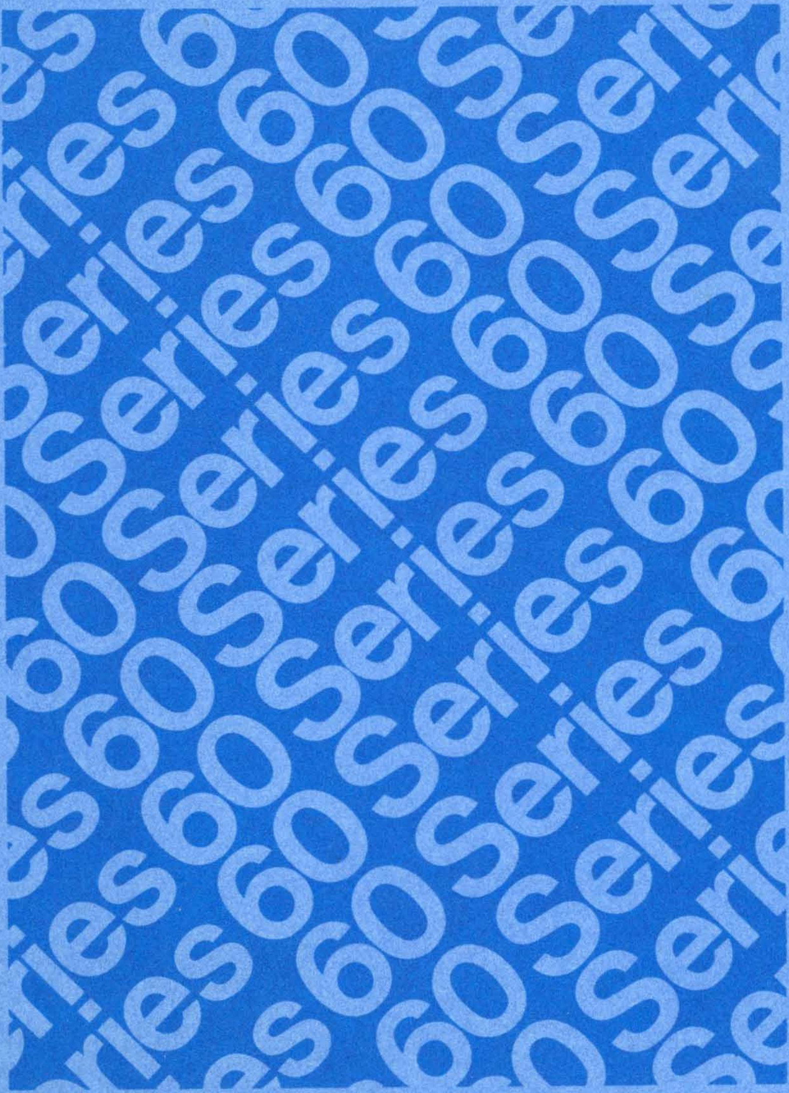
Honeywell



LEVEL 6

HARDWARE

MLCP PROGRAMMER'S REFERENCE MANUAL

SERIES 60 (LEVEL 6)

MULTILINE COMMUNICATIONS PROCESSOR (MLCP) PROGRAMMER'S REFERENCE MANUAL

SUBJECT

Multiline Communications Processor (MLCP) and Related Communications-Pacs; Reference Information for Programmer Developing Communications Applications

SPECIAL INSTRUCTIONS

This edition of the manual supersedes Revision 0, dated June 1976, and Addendum A, dated September 1976. Bars in the margins indicate new and changed technical information; asterisks denote deletions. Appendix A is entirely new and contains no change bars.

INCLUDES UPDATE PAGES ISSUED AS ADDENDUM A IN SEPTEMBER 1977.

ORDER NUMBER

AT97, Rev. 1

May 1977

Honeywell

PREFACE

This manual is intended for computer programmers who are already experienced in creating communications applications. The purpose of the manual is to assist these experienced communications programmers in creating applications for a Honeywell Series 60 (Level 6) hardware environment that includes a Multiline Communications Processor (MLCP) and one or more Communications-Pacs.

The presentation is limited to a detailed description of the MLCP and the Communications-Pac interfaces. The reader's knowledge of data communications equipment, communications line conventions, and data terminal equipment is expected to be based on additional sources, such as those listed in Appendix E.

This programmer's reference manual is intended for use in conjunction with appropriate documentation for the Level 6 operating systems referenced in the list at the end of this Preface.

New information since the last edition of the manual includes the following:

- o New main memory program and Channel Program instructions, listed in Tables 2-1 and 4-3 respectively
- o Discussion of "soft initialize" in Section 2
- o Detail on table lookup in Section 4
- o Programming considerations, Appendix A, which also incorporates information on error recovery previously documented in Appendix B
- o A list of Communications-Pacs attachable to the MLCP (Appendix B)
- o Updated information on the Asynchronous Communications-Pac (Appendix C) and the Synchronous Communications-Pac (Appendix D)

SERIES 60 (LEVEL 6) RELATED PUBLICATIONS

GCOS/BES

- o *Series 60 (Level 6) GCOS/BES Software Overview,* Order No. AS27
- o *Series 60 (Level 6) GCOS/BES Executive and Input/Output,* Order No. AS28
- o *Series 60 (Level 6) GCOS/BES Program Development Tools,* Order No. AS29
- o *Series 60 (Level 6) GCOS/BES Assembly Language,* Order No. AS31

[1] A comprehensive list of publications pertaining to the operating system in question can be found in the appropriate overview manual.

GCOS/BES2

- *Series 60 (Level 6) GCOS/BES2 Software Overview and System Conventions,* Order No. AU50
- *Series 60 (Level 6) GCOS/BES2 Executive and Input/Output,* Order No. AU45
- *Series 60 (Level 6) GCOS/BES2 Program Development Tools,* Order No. AU48
- *Series 60 (Level 6) GCOS/BES2 Assembly Language Reference Manual,* Order No. AU43

GCOS 6/MDT

- *Series 60 (Level 6) GCOS 6/MDT Overview and User's Guide,* Order No. AX11
- *Series 60 (Level 6) GCOS 6/MDT Monitor and I/O Service Calls,* Order No. AX10
- *Series 60 (Level 6) GCOS 6/MDT Program Preparation and Checkout,* Order No. AX08
- *Series 60 (Level 6) GCOS 6/MDT Assembly Language Reference Manual,* Order No. AX12

CONTENTS

Page

Page

ILLUSTRATIONS

TABLES

Table *Page*

SECTION 1

INTRODUCTION

The Multiline Communications Processor (MLCP) is a programmable communications processor that provides an interface between the Megabus and up to eight full-duplex communications lines. Both low-speed lines (up to 300 bits per second), medium-speed lines (from 600 to 20,000 bits per second), and high-speed lines (broadband) (up to 72,000 bits per second)[1] are supported.

MLCP HARDWARE OVERVIEW

The MLCP is a single primary circuit board; it uses a single interface slot of the Megabus. The MLCP provides the common elements shared by all communications lines. These elements include the firmware-controlled microprocessor, a 4096-byte random access memory (RAM), block-check logic, and the Megabus interface.

Line-specific logic is contained on Communications-Pacs, which plug into the MLCP. Each Communications-Pac connects either one or two lines. Different types of Communications-Pacs can coexist on an MLCP, which has an identical interface to each Communications-Pac. A total of four Communications-Pacs can be plugged into a single MLCP.

Figure 1-1 illustrates an MLCP with four 2-line Communications-Pacs mounted.

The MLCP is intended to relieve the central processor of most or all overhead responsibilities related to communications processing. User-created channel control programs (CCP's)—formed from an MLCP-specific instruction set—are loaded into the MLCP's RAM (along with other control information) where they typically perform the following functions during communications processing:

- Transfer of communications data characters to and/or from communications data blocks in main memory program,
- Message delimiting,
- Control character detection,
- Parity and/or cyclic redundancy check generation and verification, and
- Minor editing functions.

Each line connected to a Communications-Pac is a full-duplex data path, and each line direction is a channel to the MLCP. Each channel is capable of either receiving or transmitting communications data between communications data blocks in main memory and remote communications terminals. This data may be viewed as a sequential data stream, with the MLCP providing the necessary control and transformation of the main memory data into and out of the data formats necessary for communications lines and terminals. Each data character that passes through the MLCP is handled individually, as directed by the channel control program (CCP). In the process of transferring this data stream, the CCP is capable of translating, editing, deleting, or adding data elements; the CCP can also be programmed to recognize certain data elements—or sequences of data elements—as control characters, message delimiters, DMA block boundaries, or block-check characters.

[1] The MLCP's maximum total throughput rate is approximately 20,000 *characters* per second. However, the actual throughput rate achievable in a given communications application is governed by certain variables, which include the number of communications lines connected, the types of Communications-Pacs and data sets in use, and the extensiveness of processing performed by MLCP-resident software.

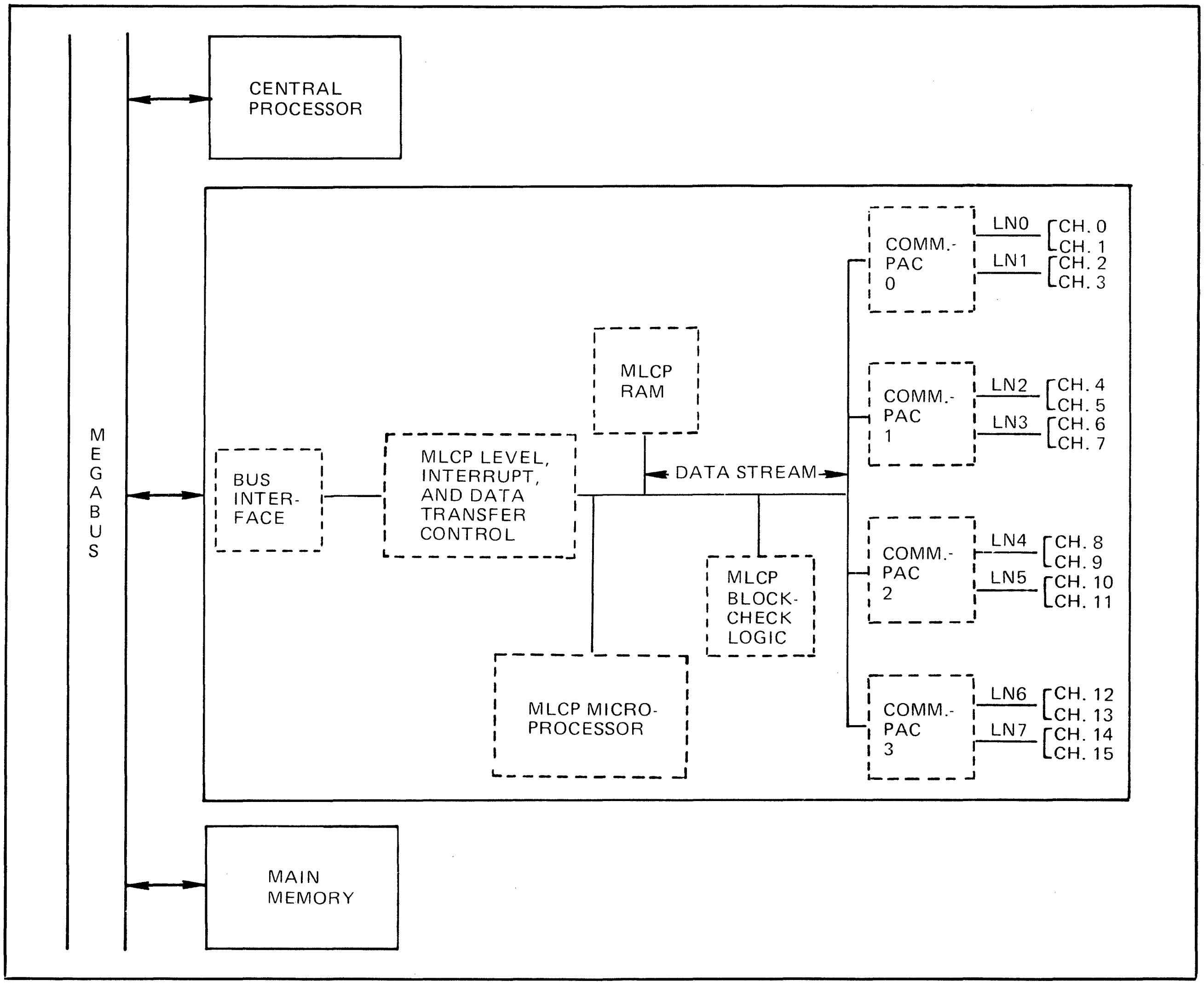


Figure 1-1. MLCP Attachment to Megabus

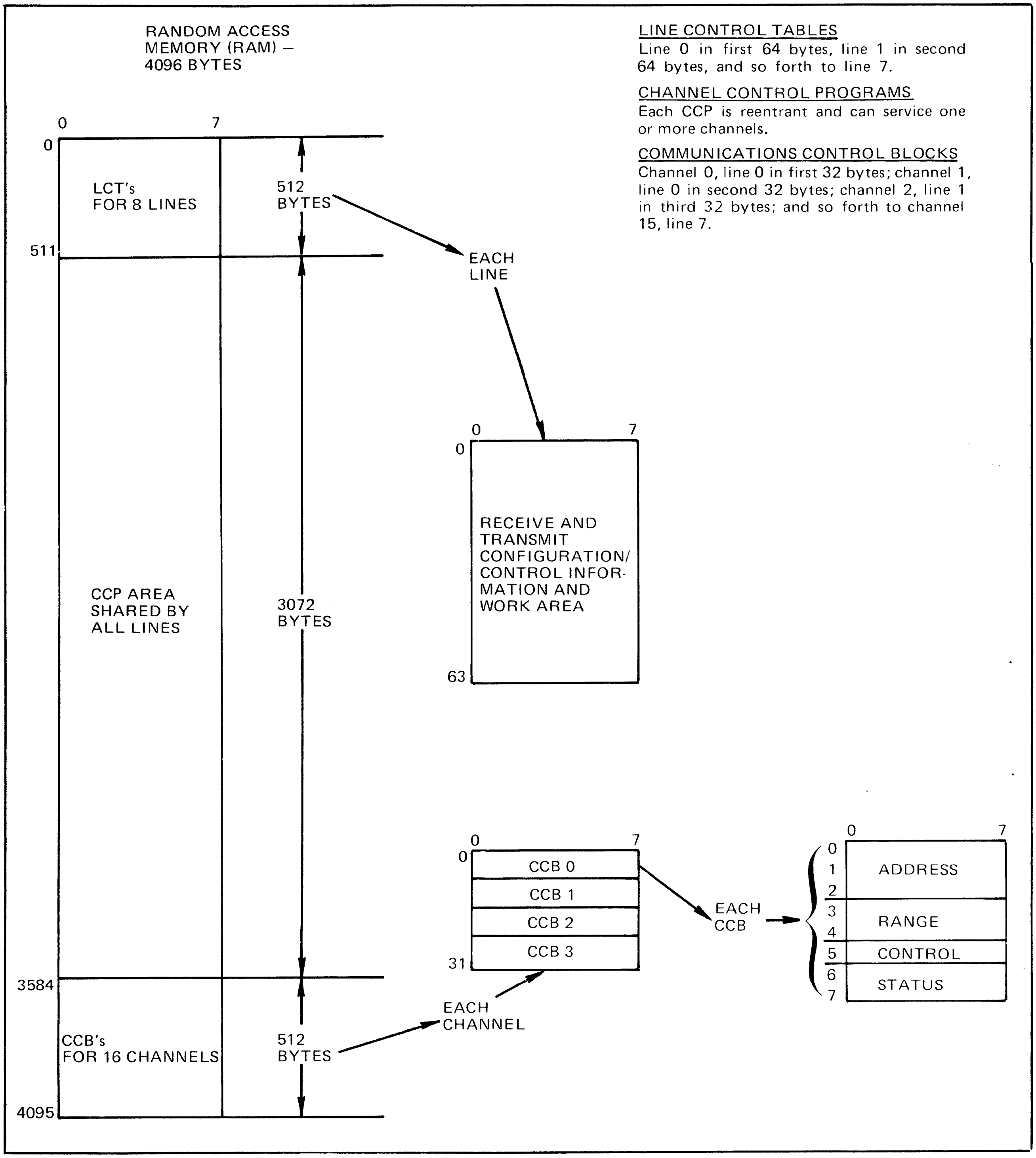
The MLCP supports the interface to main memory CDB's by means of communications control blocks (CCB's), which are set up and maintained in MLCP RAM by input/output instructions executed in the central processor.

Communications-Pacs provide line interfaces, converting output data characters into bit serial form during transmit operations and converting bit serial input into data character form during receive operations. The MLCP provides control of the line interface, supplying data characters to the Communications-Pac on transmit and data character buffers to the Communications-Pac on receive.

As data passes through the MLCP from Megabus to Communications-Pac or vice versa, the MLCP (CCP) exercises its control over the contents and format of the data stream, generating appropriate interrupts and status and control information as specified by the MLCP programmer in the channel control program or as directed by firmware as the result of indicators set by the programmer.

Control information for each channel is stored in the line control table (LCT) for the channel, which occupies part of the MLCP RAM and is accessible by the main memory program and the CCP.

The layout of the MLCP RAM, indicating the storage areas for the LCT's, CCP's, and CCB's, is shown in Figure 1-2. Note that the LCT's and CCB's are located at the low- and high-memory ends of RAM, respectively.



LINE CONTROL TABLES
Line 0 in first 64 bytes, line 1 in second 64 bytes, and so forth to line 7.

CHANNEL CONTROL PROGRAMS
Each CCP is reentrant and can service one or more channels.

COMMUNICATIONS CONTROL BLOCKS
Channel 0, line 0 in first 32 bytes; channel 1, line 0 in second 32 bytes; channel 2, line 1 in third 32 bytes; and so forth to channel 15, line 7.

Figure 1-2. MLCP Memory Map

Figures 1-3 and 1-4 illustrate how the MLCP could provide the message delimiting function by (1) looking for synchronization characters, SOH, STX, ETX, and block-check characters on *receive* and (2) sending similar control characters on *transmit*. The MLCP itself is insensitive to variations in character set and control characters, although support for different types of communications interfaces may require different Communications-Pacs to be used because of varying transmission characteristics.

Figure 1-3 illustrates a *receive* operation in which data characters are being received into two noncontiguous communications data blocks (CDB's) in main memory. The header block and the text block are receiving the data characters from the stream whose line format is shown at the bottom of the figure. Since two distinct CDB's are involved, two different communications control blocks (CCB's) are required. The MLCP (CCP) performs the required transition from the first CDB to the second CDB, receives the incoming data character from the Communications-Pac, updates the block-check accumulation for later analysis, does any necessary editing and/or status reporting, and transfers the data character to the CDB in main memory.

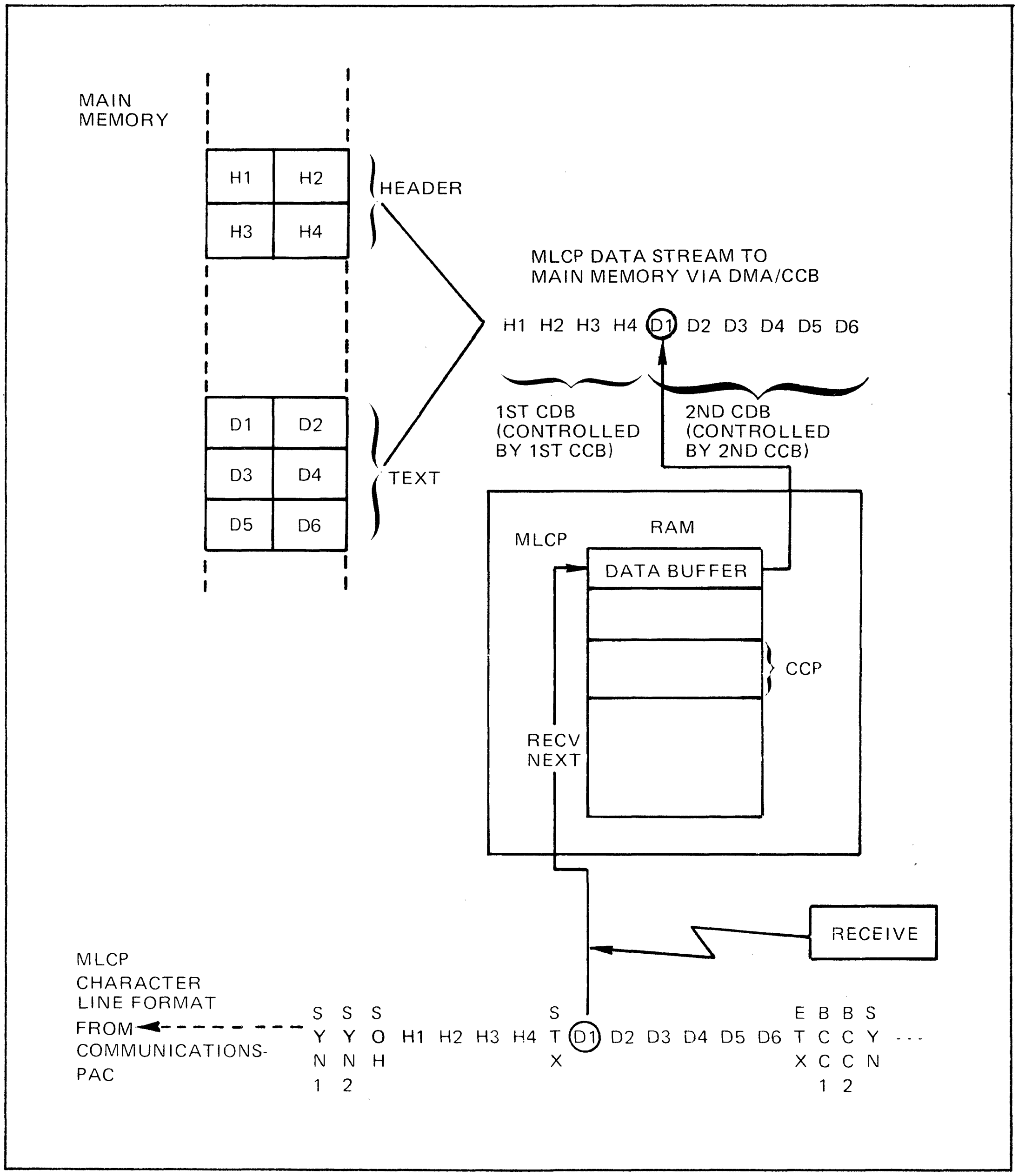

MAIN MEMORY
H1
H2
H3
H4
HEADER
D1
D2
D3
D4
D5
D6
TEXT
MLCP DATA STREAM TO MAIN MEMORY VIA DMA/CCB
H1 H2 H3 H4 D1 D2 D3 D4 D5 D6
1ST CDB (CONTROLLED BY 1ST CCB)
2ND CDB (CONTROLLED BY 2ND CCB)
MLCP
RAM
DATA BUFFER
CCP
RECV NEXT
RECEIVE
MLCP CHARACTER LINE FORMAT FROM COMMUNICATIONS-PAC
SYN1 SYN2 SOH H1 H2 H3 H4 STX D1 D2 D3 D4 D5 D6 ETX BCC1 BCC2 SYN ...

Figure 1-3. MLCP Functions–Receive

Figure 1-4 illustrates a *transmit* operation in which data characters are being obtained from two noncontiguous CDB's in main memory. The header block and the text block are providing the data characters for the stream whose line format is shown at the bottom of the figure. Since two distinct CDB's are involved, two different CCB's are again required. The MLCP (CCP) obtains the data character to be transmitted, updates the block-check accumulation for later transmission, does any necessary editing and/or status reporting, and transfers the data character to the Communications-Pac for transmission.

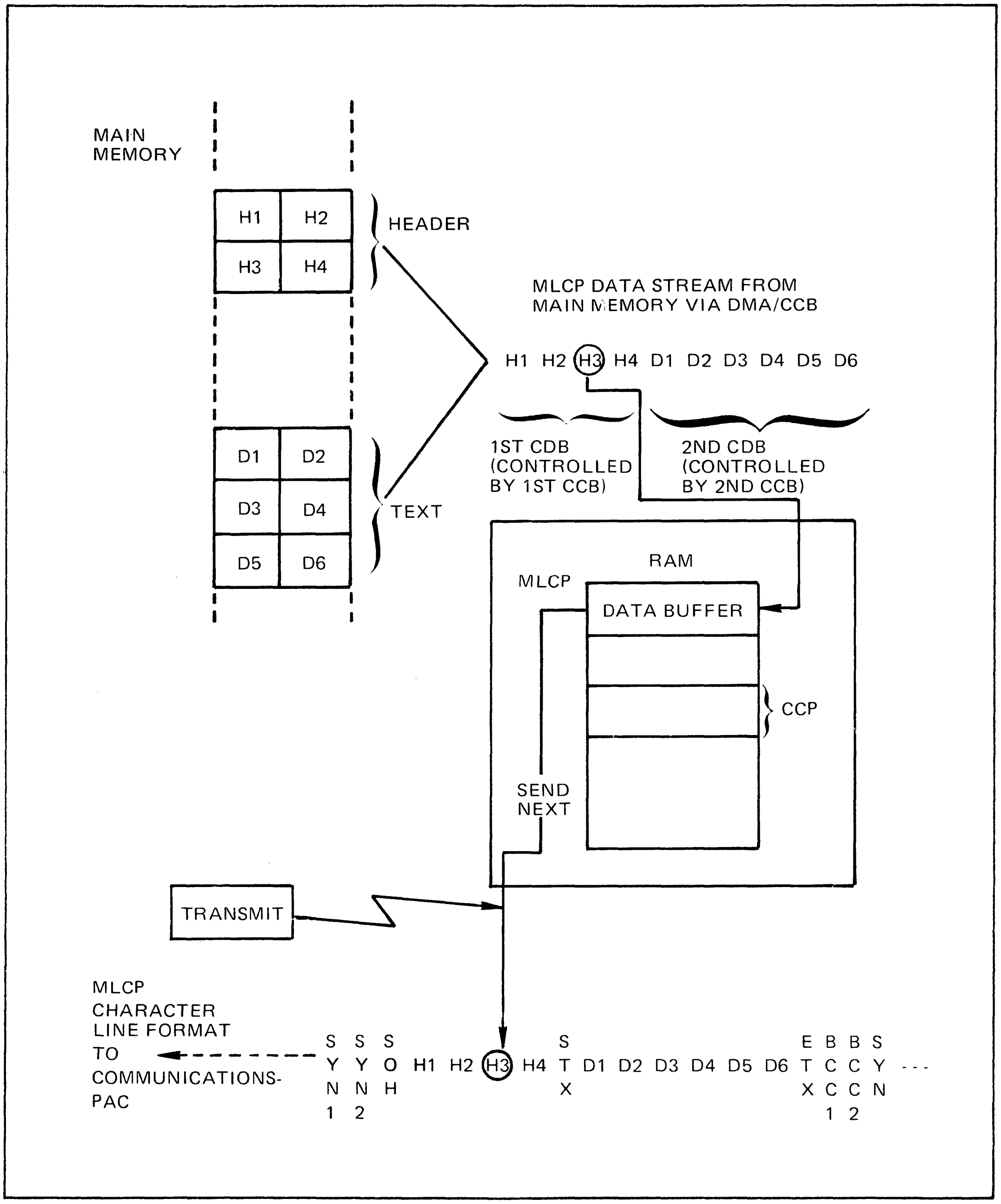

MAIN MEMORY
H1
H2
H3
H4
HEADER
D1
D2
D3
D4
D5
D6
TEXT
MLCP DATA STREAM FROM MAIN MEMORY VIA DMA/CCB
H1 H2 H3 H4 D1 D2 D3 D4 D5 D6
1ST CDB (CONTROLLED BY 1ST CCB)
2ND CDB (CONTROLLED BY 2ND CCB)
RAM
MLCP
DATA BUFFER
CCP
SEND NEXT
TRANSMIT
MLCP CHARACTER LINE FORMAT TO COMMUNICATIONS-PAC
SYN1 SYN2 SOH H1 H2 H3 H4 STX D1 D2 D3 D4 D5 D6 ETX BCC1 BCC2 SYN ---

Figure 1-4. MLCP Functions–Transmit

In addition to the MLCP's ability to accommodate data transfers and related message delimiting and editing for data communications, it also controls the data-communications-equipment/data-terminal-equipment interface provided in the Communications-Pac for each communications line. Data for this latter function is stored in dedicated bytes of the line control table; this data can be modified and controlled by CCP instructions, as required by a specific application.

Table 1-1 provides a summary of the principal characteristics of the MLCP.

