procedures of upline and downline message movement are summarized in the following subsections. Most of the operations are described in more detail later in this section. These operations are also discussed in general terms, by individual functions, in section 2.

DATA FLOW

The data flow in the network, from the terminals to the host system or vice versa, can be divided into three classes: interactive input and output, batch input, and batch output (figure 1-5). The unit of information transmission used in this flow between the host and the NPU is called a block. A block is up to 2047 bytes long, including a block header. The length of a block depends on its type and source. All blocks are either:

- Nondata blocks, which are blocks that travel on any connection and have the formal types CMD or BACK. CMD blocks are discussed in appendix H; BACK acknowledgement blocks are discussed in this section under Data Flow Control.

- Batch data blocks, which are blocks that travel on batch terminal connections and have the formal types BLK or MSG. These block types are discussed in appendix H, where the data exchange protocol required by INTERCOM 5 is summarized.

- Interactive data blocks, which are blocks that travel on interactive terminal connections and have the formal types BLK or MSG.

The first four bytes of all block headers are similar and are described in appendix H. Four additional bytes are used in data block headers. These additional bytes contain the following information:

- A data block clarifier byte. For batch data, this byte indicates whether the block contains transparent data, whether it contains an end-of-record or end-of-information indicator, whether it contains a banner message (file identification record), and whether it contains batch or interactive data. For interactive data, this byte indicates the carriage control information for the data, and whether the data is batch or interactive.

- Two bytes reserved by CDC to contain optional information.

- A level number byte. For batch data, this byte contains a record level number. For interactive data, the byte is present but the contents are not used.

The flow of the three classes of data using data blocks is described in the following paragraphs. Nondata blocks, such as service messages, are discussed separately in other portions of this manual.

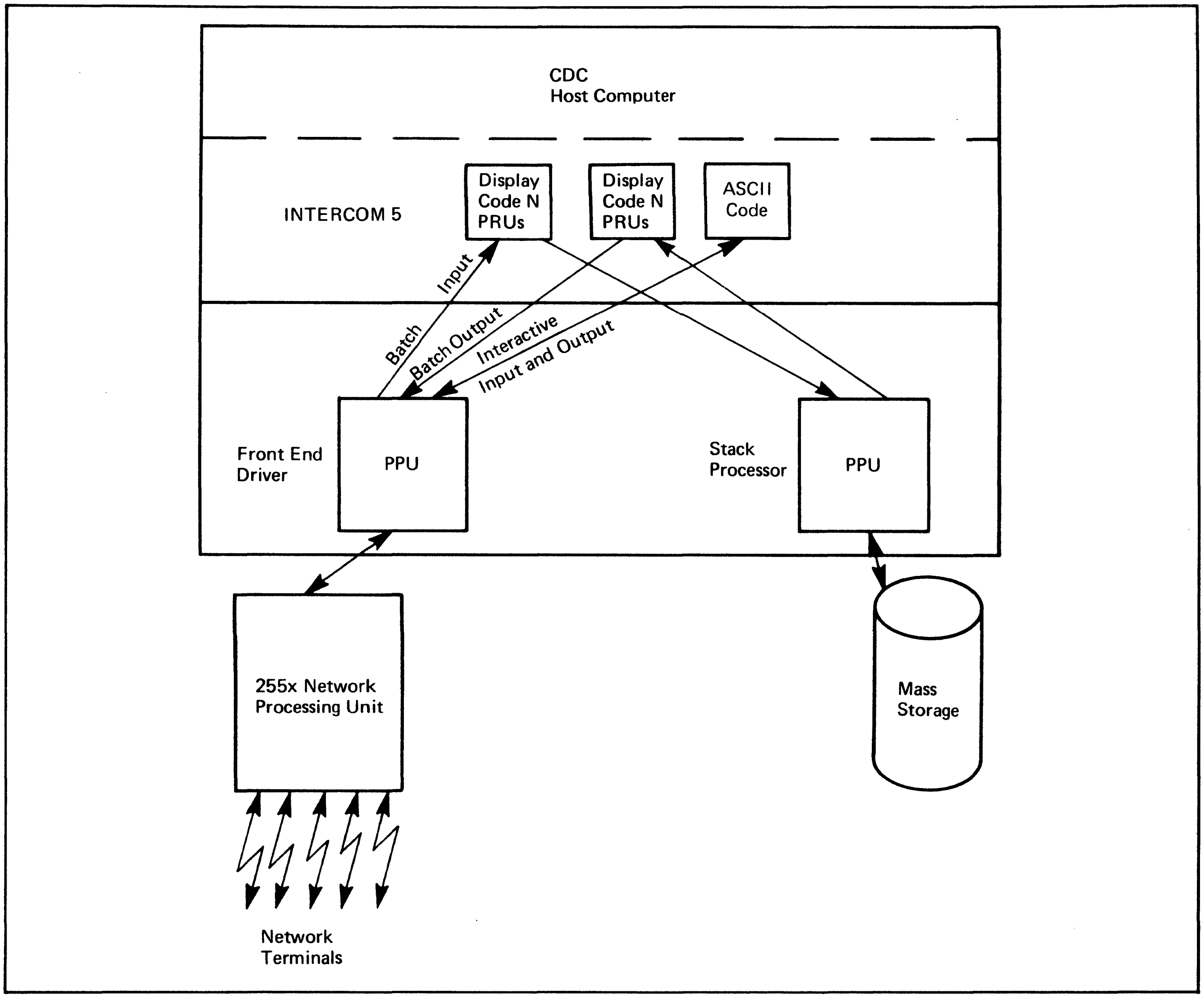


Figure 1-5. INTERCOM and CCI Data Flow

Interactive Data

Interactive data is passed between the NPU and host as line images coded in 7-bit ASCII code. Interactive input is accumulated by the NPU until a line is completed and then passed to the host system as a block of data. The host system processes the line or delivers it to the application running under INTERCOM, as required. Interactive output is passed from the host system to the NPU in 7-bit ASCII-coded blocks of line images. The NPU delivers these messages to the proper device at the terminal.

Batch Input Data

Batch input is received by the NPU, transformed to a standard format (usually host 6-bit display coded characters in 8-bit bytes), and accumulated as mass storage-sized PRU images. These PRU blocks are then transferred by the NPU to the host. The host system writes this data to mass storage. End-of-record (EOR) and end-of-information (EOI) indicators are transferred as short PRU blocks. The actual number of PRUs transferred as a block across the coupler is a system assembly parameter of 1, 2, or 3 PRUs.

Batch Output Data

Batch output is retrieved from mass storage by the host system and transferred across the coupler to the NPU as PRU blocks. End-of-record and end-of-information are indicated in the block header. The start of a new output file is indicated by the host system as a banner message block, and the NPU can optionally generate a banner transmission block for the terminal from the file name provided in the banner message block. The NPU is responsible for blocking line images from the PRU block record and resolving the logical format control in the data record to the physical characteristics of the output device on the terminal.

Data Flow Control

Data flow is controlled by connection numbers. Each terminal has at least one full-duplex connection number, the interactive data stream. The interactive connection is always defined for an active terminal. This data stream can be considered a full duplex channel by which the terminal operator can control the operations of his terminal. Because of hardware restrictions of some terminals, the use of this connection number is sometimes mutually exclusive with all batch data flow. For Mode 4A and 2780/3780, interactive data or batch data can be active, but not simultaneously. For HASP workstations and Mode 4C terminals, both batch and interactive data streams can be active simultaneously.

Batch Input Control

Each batch input stream has its own connection number and is initiated by a command to the NPU from the host system. The command specifies the mode of the transmission; either display code or transparent mode.

There are two types of input transmissions supported. The INTERCOM command READ permits data to be transferred from the input device on the terminal to a local file on the host system. Only a single data file is transferred with this command. The data consists of all data up to the EOI indicator. An EOI indicator is defined as a 6/7/8/9 card or /*EOI card for display code transmissions, and as the end of the input data stream for transparent transmissions. When the EOI indicator is received by the NPU, the final data block is sent to the host system.

The second type of input transmission is job stream input. A job stream input is inherently a display code transmission. A job within the stream is terminated by an EOI indicator card. When the EOI indicator is received by the NPU, the final data block for that job is sent to the host. INTERCOM 5 completes that job file and submits the input job to the input queue of the host.

A special set of statistics called Accounting data is kept, both for sinqle file inputs and for job streams. Accoutning data consists of the number of cards read from the remote entry device in one file or job stream. Accounting data is sent to the host under three separate conditions:

- For a normally completed input job, the count is sent to the host just before the EOI indication is sent upline.

- For a job interrupted by the host, the count is sent as a response to the stop input message.

- For extremely large inputs, it is possible for the 65K counter to overflow. If it does, the count is sent to the host at overflow time.

Input stream errors, such as input device not ready, are reported upline (if the terminal hardware permits) by use of an asynchronous upline command stating the input has stopped and the reason. The NPU maintains any accumulated data for the stream. The host either restarts or terminates the input stream with a downline command as the result of a command interchange with the terminal operator.

The NPU initiates the initial transfer of a batch input stream block. Each subsequent block waits for downline acknowledgement of the previous block from the host. The NPU queues multiple blocks of upline data for each input connection by this method.

Batch Output Control

Each batch output stream has its own connection; output occurs when it is available for the stream, and the terminal is in a state to permit reception of the data.

The output data is preceded by a file identification record containing the name of the file. The NPU converts the file name to a banner message (if required) that is consistent with the output device at the terminal. The file can be in either transparent mode or display code, as specified in the block protocol header.

For print files, the NPU blocks line images to the terminal and processes the carriage control characters consistent with the print device on the terminal. The detection of a print line with PM as the first two characters requires that the NPU direct that line to the interactive output device and notify the host with an upline command that the print stream has been stopped by a PM message. Data compression is also performed by the NPU.

Accounting data statistics are also kept for output files. Output accounting data consists of the number of unit records sent to the output device. As in the input case, the accounting data is sent to the host under three separate conditions:

- For a normally completed output job, the count is sent to the host in response to the EOI sent from the host.

- For a job interrupted by the host, the count is sent as a response to the stop output message.

- For extremely larqe outputs, it is possible for the 65K counter to overflow. If it does, the count is sent to the host at overflow time.

Output stream errors, such as output device not ready, are reported upline (if the terminal hardware permits) by use of an upline command stating the output stream has been stopped and the reason. The NPU maintains any undelivered data for the stream. Depending on the terminal characteristics, either the NPU restarts the transmission if the terminal signals the end of the error condition, or the host either restarts or terminates the output stream as a result of a command interchange with the terminal operator.

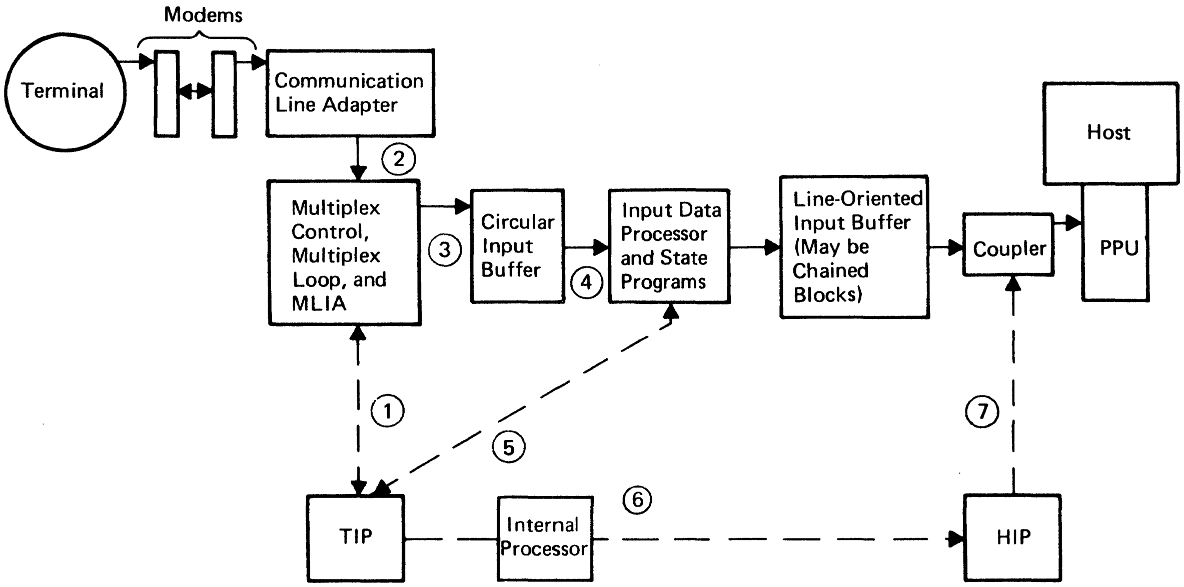
The NPU controls the buffering and flow of output data by use of an upline acknowledgment block. The upline acknowledgment block requests the next output block for that data stream. The acknowledgment block is issued whenever the output buffers for the stream are not full and does not necessarily indicate delivery of a complete PRU block to a terminal - unless the PRU block contained an EOI indicator. Occurrence of an EOI indicator in a downline block generates an acknowledgement block, regardless of buffer status.

SIMPLIFIED INPUT MESSAGE PROCESSING

Figure 1-6 shows the movement of a message from a terminal to INTERCOM 5. Solid lines indicate message (and acknowledgment message) pathways; dashed lines indicate principal control functions. The following are major features of upline message processing:

- Start of input is initiated. If this is a Mode 4 synchronous terminal, the terminal is polled to find if it has data ready to send. If this is a bisynchronous terminal, permission to send data is granted so that transmission can occur. An asynchronous terminal sends data when it is ready.

- Setting up of the multiplex subsystem and buffers when a terminal indicates it has data to send upline.


Modems
Terminal
Communication Line Adapter
2
Multiplex Control, Multiplex Loop, and MLIA
3
Circular Input Buffer
4
Input Data Processor and State Programs
Line-Oriented Input Buffer (May be Chained Blocks)
Coupler
Host
PPU
1
5
7
6
TIP
Internal Processor
HIP

(1) Sets up input transfer.

(2) Raw input message.

(3) Multiplexed input message is placed in circular input buffer.

(4) Processes characters and detects end of message (demultiplex and convert to data block format; translate terminal code if necessary).

(5) Validates message and passes it to internal processor.

(6) Determines routing with help of directory, requeues message for coupler. Control to HIP.

(7) Sets up transfer to PPU. Releases input buffers, when coupler status shows that transmission is complete.

Figure 1-6. Simplified Input Message Processing

- Collecting all data from an active terminal into a circular input buffer.

- Demultiplexing data and simultaneously converting it to interactive or batch data format. Demultiplexed data is collected in a block that uses one or more chained buffers called a line-oriented input buffer. (If code conversion is necessary, this is also accomplished.)

- When the input buffer is full (that is, the message is complete), the message is validated.

- Message routing is determined and the message is queued to the host coupler. Statistics are generated.

- The coupler transmits the message to the host PPU and unqueues the message from the coupler.

- The host (INTERCOM 5) receives the message.

- The NPU finishes input processing by releasing the message buffer.

SIMPLIFIED OUTPUT MESSAGE PROCESSING

Figure 1-7 shows the movement of a message from INTERCOM 5 to a terminal. Conventions for solid and dashed lines are the same as for the input message diagram. The major features of downline (output-to-NPU) message processing are as follows:

- An interrupt from the host indicates a buffer of data (a message) is ready for transmission. The data is normally in interactive or batch format at this point.

- If the NPU is not already saturated with other, higher priority tasks (note that output takes precedence over input), the Host Interface Program (HIP) sets up the coupler to receive the message and assigns a buffer (or chained buffers) of space to be used as an output buffer for the block.

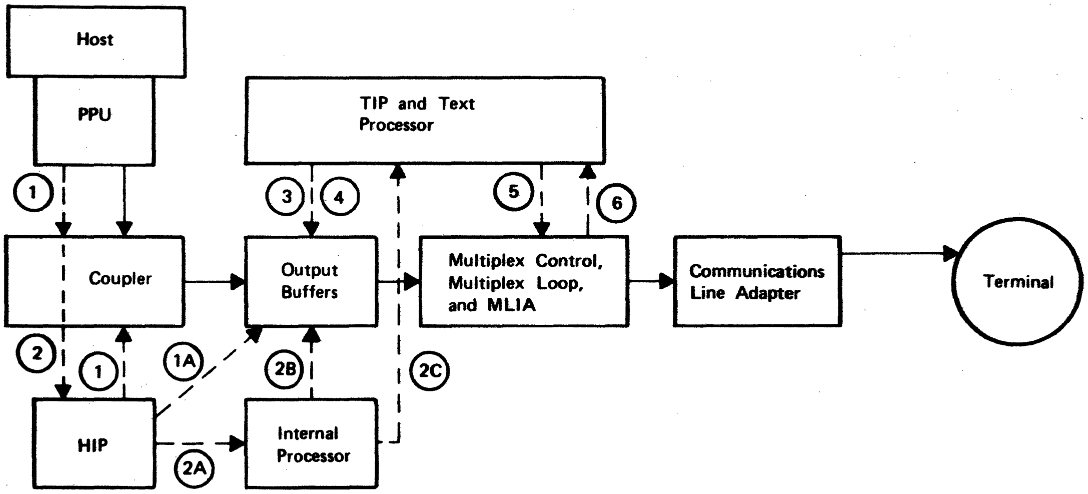
- The message is sent by a buffered transfer from the PPU through the coupler to the assigned buffer. The coupler causes the transmission complete interrupt to be sent to the HIP when all of the message data has been received.

- The message is queued to a terminal control block where the Terminal Interface Program (TIP) detects that the message is complete.

- The text processor converts the message in place from batch or interactive data format to the format required by the destination terminal.

- If the terminal is able to output data, the TIP directs the multiplex subsystem to output the message.

- When the terminal has detected the end of message, it sends an acknowledgment message.



1 Sets up both PPU and HIP for coupler to receive message

1A Sets up buffers for output message

2 Coupler sends interrupt when all of message is received

2A Notifies internal processor by a worklist that message is in buffer; converts to proper format for this terminal if data is batch type.

2B Queues message to terminal using TCB

2C Internal processor notifies TIP with worklist entry that this is the first message in the queue (if necessary) and sends an acknowledgment to the host if more data is needed to keep buffers full

3 Checks TCB for messages ready to transmit

4 Converts to proper format for this terminal if data is interactive type

5 Directs message transmission when terminal responds it is ready

6 When acknowledge message is received, buffer is released

Figure 1-7. Simplified Output Message Processing

- Upon receivinq acknowledgment that the message was received, the NPU terminates the output operation by releasing the output buffers, and sends an acknowledgment to the host if more data is required to keep buffers full. Statistics are also gathered.

CCI ROLE IN NETWORK PROCESSING

The CCI must provide the following functions for the network:

- Host interface compatibility: Data must be transferred (upline and downline) in high-speed buffered transfers. Data is usually transferred in batch or interactive data format.

- Transfers can be regulated; that is, if the NPU is busy with output (downline) processing or has many upline messages already started, new upline transfers can be refused. The rejected transfers will be made at a later time when the NPU is less busy.

- The NPU must pass an acknowledgment message to the host in order to obtain more data.

- The NPU must prepare the coupler for upline and downline transfers.

- Conversion: Upline messages are converted from terminal protocol format to batch or interactive data format. Downline messages are converted from one of these data formats to the protocol appropriate for the destination terminal. The Terminal Interface Programs are responsible for maintaining the transform tables necessary to convert the data.

- The CCI must supply main memory space in the form of chained input and output buffers.

- Supervision of the multiplex subsystem. On output, supervision consists of preparing the multiplex subsystem to output data from the output buffer. Actual transmission occurs on demand from the communications line adapter (CLA). Input supervision consists of preparing the CLA to receive data. After data has been placed on the input loop, the multiplex loop interface adapter transfers the data to the circular input buffer. From the circular input buffer, data must be individually demultiplexed to the line-oriented input buffers.

- Terminal Interface Programs (TIPs) are responsible for setting up the messages so that the terminal protocols for startina, stopping, acknowledgments, and message formatting are satisfied. The TIP also converts data from ASCII or display code to the terminal code (EBCDIC, ASCII, and so forth), if necessary.

255X NPU HARDWARE

The basic product group for the NPU and associated hardware is as follows:

- 2550 or 2551 Network Processor Unit. This consists of a communications processor with a minimum of 48K words (96K characters) of main memory, 2K words of micromemory, a maintenance panel, a maintenance monitor tape cassette for off-line diagnostic tests, a cyclic encoder, a multiplex loop interface adapter (MLIA), and a loop multiplexer.
- Optional main memory expansion units (2554-16 for 16K words of added memory or 2554-32 for 32K words of added memory) are available. Maximum main memory size is 128K words.
- A required NPU console.
- At the host interface, a 2558-3 channel coupler is required.
- At the terminal interface, one or more communications line adapters are required.

COMMUNICATIONS PROCESSOR

The communications processor is a microprogrammed 16-bit processor. Its program instructions are stored in main memory, while the code that dictates how they are to be executed is stored in micromemory. This microcode provides additional power for character and field manipulation, indexing and other communication-oriented processes. The following are major features of the communications processor:

- 48K to 128K words of 16-bit (plus parity and protect) magnetic oxide semiconductor main memory
- Eight memory-addressing modes
- Memory word and region protection
- Main memory parity detection
- High-speed input/output data transfer
- Programmable cyclic encoder

Macro Program Instruction Repertoire

The communications processor incorporates the CDC CYBER 18 series computer instruction set. Some instructions are immediate (literal), resulting in a saving of operand storage space and execution time. Multiple word instructions, like indirect addressing, are a means of addressing locations that cannot be accessed directly.

To aid in programming, most of CCI is written in CYBER Cross System PASCAL. For efficiency, some portions are written in CDC CYBER 18 Series macro assembly language. CYBER Cross System PASCAL output is executed using the CYBER 18 series instruction set.

Registers

The communications processor emulates a total of 12 registers. Eight registers are used in the execution of the normal CDC CYBER 18 instruction set, while four general-purpose registers support the extended instruction set. There are also three special-purpose registers used exclusively for machine control.

Input/Output

The communications processor features both a direct memory access (DMA) interface and an internal data channel (IDC) interface to peripherals. Using the IDC interface, the program controls the data transfer, employing the Q register to address the desired peripheral and the A register to transfer data, commands, or status between the communications processor and peripherals. The IDC transfer rate is 160,000 bytes per second. The DMA interface permits direct transfer of data between the peripherals and main memory, bypassing the communications processor main registers. The DMA transfer rate is 2.8 million bytes per second. Coupler transfers to the host use the DMA channel.

Program Protection

The communications processor offers two modes of protection from the damage that can be done by programs accessing memory outside of their own region:

- Individual words are declared protected by setting a bit in memory associated with that word. This protection method is provided by the CCI software.
- Upper and lower bounds are used to define an unprotected region. This definition has the same effect as word protection, except that a large unprotected area can be defined more quickly.

Interrupt System

The interrupt system is implemented through microprograms, and consists of 16 levels of external macro interrupt and 16 levels of internal micro interrupt. Macro interrupts are regular CDC CYBER 18 series interrupts; micro interrupts are primarily used internally to control the execution of micromemory instructions associated with the real-time processing of input/output data.

Breakpoint

The breakpoint facility, implemented in microprogram code, insures the termination of a program at a predetermined location in memory. Breakpoints are useful in program debugging. This feature is activated through the NPU maintenance panel and can be used at either the macroprogram or microprogram level.

MAINTENANCE PANEL AND TAPE CASSETTE

The communications processor features a maintenance panel which is also usable by programmers and provides for the following:

- Display and modification of register contents
- Display and modification of memory locations
- Operating switches
- Indicators

A tape cassette is provided as part of the NPU configuration. The tape cassette is used as a load device for communications processor off-line controlware diagnostics.

NPU MAIN MEMORY

The NPU uses high-speed magnetic oxide semiconductor memory. All memory configurations feature the use of 18-bit storage words comprised of the following:

- 16 data bits
- 1 parity bit
- 1 program-protect bit

Communications processor microprogram code allows full, direct access to 65K words of memory. Use of larger memory (over 65K) is restricted to TIPs, diagnostics, and so forth. Features of this large memory use are as follows:

- Virtual memory space of 65K at any time
- 32-page registers to allow mapping of each 2K page in virtual memory space to any 2K physical page
- 2 sets of page registers to allow for rapid switching

MULTIPLEX SUBSYSTEM

This system uses demand-driven multiplexing. Processor intervention is required only when incoming data characters are received or when output data blocks require servicing. The multiplex subsystem includes the following components:

- Multiplex loop interface adapter (MLIA)
- Loop multiplexer
- Communications line adapters (CLAs)

The first two are an integral part of the 255x, although the communications processor treats them as peripheral devices. The CLAs are peripheral devices attached to the input and output multiplex loops.

MULTIPLEX LOOP INTERFACE ADAPTER

The multiplex loop interface adapter connects the communications processor to the loop multiplexers via the multiplex loop. It initiates and terminates the input and output multiplex loop requests. It also provides basic timing and control. The MLIA provides serial-to-parallel and parallel-to-serial data conversions, loop control, and error monitoring.

LOOP MULTIPLEXER

The loop multiplexer provides the electrical and mechanical interface between the multiplex loop and communications line adapters which reside within the loop multiplexer. Additional loop multiplexers are available as CDC 2556-11 Loop Multiplexer Line Expansion Modules. A total of eight loop multiplexers can connect up to 254 communications lines.

COMMUNICATIONS LINE ADAPTERS

The CDC 256x Communications Line Adapters (CLAs) are used to interface the NPU to various types of terminals or communications facilities. The CLA provides the electrical interface, isolation, control, and interim character buffering for the five types of communications lines shown in table 1-1. There are two series of 256x CLAs used with CCI:

- 2560 Series Synchronous CLAs
- 2561 Series Asynchronous CLAs

Specific characteristics of the communications line adapters and modems are described in the CCI 3 system programmer's reference manual.

The 2560 CLA provides for the connection of two synchronous communications lines. Different data sets (modems) and terminals are accommodated by the selection of an appropriate cable. The 2560-1 provides for the connection via modems (data sets) of lines conforming to the EIA RS-232-C or CCITT V24 Interface Standard. The CLA is compatible with AT&T 201/208 Data Sets and provides for the termination of two communications lines.

The 2561 CLA provides for the connection of two asynchronous communications lines at all standard speeds up to 9600 bits per second. There is one model, the 2561-1. The 2561-1 provides for the termination of two lines conforming to EIA RS232C or CCITT V24 standards. This CLA is compatible with AT&T 103/113/ data sets or their equivalent. Local connection without a modem is available. The 2561-1 CLA provides many features, most of which are software controlled.

NPU CONSOLE

The following standard types of console can be used:

- CDC 713-10 Display Terminal, at variable speeds up to 30 characters per second
- CDC 752-10 Display Terminal, at variable speeds up to 9600 bits per second

Other teletypewriter devices with compatible line speed and input/output characteristics also can be used.

CYBER CHANNEL COUPLER

The CDC 2558-3 CYBER Coupler provides a direct link between a host computer and the communications processor. The primary function of the coupler is to pass 8-bit data characters directly between the computer memories with minimum software supervision. An additional four control bits are associated with each character.

The coupler provides the means for transfers between a host peripheral processing unit (PPU) and an NPU at host channel transfer rates. Supervision is provided by both PPU and NPU software commands and by control words in the NPU buffer. Buffer chaining to make one large block in NPU memory is provided.

TABLE 1-1. 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	RS-232-201A/2081 Compatible	Switched	Controlled	2 Wire	Yes	No	Synchronous
(2)	FDX†	2560-1	RS-232-201B/208A Compatible	Dedicated	Controlled	4 Wire	Yes	No	Synchronous
(3)	FDX	2560-1	RS-232-201B/208A Compatible	Dedicated	Constant	4 Wire	No	No	Synchronous
(4)	HDX	2561-1	RS-232-358-1 Compatible	Dedicated	Controlled	2 Wire	Yes	No	Asynchronous
(5)	HDX	2561-1	RS-232 Compatible	Switched	Controlled	2 Wire	Yes	No	Asynchronous
(6)	FDX	2561-1	RS-232-103E/113 Compatible	Switched	Constant	2 Wire	No	No	Asynchronous
(7)	FDX	2561-1	RS-232-103E Compatible	Dedicated	Constant	2 Wire	No	No	Asynchronous
(8)	HDX	2561-1	RS-232 Compatible	Switched	Controlled	2 Wire	Yes	Yes	Asynchronous
(9)	FDX	2561-1	RS-232-103E/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.

M-935

All coupler operations are initiated by PPU function commands and/or NPU I/O commands. The transfer of data between computers requires control words in NPU memory and additional PPU I/O instructions.

SAMPLE CONFIGURATIONS

Figure 1-8 shows a typical configuration for a 2551 series NPU. The 2551-1 is a front-end NPU limited to a maximum of 32 input lines.

The 2551-2 or 2550-2 is a front-end NPU and can service up to 254 communications lines.

MULTIPLEXING OPERATION

The multiplex subsystem has two major functions, both of them hardware related:

- Line characteristics vary for different types of terminals. To relieve the TIPs of having to process each line type according to the special characteristics of the line, the multiplex subsystem performs this function. This makes most line characteristics transparent to the TIP.

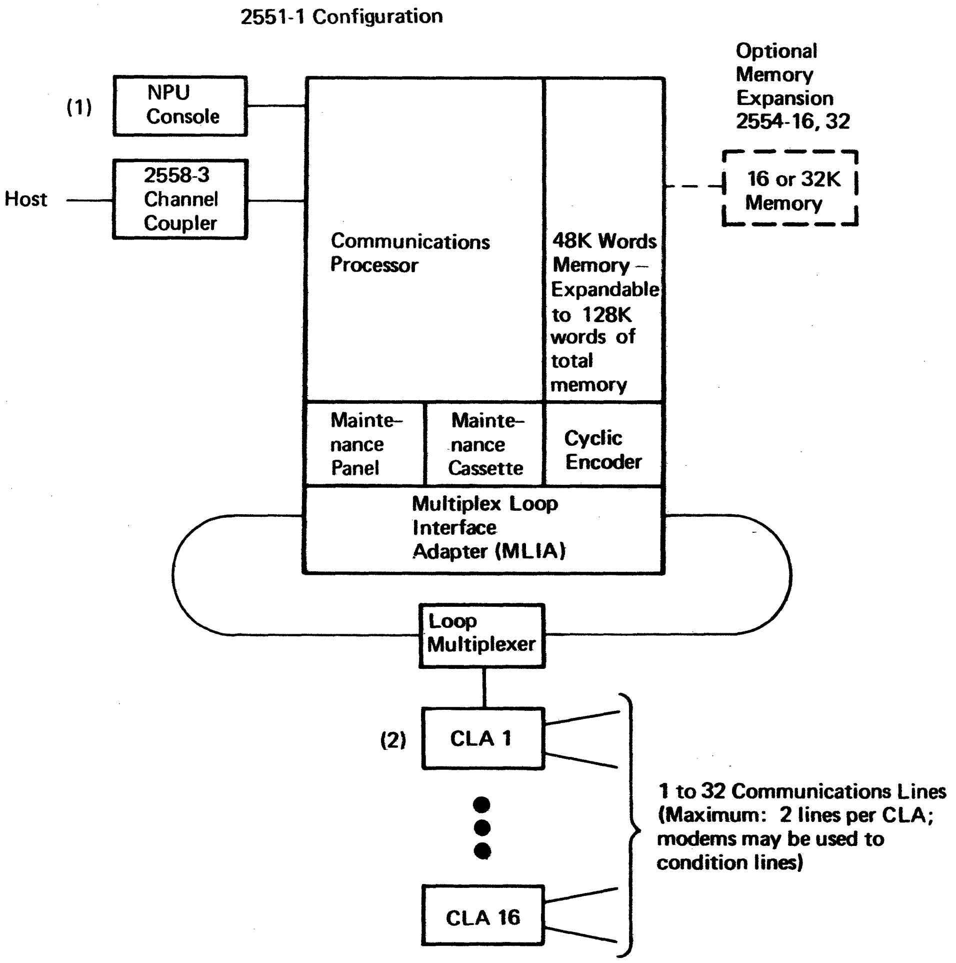
- The high-speed host works most efficiently if given a full block of data to process. A terminal, on the other hand, is often low speed; and data is transferred to and from the terminal one character at a time. The multiplex subsystem interfaces the high-speed characteristics of the host and NPU with the low-speed characteristics of the lines and terminals.

INPUT MULTIPLEXING

Each line has a communications line adapter (CLA). The CLA for each active line is sampled in sequence. If a character is ready, it is placed on the input multiplexer loop together with information identifying the source (line) and, in some cases, control information. All input multiplexer loop data is routed to a circular input buffer. The demultiplexing operation picks data from this buffer, reconstitutes messages in data buffers, and passes these buffers to appropriate processor for this terminal (line).

OUTPUT MULTIPLEXING

After the NPU has received a full message from the host and the appropriate Terminal Interface Program (TIP) has converted the data to terminal format, the multiplex subsystem can output the message. The multiplex subsystem picks the characters from the output message buffer of the line in response to an output data demand (ODD) generated by the line. (The ODD signal is the indication that the line is ready to receive another character.) The outgoing characters are placed on the output multiplex loop, along with such control characters as are needed and an address that is recognized by the active line for that terminal. The CLA for the line picks the data from the output loop. When the contents of the entire output buffer for the line have been transmitted successfully, the message buffers are released and the host is notified of a successful transmission. Several output data buffers can be serviced at the same time, so that an output multiplex loop frame normally has a character for each of several active lines.


2551-1 Configuration
Optional Memory Expansion 2554-16, 32
16 or 32K Memory
(1)
NPU Console
Host
2558-3 Channel Coupler
Communications Processor
48K Words Memory – Expandable to 128K words of total memory
Mainte- nance Panel
Mainte- nance Cassette
Cyclic Encoder
Multiplex Loop Interface Adapter (MLIA)
Loop Multiplexer
(2)
CLA 1
CLA 16
1 to 32 Communications Lines (Maximum: 2 lines per CLA; modems may be used to condition lines)

NOTES:

Required ————

Optional — — —

(1) Required console can be 713-10 CRT, 752-10 CDT or equivalent

(2) CLAs and modems selected as required. One or two communications lines can be attached to a CLA depending on line type. A maximum of 16 CLAs are allowed in this configuration.

Figure 1-8. Typical NPU and Peripheral Hardware Configuration

DEMULTIPLEXING

The multiplex subsystem is responsible for picking data from the circular input buffer as well as putting it into that buffer. When message reception starts, the multiplex subsystem firmware reserves a data buffer for the message. The data words for that line are picked from the circular input buffer and are packed into the reserved buffer. Control and tag information is discarded. If a buffer is filled before the message is complete, another buffer is assigned and is chained to the first.

When the end of text is detected, the TIP appropriate for the terminal class is called to continue the processing.

The demultiplexing of a downline transfer is a terminal function; that is, the message is reconstituted in a buffer for the screen, printer, or other device.

BASE SYSTEM SOFTWARE

The base system software, which includes the multiplex subsystem, is a set of CCI programs. As figure 1-9 shows, CCI software belongs to one of three levels:

- Base system software
- Network communications software
- Interface programs (TIPs, HIP)

Base System Software

- Operating system (space allocation, calls, interrupt handling)
- Multiplex subsystem
- Interprogram communications, worklists
- Timing services
- Standard subroutines
- NPU console services
- Common TIP subroutines

Network Communications Software

- Internal processor maintenance
- Routing (internal processing)
- Service message handling
- Common TIP subroutines

Interface Programs

- Host Interface Program (HIP)
- Terminal Interface Programs (TIPs):
    - Mode 4 TIP
    - TTY TIP
    - HASP TIP
    - 2780/3780 TIP
    - User Generated TIPs
    - Diagnostics TIP

Figure 1-9. CCI Software Levels

The primary functions supplied by the base system software are as follows:

- Basic operating system functions such as interrupt handling, calling programs (queuing requests), allocating space (buffers) to requesting programs, handling the passing of parameters from the calling to the called program (worklist processing), timing services, common areas, and control block services
- Initialization processing
- Multiplex subsystem processing
- Common subroutines, such as those for code conversion
- NPU console handling; NPU console data is not routed through the multiplex subsystem

Most base system subroutines are written in the PASCAL language; a few are written in CYBER 18 series computer macro assembler language. Programs can be called directly from another program, or requests can be queued for execution by generating a worklist entry and queuing that entry on a first-in, first-out basis to the called program. The monitor processes worklists on a strict rotation basis whenever no interrupt-driven programs are waiting for execution. Provision is made for giving control to a worklist driven program for more than one task (that is, more than one worklist entry will be processed before the called program returns control to the monitor).

Transfer of information (messages) through the host/NPU part of the network is accomplished by transmitting the data in blocks (this is called block protocol throughout this manual). A number of block types are defined. Two of these block types are dedicated to data transfer; the remaining block types transmit control information: commands, acknowledgment of message reception, and control of message interrupts.

NETWORK COMMUNICATIONS SOFTWARE

The next level of CCI software is concerned with handling network communications. It also has block handling capabilities. It provides block switching so that the data block can ultimately be routed downline to the proper terminal, or upline to the host. Data traveling in either direction is tagged with network header bytes to make this routing possible.

Upline data is blocked into PRU blocks by the internal processor before transmission to the host. Downline data is unblocked from PRUs and block acknowledgment messages can be sent before data is delivered to the terminal; such acknowledgment messages ensure that enough output is buffered so that the TIPs always have work to do.

The service module (part of the network communications software) is largely responsible for handling the set of specific commands that set up the lines, an operation usually called configuring the system. Also, each interface program (HIP or TIP) is responsible for handling one specific protocol. That interface program also shares in the block handling responsibility.

One type of block, called a CMD block, is dedicated to handling the large number of specific commands transmitted throughout the system. The information carried by these CMD blocks is called a service message if the connection number is zero. Service messages perform the following operations:

- Configure lines and terminals.
- Report loading of the NPU.
- Carry status concerning failure and recovery.
- Command that a message be broadcast to one terminal or several terminals (including sending a message from a terminal to the host system console operator).

CMD blocks with a nonzero connection number contain control and status information concerning a specific device or data stream on a terminal. The following operations are controlled by these CMD blocks:

- Control stream on or off conditions
- Report device status and break conditions
- Report end-of-file conditions

HOST INTERFACE PROGRAM

The Host Interface Program (HIP) handles the protocol governing transmissions between the host and the NPU. The route of all such transmissions is through the coupler hardware. Three coupler registers hold status or command information; one register contains the NPU address of the data to be transferred, and one set of lines connects the host PPU buffer to the direct memory access buffer register of the NPU. This set of lines handles all data transfers.

In all cases, the format of the byte in the host PPU is 12 bits and the associated half word (one byte) in the NPU is 8 bits. Adjustment is made so that the receiving unit (host or NPU) has an input only as larqe as its input word or byte size.

Usually, the NPU memory address register is set up by the NPU for upline or downline data transfer. When dumping the contents of the NPU to the host, the PPU sets up the reqister to supply the addresses for the memory to be dumped.

Three principal functions are performed across the interface between the host and the NPU:

- Control words are transferred in both directions.

  The NPU sends function commands to the coupler; these commands allow the coupler to chain buffers of data during transfer, to clear the coupler registers, to read the other status registers, to ready the coupler to read the status registers, and to set the memory address register prior to starting a data transfer.

  The PPU sends functions to clear the coupler or NPU, to start or stop the NPU, to input or output a program during the load and dump phase of NPU initialization, to load the memory address for dump operations, and to set or read the other two status type registers.

- Status words are provided. One status word is used for regulation. There are three regulation levels: to transmit all messages to the NPU, to transmit all but messages for batch type devices, and to transmit only service messages. Regulation is a function of the availability of NPU buffers to receive input messages.

- Data transfer functions are performed.

  For downline transfers, the NPU assigns the next data buffer, sets up buffer chaining if necessary, and switches the block to the internal processor.

  For upline transfers, when the message is ready, the NPU makes the address of the first buffer in the chain available. When one buffer is transferred, the starting address of the next chained buffer is provided by the coupler hardware. This continues until the full message is transmitted.

  Since the host-to-NPU interface channel is half-duplex (data can be sent in only one direction at a time), contention for channel use is normally resolved in favor of outputting blocks from the PPU. However, following this transfer, the protocol provides a short period during which the NPU can request channel use without contention with the PPU.

TERMINAL INTERFACE PROGRAMS

A Terminal Interface Program (TIP) interfaces the terminal data (messages) to the network. The terminal interface is processed through the multiplex subsystem; the system interface is processed through the line control blocks (LCBs) and terminal control blocks (TCBs). There are four standard TIPs that can be included in a system, as described earlier in this section.

Each TIP handles the protocol for its terminal classes. Specialized additional information for the terminal is contained in the TCB.

A TIP includes both hardware and software elements. The software portion of each TIP is written on three levels:

- An OPS-level (nonpriority and nonreal-time) is used for overall control of message transfer, including major error (transmission failed) processing, setting up transfers, and concluding the transfer.

- A multiplex-2-level (priority but not real-time) is occasionally used for error processing.

- A microprocessing (firmware) level is used for text processing during downline conversion of data and format. This level is also used for upline conversion of data and format for those protocols requiring two-pass input processing, and for input message processing. The latter includes demultiplexing from the circular input buffer to the line-oriented data buffer; this level also converts code and format at the same time for one-pass input processing.

The TIP operates at the first two levels. For downline data, the TIP escapes to firmware and regains control directly when text processing is finished. Upline text processing for two-pass TIPs works in a similar fashion. For input data processing (both one and two-pass TIPs), the TIP passes necessary information to the command driver of the multiplex subsystem so that the multiplex firmware can call the input state programs which move the message characters from the circular input buffers to the line-oriented input buffer and convert the characters (one-pass processing). In this way, the TIP itself is relieved by the multiplex subsystem from processing input data on the firmware level. The TIP operates on the OPS-level after the input data is fully demultiplexed and converted (one-pass processing). For two-pass processing, the TIP operates at the OPS-level prior to performing upline text processing.

The multiplexer interface to the TIPs is described in the CCI 3 system programmer's reference manual. The network interface is described in appendix H.

ASYNCHRONOUS TTY

The asynchronous TTY TIP supports dedicated and dial-up asynchronous lines using CDC Mode 3 protocol at speeds up to 9600 baud. The TIP provides software support for most teletypewriter-compatible terminals. The following terminals are included:

Terminal Class	Manufacturer	Model Number
1	Teletype	M33,35,37,38
1	CDC	713-10
1	Teletype	M40-2
1	Hazeltine	2000
1	CDC	751
1	Tektronix	4014

The TIP is prepared to receive input at all times and will attempt to deliver output whenever available, unless input is currently active. When input is detected during output, the TIP suspends output. This output will be sent later. Paper tape input and output is supported.

The TIP provides an auto-recognition feature for each line operating at rates of 300 baud or less. The result of this feature is a service message from the TIP informing the host of the line speed and terminal type for that line.

SYNCHRONOUS MODE 4

The synchronous Mode 4 TIP interfaces devices using Mode 4A or Mode 4C protocol to the network. The Mode 4 TIP supports the following equipment:

Terminal Class	Manufacturer	Model Number
10	CDC	200UT(BCD)
10	CDC	200UT(ASCII)
10	CDC	734
10	CDC	731-12
10	CDC	732-12
10	CDC	214
10	CDC	711-10
10	CDC	714-10/20

or any compatible terminals. The interface to the terminals complies with the basic Mode 4A and Mode 4C standards.

The TIP is insensitive to line speeds; it supports synchronous lines operating at rates up to 19.2K baud. Lines can be dedicated (with or without a transceiver) or switched (dial-up) with a modem. All lines are serviced as half duplex. Each line can have more than one cluster of equipment and each equipment cluster can have more than one terminal. Lines with multiple clusters must be dedicated.

The TIP performs auto-recognition when requested by the host. Auto-recognition causes the TIP to return a service message to the host which contains information on terminal type, cluster address, terminal address, and device type. Multiple-cluster auto-recognition is not supported.

The Mode 4 TIP supports remote batch terminals as separate but dependent devices.

The TIP polls the terminal to determine when data should be sent.

The TIP performs recovery for line or terminal errors. Any error from which an immediate recovery is not possible is reported to the host.

SYNCHRONOUS HASP MULTILEAVING

The HASP TIP provides network interfacing to a HASP multileaving terminal, which can contain both interactive and batch devices. These terminals include:

Terminal Class	Manufacturer	Model Number
9	CDC	CYBER 18-5 (HASP)
9	Data 100	78 (HASP)

or compatible terminals operating on point-to-point or dial-up communication lines.

The term multileaving describes the communications technique used by a HASP terminal. This protocol uses fully synchronized, pseudo-simultaneous, bidirectional transmission of a variable number of data streams between the NPU and the terminal. The HASP protocol requires binary synchronous communications facilities. In CCI, the multileaving capability is used only in the upline direction. The following features of HASP workstations are supported:

Feature	Supported
Multiple card	Yes
Character set	EBCDIC - 64
EBCDIC transparency (256 characters)	Yes
Character compression and expansion	All character strings
Console	Mandatory
No console	Not supported
Printer character set	EBCDIC - 64
Punch	Yes
Print width	80 through 150, inclusive
Binary cards	No
Plotter	As card/print emulation or transparent data
Magnetic tape	
Paper tape	
Postprint carriage control	No

The operational procedures for submittal of remote batch jobs and return of printer or punch output are patterned after procedures for the Mode 4A 200 User Terminals. The interactive INTERCOM 5 commands are available to the terminal users, and are entered through the HASP console; INTERCOM 5 interactive message responses and other unsolicited terminal operator messages appear on the HASP console, without interruption of batch input and output stream operation.

The TIP is insensitive to line speeds; it supports synchronous lines operating at rates up to 19.2K baud. Lines can be dedicated (with or without a transceiver) or switched (dial-up) with a modem. Auto-recognition, discriminating between HASP and 2780/3780 terminals, can be performed by the TIP when requested by the host. Auto-recognition causes the TIP to return a service message to the host which contains information on terminal type.

Each line can be configured with one cluster of devices serviced as separate batch and interactive terminals. This configuration recognizes up to 24 separate input and output streams for each workstation:

- interactive console input and output
- up to seven batch card reader input streams
- up to eight batch card punch output streams
- up to eight batch line printer output streams

The TIP permits all input streams to be active at any time, according to rules described later in section 2. Up to eight of the batch output streams and the console can also be active simultaneously. When contention exists between interactive input and output, output takes precedence.

The HASP console data is handled in interactive terminal format. HASP workstations must have a console device stream.

BISYNCHRONOUS 2780/3780

The binary synchronous communication 2780/3780 Terminal Interface Program supports the following terminals:

Terminal Class	Manufacturer	Model Number
7	CDC	CYBER 18-5 (2780)
7	Data 100	78 (2780)
8	Data 100	78 (3780)

or compatible terminals operating on point-to-point dial-up or dedicated communication lines. Each such terminal consists of a card reader and line printer, and can have an optional card punch. The following features of the 2780/3780 devices are supported:

Feature	2780	3780
Character set	EBCDIC	EBCDIC
Horizontal format control	No	No
EBCDIC transparency	Yes	Yes
Multiple record feature	Yes	Not applicable
Space compression/ expansion	Not applicable	Not applicable
Print width	80 through 150, inclusive	80 through 150, inclusive
Punch/component selection	Yes	Yes
Printer character set	EBCDIC-63	EBCDIC-63
Multipoint	No	No
Terminal ID	Accepted (not checked)	Accepted (not checked)
Conversational mode	Not used	Not used
Processor interrupt	Not used	Not used
Multiple cards in transparent mode	OK	OK

The operational procedures for submittal of remote batch jobs and return of printer or punch output are patterned after procedures for the Mode 4A 200 User Terminals. The interactive INTERCOM 5 commands are available to the terminal users, but must be entered on punch cards through the card reader; INTERCOM 5 interactive message responses and other unsolicited terminal operator messages appear on the line printer only at batch input and output file boundaries. Intermediate blocks of input and output files are not interleaved in the manner of Mode 4A terminal operations.

The TIP is insensitive to line speeds; it supports synchronous lines operating at rates up to 19.2K baud. Lines can be dedicated (with or without a transceiver) or switched (dial-up) with a modem. Auto-recognition, discriminating between HASP and 2780/3780 terminals, can be performed by the TIP when requested by the host. Auto-recognition causes the TIP to return a service message to the host which contains information on terminal type.

Each line can be configured with one cluster of devices serviced as separate batch and interactive terminals. This configuration recognizes up to five separate input and output streams for each terminal:

- interactive card reader input
- interactive line printer output
- batch card reader input
- batch card punch output
- batch line printer output

The TIP resolves which stream is active at any time, according to rules described in section 2. When contention exists between batch input and output streams, output takes precedence.

CHANGES REQUIRED TO WRITE A NONSTANDARD TIP

Any programmer desiring to write his own TIP should also use the CCI 3 system programmer's reference manual so that he can fully understand the functions CCI must provide to interface a terminal to INTERCOM 5. The State Programming Language Reference Manual provides a description of the macrocode necessary to write the microcode portion of the Terminal Interface Program.

CCI SOFTWARE LANGUAGES AND SUPPORT SOFTWARE

The CYBER Cross System, the primary support software for CCI, consists of the following programs:

- PASCAL compiler
- CLASS macro assembler
- Micro assembler
- MPLIB library maintenance program
- MPEDIT and MPLINK link editor programs

The support software operates in a host computer, and the principal output is the CCI load file. This file resides on host mass storage, and is available to load and initialize the NPU in the network.

CCI programs and routines are written in one of three types of source code:

- PASCAL language. This type of source code is used for most CCI programs. The PASCAL source language was chosen to simplify coding for the NPU microprocessor, which would otherwise use the relatively low-level CDC CYBER 18 series computer assembly language.

- Macro assembler code. A few frequently used routines and subroutines are coded in this assembly language. All of these routines are base system software or network communications software modules. TIPs do not use this source code.

- State programming language. A group of macro assembler macro commands has been defined as the state programming language. This special communications-oriented language is used by the TIP to convert message code and formats on the microprocessing level. Some TIPs also provide a set of upline state programs (input state programs) to demultiplex incoming data from the multiplex subsystem. Each TIP must also provide a set of downline state programs (called text processing programs) to convert data from interactive or batch data code and format to terminal code and format. In some cases, text processing is also necessary for a second stage of demultiplexing during upline processing.

PASCAL COMPLIER

The PASCAL language is defined in detail in the CYBER Cross System PASCAL reference manual. PASCAL is a high-level programming language patterned after ALGOL-60. The PASCAL user defines each task in statements that are processed by the compiler to yield a variable number of actual program instructions.

CLASS MACRO ASSEMBLER

The CLASS macro assembler is described in detail in the CYBER Cross System macro assembler reference manual. Here, the macro assembler functions are discussed in terms of the source program and the output.

CLASS Source

A source program consists of one or more subprograms. Each subprogram is a set of source statements. A source statement consists of location, instruction, address, comment, and sequence fields, respectively. Each subprogram can be assembled independently, or several can be assembled as a group. The main subprogram of a group is the one to which initial control is given; it need not be the first subprogram in the group. Communication between subprograms is accomplished by subprogram linkage pseudo instructions and by the use of common and data storage.

Pseudo instructions control the assembler, provide subprogram linkage, control output listing, reserve storage, and convert data. They can be placed anywhere in a source language subprogram.

A frequently used set of instructions can be grouped together to form a macro. Once a macro is defined, it can be used as a pseudo instruction. The CLASS assembler includes two types of macros:

- Programmer-defined - Macros declared and defined by MAC pseudo instructions. Each macro can be defined anywhere in the program prior to the first reference to it. Comment cards can be placed anywhere in the macro definition.

- Library - Definitions contained on the system library that can be called from any subprogram.

CLASS Output

There are two principal CLASS outputs:

- A relocatable binary output. The assembler outputs relocatable binary format records of variable length with a maximum of 120 characters from any peripheral device in the system. The driver for the input device verifies that the block is read correctly.

- Listed output. A number of listed output options are available, including mapping options. The assembly output listed by CLASS consists of descriptive information related to the source statement, followed by a listing of the source statement.

Three types of CYBER Cross System reference maps can be obtained:

- A complete Cross System reference map
- A short Cross System reference map
- A macro Cross System reference map

Macro Assembler Language

The macro assembler language is an assembly level language used by 255x series computers. A few base system software and network communications software modules have been written in this language to speed processing. These programs should not be modified by installations. All modules written in macro assembler language can be found in an assembly listing.

The macro assembler language allows definition of macros. A set of these macros has been predefined for CCI to form the state programming language.

State Programming Language

A set of macro assembler macro commands comprises the entire language. No other macro assembler code is included in state programs.

The state programming language is used for data code and format conversion. Standard interface routines between PASCAL coded (OPS level) programs and the state programs are provided. The state program source code can be found in an assembly listing for the appropriate TIP. The language and its use are described in the State Programming Language Reference Manual.

MICRO ASSEMBLER

The micro assembler is described in detail in the CYBER Cross System micro assembler reference manual. The assembler for the microprogrammable processors provides the mnemonic language necessary for the programmer to write a microprogram. The assembler translates symbolic source program instructions into object machine language instructions and provides a listing of assembly results. Micro assembler input and output includes:

- Input - A source input statement to the micro assembler consists of 11 fields. Of these fields, the location and comment fields are used to improve the documentation of the assembled micro instructions, while the eight remaining fields correspond to the eight fields of the micro instruction. Mnemonic instructions allow the programmer to use convenient names to specify the binary information to be inserted in each field of a macro instruction. Pseudo instructions fall into five classes: assembler control, listing control, memory management, data definition, and object code output pseudo instructions.

- Output - The output consists of an assembly listing including diagnostics (if errors occur), a zero location map, an origin location map, and a relocatable object image.

MPLIB LIBRARY MAINTENANCE PROGRAM

The MPLIB library maintenance program is described in detail in the CYBER Cross System link editor and library maintenance programs reference manual. The program is used to maintain a library of object programs as output by the PASCAL compiler, the macro assembler, and the micro assembler. The object program library is a sequential file; the first record in the library is a program name and entry point table for the programs in the library.

LINK EDITOR

The link editor is described in detail in the CYBER Cross System link editor and library maintenance programs reference manual.

The link editor accepts object text modules generated by the CYBER Cross System translators (PASCAL, macro assembler, and micro assembler) and builds a memory image load file. The link editor executes in two distinct phases, each of which is separably callable: the link phase (MPLINK) and the edit phase (MPEDIT).

The link phase inputs a set of directives which establishes the object text modules to be linked and directs their mapping into memory. The memory image load file developed by the link phase is derived from user supplied object text libraries. The user specifies the library files, as well as the modules to be selected, with the link phase directives.

The edit phase provides the ability to introduce specialized data (for example, variable field initial values, or system dependent parameters) into the formatted memory image load file. The values introduced are made available via assignment statements which possess a syntax similar to that available with the PASCAL programming language.

The edit phase produces an edited form of the memory image load file. The load file produced by the link phase is input to the edit phase where the user introduces initial values, system dependent parameters, and so forth. The mechanism by which initial values are introduced to the edit phase involves a set of input statements, which can be thought of as an edit phase language. The form of the edit phase language is comparable in many respects to the PASCAL programming language.

OVERVIEW OF CCI FUNCTIONS 2

The NPU, operating under control of CCI, provides multiplexing of terminal data for the host. The necessity for multiplexing has been explained in section 1, under the heading of Multiplexing Operation.

MULTIPLEXING, SWITCHING, AND DATA CONVERSION

On input (upline) transfers, the data from the various terminals is multiplexed, placed into interactive or batch data format, demultiplexed and gathered into message-size buffers, and passed to the host. On output (downline) transfer, the messages from the host are translated into terminal format, if requested, and are then multiplexed to the terminals, one character at a time.

The two major interfaces are buffered. On the host side, upline, full messages are placed in one or more (chained) buffers, the host is alerted that the message is ready, and the transfer takes place over the channel connecting the peripheral processing unit (PPU) of the host to the NPU. There is some communication between the host and the NPU to ready the transfer. The PPU receives the message in buffers. (The term message includes data messages, control information, and service messages that carry control and status information.) When the message is received by the host, and confirmed as valid data, the PPU passes the buffered data to INTERCOM 5 or mass storage (figure 2-1).

The downline transfer at the host interface is the converse of the operation described above. When the PPU has the complete message in a buffer, it notifies the NPU. If the NPU has sufficient buffers to start another message, it assigns the required number of data buffers and accepts a block of information until the entire message is transferred to NPU main memory. If the data is valid and translation is requested, the message is transformed to terminal format. This is accomplished by the TIP text processing routine, called by the internal processor.

Buffering also occurs at the terminal interface. All terminals are connected to the input and output multiplex loops via communication line adapters and communication lines. For an upline message, the data characters are picked from the input multiplex loop, passed to the circular input buffer along with data from all other active terminals. The message is then demultiplexed into one or more data buffers. When the message is completely assembled, the TIP and the internal processor transform the data as necessary. In most cases, this is a one-pass process. The HASP TIP, however, requires two stages of demultiplexing. The end product is a buffered message ready to be passed through the host interface to INTERCOM 5.

NOTE

> In this highly generalized discussion, it is assumed that all messages are being passed through the NPU from host to terminal or conversely. However, many control messages originate or terminate in the NPU itself.

Any buffering at the terminal is invisible to the NPU. The NPU need only assume that the line to the terminal can input or output data at the minimum rate for that line. If this does not occur, the line is identified as malfunctioning

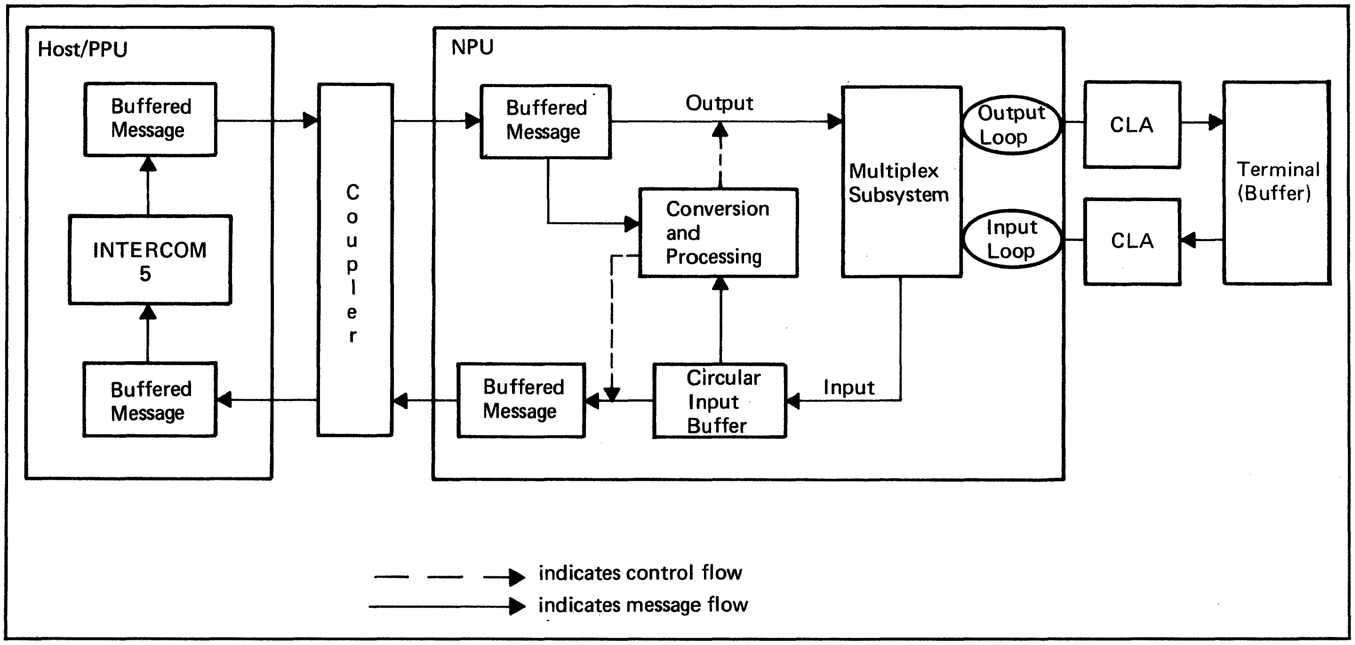


Figure 2-1. Simplified NPU Buffered Transfers

and the associated terminal is marked inoperative. The entire message associated with the line failure is held until the host tells the TIP to continue or discard the message.

For downline transfers at the terminal interface, the NPU first assembles the entire message in terminal format. When this is accomplished and the preparation for output to the terminal is completed, the multiplex subsystem picks the message, one character at a time, from the message output data buffer, formats it, and places it on the output loop. The communication line adapter for the line is responsible for recognizing its own data, passing it to the receiving terminal, and requesting more data until all data is sent.

Buffer space management is an important task for the NPU. Release of buffers occurs as shown in table 2-1.

The upline switching operation performed by the NPU is relatively simple in this CCI release, because each NPU is connected to only one host. For this release, the NPU acts as a message gatherer. If the host should fail, the NPU is responsible for delivering a message to the terminal indicating that the host is unavailable. When the host is reactivated, another message is sent to the terminals, notifying them that the system is again operational.

For downline switching, the NPU is responsible for checking the operational status of the terminal. There is no option to deliver a rejected message to an alternate terminal. However, since the host INTERCOM 5 program which originated the message is informed of the failure to communicate, it is possible for that program to route the message to an alternate terminal.

Regulation of input to an NPU can cause the unit attempting input to the NPU to have its message rejected. That unit (host or terminal) is responsible for inputting the rejected message at a later time.

Data conversion is provided for the convenience of the host. This conversion allows INTERCOM 5 to expect only two data formats: batch terminal and interactive terminal. The batch data format supports data compression. However, this is a passive data compression, and requires the terminal or INTERCOM 5 to compress the data and insert the proper data compression format effector characters in the data stream. CCI converts the compressed data into a format recognized by the receiving unit (host or terminal).

Conversion is always performed by CCI unless the message transfer expressly suppresses it. For the host, conversion is to ASCII or display code; for the terminal, conversion is to terminal format and code (External BCD, ASCII, or EBCDIC). A transparent mode (no conversion of any kind within the NPU) is also provided.

INTERFACES

Two interfaces have already been discussed in this section, the host interface and the terminal interface. There is one other interface, the interface to the NPU peripheral devices.

Each interface has its own protocol or set of protocols. It is the task of the NPU to convert data from the appropriate input protocol to the appropriate output protocol. Table 2-2 summarizes the protocols and their primary packets of data. The data packets, blocks, and special types of messages are discussed in more detail later in this section.

TRANSMISSION MEDIA

The basic transmission media for data within the NPU are blocks. Any amount of data, even one character, is collected into a data buffer, in block form, and held until the NPU delivers the entire message, aborts the message because the channel is inoperative, or acts on the data if the data is in fact a command or a request for information.

TABLE 2-1. BUFFER ASSIGNMENT AND RELEASE IN NPUs

Transfer Direction	Interface	Assignment	Release
Upline	Host	Buffers received from NPU internal processor or TIP.	When coupler acknowledges that the message block was correctly received, or when error occurs.
Downline	Host	Can postpone assignment based on lack of buffers; otherwise, assign buffers and receive the data.	For data messages: when next stage of NPU processing converts message to new format or when error occurs. For control messages or blocks: when NPU finishes action specified or when message is garbled.
Upline	Terminal	Can postpone assignment based on lack of buffers; otherwise, assign and chain as data is received.	For data messages:† when next stage of NPU processing converts message to new format or when error occurs. For control messages or blocks: when NPU finishes action specified or when message is garbled.
Downline	Terminal	Buffers received from NPU internal processor or TIP.	When terminal acknowledges that last block was correctly received or when message cannot be transmitted after several attempts (host notified in latter case).

†These are demultiplexed buffer blocks, after reconstituting data from the circular input buffer.

TABLE 2-2. INTERFACE AND PROTOCOL RELATIONSHIPS

Interface	Protocol	Operations/Responsible Modules	I/O Channel	Character Code and Format†
Host	Block	Data block is made out of (chained) buffers. Initialization technique at coupler is used to set up the transfer. TIPs handle conversion.	Direct Memory Access (DMA) (high speed)	ASCII or display code in interactive or batch data format
Terminal	TTY Mode 4C, 4A HASP 2780 3780	At NPU side, blocks are chained in data buffers. Polling is used for input, ODD is used for output, on a character-by-character transfer basis. A wide range of transfer rates is supported. TIP and internal processor handle conversion.	Multiplex interface, input or output loop	ASCII External BCD EBCDIC – in terminal format
NPU console	Internal CCI	On NPU side, blocks are chained in data buffers. Base modules handle transfers, also internal processors.	A/Q register interface, low speed	ASCII, terminal format

†Most transfers can also be in transparent data format.

In the following discussion, the initial operation to set up a channel or line for operation is omitted. This technique is discussed in detail in the HIP and base system software sections of the CCI 3 system programmer's reference manual.

There are several block types. They can be divided into two categories:

- Data transfer blocks - BLK blocks convey any part of the message but not the end. MSG blocks convey all the message or the end of a message. A typical large message requires several chained BLK blocks with a MSG block at the end of the chain.

- Control blocks - BACK blocks acknowledge receipt of other blocks and are used to control flow of data transfer blocks. CMD blocks transfer commands. CMD blocks are assigned data buffers.

The range of commands available in a CMD block is potentially very large; however, to establish a useful protocol, a set of commands called service messages has been defined. These messages transmit commands and status. In the host, the INTERCOM 5 PPU routines receive, generate, and process service messages. In the NPU, service message processing is handled by a service module. Service messages are summarized in appendix H.

Service messages ordinarily do not go through the NPU as such, unless they carry with them an ASCII message to be displayed at a terminal or at a console (host system console or NPU console).