TABLE 1-1. MLCP HARDWARE SUMMARY

Component	Comments
MLCP Memory (Random Access Memory–RAM)	Total Size: 4096 bytes Dedicated to LCT's 512 bytes Usable for CCP's 3072 bytes Dedicated to CCB's 512 bytes
MLCP Registers/Indicators	A 12-bit P-register for program (instruction) sequencing. An 8-bit R-register for data manipulation. An E-indicator for equals compare. A V-indicator for testing of valid CCB's. An LC-indicator for last character detection. An LB-indicator for last block detection.
MLCP (CCP) Instruction Set	43 Instructions: 15 Branch instructions 14 Double operand instructions (three formats). 2 Input/output instructions. 2 Send/receive instructions. 10 Generic instructions.
Line Control Tables (LCT's)	Dedicated space for eight LCT's (one per line) exists in an MLCP. Each LCT is 64 bytes long. First 32 bytes of each LCT are for receive channel; second 32 bytes of each LCT are for transmit channel. LCT's contain fixed programming and firmware data along with nine programming work bytes per channel.
Channel Control Programs (CCP's)	CCP's are reentrant and can be used by one or more channels; 3072 bytes are available for CCP's.
Communications Control Blocks (CCB's)	Dedicated space for 64 CCB's exists in an MLCP. Each channel is allotted four CCB's. Each CCB is eight bytes long. CCB's are programmer-controlled by central processor instructions.
Communications-Pacs	From one to four Communications-Pac can be attached to an MLCP. Each Communications-Pac handles one or two lines.
Lines	From one to eight lines can be attached (through Communications-Pacs) to an MLCP. Each line has two channels (one receive and one transmit).
Channels	From two to 16 channels can be handled by an MLCP. Channels are numbered from 0 to 15. Even-numbered channels are used for receive; odd-numbered channels are used for transmit.
Line Speed	Individual line speeds may be from 45 bps to 72,000 bps, depending upon communications-pac and modem type.

MLCP PROCESSING PRIORITIES

The MLCP processing priorities (from high to low) are as follows:

1. Servicing main memory program input/output instructions to the MLCP,
2. Servicing Communications-Pac channel request interrupts, and
3. Background firmware scanning.

SERVICING MAIN MEMORY PROGRAM INPUT/OUTPUT INSTRUCTIONS TO THE MLCP

Main memory program input/output instructions to the MLCP are serviced as the MLCP's highest priority activity.

SERVICING COMMUNICATIONS-PAC CHANNEL REQUEST INTERRUPTS

The MLCP responds to Communications-Pac channel request interrupts when no MLCP-related input/output instructions from the main memory program are outstanding and after the currently executing CCP has completed processing the latest character. The MLCP services its channels on a priority basis. When more than one channel has a Communications-Pac channel request interrupt pending for MLCP servicing or when simultaneous channel request interrupts occur, they are serviced by the MLCP according to their priority levels. The priority level of a channel is simply a function of its channel number, with the lowest numbered channel having the highest priority as shown in Table 1-2.

TABLE 1-2. PRIORITIES FOR SERVICING COMMUNICATIONS-PAC CHANNEL REQUEST INTERRUPTS

Priority Level	Channel Number	Receive or Transmit Channel	Line Number
Highest	0	Receive	0
	1	Transmit	0
	2	Receive	1
	3	Transmit	1
	4	Receive	2
	5	Transmit	2
	6	Receive	3
	7	Transmit	3
	8	Receive	4
	9	Transmit	4
	10	Receive	5
	11	Transmit	5
	12	Receive	6
	13	Transmit	6
	14	Receive	7
Lowest	15	Transmit	7

BACKGROUND FIRMWARE SCANNING

Background firmware scanning of activities within the MLCP will occur after the MLCP has serviced all main memory program input/output instructions to the MLCP and all Communications-Pac channel request interrupts. This scan can be used to interrupt the main memory program or to start a CCP whenever a data set or Communications-Pac status change has occurred. Firmware scanning and related actions are enabled for a channel by settings of certain bit positions in that channel's line control table. A firmware scan will typically occur at least every one-half second.

MLCP PROGRAMMING OVERVIEW

In addition to preparing a main memory program, which operates in the central processor, the MLCP programmer is responsible for creating the following software and writing it to the MLCP:

- o Communications control blocks
- o Channel control program(s)
- o Line control tables

Before communications processing begins, the channel control program(s) and line control tables must be prepared and then transferred to the MLCP by use of the Honeywell-supplied MLCP Loader or by means of a "block mode write"; both loading methods are described in Section 7. Communications control blocks are dynamically supplied by the main memory program during communications processing.

Main Memory Program

One or more programs must reside in main memory to interact with the MLCP. A main memory program interfaces with one or more communications channels. The general responsibilities of a main memory program are as follows:

- o Optionally, it writes the LCT area and the CCP area of MLCP RAM as part of the communications application loading process.
- o It stores MLCP channels' interrupt levels in LCT's (unless this action has been performed during loading of MLCP RAM) and then handles all interrupts that come back at these levels.
- o It performs MLCP and channel control functions such as initialization and starting/stopping channel operations when errors are detected.
- o It sets up the required CCB's and maintains them throughout execution of the application.
- o It maintains CDB's in main memory. This activity includes (1) handling the CDB's as they are completed, (2) supplying pointers to CDB's (for use by the CCB's) when required, and (3) monitoring the status and error conditions for each CDB and reacting appropriately.
- o It monitors the status of the data sets and Communications-Pacs and takes appropriate action when certain changes take place.

Section 2 describes the instructions that provide the communications-related functionality necessary in the main memory program.

Communications Control Blocks

For *each channel*, space exists in upper MLCP RAM for a "list" of four consecutive 8-byte communications control blocks (CCB's). Each CCB is used to store main memory address information that indicates the area *to* which communications data is to be delivered (receive operation) or *from* which communications data is to be obtained (transmit operation). The main memory area is called a communications data block (CDB). MLCP firmware uses the programmer-supplied information in the CCB when transferring data to or from the main memory CDB. The CCB also contains a control field and a firmware storage area for status and error indicators relating to the data transfer to or from the CDB.

Setup and maintenance of the four CCB's dedicated to each channel must be performed from the main memory program associated with that channel; a detailed description appears in Section 3.

Channel Control Program

A channel control program (CCP) directs the movement of each data character through the MLCP. The CCP can cause a data character to be processed in a simple, straightforward manner requiring a minimum of time—or, at the discretion of the programmer, the CCP can conduct more extensive checking and editing functions that require more MLCP processing time. If the CCP is programmed to perform data character processing beyond basic message delimiting and block-checking functions, this processing will be performed at the expense of throughput speed.

Because of the nature of the instruction set and the design of the MLCP, each CCP is reentrant and therefore usable for more than one channel. A major factor permitting reentrant CCP's is that the control information necessary to operate a channel is stored in the LCT and the CCB associated with only that *one* channel.

The following functions can be performed by a CCP:

o Data character editing,
o Parity and/or cyclic redundancy check generation and verification,
o Data set and Communications-Pac control, and
o Error detection and handling.

All CCP's concurrently resident in the MLCP share the 3072 bytes of RAM allocated for CCP usage.

The CCP is prepared by use of the appropriate program development facilities of the operating system.

Line Control Tables

For *each line*, space exists in lower MLCP RAM for one 64-byte line control table. Each LCT is logically divided in half, with the first 32 bytes dedicated to the *receive* channel of the line and the second 32 bytes dedicated to the *transmit* channel. Each channel-related half of an LCT comprises the following elements:

o Program-supplied input data,
o Programming work bytes,
o Programming information supplied by firmware, and
o Bytes reserved for firmware use.

The program-supplied input data bytes provide information required for character configuration, CCP control, interrupt control, firmware control relative to status and error conditions, and data set and Communications-Pac control.

The programming work bytes can be used in any way needed by the main memory program or CCP's.

Programming information supplied by firmware consists of status and error information related to the data set or Communications-Pac as well as to data transfer operations.

A number of bytes are reserved for firmware use. During MLCP setup, these bytes may be overwritten with zeros; during subsequent communications processing, these bytes must not be modified by software.

SETTING UP THE MLCP; RECEIVING AND TRANSMITTING DATA

Listed below are three sequences of events. The first is a general description of *one* possible way to set up the MLCP before communications processing begins; the second indicates the general order of events as data is received over a channel; the third indicates the general order of events as data is transmitted from a main memory program.

Perform the following actions from the main memory program (See Figure 1-5):

1. Use the Honeywell-supplied MLCP Loader to initialize the entire MLCP and to write a user-created block to the LCT area and the CCP area of MLCP RAM.

(This action writes the user-desired values into each LCT to be used and writes one or more user-created CCP's into the CCP area.)[2]

2. Write the desired CCB's for initial communications data transfers.
3. For each CCP, issue an appropriate IO instruction to start the CCP, allowing it to load the necessary registers of the related Communications-Pac.
   a. The CCP loads line register 6 of the Communications-Pac.
   b. The CCP loads line register 4 of the Communications-Pac.
   c. The CCP loads line register 2 of the Communications-Pac.

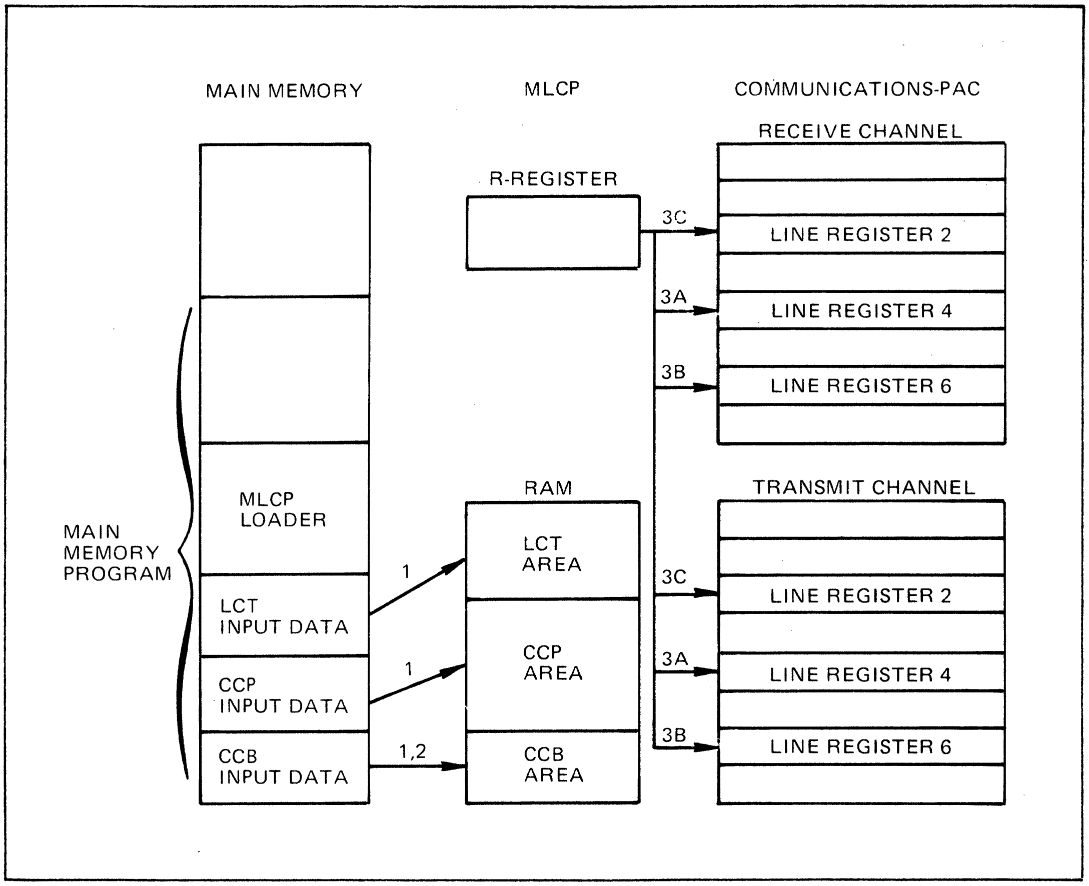


Figure 1-5. Setting Up the MLCP

Refer to Figure 1-6 for receiving data.

1R. A data character received in a receive channel's line register 1 causes the Communications-Pac to generate a channel request interrupt to the MLCP.
2R. The CCP is started. It loads the received data character into the MLCP's R-register.
3R. The CCP edits/modifies the data character in the R-register as required.
4R. The CCP transfers the data character from the R-register to a CDB in main memory.
5R. The CCP assumes a wait mode.
6R. The MLCP starts processing the next function pending for it.

[2] As an alternative, you can use a "block mode write" (a sequence of input/output instructions in the main memory program) to write a user-created block to the LCT area and the CCP area of MLCP RAM.

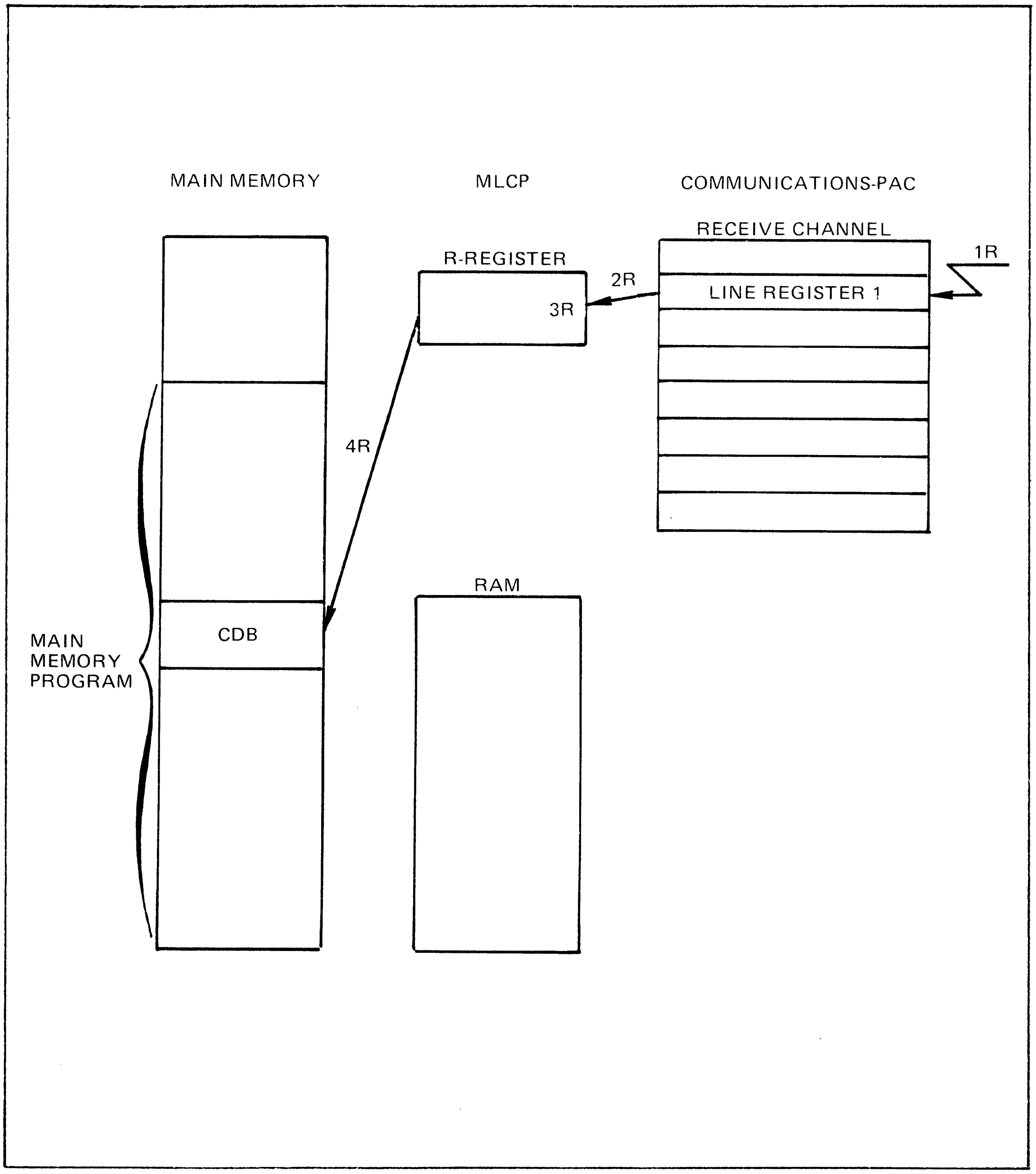


Figure 1-6. Receiving Data

See Figure 1-7 for transmitting data.

1T. The Communications-Pac generates a channel request interrupt, signifying that it can accept a data character for transmission.
2T. The CCP loads a data character from the main memory CDB into the MLCP's R-register.
3T. The CCP edits/modifies the data character in the R-register as required.
4T. The CCP sends the data character to the transmit channel's line register 1 of the Communications-Pac. From there, the data character is automatically transmitted.
5T. The CCP assumes a wait mode.
6T. The MLCP starts processing the next function pending for it.

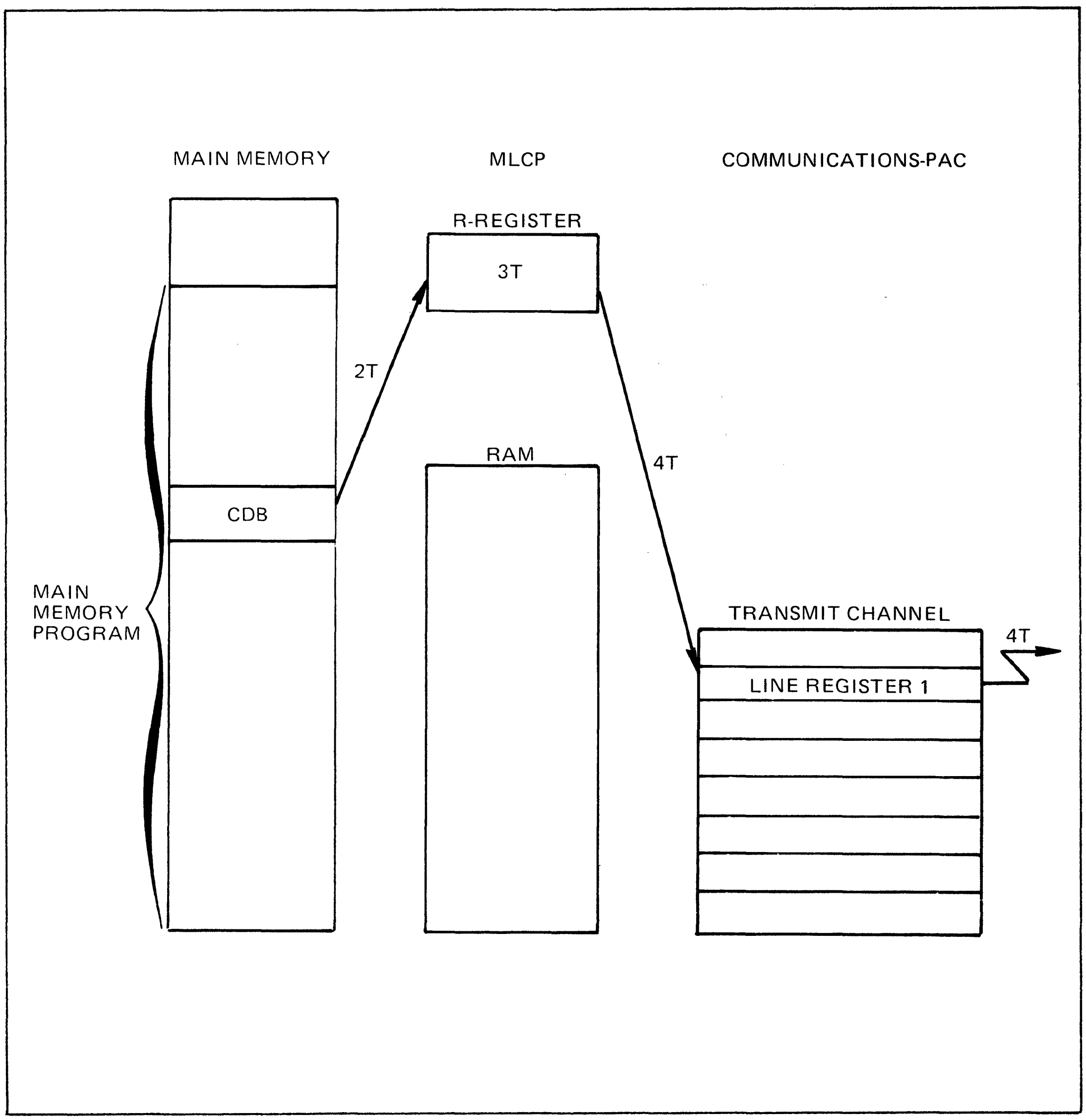


Figure 1-7. Transmitting Data

The following sequence of events provides a more detailed description of *one* way to perform MLCP setup and subsequent data communications operations.

1. *Initializing the MLCP; Writing the LCT Area and the CCP Area*[3]

   The Honeywell-supplied MLCP Loader can be used to initialize the entire MLCP and to write a user-created block to the LCT area and the CCP area of MLCP RAM. The MLCP Loader is supplied in object module form and it must be linked with an object module that contains, as a minimum, the user-created block to be written to MLCP RAM. (The latter object module has typically been processed by the GCOS/BES Macro Preprocessor and then assembled by the GCOS/BES Assembler.) Details regarding the MLCP Loader appear in Section 7.[4]

[3] Prior to this step, you may wish to have a main memory program issue a separate IO (Input Device Identification Number) instruction for each channel to be used; for each channel specified, this action returns an identification number that indicates what type of Communications-Pac is in use.

[4] Under certain circumstances, it may be preferable to use a "block mode write," also described in Section 7, as an alternative to the MLCP Loader. A "block mode write" can be used, for instance, when you wish to initialize and set up only a subset of the MLCP's channels, while previously initiated communications processing continues concurrently over other channels.

Special care must be taken when creating the image that is to be transferred to the LCT area of MLCP RAM. Certain LCT bytes must be set up with appropriate values to control hardware/firmware operations; other LCT bytes can be set up with application-specific values, as desired; still other LCT bytes must contain zero when communications processing begins. Section 5 fully describes the program-visible LCT bytes. Section 5 also provides a detailed layout of each LCT byte, including information about the initial settings and subsequent modifiability of all bit positions.

2. *Writing the CCB's*

   For each channel, one or more CCB's can now be written to define the starting location and length of one or more CDB's in the main memory program. Each CCB must be set up by the following instructions from the main memory program:

   o IOLD (Output CCB Address and Range)
   o IO (Output CCB Control–format 1)

   Thereafter, throughout execution of the application, the main memory program is responsible for supplying CCB's, as needed, for the CCB list of each channel being used.

3. *Loading Communications-Pac Line Registers*

   The line registers of each Communications-Pac are loaded next. This action is performed by the appropriate CCP, and the main memory program must issue an IO (Output Channel Control–start input/output) instruction to start the CCP. The initial part of the CCP should load line registers 4, 6, and 2; line register 2 must be loaded last. This loading is accomplished by a separate OUT (Output) instruction for each line register to be loaded.

   Line register 6 (character configuration) is loaded from LCT byte 2 or from LCT byte 34. Line register 6 is shared by both channels of one line.

   Line register 4 (line speed or synchronization/transmit-fill character) can be loaded from a byte in the programming work area of a line control table. For Asynchronous Line Communications-Pacs, line register 4 must be loaded with a value indicating the speed at which each channel of the line will operate. For Synchronous Line Communications-Pacs, a synchronization character must be loaded into line register 4 for the receive channel and a transmit fill character must be loaded into line register 4 for the transmit channel. Line register 4 may have a different function, depending upon the type of Communications-Pac used.

   Line register 2 (data set and Communications-Pac control) is loaded last from LCT byte 20. Line register 2 is shared by both channels of one line. *

   Once line register 2 is loaded, the Communications-Pac will be enabled to generate channel request interrupts, according to the configuration of the Communications-Pac and the data communications equipment. The CCP should execute a WAIT (Wait) instruction at this point. In the CCP, the WAIT instruction is followed by a data character processing loop, which usually terminates in a branch back to the WAIT. The CCP startup coding preceding the WAIT is normally executed only the first time the CCP is started.

4. *Receiving Data*

   A channel request interrupt from the Communications-Pac to the MLCP indicates that an input (receive) data character is available in a receive channel's line register 1 of the Communications-Pac. The MLCP performs a context restore for the channel (preparing the appropriate CCP for execution). The CCP is turned on at the instruction just after the previous WAIT (Wait) instruction executed by this CCP; the CCP uses an RECV (Receive) instruction to load the MLCP's R-register with the data character from line register 1. The CCP edits and manipulates the data character in the R-register as required by the application. The CCP then transfers the data character from the R-register to the CDB by means of an ST (Store) instruction. The CCP then branches back to the WAIT (Wait) instruction.

This is the basic CCP receive processing loop for each data character of a communications message. The loop can also contain branch and/or TLU (Table Look-Up) instructions for other checks relative to the data character. A CCP subroutine could also be used.

5. *Transmitting Data*

   The Communications-Pac issues a channel request interrupt to the MLCP, indicating that it is ready to accept a data character for transmission. The CCP is turned on after the context restore; the CCP then either loads a data character into the MLCP's R-register or uses the character reloaded into the R-register during the context restore. Next, the CCP can edit and manipulate the data character as required before transferring it to the transmit channel's line register 1 of the Communications-Pac by means of a SEND (Send) instruction. The data character is then automatically transmitted from the Communications-Pac.

   If desired, after issuing the SEND (Send) instruction, the CCP can immediately issue a WAIT (Wait) instruction; in this case, when the CCP is next activated, it will have to load the R-register with the data character to be transmitted next (editing and manipulating it as necessary) before issuing a SEND and a WAIT instruction. Alternatively, after issuing the SEND instruction, the CCP can load the R-register with the data character to be transmitted next (editing and manipulating it as necessary) *before* issuing the WAIT instruction; in this case, the data character will be reloaded into the R-register during the context restore that accompanies reactivation of the CCP, and the SEND can be done immediately.

6. *End of CDB Processing*

   The relationship between physical CDB's and logical communications messages is completely under programmer control.

   In receive mode, when the CCP executes an ST (Store) instruction for the last character in a CDB, the range in the CCB decreases to zero and the last character (LC-) indicator is set to 1. To check for the end of receive data *before* a CDB becomes full, the CCP can search for a specific control character in the input data stream; the CCP can use a TLU (Table Look-Up) or a C (Compare) instruction to check for this condition. Whenever processing ends relative to a CDB, the CCP can obtain the next CDB by issuing a GNB (Get Next Block) instruction.

   In transmit mode, termination of CDB processing normally occurs when CCB range decreases to zero and the last character (LC-) indicator is set. In some cases, earlier termination may be necessary because of some other condition discovered by the CCP. In any case, to continue transmission with another CDB and CCB, the CCP must issue a GNB (Get Next Block) instruction.

7. *End of Logical Message Processing*

   As mentioned above, the relationship between physical CDB's and logical communications messages is completely under programmer control.

   If a logical communications message uses only *one* CDB, processing for that CDB is basically as described in step 6; however, instead of the CCP routinely proceeding from one CDB to another (as in step 6), CDB processing should continue as required by the application.

   If logical communications messages comprise *more than one* CDB, individual messages may use either a variable or fixed number of CDB's. In any case, the last CDB in a message can be identified to the CCP if the main memory program has set the last block (LB-) indicator in the CCB control byte. The last block indicator can be set by an IO (Output CCB Control) instruction from the main memory program. In receive mode, the last CDB can be indicated by a control character in the incoming data stream. Alternatively, the CCB valid indicator can be tested by the CCP (refer to Appendix A).

8. *End of Channel or Line Usage*

   When processing relative to a channel or line is finished, you can indicate this fact to the Communications-Pac by resetting to 0 the "receive ON" and/or the "transmit ON" bit in line register 2 of the Communications-Pac.

   Subsequently, communications processing could be resumed over the channel or line by (as a minimum) reloading line register 2 with appropriate values. A more extensive restart would involve initializing the channel by means of an IO (Output Channel Control—channel initialize) instruction, performing one or more "block mode write" operations to set up the LCT and CCP, and then loading the appropriate Communications-Pac line registers from the new CCP.

SECTION 2

MAIN MEMORY PROGRAM

The main memory program has the following responsibilities:

- Control of the MLCP by means of input/output instructions issued to it,
- Control of communications data blocks (CDB's),
- Control of communications control blocks (CCB's),
- Control of channel control programs (CCP's), and
- Detection of errors and status changes related to data communications equipment and data terminal equipment.

SUMMARY OF MAIN MEMORY PROGRAM INPUT/ OUTPUT INSTRUCTIONS RELATED TO MLCP

Table 2-1 provides a summary description of the input/output instructions available to the main memory program for controlling the MLCP. These input/output instructions appear in alphabetic order in Table 2-1. They are described in greater detail later in this section.