A typical incoming service message solicits a response and causes the NPU to generate an appropriate response service message. Examples of this would be a configuration message answered by a response that the device specified has been configured as requested, or a status request answered by a message carrying the requested status. In some cases, the NPU generates an unsolicited response message. This occurs when the NPU has used the normal response form of a service message to generate a warning that the system is not operating as configured. For example, the unsolicited line status request response is a report that the line specified in the message is inoperative and further attempts to output messages to the terminal using this line will be rejected. Unsolicited response messages are also used by the NPU to report a normal line-operational condition when a dial-up line becomes active.

INITIALIZATION

Since the terminals connected through the NPU must be configured to match the host image of the network configuration, and since the NPU has no mass storage memory, the NPU image is kept on host mass storage and the NPU is loaded from the host. After the basic system is downline loaded, the host directs the NPU to configure its lines and terminals with a series of service messages. The steps of the initialization are as follows:

- Dump the NPU so that contents can be used for later analysis to determine cause of failure. This step is optional.

- Load the NPU from the host with CCI.

- Configure the NPU by establishing the parameters of the lines and terminals connected to this NPU.

The host normally attempts to load and dump an NPU only if the NPU fails or if the host system console operator specifically requests a load. If the host itself fails, the NPU must also be reloaded when the host comes back online. In the case of an NPU failure, the host (optionally) first dumps the NPU contents. The NPU is then loaded. If the first attempt to load the NPU fails, another dump is taken. After a set number of times, the NPU is marked down.

NPUs are loaded and dumped via the CYBER coupler under the control of the PPU. After the NPU is loaded, the host configures all lines in the course of establishing all logical connections for that NPU.

A logical connection is the association of two elements 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. These three numbers are used together to trace through a set of increasingly specific directories: the destination node directory, the source node directory, and the connection directory. This process ultimately points to a terminal control block (TCB). The directories also have information concerning the line control block (LCB). It is these control blocks that are the subject of the NPU configuration process.

The INTERCOM 5 PPU programs in the host are responsible for the control of connections in the network. All connections are explicitly configured and deleted by INTERCOM 5 using service messages. Configuration proceeds in stages:

- Configuring lines
- Configuring terminals on the lines

The loading operation establishes all logical links predefined within the load file. 1NI then configures the lines.

To connect the terminal, the line must be operational. When 1ND is informed of this condition, it issues a service message. The NPU then builds the terminal control block (TCB) for the terminal. When the configure action has been performed, the block protocol is initiated and the connection is in use.

BASE SYSTEM SOFTWARE

The base system software consists of most of those portions of the CCI that are normally associated with a comprehensive operating system. The base system software, the network communications software, and the interface programs (HIP and all TIPs) constitute all the standard CCI software. Base system software initialization and configuration was described previously. The major features of the base system software are as follows:

- System monitor services
- Buffer handling
- Worklist handling (indirect program calls)
- Queuing
- Direct program calls (page switching services)
- Interrupt handling
- Timing services
- Common areas
- Directory maintenance
- Standard subroutines for code translation, and special arithmetic functions
- NPU console support
- Multiplex subsystem operation including the command driver, the multiplex subsystem interfaces, worklist processing for the multiplex subsystem, multiplex subsystem timing services, CLA status handling, and the input and output data processors on the firmware level.

SYSTEM MONITOR

The NPU is an interrupt-driven processor. Interrupts are serviced according to a priority scheme in which all lower priority interrupts are disabled during execution of a program operating at a higher priority level. When no interrupt is in effect, the processor runs at its lowest priority, known as the operations system (OPS) monitor level.

Programs on the OPS level communicate with each other using worklists. Parameters for the requested task are stored in a worklist and the worklist is placed in a first in, first out queue for the program to be called. The monitor scans the list of all worklists capable of having worklist entries.

If the monitor scan discovers a worklist with one or more entries in its queue, control is passed to the program, together with the worklist that defines the parameters for the task. Control can be passed to a program for execution of more than one worklist, if that program is designated as being allowed to process more than one task before releasing control. In this case, the maximum number of queued tasks are passed in a worklist to the program and executed before the program returns control to the monitor. When the task or tasks are completed, control returns to the monitor, which resumes the scan at the next worklist.

Each time a program completes, a keep-alive timer is reset and checked by the interrupt level timer routine at specific system-defined intervals. If the timer expires, it indicates that some OPS-level program has been abnormally delayed. Monitor execution is then terminated and the NPU is stopped.

BUFFER HANDLING

This portion of the base system software allocates the four types of buffers and recovers buffers for the four buffer pools when users are finished with them. Buffers are potentially available in five sizes: 4, 8, 16, 32, and 64 words. At installation time, the user chooses any four contiguous sizes; for instance, 8, 16, 32, and 64 words. The largest size is designated as the data buffer size. Buffers are assigned one at a time; buffers can be released singly or in a chain.

In conjunction with testing buffer availability to assure a minimum threshold number, the buffer maintenance function periodically attempts to adjust distribution of buffer sizes by using buffer merging or buffer splitting to replenish any buffer pool that is at threshold level.

WORKLIST SERVICES

Worklists provide a convenient method to handle communications between software modules that do not use direct calls. The list services function manipulates worklists by making worklist entries from any priority level (including OPS level). These worklists have the following properties:

- Are first in, first out lists.
- Have one- to six-word entries, but all entries in any one list are of equal length.
- Exist in dynamically assigned space.

No limit is imposed on the number of lists serviced.

QUEUING MECHANISMS

Four major queues are defined:

- The terminal control block (TCB) queue controls input messages from the various terminals.
- The TCB output queue controls messages to be passed downline to the terminal.
- The NPU console queue controls messages to and from the NPU console.
- The timing queue.

Each of the first three queues is associated with a particular terminal control block that contains a pointer to the first segment in the queue. The timing queue is used for timing services.

DIRECT PROGRAM CALLS

Most OPS-level programs call other programs and subprograms directly. One subroutine is provided for these direct calls. The same subroutine also is the final step in the worklist calling sequence. This subroutine provides switching for programs on different main memory pages, timed and periodic calls, and overlay execution, as well as actual passing of control for programs called by the monitor.

INTERRUPT HANDLING

The NPU can recognize 16 different macrointerrupts. Each has its own address to which control is transferred when the interrupt is acknowledged. When the computer is processing a particular interrupt, it is defined as being in the interrupt state (state 00 through 15). However, before the computer can recognize an interrupt, the corresponding mask bit in the interrupt mask register must be set and the interrupt system must be activated.

Upon recognizing an interrupt, the hardware stores the appropriate program-return address to ensure that the software can return to the interrupted program after interrupt processing.

The interrupt handler that is activated also saves selected NPU registers for the interrupted program. The interrupt mask is then loaded with a mask to be used while in this interrupt state. The program then saves the current software priority level, sets the new software level, activates the interrupt system, and processes the interrupt.

During interrupt processing, an interrupt request with a higher priority can interrupt this program. Such higher level interrupts also store return address links and registers to permit sequential interrupt processing according to priority level, with eventual return to the mainstream computer program.

The computer exits from an interrupt state when processing is completed. The handler inhibits interrupts, restores the registers, and retrieves the return address of the interrupted program. Control is transferred to the return address and the interrupt system is again activated.

These micro interrupts are also serviced:

- The output data processor (ODP) services the output data demand (ODD) interrupt which each communication line adapter generates to indicate that it is ready to output another character. The ODP, 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 communication line adapter picks the character from the loop and transmits it.
- The input data processor (IDP) services the interrupt produced when the entry (data character or CLA status) into the circular input buffer is completed. The IDP, part of the multiplex subsystem, uses the designated input state program to demultiplex the character into the appropriate line-oriented input buffer.
- The timing services firmware processes the 3.3-millisecond clock interrupt which is used for the time base of all timed NPU functions.

TIMING SERVICES

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

- A firmware program handles the 3.3-millisecond micro interrupt to provide a 100-millisecond timing interval.
- Every 100 milliseconds, a timing routine searches a chain of time-lapse entries. If the time period for any entry elapses, that entry is deleted from the chain and a worklist entry is sent to the program for which the delayed call was requested. Timing services also add or delete delay requests to this delay request chain.
- Every 500 milliseconds, a timing routine checks the deadman timer. The timer is decremented and the monitor timeout routine is checked. If the monitor timer has expired, it indicates that the monitor has spent too long in one OPS-level program. If this occurs, the NPU is stopped.
- Every 100 milliseconds, a timing routine scans the list of active line control blocks (LCBs) for asynchronous terminals. If a character has been received, the timeout is reset for the next character. If the character has timed out (no character received within 100 milliseconds), the LCB is removed from the active list and the TTY TIP is notified via a worklist entry.

- Every second, a timing routine checks all active outputting lines to see if an output data demand (ODD) has been generated for the next character to output. If a second has expired with no new ODD interrupt, the multiplexer event worklist processor is notified via a worklist entry to declare a hardware failure for the line.

- Every 500 milliseconds, a timing routine scans all active lines for periodic requests. If the period for a specific request has elapsed, the TIP is called (using a worklist). Input or output can be terminated for the line if this is requested. Inactive LCBs are unchained from the set of active LCBs. Line timing services also provide chaining of LCBs to this list of LCBs requiring periodic action.

- A time-of-day routine is called every second. The time of day is incremented and, if necessary, recycled to start of day (00 hours, 00 minutes, 00 seconds).

GLOBALS

PASCAL coded programs can use global variables, tables, and constants. These globals are defined and described in the CCI 3 system programmer's reference manual.

STANDARD SUBROUTINES

This group of subroutines accomplishes the following operations:

- Converts and handles numbers.
- Handles interrupt masks.
- Maintains paging registers.
- Saves and restores registers.
- Converts code.
- Sets and clears the protect bit.
- Performs miscellaneous other tasks.

NPU CONSOLE SUPPORT

This group of base system software subroutines provides the equivalent of a Terminal Interface Program (TIP) for the NPU console. The console communicates with the NPU via the A/Q register interface, rather than through the multiplex subsystem interface.

MULTIPLEX SUBSYSTEM OPERATION

The multiplex subsystem has two principal tasks:

- Relieves the TIPs of needing to process lines according to individual line characteristics (most line characteristics are invisible to the TIPs as a result of multiplex subsystem processing).

- Multiplexes the data to match the low-speed characteristics of several individual terminals with the high-speed characteristics of the NPU and the host.

The principal multiplex subsystem functions are as follows:

- Receives serial data from communication lines, places it in a circular input buffer, and demultiplexes it into line-oriented input buffers.
- Transmits serial data to communication lines from line-oriented output buffers.
- Detects and processes special characters.
- Translates code on input.
- Checks cyclic redundancy count.
- Checks and generates character parity.
- Detects breaks.
- Assembles input data into a block.
- Controls modems and analyzes status.
- Chains data buffers as necessary to create data blocks.
- Processes logical commands issued by the Terminal Interface Programs.
- Provides input state instructions which allow the Terminal Interface Programs to dynamically alter input character processing as a function of the connected terminal and the data.

Figure 2-2 shows the basic multiplex subsystem elements.

Input Multiplexing

Each line has a communication line adapter (CLA). The CLA for each active line is sampled in sequence and if a character is ready, it is placed on the input multiplex loop together with information identifying the source (line) and, in some cases, the nature of the character byte (for instance, this byte can contain control information rather than a character of message data). All information on the input multiplex loop is routed to a circular input buffer, which is usually 512 words long. The demultiplexing operation picks data from this buffer and reconstitutes the messages on an input line basis.

Output Multiplexing

When the NPU has received a full message from the host and the appropriate Terminal Interface Program (TIP) has translated the code from ASCII or display code to terminal format, the multiplex subsystem is notified so that the message can be output. The multiplex subsystem picks characters from the line output message buffers one at a time, whenever a new output data demand (ODD) is generated by the communication line adapter. (The ODD indicates when the terminal is ready to receive a character.) The outgoing characters are placed on the output multiplex loop, along with any control characters needed and an address that will be recognized by the active line for that terminal. The CLA for the line recognizes the address, picks that data from the output loop, and marks it as being sent. When the contents of the entire output buffer for the line have been transmitted, the multiplex subsystem optionally notifies the TIP.

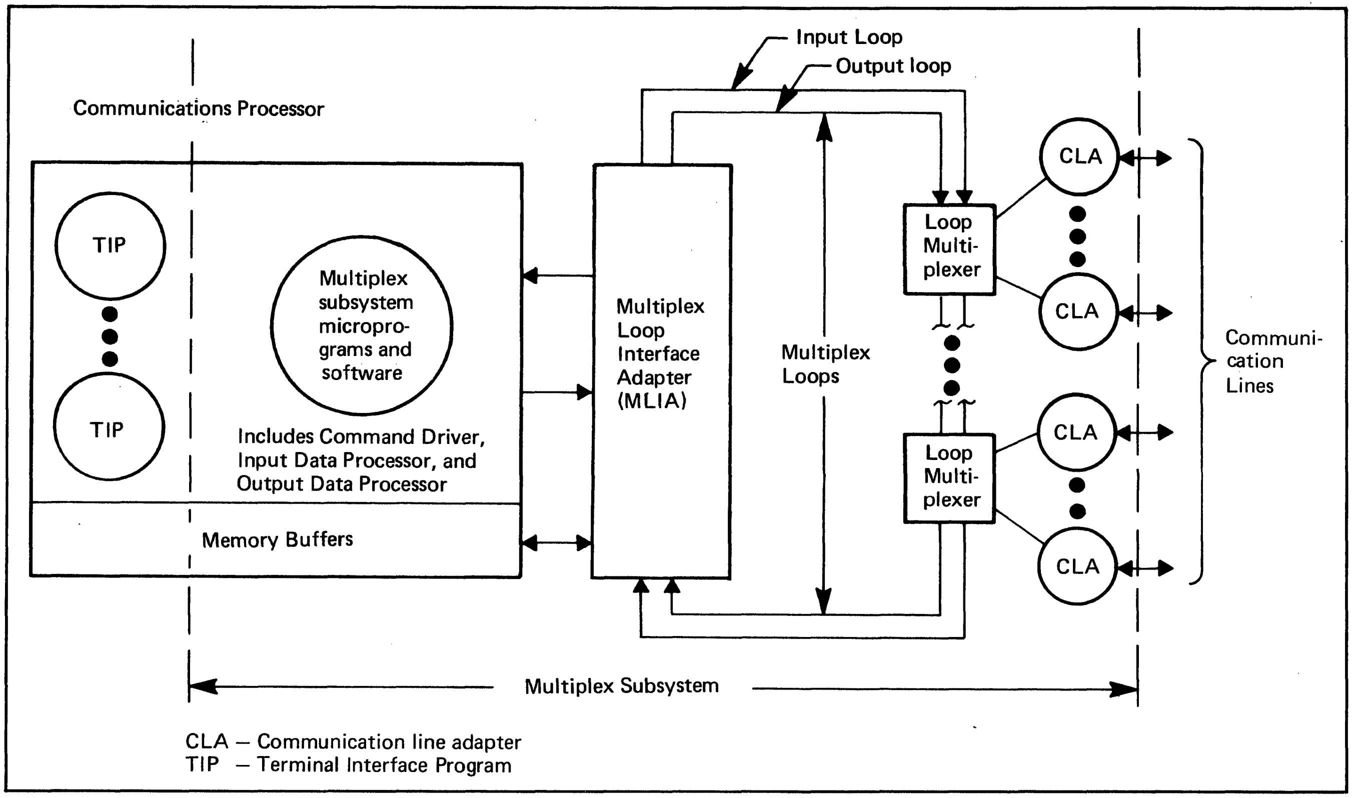

Input Loop
Output loop
Communications Processor
TIP
TIP
Multiplex subsystem micropro-grams and software
Includes Command Driver, Input Data Processor, and Output Data Processor
Memory Buffers
Multiplex Loop Interface Adapter (MLIA)
Multiplex Loops
Loop Multi-plexer
Loop Multi-plexer
CLA
CLA
CLA
CLA
Communi-cation Lines
Multiplex Subsystem
CLA – Communication line adapter
TIP – Terminal Interface Program

Figure 2-2. Basic Elements of the Multiplex Subsystem

Demultiplexing

The multiplex subsystem is responsible for picking data out of the circular input buffer as well as putting it into that buffer. When a message transmission starts, the multiplex subsystem reserves a data buffer for the line (data buffers are two characters per word). The words in the circular input buffer are identified by the communications line adapter address, and are placed into the data buffer for the appropriate line. If a buffer is filled before the message is complete, another buffer is assigned and is chained to the first.

When the end of text is detected, a worklist entry is made by the TIP input state programs to the TIP OPS-level routines. The TIP continues the processing (for instance, by sending the data to the post-input point-of-interface routine). When the message is ready for transmission to the host, the Host Interface Program (HIP) is called by the internal processor to pass the message through the coupler to the PPU of the host.

The demultiplexing of downline transfers is a terminal hardware and/or emulator function; that is, the message is reconstituted in a buffer for the screen, line printer, magnetic tape cassette, and so forth.

NETWORK COMMUNICATIONS SOFTWARE

The next level of CCI software is concerned with handling network communications. The major functions performed by this software are as follows:

- The internal processor handles routing of blocks for the CCI and maintains queues of input and output blocks (using the acknowledgement block portion of the CCI block protocol). The internal processor also assembles PRU blocks from input transmission blocks using a standard point-of-interface routine. The internal processor acknowledges PRU blocks to get more data if needed for batch connections.

- Standard point-of-interface programs provided for the TIPs have common code for pre and post input and output operations, and for locating terminal control blocks. Standard TIP support programs also generate blocks to reply to incoming blocks, to handle the text processing interface with the firmware, and to provide various other functions required by several TIPs.

- Failure and recovery routines handle error reporting and input regulation.

- System diagnostics generate CE error messages and, optionally, statistics messages for the host engineering file. Also, if the NPU stops, the diagnostics provide a reason code and other information concerning the stop. These functions are described in section 4 and appendix B.

- The service module handles service messages, which are described in appendix H.

Network communications software is discussed in detail in the CCI 3 system programmer's reference manual.

BLOCK ROUTING

The network communications software processing provides block switching so that the data block can ultimately be routed downline to the proper terminal, or upline to the host. This function is performed by the internal processor, using a point-of-interface routine. Data blocks traveling in either direction are identified by the network header bytes accompanying them. A group of directories are maintained which assure that the block control information (contained in the header) is decoded. The block is attached through the control blocks for that line and terminal to the next program that must process the block. The text processor for batch connections for the specified TIP is called to process the block in order to always have deliverable output ready.

SERVICE MODULE

The block format itself allows only a limited set of commands. Most of the large number of specific commands transmitted throughout the system are handled by the special type of command block called a service message. These messages are handled by the service module. The variety of service messages (one type for each command, with a related normal response and a related error response) is summarized in the CCI 3 system programmer's reference manual and is described in three separate sections of that manual:

- Messages for configuring lines and terminals, as well as the basic load and dump NPU service messages, are described in section 3, Initializing the NPU. These messages originate in the host for normal configuration and reconfiguration. Malfunctioning of a line or any component of the line generates an upline service message indicating that the line or component is no longer usable. INTERCOM 5 then reconfigures the network accordingly.

- Service messages regarding failure and recovery, and those that support online or inline diagnostics, are discussed in section 4, Failure, Recovery, and Diagnostics.

- The remaining service messages provide commands for such things as status, broadcasting a message to a terminal, or sending a message from a terminal to the host system operator's console. These messages are discussed in section 6, Network Communications Software. Section 6 also discusses the general operation of the service module.

STANDARD TIP SUBROUTINES

These subroutines belong to one of two classes: common TIP subroutines and point-of-interface (POI) routines. Both classes are discussed in the CCI 3 system programmer's reference manual. Four point-of-interface routines are provided:

- PBPIPOI – Post input POI. This routine accepts input from the TIPs and calls other routines to gather statistics, block data into PRUs, route blocks, and so forth.

- PBIOPOI – Internal output POI. This routine processes buffers according to the type of block in the buffer and queues the block to the TCB.

- PBPROPOI – Pre-output POI. This routine gets queued output blocks for the TIP.

- PBPOPOI – Post output POI. This routine generates statistics for the block, acknowledges the block to the host, and releases buffers for messages that the TIP has finished processing.

The common TIP subroutines perform the following operations:

- Queue output blocks. The TIP is notified that output needs to be processed.

- Handle a request from the TIP to stop output data from the host for a given line.

- Handle requests from a TIP to escape to firmware processing. This is used for the text processing operation.

- Gather statistics.

- Save and restore TIP processing entry points so that a TIP can temporarily suspend processing while waiting for some external event to occur. The TIP can be simultaneously suspended for one output and one input operation.

HOST INTERFACE PROGRAM

The Host Interface Program handles the protocol governing transmissions between the host and the NPU. The route of all such transmissions is through the coupler hardware. This hardware contains three registers which have status or command information and one register that contains the NPU address of the data to be transferred. One set of lines connects the host PPU buffer to the direct memory access buffer register of the NPU.

The size of a byte in the host is 12 bits and the associated half word (one byte) in the NPU is 8 bits. For address and data information, the 12-bit host byte does not directly use the upper four bits, making the host interface effectively an 8-bit byte. Two bytes are needed to make the associated 16-bit NPU word. For the three status type registers (coupler status, order word commands, and NPU status), only the lower 12 bits are used.

The NPU memory address register is set up by the NPU for upline or downline data transfer. The register is set up by the PPU when dumping the contents of the NPU to the host. In that case, the host uses the supplied address as the starting address of the next block of main memory to be dumped.

These principal functions are performed across the interface:

- The host issues NPU start and stop commands for the micromemory processor.

- The host loads programs into the NPU to initialize it. Dumping the contents of the NPU is usually a part of downline loading from the host.

- The host or NPU sends one word function commands to the coupler and checks status across the coupler. One of the status registers regulates transmission rate across the coupler by rejecting certain types of messages when the NPU is in danger of running low or running out of buffers.

- Blocks of data (messages) are transferred both upline and downline.

CONTROL WORD TRANSFERS

The NPU sets function commands to the coupler in four hexadecimal bytes, using one of the status type registers. This allows the NPU to chain buffers of data during transfer, clear the coupler registers, read the other two status type registers, ready the coupler to read the status type registers, and set the memory address register prior to starting a data transfer.

The PPU uses a 3-bit octal code to transmit functions. These commands clear the coupler or NPU, start the NPU, input or output a program during the load and dump phase of NPU initialization, load the memory address for dump operations, and set or read the other two status type registers.

STATUS WORD TRANSFERS

For status word transfers, the control word is set, indicating that one of the status type registers has been loaded and can be accessed by the unit on the other side of the coupler. The unit interprets the control word, reads the status, and acts on the status information.

The register used for regulation has three status values:

- Transmit all messages to the NPU. The register has this value when the buffer level is above threshold. Buffers can be assigned to receive data as rapidly as the host can transmit data.

- Do not transmit messages for batch type devices. The register has this value when buffer availability is critical. The heavy demands of chained buffers for batch type messages pose a potential hazard of exhausting the NPU data buffer supply, which causes an unconditional NPU halt. Normally, only interactive data is sent.

- Transmit only service messages or BACK blocks. The register has this value when neither batch nor interactive data transfers are allowed; the only transfers remaining are command or network coordination service messages. All service messages are short and can fit in a single data buffer of the largest size.

Note that all messages use the DMA channel of the NPU.

DATA TRANSFERS

For downline transfers, the NPU has assigned the next data buffer and has set up buffer chaining if necessary. The NPU switches the block to the internal processor after receiving the complete message.

For upline transfers, the full message (which may be in several chained blocks) is ready. The NPU makes the address of the first block in the chain available. As the blocks are transferred, the chain address is inspected. If nonzero when one block is transferred, the starting address of the buffer holding the next block is set in the address register. The transfer continues until the end of transfer bit is set in the current buffer.

Handling of the block protocol on both sides of the coupler causes the generation of acknowledgement service messages. This is not a HIP function on the NPU side.

Since the DMA/PPU buffer channel is half duplex (data can be sent in only one direction at a time), contention for channel use is normally resolved in favor of outputting blocks from the PPU. However, following this transfer, the protocol provides a 10-millisecond period during which the NPU can request channel use without contention from the PRU.

No attempt is made to resend transmissions of less than block length. A bad block is rejected in its entirety; with it, the message is rejected. For this reason an entire message is retransmitted regardless of the number of blocks composing it. The message handlers on both sides of the coupler are responsible for retaining messages until they are acknowledged as received.

HIP processing is discussed in detail in the CCI 3 system programmer's reference manual.

TERMINAL INTERFACE PROGRAMS

A Terminal Interface Program interfaces the terminal data (messages) to the network. The terminal interface is through the multiplex subsystem; the system interface is through the line control blocks (LCBs) and terminal control blocks (TCBs). There are four standard TIPs that can be included in a CCI load file:

- A synchronous one called the Mode 4 TIP after the protocol for this type of terminal

- An asynchronous one called the TTY TIP, for terminals using the CDC Mode 3 protocol

- A synchronous one called the HASP TIP for HASP multileaving terminals

- A synchronous one called the BISYNC TIP for terminals compatible with IBM 2780/3780 protocol

Each TIP has the general ability to handle the protocol for its terminal classes. Specialized additional information for any terminals that do not fit the basic TIP processing pattern is contained in the TCB for that terminal. This gives the standard TIPs sufficient flexibility to handle many terminal variations.

A TIP includes both hardware and software elements. In interfacing with the communications network, the principal concerns of the TIP are mode control and error control, with most of the software elements devoted to exception processing.

The multiplex interface to the TIPs is discussed in the CCI 3 system programmer's reference manual. The network interface to the TIPs has these principal portions:

- Output queuing. A common base system routine queues downline forward blocks, output data, and commands to the TCB. This routine discards the block unless the accept-output flag in the TCB is set.

- Upline commands. The common send-command subroutine indicates a discontinuity in the input or output streams. This routine resets the accept-output flag (preventing further queuing of output information), and sends an upline command message with a code indicating the reason for the interruption.

- Downline commands. The host commands the TIP to stop input by sending a downline command block. This block causes the accept-input flag to be reset. Blocks not acknowledged by the host will be resent by the TIP following a resume-input command. The method by which input is again accepted is a function of the particular TIP.

TTY TIP

The TTY TIP for asynchronous terminals provides a set of procedures for the interchange of interactive data between the host and terminals using CDC Mode 3 protocol. Mode 3 terminals are Teletypes or teletypewriter-compatible terminals.

The TTY TIP supports single-terminal 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, i.e., the TIP can be transmitting or receiving on a given line, but not both simultaneously. No code translation or parity check is performed on data characters input from the terminal. Data characters output to the terminal are output as received from the host with the exception of the character parity bit, which is complemented when necessary to make all output characters have even parity.

Characters transmitted between the host and terminal during both input and output are passed in ASCII coded 7-bit form, without code translation or parity generation and checking. Characters sent by the TTY TIP to the terminal always have even parity. An input message is sent in one or more blocks. The maximum size of an input block generated by the TIP is controlled by an installation parameter in the range up to 2039 characters. When the message length exceeds the maximum block size parameter, the message is divided into multiple blocks. The TTY TIP operates only in nontransparent (character) mode. It supports two input and output operation modes, the interactive mode, and the tape mode.

Input and Output Nontransparent Interactive Data Mode

In the interactive mode, the TIP interfaces the network to a teletypewriter device for either input or output. The interactive mode handles input from a TTY keyboard and also blocks of characters from a TTY compatible block mode device where the characters are received at line speed.

In the interactive mode, the TIP operates in half-duplex fashion with three basic states (idle, input, and output). The TIP is driven from the idle state to the input state by the arrival of a character which is not a carriage return, line feed or pad (FF_{16} or $7F_{16}$). If any of these three characters is received while in the idle state, the character is discarded and the TIP remains in the idle state (a single carriage return will cause a line feed echo with no further processing). Once in the input state, the TIP remains in that state until an end-of-input-message is received. The end-of-input-message for interactive mode is a carriage return. Line feed characters received from the terminal cause the TIP to output a carriage return if another character is not received within 100 milliseconds.

During input, carriage return characters are echoed with line feed characters when another character is not received from the terminal within 100 milliseconds.

The TIP is driven from the idle state to the output state by the arrival of an output message from the host. After the output message queue has been emptied, the TIP returns to the idle state. If an input character is received from the terminal when the TIP is in the output state, the current message being output is placed back on the top of the output queue, the character received from the terminal is discarded, and the TIP goes to the input state.

Input Nontransparent Tape Data Mode

In the tape mode, the TIP interfaces the network to a tape reader (paper tape or tape cassette). Tape mode only applies to input, because output is identical for all terminals. The TIP is commanded to enter the tape mode by a downline command from the host. This start input command causes the TIP to send an X-ON character (11_{16}) to the terminal which starts the tape reader. Message blocks are sent upline to the host when a carriage return character is received from the terminal. The X-OFF character must be the last data character of the message block. After the X-OFF is received, the TIP returns to the interactive mode. (Note that the X-OFF character is discarded.)

Auto-Recognition

The TIP performs auto-recognition of lines which have been configured as switched auto-baud by the host. Baud rates can be recognized at 110, 150, 300 and 1200 baud. 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. The result of this feature is a service message from the TIP informing the host of the line speed for that line.

The TTY TIP is described in detail in the CCI 3 system programmer's reference manual.

MODE 4 TIP

The Mode 4 TIP interfaces devices using Mode 4A or Mode 4C protocol to the network. A typical user would be the card reader, printer, keyboard, and CRT display of a CDC® 200 User Terminal.

The data contained within the interchanged blocks is converted to and from ASCII or display code, according to the terminal device type for that data. The interface to the terminals complies with the Mode 4A or Mode 4C standards; however, not all features of the Mode 4 protocols nor all features of supported terminals are used.

The TIP is insensitive to line speeds; it supports synchronous lines operating at rates up to 19.2K baud. Lines can be dedicated (with or without a transceiver) or switched (dial-up) with a modem. Lines are considered to be half duplex; that is, the TIP is either transmitting to the line or receiving from the line, but not simultaneously.

Each line can have more than one cluster of equipment and each equipment cluster can have more than one terminal. Lines with multiple clusters must be dedicated. Where multiple terminals are on a line, the TIP services each terminal in sequential order without priority.

The TIP performs auto-recognition when requested by the host. This procedure determines the code set of the terminal (ASCII or External BCD) and mode (Mode 4A or Mode 4C). Auto-recognition causes the TIP to return a service message to the host with the following information:

- Terminal class
- Cluster address
- Terminal address
- Device type

Multiple cluster auto-recognition is not supported.

The Mode 4 TIP supports remote batch terminals as separate but dependent devices. The dependencies are reported to the host on demand when a conflict occurs.

The TIP polls the terminal to determine when data should be input (assuming the terminal has data to input). The host requests that input be accepted, but the TIP controls the actual polling for data. A temporary pause in polling for input can occur if the NPU is regulating data input as a result of a low buffer availability condition.

The TIP performs recovery for line or terminal errors, using a slow-polling mechanism. Any error from which an immediate recovery is not possible is reported to the host.

The TIP processes each line as an independent data channel. Each terminal on a line is checked for work in the order the terminals were configured. This method allows each terminal to be processed in order without priority. The card reader and printer of the 200 User Terminal are treated as separate terminals in this scheme.

Nontransparent Input and Output Interactive Data

The interactive terminal interface to the Mode 4 TIP provides support to displays on synchronous lines. The configuration can be multicluster and each cluster can be multiterminal. The Mode 4 terminal display is supported by the interactive terminal interface; several additional features exist for control of the Mode 4A card reader and printer.

The terminals are activated either by delivery of an output message or a start-input command. When a start-input command is issued to a Mode 4A terminal, the cursor is moved to the leftmost character position. This command also clears the 200 User Terminal transmission buffer of any previous card or print block.

Polling for input continues until the terminal is deleted from the system configuration, an error occurs, buffer regulation occurs, logical link regulation occurs, or a stop-input command is received.

An input-stopped or output-stopped command is sent whenever a communication error is detected.

For the 200 User Terminal, the use of the display stops the use of the card reader and printer. When the display is finished with use of the line, the host is informed that the card reader and printer can again be used.

Nontransparent Input Batch Data

The Mode 4A card reader is activated by sending the TIP a command to start accepting input. The TIP sends card reader data, in display code, to the host. Each block of data is filled and sent to the host as a full PRU record until a 7/8/9 card or a 6/7/8/9 card is detected. That block is marked as the last block of the current message. Subsequent 6/7/8/9 cards or blank cards are discarded. The data following a 6/7/8/9 card is considered part of the next message. (It is possible that a single block from a Mode 4 device will contain more than one job. Each job is formatted in a separate message block.)

Input-stopped data is sent following the last data from the transmission block. No further input is accepted until a start-input command is sent to the NPU from INTERCOM 5. Input-stopped information is also sent if no further cards are present in the input hopper. A reason code is provided to distinguish between the two cases.

An upline command indicates that downline data or commands must not be sent or must be repeated if already sent and not acknowledged.

An input-stopped command message is also generated when the TIP detects a communication error with the terminal. A subsequent start-input message is generated when the error is resolved.

Nontransparent Output Batch Data

The Mode 4A printer can be activated by the host sending downline data to the printer. The printer connection is considered active until a last message block is sent by the host or until the display is used. The TIP converts the data from the internal batch terminal format to printer format. Each block when delivered correctly to the terminal is acknowledged.

A stop message is sent for the printer whenever data or commands cannot be processed because the display is in use. This stop occurs either when the host sends data to the display or when the remote operator interrupts a batch operation. The host must prepare to resend any data or commands not properly acknowledged. A stop message also occurs whenever an unrecoverable error is detected on the device.

A last message block is sent to the host whenever the printer is found to be not ready while the TIP is attempting to output data.

HASP TIP

The HASP TIP provides network interfacing to a HASP multileaving type of terminal which can contain both interactive and batch devices. The term multileaving describes the interleaved communications technique used by a HASP terminal. The HASP protocol uses fully synchronized, pseudo-simultaneous, bidirectional transmission of a variable number of data streams and requires binary synchronous communication facilities.

The basic element of multileaved stream transmission is a character string which is embedded in a data transmission block. One or more character strings are formed from the smallest external element of transmission – the physical record. The physical records that serve as input data can be of any normal record types (card images, printed lines, mass storage records, and so forth). For efficiency in transmission, each record is reduced to a series of character strings of two basic types: a variable length nonidentical series of characters or a variable number of identical characters. Since blanks appear frequently, a special case of the identical character string is the string of blanks.

A string control byte (SCB) precedes each character string to identify the type and length of the string. A nonduplicate character string is represented by an SCB followed by the nonduplicate characters. A consecutive, duplicate, nonblank character string is represented by an SCB (which contains the character count) and a single character. For an all-blank character string, only the SCB itself is required.

The transmitting program segments the data record to be transmitted into an optimum number of character strings, a number determined to take full advantage of the identical character compression. A special SCB indicates the grouping of character strings composing the original physical record. The receiving program then reconstructs the original record for processing.

In order to allow multiple physical records of various types to be grouped together in a single transmission block, a record control byte (RCB) precedes the group of character strings representing the original physical record. The RCB identifies the general type and function of the physical record (input stream, print stream, data set, and so forth). A particular RCB type is designated to allow the passage of control information between the various systems. To provide for simultaneous transmission of similar functions (multiple input streams), a stream identification code is included in the RCB. A subrecord control byte (SRCB) is also included immediately following the RCB. The SRCB supplies additional information concerning the record to the receiving program. (For example, if the transmitted data is to be printed, the SRCB can hold carriage control information.)

For a multileaving transmission, a variable number of records can be combined into a variable transmission block size; for instance:

RCB,SRCB,SCB1,SCB2,. . .SCBn,RCB,SRCB,SCB1,. . .

Multileaving allows an NPU and a HASP workstation to exchange transmission blocks containing multiple data streams in an interleaved fashion. For optimum use of this capability, a system must be able to control the flow of one data stream while continuing normal transmission of others. This requirement is obvious in the case of simultaneous transmission of two data streams to a system for immediate transcription to physical input/output devices with differing speeds, such as two print streams. To meter the flow of individual data streams, a function control sequence (FCS) is added to each block. The FCS is a sequence of bits, each one representing a particular transmission stream. The receiver of several data streams can temporarily stop the transmission of a particular stream by setting the corresponding FCS bit off in the next transmission to the sender of that stream. The stream can subsequently be resumed by setting the bit on. In this release, multileaving is used only by the HASP terminal. The host does not attempt to regulate device data; instead, host output is regulated by the use of FCS bits.

For error detection and correction purposes, a block control byte (BCB) is added as the first character of each block transmitted. The BCB contains control information and a block sequence count. This count is maintained and verified both by the sending and by the receiving systems to control lost or duplicated transmission blocks.

In addition to the normal binary synchronous text control characters (STX, ETB, and so forth), multileaving uses two acknowledgement signals, ACK0 and NAK. ACK0 is the normal block-received acknowledgement and it is also used as a filler by all systems to maintain communications when data is not available for transmission. NAK is used as the negative response; it indicates that the previous transmission was not successfully received.

Information blocks are of two types:

- Control transmission blocks that contain control information and an SCB.
- Data transmission blocks that contain data and the other control bytes described previously.

Input Nontransparent Data Mode

When in the nontransparent data mode, characters received are expanded from the HASP compressed format, translated to display code and stored in the standard PRU block format. Trailing blanks on each card are not stored and the end of each card is marked within the PRU block with from two to eleven zero characters (Modulo ten characters for the entire PRU block, but must have at least two).

Input Transparent Data Mode

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

Output Nontransparent Data Mode

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

Each subsequent PRU block is converted to output transmission blocks according to the BSC point-to-point and the HASP multileaving protocol. This protocol is described later in this section.

Data within the PRU block is considered print lines. The end of each line is detected according to the standard PRU block format.

If the print line taken from the PRU block is greater than the printer line width configured, the 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). Optionally, if the stream is configured for suppression of carriage control, the first character of each line is ignored and replaced by a single space before print.

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

Output Transparent Data Mode

For PRU blocks marked as transparent data, no print lines are detected within the block. 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 PRU block. Otherwise, transmission blocks are filled to the maximum size configured by the installation (either 400 or 800 bytes).

Protocol Operation

The terminal software is loaded and the communications line is initialized. After the sign-on command is transmitted, the NPU and the terminal transmit ACK0 blocks until a function is desired.

When a function other than a console message or console command is desired, the process initiating the function transmits a request-to-initiate-function-transmission RCB. The receiving process transmits a permission-to-initiate-function-transmission RCB if the data from the requesting process can be processed. If the data cannot be processed, or the function is now in process, the request-to-initiate-a-function-transmission RCB is ignored.

When a permission-to-initiate-a-function-transmission RCB is received, the requesting process begins transmitting data blocks to the other process. Data blocks can be transmitted until an EOF indicator is encountered. If more data blocks on the same device stream are to be transmitted following an end-of-file indicator, the request-to-initiate-a-function-transmission RCB sequence must be reinitiated (unless the device is a card reader). If a request-to-initiate-a-function-transmission RCB is not received before data blocks are received, the data blocks are ignored.

Data transmission blocks are transmitted one at a time. Before another block can be transmitted, the receiving process must transmit a positive response in the form of an acknowledge control block or a data block.

Console functions (operator messages and commands) do not have to follow the request-to-initiate/permission-to-initiate sequence. A console function can be initialized at any time that the wait flag in the FCS is not set and the remote console flag is set.

Control Transmission Blocks

Four types of control transmission blocks are used in the multileaving protocol:

- Acknowledge blocks which contain synchronization control, data link escape control, and positive acknowledgement information
- Negative acknowledge blocks which contain synchronization control and negative acknowledgement information
- Enquiry blocks which contain synchronization control, start of header, and enquiry control information
- Idle blocks that maintain communications. An idle block is transmitted at least once every two seconds in the absence of data transmission.

Data Transmission Blocks

In addition to the control bytes described earlier, data transmission blocks contain synchronization, escape, start of header or text, end of block transmission, and cyclic redundancy check information.

Special short blocks are defined for the following:

- Operator console blocks that deliver console messages in addition to other control information
- EOF indicator blocks (/*EOI cards)
- FCS mode change blocks (for instance, a low-speed printer has all the information it can currently handle as input)
- Sign-on blocks
- BCB error blocks

Error Handling

The following errors are recognized:

- Cyclic redundancy check errors
- Illegal block format
- Unknown responses
- Timeout over the line
- BCB recognized errors (break in the sequence of transmitted blocks)

For bad downline data, the TIP attempts to retransmit the block three times. On the fourth failure, the TIP forces a line inoperative status on the terminal. For upline data, the TIP will attempt to receive a bad block four times. On the fourth failure, the TIP forces a line inoperative status on the terminal.

Data Conversion and Compression

HASP terminals use EBCDIC code; host programs use ASCII code or display code in interactive or batch terminal format, as required. Therefore, upline code conversions are performed by the HASP TIP before the internal processor transforms the data into interactive or PRU block format. The HASP TIP converts downline data from ASCII or display code to EBCDIC when called by the internal processor during conversion of interactive data or batch blocks to terminal transmission blocks. Note that both these conversions can be bypassed when data is specified to remain in transparent form.

Because data can come from the batch terminal devices in compressed format (as directed by the SCB), the HASP TIP can translate to and from compressed data format.

HASP Console

The HASP console data is handled in interactive data format. A compression conversion is provided by the HASP TIP for any compressed data.

The HASP TIP is described in detail in the CCI 3 system programmer's reference manual.

2780/3780 TIP

The 2780/3780 TIP provides support of the standard INTERCOM remote batch features and commands, with minor extensions to support 2780 and 3780 terminals as described in the INTERCOM 5 Reference Manual. For 2780 and 3780 terminal batch input and output data, transfers are one direction at a time for the entire file transfer. That is, intermediate blocks of input and output files are not interleaved to or from the terminal.

There are four or, optionally, five different streams that must be managed to or from the two (or three) devices on a terminal. The three possible devices are card reader, line printer, and card punch; the five possible streams are interactive card input, interactive printer output, batch card input, batch card output and batch punch output. The TIP must resolve which stream is active at any one time. The general rules the TIP uses to determine which stream is active are described in the following paragraphs.

After dial-in, the host outputs the INTERCOM login banner message on the interactive stream; this is delivered to the line printer. Input from the terminal is then accepted from the card reader and is treated as interactive input. Input is accepted immediately after the terminal is configured (after the login banner is output).

All input is treated as interactive input until a start input command block on the batch connection is received from INTERCOM. (INTERCOM generates the start batch input command block to the TIP immediately upon receipt of a READ or READ,filename command from the terminal.) After receipt of the start input command block, all subsequent input is assumed to be batch input until an end-of-transmission (EOT) code is received from the terminal; at this time, the TIP reverts to processing input as interactive input until the next start input command block is received.

All queued interactive output messages are delivered to the line printer immediately following the completion of any active batch input or output file or immediately following an EOT received on the interactive input stream. After output of EOT to the terminal, a 3-second delay is initiated by the NPU to allow input before starting the next output file.

Each terminal usually operates in nontransparent mode and (if the feature is supported by the terminal) can operate in transparent mode. The operational capabilities of the 2780 and the 3780 terminals differ somewhat. The operational characteristics of each terminal for transparent and nontransparent modes are described in the following subsections.

2780 Input Nontransparent Terminal Mode

Commands are entered one per card and 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 if only one command is placed in the reader at a time and the EOF toggle switch on the terminal is ON.

The last interactive INTERCOM command entered before a batch input file or job deck must be either a READ or a READ,filename. The job deck or input file can be stacked directly behind the READ or READ,filename 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 and has special meaning to the TIP. Batch input can be terminated in one of three ways:

- A /*EOI in columns 1 through 5
- An ETX in the last column of an EOR card
- Input of the last card with the EOF toggle switch ON

If ETX is used to terminate a job, another job cannot be stacked directly because the TIP treats the next input after an ETX or EOT as interactive input.

2780 Input Transparent Terminal Mode

Input transparent terminal mode should not be confused with the transparent mode data feature. If the 2780 terminal has the transparent mode option, data can be input with the transparent switch ON, but some operational difference from the nontransparent mode applies.

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

Each card input causes a full 80 characters to be transferred across the line and each card is transferred 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.

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, with the following exceptions:

- An ETX code cannot be punched in an INTERCOM command card or on the last card of an input job or file to terminate input.
- Each INTERCOM 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 one 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 some 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

The TIP provides another mode of batch stream input for both 2780 and 3780 terminals, where the data is sent to the host without translation by the TIP. This mode of input can be 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 TR parameter is included as part of the READ,filename INTERCOM command, or TR can be included in columns 79 and 80 of either the INTERCOM job card, /*EOR card, or 7/8/9 multipunch card for normal input jobs.

When TR is specified, all data following is not translated and is stored as 8-bit characters. No /*EOI card is recognized when in this mode. Therefore, a file must be terminated using the EOF toggle switch.

Transparent 8-bit characters are stored in the PRU type blocks without marking any record (80-column card) boundaries. A transparent transmission block from the terminal can contain single records that are terminated by DLE ETB, or multiple records with each record separated by a DLE ITB (DLE ITB and DLE ETB are not stored in the PRU type block). 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 and, if used, results in 320 characters stored in the PRU type block for each transmission block received from the terminal.

In any case, characters are stored as received in internal format PRU type blocks without regard to record or transmission block boundaries. Transmission blocks are split across PRU block boundaries and input is stored until ETX is received.

2780 Output Nontransparent Transmission Mode

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

The TIP accepts PRU type blocks from the host, converts the data from display code to external EBCDIC code, and formats print lines into terminal protocol transmission blocks for output. Each transmission block is made up of multiple print lines (records) where the number of lines is either limited to two or can be up to seven, depending on the terminal option defined for the system. The transmission blocks cannot exceed 400 characters and a print line is never split across transmission block boundaries.

Transfer of an output data file to either the printer or punch normally continues until completion or a failure condition occurs. A method of interrupting an output stream is also provided to allow input of interactive commands that might be necessary to change the disposition of the output stream.

The intervention method used for interrupting output might differ operationally for 2780 and 3780 terminals and emulators of these terminals.

In general, making the printer not ready causes a timeout for some length of time (approximately 30 seconds), then interactive 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 can be duplicated. The timeout period is a function performed by the terminal itself and is usually extendable by pressing the printer stop key again.

An EOT received from the terminal as a response to an output block causes that output stream to be stopped. Interactive input from the card reader then is allowed.

Card punch output is processed by the TIP in a manner similar to print output, except the first character of each card (each terminal protocol record) is treated as data instead of carriage control and each record must be 80 characters or less in length. Characters within the PRU block record in excess of 80 are punched on the next card.

2780 Output Transparent Transmission Mode

Output to a terminal is normally transmitted as nontransparent mode 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 a NOS/BE ROUTE command. INTERCOM marks the data file as transparent in the data block clarifier field of each PRU block of the file.

The TIP processing of transparent output blocks is different from the processing of nontransparent blocks. PRU block characters are output as transmission blocks without code conversion and carriage control transforms. Characters are taken from the PRU block to make up transmission blocks without regard to record markers (FF_{16} character) or the PRU block boundary. Characters are transferred from the PRU block to the transmission block until the transmission block has reached its defined maximum or the last character of an EOR or EOI PRU block has been transferred. An EOI PRU block terminates the file. An EOR PRU block terminates only the transmission block. The next transmission to the terminal continues with the next PRU 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. There are several internal processing differences from those described for the 2780.

The number of print line records or card records is not limited to two or seven. The number of records included in a transmission block is limited only by the transmission block size (which is normally 512 characters), but can be configured to any size. No record can be split across a transmission block boundary.

Output to the 3780 follows the rules for data compression for that terminal; multiple blanks are compressed. Short records (print lines or cards) are terminated by the EBCDIC IRS character, and trailing blanks are transmitted if contained in the PRU block record.

MESSAGE PRIORITIES AND INPUT REGULATION

Messages are defined to be high or low priority as a function of the type of terminal which handles the message.

Regulation of messages is provided because all messages require buffer space, and it is possible that combined peak load demands from terminals and from the host might require more buffers that the NPU can assign. To prevent NPU stoppage due to running out of buffers, the NPU is allowed to reject input messages on the basis of priority. As current messages are processed and output, more buffers become available and the regulation level can change to accept message types previously rejected. Two types of regulation are provided:

- Input message regulation. Varying criteria apply according to the message source: host or terminal.

- Logical link regulation. Varying criteria apply according to traffic direction: upline or downline.

MESSAGE PRIORITIES

Three levels of message priority are defined for CCI:

- Service message traffic and BACK blocks (priority 0). This level handles much of the node to node command, reply, and status information.

- High level (priority 1). This level is intended for messages associated with interactive terminals. The size of the message is normally small, but the message should be quickly processed so that no processing delay is visible to the terminal user.

- Low level (priority 2). This level is intended for messages associated with batch terminals. The size of the message is large (often a thousand bytes or more), but since there is no operator interaction required, a small delay in message processing is acceptable.

The actual assignment of priorities to terminals is an initialization function of the host.

INPUT REGULATION

Four regulation levels are available to correspond to the three message priorities. Table 2-3 indicates the type of messages which are rejected at each level.

HOST INTERFACE REGULATION

Regulation applies to the high-speed DMA channel of the coupler. The PPU separates its output data into three queues:

- Service message queue (highest priority)

- High priority queue

- Low priority queue

TABLE 2-3. REGULATION LEVELS

Regulation Level	Mnemonic for State	Priorities Accepted	Remarks
3	NOREG	0, 1, 2	Sufficient buffers are available for all input. No regulation is necessary. This regulation applies to host and terminals.
2	STPLOW	0, 1	Buffers are low. NPU rejects all low priority message blocks. This type of regulation applies to host and terminals.
1	STPALL	0	Buffers are critically low. Both low and high priority message blocks are rejected, but service messages are accepted. This type of regulation applies to host and terminals.
0	ALERT	None	Buffers are so low that any input message blocks could cause the NPU to run out of buffers and therefore to stop. All input blocks are rejected. This type of regulation applies to host and terminals.

Prior to outputting data to the NPU across the coupler, the PPU sets the orderword which informs the NPU of the type and length of the next message. The NPU returns status information indicating acceptance or rejection of the message based on the NPU's current regulation state:

- NOREG – All host output messages are accepted.

- STPLOW – Messages from the low priority queue are rejected.

- STPALL – Messages from the low and high priority queues are rejected.

- ALERT – No messages are accepted from the host.

TERMINAL INTERFACE REGULATION

Prior to inviting new input from a terminal, the NPU examines its buffer state. For high priority terminals, STPALL and ALERT stop terminal input. For low priority terminals STPLOW, STPALL, and ALERT stop the terminal (if it is not already stopped). Either polling stops (Mode 4 terminals), request for input is denied (HASP, 2780, and 3780 terminals), or input will be discarded as long as the conditions for stop exist (TTY terminals). The TTY TIP generates the message

INPUT STOPPED

to the terminal operator. When the condition for stoppage disappears (NOREG for low priority, STPLOW or NOREG for high priority), input messages are again accepted. All interactive terminals that received the INPUT STOPPED message are sent a:

RESUME INPUT

message. During the period of stopped input, uncontrolled interactive terminals (serviced by the TTY TIP) can send input to the NPU. The NPU discards this input and sends the message

INPUT STOPPED

to the terminal.

Since NPU traffic load is variable, buffer availability can improve after the NPU sends some of its output data. This can lower the regulation one level so that previously stopped input can be restarted. This in turn can quickly lower the buffer availability, raise the regulation level, and therefore cause the terminals to stop again. Note however that the buffer state is examined only prior to inviting new input from a terminal. Because of this, heavy data traffic might cause oscillation of the buffer state, causing frequent start/stop terminal input sequences, especially for low priority data.

Since traffic, especially high-speed batch traffic, is capable of saturating an NPU faster than interactive traffic, low priority (which is regulated first), should be assigned to terminals featuring large blocks and/or high-speed capability.

Interactive terminals serviced by the TTY TIP, on the other hand, should have high priority assigned, because a high frequency of INPUT STOPPED and RESUME INPUT messages could have an irritating effect on the terminal user.

LOGICAL LINK REGULATION

Logical links are created in the CCI load file for any unique physical path between the host and the terminal node. (See figure 2-3.) Since logical link regulation generally affects data sources, there is a round trip delay until such regulation becomes effective. To compensate for this, logical link regulation occurs prior to NPU input regulation for a given regulation level, as shown in figure 2-4.

The need for logical link regulation is checked periodically, rather than as a function of a TIP checking input traffic prior to inviting new input traffic. See NPU Input Regulation, described previously.

Upline Data

Regulation for logical links is accomplished as a function of the origin of the problem. There are two possible origins and related actions:

- Host interface. The host usually does not regulate its input. The local NPU has a single output queue to the coupler. In rare circumstances, the host does not obtain the queued output as input.

- NPU buffers. The local NPU, which provides transient storage for the logical link data, regulates upline logical link data streams according to its own buffer availability.

The terminal node has two regulation levels that have to be examined prior to inviting new input from a terminal:

- The terminal node buffer state regulation level (affects all terminals on the NPU).

- The logical link regulation level (affects all terminals on that logical link).

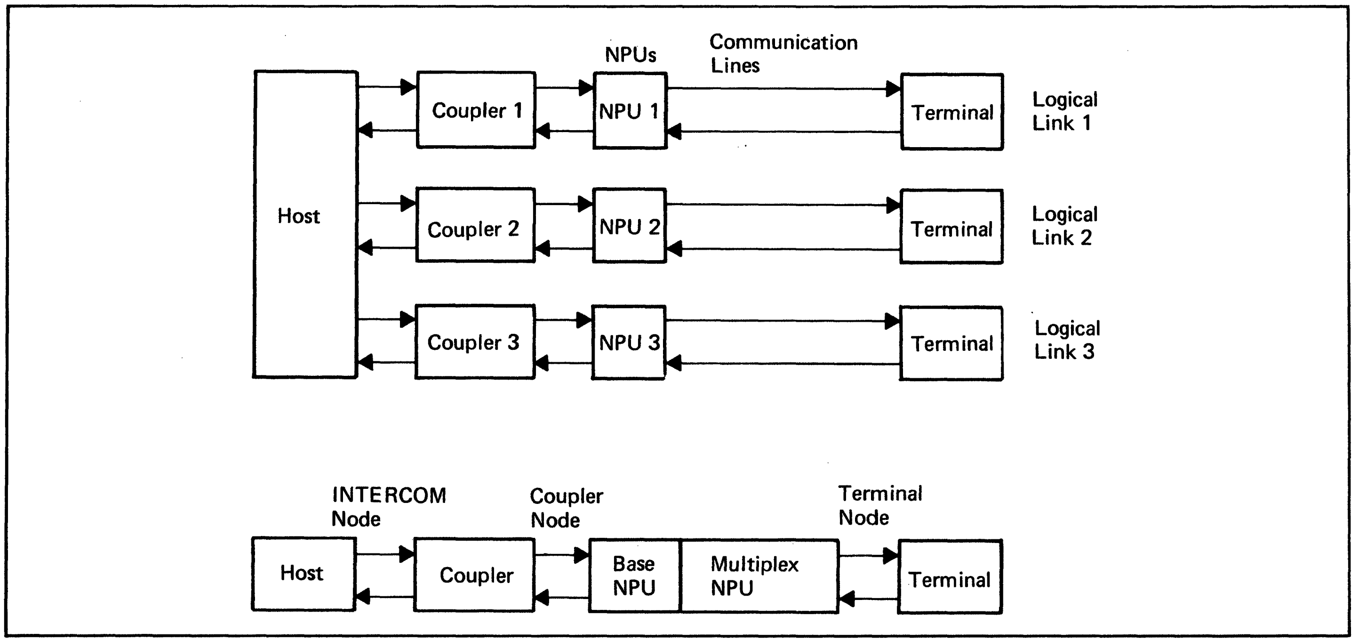


Figure 2-3. Sample Logical Link Connections

The minimum of both regulation levels is used to determine whether terminal input has to be stopped or restarted.

The ALERT level of logical link regulation indicates that a link to the host has been broken; in this case, the terminal node generates the message

HOST UNAVAILABLE

to all interactive terminals on that logical link. This normally occurs if the host does not service the HIP data channel for a period of 30 seconds.

The ALERT level for the TTY TIP terminal node buffer state prompts issuance of the additional message

INPUT STOPPED

which is probably a transient situation, whereas the message HOST UNAVAILABLE is a unique indication that either the host is down, or the link to the host has been broken.

Downline Data

As is the case with upline data, regulation action depends on the origin of the problem. There are two possible origins and related actions:

- Host interface. The host monitors the received logical link regulation level prior to output on that logical link. If STPALL regulation is in effect on that logical link, all output for that logical link is stopped. If a STPLOW regulation is in effect, low priority output is stopped by the host. All output is sent to the terminal node if a NOREG state exists. In the ALERT case, the logical link has failed and all logical connections are broken by the NPU.

- NPU buffers. The NPU converts a logical link regulation message to a logical link status service message. This service message is sent to the host.

The following logical link regulation levels exist in the NPU according to the terminal node buffer availability:

●	NOREG	–	no regulation
●	STPLOW	–	stop low priority data
●	STPALL	–	stop all data

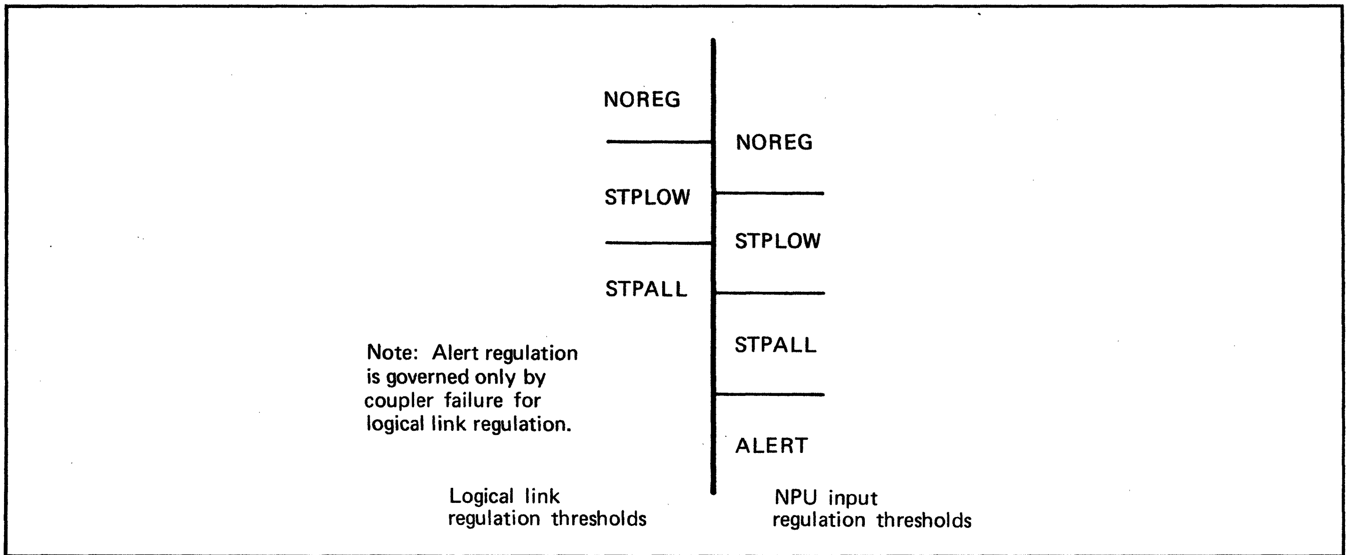

NOREG
STPLOW
STPALL
NOREG
STPLOW
STPALL
ALERT
Note: Alert regulation is governed only by coupler failure for logical link regulation.
Logical link regulation thresholds
NPU input regulation thresholds

Figure 2-4. Buffer Availability Threshold Levels for Regulation

INITIALIZING THE NPU

A three-step process makes each NPU into a fully operational network node:

- Dump the NPU to the host. This is an optional but usual procedure. The host system console operator can output the dump in a standard format for later analysis.
- Load the NPU from the host, including on-line diagnostics if these are a part of the system.
- Initialize the NPU by configuring the lines and terminals that have physical connections to this NPU.

The host normally attempts to load and dump an NPU only if the NPU fails or if the host system console operator specifically requests a load. (Note that the host itself might fail; the NPU is reloaded when the host comes back on line.) In the case of a failure or suspected failure, the host first dumps the NPU contents. The NPU is then loaded. If the first attempt to load the NPU fails, another dump is taken.

Failure of an NPU is always detected by the host PPU channel coupler driver. Upon detecting a failure condition, the NPU stops servicing the channel coupler. The PPU is then able to detect the NPU failure by a timeout of the protocol over the channel coupler.

The load and dump process varies with the type of the failed NPU, as described later in this section. NPUs are loaded and dumped via the host coupler, using a downline procedure.

NPU LOAD AND DUMP PHASES

Both load and dump operations for a 255x series NPU are multiphase (see table 3-1).

TABLE 3-1. LOAD/DUMP PHASES

Operation	2550/2551
Dump	1. Dumps main memory. 2. Loads main memory from host with small program to: • read file 1 registers • checksum the RAM 3. Dumps results of program to host.
Load	1. Loads RAM contents into main memory. Loads and executes programs to load RAM. 2. Loads main memory.

NPU LOADING

Any NPU connected to the host is loaded directly over a channel coupler by a PPU. Micromemory (RAM) is always loaded before main memory.

Micromemory cannot be directly loaded by the PPU; it is loaded only by a program executing in the NPU. The PPU loads a special micromemory loading program into the NPU main memory and causes the program to be executed. The NPU then loads its own micromemory and issues an idle response to the PPU.

Main memory is written directly by the PPU to the NPU. The NPU first specifies a start location by writing main memory addresses zero and one. The PPU then performs successive data transfers to the NPU, rereading each area and comparing it word-for-word to ensure a correct transfer. When loading is completed, the PPU issues a start-NPU function. The NPU executes the program just loaded and responds to the PPU with an idle response. If there is no idle response, the NPU has failed.

NPU operating programs and tables (all of CCI 3) are formatted into a load file that is resident in the mass storage of the host. To start NPU operation, that load file (containing both the main-memory-resident programs and writable micromemory-resident programs) is transferred (loaded) into the NPU.

The CCI load file contains all programs and tables except line control blocks and terminal control blocks. Since the load file defines a contiguous space in main memory, the maximum number of line control blocks that can be configured is fixed. Terminal control blocks, however, are built in dynamically acquired space and, therefore, are not similarly limited.

The format of a load file record includes a prefix made up of 15 60-bit words, a header that is a single 60-bit word, one or more blocks (each having a maximum of 120 16-bit words), and an end-of-record indicator. A complete description is given in the CYBER Cross System link editor reference manual.

NPU DUMPING

The NPU dumps consist of a main memory image, file 1 registers image, and a firmware (micromemory) checksum. The program which loads the file registers and makes the checksum is in the main memory at the time the latter two images are saved, so that program is also dumped. Coupler registers are saved by the host and are also dumped.

To transfer (dump) information from the NPU to the host, the host executes the following procedure:

1. The host reads the three coupler registers (coupler status register, NPU status register, and order word register) and retains these values for incorporation into the register dump record.

2. The host builds the dump header record (record 1) containing the channel and equipment number of the coupler, the date, and the time.

3. The host reads the entire NPU main memory (starting at address zero) and formats the data into blocks containing up to 120 16-bit words each, with the entire group of blocks thus constructed comprising the main memory dump record (record 2) of the dump file.

4. The host loads a dump bootstrap program into the NPU main memory starting at address zero and causes the program to be executed. This program overwrites a portion of the micromemory with a micromemory dump routine that generates a 16-bit micromemory checksum. The dump bootstrap program then copies the NPU file 1 registers, writes the value 8 into the NPU status register of the coupler (to indicate ready for dump), and halts.

5. The host again reads the NPU main memory and formats the register dump record.

The format of the NPU dump is shown in figure 3-1. Formats of various types of words used in the dump are shown in figure 3-2. All main memory words are four hexadecimal digits.

Record		Main memory address
1	Header	
2	Main memory	0000_{16} End of memory
	Dump bootstrap program	0000_{16}
	Number of words in second dump	$01FF_{16}$
	File 1 registers	0200_{16}
	Firmware checksum and coupler registers	0300_{16}

Figure 3-1. Format of an NPU Dump

CONFIGURING NPUs

After the NPU is loaded, INTERCOM 5 is informed of an NPU entering this active state by the arrival of an NPU-initialized service message. The host configures the unit by establishing all lines and logical connections for that NPU.

A logical connection is the association of two elements 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. The two node IDs represent the nodes at which each element 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 elements. Connection number zero is reserved as a permanent service channel for service message communications.

The set of logical connections which potentially exists between elements supported by a node pair is referred to as a logical link. A logical link must exist before logical connections can be assigned to it. (The service channel, which is designated as logical connection zero, is an exception to this rule.) Logical links are configured and therefore established as active when the CCI load file is built.

INTERCOM 5 in the host is responsible for the control of lines and logical connections in the network. All logical connections are explicitly configured and deleted by INTERCOM 5 use of NPU service messages. (See appendix H.) Note that the configuration is part of the general scheme of making a network operational. Deletion can occur at any time while the network is running. Both processes are described together in this section. Configuration proceeds in stages:

- Configures logical lines
- Configures terminals on the lines

INTERCOM 5 configures the lines and terminals using service messages in response to NPU service messages. Whenever a line is reported to INTERCOM 5 as operational, INTERCOM 5 configures and establishes connections for each terminal on the line.

To connect the terminal, the line must be operational. The terminal is connected by the NPU building a terminal control block (TCB) for the terminal. This process is initiated in the NPU when INTERCOM 5 dispatches a configure terminal service message. The message includes the connection number assigned by INTERCOM 5 for the connection. When the configure action has been performed, the block protocol is initiated and the connection is in use. INTERCOM 5 is informed of the successful completion of the configuration by a normal response.