TABLE 2-1. SUMMARY OF MAIN MEMORY PROGRAM INPUT/OUTPUT INSTRUCTIONS RELATED TO MLCP

Instruction	Function Code	Description
IO (Input CCB Range)[a]	OC	Transfers, to the main memory program, the two range bytes of the "status" CCB.
IO (Input CCB Status)[a]	18	Transfers, to the main memory program, the two status bytes of the "status" CCB.
IO (Input Data Set Status)	1C	Transfers, to the main memory program, the contents of Communications-Pac line register 5 for a specified MLCP channel.
IO (Input Device Identification Number)	26	Transfers, to the main memory program, the identification number that indicates the type of Communications-Pac associated with a specified MLCP channel. If response is 2178_{16}, issue function code 08 (see below).
IO (Input Extended Device Identification Number)	08	Transfers, to the main memory program, the identification number that indicates the type of Communications-Pac associated with a specified MLCP channel. Issued when response to Input Device Identification Number (code 26) is 2178_{16}.

TABLE 2-1 (CONT). SUMMARY OF MAIN MEMORY PROGRAM INPUT/OUTPUT INSTRUCTIONS RELATED TO MLCP

Instruction	Function Code	Description
IO (Input LCT Byte)	1E	Transfers, to the main memory program, the LCT byte addressed by the contents of LCT byte 55 for a specified MLCP channel, which must be preloaded by the Output LCT Byte instruction.
IO (Input Next CCB Status)[a]	1A	Moves the "status" CCB pointer to the following CCB in the CCB list; transfers the two status bytes of *that* CCB (which is now the "status" CCB) to the main memory program.
IOLD (Output CCB Address and Range)[a]	09	Transfers, to the "load" CCB, the starting address and range of a CDB in main memory.
IO (Output CCB Control)[a]	0F	Format 1: Transfers control information from the main memory program to the control byte of the "current" CCB. Moves the "load" CCB pointer to the following CCB in the CCB list. Format 2: Transfers, to the "current" CCB, the starting MLCP RAM address for "block mode" input/output. Moves the "load" CCB pointer to the following CCB in the CCB list.
IO (Output Channel Control)	05	Causes the MLCP to perform *one* of the following actions: o Channel initialization o Start or stop input/output o Start "block mode read" or "block mode write" o CCB list reset
IO (Output Interrupt Control)	03	Transfers, to the LCT for a specified MLCP channel, the return channel number and interrupt level to be used during interrupts from that channel.
IO (Output LCT Byte)	0B	Transfers one byte from the main memory program to a specified byte in a specified LCT.
IO (Output MLCP Control)	01	Causes initialization of the MLCP.

[a] All terms related to CCB's are defined at the beginning of Section 3.

*

CONTROL OF COMMUNICATIONS DATA BLOCKS

The main memory program has total responsibility for the control of CDB's. This responsibility includes (1) supplying *new* CDB's, as needed, for use in communications data transfers and (2) servicing CDB's *after* they have been used for communications data transfers. Since communications data transfers to and from main memory are controlled by a fixed number of reusable communications control blocks (CCB's), the main memory program must know the completion status of CCB's in order to coordinate its control of CDB's. If desired, the main memory program can arrange for the MLCP to generate an interrupt as soon as a CCB is marked completed; this technique is described in Section 6. Alternatively, the main memory program can perform its *own* checking of CCB completion status; the format of the two CCB status bytes is described in Section 3.

CONTROL OF COMMUNICATIONS CONTROL BLOCKS

The main memory program completely controls additions to and deletions from the list of CCB's available to each channel of the MLCP. This control is achieved by use of the MLCP-related input/output instructions described in this section.

CONTROL OF CHANNEL CONTROL PROGRAMS

The main memory program is responsible for loading and starting CCP's. The main memory program can load CCP's by use of the Honeywell-provided MLCP Loader or it can use one or more "block mode writes" for this purpose; both techniques are described in Section 7. The main memory program starts initial execution of each CCP by issuing an IO (Output Channel Control) instruction.

DETECTION OF ERRORS AND STATUS CHANGES RELATED TO DATA COMMUNICATIONS EQUIPMENT AND DATA TERMINAL EQUIPMENT

The main memory program can detect and respond to errors and status changes related to data communications equipment and data terminal equipment. Relevant information is available in the two status bytes of a CCB and in line register 5 of a Communications-Pac. Most of this information is also available to CCP's, allowing *them* to detect and respond to errors and status changes, if desired.

Note that if a CCP is designed to perform *extensive* responses to errors and status changes, its execution time will be increased accordingly. (This increase in execution time may be perfectly acceptable in some applications–e.g., those using low-speed communications lines.)

One approach to detecting and responding to errors and status changes related to data communications equipment and data terminal equipment would be to have the main memory program and the CCP *share* this responsibility. The CCP could be designed to react to errors and status changes as individual characters in a data stream are processed; the main memory program could be designed to react to these errors and status changes as they affect an entire CDB.

At any rate, the designer of a communications application must decide how much handling of errors and status changes will be performed by the main memory program and how much (if any) will be performed by CCP's. Section 6 and Appendix A provide more information on detecting errors and status changes related to data communications equipment and data terminal equipment.

DETAILED DESCRIPTION OF MAIN MEMORY PROGRAM INPUT/OUTPUT INSTRUCTIONS RELATED TO MLCP

The following subsections describe the input/output instructions usable in a main memory program interfacing with an MLCP. See the appropriate Assembly Language manual for details about coding these and other instructions used in the main memory program.

All but one of these input/output instructions are IO instructions. (The other one is an IOLD instruction.) The format of these IO instructions is shown below.

IO ML,CF

ML

An address expression identifying a memory location or register *to* which or *from* which information is to be transferred.

CF

An address expression identifying a memory location or register that contains a channel number and a function code. The format of this information is shown below.

0 9	10 15
Channel Number	FC

The *channel number* comprises two parts: bits 0 through 5 contain the six bits of the MLCP's fixed (switch-selectable) channel number for the Megabus; bits 6 through 9 identify one of the 16 communications channels of the MLCP. (The designated communications channel must be serviced by a Communications-Pac.)

FC indicates the function code, which specifies the exact input/output operation to be performed. An *odd* function code signifies an *output* instruction; the contents of ML will be transferred to the MLCP. An *even* function code signifies an *input* instruction; data will be transferred from the MLCP to ML.

The format of the IOLD instruction is shown under "IOLD (Output CCB Address and Range) Instruction," later in this section.

Input/output instructions to the MLCP are generally executed immediately without causing a NAK. However, you may wish to code a BIOF (Branch if Input/Output Indicator False) instruction after the first input/output instruction following an IO (Output MLCP Control) instruction; see Appendix B. (A NAK will occur if the MLCP cannot honor the input/output instruction because MLCP initialization is still in progress.)

IO (Input CCB Range) Instruction

This instruction (function code: 0C) transfers, to ML, the two range bytes in the CCB at the top of the CCB list for the MLCP channel specified in CF. As shown below, the range byte from CCB byte 4 is transferred to the *left* byte of ML and the range byte from CCB byte 3 is transferred to the *right* byte of ML.

	0 7	8 15
ML	CCB Byte 4	CCB Byte 3
	MSB	LSB

IO (Input CCB Status) Instruction

This instruction (function code: 18) transfers, to ML, the two status bytes in the CCB at the top of the CCB list for the MLCP channel specified in CF. As shown below, the status byte from CCB byte 7 is transferred to the *left* byte of ML and the status byte from CCB byte 6 is transferred to the *right* byte of ML.

	0 7	8 15
ML	CCB Byte 7	CCB Byte 6
	Status Byte 1	Status Byte 2

The CCB status bytes will contain zero if the CCB has not yet been marked as completed.

CAUTION:

Input CCP Status does not advance the CCB status pointer. When a status incomplete is detected, after an Input Next CCB Status, then an Input CCB Status would be issued to detect the status complete bit. This technique is used for non-CPU interrupt mode where the main memory program is waiting for CCP completion.

IO (Input Data Set Status) Instruction

This instruction (function code: 1C) causes a *direct* transfer, to ML, of the contents of Communications-Pac line register 5 for the MLCP channel specified in CF. The format of the word transferred to ML is shown below.

	0 7	8 15
ML	Line Register 5	0 0 0 0 0 0 0 0

The contents of line register 5 vary according to the type of Communications-Pac; see Appendixes C and D.

IO (Input Device Identification Number) Instruction

This instruction (function code: 26) transfers, to ML, an identification number that indicates the type of Communications-Pac that services the MLCP channel specified in CF. The device identification numbers for the Communications-Pacs are given in the appendixes of this manual.

IO (Input LCT Byte) Instruction

This instruction (function code: IE) transfers, to ML, the LCT byte addressed by the contents of LCT byte 55. The contents of LCT byte 55 are loaded by the Output LCT Byte instruction (function code: 0B) or by a Block Mode Write. Refer to Section 5 for an additional definition of LCT byte 55. The contents of the LCT byte addressed by the contents of LCT byte 55 are delivered to main memory as follows.

0 7	8 15
Byte	RFU

IO (Input Next CCB Status) Instruction

This instruction (function code: 1A) causes the "status" CCB pointer to be moved to the following CCB in the CCB list for the MLCP channel specified in CF. The two status bytes of *that* CCB (which is now at the top of the CCB list) are then transferred to ML. As shown below, the status byte from CCB byte 7 is transferred to the *left* byte of ML and the status byte from CCB byte 6 is transferred to the *right* byte of ML.

	0 7	8 15
ML	CCB Byte 7	CCB Byte 6
	Status Byte 1	Status Byte 2

The CCB status bytes will contain zero if the CCB has not yet been marked as completed. If the CCB status is zero, then only the Input CCB Status instruction can be used if waiting for completion. If an Input Next CCB Status instruction is issued, then the CCB pointer will move to the next CCB and the previous status will be lost. If the CCB does not complete, the the LCT status bytes 16/48 and 17/49 should be read by using the Input LCT Byte instruction. A decision based on the LCT status can then be made.

In the CCB list, the CCB that was formerly at the top (before this instruction was executed) is now available for re-use.

Note that the IO (Input Next CCB Status) instruction must be used the *first time* status is obtained from a CCB list. This use of the instruction moves the "status" CCB pointer to CCB 1, which is the first CCB used after the CCB list is initialized (as described in Section 3).

Under certain circumstances, an attempt to execute an IO (Input Next CCB Status) instruction will cause the MLCP to issue a NAK; see "Conditions Under Which MLCP Will Issue a NAK" in Appendix A.

IO (Input Extended Identification Number) Instruction

This instruction (function code: 08) transfers, to the main memory program, the identification number that indicates the type of Communications-Pac associated with a specified MLCP channel. It is issued when response to Input Device Identification Number (function code: 26) is 2178_{16}. When this command (Input Extended Identification Number) is issued, the MLCP will return the contents of line register 0 of the Communications-Pac, which is designated to hold an extended ID number for certain Pacs.

The format is as follows.

0 7	8 15
Extended ID	RFU

IOLD (Output CCB Address and Range) Instruction

This instruction (function code: 09) transfers, from "address" and "range" (see format below), the starting byte address and range (in bytes) of a CDB. The transfer is made to the "load" CCB in the CCB list for the MLCP channel specified in CF.

This is an IOLD instruction. Its format is shown below.

IOLD address,CF,range

address

An address expression identifying a memory location or register that indicates the starting byte address of the CDB in main memory.

CF

An address expression identifying a memory location or register that contains a channel number and a function code. The format of CF is the same as that described under CF for an IO instruction, earlier in this section.

range

An address expression identifying a memory location or register that indicates the number of bytes in the CDB. The range must be an integer from 1 to 32,767.

The starting address of the CDB is stored in bytes 0, 1, and 2 of the "load" CCB; the least significant bits of the address are stored in byte 0. The range for the CDB is stored in bytes 3 and 4 of the "load" CGB; the least significant bits of the range are stored in byte 3.

Under certain circumstances, an attempt to execute an IOLD (Output CCB Address and Range) instruction will cause the MLCP to issue a NAK; see "Conditions Under Which MLCP Will Issue a NAK" in Appendix A.

IO (Output CCB Control) Instruction

This instruction (function code: 0F) is used for either of two purposes:

1. It transfers, from the right byte of ML, a control byte to byte 5 of a CCB, resetting bytes 6 and 7 (the status bytes) of the CCB to zero.
2. It transfers, from ML, a RAM address (where a "block mode read" or a "block mode write" will begin) to bytes 5 and 6 of a CCB, resetting byte 7 to zero.

In both cases, execution of this instruction completes CCB setup initiated by an IOLD (Output CCB Address and Range) instruction and moves the "load" CCB pointer to the following CCB in the CCB list.

If a *control byte* is to be transferred to byte 5 of a CCB, ML must be formatted as shown below.

```
        0             7 8            15
ML     |0 0 0 0 0 0 0 0| Control Byte  |
```

The bits in the control byte have the following significance:

Bit 8–interrupt control
- 0 - No action.
- 1 - Interrupt the main memory program when this CCB is marked as completed. (The interrupt will occur at the interrupt level assigned to the related channel.)

Bit 9–"valid" CCB
- 0 - Firmware sets this bit to zero when this CCB has been marked as completed and is therefore no longer "valid" (i.e., usable as an "active" CCB).
- 1 - This CCB is "valid" (i.e., usable as an "active" CCB). This bit must be set to 1 to complete setup of the CCB.

Bit 10–last CDB
- 0 - No action.
- 1 - This CCB pertains to the *last* CDB in a message. If this bit is set to 1, it serves as a flag that can be used by the CCP for special processing of the last CDB in a message. The MLCP's LB-indicator will be set to 1 when this CCB is "active."

Bits 11 through 15–must be zero

If a *RAM address* is to be transferred to bytes 5 and 6 of a CCB, ML must be formatted as shown below.

```
        0     3 4                    15
ML     |0 0 0 0|      RAM Address      |
```

The right byte of this word is transferred to CCB byte 5; the left byte is transferred to CCB byte 6. The 12-bit RAM address indicates the MLCP RAM byte at which the "block mode read" or "block mode write" will begin.

IO (Output Channel Control) Instruction

This instruction (function code: 05) transfers a control word from ML to the MLCP. * Each execution of this instruction affects only *one* MLCP channel. Only one bit in the control word can be set to 1.

The operations achieved by the bits of the control word (if set to 1) are summarized below.

Bit 0–channel initialize
Bit 1–start input/output
Bit 2–stop inout/output
Bit 3–reserved
Bit 4–start "block mode read"
Bit 5–start "block mode write"
Bit 6–reserved
Bit 7–CCB list reset
Bits 8 through 15–reserved

The following actions are performed by the MLCP microprocessor when it receives a control word with a bit set to 1.

Bit 0—channel initialize

- o Resets to zero the data set control bits (bits 0 through 4) in line register 2 of the Communications-Pac.
- o Resets to zero "receive ON" (bit 6) or "transmit ON" (bit 7) in line register 2 of the Communications-Pac. This action prevents subsequent data-generated channel request interrupts.
- o Halts execution of CCP.
- o Stops all activity for this channel.
- o Resets to zero the entire LCT area for this channel (LCT bytes 0 through 31 for receive channel, LCT bytes 32 through 63 for transmit channel).
- o Resets CCB list (see bit 7, below, for a description of this operation).

Bit 1—start input/output

- o Starts input/output. Start I/O causes the CCP to begin execution. The starting address of the CCP is contained in LCT bytes 6 and 7 for receive, and LCT bytes 38 and 39 for transmit. The main memory program must have previously set these locations to the CCP starting address at least once during the program.
- o Starts execution of the CCP. If the CCP is already running and a Start I/O is issued, a loss of data characters may result.

Bit 2—stop input/output

- o Resets to zero "receive ON" (bit 6) or "transmit ON" (bit 7) in line register 2 of the Communications-Pac. This action prevents subsequent data-generated channel request interrupts.
- o Resets to zero "receive ON" (bit 6) or "transmit ON" (bit 7) in the LCT byte 20 copy of LR2.
- o Halts execution of CCP.
- o Resets to zero the LCT status bytes (LCT bytes 16 and 17 for receive channel, LCT bytes 48 and 49 for transmit channel)—*after* they have been transferred to the appropriate CCB.
- o Terminates "active" CCB (with "meaningful" status information) and stops CCB list processing.
- o Inhibits interrupts to the main memory program (but does not change channel's interrupt level).
- o Stops all activity for this channel.

Bit 4—start "block mode read"

This operation is used to read any portion of MLCP RAM. A *receive* (even-numbered) MLCP channel must be designated in CF when this operation is performed.

- o Uses the CCB next in line to be "active" to read a block of consecutive RAM locations into the main memory program. (Details appear in Appendix B.) When the read is completed (CCB range equals zero), the CCB will be marked as completed (with "meaningful" status information). The main memory program will be interrupted at the interrupt level of the receive channel used for the "block mode read" (unless this channel's interrupt level is zero).

Bit 5—start "block mode write"

This operation is used to write a block of consecutive RAM locations. It is a convenient way to write the LCT area and the CCP area. A *transmit* (odd-numbered) channel must be designated in CF when this operation is performed.

Note that the LCT bytes for the MLCP channel used for a "block mode write" cannot themselves be written into at this time because these bytes are used by the firmware during the course of this operation. Also, any *firmware-reserved* LCT bytes written into during a "block mode write" must be written with zeros.

- o Uses the CCB next in line to be "active" to write a block of consecutive RAM locations from the main memory program. (Details appear in Section 7.) When the write is completed (CCB range equals zero), the CCB will be marked as completed (with "meaningful" status information). The main memory program will be interrupted at the interrupt level of the transmit channel used for the "block mode write" (unless this channel's interrupt level is zero).

Bit 6–reserved

Bit 7–CCB list reset

- o Resets the channel's CCB pointers to their initialized state (i.e., the "status" CCB pointer is set to point to CCB 0 of the CCB list and the "load" CCB pointer and "active" CCB pointer are set to point to CCB 1 of the CCB list).
- o Resets the control byte of each of the four CCB's of the channel. This action resets the "valid" bits.

IO (Output Interrupt Control) Instruction

This instruction (function code: 03) transfers, from ML, a return channel number (the central processor's channel number) and an interrupt level to be used during interrupts from the MLCP channel specified in CF. For a receive channel, the left byte of ML is transferred to LCT byte 12 and the right byte is transferred to LCT byte 13. For a transmit channel, the left byte of ML is transferred to LCT byte 44 and the right byte is transferred to LCT byte 45.

The format of ML is shown below.

	0 9	10 15
ML	Return Channel Number	Interrupt Level

IO (Output LCT Byte) Instruction

This instruction (function code: 0B) transfers, from ML, a byte of information to a specific LCT byte address. Bits 0 through 8 of the MLCP channel number specified in CF establish the LCT to which the information is transferred; bit 9 of CF is not meaningful in this case because *one* LCT applies to *both* channels of a line. (The base from which "LCT address" is an offset is byte 0 of the LCT that applies to the line indicated by bits 0 through 8 of CF.)

The format of ML is shown below.

	0 7	8 9	10 15
ML	New LCT Byte	0 0	LCT Address

New LCT byte indicates an 8-bit value to be transferred to the LCT address indicated in bits 10 through 15.

LCT address is relative to byte 0 of the LCT for the line indicated by bits 0 through 8 of CF. The six bits of the LCT address permit *any* of the 64 LCT bytes to be designated.

IO (Output MLCP Control) Instruction

This instruction (function code: 01) transfers, from ML, a control word to the MLCP microprocessor. All MLCP channels are affected by this control word. Any channel can be specified in CF, provided that channel is serviced by a Communications-Pac.

The format of ML is shown below.

	0	1 15
ML	I	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

If *I* is set to 1, MLCP initialization will be performed. Otherwise, no action is taken. MLCP initialization comprises the following actions:

- o The MLCP executes its basic logic test.
- o Each line register 2 of each Communications-Pac is reset to zero; channel request interrupts are thus inhibited.
- o All of RAM is reset to zero.
- o LCT byte 1 of channel 0 contains the hexadecimal number of the firmware revision.
- o All channels are initialized. (This operation is described under the "IO (Output Channel Control) Instruction," earlier in this section.)
- o The MLCP is placed in a quiescent state; no interrupts or data transfers can occur.

*

Soft Initialize

The pressing of the Clear (CLR) push button on the central processor control panel causes the MLCP to perform a "soft initialize" which does the following:

- o Clears the Communications-Pacs
- o Clears LCT bytes 8, 9, 40 and 41 of each channel
- o Clears the MLCP internal registers
- o Runs the basic logic test (BLT)

The soft initialize differs from the hard initialize caused by the command Output MLCP Control (code: 01) in that the hard initialize, in addition to performing the soft initialize functions defined above, also clears the RAM.

When in a software debug mode, it is preferable that the RAM not be cleared so that meaningful dumps can be taken to indicate the MLCP condition.

SECTION 3

COMMUNICATIONS CONTROL BLOCKS

Communications control blocks (CCB's) are data structures in MLCP RAM; they are written from a main memory program and then used by MLCP firmware to control the flow of data between a main memory program and the MLCP. Space for 64 CCB's exists at the high end of MLCP RAM (see Figure 1-2). A "list" of four consecutive CCB's is available for *each* of the 16 communications channels of the MLCP. The CCB list pertaining to channel 0 begins at the low memory end of the CCB area; the remaining CCB lists are arranged by ascending order of channel number toward high RAM. (Refer to Appendix A for a tabular representation.)

Each CCB is used to contain address and range information that describes a communications data block (CDB) in the main memory program. In addition, a CCB must be written with control information from the main memory program before the CDB can be used in a data transfer. Finally, when a data transfer to or from a CDB is complete, the related CCB is updated with status information. Some of this status information is obtained from the line control table (LCT) associated with the communications channel to which the CCB pertains.

Note the definitions of the following terms, which are used throughout this section:

o *"Valid" CCB*—A CCB that has been set up by an IOLD (Output CCB Address and Range) instruction and an IO (Output CCB Control) instruction and is usable (or in use) as an "active" CCB. As many as *four* CCB's in a list can be "valid" at any point in time.

o *"Active" CCB*—The CCB (in its list) that is actively in use by firmware to control a data transfer to or from the related CDB. Only *one* CCB in a list can be "active" at any point in time.

o *"Current" CCB*—The CCB (in its list) that was most recently written by an IOLD (Output CCB Address and Range) instruction. This CCB is at the logical *bottom* of its CCB list.

o *"Load" CCB*—The CCB (in its list) that will be written by the next IOLD (Output CCB Address and Range) instruction.[1]

o *"Status" CCB*—The CCB at the logical *top* of its CCB list. This is the CCB whose status was delivered to the main memory program by the most recent IO (Input Next CCB Status) instruction for this CCB list.

o *"Load" CCB pointer*—This pointer indicates the "load" CCB. This pointer is moved from the "current" CCB to the following CCB by an IO (Output CCB Control) instruction, which at the same time completes setup of the "current" CCB, thus making it "valid."

o *"Active" CCB pointer*—This pointer indicates the "active" CCB. This pointer is moved from the "active" CCB to the following CCB (which then becomes the "active" CCB) by execution of a GNB (Get Next Block) instruction in the channel control program (CCP).

o *"Status" CCB pointer*—This pointer indicates the "status" CCB. This pointer is moved from the "status" CCB to the following CCB (which then becomes the "status" CCB) by an IO (Input Next CCB Status) instruction.

[1] Note that during the interval between execution of an IOLD (Output CCB Address and Range) instruction and execution of an IO (Output CCB Control) instruction, the "current" CCB and the "load" CCB are the *same one*. Thus, if a *second* IOLD (Output CCB Address and Range) instruction is issued *before* an IO (Output CCB Control) instruction is issued, the address field and the range field of the "current"-"load" CCB will be overwritten.

- *CCB list*—A CCB list of four consecutive CCB's exists in upper RAM for *each* of the 16 communications channels of the MLCP. Each CCB comprises eight bytes; each CCB list comprises 32 bytes. The CCB list for channel 0 begins at the bottom of the CCB area; the remaining CCB lists are arranged by ascending order of channel number toward high RAM.

  The programming interface to the CCB list is achieved by means of input/output instructions in the main memory program. These instructions are described in Section 2.

  A CCB list is in its "initialized" state (i.e., the "status" CCB pointer is set to point to CCB 0 of the CCB list and the "load" CCB pointer and "active" CCB pointer are set to point to CCB 1 of the CCB list) immediately after one of the following instructions is executed:

  - IO (Output MLCP Control)
  - IO (Output Channel Control—channel initialize)
  - IO (Output Channel Control—CCB list reset)

  Execution of the first instruction above initializes *all* CCB lists in the MLCP RAM; it also resets all of RAM (including all CCB lists) to zero. Execution of either of the other two instructions initializes the CCB list for only *one* channel. (Note that a CCB list is not in its initialized state if, during processing, it becomes empty because all its CCB's have been used and marked as completed and no other CCB has been written.)

  The physical format of a CCB list is shown below.

CCB 0	8 bytes
CCB 1	8 bytes (used first)
CCB 2	8 bytes
CCB 3	8 bytes

  The physical CCB's in each list are used in "circular" fashion. That is, when a CCB list is first used after being initialized, CCB 1 is written first, CCB 2 is written second, CCB 3 is written third, and CCB 0 is written fourth. (The CCB's will be obtained by the CCP in this same order.) As CCB's are re-used, the same circular order is maintained; CCB 1 is written fifth, CCB 2 is written sixth, and so on.

  The CCB list has a logical top and bottom. The CCB at the *top* of the list is the CCB whose status was delivered to the main memory program by the most recent IO (Input Next CCB Status) instruction; this CCB is known as the "status" CCB. The CCB at the *bottom* of the list is the CCB that was most recently written by an IOLD (Output CCB Address and Range) instruction; this CCB is known as the "current" CCB.

  The CCB list is processed dynamically. As a CCB at the logical top of the list (e.g., CCB 2) is marked as completed, the main memory program can obtain its status information; then the main memory program can move the "status" CCB pointer to the following CCB (CCB 3) by use of an IO (Input Next CCB Status) instruction. The CCB formerly at the top of the list (CCB 2) is now available for re-use. Similarly, as each CCB is written, a new logical bottom is established for the list. CCB list processing continues with "deletions" from the logical top of the list and "additions" to the logical bottom of the list, according to the requirements of the communications application.

The IO (Input CCB Status) and IO (Input CCB Range) instructions always obtain information from the "status" CCB, at the top of the CCB list. Typically, this CCB is either still active or it is marked as completed, with some action pending for its status or range information. The IO (Input Next CCB Status) instruction—as mentioned above—causes the "status" CCB pointer to be moved from the CCB at the top of the list to the following CCB.

Typically, the IO (Output Channel Control–start input/output) instruction is used to initiate processing with the CCB next in line to be active. Note that this instruction can be executed before this CCB becomes the "status" CCB.

The IOLD (Output CCB Address and Range) instruction establishes a new logical bottom of the CCB list. The new logical bottom is the physical CCB one beyond the previous logical bottom of the list (using the circular order described above). The IO (Output CCB Control) instruction refers to the CCB at the logical bottom of the list; this instruction writes the control field in the CCB, completing setup of that CCB, and moves the "load" CCB pointer to the following CCB in the list.

CCB list setup may continue until four CCB's have been set up. At this point, an IO (Input Next CCB Status) instruction must be issued before another CCB can be set up.

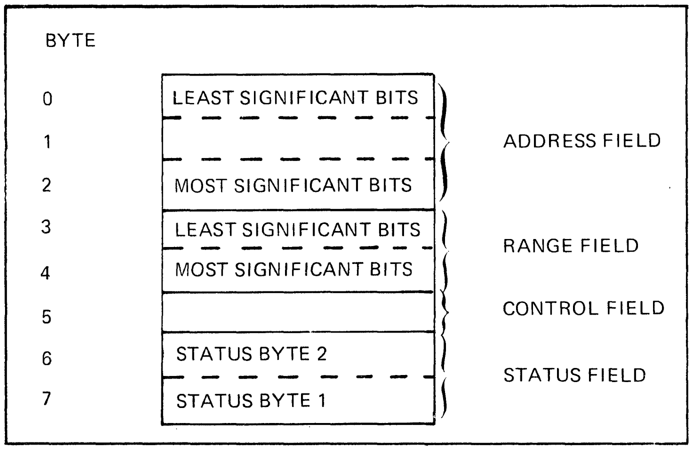
CCB list processing can be started or stopped by use of the IO (Output Channel Control) instruction. Setting bit 1 (start input/output) in the control word used with this instruction starts CCB list processing; setting bit 2 (stop input/output) in this control word stops CCB list processing. When CCB list processing is stopped in this manner, two status bytes in the "active" CCB are updated; the status complete bit for this CCB is set to 1. If CCB list processing is stopped and then restarted by use of IO (Output Channel Control) instructions, the CCB next in line to become active will be used.

The CCB area of RAM is accessible, if desired, through a "block mode read" operation. See Appendix B.

CCB FORMAT

Figure 3-1 shows the format of a CCB used to control a communications data transfer. The fields of the CCB are described in the following subsections.