Header record format

59	53	47	41	35	29	23	17	11	5 0
cn	en	b̸	h	h	.	n	n	.	s
s	b̸	m	m	/	d	d	/	y	y
b̸									

cn	binary channel number
en	binary equipment number of NPU coupler
b̸	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 digit 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

59	23	19	11	7 0
varies	0	uhw	0	lhw

uhw	upper half of word (character 1)
lhw	lower half of word (character 2)

Figure 3-2. Format of Words in Dumps

FAILURE, RECOVERY, AND DIAGNOSTIC AIDS 4

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

- Host failure – If a host fails, the NPU and its software must necessarily stop message processing. Also, the NPU sends an informative service message to all interactive connections informing the terminal that the host is unavailable.

- NPU failure – If an NPU fails, it must be reloaded and reinitiated from the host. Offline diagnostic tests may be desirable during this period to help identify the cause of failure. Failure is detected by the means of a 10-second timeout across the coupler. A manually initiated halt (section 5) is also interpreted as an NPU failure.

  Recovery consists of a dump (optional), load, and reconfigure operation. After n successive attempts to load (n is an installation parameter), the loading operation is aborted, and the NPU is ignored until manually reactivated. If the NPU is successfully loaded and initialized, INTERCOM 5 sets up all lines and terminals for that NPU which the present state of the network allows.

- Line failure – Lines are disconnected and terminal control blocks (TCBs) associated with the lines are deleted. A line failure is detected by an abnormal modem status or by line protocol failure. The change of status is reported to the host.

  A line cannot recover spontaneously. INTERCOM 5 deletes the supported TCBs. Then it disables and reenables the line, using the appropriate service messages. When the line status changes to operational and this is reported, INTERCOM 5 attempts to configure the supported terminals. If the line cannot be recovered, the host system console operator is notified; line recovery is attempted again after a long timeout (typically, 10 minutes).

- Terminal failure – Terminal status is reported. TCBs are not released. Once terminal failure has been detected, possible terminal recovery is monitored by a periodic status check or a slow diagnostic poll (Mode 4 only), made from the NPU to the terminal. Terminal recovery status is reported to INTERCOM 5.

To aid recovery and to assure dependable network operations that involve the CCI, three sets of diagnostic programs are available:

- Inline diagnostics. These include CE error messages, statistics messages, halt code messages that specify the reason for an NPU failure, and offline dumps. These messages are briefly described in appendix B.

- Optional online diagnostic tests that allow closed loop checking of circuits to terminals. These aids are available as a Terminal Interface Program only if a maintenance contract is purchased.

- Offline diagnostics. These hardware tests for NPU circuits are described in detail in the host communications processor hardware maintenance manual.

OPERATING PROCEDURES 5

Operating procedures for the CCI consist of loading and initializing the NPU, suspension of NPU operation, and entry of operator commands through the NPU console keyboard.

LOADING AND INITIALIZATION

The NPU is loaded and initialized by the host computer system. Therefore, few procedures associated with these functions are the concern of the NPU operator. However, to prepare for such a downline load, the NPU operator must perform the following functions:

1. Verify that numbers of ports (CLA thumbwheel addresses) for the communications network connections are correct.
2. On the loop multiplex circuit card, set the power (PWR) switch to ON. See figure 5-1.
3. On the CLA circuit cards, set CLA/OFF switches to CLA (on). See figure 5-2. Only those cards that are configured require this.
4. Verify the NPU console is in the normal ON condition.

Upon successful completion of the downline load operation by the host, a message containing the CCI version, host identification number, and NPU identification number is output at the NPU console. The format of that message is shown in figure 5-3. The host then configures the terminals, and normal system operation begins.

If the downline load is unsuccessful, the host initiates and receives a dump of the NPU memory, micromemory checksum, and file 1 registers. The initiation of another downline load attempt is under control of the host.

SUSPENSION OF OPERATION

To stop operation of the NPU for any reason, momentarily press the MASTER CLEAR switch on the maintenance panel. See figure 5-4.

NPU CONSOLE

The NPU console is primarily used for diagnostic control. Appendix B of this manual describes the standard message control mnemonics used with the inline diagnostics.

COMMAND ENTRY

Operator commands are entered through the NPU console keyboard. These statements specify either supervisory or diagnostic functions that can be selectively activated or deactivated.

The NPU console can be in either the read or write mode, selected by the Ⓖ key on the console keyboard. Pressing the Ⓖ key causes a manual interrupt that, in turn, causes the console to alternate between the read and write modes with the mode changing each time the key is pressed. To input operator commands, the console must be in the read mode; to output responses, the console must be in the write mode. All operator commands begin with a slash (/) character and terminate with an EOT (control D) character. Each parameter within the command is separated by either a comma or a blank character. Any number of commands can be entered before the write mode is activated to receive responses. When the write mode is activated, the following response is output at the console:

```
*WM
```

If an input error is made during entry of commands, the console response is an echo of the input message followed by

```
*ERR
```

CONSOLE COMMANDS

Seven function control commands can be entered from the NPU console. These function commands in turn permit entry of other data as commands to CCI.

SUPERVISORY FUNCTIONS

The console command

```
/SUP
```

causes the console to engage the supervisory function. While this function is active, the supervisory commands, which are for routing service messages, are as follows:

```
OUT, pfc, sfc, y Ⓓ

LOC, pfc, sfc, y Ⓓ
```

OUT designates a downline service message to be routed to the NPU console.

LOC designates an upline service message to be routed to the NPU console.

pfc is the value of the primary function code of the service message.

sfc is the value of the secondary function code of the service message. An sfc value of FF_{16} affects all service messages that have the pfc value given.

y specifies routing for messages of pfc and sfc values, as follows:

0 discards all messages.

1 prints all messages on the NPU console.

2 sends all messages to the host or service module.

3 sends all messages to the host or service module and also prints them on the NPU console.

Service messages to the console can cause system overload because of the excessive print time.

Ⓓ indicates the control D key on the console keyboard.

The system default is the supervisory mode with all upline service messages sent to the host and all downline service messages sent to the service module.

ORDERWIRE FUNCTIONS

The console command

/ORD

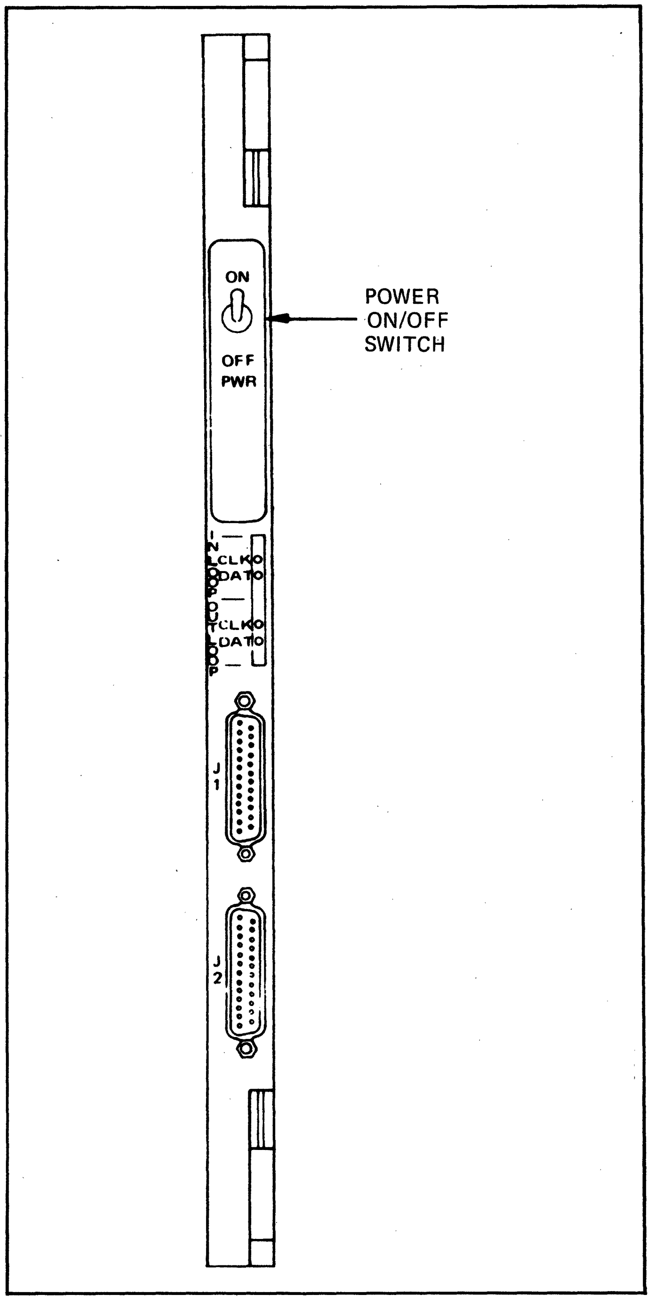


Figure 5-1. Loop Multiplex Circuit Card PWR ON/OFF Switch Location

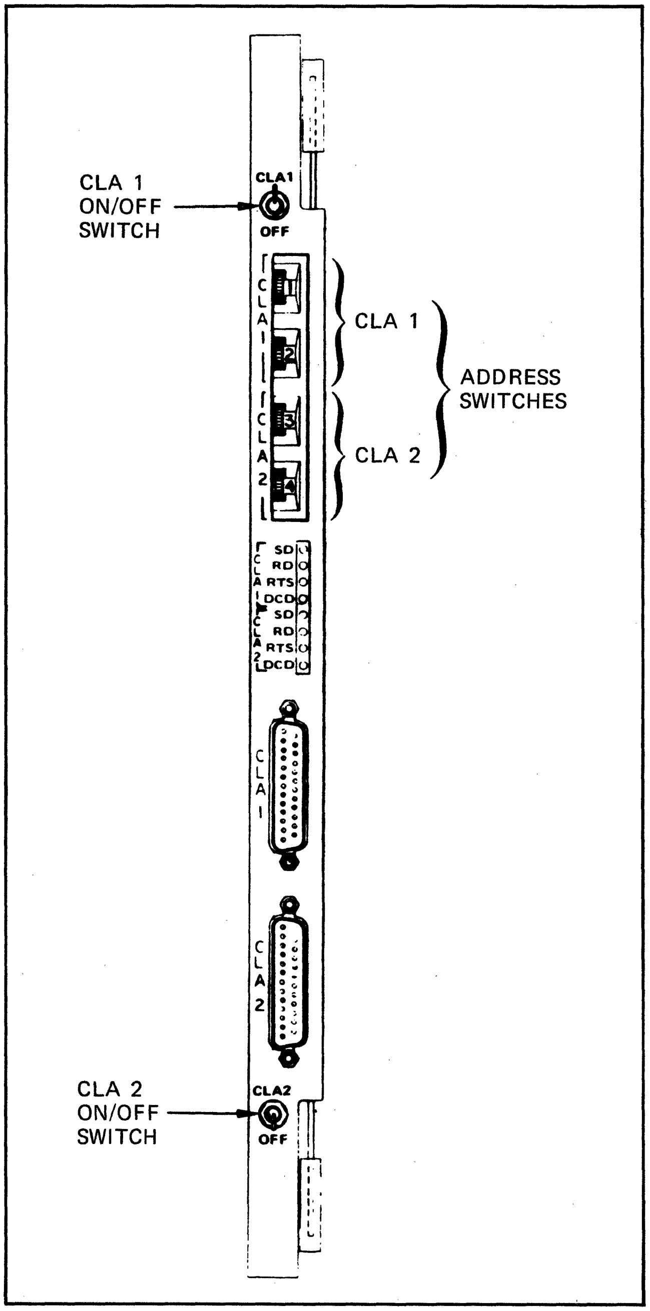


Figure 5-2. CLA Circuit Card ON/OFF Switch Location

```
CCI      3.0

HOST ID  : 0
NPU ID   : x
LEVEL    : yyyy
VARIANT  : abzz
```

x — The node identification number, $0 \leq x \leq FF_{16}$, identifying the source and destination node of all terminal control blocks for this NPU. The release of CCI assigns x a value of 2.

yyyy — The level identifier for the build version of CCI on which the copy in this NPU is based.

a — Identifies the type of NPU and memory access supported by the copy of CCI in this NPU; a can have the values:

0	2550 or 2551 local NPU, nonpaged memory
1	2550 or 2551 local NPU, paged memory
4	2552 local NPU, nonpaged memory
5	2552 local NPU, paged memory

b — The TIP combination number, identifying the TIPs included in the copy of CCI in this NPU; b is the hexadecimal sum of the values:

1	TTY TIP
2	Mode 4 TIP
4	2780/3780 TIP
8	HASP TIP

zz — The variant identifier, $01 \leq zz \leq FF_{16}$, identifying the build within the build version level for the copy of CCI in this NPU.

Figure 5-3. CCI Initialization Message

causes the console to engage the orderwire function. While the orderwire function is active, service messages can be entered as commands. These commands are described in appendix H. Although all input parameters for each command are shown as two hexadecimal characters, the leftmost character can be omitted if it is zero.

QUESTION IF SUPERVISORY FUNCTION

The command

```
/QIS
```

causes the current console function (supervisory or orderwire) to be printed at the console in the following format:

```
Q = xxx
```

where xxx is SUP or ORD.

The purpose of /QIS is to allow the operator to determine whether the console is in the supervisory or the orderwire mode.

ACTIVATE FUNCTION

The command

```
/ACT xxx
```

activates either the supervisory or orderwire function, as specified by xxx. Activating the function does not select it as the current console function, but only prepares the function so that it can be selected if desired.

DEACTIVATE FUNCTION

The command

```
/DEA xxx
```

deactivates either the supervisory or orderwire function, as specified by xxx. Deactivating a function means that it cannot be selected as a console function.

REQUEUE FUNCTION

The command

```
/REQ
```

causes requeuing of an NPU console output message that has been interrupted by a manual interrupt. The message is output the next time the console enters the write mode.

CANCEL FUNCTION

The command

```
/CAN
```

cancels a console output message that has been interrupted by a manual interrupt.

USE OF MANUAL INTERRUPT

The manual interrupt is caused by pressing the Ⓖ (control G) key on the console keyboard. This act causes the console to alternate between the read and write mode, with the mode changing each time the key is pressed.

If a manual interrupt occurs while output is in progress, the following applies:

- A manual interrupt followed by /REQ causes the current output message to be requeued.
- A manual interrupt followed by /CAN causes the current output message to be canceled and discarded.

- A manual interrupt followed by any input other than the foregoing causes the interrupted output message to continue printing after return to the write mode (from the point at which it was interrupted).

EDITING CONSOLE INPUT

The following console editing standards apply to all console input:

- Carriage returns (CR) and line feeds (LF) are ignored in that they are used as local characters only.

- Control shift N is replaced by CR.

- Control shift M is replaced by LF.

- Control C discards input. The response to a discarded input is the input message discarded followed by *ERR.

- Data can be overwritten by using the backspace ← (control H) key, with n backspaces causing n characters to be removed. Corrections can then be entered in place of the removed characters.

SYSTEM HALTS

When the CCI detects an unrecoverable error, the system immediately halts execution and prints a system halt message at the console. This message is described in appendix B.

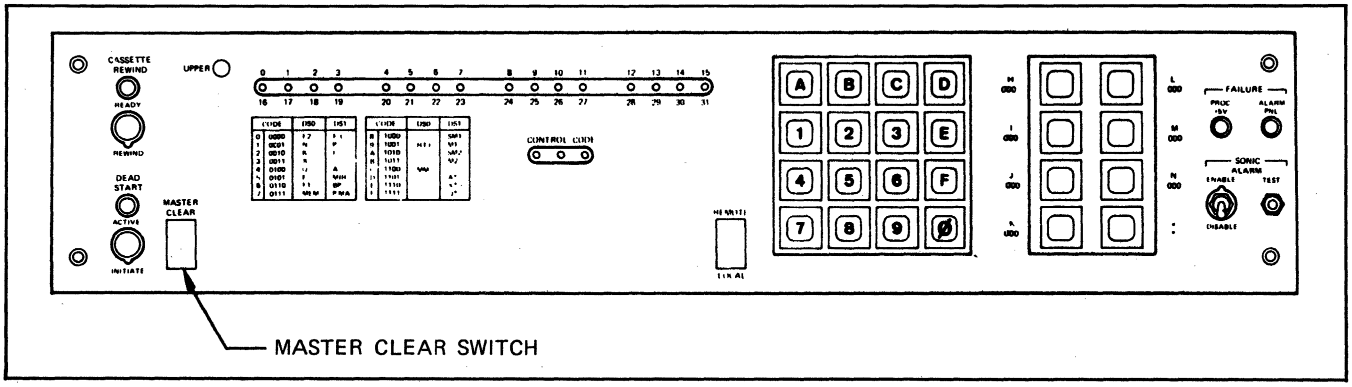


Figure 5-4. Maintenance Panel MASTER CLEAR Switch Location

STANDARD CHARACTER SETS A

OPERATING SYSTEM CHARACTER SETS

Control Data operating systems offer the following variations of a basic character set:

- CDC 64-character set
- CDC 63-character set
- ASCII 64-character set
- ASCII 63-character set

The set in use at a particular installation is specified when the operating system is installed or deadstarted.

Depending on another installation option, the system assumes an input deck has been punched in either 026 or 029 mode (regardless of the character set in use).

Under NOS/BE, alternate keypunch modes can be specified by a 26 or 29 punched in columns 79 and 80 of any job card or 7/8/9 card. The specified mode remains in effect throughout the job unless it is reset by specification of the alternate mode on a subsequent 7/8/9 card. Mode 4A terminals do not recognize this specification. HASP and 2780/3780 terminals recognize the 26 or 29 when punched in columns 79 and 80 of any job card or /*EOR card.

Graphic character representation appearing at a terminal or printer depends on the installation character set and the terminal class. Characters shown in the CDC Graphic column of the standard character set table (table A-1) are applicable to BCD terminals; ASCII graphic characters are applicable to ASCII CRT and ASCII TTY terminals. Tables A-1 through A-4 are provided for the reader's use while coding a program to run under the operating system. These tables do not describe character transmissions between an application program and the network. The external BCD code given in table A-1 is a 7-track magnetic tape code without parity, and is not the external BCD code used by some Mode 4A terminals; the mode 4A external BCD code is a 7-bit code, plus odd parity.

128-CHARACTER ASCII SET

Table A-5 contains the complete ASCII 7-bit (excluding parity) character set supported by CCI 3 for interactive terminal blocks. Binary designations are shown so that conversion of the leftmost two columns and rightmost two columns to octal or hexadecimal can be made; no octal display code equivalents appear for the characters of these four columns in table A-1.

During output operations to a terminal that does not support the full 128- or 96-character sets, conversion from the 96-character set to the 64-character set is accomplished by folding column 6 into column 4 and folding column 7 into column 5. Folding the 128-character set into the 96-character set consists of replacing the extra characters with blanks.

TABLE A-1. STANDARD CHARACTER SETS

Display Code (octal)	CDC			ASCII		
	Graphic	Hollerith Punch (026)	External BCD Code	Graphic Subset	Punch (029)	Code (octal)
00†	: (colon)††	8-2	00	: (colon)††	8-2	072
01	A	12-1	61	A	12-1	101
02	B	12-2	62	B	12-2	102
03	C	12-3	63	C	12-3	103
04	D	12-4	64	D	12-4	104
05	E	12-5	65	E	12-5	105
06	F	12-6	66	F	12-6	106
07	G	12-7	67	G	12-7	107
10	H	12-8	70	H	12-8	110
11	I	12-9	71	I	12-9	111
12	J	11-1	41	J	11-1	112
13	K	11-2	42	K	11-2	113
14	L	11-3	43	L	11-3	114
15	M	11-4	44	M	11-4	115
16	N	11-5	45	N	11-5	116
17	O	11-6	46	O	11-6	117
20	P	11-7	47	P	11-7	120
21	Q	11-8	50	Q	11-8	121
22	R	11-9	51	R	11-9	122
23	S	0-2	22	S	0-2	123
24	T	0-3	23	T	0-3	124
25	U	0-4	24	U	0-4	125
26	V	0-5	25	V	0-5	126
27	W	0-6	26	W	0-6	127
30	X	0-7	27	X	0-7	130
31	Y	0-8	30	Y	0-8	131
32	Z	0-9	31	Z	0-9	132
33	0	0	12	0	0	060
34	1	1	01	1	1	061
35	2	2	02	2	2	062
36	3	3	03	3	3	063
37	4	4	04	4	4	064
40	5	5	05	5	5	065
41	6	6	06	6	6	066
42	7	7	07	7	7	067
43	8	8	10	8	8	070
44	9	9	11	9	9	071
45	+	12	60	+	12-8-6	053
46	-	11	40	-	11	055
47	*	11-8-4	54	*	11-8-4	052
50	/	0-1	21	/	0-1	057
51	(	0-8-4	34	(	12-8-5	050
52	)	12-8-4	74	)	11-8-5	051
53	$	11-8-3	53	$	11-8-3	044
54	=	8-3	13	=	8-6	075
55	blank	no punch	20	blank	no punch	040
56	, (comma)	0-8-3	33	, (comma)	0-8-3	054
57	. (period)	12-8-3	73	. (period)	12-8-3	056
60	≡	0-8-6	36	#	8-3	043
61	[	8-7	17	[	12-8-2	133
62	]	0-8-2	32	]	11-8-2	135
63	% ††	8-6	16	% ††	0-8-4	045
64	≠	8-4	14	" (quote)	8-7	042
65	↱	0-8-5	35	_ (underline)	0-8-5	137
66	∨	11-0 or 11-8-2†††	52	!	12-8-7 or 11-0†††	041
67	∧	0-8-7	37	&	12	046
70	↑	11-8-5	55	' (apostrophe)	8-5	047
71	↓	11-8-6	56	?	0-8-7	077
72	<	12-0 or 12-8-2†††	72	<	12-8-4 or 12-0†††	074
73	>	11-8-7	57	>	0-8-6	076
74	≤	8-5	15	@	8-4	100
75	≥	12-8-5	75	\	0-8-2	134
76	¬	12-8-6	76	‾ (circumflex)	11-8-7	136
77	; (semicolon)	12-8-7	77	; (semicolon)	11-8-6	073

†Twelve zero bits at the end of a 60-bit word in a zero byte record are an end of recora mark rather than two colons.

††In installations using a 63-graphic set, display code 00 has no associated graphic or card code; display code 63 is the colon (8-2 punch). The % graphic and related card codes do not exist and translations yield a blank (55_8).

†††The alternate Hollerith (026) and ASCII (029) punches are accepted for input only.

TABLE A-2. AMERICAN NATIONAL STANDARD CODE FOR INFORMATION INTERCHANGE (ASCII) WITH PUNCHED CARD CODES AND EBCDIC TRANSLATION

AMERICAN NATIONAL STANDARD CODE FOR INFORMATION INTERCHANGE (ASCII) WITH PUNCHED CARD CODES AND EBCDIC TRANSLATION

	b8 b7 b6 b5	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1	1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1
b4 b3 b2 b1	COL ROW	0	1	2	3	4	5	6	7	8	9	10 (A)	11 (B)	12 (C)	13 (D)	14 (E)	15 (F)
0 0 0 0	0	NUL 12–0–9–8–1 NUL 00	DLE 12–11–9–8–1 DLE 10	SP no-punch SP 40	0 0 0 F0	@ 8–4 @ 7C	P 11–7 P D7	` 8–1 ` 79	p 12–11–7 p 97	11–0–9–8–1 DS 20	12–11–0–9–8–1 30	12–0–9–1 41	12–11–9–8 58	12–11–0–9–6 76	12–11–8–7 9F	12–11–0–8 B8	12–11–9–8–4 DC
0 0 0 1	1	SOH 12–9–1 SOH 01	DC1 11–9–1 DC1 11	! 12–8–7 | 4F	1 1 1 F1	A 12–1 A C1	Q 11–8 Q D8	a 12–0–1 a 81	q 12–11–8 q 98	0–9–1 SOS 21	9–1 31	12–0–9–2 42	11–8–1 59	12–11–0–9–7 77	11–0–8–1 A0	12–11–0–9 B9	12–11–9–8–5 DD
0 0 1 0	2	STX 12–9–2 STX 02	DC2 11–9–2 DC2 12	″ 8–7 ″ 7F	2 2 2 F2	B 12–2 B C2	R 11–9 R D9	b 12–0–2 b 82	r 12–11–9 r 99	0–9–2 FS 22	11–9–8–2 CC 1A	12–0–9–3 43	11–0–9–2 62	12–11–0–9–8 78	11–0–8–2 AA	12–11–0–8–2 BA	12–11–9–8–6 DE
0 0 1 1	3	ETX 12–9–3 ETX 03	DC3 11–9–3 TM 13	# 8–3 # 7B	3 3 3 F3	C 12–3 C C3	S 0–2 S E2	c 12–0–3 c 83	s 11–0–2 s A2	0–9–3 23	9–3 33	12–0–9–4 44	11–0–9–3 63	12–0–8–1 80	11–0–8–3 AB	12–11–0–8–3 BB	12–11–9–8–7 DF
0 1 0 0	4	EOT 9–7 EOT 37	DC4 9–8–4 DC4 3C	$ 11–8–3 $ 5B	4 4 4 F4	D 12–4 D C4	T 0–3 T E3	d 12–0–4 d 84	t 11–0–3 t A3	0–9–4 BYP 24	9–4 PN 34	12–0–9–5 45	11–0–9–4 64	12–0–8–2 8A	11–0–8–4 AC	12–11–0–8–4 BC	11–0–9–8–2 EA
0 1 0 1	5	ENQ 0–9–8–5 ENQ 2D	NAK 9–8–5 NAK 3D	% 0–8–4 % 6C	5 5 5 F5	E 12–5 E C5	U 0–4 U E4	e 12–0–5 e 85	u 11–0–4 u A4	11–9–5 NL 15	9–5 RS 35	12–0–9–6 46	11–0–9–5 65	12–0–8–3 8B	11–0–8–5 AD	12–11–0–8–5 BD	11–0–9–8–3 EB
0 1 1 0	6	ACK 0–9–8–6 ACK 2E	SYN 9-2 SYN 32	& 12 & 50	6 6 6 F6	F 12–6 F C6	V 0–5 V E5	f 12–0–6 f 86	v 11–0–5 v A5	12–9–6 LC 06	9–6 UC 36	12–0–9–7 47	11–0–9–6 66	12–0–8–4 8C	11–0–8–6 AE	12–11–0–8–6 BE	11–0–9–8–4 ⑀ EC
0 1 1 1	7	BEL 0–9–8–7 BEL 2F	ETB 0–9–6 ETB 26	′ 8–5 ′ 7D	7 7 7 F7	G 12–7 G C7	W 0–6 W E6	g 12–0–7 g 87	w 11–0–6 w A6	11–9–7 IL 17	12–9–8 GE 08	12–0–9–8 48	11–0–9–7 67	12–0–8–5 8D	11–0–8–7 AF	12–11–0–8–7 BF	11–0–9–8–5 ED
1 0 0 0	8	BS 11–9–6 BS 16	CAN 11–9–8 CAN 18	(12–8–5 (4D	8 8 8 F8	H 12–8 H C8	X 0–7 X E7	h 12–0–8 h 88	x 11–0–7 x A7	0–9–8 28	9–8 38	12–8–1 49	11–0–9–8 68	12–0–8–6 8E	12–11–0–8–1 B0	12–0–9–8–2 CA	11–0–9–8–6 EE
1 0 0 1	9	HT 12–9–5 HT 05	EM 11–9–8–1 EM 19	) 11–8–5) 5D	9 9 9 F9	I 12–9 I C9	Y 0–8 Y E8	i 12–0–9 i 89	y 11–0–8 y A8	0–9–8–1 29	9–8–1 39	12–11–9–1 51	0–8–1 69	12–0–8–7 8F	12–11–0–1 B1	12–0–9–8–3 CB	11–0–9–8–7 EF
1 0 1 0	10 (A)	LF 0–9–5 LF 25	SUB 9–8–7 SUB 3F	* 11–8–4 * 5C	: 8–2 : 7A	J 11–1 J D1	Z 0–9 Z E9	j 12–11–1 j 91	z 11–0–9 z A9	0–9–8–2 SM 2A	9–8–2 3A	12–11–9–2 52	12–11–0 70	12–11–8–1 90	12–11–0–2 B2	12–0–9–8–4 ⑁ CC	12–11–0–9–8–2 | (LVM) FA
1 0 1 1	11 (B)	VT 12–9–8–3 VT 0B	ESC 0–9–7 ESC 27	+ 12–8–6 + 4E	; 11–8–6 ; 5E	K 11–2 K D2	[12-8-2 ¢ 4A	k 12–11–2 k 92	{ 12–0 { C0	0–9–8–3 CU2 2B	9–8–3 CU3 3B	12–11–9–3 53	12–11–0–9–1 71	12–11–8–2 9A	12–11–0–3 B3	12–0–9–8–5 CD	12–11–0–9–8–3 FB
1 1 0 0	12 (C)	FF 12–9–8–4 FF 0C	FS 11–9–8–4 IFS 1C	, 0–8–3 , 6B	< 12–8–4 < 4C	L 11–3 L D3	\ 0–8–2 \ E0	l 12–11–3 l 93	¦ 12–11 ¦ 6A	0–9–8–4 2C	12–9–4 PF 04	12–11–9–4 54	12–11–0–9–2 72	12–11–8–3 9B	12–11–0–4 B4	12–0–9–8–6 ⑂ CE	12–11–0–9–8–4 FC
1 1 0 1	13 (D)	CR 12–9–8–5 CR 0D	GS 11–9–8–5 IGS 1D	– 11 – 60	= 8–6 = 7E	M 11–4 M D4	] 11-8-2 ! 5A	m 12–11–4 m 94	} 11–0 } D0	12–9–8–1 RLF 09	11–9–4 RES 14	12–11–9–5 55	12–11–0–9–3 73	12–11–8–4 9C	12–11–0–5 B5	12–0–9–8–7 CF	12–11–0–9–8–5 FD
1 1 1 0	14 (E)	SO 12–9–8–6 SO 0E	RS 11–9–8–6 IRS 1E	. 12–8–3 . 4B	> 0–8–6 > 6E	N 11–5 N D5	^ 11-8-7 ¬ 5F	n 12–11–5 n 95	~ 11–0–1 ~ A1	12–9–8–2 SMM 0A	9–8–6 3E	12–11–9–6 56	12–11–0–9–4 74	12–11–8–5 9D	12–11–0–6 B6	12–11–9–8–2 DA	12–11–0–9–8–6 FE
1 1 1 1	15 (F)	SI 12–9–8–7 SI 0F	US 11–9–8–7 IUS 1F	/ 0–1 / 61	? 0–8–7 ? 6F	O 11–6 O D6	0̄–8–5 _ 6D	o 12–11–6 o 96	DEL 12–9–7 DEL 07	11–9–8–3 CU1 1B	11–0–9–1 E1	12–11–9–7 57	12–11–0–9–5 75	12–11–8–6 9E	12–11–0–7 B7	12–11–9–8–3 DB	EO 12–11–0–9–8–7 FF

LEGEND

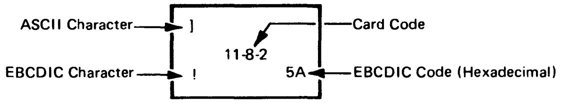


TABLE A-3. EXTENDED BINARY CODED DECIMAL INTERCHANGE CODE (EBCDIC) WITH PUNCHED CARD CODES AND ASCII TRANSLATION

EXTENDED BINARY CODED DECIMAL INTERCHANGE CODE (EBCDIC) WITH PUNCHED CARD CODES AND ASCII TRANSLATION

BITS 0 1 2 3		0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1	1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1
BITS 4 5 6 7	1ST HEX 2ND	0	1	2	3	4	5	6	7	8	9	A (10)	B (11)	C (12)	D (13)	E (14)	F (15)
0 0 0 0	0	NUL 12–0–9–8–1 NUL 00	DLE 12–11–9–8–1 DLE 10	DS 11–0–9–8–1 80	12–11–0–9–8–1 90	SP no punch SP 20	& 12 & 26	– 11 – 2D	12–11–0 BA	12–0–8–1 C3	12–11–8–1 CA	11–0–8–1 D1	12–11–0–8–1 D8	{ 12–0 { 7B	} 11–0 } 7D	\ 0–8–2 \ 5C	0 0 0 30
0 0 0 1	1	SOH 12–9–1 SOH 01	DC1 11–9–1 DC1 11	SOS 0-9-1 81	9–1 91	12–0–9–1 A0	12–11–9–1 A9	/ 0–1 / 2F	12–11–0–9–1 BB	a 12–0–1 a 61	j 12–11–1 j 6A	~ 11–0–1 ~ 7E	12–11–0–1 D9	A 12–1 A 41	J 11–1 J 4A	11–0–9–1 9F	1 1 1 31
0 0 1 0	2	STX 12–9–2 STX 02	DC2 11–9–2 DC2 12	FS 0–9–2 82	SYN 9–2 SYN 16	12–0–9–2 A1	12–11–9–2 AA	11–0–9–2 B2	12–11–0–9–2 BC	b 12–0–2 b 62	k 12–11–2 k 6B	s 11–0–2 s 73	12–11–0–2 DA	B 12–2 B 42	K 11–2 K 4B	S 0–2 S 53	2 2 2 32
0 0 1 1	3	ETX 12–9–3 ETX 03	TM 11–9–3 DC3 13	0–9–3 83	9–3 93	12–0–9–3 A2	12–11–9–3 AB	11–0–9–3 B3	12–11–0–9–3 BD	c 12–0–3 c 63	l 12–11–3 l 6C	t 11–0–3 t 74	12–11–0–3 DB	C 12–3 C 43	L 11–3 L 4C	T 0–3 T 54	3 3 3 33
0 1 0 0	4	PF 12–9–4 9C	RES 11–9–4 9D	BYP 0–9–4 84	PN 9–4 94	12–0–9–4 A3	12–11–9–4 AC	11–0–9–4 B4	12–11–0–9–4 BE	d 12–0–4 d 64	m 12–11–4 m 6D	u 11–0–4 u 75	12–11–0–4 DC	D 12–4 D 44	M 11–4 M 4D	U 0–4 U 55	4 4 4 34
0 1 0 1	5	HT 12–9–5 HT 09	NL 11–9–5 85	LF 0–9–5 LF 0A	RS 9–5 95	12–0–9–5 A4	12–11–9–5 AD	11–0–9–5 B5	12–11–0–9–5 BF	e 12–0–5 e 65	n 12–11–5 n 6E	v 11–0–5 v 76	12–11–0–5 DD	E 12–5 E 45	N 11–5 N 4E	V 0–5 V 56	5 5 5 35
0 1 1 0	6	LC 12–9–6 86	BS 11–9–6 BS 08	ETB 0–9–6 ETB 17	UC 9–6 96	12–0–9–6 A5	12–11–9–6 AE	11–0–9–6 B6	12–11–0–9–6 C0	f 12–0–6 f 66	o 12–11–6 o 6F	w 11–0–6 w 77	12–11–0–6 DE	F 12–6 F 46	O 11–6 O 4F	W 0–6 W 57	6 6 6 36
0 1 1 1	7	DEL 12–9–7 DEL 7F	IL 11–9–7 87	ESC 0–9–7 ESC 1B	EOT 9–7 EOT 04	12–0–9–7 A6	12–11–9–7 AF	11–0–9–7 B7	12–11–0–9–7 C1	g 12–0–7 g 67	p 12–11–7 p 70	x 11–0–7 x 78	12–11–0–7 DF	G 12–7 G 47	P 11–7 P 50	X 0–7 X 58	7 7 7 37
1 0 0 0	8	GE 12–9–8 97	CAN 11–9–8 CAN 18	0–9–8 88	9–8 98	12–0–9–8 A7	12–11–9–8 B0	11–0–9–8 B8	12–11–0–9–8 C2	h 12–0–8 h 68	q 12–11–8 q 71	y 11–0–8 y 79	12–11–0–8 E0	H 12–8 H 48	Q 11–8 Q 51	Y 0–8 Y 59	8 8 8 38
1 0 0 1	9	RLF 12–9–8–1 8D	EM 11–9–8–1 EM 19	0–9–8–1 89	9–8–1 99	12–8–1 A8	11–8–1 B1	0–8–1 B9	\` 8–1 \` 60	i 12–0–9 i 69	r 12–11–9 r 72	z 11–0–9 z 7A	12–11–0–9 E1	I 12–9 I 49	R 11–9 R 52	Z 0–9 Z 5A	9 9 9 39
1 0 1 0	A (10)	SMM 12–9–8–2 8E	CC 11–9–8–2 92	SM 0–9–8–2 8A	9–8–2 9A	¢ 12–8–2 [5B	! 11–8–2] 5D	¦ 12–11 ¦ 7C	: 8–2 : 3A	12–0–8–2 C4	12–11–8–2 CB	11–0–8–2 D2	12–11–0–8–2 E2	12–0–9–8–2 E8	12–11–9–8–2 EE	11–0–9–8–2 F4	I (LVM) 12–11–0–9–8–2 FA
1 0 1 1	B (11)	VT 12–9–8–3 VT 0B	CU1 11–9–8–3 8F	CU2 0–9–8–3 8B	CU3 9–8–3 9B	. 12–8–3 . 2E	$ 11–8–3 $ 24	, 0–8–3 , 2C	# 8–3 # 23	12–0–8–3 C5	12–11–8–3 CC	11–0–8–3 D3	12–11–0–8–3 E3	12–0–9–8–3 E9	12–11–9–8–3 EF	11–0–9–8–3 F5	12–11–0–9–8–3 FB
1 1 0 0	C (12)	FF 12–9–8–4 FF 0C	IFS 11–9–8–4 FS 1C	0–9–8–4 8C	DC4 9–8–4 DC4 14	< 12–8–4 < 3C	* 11–8–4 * 2A	% 0–8–4 % 25	@ 8–4 @ 40	12–0–8–4 C6	12–11–8–4 CD	11–0–8–4 D4	12–11–0–8–4 E4	⑁ 12–0–9–8–4 EA	12–11–9–8–4 F0	⑂ 11–0–9–8–4 F6	12–11–0–9–8–4 FC
1 1 0 1	D (13)	CR 12–9–8–5 CR 0D	IGS 11–9–8–5 GS 1D	ENQ 0–9–8–5 ENQ 05	NAK 9–8–5 NAK 15	(12–8–5 (28	) 11–8–5) 29	0̄–8–5 _ 5F	ʼ 8–5 ʼ 27	12–0–8–5 C7	12–11–8–5 CE	11–0–8–5 D5	12–11–0–8–5 E5	12–0–9–8–5 EB	12–11–9–8–5 F1	11–0–9–8–5 F7	12–11–0–9–8–5 FD
1 1 1 0	E (14)	SO 12–9–8–6 SO 0E	IRS 11–9–8–6 RS 1E	ACK 0–9–8–6 ACK 06	9–8–6 9E	+ 12–8–6 + 2B	; 11–8–6 ; 3B	> 0–8–6 > 3E	= 8–6 = 3D	12–0–8–6 C8	12–11–8–6 CF	11–0–8–6 D6	12–11–0–8–6 E6	⑃ 12–0–9–8–6 EC	12–11–9–8–6 F2	11–0–9–8–6 F8	12–11–0–9–8–6 FE
1 1 1 1	F (15)	SI 12–9–8–7 SI 0F	IUS 11–9–8–7 US 1F	BEL 0–9–8–7 BEL 07	SUB 9–8–7 SUB 1A	| 12–8–7 ! 21	¬ 11–8–7 ^ 5E	? 0–8–7 ? 3F	" 8–7 " 22	12–0–8–7 C9	12–11–8–7 D0	11–0–8–7 D7	12–11–0–8–7 E7	12–0–9–8–7 ED	12–11–9–8–7 F3	11–0–9–8–7 F9	12–11–0–9–8–7 EO FF

LEGEND

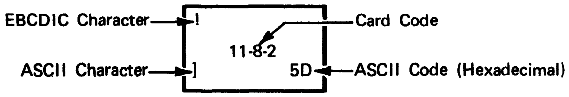


TABLE A-4. CONTROL DATA CHARACTER SETS SHOWING TRANSLATIONS BETWEEN DISPLAY CODE AND ASCII/EBCDIC

DISPLAY CODE		ASCII				EBCDIC			
		UPPER CASE		LOWER CASE		UPPER CASE		LOWER CASE	
OCTAL	CH	CH	HEX	CH	HEX	CH	HEX	CH	HEX
00	:	:	3A	SUB	1A	:	7A	SUB	3F
01	A	A	41	a	61	A	C1	a	81
02	B	B	42	b	62	B	C2	b	82
03	C	C	43	c	63	C	C3	c	83
04	D	D	44	d	64	D	C4	d	84
05	E	E	45	e	65	E	C5	e	85
06	F	F	46	f	66	F	C6	f	86
07	G	G	47	g	67	G	C7	g	87
10	H	H	48	h	68	H	C8	h	88
11	I	I	49	i	69	I	C9	i	89
12	J	J	4A	j	6A	J	D1	j	91
13	K	K	4B	k	6B	K	D2	k	92
14	L	L	4C	l	6C	L	D3	l	93
15	M	M	4D	m	6D	M	D4	m	94
16	N	N	4E	n	6E	N	D5	n	95
17	O	O	4F	o	6F	O	D6	o	96
20	P	P	50	p	70	P	D7	p	97
21	Q	Q	51	q	71	Q	D8	q	98
22	R	R	52	r	72	R	D9	r	99
23	S	S	53	s	73	S	E2	s	A2
24	T	T	54	t	74	T	E3	t	A3
25	U	U	55	u	75	U	E4	u	A4
26	V	V	56	v	76	V	E5	v	A5
27	W	W	57	w	77	W	E6	w	A6
30	X	X	58	x	78	X	E7	x	A7
31	Y	Y	59	y	79	Y	E8	y	A8
32	Z	Z	5A	z	7A	Z	E9	z	A9
33	0	0	30	DLE	10	0	F0	DLE	10
34	1	1	31	DC1	11	1	F1	DC1	11
35	2	2	32	DC2	12	2	F2	DC2	12
36	3	3	33	DC3	13	3	F3	TM	13
37	4	4	34	DC4	14	4	F4	DC4	3C

DISPLAY CODE		ASCII				EBCDIC			
		UPPER CASE		LOWER CASE		UPPER CASE		LOWER CASE	
OCTAL	CH	CH	HEX	CH	HEX	CH	HEX	CH	HEX
40	5	5	35	NAK	15	5	F5	NAK	3D
41	6	6	36	SYN	16	6	F6	SYN	32
42	7	7	37	ETB	17	7	F7	ETB	26
43	8	8	38	CAN	18	8	F8	CAN	18
44	9	9	39	EM	19	9	F9	EM	19
45	+	+	2B	VT	0B	+	4E	VT	0B
46	—	—	2D	CR	0D	—	60	CR	0D
47	*	*	2A	LF	0A	*	5C	LF	25
50	/	/	2F	SI	0F	/	61	SI	0F
51	(	(	28	BS	08	(	4D	BS	16
52	)	)	29	HT	09	)	5D	HT	05
53	$	$	24	EOT	04	$	5B	EOT	37
54	=	=	3D	GS	1D	=	7E	IGS	1D
55	SP	SP	20	NUL	00	SP	40	NUL	00
56	,	,	2C	FF	0C	,	6B	FF	0C
57	.	.	2E	SO	0E	.	4B	SO	0E
60	≡ =	=	23	ETX	03	#	7B	ETX	03
61	[	[	5B	FS	1C	¢	4A	IFS	1C
62	]	]	5D	SOH	01	!	5A	SOH	01
63	%	%	25	ENQ	05	%	6C	ENQ	2D
64	≠ "	"	22	STX	02	"	7F	STX	02
65	→ _	_	5F	DEL	7F	_	6D	DEL	07
66	∨ !	!	21	}	7D	|	4F	}	D0
67	∧ &	&	26	ACK	06	&	50	ACK	2E
70	↑ '	'	27	BEL	07	'	7D	BEL	2F
71	↓ ?	?	3F	US	1F	?	6F	IUS	1F
72	<	<	3C	{	7B	<	4C	{	C0
73	>	>	3E	RS	1E	>	6E	IRS	1E
74	≤ @	@	40	`	60	@	7C	`	79
75	≥ \	\	5C	¦	7C	\	E0	¦	6A
76	¬ ^	^	5E	~	7E	¬	5F	~	A1
77	;	;	3B	ESC	1B	;	5E	ESC	27

NOTES:

1. The terms "upper case" and "lower case" apply only to the case conversions, and do not necessarily reflect any true "case".
2. When translating from Display Code to ASCII/EBCDIC, the "upper case" equivalent character is taken.
3. When translating from ASCII/EBCDIC to Display Code, the "upper case" and "lower case" characters fold together to a single Display Code equivalent character.
4. All ASCII and EBCDIC codes not listed are translated to Display Code 55 (SP).
5. Where two Display Code graphics are shown for a single octal code, the leftmost graphic corresponds to the CDC 64-character set (system assembled with IP.CSET set to C64.1), and the rightmost graphic corresponds to the CDC 64-character ASCII subset (system assembled with IP.CSET set to C64.2).
6. In a 63-character set system, the display code for the : graphic is 63. The % character does not exist, and translations from ASCII-EBCDIC % or ENQ yield blank (55_8).

TABLE A-5. FULL ASCII CHARACTER SET

← 128-Character Set →

← 96-Character Subset →

← 64-Character Subset →

Bits				b_7 →	0	0	0	0	1	1	1	1
				b_6 →	0	0	1	1	0	0	1	1
				b_5 →	0	1	0	1	0	1	0	1
b_4 ↓	b_3 ↓	b_2 ↓	b_1 ↓	ROW ↓ / COLUMN →	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	A	LF	SUB	*	:	J	Z	j	z
1	0	1	1	B	VT	ESC	+	;	K	[	k	{
1	1	0	0	C	FF	FS	,	<	L	\	l	¦
1	1	0	1	D	CR	GS	–	=	M	]	m	}
1	1	1	0	E	SO	RS	.	>	N	⌢	n	~
1	1	1	1	F	SI	US	/	?	O	—	o	DEL

Three types of inline diagnostics are available for CCI:

- CE error messages. These messages, which reflect hardware errors, are delivered to the host engineering file (called the system error file). The messages should be processed by the Hardware Performance Analyzer or the user's analysis program.

NOTE

> If the user has elected to purchase a maintenance contract, the contents of the host engineering file can be easily analyzed by the Hardware Performance Analyzer (HPA); otherwise, the user must devise his own method for making the contents available.

- Halt messages. These are delivered to the NPU console when the NPU stops. Normally, the NPU contents are dumped to the host prior to restarting.

- Statistics messages. These messages reflect hardware performance (normal or erroneous) and also are delivered to the host engineering file as a user option. This file should be processed by the HPA or the user's analysis program.

CE ERROR MESSAGES

CE error messages can be divided into the following categories:

- Modem signal messages (error codes 01 through 03, 0B and 0C)
- CLA messages (error codes 04 through 0A and 0D through 10)
- MLIA messages (error code 11)
- TIP messages (error codes 12 and 13)
- Coupler messages (error codes 20 through 24 and 26 through 29)
- Unused codes (all others).

These codes are described in tables B-1 and B-2. The first six bytes of these messages all contain the same header information: the destination node, source node, connection number (0), block serial number and block type, primary function code (A_{16}), and secondary function code (0). The seventh byte contains the error code, and any additional bytes explain the nature of the error condition. In table B-2, the byte containing the error code is numbered as byte 1.

TABLE B-1. CE ERROR MESSAGE CODES

Error Code	Significance
01_{16}	Not used
02_{16}	Abnormal data set ready (DSR) or clear to send (CTS)
03_{16}	Abnormal operation of data carrier detect (DCD)
04_{16}	Unsolicited output data demand (ODD)
05_{16}	CLA address out of range
06_{16}	Illegal multiplex loop format
07_{16}	Unsolicited input
08_{16}	Multiplex input loop error
09_{16}	Multiplex output loop error
$0A_{16}$	ODD timeout
$0B_{16}$	DCD timeout
$0C_{16}$	Abnormal secondary data carrier detect (SDCD)
$0D_{16}$	Excessive CLA status messages
$0E_{16}$	Not used
$0F_{16}$	Next character not available (output)
10_{16}	Data transfer overrun (input)
11_{16}	MLIA error
12_{16}	Upline interruption occurred during a transmission from a Mode 4 terminal
13_{16}	Bad data block clarifier was encountered by the TTY TIP in a data block
14_{16} thru $1F_{16}$	Not used
20_{16}	Deadman timeout
21_{16}	Spurious coupler interrupt
22_{16}	Not used
23_{16}	Coupler hardware timeout on input
24_{16}	Input data transfer terminated by PPU
25_{16}	Not used
26_{16}	Not used
27_{16}	Output data transfer terminated by PPU
28_{16}	Hardware timeout on output
29_{16}	End of operation (EOP) missing

TABLE B-2. CE ERROR MESSAGE TEXT INTERPRETATION

Error Code (Byte 1)	Interpretation	
	Byte	Contents
02_{16} thru 10_{16}	2	Port number (CLA thumbwheel setting) of communications line
	3	Always zero
	4†	Byte 1 of the CLA status, as follows: bit 7 clear to send bit 6 data set ready bit 5 data carrier detect bit 4 ring indicator bit 3 quality monitor bit 2 signal quality detector bit 1 input loop error bit 0 output loop error
	5†	Byte 2 of the CLA status, as follows: bit 7 parity error status bit 6 data transfer overrun bit 5 framing error status bit 4 next character not available bits 3 thru 0 unused
11_{16}	2	Error type indicator, as follows: 0 error condition was restored (this is the only text byte) 1 error counts given in accompanying bytes 2 MLIA failure occurred (this is the only text byte)
	3	Input loop error count
	4	Lost data count
	5	Alarm count
12_{16}	2	Port number (CLA thumbwheel setting) of the communications line
	3	Always zero
	4	Cluster address of the terminal, as follows: Mode 4 terminals 70_{16} thru $7F_{16}$ Teletypewriter, HASP, 2780 and 3780 terminals 0
	5	Terminal address of the terminal, as follows: Mode 4A terminals 60_{16} Mode 4C terminals 61_{16} thru $6F_{16}$ Teletypewriter terminals 0 HASP terminals 0 thru 7†† 2780 and 3780 terminals 0, 12_{16}, 13_{16}†††

TABLE B-2. CE ERROR MESSAGE TEXT INTERPRETATION (Contd)

Error Code (Byte 1)	Interpretation	
	Byte	Contents
	6	Upper three bits contain the device type of the terminal, as follows:
		Console 0
		Card reader 1
		Line printer 2
		Card punch 3
		Nonimpact printer or plotter 4
		Lower five bits contain the terminal class of the terminal, as follows:
		Teletypewriter terminal 1
		2780 terminal 7
		3780 terminal 8
		HASP terminal 9
		Mode 4 terminal 10
	7	Error code, explaining the cause of the message, as follows:
		2 No response error counter for the terminal overflowed
		3 Bad response error counter for the terminal overflowed
		4 Error response error counter for the terminal overflowed
13_{16}	2	Data block clarifier byte that caused the message
20_{16}	2	Last coupler state byte
	3	Current coupler state byte
21_{16} thru 29_{16}	2	Upper portion of coupler status register contents, as follows:
		bit 15 alarm condition flag
		bit 14 chain address of zero detected (last word of NPU buffer contained a zero)
		bit 13 unused
		bit 12 unused
		bit 11 channel parity error detected (a 12-bit word plus parity from data channel did not have odd parity and enable parity switch was on)
		bit 10 hardware timeout occurred (inactive status returned during PPU data input or output because coupler was selected and active more than 3 seconds)
		bit 9 NPU status accepted flag (PPU reads status word)
		bit 8 orderword register loaded (PPU writes order word)
	3	Lower portion of coupler status register contents, as follows:
		bit 7 transfer terminated by PPU
		bit 6 unused
		bit 5 transmission complete (set by PPU)
		bit 4 power failure flag
		bit 3 memory address register (main memory address 1) loaded by PPU or NPU
		bit 2 NPU status register word loaded by NPU
		bit 1 NPU memory protect fault flag
		bit 0 NPU memory parity error flag

†Not used for codes 04_{16} through 07_{16}, $0A_{16}$, $0B_{16}$, and $0F_{16}$

††Corresponds to stream number of device; consoles on stream 0, other devices on zero or nonzero streams

†††Corresponds to stream number of device; punches on nonzero streams, all other devices on stream 0

HALT MESSAGES AND DUMP INTERPRETATION

When the CCI stops the NPU because of an unrecoverable condition caused either by software or hardware errors, the CCI delivers a halt message to the NPU console. The format of the halt message is given in figure B-1. The halt codes in the message are given in table B-3.

When such a halt occurs, the host processor normally executes an upline dump of the NPU main memory, micromemory, and the file 1 registers. Thereafter, the host attempts to reload the NPU main memory. This is accomplished directly through the coupler. Before any loading attempt, a dump is normally taken.

The NPU can be stopped locally by master clearing it using the MASTER CLEAR switch on the maintenance control panel.

In some cases, a halt code message is not generated. In these cases the dump listing must be consulted to find the cause of the failure. In all cases where the cause of stoppage is not apparent, the customer engineer or system analyst will probably want to consult the dump listing.

HALT CODES

Halt codes can be divided into several categories: those primarily resulting from incorrect switch settings, those caused by hardware malfunctions, and those that can be either hardware or software problems.

The first category includes detection of a duplicate CLA address (halt code 0012). This condition is usually caused by two CLA switches being set to the same address. Such a fault can normally be corrected by the operator resetting the switches.

```
mm/dd        hh:nn:ss
HALT xxxxx   yyyy
wwww
zzzz
```

mm/dd — The month and day from the system clock.

hh:nn:ss — The hour, minute, and second the halt message was issued, taken from the system clock.

xxxxx — The hexadecimal address of the program in control when the halt occurred, or a code value dependent on the accompanying halt code yyyy.

yyyy — The hexadecimal halt code, as described in table B-3.

wwww — The number of the port associated with the halt condition; this time appears only for halt codes 5 and D_{16}.

zzzz — The hexadecimal buffer address associated with the halt condition; this line appears only for the buffer halt codes A_{16}, B_{16}, and C_{16}.

Figure B-1. Halt Message Format

TABLE B-3. HALT CODES

Code	Significance
0001_{16}	Power failure
0002_{16}	Memory parity error
0003_{16}	Program protect error
0004_{16}	Interrupt count less than 0
0005_{16}	MLIA failure (reported by MLIA hardware status)
0006_{16}	Overran circular input buffer
0007_{16}	Branch to zero detected
0008_{16}	Invalid halt code
0009_{16}	Ran out of buffers
$000A_{16}$	Duplicate release of buffer
$000B_{16}$	Buffer chain error during buffer get
$000C_{16}$	Buffer out of range
$000D_{16}$	Coupler alarm condition
$000E_{16}$	Monitor stopped
$000F_{16}$	Too many worklists from one CLA
0010_{16}	Invalid halt code
0011_{16}	Bad MLIA initialization status
0012_{16}	Duplicate CLA address detected
0013_{16}	Chain address is 0
0014_{16}	Invalid halt code
0015_{16}	Invalid coupler orderword
0016_{16}	Invalid halt code
0017_{16}	Invalid halt code

In the second category, the halt codes are the following:

- Power failures (code 0001)
- Memory parity error (code 0002)
- Memory protect bit error (code 0003)
- Bad MLIA initialization status (code 0011)

Such conditions are usually caused by some type of hardware failure and normally must be repaired by a customer engineer.

The third category of halt codes (all those not already specified) are caused either by a hardware failure or by a software error. To correct this category of problems, the customer engineer should normally first be called to check the hardware. If the hardware is functioning properly, a system analyst should be called. Have all upline dumps taken by the host available for the customer engineer and/or the system analyst.

At most times when a halt occurs and at all times when using the online diagnostic programs, halt codes and diagnostic test responses are printed at the NPU console and dump interpretation is not needed. However, 1) if a halt occurs after loading but before completion of initialization, or 2) if the system becomes trapped in a looping condition during initialization (before the CCI system header prints on the NPU console), dump interpretation might be necessary to determine which halt has occurred, or in which subroutine of the initialization section the program is looping.

DUMP INTERPRETATION INSTRUCTIONS

When interpreting the upline dump printout to determine the cause of a halt or looping condition, first examine the contents of memory location 30_{16} as reflected in the dump printout. If nonzero, a halt has occurred and the halt code value is contained in that location.

If memory location 30_{16} equals zero, examine the address of the NPINTAB entry in the address table which begins at fixed memory address 150_{16}. (This is the table which is displayed on the NPU console at initialization. NPINTAB has a fixed address.) Table B-4 lists the contents of the address table. Entry NPINTAB gives the starting address for the NPINTAB table, the format of which is illustrated in figure B-2. The NPISFL entry in the NPINTAB table contains the flags which mark the initialization subroutines that had completed running when the looping condition occurred. This information should be given to the system analyst along with the dump printouts.

STATISTICS MESSAGES

The first five 8-bit bytes of these messages all contain the same header information: the destination node, source node, connection number (0), block serial number and block type, and primary function code (7). The rest of the bytes vary according to the secondary function code in the sixth byte, which can be 0 for NPU statistics, 1 for communications line statistics, or 2 for terminal statistics. If the statistics service messages are routed to the NPU console, these six prefixing bytes appear. When the messages are entered in the host engineering file, all six bytes are replaced by a record header and an associated message type code. Regardless of routing, the balance of the messages are described in table B-5.

	15				11				7				3			0
Word 0 (NPSODD)	0	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x
Word 1 (NPISFL)	B15	0	0	0	0	0	0	0	B7	B6	B5	B4	B3	B2	B1	B0
Word 2 (NPBMLS)	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y

NPSODD — Duplicate CLA address, where xx . . . x is the duplicated CLA address between 01_{16} and FE_{16}; 00_{16} indicates a preset value (no duplicates), and FF_{16} indicates no response.

NPISFL — Initialization completion sequence flags, where B15 and B7 through B0 indicate start or completion of various tasks as follows:

B15	All buffers initialized, system initialization completed
B7	Second phase of buffer initialization started or completed
B6	Initialization of fixed lines started or completed
B5	Initialization of MLIA started or completed
B4	TIP, NPU console, and diagnostics initialization started or completed
B3	Miscellaneous NPU console initialization started or completed
B2	Initialization of worklist control blocks started or completed
B1	Initialization of buffers started or completed
B0	Setup of program protect bits started or completed

A task was completed if the next higher bit is set.

A task was started but not completed if only the task sequence bit is set.

NPBMLS — Bad MLIA initialization status, where any value for yy . . . y other than 0009_{16} indicates bad status. Call a customer engineer to run MLIA diagnostics.

Figure B-2. NPINTAB Format

TABLE B-4. ADDRESS TABLE

Starting Address	Word	Mnemonic	Contents†	
150_{16}	0	BYWLCB	Worklist control block	Base system
	1	JSWLADDR	Worklist entry by level number	
	2	BITCB	Internal processing TCB	
	3	B1BUFF	Internal processing block	
	4	JKMASK	Interrupt masks	
	5	JKTMASK	PBAMASK save area	
	6	CBTIMTBL	TIMAL table	
	7	JACT	PD controller table	
	8	BECTLBK	Buffer control block (BCB)	
	9	BYSTAMP	Buffer stamp area	
	A_{16}		Unused	
	B_{16}		Unused	
	C_{16}	NAPORT	Port table	Multiplex Subsystem
	D_{16}	BQCIB	Circular input buffer (CIB)	
	E_{16}		Unused	
	F_{16}	CGLCBS	Line control blocks (LCB)	Lines and TIPs
	10_{16}	CHSUBLCB	SUB LCBs	
	11_{16}	CGTCBS	Terminal control blocks (TCB)	
	12_{16}	BJTIPTYPT	TIP type table	
	13_{16}	NJTECT	Terminal characteristics table	
	14_{16}		Unused	
	15_{16}	NPINTAB	Initialization status table	Initialization Information
	16_{16}		Unused	
	17_{16}	CCPVER	CCI version address	
	18_{16}	CCPCYC	CCI cycle address	
	19_{16}	CCPLEV	CCI level address	

†Contents of table are displayed at end of a successful initialization.

TABLE B-5. STATISTICS MESSAGE INTERPRETATION

Message Type	Message Interpretation		
	Bytes	Contents	
NPU statistics	1 and 2	Number of upline service messages generated	
	3 and 4	Number of downline service messages processed	
	5 and 6	Number of bad service messages received	
	7 and 8	Number of blocks discarded because of a bad network address	
	9 and 10	Number of blocks discarded because of bad format	
	11 and 12	Number of times NPU ran at regulation level 4 (no regulation)	
	13 and 14	Number of times NPU ran at regulation level 3, described in section 2	
	15 and 16	Number of times NPU ran at regulation level 2, described in section 2	
	17 and 18	Number of times NPU ran at regulation level 1, described in section 2	
	19 and 20	Number of times NPU ran at regulation level 0, described in section 2	
	21 and 22	Number of network assurance protocol timeouts	
Line statistics	1	Port number of the communications line	
	2 thru 4	Always zero	
	5 and 6	Number of blocks transmitted upline	
	7 and 8	Number of blocks received downline	
	9 and 10	Number of characters transmitted upline in good blocks	
	11 and 12	Number of characters received downline in good blocks	
Terminal statistics[†††]	1	Port number of the communications line servicing the terminal	
	2	Always zero	
	3	Cluster address of the terminal, as follows:	
		Mode 4 terminals	70_{16} thru $7F_{16}$
		Teletypewriter, HASP, 2780 and 3780 terminals	0
		Terminal address of the terminal, as follows:	
		Mode 4A terminals	60_{16}
		Mode 4C terminals	61_{16} thru $6F_{16}$
		Teletypewriter terminals	0
		HASP terminals	0 thru 7[†]
		2780 and 3780 terminals	0, 12_{16}, 13_{16}[††]
	5	Upper three bits contain the device type of the terminal, as follows:	
		Console	0
		Card reader	1
		Line printer	2
		Card punch	3
		Nonimpact printer or plotter	4
		Lower five bits contain the terminal class of the terminal, as follows:	
		Teletypewriter terminal	1
		2780 terminal	7
		3780 terminal	8
		HASP terminal	9
		Mode 4 terminal	10
	6	Connection number assigned by host to terminal	
	7 and 8	Number blocks transmitted upline	
	9 and 10	Number blocks received downline (good blocks only)	
	11 and 12	Number blocks received but not transmitted because of errors encountered in them	

[†] Corresponds to stream number of device; consoles on stream 0, other devices on zero or nonzero streams
[††] Corresponds to stream number of device; punches on nonzero streams, all other devices on stream 0
[†††] For non-batch devices only. Batch devices use record accounting instead of statistics. See appendix H.

TABLE B-4. ADDRESS TABLE (Cont'd)

Starting Address	Word	Mnemonic	Contents[†]	
	C_{16}	NAPORT	Port table address	Multiplex Subsystem
	D_{16}	BQCIB	Circular input buffer (CIB)	
	E_{16}		Unused	
	F_{16}	CGLCBS	Line control blocks (LCB)	Lines and TIPs
	10_{16}	CHSUBLCB	SUB LCBs	
	11_{16}	CGTCBS	Terminal control blocks (TCB)	
	12_{16}	BJTIPTYPT	TIP type table	
	13_{16}	NJTECT	Terminal characteristics table	
	14_{16}		Unused	
	15_{16}	NPINTAB	Initialization status table	Initialization Information
	16_{16}		Unused	
	17_{16}	CCPVER	CCI version ~~address~~	
	18_{16}	CCPCYC	CCI cycle ~~address~~	
	19_{16}	CCPLEV	CCI level ~~address~~	

[†] Contents of table are displayed at end of a successful initialization.

TABLE B-5. STATISTICS MESSAGE INTERPRETATION

Message Type	Message Interpretation	
	Bytes	Contents
NPU statistics	1 and 2	Number of upline service messages generated.
	3 and 4	Number of downline service messages processed.
	5 and 6	Number of bad service messages received.
	7 and 8	Number of blocks discarded because of a bad network address.
	9 and 10	Number of blocks discarded because of bad format.
	11 and 12	Number of times NPU ran at regulation level 4 (no regulation).
	13 and 14	Number of times NPU ran at regulation level 3, described in section 2.
	15 and 16	Number of times NPU ran at regulation level 2, described in section 2.
	17 and 18	Number of times NPU ran at regulation level 1, described in section 2.
	19 and 20	Number of times NPU ran at regulation level 0, described in section 2.
	21 and 22	Number of network assurance protocol timeouts.

TABLE B-5. STATISTICS MESSAGE INTERPRETATION (Cont'd)

Message Type	Message Interpretation		
	Bytes	Contents	
Line statistics	1	Port number of the communication line.	
	2 thru 4	Always zero.	
	5 and 6	Number of blocks transmitted upline.	
	7 and 8	Number of blocks received downline.	
	9 and 10	Number of characters transmitted upline in good blocks.	
	11 and 12	Number of characters received downline in good blocks.	
Terminal statistics	1	Port number of the communication line servicing the terminal.	
	2	Always zero.	
	3	Cluster address of the terminal, as follows:	
		Mode 4 terminals	70_{16} thru $7F_{16}$
		Teletypewriter, HASP, 2780 and 3780 terminals	0
	4	Terminal address of the terminal, as follows:	
		Mode 4A terminals	60_{16}
		Mode 4C terminals	61_{16} thru $6F_{16}$
		Teletypewriter terminals	0
		HASP terminals	0 thru 7[†]
		2780 and 3780 terminals	$0, 12_{16}, 13_{16}$[††]
	5	Upper three bits contain the device type of the terminal, as follows:	
		Console	0
		Card reader	1
		Line printer	2
		Card punch	3
		Nonimpact printer or plotter	4
		Lower five bits contain the terminal class of the terminal, as follows:	
		Teletypewriter terminal	1
		2780 terminal	7
		3780 terminal	8
		HASP terminal	9
		Mode 4 terminal	10
	6	Connection number assigned by the host to the terminal.	
	7 and 8	Number of blocks transmitted upline.	
	9 and 10	Number of blocks received downline (good blocks only).	
	11 and 12	Number of blocks received but not transmitted because of errors encountered in them.	