A specially formatted CCB is used to control a "block mode" input/output operation from/to MLCP RAM. For more information about this "special" type of CCB, see "Format of CCB for 'Block Mode Write'" and "CCB Status Field After 'Block Mode Write'" in Section 7.



BYTE		
0	LEAST SIGNIFICANT BITS	ADDRESS FIELD
1		
2	MOST SIGNIFICANT BITS	
3	LEAST SIGNIFICANT BITS	RANGE FIELD
4	MOST SIGNIFICANT BITS	
5		CONTROL FIELD
6	STATUS BYTE 2	STATUS FIELD
7	STATUS BYTE 1	

Figure 3-1. Format of a CCB

CCB Address Field

The *address* field occupies bytes 0, 1, and 2 of the CCB. This field is written from the main memory program by an IOLD (Output CCB Address and Range) instruction. When first written, the address field contains the starting *byte* address of a CDB in the main memory program; the low-order end of the starting byte address is contained in byte 0 of the CCB.

The contents of the address field are increased as data characters are physically transferred between the CDB and the MLCP based on the CCP's execution of format 1 LD (Load) or ST (Store) instructions.

The address field value is increased by *1* each time execution of a format 1 LD (Load) or ST (Store) instruction causes *one* data character to be physically transferred between the CDB and the MLCP. This case applies only upon execution of the first (or last) LD or ST instruction pertaining to a CDB that begins (or ends) at an odd byte boundary.

The address field value is increased by *2* each time execution of a format 1 LD (Load) or ST (Store) instruction causes *two* data characters to be physically transferred between the CDB and the MLCP. This case applies upon execution of the *first* of two LD instructions or upon execution of the *second* of two ST instructions. (Execution of the *second* LD instruction or of the *first* ST instruction does not cause a physical data transfer between the CDB and the MLCP; hence the address field value is not increased as either of these instructions is executed.)

CCB Range Field

The *range* field occupies bytes 3 and 4 of the CCB. This field is written from the main memory program by an IOLD (Output CCB Address and Range) instruction. When first written, the range field indicates the number of bytes in the CDB. The low-order end of the range value is contained in byte 3 of the CCB.

The contents of the range field are decreased by 1 each time a format 1 LD (Load) or ST (Store) instruction is executed in the CCP (regardless of whether that instruction causes a physical data transfer between the CDB and the MLCP).

CCB Control Field

The *control* field occupies byte 5 of the CCB. The control field is written from the main memory program by an IO (Output CCB Control) instruction. The format and significance of this field are shown below.

0	1	2	3	4	5	6	7
I	V	LB	0	0	0	0	0

Bit 0—interrupt control

0 - No action.

1 - Interrupt the main memory program when this CCB is marked as completed (CCB byte 7, bit 3). The interrupt will occur at the interrupt level assigned to this channel. If no interrupt level has been assigned, no interrupt will occur.

Bit 1—"valid" CCB

0 - This is not a "valid" CCB; it cannot be used as an "active" CCB. This condition exists before this bit is set to 1 by an IO (Output CCB Control) instruction. This bit is reset to 0 by firmware when this CCB is marked as completed (CCB byte 7, bit 3) after it has been used during processing of a CDB.

1 - This is a "valid" CCB; it is usable as an "active" CCB. This bit must be set to 1 to complete setup of the CCB.

Bit 2–last CDB

0 - No action.

1 - This CCB pertains to the last CDB in a message. This is a flag that can be used by the CCP for special processing of the last CDB in a message. If this bit is set to 1, the MLCP's LB-indicator will be set to 1 when this CCB is "active." The CCP can test the LB-indicator by means of BLBT (Branch if Last Block True) and BLBF (Branch if Last Block False) instructions.

CCB Status Field

The *status* field comprises bytes 6 and 7 of the CCB. The CCB status field is reset to zero as setup of the CCB is completed by execution of an IO (Output CCB Control) instruction. Later, as processing ends relative to a CCB, its status field is updated by firmware and the CCB status complete bit (CCB byte 7, bit 3) is set to 1. (Table 3-1 indicates the conditions under which processing relative to a CCB can end.)

The CCB status field is updated with information from the two LCT status bytes combined with other information. The LCT status bytes are bytes 16 and 17 for a receive channel and bytes 48 and 49 for a transmit channel. Once the status field of the CCB has been updated, the CCB's status is said to be "meaningful."

The status bytes of the "status" CCB can be read from the main memory program, whenever appropriate, by an IO (Input CCB Status) instruction. An IO (Input Next CCB Status) instruction moves the "status" CCB pointer to the following CCB (which then becomes the "status" CCB) and reads the status field of this new "status" CCB.

The format of the two bytes of the CCB status field is shown below. The shaded bit positions are those to which information is passed from the LCT status bytes. Note that, in the CCB status field, status byte 1 is stored *above* status byte 2; this order is the opposite of the order of the status bytes in the LCT.

	0	1	2	3	4	5	6	7
BYTE 6 CCB STATUS BYTE 2	RESERVED	DATA CHECK ERROR	RECEIVE NONZERO RESIDUAL RANGE TRANSMIT LAST BLOCK (SEE NOTE)	DATA SET OR COMMUNICATIONS-PAC STATUS CHANGE	CORRECTED MEMORY ERROR	INVALID MEMORY ADDRESS	MEGABUS PARITY ERROR	UNCORRECTED MEMORY ERROR
BYTE 7 CCB STATUS BYTE 1	INTERRUPT MAIN MEMORY PROGRAM FROM CCP	INTERRUPT MAIN MEMORY PROGRAM FROM CCB	DATA SERVICE ERROR	CCB STATUS COMPLETE	CCB SERVICE ERROR	FOR PROGRAMMING USE	FOR PROGRAMMING USE	RESERVED

NOTE: For transmit, bit 2 equals last CCB block.

CCB Byte 7 (Status Byte 1)

Bit 0–interrupt main memory program from CCP

0 - No action.

1 - The main memory program has been interrupted due to execution of an INTR instruction in the CCP.

Bit 1–interrupt main memory program

0 - No action.

1 - The main memory program has been interrupted when processing ends relative to this CCB. This bit is set to 1 in either of two cases: (1) if bit 0 of CCB byte 5 has been set to 1–by an IO (Output CCB Control) instruction in the main memory program or (2) if bits 0 and 2 of LCT byte 8/40 have been set to 1 and a data set or Communications-Pac status change has been recorded in LCT byte 14/46 (Data Set Scan).

Bit 2—data service error

0 - No data service error has occurred.

1 - A data "timing window" has been missed. On receive, the Communications-Pac has detected a receive overrun (see bit 6 of LCT byte 14/46 in Section 5). On transmit, the Communications-Pac has detected a transmit underrun (see bit 7 of LCT byte 14/46 in Section 5).

Bit 3—CCB status complete

This bit is always set to 1 as the CCB status field is written by MLCP firmware. This setting indicates that processing relative to this CCB has ended and the contents of its status field are meaningful. Table 3-1 indicates the conditions under which processing relative to a CCB can end.

Bit 4—CCB service error

0 - No CCB service error has occurred.

1 - This bit setting pertains to an error that occurred before this CCB became "valid."

On receive, a format 1 ST (Store) instruction was attempted when there was no "valid" CCB. The instruction was not executed; instead, MLCP firmware set this bit to 1 (in LCT status byte 1) and proceeded to the next sequential instruction in the CCP.

On transmit, a format 1 LD (Load) instruction was attempted when there was no "valid" CCB. The instruction was not executed; instead, MLCP firmware set this bit to 1 (in LCT status byte 1), returned the CCP pointer to the address of this LD instruction, and executed a WAIT (Wait) instruction. At the next channel request interrupt for this channel, this instruction was attempted again.

See the description of bit 4 of LCT byte 16/48 in Section 5.

Bits 5 and 6—for programming use

Within LCT status byte 1, these two bits can be used by the CCP for application-specific purposes. Later, when the contents of the LCT status bytes are transferred to the CCB status field, these two bit positions become available for scrutiny by the main memory program as it issues an IO (Input CCB Status) or IO (Input Next CCB Status) instruction. Thus, these two bit positions can be used as a means for the CCP to pass application-specific status information to the main memory program.

CCB Byte 6 (Status Byte 2)

Bit 1—data check error

0 - No data check error has occurred.

1 - A data parity error has been detected by firmware, or the CCP has set this bit after detecting a cyclic redundancy check error. (In both cases, this bit setting is relevant only to receive operations.)

Bit 2—CCB nonzero range residue for receive only (see "NOTE" on previous page under diagram)

0 - No CCB range residue exists.

1 - The CCB has been terminated before its range field value decreased to 0.

Bit 2—last block for transmit only

0 - Not last block

1 - Last block

Bit 3—data set or Communications-Pac status change

0 - No data set or Communications-Pac status change has been recorded.

1 - Bits 0 and 1 of LCT byte 8/40 were set to 1 and a data set or Communications-Pac status change was recorded in LCT byte 14/46.

Bit 4—corrected memory error

0 - No corrected memory error has occurred.

1 - One or more hardware-corrected memory errors occurred in the CDB related to this CCB.

Bit 5—invalid memory address

0 - No invalid memory address has occurred.

1 - A reference to a CDB has resulted in an invalid memory address on the Megabus; main memory has issued a NAK. This condition has caused the CCB to be terminated.

Bit 6—Megabus parity error

0 - No Megabus parity error has occurred.

1 - Incorrect parity existed on the Megabus as a data character was transferred *to* the MLCP. This condition has caused the CCB to be terminated.

Bit 7—uncorrected memory error

0 - No uncorrected memory error has occurred.

1 - An uncorrected memory error occurred in the CDB related to this CCB. This condition has caused the CCB to be terminated.

WRITING A CCB[2]

A CCB is normally set up by the execution of two instructions by the main memory program. The first instruction is an IOLD (Output CCB Address and Range), which writes the CCB's address and range fields with appropriate values. The second instruction is an IO (Output CCB Control), which writes an appropriate value into the CCB's control field and resets the CCB's status field to zero. A CCB is not available for use until it has been set up by these two instructions.

CCB DESCRIPTION OF A CDB

When a CCB is first written by an IOLD (Output CCB Address and Range) instruction, it identifies the starting byte address of a CDB in the main memory program. The starting byte address can be either the first (left) or second (right) byte of a word in main memory. (Likewise, the CDB can end with the first or second byte of a main memory word.)

A CDB is a consecutive series of bytes in main memory. It can be used as either an input (receive) buffer or an output (transmit) buffer during data transfers to or from the MLCP.

Data in a CDB is arranged in increasing order from the lowest byte address towards the highest byte address in the CDB. The first character transmitted or received is contained in the lowest byte address of the CDB.

Note that the contents of the CCB's address field are increased only as data characters are *physically* transferred between the CDB and the MLCP; see "CCB Address Field," earlier in this section. The contents of the CCB's range field decrease by 1 each time a format 1 LD (Load) or ST (Store) instruction is executed in the CCP (regardless of whether that instruction causes a physical data transfer between the CDB and the MLCP).

PROCESSING OF AN "ACTIVE" CCB

A CCB is "active" when it is being used by MLCP firmware to control the flow of data to or from a CDB. A CCB becomes "active" when the CCP issues a GNB (Get Next Block) instruction that moves the "active" CCB pointer to it. During active processing, the CCB is used only by MLCP firmware; it is not accessible from the CCP.[3]

[2] For a description of how to set up a CCB to control a block mode input/output operation, see "Format of CCB for 'Block Mode Write'" in Section 7.

[3] Exception: The "valid" and LB (last block) bits may be tested by the BVBT, BVBF, and BLBT, BLBF instructions, respectively.

As data characters are physically transferred to or from the CDB by way of the MLCP, the contents of the CCB address field are increased appropriately; see "CCB Address Field," earlier in this section. If the CCB describes a CDB input (receive) buffer, the CCB address field points to the location that will be used to store the next data character to be received from the MLCP. If the CCB describes a CDB output (transmit) buffer, the CCB address field points to the next data character to be transferred from main memory to the MLCP.

The contents of the CCB range field are decreased by 1 as each format 1 LD (Load) or ST (Store) instruction is executed by the CCP (regardless of whether that instruction causes a physical data transfer between the CDB and the MLCP). This process continues until the value of the range field decreases to zero or—as in the case of a receive operation, until a termination condition (EOT, EOB, etc.) establishes the end of a data block, causing a nonzero range residue. Range residue in the range field indicates the number of bytes in the CDB to or from which direct memory access data transfers had not been made when the CCB was marked as completed.

When processing of an "active" CCB is completed, the status of the related data transfer is recorded in the two status bytes of the CCB. (Some of the status information comes from the LCT status bytes; see "CCB Status Field," earlier in this section.) At this point, the CCB status complete bit (CCB byte 7, bit 3) is set to 1.

COMPLETION OF A CCB

Table 3-1 describes various conditions that cause a CCB to be marked as completed and the means by which each condition can be ascertained by the main memory program.

TABLE 3-1. CCB COMPLETION CONDITIONS

CCB Completion Condition	Action by Which Main Memory Program Can Ascertain Condition
CCP has issued a GNB (Get Next Block) instruction, causing the "active" CCB to be marked as completed.	CCP could indicate this action by setting bit 5 or 6 in LCT byte 16 (for receive channel) or LCT byte 48 (for transmit channel) before issuing the GNB instruction. This information would then be available to the main memory program through bit 5 or 6 of CCB byte 7.
Main memory program has issued an IO (Output Channel Control—stop input/output) instruction.	Self-evident.
* Invalid memory address.	IO (Input CCB Status) or IO (Input Next CCB Status). CCB byte 6, bit 5 is set to 1.
Megabus parity error.	IO (Input CCB Status) or IO (Input Next CCB Status). CCB byte 6, bit 6 is set to 1.
Uncorrected memory error.	IO (Input CCB Status) or IO (Input Next CCB Status). CCB byte 6, bit 7 is set to 1.

SECTION 4

CHANNEL CONTROL PROGRAM

CCP STRUCTURE AND COMPONENTS

A channel control program (CCP) must be created and then stored in MLCP RAM, where it serves as the interface between one or more MLCP communications channels and communications data blocks (CDB's) in the main memory program. Each CCP handles a communications data stream being received by, or transmitted from, these CDB's. The data stream is handled one character at a time, and the CCP can modify or delete an individual character in the data stream or it can transfer the character unchanged. The CCP can also manipulate certain bytes in the line control table (LCT) pertaining to the channel serviced by the CCP; this LCT information relates to communications control blocks (CCB's), the Communications-Pac, and data communications equipment.

CCP Setup

A CCP must be coded as a set of macro calls and processed by the appropriate operating system macro preprocessor before being assembled. A detailed description of each CCP generation control statement and executable instruction, along with the proper macro call format, appears later in this section.

Each CCP in the MLCP must reside in consecutive locations of the CCP area of RAM, which extends from byte 512 to byte 3583, inclusive (see Figure 1-2). Multiple CCP's can coexist in RAM—provided they do not overlap.

A CCP can service more than one communications channel, but each channel's LCT and CCB's exist in channel-specific RAM locations outside the CCP area, regardless of whether that channel is serviced by a "dedicated" CCP or by a CCP that services multiple channels. The channel-specific LCT and CCB storage areas permit CCP's to be reentrant and therefore able to service more than one channel.

MLCP firmware allows a CCP to call one level of subroutine (outside the CCP's consecutive RAM locations) and later to be returned to at the next sequential instruction following the call.

A CCP is stored in MLCP RAM by the MLCP Loader or by a main memory program's use of a "block mode write" operation. (These subjects are described in Section 7.) In either case, the CCP's initial starting address must be written into the appropriate bytes of the LCT for the channel to be serviced by this CCP. (The format of LCT's is described in Section 5.) When the CCP is started for the first time, its initial starting address—stored in the LCT—will be loaded into the MLCP's P-register (program counter) by MLCP firmware.

Starting CCP

Once all desired CCP's have been stored in RAM and all setup activity has occurred, the CCP can be started by the main memory program's execution of an IO (Output Channel Control—start input/output) instruction. Normally, the first action of the CCP is to load line registers 6, 4, and 2 of the appropriate channel of the Communications-Pac; line register 2 must be loaded last. The CCP may then execute a WAIT (Wait) instruction, pending the first communications message activity.

During processing, a CCP can be started by any of the following means:

o A channel request interrupt from the appropriate Communications-Pac (a request for CCP service).

- Execution of an IO (Output Channel Control—start input/output) instruction in the main memory program.
- A change in data set status (provided the firmware scan bit and the start CCP bit are both set to 1 in the appropriate LCT byte).
- Either channel of a line pair can turn on the other channel by setting the appropriate bit (6, 7) of the LCT byte 20 and line register 2 (LR2). This technique is used for echoplexing the received data back to the terminal in full-duplex mode, or for managing line turnaround in two-way alternate mode.

Each time a CCP is started, the MLCP restores its channel-specific "context" from the appropriate LCT. This context includes the proper settings of the MLCP P-register (program counter), R-register (general register), and program indicators.

CCP Execution

When the CCP is started, the MLCP allows it to execute, without interruption, as many as 31 instructions. (Communications-Pac buffering is provided to ensure that consecutive execution of 31 instructions in one CCP does not cause an error—receive overrun or transmit underrun—on another communications channel.) After 31 instructions have been executed, a firmware pause occurs and the CCP is interrupted; the CCP's context is stored in firmware-reserved bytes of the appropriate LCT. The firmware pause allows background firmware scanning to occur and channel request interrupts to be serviced. When the CCP is resumed following the pause, its saved context is automatically restored by firmware; the pause is not apparent to the CCP unless it contains a timing loop.

The data character processing loop of a CCP typically handles a single character of the data stream and terminates with a WAIT (Wait) instruction. When the CCP's WAIT instruction is executed, MLCP firmware stores, in the appropriate LCT, the current contents of the P-register, R-register, and program indicators. This context will be restored by the MLCP when this CCP is started again.[1]

MLCP Registers and Program Indicators Used by CCP

The MLCP registers and program indicators of particular significance to the CCP are as follows:

- *P-register (program counter)*—a 12-bit register that contains the RAM address of the next CCP instruction to be executed.
- *R-register*—an 8-bit general register used by CCP instructions.
- *E(Equal)-indicator*—an indicator that stores the results of the last execution of a C (Compare) instruction. If the comparison was equal, the E-indicator is set to 1 (true); if the comparison was unequal, the E-indicator is reset to 0 (false). This indicator can be tested by the BET (Branch if Equal True) and BEF (Branch if Equal False) instructions.
- *LC(Last Character)-indicator*—an indicator that is set to 1 (true) after execution of a format 1 LD (Load) or ST (Store) instruction that has caused the value of the CCB range field to reach zero. The LC-indicator remains set to 1 until the first format 1 LD or ST instruction is executed for the next CCB, at which time the LC-indicator is reset to 0 (false). This indicator can be tested by the BLCT (Branch if Last Character True) and BLCF (Branch if Last Character False) instructions.
- *LB(Last Block)-indicator*—an indicator that is set to 1 (true) when the "active" CCB describes the last CDB in a message (the LB-indicator will be set to 1 *provided* the LB-bit in this CCB's control field was set to 1 when this CCB was set up). At other times the LB-indicator is reset to 0 (false). This indicator acts as a flag to the CCP—it can be tested by the BLBT (Branch if Last Block True)

[1] If, desired, the main memory program can alter the stored value of the P-register by issuing IO (Output LCT Byte) instructions only when the CCP is not executing; this action will cause the CCP to resume at a different RAM address when it is started again.

and BLBF (Branch if Last Block False) instructions. It is not used by MLCP firmware. This indicator is not valid until at least one LD or two ST (format 1) instructions have been executed by the CCP.

BV(Block Valid)-indicator—this indicator is the valid bit, bit 1 of the CCB control byte.

AR(Adapter Ready)-indicator—this indicator, applicable only to broadband Communications-Pacs, means, on the transmit channel, that the buffer is not full, or, on the receive channel, that the buffer is not empty.

All communications data is stored right-justified in each Communications-Pac line register 1 and in the MLCP's R-register; the same format is used in both registers. Parity, if used, is stored in the leftmost bit of the communications character.

The CCP's send/receive instructions and the CCH (Calculate Block Check) instruction can use the MLCP's block-check hardware to support cyclic redundancy checking.

USING CCP GENERATION CONTROL STATEMENTS AND EXECUTABLE INSTRUCTIONS

CCP control statements and executable instructions are used to create the channel program. They are coded as macro calls and constitute the source of the CCP. This source is processed by the macro preprocessor and then the expanded source module produced by the macro preprocessor is used as input to the assembler.

Program Development Tools

Before attempting to create a CCP, you should be thoroughly familiar with the description of the macro facility in the appropriate assembly language manual. Refer also to the documentation listed in the appropriate software overview manual (listed in the Preface). You should be familiar with the program development tools of the operating system, and with the operating system diskette that contains the macro preprocessor and assembler, and the library of MLCP macro routines. The library of macro routines is used whenever the macro preprocessor services a CCP.

Programming Rules

The CCP instructions that are described in the remainder of this section are those control and executable instructions that are used to create channel programs in the MLCP.

In a few instances, assembly language instructions have mnemonic op codes that are identical to the macro-names of CCP generation control statements and CCP executable instructions (e.g., ORG, NOP, AND, OR, XOR, B, DEC). If any of these assembly language instructions appear in a source module that includes a CCP, they must be "protected" from being misinterpreted as macro calls during execution of the macro preprocessor. See the appropriate assembly language manual for a definition.

Note also that the global macro variables GX, GY, and GZ are reserved for use by the MLCP macro routines and should not appear anywhere in input to the macro preprocessor.

The use of assembler control and executable instructions in the CCP is not permitted. A single exception exists to this rule, namely the assembler instruction EQU (Equate), which may be used within the CCP. The EQU has no effect during the macro preprocessor phase, but, rather, is passed along to the assembler and assembled during assembly of the CCP. The operand of the EQU statement must be a value, i.e., $ cannot be used as an operand. Refer also to the LOC statement defined later.

Macro Preprocessor and Assembly Operation

Refer to the macro preprocessor section of the assembly language manual for the correct control information that must be provided to the macro preprocessor itself and to the assembler portion for proper operation.

Internal Formats

The assembler manual also includes a discussion of the internal data formats and hardware registers in the Level 6 system. *These should be thoroughly understood before attempting to program the MLCP.* To assist the user, the CCP instructions in this manual contain the ranges for internal value expressions that are generated; the definitions of internal value expressions, constants, arithmetic expressions, etc. are found in the assembler manual itself.

In the coding examples used for the CCP instructions, representative types of coding usage are shown.

MLCP Loader

The MLCP Loader is used to load the image text that is the output of the CCP program development procedure. Refer to Section 7 for a description of the Loader.

CCP GENERATION CONTROL STATEMENTS

Four CCP generation control statements are available to assist in the creation of a CCP:

- o LOC
- o ORG
- o MORG
- o DATA

These statements, which are analogous to Assembler control statements, are described below.

LOC Statement

Format of Macro Call:

LOC operand [comments]

operand

A user-supplied label.

Description of Statement:

The LOC statement assigns a user-supplied label to the immediately following CCP byte location. The "LOC label" statement is comparable to the Assembler control statement "label EQU $".

Example:

```
LOC START
```

ORG Statement

Format of Macro Call:

ORG operand [comments]

operand

A decimal integer constant or a hexadecimal integer constant. The value of a decimal integer constant can be from 0 to 3583, inclusive; the value of a hexadecimal integer constant can be from X'0' to X'0DFF' inclusive.

Description of Statement:

The ORG statement assigns a user-supplied value to the Macro Preprocessor's byte allocation counter. The CCP locations following the ORG statement will have byte addresses based on the value supplied in the ORG statement.

NOTE: The "addresses" established by the ORG statement do not necessarily indicate the MLCP RAM locations into which the generated CCP will eventually be loaded. This decision need not be made until the MLCP is loaded.

If the first CCP generation control statement is not an ORG statement, the Macro Preprocessor assumes an implied ORG statement whose operand is zero.

Example:

```
ORG 256
ORG X'0100'
```

MORG Statement

Format of Macro Call:

MORG [comments]

Description of Statement:

The MORG statement assigns, to the Macro Preprocessor's byte allocation counter, a modulo 2 value relative to the address of the most recent TLU (Table Look-Up) instruction. If no TLU instruction has yet appeared in the CCP, the assigned modulo 2 value is relative to the CCP's most recent ORG statement (either explicit or implied; see preceding subsection).

This statement is used to ensure that a branch resulting from a TLU (Table Look-Up) instruction will reach the proper memory address relative to the address of the TLU instruction. (The TLU instruction is described later in this section.)

The use of the MORG statement is illustrated in the sample table look-up program shown later in this Section (Figure 4-1).

Example:

```
MORG
```

DATA Statement

Format of Macro Call:

DATA OPERAND[,operand. . .] [comments]

operand

An internal value expression that, when resolved, is a data constant modulo 256 (value from 0 to 255, inclusive); this data constant will be included in the CCP. From 1 to 35 operands can be included in a DATA statement.

Generated Code:

```
0             7
+---------------+
| data constant |    0 ≤ value ≤ 255
+---------------+
```

Description of Statement:

The DATA statement creates a string of 1 to 35 data constant bytes in the CCP. One use of this statement is to create a CCP table that can be used in conjunction with TLU (Table Look-Up) instructions. (The TLU instruction is described later in this section.)

Examples:

```
DATA 0,1,2,3
DATA X'01',X'02',X'03'
DATA NUL,SOH,STX,ETX
DATA (START-BASE)/2+X'80'
```

CCP EXECUTABLE INSTRUCTIONS

Five types of executable instructions are available to the CCP:

- o Branch instructions
- o Double operand instructions
- o Input/output instructions
- o Send/receive instructions
- o Generic instructions

These instructions are listed in Table 4-1 and described in the following subsections.

TABLE 4-1. CCP EXECUTABLE INSTRUCTIONS

Branch Instructions		
B	Branch	
BS	Branch to Subroutine	
BET	Branch if Equal True	
BEF	Branch if Equal False	
BZT	Branch if Zero True	
BZF	Branch if Zero False	
BLCT	Branch if Last Character True	
BLCF	Branch if Last Character False	
BLBT	Branch if Last Block True	
BLBF	Branch if Last Block False	
BART	Branch if Adapter Ready True	
BARF	Branch if Adapter Ready False	
BVBT	Branch if Valid CCB True	
BVBF	Branch if Valid CCB False	
JUMP	Branch to any RAM Memory Location (3 byte)	
Double Operand Instructions–Format 1		
LD	Load	MLCP R-Register and Next Byte in Appropriate CDB
ST	Store	
Double Operand Instructions–Format 2		
LD	Load	MLCP R-Register and LCT Byte n
ST	Store	
C	Compare	
AND	Logical AND	
OR	Inclusive OR	
XOR	Exclusive OR	
TLU	Table Look-Up	
Double Operand Instructions–Format 3		
LD	Load	MLCP R-Register and Immediate Operand
C	Compare	
AND	Logical AND	
OR	Inclusive OR	
XOR	Exclusive OR	
Input/Output Instructions		
IN	Input	Communications-Pac Line Registers 0-7
OUT	Output	
Send/Receive Instructions		
SEND	Send	Type 0-3
RECV	Receive	

TABLE 4-1 (cont). CCP EXECUTABLE INSTRUCTIONS

Generic Instructions	
NOP	No Operation
WAIT	Wait
GNB	Get Next Block
SFS	Search for Synchronization
CCH	Calculate Block Check
DEC	Decrement R-Register
RET	Return From Subroutine
SR	Shift R-Register Right
INTR	Interrupt CPU
INZ	Soft Initialize all Channels (debug use only)

Branch Instructions

Fifteen branch instructions are available to the CCP.

Format of Short Displacement Macro Call:

macro-name operand [comments]

macro-name

B BS BET BEF BZT BZF BLCT BLCF BLBT BLBF BART BARF BVBT BVBF

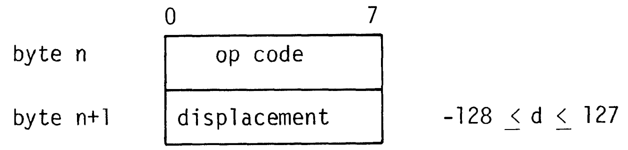
operand

An internal value expression that identifies the target of the branch instruction. The displacement d (between the current byte address and the target byte address) must be in the following range:

$$-128 \leqslant d \leqslant 127$$

The displacement is from P (the current value of the MLCP P-register, which is pointing to the byte that contains the displacement).