[†] Corresponds to the stream number of the device; consoles are on stream 0, other devices are on zero or nonzero streams.

[††] Corresponds to the stream number of the device; punches are on nonzero streams, all other devices are on stream 0.

GLOSSARY C

ACCOUNTING DATA -
Data kept for each batch input or output data stream. This data is a count of the number of records transferred to/from a batch device. The records can be input cards from a card reader, output cards to a card punch, or print lines to a printer. The data is kept for each remote job entry and is sent to the host when the job's EOI card is detected. Accounting data can be used to bill customers for I/O equipment usage.

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 low-speed 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.

AUTO-RECOGNITION -
A capability offered to some terminals which allows the TIP to generate some device characteristics for the terminal, rather than having the terminal generate the information for itself.

-B-

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 one or more words (2 bytes/word) and contains sufficient information to identify the type of block, its origin, destination, and routing. Differing block protocols apply to the host/NPU and the NPU/terminal interfaces.

BLOCK PROTOCOL -
The protocol governing block transfers of information between the host and the local 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. Ordinarily no action on the data is taken until the buffer is filled. 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 to new tasks. As the buffer level falls toward 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.

-C-

CASSETTE -
The magnetic tape device in an NPU used for bootstrap loading of off-line 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 multiplexer. 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 hardware driver that controls the multiplex subsystem.

COMMON AREA -
Area of main memory dedicated to system and global data. These are usually below address 200016.

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 (line) used to connect the terminal through the NPU to the host.

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

CONTENTION -
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. All protocols contain logic to resolve the contention situation.

CONTROL BLOCK -
(1) The type of block used to transmit control (as opposed to data) information.
(2) A block assigned for special configuration or status purposes in the NPU. The major blocks are line control blocks (LCB), logical link control blocks (LLCB), terminal control blocks (TCB), worklist control blocks (WLCB), buffer maintenance control blocks (BCB), multiplexer 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. The output is normally a CCI program in assembler format 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 protocol.

-D-

DATA -
Information processed by the network or some components of the network. Data usually has the form of messages, but commands and status are frequently transmitted using the same information packets as data (for instance, system messages).

DATA COMPRESSION -
The technique of transmitting a sequence of identical characters as a control character and a number representing the length of the sequence. HASP and Mode 4 protocols support data compression, as do other terminal formats.

DATA SET -
A hardware interface that transforms analog data to digital data and vice versa.

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 or 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 for host/NPU buffered transfers.

DIRECTORY -
A table in CCI which 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.

-E-

-F-

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) and secondary function code (SFC). See appendix H for definitions of these codes.

-G-

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

-H-

HALF DUPLEX (HDX) -
A transmission mode allowing data transfer in one direction at a time. Normally a single set of data lines carries input, output, 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 -
Protocol which is 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 PROGRAM (HIP) -
The CCI program that handles block transfers across the host/local NPU interface. The HIP normally uses CCI block protocol.

-I-

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.

INTERFACE (NPU) -
The set of hardware and software that permits transfers between the NPU and an external device. There are four principal interfaces: to the host (block protocol in 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 allows 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.

-J-

-K-

-L-

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 touches 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.

-M-

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 -

A bit pattern used in the interrupt subsystem to check if an interrupt is of sufficiently high priority to be processed now. An interrupt mask (M) register is used in this processing.

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 3,072 words of 32-bit length. 1024 words are read only memory (ROM); the remaining 2048 words are random access memory (RAM) and are alterable. The ROM contains the emulator microprogram that allows use of CYBER 18 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 both Mode 4A 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 as follows:

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 OPSMON, which executes OPS level programs by scanning a table of programs that have pending tasks.

MULTILEAVING -

The technique of interleaving several similar data streams in one transmission stream, while preserving the identity of the data stream source or destination.

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.

-N-

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 that includes a set of one or more directly connected 255x series Communications Control INTERCOM (CCI) macromemory modules. These CCI programs buffer and transmit data between terminals and the host computer.

NODE -
A network element which creates, absorbs, switches, and/or buffers message blocks. Typical system nodes are INTERCOM in the host, and the coupler node of an NPU.

-O-

OFF-LINE DIAGNOSTICS -
Optional diagnostics for the NPU which require the NPU to be disconnected from the network.

ON-LINE 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 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.

OVERLAY AREA -
An area in upper main memory which is used to execute on-line diagnostics.

-P-

PAGING -
A method of executing programs and accessing data in the NPU main memory region above 65K. Paging is required to simplify 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.

PASCAL -
A high-level programming language used for CCI programs. Almost all CCI 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 directly to an NPU.

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 follows:

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 which 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 connected into a PRP prior to output to the device.

POINT OF INTERFACE 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 -
The action of checking CLAs 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.

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 codes.

PRIORITY LEVEL -
A set of 17 levels of processing in the NPU. 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 the 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.

-Q-

QUEUE -
A sequence of blocks, tables, messages, and so forth. Most NPU queues are maintained by leaving the queued elements in place and using tables of pointers to the next queued element. 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.

-R-

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 data. The host has one regulation scheme, the host and multiplex interface of a local NPU have another scheme. Some types of terminals (for instance, HASP workstations) can also regulate data. Data classifications are usually based on batch, interactive, and control message criteria.

RESPONSE MESSAGES -
A subclass of service (network control) messages directed to the host which are normally generated to respond to a service message from the host. Response messages normally contain the requested information or indicate the requested task has been started or performed. Error responses are sent when the NPU cannot deliver the information or start the task. A class of unsolicited response messages are generated by the NPU to report hardware failures.

ROUTING -
The process of sending data or commands through the NPU to the internal NPU process or to an external device (for instance, a terminal). The network logical address (DN, SN, and CN) is the primary criterion for routing. The NPU directories are used to accomplish routing.

-S-

SECONDARY FUNCTION CODE (SFC) -
See Function Codes.

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 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.

-T-

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. TCBs are dynamically assigned.

TERMINAL INTERFACE PROGRAMS (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.

-U-

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.

-V-

-W-

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 but a 12-bit byte 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 incharacter format. Therefore, workstation word size is transparent to the NPU.

WORKLIST PROCESSOR -
The base system program responsible for processing worklists from the firmware. Primarily used for CLA errors and statistics processing.

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.

-X-Y-Z-

CCI MNEMONICS D

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.

Mnemonic	Meaning
ABL	Allowable block limit
ACK0	Acknowledge block (HASP protocol)
A/Q	The A/Q (low speed) I/O channel of the NPU
ASCII	American Standard Code for Information Interchange
ASYNC	Asynchronous
BACK	Acknowledgement block
BCB	Block control byte (HASP protocol)
BFC	Block flow control
BFR	Buffer
BLK	Message block
BRK	Break block
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 block
CN	Connection number
CND	Connection number directory
CR	Carriage return
CRC	Cyclic redundancy checksum
DBC	Data block clarifier (for blocks/SVM)
DCB	Diagnostics control block
DCD	Data carrier detected
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 protocol)
EOF	End of file
EOI	End of information (6/7/8/9 punch or /*EOI for HASP and 2780/3780 terminals)
EOM	End of message
EOR	End of record (7/8/9 punch or optionally /*EOR for HASP and 2780/3780 terminals)
ETB	End of block
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 Program
HCP	Host Communications Processor (alternate name for NPU)
HDLC	High level data link control
HDX	Half duplex
HIP	Host Interface Program
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 (LCD)
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
MM	Main memory
MPLINK	The CYBER Cross System linking editor
MSG	Message block
MTI	Message type indicators (Mode 4 protocol)
M4	Mode 4
MD4	Mode 4
NAK	Negative acknowledgement block (HASP protocol)

NBL	Network block limit
NL	Number of lines
NPINTAB	NPU initialization table
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	Padding element
PFC	Primary function code (for SVM)
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 (for SVM) 3. Response code (same as reason code)
RCB	Record control byte (HASP protocol)
RCV	Receive state
RL	Regulation level
RM	Response message (SM)
RS	Reverse supervision (block protocol)
RST	Reset block
RT	Record type
SCB	String control byte (HASP protocol)
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)	
STRT	Start data block	
STX	Start of text	
SVM	Service module for processing service messages	
SYNC	Synchronizing element	
TA	Terminal address	
TC	Terminal class	
TCB	Terminal control block in NPU	
TDP	Time dependent program	
TIP	Terminal Interface Program in NPU	
TO	Timeout	
TOT	Total number of status service messages	
TPCB	Text processor control block	
TT	Terminal type	
TTY	Teletypewriter (asynchronous device)	
VAR	PASCAL variable	
WL	Worklist	
WLCB	Worklist control block	
WLE	Worklist entry	
WLP	Worklist processor	
X-OFF	Stop tape character	(asynchronous protocol)
X-ON	Start tape character	

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.

		Program Name	
0000_{16}	Jump to BEGINX	ZEROX	
0100_{16}	Interrupt trap locations	PBINTRP	
0150_{16}	Address pointer table	Addressess	
0170_{16}	Console interrupt routines		
$0D70_{16}$	PASCAL globals	GLOBL$	
$1D50_{16}$	Assembly language routines		
$23BC_{16}$	State programs		
$35A3_{16}$	PASCAL programs		
$7F00_{16}$	ID table	PIDTBL	
8000_{16}	Circular input buffer		Set up at initialization time
8200_{16}	Line port table		
	Line control blocks		
			Becomes buffers when needed
$D980_{16}$	Set up stack, go to PINIT	MAIN$	
$D998_{16}$	LOAD R1, R2, R3, R4, go to MAIN$	BEGINX	
	Part I of Initialization programs		
$DE7F_{16}$	Initialize system	PINIT	
	Part II of Initialization programs		
$EFA0_{16}$	Initialize last of buffer	PIBUF2	
$F000_{16}$	System Paged Overlay service module		

Figure E-1. Sample Main Memory Map

CCI NAMING CONVENTIONS F

The following naming conventions for CCI PASCAL program labels should be regarded as guidelines rather than as strict requirements.

The general format of a label is

pirrrrssss

where the usual length is six bytes, but additional bytes can be used.

p values are:	A thru O	Global data
	P	Procedure or function
	Q thru W	Local data
	X thru Z	Non-CDC user
i values are:	0	Transparent or not tied down
	1 thru 9	Not a structure
	A thru 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 software)
	N	Network communications programs
	T	TIP or HIP

For names of types, variables, fields, and so forth, the following prefixes are used:

AO...	OPS-level workcodes
BA...	Overlay
BC...	Physical or logical request packet (PRP or LRP)
BF...	Buffer
BJ ...	TIP type table
BL ...	Logical link control block (LLCB)
BS ...	Terminal CB (TCB)
BT ...	Timing, monitor controlled
BW...	Intermediate array for worklist
BY...	Worklist CB (WLCB)
BZ ...	Line control block
D ...	Service module
J ...	Input/Output
JC ...	Logical or physical I/O request packet
LD...	Load/dump
M ...	Multiplex subsystem
MM ...	Event worklists (multiplex subsystem)
N ...	Multiplex subsystem
NA...	Port table
NB...	Line types or text processing CB (TPCB)
NC...	Multiplex line CB (MLCB)
NJ ...	Terminal characteristics
NK...	Multiplexer command driver inputs
NZ...	Diagnostics CB (DCB)
SI ...	System interfaces

TERMINAL MESSAGES

Interactive terminals controlled by the NPU receive preformatted messages when the host becomes unavailable, and when the host is again available. When the host is unavailable, the following message appears:

HOST UNAVAILABLE

As a reply to a message sent upline to the host after the host unavailable message has been sent to the terminal, the following message appears at terminals serviced by the TTY TIP:

INPUT STOPPED

When the host again becomes available, the following message appears:

HOST AVAILABLE

When regulation ends, the following message appears:

RESUME INPUT

BLOCK AND SERVICE MESSAGE SUMMARY

BLOCK TYPES

The unit of 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.

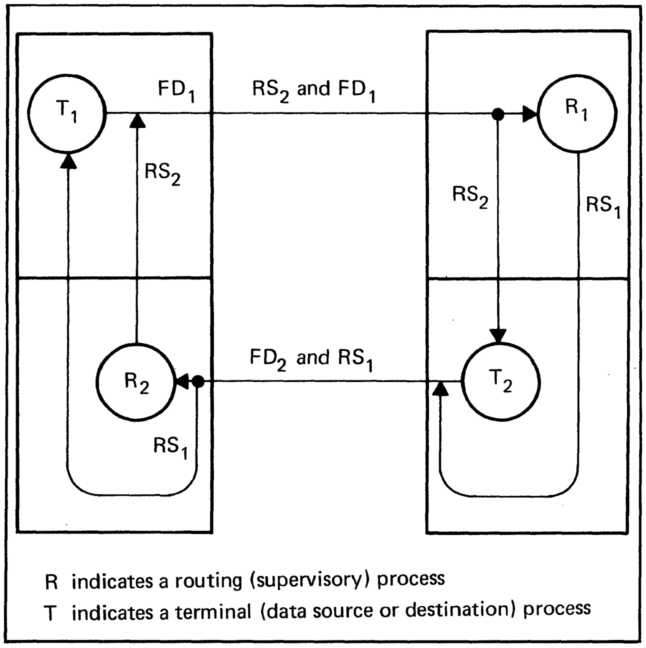
Figure H-1 shows that the block flow control interface between two logically connected processes can be envisioned as two simultaneously active communications paths. The figure also shows that the procedure is fully symmetric. Sequence is maintained on each of the four paths but not between the separate 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.



T_1
FD_1
RS_2 and FD_1
R_1
RS_2
RS_2
RS_1
FD_2 and RS_1
R_2
T_2
RS_1
R indicates a routing (supervisory) process
T indicates a terminal (data source or destination) process

Figure H-1. Communications Paths for Block Flow Control

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 among block type values in the block header, traffic type, and block function are shown in table H-1. The block type values 0, 5 through 9, and 12 through 15 are reserved by CDC; block type values 10 and 11 are unused.

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 eight-byte header. The eight-byte header consists of the standard four-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.

TABLE H-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 Acknowledgement	RS	This block is an acknowledgement 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 connection number.

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 header.

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 system 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 asynchronous 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. Table H-2 summarizes the general format of CMD blocks used on nonzero connections within CCI 3. 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.

CMD blocks on nonzero connections contain no information other than the function code bytes. CMD blocks containing service messages include additional bytes of information, as described in the following paragraphs.

SERVICE MESSAGES

Table H-3 lists all service messages recognized as part of the CCI block protocol. The primary and secondary function code columns contain the code values appearing in the first and second bytes of these messages after the block header bytes. Table H-4 lists the contents of the remainder of each service message; byte numbering in table H-4 treats the function code bytes as prefixing information and begins with the first message byte following the secondary function code byte.

The processing of all request service messages using a primary function code of 3 is driven by a control block descriptor string. There is one such descriptor string for each type of configurable control block in the NPU. This descriptor string equates a field number (FN) to a field position within the control block and allows the field value (FV) to be correctly assigned. Additionally, an optional field action (FA) can be defined and associated with a field number. This field action allows such features as validating the field value, assigning chains to other structures, and performing other associated actions required by the field.

The functional use of all service messages is described in detail in the CCI 3 system programmer's reference manual. Mnemonics for the service messages are also described there, and control block field numbers and field values are explained.

BLOCK PROTOCOL FLOW

Figures H-2 through H-10 give examples of the block protocol sequences between the host and NPU for normal and abnormal cases. These protocol sequences must be preserved when site-originated modifications or extensions are made to the CCI. The notation used in the diagrams is as follows:

CN	Connection number
SM	Service message

I	Interactive input or output (nonzero connection)
B,I	Batch input (nonzero connection)
B,O	Batch output (nonzero connection)
CMD	Block protocol command block type (see table H-1).
BLK	Any message block except the last (block protocol)
MSG	The last message block (block protocol)
BACK	Block protocol acknowledgment block type
COMMAND	INTERCOM prompting message to terminal user

TABLE H-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 Initiate, transparent Resume	0 1 2	Start input from batch device Start input from batch device Start input from batch device
RS	Stop input	2	Terminate Suspend	0 1	Discard data and stop polling Stop polling and wait
FD	Input stopped	3	End Break 1 Break 2	0 1 2	Normal end Reason for break defined by TIP Reason for break defined by TIP
RD	Input started	4	Restart after stop	0	Interactive resume after break (status only)
RS	Output stopped	5	Break 1 Break 2 Break 3 Break 4 Break 5 Break 6	0 1 2 3 4 5	Reason for break defined by TIP Reason for break defined by TIP Reason for break defined by TIP Reason for break defined by TIP Reason for break defined by TIP 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 resume output stream
RS	BSN error	9	NPU has detected block sequence number out of order	0	Diagnostic purposes only
FD	Input accounting data	A	Overflow	0	Input active count overflow
FD	Input accounting data	A	End-of-Data Indication (EDI)	1	EDI seen, final accounting data for this file
FD	Output accounting data	B	Overflow	0	Output active, count overflow

TABLE H-3. COMMAND BLOCKS USED ON CONNECTION ZERO

Traffic Type	Primary Function	Primary Function Code	Secondary Function	Secondary Function Code	Use
FD	NPU initialized	1	Status to host	2	Network operation is possible
FD	Configure line	3	Request	0	Create new LCB
			Normal response	40_{16}	New LCB created
			Error response	80_{16}	No new LCB created
FD	Delete line	3	Request	1	Delete existing LCB
			Normal response	41_{16}	LCB deleted
			Error response	81_{16}	LCB not deleted
FD	Configure terminal	3	Request	2	Create new TCB
			Normal response	42_{16}	New TCB created
			Error response	82_{16}	No new TCB created
FD	Reconfigure terminal	3	Request	3	Change TCB
			Normal response	43_{16}	TCB changed
			Error response	83_{16}	TCB not changed
FD	Delete terminal	3	Request	4	Delete existing TCB
			Normal response	44_{16}	TCB deleted
			Error response	84_{16}	TCB not deleted
FD	Diagnostic test control	5	Reserved by CDC	All	Online diagnostics
FD	Line status	6	Request	2	Return status in response messages
			Normal response	42_{16}	Indicates status
			Error response	82_{16}	Status cannot be returned
			Unsolicited response	2	Auto-recognition lines
FD	Terminal status	6	Request	3	Return status in response message
			Normal response	43_{16}	Indicates status
			Error response	83_{16}	Status cannot be returned
			Unsolicited response	3	Auto-recognition lines
FD	Line count	6	Request	5	Return number of configured lines
			Response	45_{16}	Indicates number
FD	Statistics	7	NPU	0	Indicates operational counter contents
			Line	1	Indicates operational counter contents
			Terminal	2	Indicates operational counter contents

TABLE H-3. COMMAND BLOCKS USED ON CONNECTION ZERO (Cont'd)

Traffic Type	Primary Function	Primary Function Code	Secondary Function	Secondary Function Code	Use
FD	Line control	8	Enable request	0	Make line operational
			Normal response	40_{16}	Line operational
			Error response	80_{16}	Line cannot be operational
			Disable request	1	Make line inoperative
			Normal response	41_{16}	Line inoperative
			Error response	81_{16}	Line cannot be made inoperative
			Disconnect request	2	Combined disable/-enable for dial-up lines
			Normal response	40_{16}	Line operational
			Error response (RC 4)	82_{16}	Line inoperative
			Error response (RC 4)	80_{16}	Line inoperative
FD	Customer engineering error message	A_{16}	Diagnosed error	0	Sent to host engineering file
FD	Host broadcast	C_{16}	Request, single terminal	0	Send text from host operator to interactive terminal
			Normal response, single terminal request	40_{16}	Text delivered
			Error response, single terminal request	80_{16}	Text not delivered
			Request, all terminals	1	Send text from host operator to all interactive terminals
FD			Normal response, all terminals request	41_{16}	Text delivered
			Error response, all terminals request	81_{16}	Text not delivered

TABLE H-4. SERVICE MESSAGE INTERPRETATION

Message Type	Bytes	Contents
NPU initialized	1	CCI version number on load file used.
	2	CCI build cycle on load file used.
	3	CCI PSR summary level of load file used.
Configure line request	1	Port number (CLA thumbwheel setting) of the communication line.
	2	Always zero.
	3	Line type number, as defined in table 1-1.
	4	If the upper bit is zero, the line is not configured for auto-recognition; if the upper bit is one, the line is configured for auto-recognition. Bits 6 through 3 indicate the type of TIP servicing the line, as follows: 0 Not applicable 1 Serviced by TTY TIP 2 Serviced by Mode 4 TIP 3 Serviced by HASP TIP 4 Serviced by 2780/3780 TIP, or by HASP TIP on auto-recognition lines Bits 2 through 0 are zero.
	5	Field number byte, containing a 5.
	6	Field value byte, containing zero.
	7	Field number byte, containing a 21. This byte is omitted for lines serviced by the HASP, Mode 4 or 2780/3780 TIPs, and can be omitted for nonauto-recognition lines serviced by the TTY TIP.
	8	Field value byte, when the seventh byte is present. This byte contains the line speed index number, as follows: 0 110 baud 2 150 baud 3 300 baud 4 600 baud 5 1200 baud 6 2400 baud 7 4800 baud 8 9600 baud
Configure line normal response	1 thru 4	Same as for request message.
	5	Always zero.
Configure line error response	1 thru 4	Same as for normal response.
	5	Reason code for the response, as follows: 1 Request contained an invalid field number and field value pair 2 Request contained an invalid port number 3 LCB is already configured for the line 4 Request contained an invalid line type number 5 Request contained an invalid byte 4
	6 and 7	Omitted code for the response, as follows:
Delete line request	1	Port number (CLA thumbwheel setting) of the communication line.
	2	Always zero.

TABLE H-4. SERVICE MESSAGE INTERPRETATION (Contd)

Message Type	Bytes	Contents
Delete line normal response	1 and 2	Same as request message.
	3	Always zero.
Delete line error response	1 and 2	Same as for normal response.
	3	Reason code for the response, as follows: 2 Request contained an invalid port number 3 LCB is already deleted
Configure terminal request	1	Port number (CLA thumbwheel setting) of the communication line.
	2	Always zero.
	3	Cluster address of the terminal, as follows: 70_{16} thru $7F_{16}$ Mode 4 terminals 0 Teletypewriter, HASP, 2780 and 3780 terminals
	4	Terminal address of the terminal, as follows: 60_{16} Mode 4A terminals 61_{16} thru $6F_{16}$ Mode 4C terminals 0 Teletypewriter terminals 0 thru 7† HASP terminals 0, 12_{16}, 13_{16}†† 2780 and 3780 terminals
	5	Upper 3 bits contain the device type of the terminal, as follows: 0 Console 1 Card reader 2 Line printer 3 Card punch 4 Nonimpact printer or plotter Lower 5 bits contain the terminal class of the terminal, as follows: 1 Teletypewriter terminal 7 2780 terminal 8 3780 terminal 9 HASP terminal 10 Mode 4 terminal
	6	Connection number assigned by the host to the terminal.
	7 thru n	Field number and field value byte pairs.
Configure terminal normal response	1 thru 6	Same as for request message.

TABLE H-4. SERVICE MESSAGE INTERPRETATION (Contd)

Message Type	Bytes	Contents
	7	Reason code for the response, as follows: 1 Request contained an invalid field number and field value pair 2 Request contained an invalid port number of terminal address 3 TCB already exists for the terminal 4 No buffer exists for the TCB 5 Request contained an invalid terminal class for this line 6 Line is inoperative or not enabled 8 Logical link is not established for the connection number in the request 9 Request contained a connection number already in use 10 The console is not configured and the request is for a Mode 4 batch device
	8 and 9	Omitted unless byte 7 contains a 1 or a 9. Then these bytes contain a field number and field value pair.
Reconfigure terminal request	1 thru 6	Same as for configure terminal request message.
Reconfigure terminal normal response	1 thru 7	Same as for configure terminal normal response.
Reconfigure terminal error response	1 thru 6	Same as for configure terminal error response.
	7	Reason code for the response, as follows: 1 Request contained an invalid field number and field value pair 2 Request contained an invalid port number or terminal address 3 TCB does not yet exist for the terminal
	8 and 9	Omitted unless byte 7 contains a 1. Then these bytes contain the invalid field number and field value pair.
Delete terminal request	1	Port number (CLA thumbwheel setting) of the communication line.
	2	Always zero.
	3	Cluster address of the terminal, as follows: 70_{16} thru $7F_{16}$ Mode 4 terminals 0 Teletypewriter, HASP, 2780 and 3780 terminals
	4	Terminal address of the terminal, as follows: 60_{16} Mode 4A terminals 61_{16} thru $6F_{16}$ Mode 4C terminals 0 Teletypewriter terminals 0 thru 7 † HASP terminals 0, 12_{16}', 13_{16}†† 2780 and 3780 terminals

TABLE H-4. SERVICE MESSAGE INTERPRETATION (Contd)

Message Type	Bytes	Contents
	5	Upper three bits contain the device type of the terminal, as follows: 0 Console 1 Card reader 2 Line printer 3 Card punch 4 Nonimpact printer or plotter Lower five bits contain the terminal class of the terminal, as follows: 1 Teletypewriter terminal 7 2780 terminal 8 3780 terminal 9 HASP terminal 10 Mode 4 terminal
	6	Connection number assigned by the host to the terminal.
Delete terminal normal response	1 thru 6	Same as for request message.
	7	Always zero.
Delete terminal error response	1 thru 6	Same as for normal response.
	7	Reason code for the response, as follows: 2 Request contains an invalid port number 3 TCB does not exist for the connection number specified in the request
Line status request	None	Return status for all communication lines.
	1	Port number (CLA thumbwheel setting) of the communication line.
	2	Always zero.
	3	Reason code explaining the status, as follows: 0 Line is operational 4 Line is inoperative 5 There is no ring indicator or auto-recognition in progress 6 Stopped (CLA not responding; contact a customer engineer)
	4	Line type code for the line, as described in table 1-1.
	5	Configuration state of the line known by the service module, as follows: 0 LCB is not configured 1 LCB is configured but not enabled 2 Enable requested to TIP 3 Line is operational, no TCBs exist 4 Line is operational, TCBs are configured 5 Disable requested to TIP 6 Line is inoperative, no TCBs exist 7 Line is inoperative, TCBs are configured 8 Disconnect requested to TIP 9 Waiting for ring indicator
	6	Number of terminals detected on the line.
Line status error response	1 and 2	Same as for normal response
	3	Reason code explaining the response, as follows: 1 Request contained an invalid port number or there are no lines configured

TABLE H-4. SERVICE MESSAGE INTERPRETATION (Contd)

Message Type	Bytes	Contents
		2 Line status request already in progress 3 Line is in an illegal configuration state
Line status unsolicited response	1 thru 6	Same a for normal and error responses.
	7	If the upper bit is zero, the line is not configured for auto-recognition; if the upper bit is one, the line is configured for auto-recognition. Bits 6 through 3 indicate the type of TIP servicing the line, as follows: 0 Not applicable 1 Serviced by TTY TIP 2 Serviced by Mode 4 TIP 3 Serviced by HASP TIP 4 Serviced by 2780/3780 TIP, or by HASP TIP on auto-recognition lines Bits 2 through 0 indicate the sub- TIP type used for the line, as follows: 0 110 baud ASCII code TTY TIP, or HASP TIP, or 2780/3780 TIP 1 BCD code Mode 4A TIP 2 150 baud ASCII code TTY TIP, or ASCII code Mode 4A TIP 3 300 baud ASCII code TTY TIP, or ASCII code Mode 4C TIP
	8	Cluster address of the terminals auto-recognized, as follows: 70_{16} thru $7F_{16}$ Mode 4 terminals 0 Teletypewriter, HASP, 2780 and 3780 terminals
	9 thru n	One terminal address byte and one device type byte for each terminal on the line that can be detected by the TIP. These bytes are returned only for Mode 4 terminals, and up to 15 unique pairs are possible, as follows. For terminal address: 60_{16} Mode 4A terminals 61_{16} thru $6F_{16}$ Mode 4C terminals For device type, the upper three bits contain the device type: 0 Console 1 Card reader 2 Line printer 4 Nonimpact printer or plotter The lower five bits contain the terminal class of the terminal, which is 10.
Terminal status request	1	Port number (CLA thumbwheel setting) of the communication line.
	2	Always zero.
Terminal status normal response	1 and 2	Same as for request message.
	3	Cluster address of the terminal, as follows: 70_{16} thru $7F_{16}$ Mode 4 terminals 0 Teletypewriter, HASP, 2780 and 3780 terminals

TABLE H-4. SERVICE MESSAGE INTERPRETATION (Contd)

Message Type	Bytes	Contents
	4	Terminal address of the terminal, as follows: 60_{16} Mode 4A terminals 61_{16} thru $6F_{16}$ Mode 4C terminals 0 Teletypewriter terminals 0 thru 7† HASP terminals 0, 12_{16}, 13_{16}†† 2780 and 3780 terminals
	5	Upper three bits contain the device type of the terminal, as follows: 0 Console 1 Card reader 2 Line printer 3 Card punch 4 Nonimpact printer or plotter Lower five bits contain the terminal class of the terminal, as follows: 1 Teletypewriter terminal 7 2780 terminal 8 3780 terminal 9 HASP terminal 10 Mode 4 terminal
	6	Reason code explaining the status, as follows: 0 Terminal is operative 4 Terminal is inoperative 7 Source node number configured for the terminal. 8 Destination node number configured for the terminal. 9 Connection number configured for the terminal. 10 Total number of status service message responses to be sent for this request.
Terminal status error response	1 and 2	Same as for normal response.
	3	Reason code explaining the error response, as follows: 1 Request contained an invalid port number 2, 3 No terminals are configured on the line 4 Terminal status request is already in progress
Terminal status unsolicited response	1 thru 10	Same as for normal response.
Line count request	None	
Line count normal request	1	Number of configured lines.
NPU statistics		See appendix B, table B-5.
Line statistics		See appendix B, table B-5.

TABLE H-4. SERVICE MESSAGE INTERPRETATION (Contd)

Message Type	Bytes	Contents
Terminal statistics (non-batch)		See appendix B, table B-5.
Unit record accounting (batch)	1 and 2	Count of records input or records output (16-bit binary field, range 00-65K).
Enable line request	1	Port number (CLA thumbwheel setting) of the communication line.
	2	Always zero.
Enable line normal response (line is enabled)	1 and 2	Same as for request message.
	3	Reason code explaining the status, as follows: 0 Line is operational 4 Line is inoperative 5 There is no ring indicator or auto-recognition in progress 6 Stopped (CLA not responding; contact a customer engineer)
	4	Line type number of the line, as described in table 1-1.
	5	Configuration state of the line known to the service module, as follows: 0 LCB is not configured 1 LCB is configured but not enabled 2 Enable requested to TIP 3 Line is operational, no TCBs exist 4 Line is operational, TCBs are configured 5 Disable requested to TIP 6 Line is inoperative, no TCBs are configured 7 Line is inoperative, TCBs are configured 8 Disconnect requested to TIP 9 Waiting for ring indicator
	6	Always zero.
Enable line error response (line is not enabled)	1 and 2	Same as for normal response.
	3	Reason code explaining the error response, as follows: 1 Request contained an invalid port number or there is no line configured 2 A line status request is in progress 3 Line is in an illegal configuration state
Disable line request	1	Port number (CLA thumbwheel setting) of the communication line.
	2	Always zero.
Disable line normal response (line disabled)	1 and 2	Same as for request message.
	3	Always zero.
	4	Line type number for the line, as described in table 1-1.
	5	Configuration state of the line known to the service module, as follows: 0 LCB is not configured 1 LCB is configured but not enabled 2 Enable requested to TIP 3 Line is operational, no TCBs exist 4 Line is operational, TCBs are configured 5 Disable requested to TIP 6 Line is inoperative, no TCBs are configured 7 Line is inoperative, TCBs are configured

TABLE H-4. SERVICE MESSAGE INTERPRETATION (Contd)

Message Type	Bytes	Contents
		8 Disconnect requested to TIP 9 Waiting for ring indicator
	6	Number of terminals configured on the line.
Disable line error response	1 and 2	Same as for normal response.
	3	Reason code explaining the error response, as follows: 1 Request contained an invalid port number of there is no line configured 2 A line status request is in progress 3 Line is in an illegal configuration state
Disconnect line request	1	Port number (CLA thumbwheel setting) of the communication line.
	2	Always zero.
Disconnect line normal response		Response is an enable line normal response message.
Disconnect line error response	1 and 2	Same as for request message.
	3	Reason code explaining the error response, as follows: 1 Request contained in invalid port number or there is no line configured 2 A line status request is in progress 3 Line is in an illegal configuration state
CE error	1	Error code, as described in appendix B, table B-1.
	2 thru n	Data bytes, dependent on the error code (see appendix B).
Host broadcast, single terminal request	1	Port number (CLA thumbwheel setting) of the communication line.
	2	Always zero.
	3	Cluster address of the terminal, as follows: 70_{16} thru $7F_{16}$ Mode 4 terminals 0 Teletypewriter, HASP, 2780 and 3780 terminals
	4	Terminal address of the terminal, as follows: 60_{16} Mode 4A terminals 61_{16} thru $6F_{16}$ Mode 4C terminals 0 Teletypewriter terminals 0 thru 7† HASP terminals 0, 12_{16}, 13_{16}†† 2780 and 3780 terminals

TABLE H-4. SERVICE MESSAGE INTERPRETATION (Contd)

Message Type	Bytes	Contents
	5	Upper three bits contain the device type of the terminal, as follows: 0 Console 1 Card reader 2 Line printer 3 Card punch 4 Nonimpact printer or plotter Lower five bits contain the terminal class of the terminal, as follows: 1 Teletypewriter terminal 7 2780 terminal 8 3780 terminal 9 HASP terminal 10 Mode 4 terminal
	6 thru 31	Text in characters to be delivered to the terminal.
Host broadcast, single terminal error response	1 thru 5	Same as for request message.
	6	Reason code for the response, as follows: 1 Request contained an invalid port number 2 Request contained an invalid byte 5 3 Terminal identified in the request is not configured 4 Terminal identified in the request is inoperative 5 Host broadcast is already in progress
Host broadcast, all terminals request	1	Always zero.
	2	Always zero.
	3	Always 20_{16}, providing filler space for the data block clarifier byte.
	4 thru 29	Text in characters to be delivered to the terminals.
Host broadcast, all terminals normal response	1	Always zero.
Host broadcast, all terminals error response	1	Reason code explaining the error response as follows: 2 Broadcast already in progress

† Corresponds to the stream number of the device; consoles are on stream 0, other devices are on zero or nonzero streams.