Generated Code:



Format of Long Displacement Macro Call:

macro-name operand [comments]

macro-name

JUMP

operand

An internal value expression that identifies the target of the branch instruction. The displacement d (between the current byte address and the target byte address) must be in the following range:

$$-4096 \leqslant d \leqslant +4096$$

Generated Code:

```
0              7
+---------------+
|      OP       |
+---------------+
| X X X X 4 M S B |
+---------------+
|    8 L S B    |
+---------------+
```

```
D = 12 bit displacement

    -4096 ≤ D ≤ +4096
(Assume P is pointing to
 the third location when
 the effective address is
 calculated.)
```

"D" is a twos complement integer. The four bits marked XXXX will be disregarded by the MLCP firmware. Because the branch displacement may be equal to the address space size of the MLCP memory, care should be taken to ensure that the CCP does not branch into the CCB or the LCT area. The MLCP does not include protection in this instance.

Description of Instructions:

B (Branch)–Unconditional branch to the target location.

BS (Branch to Subroutine)–Store the address of the next CCP instruction (i.e., P+1) in firmware-reserved bytes of the LCT; next, branch to the target location.

BET (Branch if Equal True)–Branch to the target location only if the MLCP E-indicator is set to 1.

BEF (Branch if Equal False)–Branch to the target location only if the MLCP E-indicator is reset to 0.

BZT (Branch if Zero True)–Branch to the target location only if the MLCP R-register contains zero.

BZF (Branch if Zero False)–Branch to the target location only if the MLCP R-register contains a nonzero value.

BLCT (Branch if Last Character True)–Branch to the target location only if the MLCP LC-indicator is set to 1.

BLCF (Branch if Last Character False)–Branch to the target location only if the MLCP LC-indicator is reset to 0.

BLBT (Branch if Last Block True)–Branch to the target location only if the MLCP LB-indicator is set to 1.

BLBF (Branch if Last Block False)–Branch to the target location only if the MLCP LB-indicator is reset to 0.

BART (Branch if Adapter Ready True)–Branch to the target location only if the adapter data buffer is not full (Broadband Communications-Pacs).

BARF (Branch if Adapter Ready False)–Branch to the target location only if the adapter data buffer is full (Broadband Communications-Pacs).

BVBT (Branch if CCB Valid True)–Branch to the target location only if valid block is true.

BVBF (Branch if CCB Valid False)–Branch to target location if valid block is false.

Examples:
The B (Branch) instruction is used in the examples; however, the format for all the branch instructions, both short and long displacement, is the same.

```
B START
B START+4
B START-DIF+X'13'
```

Double Operand Instructions

Double operand instructions are usable in three formats:

Format	Operands
1	R-register and next byte in appropiate CDB
2	R-register and byte n of appropriate LCT
3	R-register and immediate operand

The use of double operand instructions in these three formats is described in the following subsections. Note that not all double operand instructions can be used in all formats.

Format 1

Double operand instructions in this format refer to the MLCP R-register and the next byte in the appropriate CDB of the main memory program.

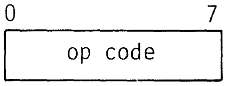
Double operand instructions in this format will not be executed if the CCB list for the appropriate channel does not contain a "valid" CCB.

Format of Macro Call:

macro-name[,comments]

macro-name
  LD ST

Generated Code:



Description of Instructions:

LD (Load)—Load the next byte from the main memory program CDB into the MLCP R-register.

After execution of this instruction, the contents of the "active" CCB's *range* field will be decreased by 1; the contents of the "active" CCB's *address* field may remain unchanged, increase by 1, or increase by 2—depending on circumstances (see "CCB Address Field" in Section 3).

This instruction (in format 1) is valid only in transmit mode.

ST (Store)—Store the contents of the MLCP R-register into the next byte location in the main memory program CDB.

After execution of this instruction, the contents of the "active" CCB's *range* field will be decreased by 1; the contents of the "active" CCB's *address* field may remain unchanged, increase by 1, or increase by 2—depending on circumstances (see "CCB Address Field" in Section 3).

This instruction (in format 1) is valid only in receive mode.

Examples:

LD
ST (no operand)

Format 2

Double operand instructions in this format refer to the MLCP R-register and byte n of the LCT for the channel serviced by the CCP.

All of the double operand instructions in this format are valid in either transmit or receive mode.

Format of Macro Call:

macro-name operand [comments]

macro-name

LD ST C AND OR XOR TLU

operand

An internal value expression that, when resolved, is an integer value from 0 to 63, inclusive. This integer value is the number of a byte in the LCT for the channel serviced by this CCP. (As indicated in Section 5, certain LCT bytes are reserved for firmware use and must not be modified by software.)

Generated Code:

```
               0               7
              +-----------------+
byte n        |     op code     |
              +-----------------+
byte n+1      |  byte n of LCT  |       0 ≤ n ≤ 63
              +-----------------+
```

Description of Instructions:

LD (Load)—Load into the MLCP R-register the contents of byte n of the LCT for this channel.

ST (Store)—Store the contents of the MLCP R-register in byte n of the LCT for this channel.

C (Compare)—Compare the contents of the MLCP R-register with the contents of byte n of the LCT for this channel. If the comparison is equal, set the MLCP E-indicator to 1; otherwise, reset it to 0.

AND (Logical AND)—Perform a logical AND operation on the contents of the MLCP R-register and the contents of byte n of the LCT for this channel. Store the results of this operation in the MLCP R-register.

OR (Inclusive OR)—Perform an inclusive OR operation on the contents of the MLCP R-register and the contents of byte n of the LCT for this channel. Store the results of this operation in the MLCP R-register.

XOR (Exclusive OR)—Perform an exclusive OR operation on the contents of the MLCP R-register and the contents of byte n of the LCT for this channel. Store the results of this operation in the MLCP R-register.

TLU (Table Look-Up)—This instruction permits the contents of a "table location" in RAM to be evaluated so as to produce either a "translation" of the contents of that location or a branch to another location in the CCP. Execution of this instruction produces the following sequence of events:

Examples:

```
LD 31
LD WORK
LD WORK+2
```

TABLE LOOK-UP INSTRUCTION PROGRAMMING EXAMPLE

The table look-up (TLU) operand specifies the first LCT location of a pair of consecutive LCT work locations. These two locations contain a 12-bit RAM address pointer to the beginning of the TLU data table.

```
LCT Work 1   | 0 0 0 0  X X X X |  XXXX = Most Significant
                                         4 Bits

LCT Work 2   | Y Y Y Y  Y Y Y Y |
             \________ ________/
                      V
               8 Least Significant Bits
```

The 12-bit address must be within the channel program portion of the RAM X'200' to X'DEF'. If the address is not in this range, the results are unspecified. It is the responsibility of the main memory program to load the correct address of the TLU data table into the corresponding LCT locations specified by the TLU instruction. This can be done by either a block write or an output LCT byte instruction to each of the LCT WORK locations.

Having resolved the location of the TLU data table, the TLU now adds the contents of the MLCP R-register previously loaded by the channel program. The resulting address now points to a specific byte in the TLU data table.

Bit 0 of the specific TLU data table byte is tested to determine if action requested is a data translation (bit 0 = 0) or an indexed branch (bit 0 = 1).

1. Data Translation (Bit 0 = 0)

   The contents of the specific TLU table byte is loaded into the R-register overwriting the previous value. The control returns to the instruction immediately following the TLU.

   The maximum table sizes for data translation would be 2^8 (256); however only a 7-bit translation is possible because of the use of bit 0 as an indicator. Refer to the sample program shown in Figure 4-1.[2]

2. Indexed Branch (Bit 0 = 1)

   The contents of the specific TLU data table byte is not loaded into the MLCP R-register. The R-register retains the intial contents through this mode of the TLU.

   The address of the indexed branch is generated as follows:

   a. The channel program sequence counter currently pointing to the LCT displacement byte of the TLU instruction is incremented by 3 bytes. This spaces the sequence counter over the 2-byte translation branch following the TLU. This allows a minimum of one branch instruction for processing translated characters.

   b. The specific TLU data table byte, bits 1-7, are multiplied by a factor of 2 because a short displacement branch requires 2 bytes. This allows a table of branches to direct action to the proper routines. The result is then added to the sequence counter causing it to point to a specific branch or macro which is coded inline and after the TLU instruction. The processing continues from this point under control of the channel program.

   The TLU data table can contain a mixture of translation data and indexed branch data. The following example illustrates how the indexed branch option can be used to recognize escape characters and bad parity while translating to other values including stripping the parity bit.

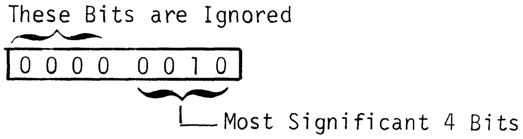
[2] This program is a sample of one of several techniques that can be used for table look-up. Note that the MORG techniques used in the sample program can be used *only* if the TLU instruction starts on an even boundary.

The TLU instruction makes it very easy for the channel program to convert from or to the native ASCII code to any other 7-bit or less code. Thus the MLCP can be fluent in any language without affecting system performance.

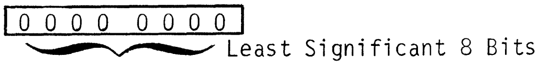
Example 1:
Translate lowercase c X'63' to uppercase C X'43' (Refer to Figure 4-1).

Instruction
TLU 23-convert lowercase ASCII to uppercase ASCII

LCT 23



LCT 24



R-register:

0 1 1 0 0 0 1 1	Lowercase ASCII (c)

Explanation
The TLU table base in LCT 23, 24 (X'0200') is added to the contents of the R-register (X'63') which generated a TLU data table address 'of X'0263'.
The contents of location X'0263' is X'43' and since the value X'43' has bit 0 = 0, a translation is required. The value X'43' is loaded into the R-register and control returns to X'0304' the instruction immediately following the TLU.
The original value of X'63' has been translated to X'43'. The channel program then transfers the X'43' to main memory, branches if last character is false to wait for next character to come in.

Example 2:
Indexed branch on ETX (Refer to Figure 4-1).

LCT 23, 24 Equal X'200' (as in Example 1)

R-register 0 0 0 0 0 0 1 1

Explanation
The TLU table base in LCT 23, 24 (X'200') is added to the contents of the R-Register (X'03') which generates a TLU data table address of X'203'.
The contents of this location is X'8B' and bit 0 = 1. The TLU switches into indexed branch mode and the CCP takes the following action.

a. The R-register is preserved and remains at X'03'.
b. The address of the TLU instruction itself is already in the P-counter (X'303') and points to the LCT displacement byte of the TLU instruction being executed. The value 3 is added to the P-counter making P=X'306'. This constant of 3 is added to allow a branch out to process a translation as in Example 1.

c. The value of bits 1-7 (X‘0B’) of the data extracted from the TLU table is then multiplied by 2. This is done to allow a table of branch instructions which require 2 bytes minimum per branch. 2 (X‘0B’)=X‘16’.

d. The previous value X‘16’ is added to the contents of the P-counter which is X‘306’. The new P-counter value is now the target address of the branch (X‘31C’) and processing continues at this location.

e. The first instruction at X‘31C’ is a long displacement jump to process the ETX1. P-counter +3 +2* (TLU data bits 1-7) = Target.

NOTE: The programmer must generate the branch information contained in the TLU table to match the indexed branch entry points coded after the TLU instruction. In the example, the table value was calculated as follows:

$$\frac{\boxed{\text{target address}} - \boxed{\text{address of second byte of TLU instruction}} -3}{2} = \frac{\text{X'3IC-X'303')-3}}{2} = \text{X'B'}$$

Bit 0 was set to 1 to indicate unconditional branch and the value placed in the table of position X‘03’ = X‘8B’.

Each target area must have an even number of bytes. This is why the MORG instruction is inserted after each target. This instruction will fill with an NOP to make an even number of bytes. The programmer can also directly insert an NOP. If changes were made in the target area, the target area indexes affected have to be recalculated.

The DATA statement can be used to program the target area indexes to adjust automatically to any changes. The target index used in this example could have been calculated by the assembler if the following DATA statement had been substituted for the (X‘8C’) at location X‘203’.

DATA(TARG4-TLU1-X‘04’)/X‘02’+X‘80’

The assembler would evaluate this expression as follows:

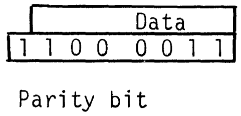
$$\frac{\text{(X‘31C’-X‘302’-X‘04’)}}{2} + \text{X‘80’} = \text{X‘8B’}$$

NOTE: Refer to the DATA statement for rules governing internal value expressions.

Example 3:
Branch on odd parity

Same as Example 2, unconditional branch, except all cases of odd parity in the table have been replaced by X‘82’ which causes a branch to the TARG1 for bad parity at X‘30A’. The bad parity condition can then be programmed as desired.

Example 4: Strip parity bit
R-register = X‘C3’



This value is added to the table base of X‘200’ for a result of X‘23C’.

The contents of X'2C3' = X'43' bit 0 = 0, so a translation is required, the X'43' is loaded into the R-Register, and processing continues at location X'304'. The parity bit has, in effect, been stripped off.

NOTE: The previous examples illustrate the potential of the TLU instruction for normal synchronous data received processing. There are many other uses including user status evaluation, error code processing, decoding action codes passed by the main memory program, etc.

```
     ::TLU TABLE FOR SAMPLE PROGRAM OF TLU USAGE
     ::
     ::          1. UNCONDITIONAL BRANCH ON THE FOLLOWING CONDITIONS
     ::                  A. ODD PARITY IS PARITY ERROR =X'82'
     ::                  B. SOH =X'81' EVEN PARITY START OF HEADER
     ::                  C. STX =X'82' EVEN PARITY START OF TEXT
     ::                  D. ETX =X'03' EVEN PARITY END OF TEXT
     ::                  E. EOT =X'84' EVEN PARITY END OF TRANSMISSION
     ::                  F. SYN =X'96' EVEN PARITY SYNC CHARACTER
     ::
     ::          2. TRANSLATE LOWER CASE 60-7F TO UPPER CASE 40-5F
     ::
     ::          3. STRIP PARITY BIT
     ::
     ::TLU TABLE FOLLOWS
     ::CHARACTERS 00-0F
 200              DATA      X'00',X'82',X'82',X'8B',X'82',X'05',X'06',X'82',X'82',X'09',X'0A', X'82',X'0C',X'82',X'82',X'0F'
     ::CHARACTERS 10-1F
 210              DATA      X'82',X'11',X'12',X'82',X'14',X'82',X'82',X'17',X'18',X'82',X'82', X'1B',X'82',X'1D',X'1E',X'82'
     ::CHARACTERS 20-2F
 220              DATA      X'82',X'21',X'22',X'82',X'24',X'82',X'82',X'27',X'28',X'82',X'82', X'2B',X'82',X'2D',X'2E',X'82'
     ::CHARACTERS 30-3F
 230              DATA      X'30',X'82',X'82',X'33',X'82',X'35',X'36',X'82',X'82',X'39',X'3A', X'82',X'3C',X'82',X'82',X'3F'
     ::CHARACTERS 40-4F
 240              DATA      X'82',X'41',X'42',X'82',X'44',X'82',X'82',X'47',X'48',X'82',X'82', X'4B',X'82',X'4D',X'4E',X'82'
     ::CHARACTERS 50-5F
 250              DATA      X'50',X'82',X'82',X'53',X'82',X'55',X'56',X'82',X'82',X'59',X'5A', X'82',X'5C',X'82',X'82',X'5F'
     ::CHARACTERS 60-6F
 260              DATA      X'40',X'82',X'82',X'43',X'82',X'45',X'46',X'82',X'82',X'49',X'4A', X'82',X'4C',X'82',X'82',X'4F'
     ::CHARACTERS 70-7F
 270              DATA      X'82',X'51',X'52',X'82',X'54',X'82',X'82',X'57',X'58',X'82',X'82', X'5B',X'82',X'5D',X'5E',X'82'
     ::CHARACTERS 80-8F
 280              DATA      X'82',X'86',X'89',X'82',X'8D',X'82',X'82',X'07',X'08',X'82',X'82', X'0B',X'82',X'0D',X'0E',X'82'
     ::CHARACTERS 90-9F
 290              DATA      X'10',X'82',X'82',X'13',X'82',X'15',X'90',X'82',X'82',X'19',X'1A', X'82',X'1C',X'82',X'82',X'1F'
     ::CHARACTERS A0-AF
 2A0              DATA      X'20',X'82',X'82',X'23',X'82',X'25',X'26',X'82',X'82',X'29',X'2A', X'82',X'2C',X'82',X'82',X'2F'
     ::CHARACTERS B0-BF
 2B0              DATA      X'82',X'31',X'32',X'82',X'34',X'82',X'82',X'37',X'38',X'82',X'82', X'3B',X'82',X'3D',X'3E',X'82'
     ::CHARACTERS C0-CF
 2C0              DATA      X'40',X'82',X'82',X'43',X'82',X'45',X'46',X'82',X'82',X'49',X'4A', X'82',X'4C',X'82',X'82',X'4F'
     ::CHARACTERS D0-DF
 2D0              DATA      X'82',X'51',X'52',X'82',X'54',X'82',X'82',X'57',X'58',X'82',X'82', X'5B',X'82',X'5D',X'5E',X'82'
     ::CHARACTERS E0-EF
 2E0              DATA      X'82',X'41',X'42',X'82',X'44',X'82',X'82',X'47',X'48',X'82',X'82', X'4B',X'82',X'4D',X'4E',X'82'
     ::CHARACTERS F0-FF
 2F0              DATA      X'50',X'82',X'82',X'53',X'82',X'55',X'56',X'82',X'82',X'59',X'5A', X'82',X'5C',X'82',X'82',X'5F'
     :::::::::
     :::::::
     :::::
     ::::
```

RESOLVED ADDRESSES

```
     ::EXAMPLE OF THE USE OF THE TLU INSTRUCTION FOR A RECEIVE CCP
     ::ALL DATA RECEIVED IS EVEN PARITY
                LOC       NEXT      INPUT NEXT CHARACTER ENTRY POINT
300             WAIT                WAIT FOR NEXT CHARACTER TO COME IN
301             RECV      2         CHAR AVAILABLE,CHECK EVEN PARITY
     ::ADDRESS IN LCT 23,24=TLU DATA TABLE,WAS LOADED BY MAIN PROGRAM
                LOC       TLU1      ADDRESS OF TLU INSTRUCTION
302,303         TLU       23        TABLE LOOK-UP
304             ST        ,         TRANSLATE COMES HERE SHIP R TO MAIN MEMORY
305,306         BLCF      NEXT      IF NOT LAST CHARACTER GET NEXT ONE
     ::LAST CHARACTER, GO TO MESSAGE FOR LONG ERROR
307,308,309     JUMP      LONG      3 BYTE JUMP
                MORG                INSURE THAT NEXT ADDRESS IS EVEN DISPLACEMENT
     ::ENTRY FOR BAD PARITY DETECTED
```

Figure 4-1. Sample Table Look-Up Program

```
                  LOC        TARG1       ENTRY POINT
30A               ST         ,           STORE IT AS IS
30B,30C           LD         25          GET PARITY ERROR COUNT
30D               DEC                    COUNT 2S COMPLEMENT
30E,30F           ST         25          PUT PARITY COUNT BACK
310,311           B          NEXT        GO TO NEXT CHARACTER
                  MORG                   INSURE EVEN DISPLACEMENT
     *ENTRY FOR SOH =X'81'
                  LOC        TARG2       ENTRY POINT
312,313           AND        =X'7F'      STRIP PARITY BIT
314,315,316       JUMP       SOH1        GO TO PROCESS SOH
                  MORG                   INSURE EVEN ENTRY POINT
     *ENTRY FOR STX X'82'
                  LOC        TARG3       ENTRY POINT
318,319           AND        =X'7F'      STRIP PARITY
31A,31B           B          STX1        GO TO PROCESS SIX
                  MORG                   INSURE EVEN DISPLACEMENT
     *ENTRY FOR ETX X'03'
                  LOC        TARG4       ENTRY POINT
31C,31D,31E       JUMP       ETX1        GO TO PROCESS ETX
                  MORG                   INSURE EVEN DISPLACEMENT
     *ENTRY FOR EOT X'84'
                  LOC        TARG 5      ENTRY POINT
320,321           AND        =X'7F'      STRIP PARITY BIT
322,323,324       JUMP       EOT1        GO TO PROCESS END OF TRANSMISSION
                  MORG                   INSURE EVEN DISPLACEMENT
     *ENTRY FOR SYN X'96' EVEN PARITY SYNC
                  LOC        TARG6       ENTRY POINT
326,327           LD         26          GET SYNC COUNT LOCATION
328               DEC                    COUNT TWOS COMPLEMENT
329,32A           ST         26          PUT COUNT BACK
32B,32C           B          NEXT        IGNORE SYNC GET NEXT CHARACTER
                  MORG                   INSURE EVEN DISPLACEMENT
```

Figure 4-1 (cont). Sample Table Look-Up Program

Format 3

Double operand instructions in this format refer to the MLCP R-register and an immediate operand.

All of the double operand instructions usable in this format are valid in either transmit or receive mode.

Format of Macro Call:

macro-name = operand [comments]

macro-name

LD C AND OR XOR

operand

An internal value expression that, when resolved, is an integer value from 0 to 255, inclusive. This integer value is an immediate operand, which is stored directly after the byte that contains the op code. Note that the internal value expression must be preceded by an equals sign (in the macro call).

Generated Code:

```
                 0                 7
                 ┌──────────────────┐
    byte n       │     op code      │
                 ├──────────────────┤
    byte n+1     │immediate operand │    0 ≤ IMO ≤ 255
                 └──────────────────┘
```

Description of Instructions:

LD (Load)—Load the immediate operand into the MLCP-register.

C (Compare) —Compare the immediate operand with the contents of the MLCP R-register. If the comparison is equal, set the MLCP E-indicator to 1; otherwise, reset it to 0.

AND (Logical AND)—Perform a logical AND operation on the immediate operand and the contents of the MLCP R-register. Store the results of this operation in the MLCP R-register.

OR (Inclusive OR)—Perform an inclusive OR operation on the immediate operand and the contents of the MLCP R-register. Store the results of this operation in the MLCP R-register.

XOR (Exclusive OR)—Perform an exclusive OR operation on the immediate operand and the contents of the MLCP R-register. Store the results of this operation in the MLCP R-register.

Examples:

LDΔ = X'02'
LDΔ = STX
LDΔ = VAL+2

NOTE: Format must conform to the macro preprocessor.

Input/Output Instructions

These instructions are used to transfer control, synchronization, transmit fill, status, and character configuration information between the MLCP R-register and the appropriate line registers of a Communications-Pac. The input/output instructions can also be used to transfer data characters between the MLCP and line register 1 of a Communications-Pac; however, input/output instructions do not provide parity checking or generation, cyclic redundancy checking, or receive overrun or transmit underrun checking and notification.[3]

Format of Macro Call:

macro-name operand [comments]

macro-name

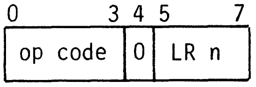
IN,OUT

operand

An internal value expression that, when resolved, is an integer value from 0 to 7, inclusive. This integer is the number of a Communications-Pac line register that will be read or written.

NOTE: The line register must be legitimate for the connected Communications-Pac. If the register does not exist and the instruction is in the CCP, the IN or OUT will be treated as no-op. The IN will cause the R-register to be altered.

Generated Code:



0 ≤ n ≤ 7

Description of Instructions:

IN (Input)—Transfer the contents of Communications-Pac line register n to the MLCP R-register. (Legitimate values for n depend on the specific type of Communications-Pac being used; if an inappropriate line register is indicated by the operand of an IN instruction, a value of 255 (eight 1 bits) will be loaded into the MLCP's R-register.)

If the operand of the IN instruction is 1, the contents of the receive channel's line register 1 will be loaded into the MLCP's R-register; however, parity and/or cyclic redundancy checking cannot be performed nor can a receive overrun be checked for and reported.

[3] Send/receive instructions, described in the following subsection, offer these capabilities.

OUT (Output)—Transfer the contents of the MLCP R-register to line register n of the Communications-Pac. (Legitimate values for n depend on the specific type of Communications-Pac being used; if an inappropriate line register is indicated by the operand of an OUT instruction, the effect is the same as a NOP (No Operation) instruction.)

If the operand of the OUT instruction is 1, the contents of the MLCP's R-register will be transferred to the transmit channel's line register 1; however, parity generation and/or cyclic redundancy checking cannot be performed nor can a transmit underrun be checked for and reported.

Examples:

```
IN 5
IN DSS
OUT 2
OUT SIG
```

Send/Receive Instructions

These instructions are used to transfer *data* between the MLCP R-register and line register 1 of a Communications-Pac. Other information is transferred between the R-register and Communications-Pac line registers by means of input/output instructions, which are described in the preceding subsection.

Format of Macro Call:

macro-name operand [comments]

macro-name

SEND RECV

operand

An internal value expression that, when resolved, is an integer value from 0 to 3, inclusive. These integers have the following significance:

0 – During the data transfer, do not apply the parity or cyclic redundancy check characteristics indicated in LCT byte 2 for a receive channel or in LCT byte 34 for a transmit channel.

1 – During the data transfer, apply the cyclic redundancy check characteristics indicated in bits 5 and 6 of LCT byte 2 for a receive channel or of LCT byte 34 for a transmit channel.

2 – During the data transfer, apply the parity characteristics indicated in bit 3 of LCT byte 2 for a receive channel or of LCT byte 34 for a transmit channel.

3 – During the data transfer, apply the parity and cyclic redundancy check characteristics indicated in bit 3 and bits 5 and 6, respectively, of LCT byte 2 for a receive channel or of LCT byte 34 for a transmit channel.

Generated Code:

```
0          3 4 5 6 7
┌──────────┬───┬───┐
│ op code  │0 0│IVE│     0 ≤ IVE ≤ 3
└──────────┴───┴───┘
```

Description of Instructions:

SEND (Send)—Transfer the data character in the MLCP R-register to the transmit channel's line register 1 of the Communications-Pac. Apply the parity and/or cyclic redundancy check characteristics indicated by the operand.

If parity is to be generated, the parity bit (the leftmost bit of the defined character length) must be specified as a zero (when the data character is transferred from the R-register to the transmit channel's line register 1); the MLCP generates the correct parity during the transfer to line register 1.

If both parity generation and a cyclic redundancy check are performed, correct parity is generated by the MLCP *before* the cyclic redundancy check is performed; thus the cyclic redundancy check is performed on the data character *including* generated parity. (Note that parity generation must not be performed on the actual cyclic redundancy check characters transmitted at the end of a block.)

If, as a SEND instruction is executed, firmware detects a transmit underrun, bit 2 (data service error) of LCT byte 48 (LCT status byte 1) is set to 1.

RECV (Receive)–Transfer, to the MLCP R-register, the data character in the receive channel's line register 1 of the Communications-Pac. Apply the parity and/or cyclic redundancy check characteristics indicated by the operand.

If parity is to be checked, the parity bit (the leftmost bit of the defined character length) may be either 0 or 1 as the data character arrives in the R-register. The parity check includes all bits of the defined character length. If firmware detects a parity error, bit 1 (data check error) of LCT byte 17 (LCT status byte 2) is set to 1.

If both a parity check and a cyclic redundancy check are performed, the cyclic redundancy check is performed on the entire data character, including the parity bit. (Note that the actual cyclic redundancy check characters received at the end of a block normally do not include parity; therefore, a parity check should not be performed on these characters.)

If, as an RECV instruction is executed, firmware detects a receive overrun, bit 2 (data service error) of LCT byte 16 (LCT status byte 1) is set to 1.

Examples:

```
SEND  0           No parity, no block check
SEND  PAR         Parity
SEND  PAR+CRC     Parity and block check

REV  0
RECV  PAR
RECV  PAR+CRC
```

Generic Instructions

Ten generic instructions are available to the CCP.

Format of Macro Call:

macro-name [comments]

macro-name

NOP WAIT GNB SFS CCH DEC RET SR INTR INZ

Generated Code:

```
0           7
┌────────────┐
│  op code   │
└────────────┘
```

Description of Instructions:

NOP (No Operation)–No operation is performed. (This instruction must be used as the last instruction of each CCP to guarantee one complete final word of input to the Assembler.)

WAIT (Wait)–Suspend execution of this CCP. Store, in the appropriate LCT, the contents of the MLCP P-register, R-register, and program indicators. (These contents will be restored when this CCP is again serviced by the MLCP.)

GNB (Get Next Block)–Transfer the contents of the LCT status bytes to the status field of the "active" CCB; at the same time, set the CCB status complete bit to 1, interrupt the CPU if the "I" bit (bit 0 of CCB control byte) is set, and reset to 0 the entire CCB control field; next, move the "active" CCB pointer to the following CCB in the list for this channel. (The next format 1 LD (Load) or ST (Store) instruction will refer to this "new" CCB.)

SFS (Search for Synchronization)—The appropriate receive channel of a Synchronous Line Communications-Pac is placed in "search for synchronization character" mode.

Normally, this instruction should be followed by a WAIT (Wait) instruction because a channel request interrupt (from the channel serviced by this CCP) will not be generated until a synchronization character is detected.

CCH (Calculate Block Check)—Perform a cyclic redundancy check on the contents of the MLCP R-register; update the cyclic redundancy check residue in the appropriate LCT bytes.

DEC (Decrement R-Register)—Decrease by 1 the contents of the MLCP R-register.

RET (Return From Subroutine)—Restore to the MLCP P-register the 12-bit CCP address stored in firmware-reserved LCT bytes when a BS (Branch to Subroutine) instruction was executed.

SR (Shift R-Register Right)—Shift all bits in the MLCP R-register one bit position to the right. The rightmost bit is lost and a 0 is added at the leftmost bit position in the R-register.

One use of this instruction is to right justify, in LCT byte 3 or 35, the six most significant bits of 12-bit cyclic redundancy check residue.

INTR (Interrupt CPU) – Cause the MCLP to interrupt the CPU unconditionally on behalf of the channel that issued the INTR instruction unless the interrupt level zero. This instruction is independent of any block boundary or other CCP instruction. CCP execution will proceed following execution of this instruction. (Refer to Appendix A for further details on this instruction.)

INZ (CCP-Inititated Soft Inititalize) – Cause the MLCP to perform a soft initialize. This causes operation to cease on all MLCP lines. This instruction is used primarily in a debug environment. When used in conjunction with a branch to subroutine instruction (BS) and the INTR instruction, a breakpoint can be created. The BS instruction loads LCT byte 18/50 and LCT byte 19/51 with a 12-bit return address which indicates the location where the breakpoint occurred.
The soft initialize function itself is described in Section 2.

Examples:
The generic type instructions do not have operands and therefore are coded as is, e.g.,

```
NOP
GNB
INTR
```

SUMMARY TABLES

Tables 4-2 and 4-3 respectively illustrate the general format of macro calls for CCP generation control statements and for CCP executable instructions.

Tables 4-4 and 4-5 are bit maps of CCP executable instruction op codes. Table 4-4 indicates those instructions that can be used in a CCP servicing a *receive* channel; Table 4-5 indicates those instructions that can be used in a CCP servicing a *transmit* channel.

Tables 4-6 through 4-10 collectively contain timings for all CCP executable instructions; the tables are arranged by type of instruction.

TABLE 4-2. FORMAT OF MACRO CALLS FOR CCP GENERATION CONTROL STATEMENTS

Instruction	Macro Call Format	Comments
CCP Generation Control Statements	LOC operand [comments]	Operand is a user-supplied label.
	ORG operand [comments]	Operand is a decimal or hexadecimal constant. If decimal: $0 \leqslant \text{operand} \leqslant 3583$ If hexadecimal: $\text{X'0'} \leqslant \text{operand} \leqslant \text{X'0DFF'}$
	MORG [comments]	A modulo 2 value is assigned to the Macro Preprocessor's byte allocation counter.
	DATA operand [,operand] [comments]	One to 35 operands are possible. Each operand is an internal value expression. $0 \leqslant \text{IVE} \leqslant 255$

TABLE 4-3. FORMAT OF MACRO CALLS FOR CCP EXECUTABLE INSTRUCTIONS

Instruction	Macro Call Format	Comments
Branch Instructions Short Displacement	B operand [comments] BS operand [comments] BET operand [comments] BEF operand [comments] BZT operand [comments] BZF operand [comments] BLCT operand [comments] BLCF operand [comments] BLBT operand [comments] BLBF operand [comments] BART operand [comments] BARF operand [comments] BVBT operand [comments] BVBF operand [comments]	Operand is an internal value expression. $-128 \leqslant \text{IVE} \leqslant 127$
Long Displacement Branch	JUMP operand [comments]	Operand is an internal value expression. $-4096 \leqslant \text{IVE} \leqslant 4095$
Double Operand Instructions–Format 1	LD, [comments] ST, [comments]	Refers to MLCP R-register and CDB.
Double Operand Instructions–Format 2	LD operand [comments] ST operand [comments] C operand [comments] AND operand [comments] OR operand [comments] XOR operand [comments]	Refers to MLCP R-register and LCT byte n. Operand is an internal value expression that identifies LCT byte n. $0 \leqslant \text{IVE} \leqslant 63$
	TLU operand [comments]	$23 \leqslant \text{IVE} \leqslant 30$ $55 \leqslant \text{IVE} \leqslant 62$

TABLE 4-3 (cont). FORMAT OF MACRO CALLS FOR CCP EXECUTABLE INSTRUCTIONS

Instruction	Macro Call Format	Comments

Double Operand Instructions–Format 3	LD =operand [comments]
	C =operand [comments]
	AND =operand [comments]
	OR =operand [comments]
	XOR =operand [comments]
		Refers to MLCP R-register and an immediate operand.
		Operand is an internal value expression.
		0 ⩽ IVE ⩽ 255

Input/Output Instructions	IN operand [comments] OUT operand [comments]	Operand is an internal value expression. 0 ⩽ IVE ⩽ 7
Send/Receive Instructions	SEND operand [comments] RECV operand [comments]	Operand is an internal value expression. 0⩽IVE⩽3
Generic Instructions	NOP [comments] WAIT [comments] GNB [comments] SFS [comments] CCH [comments] DEC [comments] RET [comments] SR [comments] INTR [comments] INZ [comments]	

R, C_ = N, A2, B, B

TABLE 4-4. BIT MAP OF CCP EXECUTABLE INSTRUCTIONS' OP CODE WORDS *RECEIVE* MODE

Instruction Type/Format	Bits 0-3	Bits 4-7															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Generic	0	NOP	WAIT	GNB	SFS	CCH	DEC	RET	SR	INTR	INZ						
Double Operand – Reference to CDB	1		ST,CDB														
Input/Output IN[a]	2	LR0	LR1	LR2	LR3	LR4	LR5	LR6	LR7								
Input/Output OUT[a]	3	LR0	LR1	LR2	LR3	LR4	LR5	LR6	LR7								
Reserved	4																
Double Operand – Reference to LCT Byte n	5	LD,LCT	ST	C	AND	OR	XOR	TLU									
Reserved for transmit channel[c]	6																
Reserved	7, 8																
Double Operand-Reference to IMO	9	LD		C	AND	OR	XOR										
Send/Receive–RECV	A	RECV 0 No Parity or CRC[b]	RECV 1 CRC[b]	RECV 2 Parity[b]	RECV3 Parity and CRC[b]												
Reserved	B,C,D																
Branch–Branch True	E	B	BET	BZT	BLCT	BLBT	BART	JUMP	BVBT								
Branch–Branch False	F	BS	BEF	BZF	BLCF	BLBF	BART		BVBF								

[a] Existence of line register is dependent on type of Communications-Pac being used. If a nonexistent line register is used in the IN or OUT statement, the statement will be treated as a no-op.

[b] Indicates whether the parity check and/or cyclic redundancy check information in LCT byte 2 applies to the data character being transferred to the MLCP's R-register from the receive channel's line register 1 in the Communications-Pac.

[c] Do not use.

TABLE 4-5. BIT MAP OF CCP EXECUTABLE INSTRUCTIONS' OP CODE WORDS – *TRANSMIT* MODE

Instruction Type/Format	Bits 0-3	Bits 4-7															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Generic	0	NOP	WAIT	GNB	SFS	CCH	DEC	RET	SR	INTR	INZ						
Double Operand–Reference to CDB	1	LD CDB															
Input/Output – IN[a]	2	LR0	LR1	LR2	LR3	LR4	LR5[a]	LR6	LR7								
Input/Output – OUT[a]	3	LR0	LR1	LR2	LR3	LR4	LR5	LR6	LR7								
Reserved	4																
Double Operand–Reference to LCT Byte n	5	LD	ST	C	AND	OR	XOR	TLU									
Send/Receive–SEND	6	SEND0 No Parity or CRC[b]	SEND1 CRC[b]	SEND2 Parity[b]	SEND3 Parity and CRC[b]												
Reserved	7, 8																
Double Operand–Reference to IMO	9	LD		C	AND	OR	XOR										
Reserved for Receive Channel[c]																	
Reserved	B,C,D																
Branch–Branch True	E	B	BET	BZT	BLCT	BLBT	BART	JUMP	BVBT								
Branch–Branch False	F	BS	BEF	BZF	BLCF	BLBF	BARF		BVBF								

[a] Existence of a line register is dependent on type of Communications-Pac being used. If a nonexistent line register is used in the IN or OUT statement, the statement will be treated as a no-op.

[b] Indicates whether the parity generation and/or cyclic redundancy check information LCT byte 34 applies to the data character being transferred from the MLCP's R-register to the transmit channel's line register 1 of the Communications-Pac. The parity generation is done before the CRC operation (i.e., the parity bit is included in the CRC calculation).

[c] Do not use.

TABLE 4-6. TIMINGS FOR BRANCH INSTRUCTIONS

Instruction	Time No Branch	Branch
B	N/A	3.6 μs
BS	N/A	7.0 μs
BET	2.1 μs	4.1 μs
BEF	2.1 μs	4.1 μs
BZT	2.1 μs	4.1 μs
BZF	2.1 μs	4.1 μs
BLCT	2.1 μs	4.1 μs
BLCF	2.1 μs	4.1 μs
BLBT	2.1 μs	4.1 μs
BLBF	2.1 μs	4.1 μs
BART	3.3 μs	5.3 μs
BARF	3.3 μs	5.3 μs
BVBT	5.7 μs	7.7 μs
BVBF	4.0 μs	6.0 μs
JUMP	N/A	4.8 μs

TABLE 4-7. TIMINGS FOR DOUBLE OPERAND INSTRUCTIONS

	Time		
Instruction	Format 1 R & CDB	Format 2 R & LCT n	Format 3 R & IMO
LD	22.6 μs	5.3 μs	3.2 μs
ST	21.7 μs	5.1 μs	N/A
C	N/A	5.8 μs	3.9 μs
AND	N/A	5.3 μs	3.2 μs
OR	N/A	5.3 μs	3.2 μs
XOR	N/A	5.6 μs	3.4 μs
TLU (Translate)	N/A	8.5 μs	N/A
TLU (Branch)	N/A	9.1 μs	N/A

TABLE 4-8. TIMINGS FOR INPUT/OUTPUT INSTRUCTIONS

Instruction	Time
IN	3.6 μs
OUT	3.1 μs

TABLE 4-9. TIMINGS FOR SEND/RECEIVE INSTRUCTIONS

Instruction	Time Without CRC	With CRC
SEND	6.3 μs	13.4 μs
RECV	7.2 μs	14.2 μs

TABLE 4-10. TIMINGS FOR GENERIC INSTRUCTIONS

Instruction	Time
NOP	1.7 μs
WAIT	15.0 μs[a]
GNB	n μs[b]
SFS	5.6 μs
CCH	8.4 μs
DEC	2.0 μs
RET	4.9 μs
SR	2.4 μs
INTR	11.0 μs
INZ	200.00 μs

[a] This figure includes the time necessary to perform a context swap from the currently running CCP to the next CCP (approximately 8 μs).

[b] GNB–receive mode and previously active CCB causes interrupt to central processor: 32.2 μs
GNB–receive mode without interrupt: 25.2 μs
GNB–transmit mode with interrupt: 28.8 μs
GNB–transmit mode without interrupt: 21.8 μs

SECTION 5
LINE CONTROL TABLES

Line control tables (LCT's) are line-specific data structures in MLCP RAM. Space for eight LCT's of 64 bytes each exists at the low end of RAM (see Figure 1-2). The LCT for line 0 extends from RAM byte 0 to byte 63; the remaining LCT's are arranged by ascending order of line number up to the LCT for line 7, which extends from RAM byte 448 to byte 511.

Each LCT is dedicated to *one* communications line. The LCT stores line-specific configuration, setup, status, and control information used during communications processing. The LCT is of key importance to MLCP operations because it is visible to, and alterable by, the main memory program (through input/output instructions), the CCP, and MLCP firmware.

An LCT's 64 consecutive bytes are divided so that the first 32 (LCT bytes 0 through 31) are dedicated to the receive (even-numbered) channel of the line and the second 32 (LCT bytes 32 through 63) are dedicated to the line's transmit (odd-numbered) channel. Each of the 32 bytes of the "receive" side of the LCT has a direct counterpart on the "transmit" side. Throughout this section, descriptions of LCT bytes are arranged so that one narrative pertains to both an LCT "receive" byte and to its "transmit" counterpart. A slash is used to join the two equivalent bytes—e.g., LCT byte 2/34.

The general function of each LCT byte is listed in Table 5-1. Notice that a number of LCT bytes are reserved for firmware use. Notice also that only certain of the remaining LCT bytes may be modified by a CCP.

TABLE 5-1. SUMMARY OF LINE CONTROL TABLE BYTES

LCT Byte Number	Contents
0	Receive Firmware Use Only
1	Receive Firmware Use Only
2	Receive Character Configuration
3[a]	Receive Cyclic Redundancy Check Residue – Byte 1
4[a]	Receive Cyclic Redundancy Check Residue – Byte 2
5	Receive Firmware Use Only
6	Receive CCP Pointer – Four Most Significant Bits
7	Receive CCP Pointer – Eight Least Significant Bits
8[a]	Receive Change Control for Data Set and Communications-Pac Status
9	Receive Firmware Use Only
10	Receive Firmware Use Only
11	Receive Firmware Use Only
12	Receive Return Channel Number – Eight Most Significant Bits
13	Receive Return Channel Number – Two Least Significant Bits and Interrupt Level
14	Receive Data Set and Communications-Pac Status
15[a]	Receive Mask for Data Set and Communications-Pac Status
16[a]	Receive LCT Status – Byte 1
17[a]	Receive LCT Status – Byte 2
18	Receive Firmware Use Only
19	Receive Firmware Use Only
20[a]	Receive/Transmit Data Set and Communications-Pac Control

TABLE 5-1 (CONT). SUMMARY OF LINE CONTROL TABLE BYTES

LCT Byte Number	Contents
21	Receive Firmware Use Only
22	Receive Firmware Use Only
23[a]	Programming Work Area
24[a]	Programming Work Area
25[a]	Programming Work Area
26[a]	Programming Work Area
27[a]	Programming Work Area
28[a]	Programming Work Area
29[a]	Programming Work Area
30[a]	Programming Work Area
31[a]	Programming Work Area
32	Transmit Firmware Use Only
33	Transmit Firmware Use Only
34	Transmit Character Configuration
35[a]	Transmit Cyclic Redundancy Check Residue – Byte 1
36[a]	Transmit Cyclic Redundancy Check Residue – Byte 2
37	Transmit Firmware Use Only
38	Transmit CCP Pointer – Four Most Significant Bits
39	Transmit CCP Pointer – Eight Least Significant Bits
40[a]	Transmit Change Control for Data Set and Communications-Pac Status
41	Transmit Firmware Use Only
42	Transmit Firmware Use Only
43	Transmit Firmware Use Only
44	Transmit Return Channel Number – Eight Most Significant Bits
45	Transmit Return Channel Number – Two Least Significant Bits and Interrupt Level
46	Transmit Data Set and Communications-Pac Status
47[a]	Transmit Mask for Data Set and Communications-Pac Status
48[a]	Transmit LCT Status – Byte 1
49[a]	Transmit LCT Status – Byte 2
50	Transmit Firmware Use Only
51	Transmit Firmware Use Only
52[a]	Programming Work Area
53	Transmit Firmware Use Only
54	Transmit Firmware Use Only
55	Reserved for Input LCT Byte Address (if used)
56[a]	Programming Work Area
57[a]	Programming Work Area
58[a]	Programming Work Area
59[a]	Programming Work Area
60[a]	Programming Work Area
61[a]	Programming Work Area
62[a]	Programming Work Area
63[a]	Programming Work Area

[a] This byte may be modified by the CCP.

Because each LCT is independent and pertains to only one communications line, it cannot be used for operations pertaining to any other line. If, for example, a given CCP is exclusively associated with line 0, it can never gain access to the LCT for line 1. (Even if a CCP services several lines, it has a separate, line-specific relationship with each LCT that is dedicated to one of the lines it services.)

An LCT must be properly set up as one of the preliminary steps before data transfers can begin over the associated communications line. The LCT must first be initialized to 0 – by the MLCP Loader or by the use of an IO (Output MLCP Control) or IO (Output Channel Control) instruction in the main memory program. Next, certain LCT bytes must be written with control information from main memory. (Normally, these are LCT bytes 2/34, 6/38, 7/39, 8/40, 12/44, 13/45, 15/47, 20/52, and at least one byte in the LCT programming work area; this last byte is normally used to load line register 4 of the associated Communications-Pac.)

The LCT bytes can be initially written by use of the Honeywell-supplied MLCP Loader or by a "block mode write" from the main memory program; alternatively, LCT bytes can be written by IO (Output LCT Byte) and IO (Output Interrupt Control) instructions from the main memory program. The values to be written into the LCT bytes can be established by the use of DC (Define Constants) Assembler control statements in the main memory program.

The remainder of this section provides a detailed description of those LCT bytes that have user-visible significance. In addition, a detailed layout of all LCT bytes appears at the end of this section.

LCT BYTE 2/34 – CHARACTER CONFIGURATION

Description:

This byte describes the data characters to be processed. It records character length, character parity, number of stop bits, and type of cyclic redundancy check polynomial. The contents of this byte are used as input when line register 6 of a Communications-Pac is loaded.

NOTE: Not all information in this byte is meaningful to the Communications-Pac; see the descriptions of line register 6 in Appendixes C and D.

Programming Considerations:

This byte must be loaded and maintained from the main memory program. This byte should not be modified by the CCP. Note that bits 2 and 7 of this byte must always be 0.

Byte Layout:

	0	1	2	3	4	5	6	7
2/34	CHARACTER LENGTH		0	CHARACTER PARITY	ASYNCHRONOUS STOP BITS SYNCHRONOUS 0	CYCLIC REDUNDANCY CHECK POLYNOMIAL		0

Bits 0 and 1 – character length

00 - 5 bits – 5-bit data only, parity not allowed
01 - 6 bits – 6-bit data or 5-bit data with parity
10 - 7 bits – 7-bit data or 6-bit data with parity
11 - 8 bits – 8-bit data or 7-bit data with parity

During processing, the character length specified in this byte must be the same as the character length specified in line register 6. Parity, if used, is stored in the leftmost bit of the character length defined here. The unused bits must be zero for the parity generation to be correct.

Bit 3–character parity

0 - Odd parity.

1 - Even parity.

All character parity checking and generation is based on the setting of this bit. MLCP firmware performs parity *checking* for characters loaded into the MLCP R-register by an RECV (Receive) instruction whose operand value is 2 or 3. MLCP firmware performs parity *generation* for characters loaded into a transmit channel's line register 1 of a Communications-Pac by a SEND (Send) instruction whose operand value is 2 or 3.

Bit 4 – number of stop bits per data character

(Asynchronous Line Communications-Pacs only)

0 - 1 stop bit per 5-, 6-, 7-, or 8-bit data character.

1 - 1.5 stop bits per 5-bit data character.

1 - 2 stop bits per 6-, 7-, or 8-bit data character.

Bits 5 and 6 – cyclic redundancy check polynomial

00 - $x^{16} + x^{15} + x^2 + 1$	Used for 8-bit or 7-bit with parity; IBM BSC CRC-16 code
01 - $x^{16} + x^{12} + x^5 + 1$	Used for 8-bit or 7-bit with parity; CCITT HDLC and Transparent Mode ASCII code
10 - $x^{12} + x^{11} + x^3 + x^2 + x + 1$	Used for 6-bit or 5-bit with parity; CRC-12 Transcode
11 - $x^8 + 1$	Used for 7-bit with partiy; LRC – Basic Mode ASCII code

The settings of these bits govern which one of four polynomials will be used when cyclic redundancy checking is performed by MLCP hardware. Cyclic redundancy checking is performed by execution of the CCH (Calculate Block Check) instruction in the CCP and by the execution of SEND (Send) and RECV (Receive) instructions whose operand values are 1 or 3.

More information regarding cyclic redundancy checking appears in Section 6.

LCT BYTES 3/35 AND 4/36–CYCLIC REDUNDANCY CHECK RESIDUE

Description:

These bytes store residue following a cyclic redundancy check. More information regarding cyclic redundancy checking appears in Section 6.

Programming Considerations:

The cyclic redundancy check polynomial is specified in bits 5 and 6 of LCT byte 2/34. Cyclic redundancy checking is performed by MLCP hardware during execution of the CCH (Calculate Block Check) instruction and during execution of SEND (Send) and RECV (Receive) instructions whose operand values are 1 or 3. The resultant cyclic redundancy check residue in LCT bytes 3/35 and 4/36 may be examined from the CCP. These bytes must be initialized by the CCP or the main memory program whenever a block CRC calculation is restarted.

The appropriate initial value for LCT byte 3/35 and 4/36 is all 1s for a CRC polynomial code equal to 01 (HDLC). For the other CRC polynomial codes (CRC 16, LRC, CRC 12), the appropriate initial value is all zeros. The program, typically the CCP, must ensure that these initial values are provided.

Byte Layout:

The format of these bytes is governed by which cyclic redundancy check polynomial is used; see Section 6.

LCT BYTES 6/38 AND 7/39–CCP POINTER

Description:

These bytes store the 12-bit RAM address at which the CCP will begin execution when it is started again. When the MLCP services this CCP, this RAM address will be loaded into the MLCP's P-register (program counter).

Programming Considerations:

The initial starting address of the CCP must be written into these bytes from the main memory program.

During processing, MLCP firmware uses these bytes to store the contents of the P-register whenever the CCP executes a WAIT (Wait) instruction or whenever a firmware pause occurs. When the CCP resumes, firmware restores the contents of these bytes to the P-register, thereby allowing the CCP to begin at its next sequential instruction.

The contents of these bytes may be modified from the main memory program, if you wish to have the CCP resume at an address other than the one stored here. IO (Output LCT Byte) instructions can be used for this purpose. However, the CCP must not be running at the time or the results are unspecified.

When LCT byte 6/38 is written, bits 0 through 3 must always be reset to zero.

These bytes must not be modified by the CCP.

Byte Layout:

	0	1	2	3	4	5	6	7
6/38	0	0	0	0	MOST SIGNIFICANT BITS			
7/39	LEAST SIGNIFICANT BITS							

LCT BYTE 8/40–CHANGE CONTROL FOR DATA SET AND COMMUNICATIONS-PAC STATUS

Description:

This byte may be written from the CCP to control (1) whether MLCP firmware will scan for changes in data set status and Communications-Pac status, recording these changes in LCT byte 14/46 and (2) what action(s) will be taken when a status change is recorded. (A mask in LCT byte 15/47 governs exactly which types of status change will cause LCT byte 14/46 to be updated with the contents of Communications-Pac line register 5.)

Programming Considerdations:

This byte may be initially written from the main memory program or from the CCP. It may subsequently be changed by either program–at points in time appropriate to the communications application.

Whenever this byte is written, bits 4 through 7 must always be reset to zero. If bits 2 and 3 are both set to 1, only bit 2 (terminate the active CCB and interrupt the main memory program) will be acted upon when a data set or Communications-Pac status change is recorded in LCT byte 14/46.

Refer also to the programming guidelines in Appendix A.

Byte Layout:

	0	1	2	3	4	5	6	7
8/40	SCAN CONTROL	SET BIT 3 OF LCT BYTE 17/49 IF CHANGE	INTERRUPT MAIN MEMORY PROGRAM IF CHANGE	START CHANNEL CONTROL PROGRAM IF CHANGE	0	0	0	0

Bit 0–scan control

0 - MLCP firmware will not scan Communications-Pac line register 5 for changes in data set status and Communications Pac status.

1 - MLCP firmware will scan Communications-Pac line register 5 for changes in data set status and Communications-Pac status. Firmware will write the entire contents of line register 5 into LCT byte 14/46 whenever it detects a difference between the contents of a bit position in line register 5 and the contents of the same bit position in LCT byte 14/46 (*provided* there is a 1 in the corresponding bit position of the mask contained in LCT byte 15/47). Next, the action(s) specified in bits 1, 2, and 3 of LCT byte 8/40 will be taken.

Bit 1–set bit 3 of LCT byte 17/49 if change

0 - No action.

1 - When a data set or Communications-Pac status change is recorded in LCT byte 14/46, set to 1 bit 3 of LCT byte 17/49 (LCT status byte 2).

Bit 2–interrupt main memory program if change (refer also to LCT status byte 16/48)

0 - No action.

1 - When a data set or Communications-Pac status change is recorded in LCT byte 14/46, terminate the active CCB and interrupt the main memory program at the interrupt level established for this channel. (Ignore the setting of bit 3 of this byte and set bit 1 of LCT status byte 16/48.)

Bit 3–start CCP if change

0 - No action.

1 - When a data set or Communications-Pac status change is recorded in LCT byte 14/46, start the CCP. (This action will be taken only if bit 2 of this byte is reset to 0.)

LCT BYTES 12/44 AND 13/45–RETURN CHANNEL NUMBER AND INTERRUPT LEVEL

Description:

These bytes may be used to store the 10-bit return channel number and the interrupt level that will be placed on the Megabus whenever the main memory program is interrupted from this channel. (The return channel number is the central processor's channel number.)

Programming Considerations:

The use of these bytes is optional. They are required if the main memory program is to be interrupted from this channel.

The contents of these bytes can be established as part of the block transferred to MLCP RAM by the MLCP Loader or by a "block mode write." Alternatively, the contents of these bytes can be established (or modified) from the main memory program by use of one IO (Output Interrupt Control) instruction.

These bytes must not be written from the CCP.

Byte Layout:

	0	1	2	3	4	5	6	7
12/44	RETURN CHANNEL NUMBER–MOST SIGNIFICANT BITS							
13/45	RETURN CHANNEL NUMBER–LEAST SIGNIFICANT BITS		INTERRUPT LEVEL					

LCT BYTE 14/46–DATA SET AND COMMUNICATIONS-PAC STATUS

Description:

This byte is used to record changes in data set status and Communications-Pac status if bit 0 of LCT byte 8/40 is set to 1 and one or more bits of a mask in LCT byte 15/47 are also set to 1. If these conditions exist, MLCP firmware will write the entire contents of Communications-Pac line register 5 into LCT byte 14/46 whenever it detects a difference between the contents of a bit position in line register 5 and the contents of the same bit position in LCT byte 14/46 (*provided* there is a 1 in the corresponding bit position of the mask contained in LCT byte 15/47). Subsequent action(s) depend on the settings of bits 1, 2, and 3 of LCT byte 8/40.

LCT byte 14/46 is not used by MLCP firmware if bit 0 of LCT byte 8/40 is reset to 0.

Programming Considerations:

The contents of this byte may be examined by the CCP, but the CCP must not modify the contents.

Proper settings of bits 1, 2, and 3 of LCT byte 8/40 permit appropriate action(s) to be taken when a data set or Communications-Pac status change is recorded in LCT byte 14/46.

Additional information on this subject appears in Section 6.

Byte Layout (Typical): See the appendixes for specific Communication-Pacs.

	0	1	2	3	4	5	6	7
	DATA SET STATUS					COMMUNICATIONS-PAC STATUS		
14/46	DATA SET READY	CLEAR TO SEND	CARRIER DETECTOR	RING INDICATOR	ASYNCH. SECONDARY CHL. RECV. - - - - - - - - - SYNCH. RESERVED - - - - - - - - - BROADBAND ADAPTER READY	RESERVED	RECEIVE OVERRUN	ASYNCHRONOUS FRAMING ERROR - - - - - - - - - SYNCHRONOUS TRANSMIT UNDERRUN

Each bit position has only one meaning, regardless of whether the LCT byte relates to a receive channel (LCT byte 14) or to a transmit channel (LCT byte 46). The meanings of bits 0, 1, 2, and 3 are given in appropriate Bell System Technical References (see Preface).

Bit 4–secondary channel receive (Asynchronous Line Communications-Pacs only)
Secondary channel receive is described in appropriate Bell System Technical References (see Preface).

Bit 4–adapter ready (Broadband Communications-Pac)
Transmit
0 - Adapter data buffer full.
1 - Adapter data buffer not full.
Receive
0 - Character in adapter data buffer.
1 - Adapter data buffer empty.

Adapter ready is applicable to Broadband Communications-Pacs that contain a first-in, first-out buffer. In the case of the broadband pacs, this buffer is 64 characters. On receive, characters from the line are put into one end of the buffer and the CCP retrieves characters from the other end, thus allowing much greater timing variation. On transmit, the operation is just the opposite. Refer to the next paragraph also.

On receive, adapter ready = 1 whenever there is one or more characters in the FIFO buffer requiring processing. The adapter ready condition will be made known to the MLCP by the condition of bit 4 of line register 5. When either the BART or BARF instruction is executed, the MLCP firmware inputs line register 5 and branches on the condition of bit 4. Because non-Broadband Communications-Pacs may make other use of this bit, BART and BARF are not meaningful in these cases.

Bit 6—receive overrun (this bit may be on before receiver is activated; see Appendix A).

0 - No receive overrun has occurred.