†† Corresponds to the stream number of the device; punches are on nonzero streams, all other devices are on stream 0.

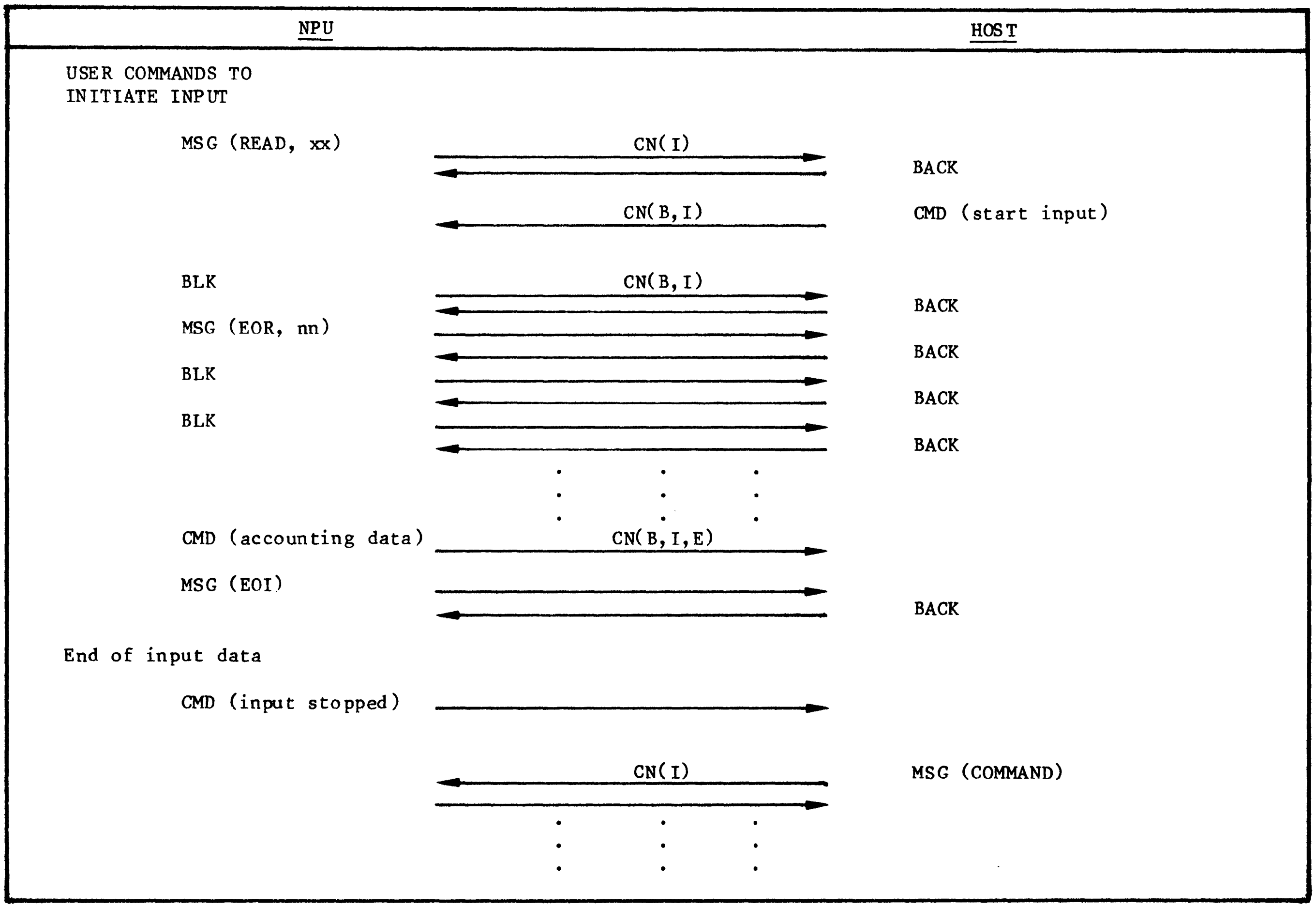


Figure H-2. General Batch Input Flow

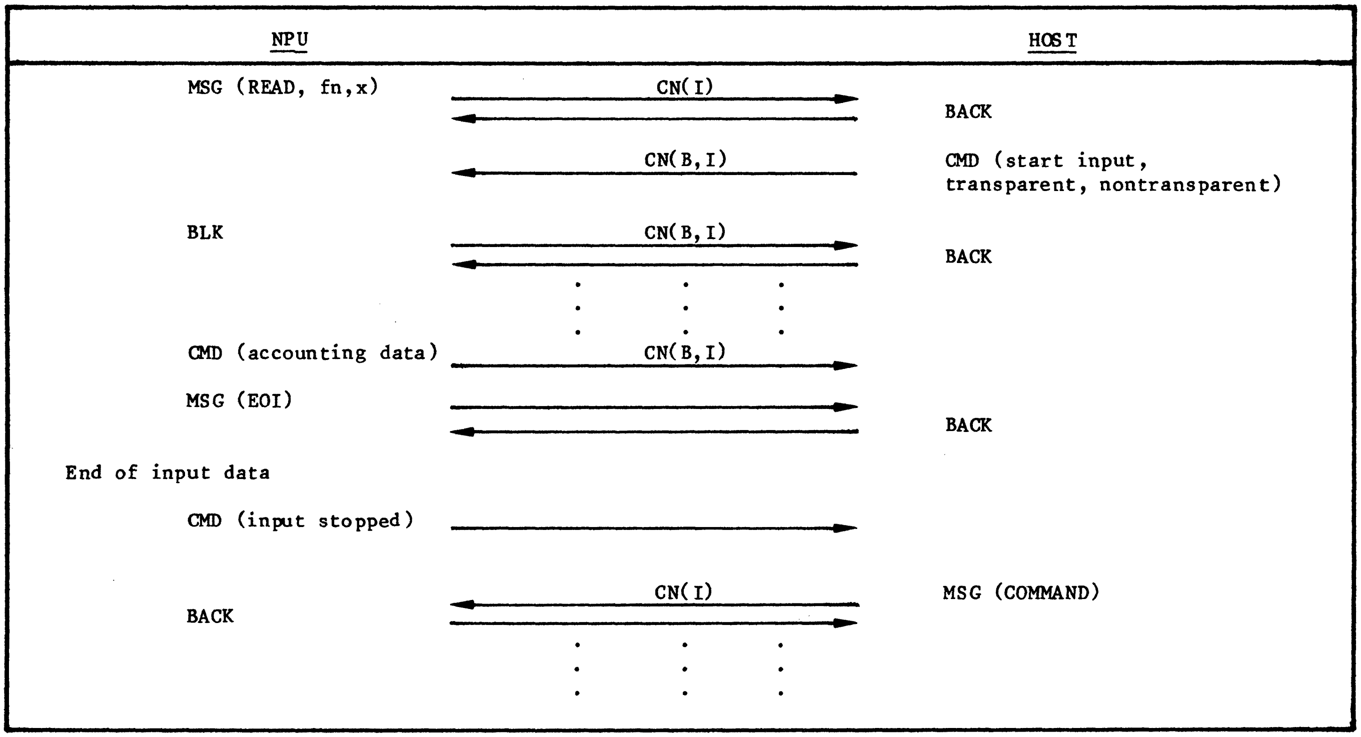


Figure H-3. INTERCOM READ, Filename, Mode Command

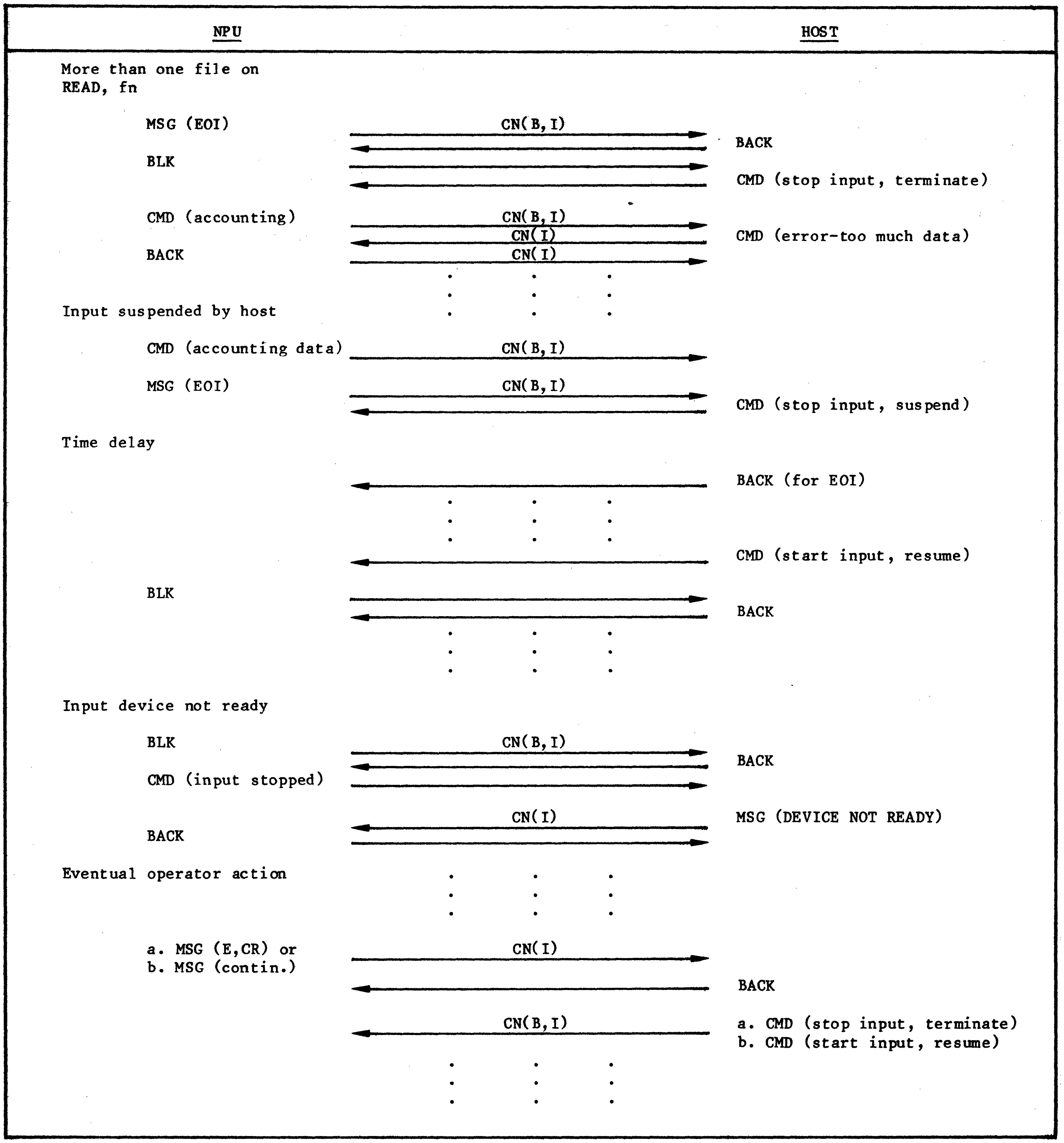


Figure H-4. Input Error Conditions

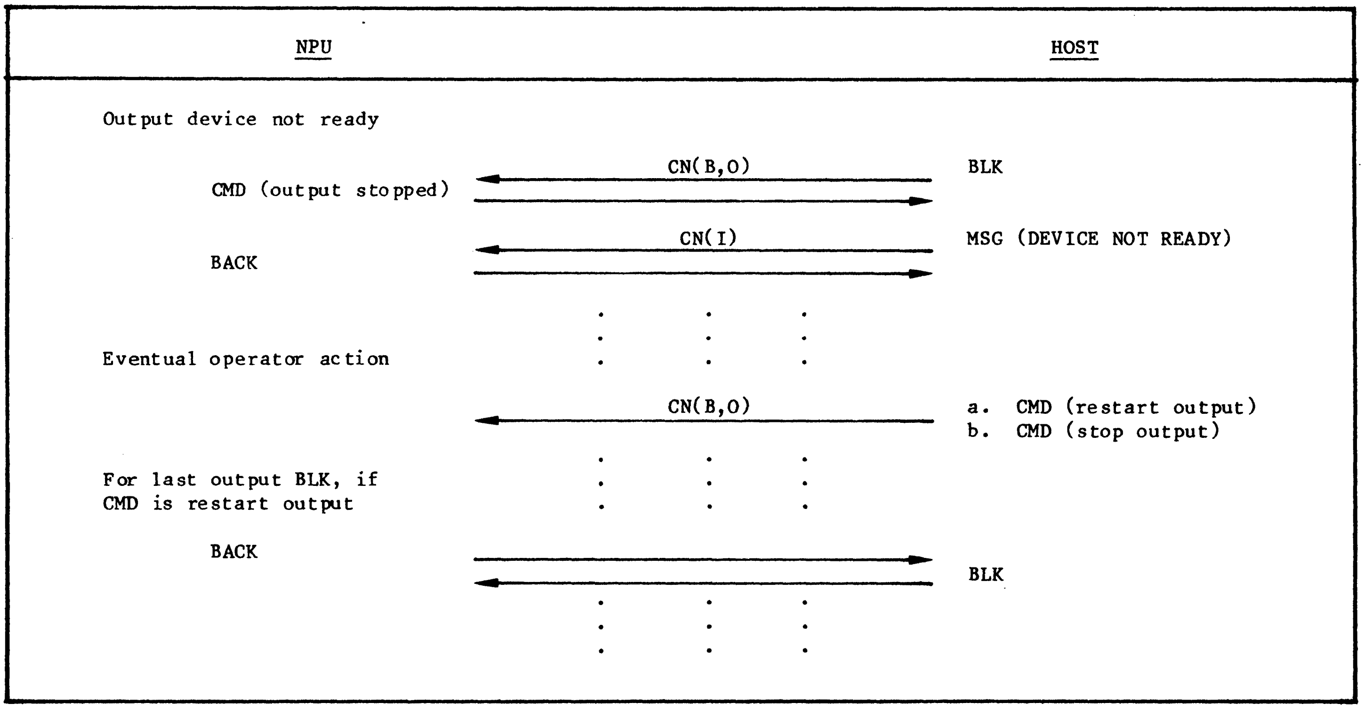


Figure H-5. Output Error Conditions

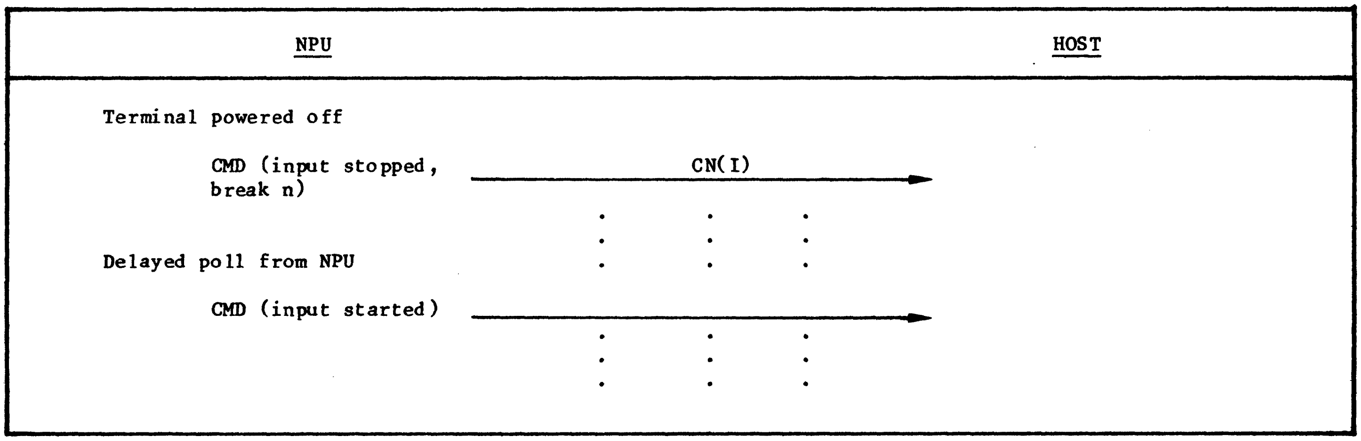


Figure H-6. General Errors

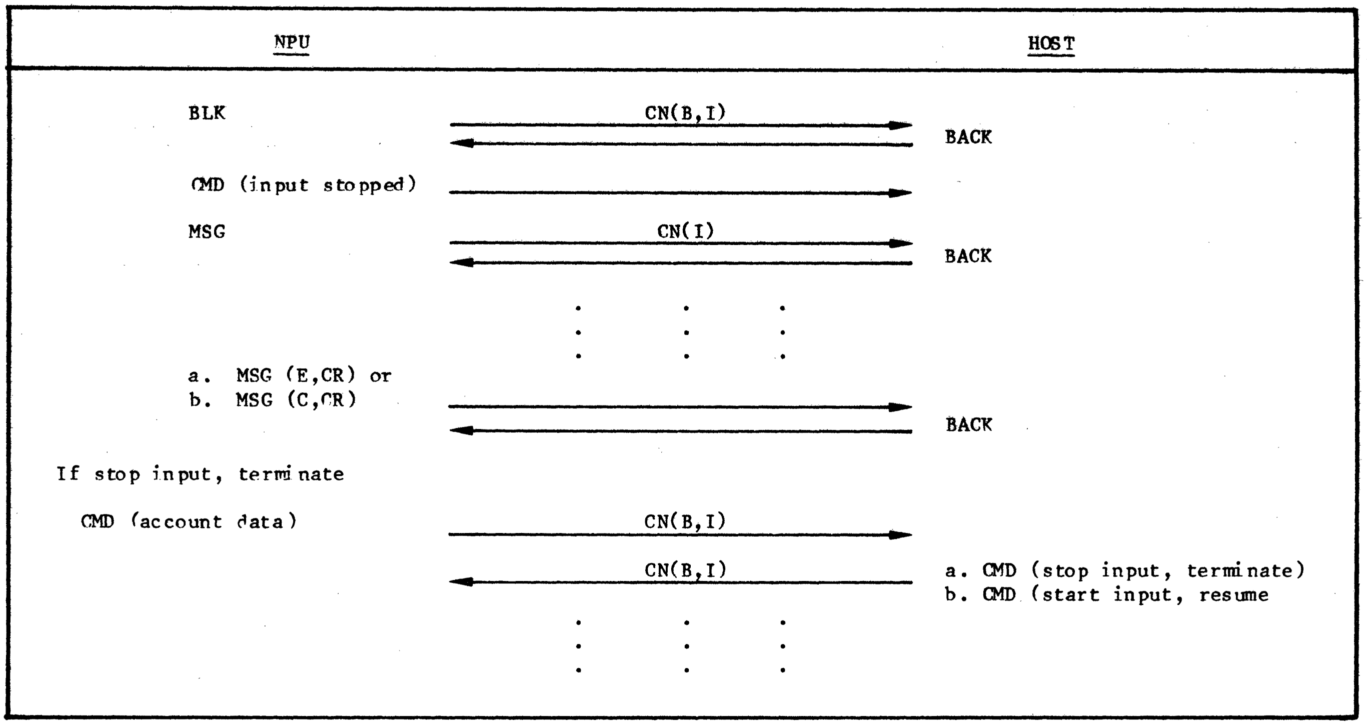


Figure H-7. Input Intervention

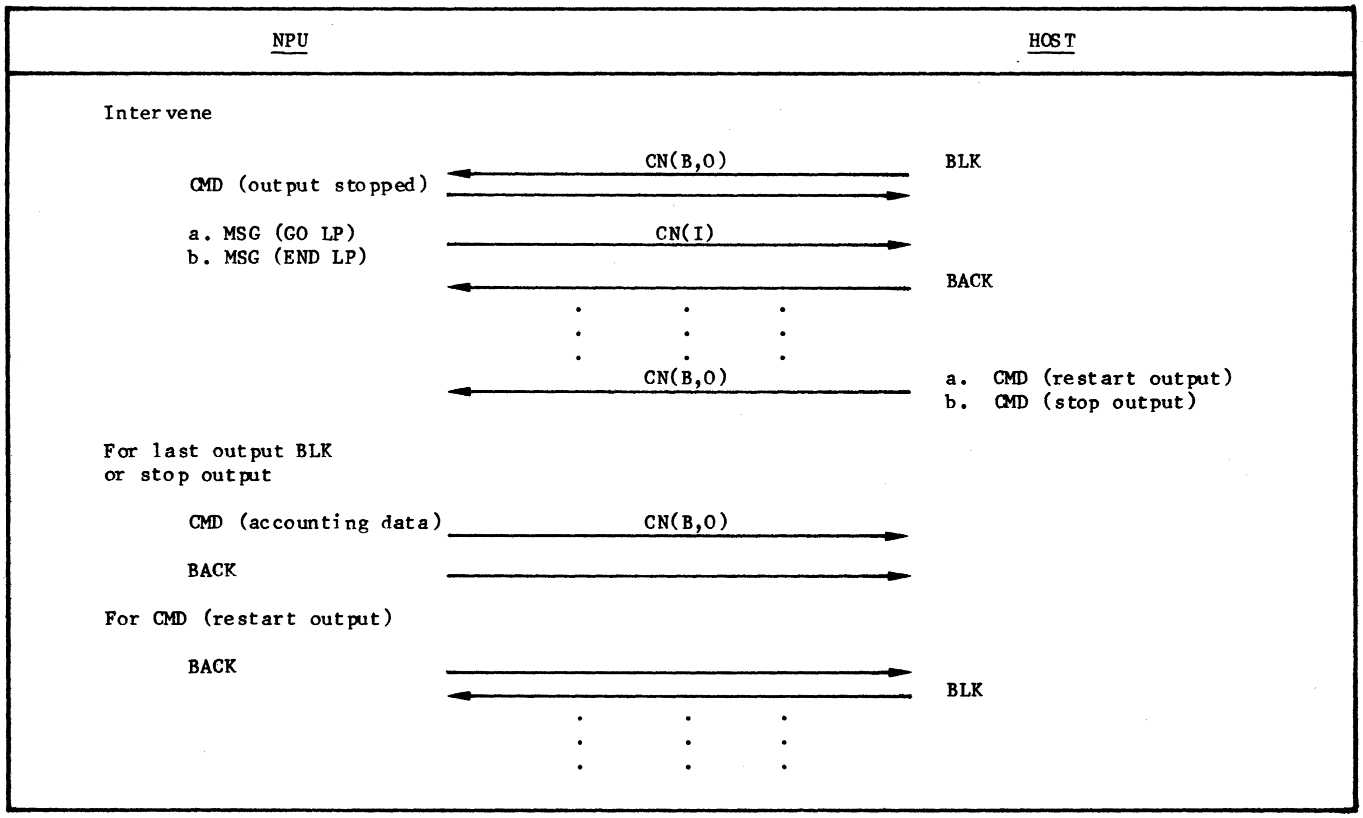


Figure H-8. Output Intervention

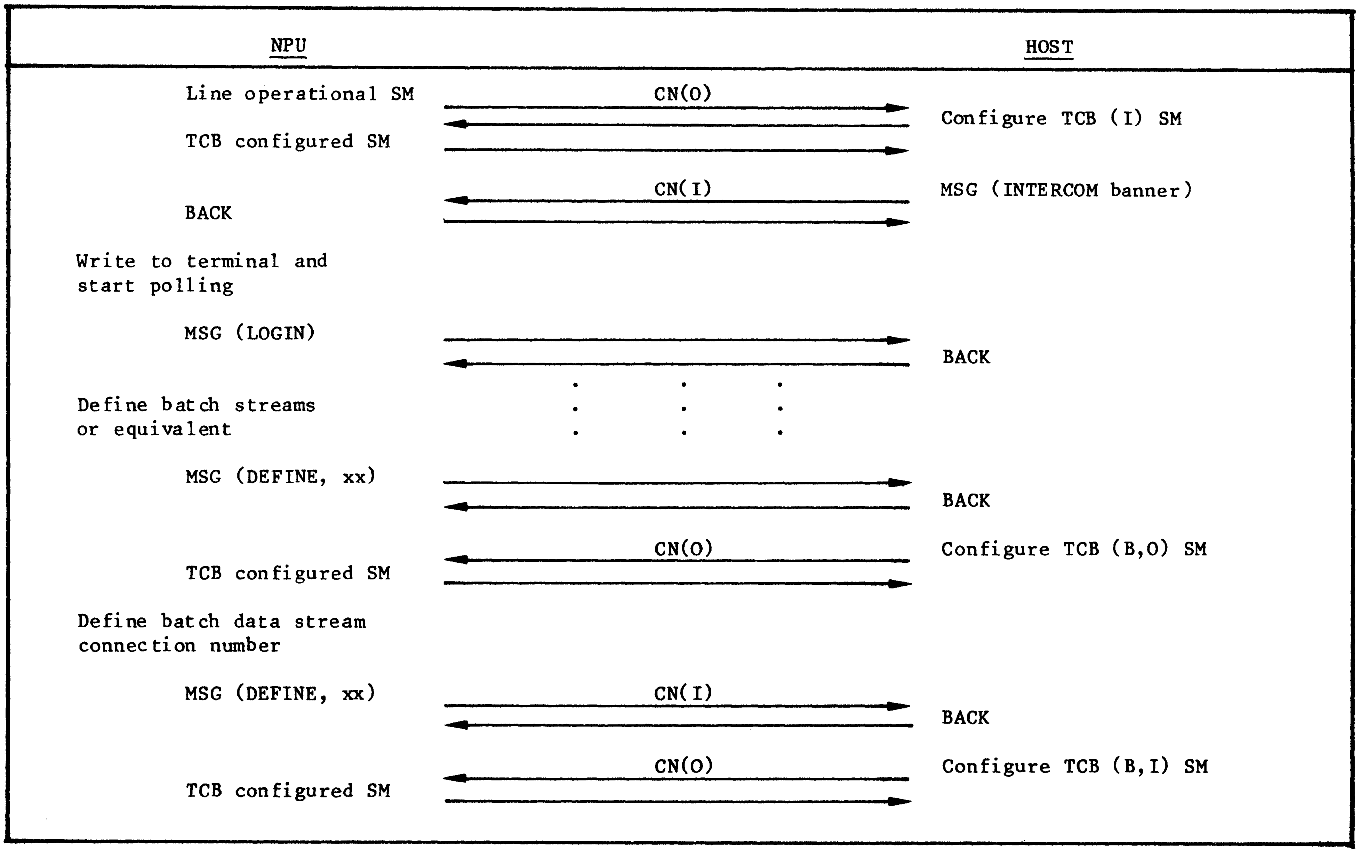


Figure H-9. Initialization

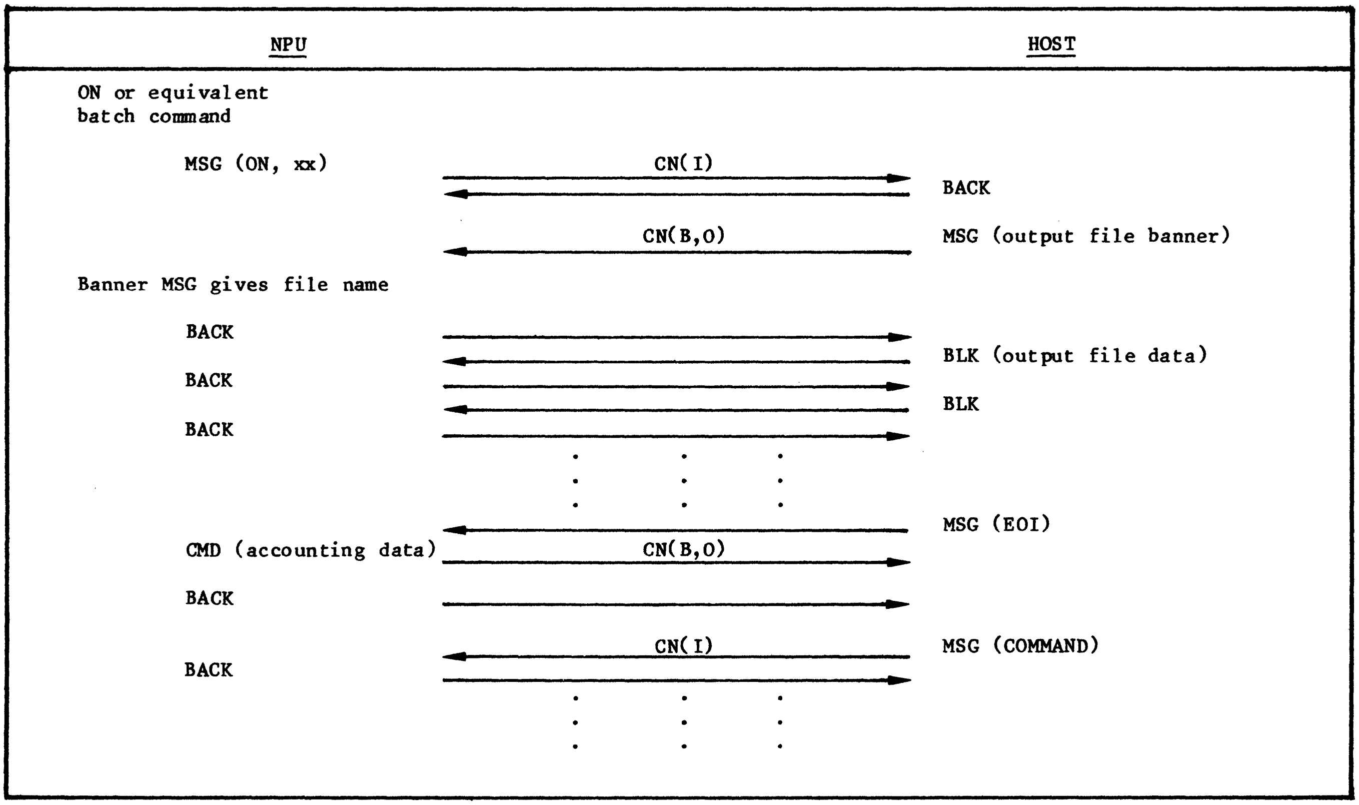


Figure H-10. Batch Output

INDEX

COMMENT SHEET

MANUAL TITLE: Communications Control Intercom
Version 3 Reference Manual

PUBLICATION NO.: 60471150 **REVISION:** C

NAME:____

COMPANY:____

STREET ADDRESS:____________________________________

CITY:____________________ **STATE:**__________ **ZIP CODE:**__________

This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments below (please include page number references).

CUT ALONG LINE

AA3419 REV. 4/79 PRINTED IN U.S.A.

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

FOLD ON DOTTED LINES AND TAPE

TAPE TAPE

FOLD FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 8241 MINNEAPOLIS, MINN.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION

Publications and Graphics Division
P. O. Box 4380-P
Anaheim, California 92803

CUT ALONG LINE

FOLD FOLD

CORPORATE HEADQUARTERS, P.O. BOX 0, MINNEAPOLIS, MINN. 55440 LITHO IN U.S.A.
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD

CONTROL DATA CORPORATION