1 - A receive overrun has occurred. The Communications-Pac was not serviced fast enough by the receive CCP and one or more data characters have been overwritten (and thus lost) in a receive channel's line register 1 of the Communications-Pac.

Bit 7—framing error (Asynchronous Line Communications-Pacs);
transmit underrun (Synchronous Line Communications-Pacs)

For Asynchronous Line Communications-Pacs, the values of bit 7 have the meanings described below.

0 - No framing error has occurred. (See Appendix A.)

1 - A framing error has occurred; the Asynchronous Line Communications-Pac has detected a missing stop bit. (The preceding data bits should be analyzed to ascertain whether a line break has occurred.)

Bit 7 normally indicates a transmit underrun error but on the asynchronous Communications-Pacs it is used to indicate a framing error. The transmit send order picks up this bit and sets data error bit 2 of LCT byte 16/48. Since there is no transmit underrun error for asynchronous adapters, the channel program should reset this bit or output two null characters to LR1 (OUT 1) after bit 7 of LCT byte 20 (line register 2) is turned on. This also clears the error.

For Synchronous Line Communications-Pacs, the values of bit 7 have the meanings described below.

0 - No transmit underrun has occurred.

1 - A transmit underrun has occurred. The Communications-Pac was not serviced fast enough by the transmit CCP and the transmit fill character (usually SYN) in the transmit channel's line register 4 of the Communications-Pac has been sent instead of a data character. Normally this is not a fatal condition for character-oriented procedures but is fatal for bit-oriented procedures, e.g., high-level data link (HDLC) procedure.

LCT BYTE 15/47—MASK FOR DATA SET AND COMMUNICATIONS-PAC STATUS

Description:

If bit 0 of LCT byte 8/40 is set to 1, LCT byte 15/47 is used as a mask. The mask governs which changes in data set status and Communications-Pac status will cause (1) the entire contents of Communications-Pac line register 5 to be written to LCT byte 14/46 and (2) subsequent action(s) to be taken based on the settings of bits 1, 2, and 3 of LCT byte 8/40.

Each of the bit positions in the mask corresponds to the status category indicated by that same bit position in LCT byte 14/46 and in line register 5 of the Communications-Pac. If a bit in the mask is set to 1, the action described in the preceding paragraph will occur whenever an MLCP firmware scan reveals that the contents of that same bit position in line register 5 are not identical to the contents of that bit position in LCT byte 14/46.

Programming Considerations:

This byte may be initially written by the main memory program or the CCP. It may subsequently be modified, as needed, by either program.

Byte Layout:

	0	1	2	3	4	5	6	7
15/47	MASK							

Bits 0 through 7–mask

Each bit position in the mask corresponds to the status category indicated by the same bit position in LCT byte 14/46 and in line register 5 of the Communications-Pac. The mask must contain a 1 in each bit position for which a status discrepancy between LCT byte 14/46 and line register 5 is to cause the action described above.

LCT BYTES 16/48 AND 17/49–LCT STATUS

Description:

These two bytes are used by MLCP firmware to store certain status and error information while a given CCB is active. The information begins to be collected when processing starts relative to the "active" CCB. When processing ends relative to this CCB, the contents of these two bytes are combined with other information and transferred to the status field (bytes 6 and 7) of the CCB.

Programming Considerations:

These two bytes may be examined by the main memory program–through the IO (Input LCT Status) instruction–and by the CCP–through format 2 LD (Load) instructions. The CCP may set and reset bits 5 and 6 of LCT byte 16/48, as appropriate; these bits are shown shaded in the byte layout below.

It is important to note that the LCT status bytes are *dynamic* and continually subject to change by MLCP firmware during processing related to the "active" CCB.

When CCB is terminated, the contents of LCT byte 16/48 (LCT status byte 1) are combined with other information and moved to byte 7 of the CCB. The contents of LCT byte 17/49 (LCT status byte 2) are combined with other information and moved to byte 6 of the CCB. (The complete format of the CCB status field is shown in Section 3.)

Byte Layout:

	0	1	2	3	4	5	6	7
16/48 LCT STATUS BYTE 1	CPU INTERRUPT FROM INTR INSTRUCTION	CPU INTERRUPT FROM DATA SET STATUS CHANGE	DATA SERVICE ERROR	0	CCB SERVICE ERROR	FOR PROGRAMMING USE	FOR PROGRAMMING USE	RESERVED
17/49 LCT STATUS BYTE 2	RESERVED	DATA CHECK ERROR	0	DATA SET OR COMMUNICATIONS-PAC STATUS CHANGE	CORRECTED MEMORY ERROR	0	0	0

The bits that may be modified by the CCP are shown shaded.

LCT Byte 16/48

Bit 0—Interrupt main memory program from INTR instruction in CCP

0 - No action.

1 - Interrupt main memory program from INTR instruction in CCP. Note that the INTR instruction can also be used as an error escape from channel program that has detected an abort condition.

Bit 1—Interrupt main memory program from data set status change (see also Appendix A)

0 - No action.

1 - Data set scan has detected a data set status change and bit 3 of LCT byte 8/40 is on. The main memory program has been interrupted.

NOTE: The Input Next CCB Status instruction issued after the interrupt usually results in a CCB status equal to zero. If this is the case, LCT byte 16/48 can be input by the Input LCT Byte instruction and the type of interrupt can then be determined.

Bit 2—data service error

0 - No data service error has occurred.

1 - A data "timing window" has been missed. On receive, the Communications-Pac has detected a receive overrun (see bit 6 of LCT byte 14/46). On transmit, the Communications-Pac has detected a transmit underrun (see bit 7 of LCT byte 14/46).

Bit 4—CCB service error

0 - No CCB service error has occurred.

1 - On receive, a format 1 ST (Store) instruction has been attempted and there is no "valid" CCB. (MLCP firmware will set this bit to 1 and proceed to the next sequential instruction in the CCP.) Data characters will be lost if they are received by CCP RECV (Receive) instructions while there is no valid CCB. The Communications-Pac will not set the receive overrun bit in line register 5 so long as the CCP continues to execute RECV instructions and accept data characters from the Communications-Pac.

On transmit, a format 1 LD (Load) instruction has been attempted and there is no "valid" CCB. (MLCP firmware will set this bit to 1, return the CCP pointer to the address of this LD instruction, and execute a WAIT (Wait) instruction. At the next channel request interrupt for this channel, this instruction will be attempted again. The significance of this firmware action is specific to the related Communications-Pac and line protocol.)

Bits 5 and 6—for programming use

These two bits may be used by the CCP as indicators to the main memory program. For example, these bits could be used to indicate (1) the type of CDB (such as header, text, or end of transmission) to which this CCB is related or (2) that a cyclic redundancy check error has been detected (if bit 1 of LCT byte 17/49 is not used for this purpose).

LCT Byte 17/49

Bit 1—data check error

0 - No data check error has occurred.

1 - A data parity error has been detected by firmware, or the CCP has set this bit after detecting a cyclic redundancy check error. (In both cases, this bit setting is relevant only for a receive operation and hence only for LCT byte 17.)

Bit 3—data set or Communications-Pac status change

0 - The condition described for bit value 1 does not exist.

1 - A data set or Communications-Pac status change has been recorded in LCT byte 14/46 and bit 1 of LCT byte 8/40 was set to 1.

Bit 3—data set or Communications-Pac status change

0 - The condition described for bit value 1 does not exist.

1 - A data set or Communications-Pac status change has been recorded in LCT byte 14/46 and bit 1 of LCT byte 8/40 was set to 1.

Bit 4—corrected memory error

0 - No corrected memory error has occurred.

1 - One or more hardware-corrected memory errors occurred in the CDB related to this CCB.

LCT BYTE 20—DATA SET AND COMMUNICATIONS-PAC CONTROL

Description:

This byte is used to load Communications-Pac line register 2 with data set and Communications-Pac control information; the contents of this byte are not meaningful until they have been loaded into line register 2. The significance of the information in this byte depends on the type of data set and Communications-Pac in use.

Programming Considerations:

This byte may be written and maintained from the main memory program or CCP. Its contents may be changed and line register 2 loaded at times appropriate to the communications application.

Line register 2 of a Communications-Pac line is shared by a receive channel and a transmit channel. Moreover, each bit position in line register 2 has only one meaning for both channels. Therefore, line register 2 is loaded from LCT byte 20. If the receive channel is to be serviced by *one* CCP and the transmit channel is to be serviced by *another* CCP—and if simultaneous data transfers will occur over the line either CCP must perform an inclusive OR operation on the contents of LCT byte 20, loading the result into line register 2.

Byte Layout: (Typical RS232 Application)

	0	1	2	3	4	5	6	7
	DATA SET CONTROL					COMMUNICATIONS-PAC CONTROL		
20	DATA TERMINAL READY	REQUEST TO SEND	ASYNCHRONOUS SECONDARY CHANNEL TRANSMIT SYNCHRONOUS NEW SYNC	ASYNCHRONOUS TRANSMIT SPACE SYNCHRONOUS SPEED SELECT	ASYNCHRONOUS TRANSMIT MARK SYNCHRONOUS DIRECT CONNECT	LOOP-BACK TEST	RECEIVE ON	TRANSMIT ON

The meanings of bits 0, 1, and 2 are given in appropriate Bell System Technical References (see Preface).

Bit 3—transmit space (Asynchronous Line Communications-Pacs); speed select (Synchronous Line Communications-Pacs)

For Asynchronous Line Communications-Pacs, the values of bit 3 have the meanings described below.

0 - Transmit data supplied from the CCP.

1 - Hold the data output in a *space* (logical 0) condition. The Asynchronous Line Communications-Pac will continue to generate channel request interrupts at the normal character rate. (If the transmit space feature is to be used, a CCP must first "flush" the Communications-Pac by sending two padding characters after sending the last meaningful data character.)[1]

For Synchronous Line Communications-Pacs, bit 3 is used only if the data set supports data transfers at either of two speeds; bit 3 indicates which speed is desired. These speeds are defined in appropriate Bell System Technical References (see Preface).

[1] Transmit, space and transmit mark are mutually exclusive. Therefore, bits 3 and 4 of LCT byte 20 must not both be set to 1 if an Asynchronous Line Communications-Pac is in use.

Bit4—transmit mark (Asynchronous Line Communications-Pacs); direct connect (Synchronous Line Communications-Pacs)

For Asynchronous Line Communications-Pacs, the values of bit 4 have the meanings described below.

0 - Transmit data supplied from the CCP.

1 - Hold the data output in a *mark* (logical 1) condition. The Asynchronous Line Communications-Pac will continue to generate channel request interrupts at the normal character rate. (If the transmit mark feature is to be used, a CCP must first "flush" the Communications-Pac by sending two padding characters—usually DEL—after sending the last meaningful data character.)[2]

For Synchronous Line Communications-Pacs, the values of bit 4 have the meanings described below.

0 - The data set clock is enabled and used (the normal case).

1 - The MLCP's fixed-rate clock is enabled and used (the data terminal equipment is direct-connected to the Synchronous Line Communications-Pac).

Bit 5—"loop-back" test

0 - No "loop-back" test.

1 - "Loop back" to the Communications-Pac's receive channel the data sent to the transmit channel of the same line. For Asynchronous Line Communications-Pacs, the data transfer will take place at the line speed indicated in bits 4 through 7 of line register 4 of the Communications-Pac. For Synchronous Line Communications-Pacs, the data transfer will take place at the speed of the MLCP's fixed-rate clock.

Bit 6—receive ON

0 - Set the receiver OFF for this communications line.

1 - Set the receiver ON for this communications line.

Bit 7—transmit ON

0 - Set the transmitter OFF for this communications line.

1 - Set the transmitter ON for this communications line.

PROGRAMMING WORK AREA

LCT bytes 23 through 31, 52, and 56 through 63 are available to the communications application as channel- and line-specific work areas to be used as necessary. These bytes may be written and read by the CCP; they may also be written by the main memory program – through IO (Output LCT Byte) instructions. (LCT byte 55 which was formerly a work byte is now used by the Input LCT Byte instruction. When used by the Input LCT Byte instruction, the address is a 6-bit quantity located in bits 2 through 7. Bits 0 and 1 are ignored by the firmware. This byte can be used by the channel program for temporary storage if the main memory program reloads LCT byte 55 before executing each Input LCT Byte instruction.

Normally, one of these bytes is used to load line register 4 of the Communications-Pac since no other LCT byte is used for that purpose.

LAYOUT OF LCT BYTES

This subsection presents a detailed layout of each byte in a line control table (LCT). The purpose is to indicate (1) how each bit position of an LCT should be set up before communications processing begins over the related channels and (2) how each bit position of each LCT byte is used after communications processing has begun.

[2] Transmit space and transmit mark are mutually exclusive. Therefore, bits 3 and 4 of LCT byte 20 must not both be set to 1 if an Asynchronous Line Communications-Pac is in use.

The following key applies to the LCT layout. One of the terms appears in the lower right-hand corner of each bit position.

1(A/B/C)—This bit position *must* be written with an appropriate value by software before communications processing begins; the value of this bit position affects hardware/firmware operations. (Distinctions among 1A, 1B, and 1C are described below.)

1A This bit position may be modified by software at appropriate times after communications processing has begun.

1B This bit position may be modified by the main memory program—but not by the CCP—at appropriate times after communications processing has begun.

1C This bit position normally should not be modified by software after communications processing has begun.

2(A/B/C)—This bit position *may* be written with an appropriate value by software before communications processing begins; the value of this bit position affects hardware/firmware operations. (Distinctions among 2A, 2B, and 2C are described below.)

2A This bit position may be modified by software at appropriate times after communications processing has begun.

2B This bit position may be modified by the main memory program—but not by the CCP—at appropriate times after communications processing has begun.

2C This bit position normally should not be modified by software after communications processing has begun.

3 —This bit position may be used by software as required, the value in this bit position does not directly affect hardware/firmware operations. This bit position may be written with an application-specific value before communications processing begins and/or it may be modified at appropriate times thereafter.

4(A/B/C)—This bit position must be reset to 0 before communications processing begins. Thereafter it is maintained by firmware. (Distinctions among 4A, 4B, and 4C are described below.)

4A This bit position may be read by the CCP at appropriate times after communications processing has begun. The CCP may reset this bit position to 0, when appropriate.

4B This bit position may be read by the CCP at appropriate times after communications processing has begun. The CCP should not modify this bit position, however.

4C This bit position's contents should not be consulted by software.

5 —This bit position must be reset to 0 before communications processing begins. Thereafter is not used.

A detailed description of those LCT bytes of user-visible significance appears earlier in this section. In a few cases, a bit position's significance depends on the type of Communications-Pac used for the line to which the LCT is related.

LCT Bytes Used by Firmware

The description that follows is intended for use as an aid in program development.

0/32— This location contains the most significant 10 bits of the channel address. This field indicates that this channel has accepted one of the following function codes:

1C, 08, 1E, 05, 03, 0B

(Refer to Table 2-1.)

5/37— Bits 2, 5, and 6 are used by BLBT, BLBF, BLCT, BLCF, BET, BEF.
Bit 2 last block indicator
Bit 5 last character indicator
Bit 6 equal indicator

9/41— Refer to this chart for value of bits 3, 4.

	CCB Order in RAM	*LCT9/41 Bit 3, 4 (Note 2)*
	CCB	
	0	11
(Note 1)	1	00
	2	01
	3	10

NOTES: 1. After an initialize or CCB reset, CCB 1 is the first CCB to be executed.

2. The value in bits 3, 4 is always one less than the active CCB (i.e., the CCB currently being processed).

18/50— Bits 4 through 7 are the four most significant bits of the Branch to Subroutine (BS) instruction return address.

19/51— Bits 0 through 7 are the eight least significant bits of the Branch to Subroutine (BS) instruction return address (i.e., the address of the next sequential instruction of the CCP after the BS instruction).

21/53— Bits 0 through 7 are used by the MLCP firmware to store the contents of the R-register whenever a CCP executes a Wait instruction or whenever a firmware pause occurs. When the CCP resumes, the firmware restores the contents of LCT bytes 21 and 53 to the R-register.

55— This location is normally used for the address of the LCT location to be input by the Input LCT Byte instruction. (See also "Programming Work Area" earlier in this section.)

CAUTION

In the layout diagrams that follow, the LCT bytes labeled "firmware use only" are presented solely for the interpretation of memory dumps. These bits must *not* be manipulated or tested by the CCP.

FIRMWARE USE ONLY

	0	1	2	3	4	5	6	7
0/32	CHANNEL NUMBER (WRITTEN BY FIRMWARE I/O) 4C	4C	4C	4C	4C	4C	4C	4C
1/33	4C	4C	4C	4C	4C	4C	4C	4C

CHARACTER CONFIGURATION

	0	1	2	3	4	5	6	7
2/34	CHARACTER LENGTH 1C	1C	0 5	CHARACTER PARITY 2C	ASYNCHRONOUS STOP BITS 1C SYNCHRONOUS 0 5	CYCLIC REDUNDANCY CHECK POLYNOMIAL 2C	2C	0 5

CYCLIC REDUNDANCY CHECK RESIDUE

	0	1	2	3	4	5	6	7
3/35	4A	4A	4A	4A	4A	4A	4A	4A
4/36	4A	4A	4A	4A	4A	4A	4A	4A

FIRMWARE USE ONLY

	0	1	2	3	4	5	6	7
5/37	4C	4C	LAST BLOCK INDICATOR 4C	4C	4C	LAST CHARACTER INDICATOR 4C	EQUAL INDICATOR 4C	4C

CCP POINTER

	0	1	2	3	4	5	6	7
6/38	0 5	0 5	0 5	0 5	MOST SIGNIFICANT BITS 1B	1B	1B	1B
7/39	LEAST SIGNIFICANT BITS 1B	1B	1B	1B	1B	1B	1B	1B

CHANGE CONTROL FOR DATA SET AND COMMUNICATIONS-PAC STATUS

	0	1	2	3	4	5	6	7
8/40	SCAN CONTROL 2A	SET BIT 3 OF LCT BYTE 17/49 IF CHANGE 2A	INTERRUPT MAIN MEMORY PROGRAM IF CHANGE 2A	START CHANNEL CONTROL PROGRAM IF CHANGE 2A	0 5	0 5	0 5	0 5

FIRMWARE USE ONLY

	0	1	2	3	4	5	6	7
9/41	4C	4C	4C	CCB POINTER 1 LESS THAN ACTIVE CCB 4C	4C	4C	4C	4C

FIRMWARE USE ONLY

	0	1	2	3	4	5	6	7
10/42	4C	4C	4C	4C	4C	4C	4C	4C
11/43	4C	4C	4C	4C	4C	4C	4C	4C

RETURN CHANNEL NUMBER AND INTERRUPT LEVEL

	0	1	2	3	4	5	6	7
12/44	RETURN CHANNEL NUMBER–MOST SIGNIFICANT BITS 2B	2B	2B	2B	2B	2B	2B	2B
13/45	RETURN CHANNEL NUMBER–LEAST SIGNIFICANT BITS 2B	2B	INTERRUPT LEVEL 2B	2B	2B	2B	2B	2B

DATA SET AND COMMUNICATIONS-PAC STATUS

	0	1	2	3	4	5	6	7
	DATA SET STATUS					COMMUNICATIONS-PAC STATUS		
14/46	DATA SET READY 4B	CLEAR TO SEND 4B	CARRIER DETECTOR 4B	RING INDICATOR 4B	ASYNCHRONOUS SECONDARY CHANNEL RECEIVE 4B SYNCHRONOUS 0 5	0 5	RECEIVE OVERRUN 4B	ASYNCHRONOUS FRAMING ERROR 4B SYNCHRONOUS TRANSMIT UNDERRUN 4B

MASK FOR DATA SET AND COMMUNICATIONS-PAC STATUS

	0	1	2	3	4	5	6	7
15/47	2A	2A	2A	2A	2A	2A	2A	2A

LCT STATUS BYTES

	0	1	2	3	4	5	6	7
16/48	INTERRUPT MAIN MEM FROM CCP (INTR) 5	INTERRUPT MAIN MEM FROM DATA SET SCAN 5	DATA SERVICE ERROR 4B	0 5	CCB SERVICE ERROR 4B	FOR PROGRAMMING USE 3	FOR PROGRAMMING USE 3	0 5
17/49	0 5	DATA CHECK ERROR 4B	0 5	DATA SET OR COMMUNICATIONS-PAC STATUS CHANGE 4B	CORRECTED MEMORY ERROR 4B	0 5	0 5	0 5

	0	1	2	3	4	5	6	7
	SUBROUTINE RETURN		FIRMWARE USE ONLY		POINTER			
18/50	4C	4C	4C	4C	MOST SIGNIFICANT BITS 4C	4C	4C	4C
19/51	LEAST SIGNIFICANT BITS 4C	4C	4C	4C	4C	4C	4C	4C

	0	1	2	3	4	5	6	7
	DATA SET AND COMMUNICATIONS-PAC CONTROL							
	DATA SET CONTROL					COMMUNICATIONS-PAC CONTROL		
20 (NOTE a)→	DATA TERMINAL READY 1A	REQUEST TO SEND 1A	ASYNCHRONOUS SECONDARY CHANNEL TRANSMIT 1A SYNCHRONOUS NEW SYNCH 1A	ASYNCHRONOUS TRANSMIT SPACE 1A SYNCHRONOUS SPEED SELECT 1A	ASYNCHRONOUS TRANSMIT MARK 1A SYNCHRONOUS DIRECT CONNECT 1A	LOOP-BACK TEST 1A	RECEIVE ON 1A	TRANSMIT ON 1A
	FIRMWARE USE ONLY							
21/53	CONTAINS R-REGISTER WHEN CCP EXECUTION SUSPENDED 4C	4C	4C	4C	4C	4C	4C	4C
22/54	4C	4C	4C	4C	4C	4C	4C	4C
	PROGRAMMING WORK AREA							
23 (NOTE b)→	3	3	3	3	3	3	3	3
24/56	3	3	3	3	3	3	3	3
25/57	3	3	3	3	3	3	3	3
26/58	3	3	3	3	3	3	3	3
27/59	3	3	3	3	3	3	3	3
28/60	3	3	3	3	3	3	3	3
29/61	3	3	3	3	3	3	3	3
30/62	3	3	3	3	3	3	3	3
31/63	3	3	3	3	3	3	3	3

NOTES: a. LCT 52 IS A NEW WORK AREA.
b. LCT 55 IS NOW USED BY INPUT LCT BYTE COMMAND

Worksheet

The following worksheet may be copied and used to indicate the value to be written into each bit position of an LCT before communications processing begins. Those bit positions that must be reset to 0 before communications processing begins are so indicated on the worksheet.

	0	1	2	3	4	5	6	7
0/32	0	0	0	0	0	0	0	0
1/33	0	0	0	0	0	0	0	0
2/34			0					0
3/35	0	0	0	0	0	0	0	0
4/36	0	0	0	0	0	0	0	0
5/37	0	0	0	0	0	0	0	0
6/38	0	0	0	0				
7/39								
8/40					0	0	0	0
9/41	0	0	0	0	0	0	0	0
10/42	0	0	0	0	0	0	0	0
11/43	0	0	0	0	0	0	0	0
12/44								
13/45								
14/46	0	0	0	0	0	0	0	0
15/47								
16/48	0	0	0	0	0			0
17/49	0	0	0	0	0	0	0	0
18/50	0	0	0	0	0	0	0	0
19/51	0	0	0	0	0	0	0	0
* 20								
21/53	0	0	0	0	0	0	0	0
22/54	0	0	0	0	0	0	0	0
23/55								
24/56								
25/57								
26/58								
27/59								
28/60								
29/61								
30/62								
31/63								
52								

SECTION 6

MLCP INTERFACES

This section describes the following subjects, all of which relate to interfaces between the MLCP and either the main memory program or data communications equipment/lines:

- MLCP channel number addressing from main memory program
- MLCP interrupts to main memory program
- MLCP control of data sets and Communications-Pacs
- MLCP monitoring of data set and Communications-Pac status
- MLCP parity checking and generation
- MLCP cyclic redundancy checking
- Data transfer rates for MLCP communications lines

MLCP CHANNEL NUMBER ADDRESSING FROM MAIN MEMORY PROGRAM

As described in Section 2, whenever the main memory program issues an input/output instruction to the MLCP, it must provide a 10-bit channel number (along with an appropriate 6-bit function code). This information is placed on the Megabus during execution of the input/output instruction. The 10-bit channel number identifies a specific MLCP (as distinguished from any other physical unit on the Megabus) *and* one of the MLCP's 16 communications channels.

The 10 bits of the channel number are divided into two parts: the six high-order bits identify the MLCP's unique, fixed channel number (Megabus address), which is set by a switch on the MLCP; the four low-order bits indicate one of the 16 communications channels.

The 10-bit channel number is stored in bits 0 through 9 of a word in main memory or a register (with the function code stored in bits 10 through 15). On the Megabus, the 10-bit channel number occupies bits 8 through 17 of the address portion of the Megabus (with the function code occupying bits 18 through 23).

Table 6-1 shows the format of the 10-bit channel number and its relationship to MLCP channels and line numbers.

MLCP INTERRUPTS TO MAIN MEMORY PROGRAM

Under certain conditions (described below), the MLCP can interrupt execution of the main memory program. The interrupt is generated on behalf of one of the MLCP's 16 communications channels, each of which can have an interrupt level assigned to it. (The interrupt level for a channel can be unique or it can be shared with one or more other channels.)

A channel can be assigned an interrupt level when its LCT area is set up by the MLCP Loader or by a "block mode write"; alternatively, a channel can be assigned an interrupt level by execution of an IO (Output Interrupt Control) instruction in the main memory program. A channel's interrupt level (together with the return channel number–i.e., the central processor's channel number) is stored in LCT bytes 12 and 13 (for a receive channel) or in LCT bytes 44 and 45 (for a transmit channel). The format of the LCT bytes is described in Section 5.

TABLE 6-1. MLCP CHANNEL NUMBER ADDRESSING

10-Bit Channel Number in Memory and on Megabus[a]	MLCP Channel Number	Line Number	Channel Type
xxxxxx0000	0	0	Receive
xxxxxx0001	1	0	Transmit
xxxxxx0010	2	1	Receive
xxxxxx0011	3	1	Transmit
xxxxxx0100	4	2	Receive
xxxxxx0101	5	2	Transmit
xxxxxx0110	6	3	Receive
xxxxxx0111	7	3	Transmit
xxxxxx1000	8	4	Receive
xxxxxx1001	9	4	Transmit
xxxxxx1010	10	5	Receive
xxxxxx1011	11	5	Transmit
xxxxxx1100	12	6	Receive
xxxxxx1101	13	6	Transmit
xxxxxx1110	14	7	Receive
xxxxxx1111	15	7	Transmit

[a] xxxxxx represents the six bits of the MLCP's fixed channel number. This value is set by a switch on the MLCP. The value is the same for each communications channel of a given MLCP.

After a channel has been assigned a nonzero interrupt level as described above, the MLCP will interrupt the main memory program on behalf of this channel under any of the circumstances described below. Refer also to the discussion of CPU interrupts in Appendix A.

- o At the end of processing relative to a CCB whose control byte (byte 5) has a 1 in bit 0 (interrupt control). (The bit can be set to 1 by an IO (Output CCB Control) instruction in the main memory program.)
- o When a data set or Communications-Pac status change is recorded in LCT byte 14 or 46 and bit 2 of LCT byte 8 or 40 is set to 1.
- o Upon completion of a "block mode read" or "block mode write" that used this channel.
- o Upon execution of an INTR instruction in the CCP.

Remember that an interrupt will occur under the above circumstances only if the channel has a *nonzero* interrupt level when the circumstance occurs.

Note that any of the following actions causes a channel's interrupt level to be reset to zero:

- o Execution, in the main memory program, of an IO (Output MLCP Control) instruction that initializes the MLCP.
- o Execution, in the main memory program, of an IO (Output Channel Control—channel initialize) instruction.
- o Execution, in the main memory program, of an IO (Output Interrupt Control) instruction that specifies a zero value for the interrupt level.

If a CCB is active when the channel's interrupt level is reset to zero, an interrupt of the main memory program will *not* occur at the end of processing relative to the CCB—even if bit 0 (interrupt control) of the CCB control byte (byte 5) is set to 1.

In those cases where a channel's interrupt level is lower (higher-numbered) than the interrrupt level at which the main memory program is currently running, an attempt to interrupt the main memory program will result in a NAK from the central processor. The interrupt will be "deferred" and placed in a queue, which can store as many as three deferred interrupts per channel.

Following a change in the interrupt level at which the main memory program is running, the MLCP will attempt to "work off" any backlog of deferred interrupts during the background firmware scanning periods mentioned in Section 1. Beginning with the lowest numbered channel that has a deferred interrupt queued, the MLCP will attempt one interrupt for each such channel before cycling back to the queue for the lowest numbered channel that has a deferred interrupt queued.

MLCP CONTROL OF DATA SETS AND COMMUNICATIONS-PACS

For each communications line, the MLCP controls the data set (if any) and Communications-Pac by means of a Communications-Pac control register (line register 2) for that line. (A Communications-Pac has a separate line register 2 for each line it accommodates.) The value in line register 2 affects the operations of both channels of one line.

Line register 2 of a Communications-Pac is loaded from a CCP that services one or both channels of the related communications line. The CCP must first ensure that the desired control value exists in LCT byte 20; then, at an appropriate time, the CCP places this control value in the MLCP's R-register and loads it into line register 2 of the Communications-Pac. The control value is effective immediately. *

Refer to the appropriate appendixes for programming considerations for the Communications-Pacs available for attachment to the MLCP.

MLCP MONITORING OF DATA SET AND COMMUNICATIONS-PAC STATUS

If so directed by the CCP, MLCP firmware will periodically scan data set and Communications-Pac status relative to a given communications line—as reflected in a Communications-Pac status register (line register 5) for that line. Whenever the MLCP firmware scan detects certain types of status changes (as predefined by the CCP), it will store the entire contents of line register 5 in LCT byte 14 or LCT byte 46. Subsequent actions to be taken are also predefined by the CCP.

A CCP requests an MLCP firmware scan of line register 5 by setting to 1 bit 0 (scan control) of LCT byte 8 or LCT byte 40. By setting appropriate bits in a mask contained in LCT byte 15 or LCT byte 47, the CCP predefines which types of data set or Communications-Pac status changes will cause the entire contents of line register 5 to be copied into LCT byte 14 or LCT byte 46. The CCP predefines the subsequent action(s) to be taken by appropriate settings of bits 1, 2, and 3 in LCT byte 8 or LCT byte 40:

- o If bit 1 is set to 1, a detected status change will cause bit 3 of LCT byte 17 or LCT byte 49 to be set to 1.
- o If bit 2 is set to 1, a detected status change will cause the "active" CCB to be terminated and the main memory program to be interrupted.
- o If bit 3 is set to 1, a detected status change will cause the CCP to be started. (This action will be taken only if bit 2 is reset to 0.)

Each time a firmware scan occurs, the following sequence of actions is performed:

1. An exclusive OR operation is performed on (1) the contents of LCT byte 14 or LCT byte 46 and (2) the contents of line register 5.
2. A logical AND operation is performed on the result from step 1 and the contents of the mask in LCT byte 15 or LCT byte 47.

If a nonzero result is produced by the logical AND operation in step 2, a data set or Communications-Pac status change has occurred. The contents of line register 5 are copied into LCT byte 14 or LCT byte 46 and subsequent action(s) are based on the settings of bits 1, 2, and 3 of LCT byte 8 or LCT byte 40. (If the CCP is started in response to a status change, it can read LCT byte 14 or 46 to ascertain what type of status change has occurred.)

The MLCP firmware scan is a low-priority activity that is independent of input/output operations from the main memory program and CCP processing. The firmware scan will occur at least every one-half second.

MLCP PARITY CHECKING AND GENERATION

A CCP can direct the MLCP to check parity during receive operations and to generate parity during transmit operations.

For receive operations, the CCP must first ensure that bit 3 of LCT byte 2 indicates the desired type of parity. Then, as data characters are transferred from the receive channel's line register 1 to the MLCP's R-register, the operand of each RECV (Receive) instruction can indicate that a parity check is to be performed. The parity check will include all bits of the character length specified in bits 0 and 1 of LCT byte 2. (In this case, the leftmost bit of each data character is a parity bit; the parity bit may be either 0 or 1 as the data character arrives in the R-register.) If the MLCP detects a parity error, it will set to 1 bit 1 (data check error) of LCT byte 17 (LCT status byte 2).

For transmit operations, the CCP must first ensure that bit 3 of LCT byte 34 indicates the desired type of parity. Then, as data characters are transferred from the MLCP's R-register to the transmit channel's line register 1, the operand of each SEND (Send) instruction can indicate that parity is to be generated. (In this case, the leftmost bit of each data character—whose length is specified in bits 0 and 1 of LCT byte 34—is a parity bit; the parity bit must be 0 as the SEND instruction is issued.) The MLCP generates the proper value for the parity bit as the data character is transferred to the transmit channel's line register 1.

Remember that both channels of one line use the same character length. Both channels of a line share one line register 6, which defines character configuration, in a Communications-Pac.

MLCP CYCLIC REDUNDANCY CHECKING

A CCP can direct the MLCP to use its block-check hardware to perform cyclic redundancy checking during receive or transmit operations. Normally, cyclic redundancy checking is requested only for higher speed data transfers.

For receive operations, the CCP must first ensure that bits 5 and 6 of LCT byte 2 indicate the desired cyclic redundancy check polynomial. See Table 6-2. Then, as data characters are received, the CCP can obtain either unconditional or conditional cyclic redundancy checking.

- *Unconditional* cyclic redundancy checking is obtained if the operand of each RECV (Receive) type 1 or 3 instruction indicates that a check is to be performed; the cyclic redundancy check residue (in LCT bytes 3 and 4) will be updated as each data character is loaded into the MLCP's R-register.
- *Conditional* cyclic redundancy checking is obtained if RECV instructions type 0 or 2 do not indicate that a check is to be performed but each received data character is analyzed in the MLCP R-register and a check is performed as a separate operation—by means of a CCH (Calculate Block Check) instruction—only when desired; cyclic redundancy check residue will be updated each time a CCH instruction is executed.

At the end of a received data block, a cyclic redundancy check must be performed on the block-check character(s); afterwards, the cyclic redundancy check residue should have the value expected. (If 12-bit cyclic redundancy check residue has been accumulated, the contents of LCT byte 3 may need to be shifted two bit positions to the right; the SR (Shift R-Register Right) instruction is available for this purpose.)

For transmit operations, the CCP must first ensure that bits 5 and 6 of LCT byte 34 indicate the desired cyclic redundancy check polynomial. See Table 6-2. Then, as each data character is transferred from the MLCP's R-register to the transmit channel's line register 1, the operand of each SEND (Send) instruction can indicate that a check is to be performed; the cyclic redundancy check residue (in LCT bytes 35 and 36) will be updated as each data character is transferred into the transmit channel's line register 1. At the end of each block, the CCP must transmit the cyclic redundancy check residue accumulated in LCT bytes 35 and 36. (If 12-bit cyclic redundancy check residue has been accumulated, the contents of LCT byte 35 may need to be shifted two bit positions to the right before being transmitted; the SR (Shift R-Register Right) instruction is available for this purpose.)

Cyclic redundancy check residue can be reset to zero at appropriate times by the CCP or by the main memory program. (The main memory program can use the IO (Output LCT Byte) instruction for this purpose.)

Whenever a CRC error is detected, it is the channel program responsibility to set LCT 17 bit 1, data check error.

TABLE 6-2. CYCLIC REDUNDANCY CHECK INFORMATION

Bits 5 and 6 of LCT Byte 2 or LCT Byte 34	**CRC Polynomial Used**	**CRC Residue Bit Length**	**Configuration of CRC Residue in LCT Bytes 3/35 and 4/36**	**Transmission Modes**
00	$x^{16}+x^{15}+x^2+1$	16	0 7 3/35 MSB Must be zero at start. 4/36 LSB	IBM BSC CRC-16
01	$x^{16}+x^{12}+x^5+1$	16	0 7 3/35 MSB LCT 3/35, 4/36 must be all ones at start. 4/36 LSB	HDLC, SDLC, ADCCP, CCITT Recommendation
10	$x^{12}+x^{11}+x^3+x^2+x+1$	12	012 567 3/35 MSB ▨ Must be zero at start. 4/36 ▨ LSB	IBM Transcode CRC-12
11	x^8+1	8	0 7 3/35 LRC Must be zero at start.	LRC-8, Basic Mode ASCII

LSB – Least significant bits; MSB – Most significant bits; MBZ – Must be zero.

NOTES:
1. On a transmit, LCT byte 35 should be sent first.
2. On a receive, LCT byte 3 should be compared with the first CRC byte received.
3. For code 11, only LCT byte 3 or LCT byte 35 is applicable. LCT byte 35 must be sent with odd parity.

DATA TRANSFER RATES FOR MLCP COMMUNICATIONS LINES

The data transfer rate for a given communications line depends on two factors:

1. The type of Communications-Pac with which the line is associated and
2. The mode ("normal,"[1] direct-connect, or "loop-back" test) in which the line is being used.

Each channel of the communications line uses the line's data transfer rate. A channel's bit transfer rate (together with the defined length of the data characters) establishes the interval between channel request interrupts for that channel.

The following diagram summarizes the relationship between data transfer rate, type of Communications-Pac, and mode of line operation.

Type of Communications-Pac	Mode of Line Operation	Data Transfer Rate
Asynchronous Type Communications-Pac	Normal Direct-connect Loop-back test	Governed by clock in Communications-Pac. Speed indicated by settings of bits 4 through 7 of line register 4 of Communications-Pac.
Synchronous Type Communications-Pac	Normal	Governed by clock in data set. (If the data set clock permits either of two data transfer rates to be used, bit 3 of line register 2 can be used to select the desired rate.)
	Direct-connect Loop-back test	Governed by MLCP's fixed-rate clock. This clock is set to *one* rate by hardware configuration; the possible settings are shown in Table 6-3. The selected rate applies to any line that uses the MLCP's fixed-rate clock.

TABLE 6-3. POSSIBLE SETTINGS FOR MLCP's FIXED-RATE CLOCK

Clock Rate (in Bits per Second)	Switch Setting (Hexadecimal)
800	1
1,200	2
1,760	3
2,152	4
2,400	5
4,800	6
9,600	7
14,400	8
19,200	9
28,800[a]	A
38,400[a]	B
57,600[a]	C

[a] Maximum rate depends on adapter type; refer to the appendixes for specific Communications-Pacs.

[1] "Normal" mode signifies that the line is connected to the Communications-Pac by means of data communications equipment (i.e., the line is not direct-connected) and no "loop-back" of transmitted data is taking place.

SECTION 7

PROGRAM PREPARATION AND LOADING

Either of two methods can be used to write into the LCT area and the CCP area of MLCP RAM before communications processing begins.

- The first method involves the use of the Honeywell-provided MLCP Loader, which is supplied as an object module. The MLCP Loader must be linked with another object module that contains the user-created block to be written into MLCP RAM. The resultant load module is executed in the central processor to write into MLCP RAM.
- The second method involves the use of one or more "block mode write" operations from a program executed in the central processor. The "block mode write" is a prescribed sequence of central processor input/output instructions that cause a user-created block to be written into a designated area of MLCP RAM.

The result of both operations is similar: the desired LCT's are written with values appropriate to the beginning of communications processing and one or more channel control programs are written into the CCP area of MLCP RAM.

The remainder of this section is devoted to a description of the MLCP Loader and the "block mode write."

MLCP LOADER

The MLCP Loader can be used to write a user-created block into MLCP RAM, beginning with RAM byte 0. The block (RAM image) can contain up to 3584 bytes. The block establishes the user-specified values in the LCT area and the CCP area of RAM before communications processing begins.[1] (The CCB area of RAM is not written by the MLCP Loader; instead, the CCB area is written by input/output instructions issued dynamically by the main memory program.)

The following steps are normally followed to use the MLCP Loader to write a block into MLCP RAM.

1. Create a source language module that contains a load control block (described below), a call to the MLCP Loader, an error check routine, and additional executable instructions (as desired). (Reference: *GCOS/BES Assembly Language* manual.)
2. Submit the source language module to the Macro Preprocessor. (Reference: *GCOS/BES Program Development Tools* manual.)
3. Assemble the source language module. (Reference: *GCOS/BES Program Development Tools* manual.)
4. Link the resultant object module with the MLCP Loader (which is supplied as an object module named ZQMLIN). (References: *GCOS/BES Software Overview* and *GCOS/BES Program Development Tools* manual.)
5. Load and execute the resultant program.

The MLCP Loader uses channel 1 of communications line 0 to write into MLCP RAM. Therefore, line 0 must be serviced by a Communications-Pac.

[1] Remember that those LCT bytes reserved for firmware use must be written with zeros before communications processing begins. Section 5 provides a detailed layout of LCT bytes.

When the MLCP Loader is called, it first initializes the MLCP; all of RAM is reset to zero. The MLCP Loader next writes the LCT bytes for line 0 by issuing 64 IO (Output LCT Byte) instructions. Finally, the MLCP Loader performs a "block mode write" to transfer the remainder of the user-supplied block to RAM. If an error condition is encountered, the MLCP Loader writes an appropriate error code into the "return value word" of the load control block and returns control to its caller (which should always inspect the "return value word" upon being returned to by the MLCP Loader).

Load Control Block

Table 7-1 describes the format of the load control block, which must contain the following five elements in the prescribed order.

1. Range of RAM image
2. MLCP address on Megabus
3. Return value word
4. Register save area
5. RAM image

The load control block exists in a user-created module that is processed by the Macro Preprocessor, assembled, and linked with the MLCP Loader.

The "range of RAM image" (word 0 of the load control block) can be calculated by the Assembler if you use the technique shown under "Sample Program," below.

The "MLCP address on Megabus" (word 1 of the load control block) provides the high-order six bits for the channel/function-code words used in input/output instructions issued by the MLCP Loader. (The MLCP Loader provides the low-order 10 bits of the channel/function-code words.)

The "RAM image" (words 19 and following of the load control block) is a straightforward image of the bytes that are to be written into MLCP RAM beginning at byte 0. The first 512 bytes will be written into the LCT area of RAM; those bytes reserved for firmware use *must* be written with zeros (see Section 5). Bytes after the first 512 are written into the CCP area of RAM. The total "RAM image" must not exceed 3584 bytes; otherwise, the MLCP Loader will return to its caller with an error indicator (1) in "return value word" (word 2 of the load control block).

Sample Program

The following source language code shows a sample load control block followed by a call to the MLCP Loader. The call could in turn be followed by error check coding and other executable instructions (as desired).

This module must be processed by the Macro Preprocessor, assembled, and linked with the MLCP Loader.

```
          TITLE   LOADIT
ZQRAMS    DC      END-START          Range of RAM image (in words)
          DC      Z'FC00'            MLCP address on Megabus
          DC      0                  Return value word
          RESV    16,0               Register save area
START     EQU     $
          .                          RAM image to be written
          .                          beginning at RAM byte 0
          .
END       EQU     $
LOAD      CALL    ZQMLIN,ZQRAMS      Call to MLCP Loader
          .
          .
          .
          END     LOADIT,LOAD
```

TABLE 7-1. FORMAT OF LOAD CONTROL BLOCK

Word	Length	Contents	Description
0	1 word	Range of RAM image	This word indicates the range (length) of the image to be written to MLCP RAM. The binary value of this word indicates the range in terms of central processor *words.* The label of word 0 must appear as an argument in the CALL statement that invokes the MLCP Loader.
1	1 word	MLCP address on Megabus	This word indicates the six high-order bits of the MLCP address (i.e., the address physically set by a switch on the MLCP). The remainder of the word must be zeros. Format of word 1: Bits 0-5 - MLCP address Bits 6-15 - Zeros
2	1 word	Return value word	This word must initially be cleared to zero. When the MLCP Loader returns to its caller, this word will contain one of the values listed below. This word should be checked by the caller when the MLCP Loader returns to it. 0 - The load has been completed without error. 1 - The range of the RAM image is inappropriate (i.e., 0 < range < 3584 bytes). 2 - An uncorrectable input/output error has occurred during loading. 3 - No unit exists at the MLCP address specified in word 1. 4 - A non-MLCP unit exists at the MLCP address specified in word 1.
3-18	16 words	Register save area	When the MLCP Loader is called, it saves here the contents of the following 16 registers: B1- through B7-register, R1- through R7-register, I-register, M-register. The MLCP Loader restores the saved contents of these registers before returning to the caller.
19-n	Up to 1792 words (3584 bytes)	RAM image	This image will be written into MLCP RAM beginning at byte 0. This image must not exceed 3584 bytes (the combined length of the LCT area and the CCP area of RAM).

"BLOCK MODE WRITE"

A "block mode write" is a convenient means of transferring a user-created block from main memory into the LCT area and/or the CCP area of MLCP RAM.[2] The

[2] A "block mode read" is similar to a "block mode write" except that (1) a *receive* channel is used, (2) a different bit is set in the control word used with the IO (Output Channel Control) instruction, and (3) there are no restrictions on the MLCP RAM area that can be included in the data transfer. See Appendix B.

"block mode write" can be used prior to the beginning of any communications processing over the MLCP or it can be used on a more limited, channel-by-channel basis concurrent with communications processing over other channels (i.e., channels not affected by the "block mode write").

A "block mode write" differs from the MLCP Loader (described above) in the following respects:

- o A "block mode write" does not cause the MLCP (or any channel) to be initialized.
- o A "block mode write" can be performed over *any* transmit channel serviced by a Communications-Pac.
- o A "block mode write" can transfer a block to start anywhere in the LCT area or CCP area of RAM.
- o A "block mode write" must not write into the portion of the LCT dedicated to the transmit channel being used for the write operation.

To perform a "block mode write," the following sequence of steps is used in a main memory program:

1. Ensure that the desired block exists in main memory.
2. Ensure that the transmit channel to be used for the "block mode write" is in a "stop input/output" condition.
3. Execute an IOLD (Output CCB Address and Range) instruction.
4. Execute an IO (Output CCB Control–format 2) instruction.
5. Execute an IO (Output Channel Control–start "block mode write") instruction.

Each "block mode write" uses one CCB for the designated transmit channel. As the "block mode write" is taking place, no CCP is executed for the channel involved and no data is transmitted.

When a "block mode write" is completed, the transmit channel will be returned to a "stop input/output" condition. The main memory program will be automatically interrupted at the interrupt level previously established for the transmit channel. (If no interrupt level has been established for this channel, the interrupt will not occur.)

If the *LCT area* of MLCP RAM is involved in a "block mode write," remember that those bytes reserved for firmware use must be written with zeros; nonzero values in any of these bytes may produce unspecified results. Remember too that the "block mode write" must not write into the portion of the LCT dedicated to the transmit channel being used for the write operation.[3]

Format of CCB for "Block Mode Write"

The CCB for a "block mode write" is prepared by execution of an IOLD (Output CCB Address and Range) instruction and an IO (Output CCB Control–format 2) instruction in the main memory program. The resultant CCB has the format shown below.

BYTE		
0	LEAST SIGNIFICANT BITS	ADDRESS FIELD
1		
2	MOST SIGNIFICANT BITS	
3	LEAST SIGNIFICANT BITS	RANGE FIELD
4	MOST SIGNIFICANT BITS	
5	LEAST SIGNIFICANT BITS	RAM ADDRESS FIELD
6	0 0 0 0 | MSB	
7	0 0 0 0 0 0 0 0	

[3] If desired, the portion of the LCT dedicated to the transmit channel can be written by a "block mode write" performed over some *other* transmit channel.

The address field and the range field (bytes 0 through 4) of this CCB are similar to those fields in any other CCB. The RAM address field is unique. Initially it contains the 12-bit address of the RAM byte at which the "block mode write" will begin. Thereafter, the contents of the RAM address field increase by 1 as each byte is written to the MLCP.

CCB Status Field After "Block Mode Write"

After completion of a "block mode write," the status field (bytes 6 and 7) of the related CCB is updated. (Note that the function of CCB byte 6 changes from "most significant bits of RAM address" to "CCB status byte 2" at this time.)

The CCB status field (whose format is shown in Section 3) can be checked by use of an IO (Input CCB Status) or IO (Input Next CCB Status) instruction.

*

APPENDIX A

PROGRAMMING GUIDELINES FOR SELECTED MLCP FEATURES

This Appendix contains supplementary information on programming certain MLCP features or operations, such as initialization, Communications-Pac set up, line registers, certain CCP instructions, CPU interrupts, and error recovery. Cross-references are made to the text where appropriate.

Special programming techniques specific to Communications-Pac type are discussed in the separate appendixes on the Communications-Pacs. References in this appendix are made to those later appendixes where appropriate.

INITIALIZATION

A complete MLCP initialization occurs when the Power On switch is switched on the central processor control panel and also in response to the Output MLCP Control instruction (function code 01) with bit 0 set *(hard initialize)*. Because the hard initialize affects all channels, it normally occurs only at startup. In normal operation, the Output Channel Control instruction (function code 05) with bit 0 set *(channel initialize)*, is more appropriate because it allows other lines to continue operation. Channel initialize kills both the receive and transmit channels of the communications line because it causes line register 2 (LR2) to be cleared. In that sense it can always be assumed that channel initialize on one channel will cause the paired channel to cease operation. The paired channel is not completely initialized, however, because its CCB list has not been reset nor has its LCT area been cleared. In general, it is best to treat channel initialize as line initialize and issue commands in pairs, one to each channel pertaining to a particular line.

Channel initialize does not result in the clearing of the LSI chip which performs the serial/parallel conversion at the data set interface, nor does it clear anything other than LR2. It is common, therefore, to find error bits (flags) such as overrun, underrun, and framing error set in line register 5 after a channel initialize. It is also common to find line configuration information such as byte size still resident in the Communications-Pac. The specific conditions left in a noninitialized state are unique to each pac. In order to completely clear the pac and the LSI chip which performs the serial/parallel conversion at the data set interface, the central processor Clear push button may be used or a hard initialize command issued, but that will disrupt all MLCP channels. If it is not desirable to disrupt all channels, then special care should be taken to initialize the Communications-Pac via the CCP as discussed below in "Communications-Pac Setup."

In the normal case, it is preferable to use the Output Channel Control instruction (function code 05) with bit 2 set *(stop I/O)* or the CCP to control LR2 directly in order to cease communication on a line. This causes minimum change to the setup of the MLCP relative to the line and simplifies restart of communication.

COMMUNICATIONS-PAC SETUP

Communications-Pac setup is performed by the CCP between the Output Channel Control instruction (function code 05) with bit 1 set *(start I/O)* and the first WAIT instruction. In that interval the following actions must be performed as a minimum:

- o Load LR6
- o Load LR4 if applicable
- o Load LR2

The information to be loaded in LR6 will be found in the LCT byte 2 for receive and in LCT byte 34 for transmit. The information for LR4 has no specific LCT location assigned and could come from a LCT work location or from the CCP itself as a result of a Load (LD) instruction, Format 3 *(load immediate)*. The information for LR2 will be found in LCT byte 20 for both receive and transmit.

The LCT area must have already been set up by the main memory program. It is the task of the CCP to get that information to the Communications- Pac where it will be stored. The specific information to be output to the Communications-Pac is unique to each type. There is, however, very likely a sequence in which it must be done. The general sequence is configuration-word first and data set control-word last.

LCT byte 20 is one case where the same byte applies to both sides of the line. In most other cases, separate LCT locations exist for transmit and receive. In the case of LCT byte 20 which contains such indicators as request to send, data terminal ready, transmitter on, receiver on, there can only be one such set of indicators since there is only one LR2 and only one modem.

The number of Communications-Pac registers which must be reloaded to set up the Communications-Pac depends on what has previously been programmed. At the very first usage of the Communications-Pac after power on, all the registers (LR6, LR4, LR2) need to be set up. If a line has been in use and a stop I/O was issued, only LR2 need be reloaded to resume communications. If a channel initialize occurred, only LR2 need be reloaded, although one might prefer to completely reload the Communications-Pac registers if the reason for the channel initialize was an error.

Previously it was noted that the LSI chip performing the serial/parallel conversion at the data set interface was not cleared as a result of a channel initialize. If there is a possibility of error flags remaining set, the chip may be initialized by the CCP. On the transmit channel, two OUT LR1 with zero data for asynchronous Communications-Pacs or pad data (e.g., for synchronous Communications-Pacs) may be issued just after loading LR2 and that will clear any transmit errors remaining. On the receive channel, an IN LR1 may be issued just before loading LR2 and that will clear any received errors remaining. The data received as a result of this operation can be ignored.

Refer to the later appendixes for the specific information on each Communications-Pac.

ACCESS TO LINE REGISTERS

A Communications-Pac can have as many as eight line registers on the transmit channel and an additional eight on the receive channel; in many, the number is more like six or seven. Some registers are accessible on either channel (e.g., LR2) while other registers are specific to the transmit or the receive channel (e.g., LR1). This type of information is Communications-Pac-specific.

In practice, one programming difficulty arises: certain registers can only be written while others can only be read. (If a line register is used incorrectly or does not exist, the IN or OUT instruction will act as a no-op.) In the case of two important line registers, this means that once they are loaded, the CCP cannot directly determine their contents. The two most critical of these are LR2 and LR6. The programming solution is to make use of the association of LCT location and LR as follows:

LR2	=	LCT 20
LR6	=	LCT 2 (receive channel)
LR6	=	LCT 34 (transmit channel)

LR2, which controls the data set, should be kept identical to LCT byte 20 at all times. If, for example, the contents of LCT byte 20 show the *transmit* on set and the *receive* on reset in LR2 and it is desired to reverse this, the proper technique would be:

o Load LCT 20 into R
o Set receive on bit

- o Reset transmit on bit
- o Output to LR2
- o Store R into LCT 20

With this approach, the state of the data set interface can always be determined by reading LCT byte 20. This is particularly useful in case of error recovery activity.

Concerning LR6, which contains configuration information, the same kind of situation exists. In some instances there is a single LR6 while in other instances there are two. The configuration information does not have to be identical on the receive and tranmit channels. In any case, it is important to use the same technique to be sure that LR6 is always identical to the related LCT locations.

CCP INSTRUCTIONS

The following paragraphs are intended as an aid in programming the referenced instructions. This information supplements Section 4.

SEND Instruction

SEND causes the firmware to OR the contents of bit 7 of LR5 into LCT status, LCT byte 48 bit 2. This status bit is used to designate different things in different Communications-Pacs. For example, in the Synchronous Communications-Pac, it is a transmit underrun while in the Asynchronous Communications-Pac it is a framing error. The important point here is that LCT byte 48 bit 2 is not necessarily a transmit error but is only a copy of LR5 bit 7. In the case of the Asynchronous Communications-Pac, that represents a framing error so that a receive error is being reported on a transmit channel. (The transmit channel for that pac should reset this status bit before returning to the main memory program. There is no meaningful Asynchronous Communications-Pac transmit error. Refer also to Appendix C.)

RECV Instruction

RECV causes the firmware to OR the contents of bit 6 or LR5 into LCT byte 16 bit 2. This status bit may be used to designate different things in different Communications-Pacs. In most cases it means receive overrun. This status bit is on the receive channel.

OUT LR1 Instruction

OUT LR1 has the same function as SEND except that error status is not ORed from LR5 into the LCT area. Also, CRC and/or parity are not possible on OUT instructions. Because of the fact that error status is ignored, OUT LR1 is a means of avoiding conditions where error flags exist relative to the LSI chip performing the serial/parallel conversion.

IN LR1 Instruction

IN LR1 has the same function as RECV except that CRC and parity are not possible and the error status bit is not copied from LR5 into the LCT area. The fact that the error status is ignored makes IN LR1 a means of avoiding conditions where error flags may be set relative to the LSI chip performing the serial/parallel conversion (see above).

SFS Instruction

SFS causes the Communications-Pac to search for a synchronization character by manipulating the receive on bit in LR2. When SFS is executed, the firmware takes the following actions:

- o Loads LCT 20 into R
- o Masks off the receive on bit

- Outputs to LR2
- Turns on the receive on bit
- Outputs to LR2

This is another case where the CCP should always be sure that LCT 20 = LR2 so that SFS does not cause something else in LR2 to change.

BLBT, BLBF Instructions

The last block indicator in the CCB control byte can be tested by the CCP with the BLBT and BLBF instructions, but only after a DMA transfer has occurred. (The indicator is not valid until then.) This indicator is designed so that the main memory program can notify the CCP when the last block of a multiblock message has arrived so that some special terminating activity (e.g., disconnect) can take place. With this purpose, the BLBT or BLBF may be used at or near the end of the last block, possibly after deciding to end the block but before doing a GNB *(get next block).* (Refer to the sample table look-up program in Section 4.)

The last block indicator in the CCB is copied by firmware into a firmware use only location in the LCT area. BLBT and BLBF operate on that bit, not on the bit in the CCB. This copying of the bit occurs during the first DMA transfer initiated by an LD or ST. As a result, any attempt to use BLBT or BLBF before the first LD or second ST will not lead to meaningful results.

LD Instruction

In order for an LD to execute properly, there must be a valid CCB. If this is not the case and an LD is executed, all lower priority channels on that MLCP will overrun or underrun. Refer to "Valid CCB's" for detail on avoiding this problem. In addition, if there was no valid CCB, when a CCB is finally set up, a CCB service error will be indicated in bit 4 (LCT byte 48 bit 4) *CCB status.* The implication here is that a CCB service error indicates possible trouble on other channels as well as the one on which it is reported, if the valid bit has not been tested.

WAIT Instruction

The WAIT instruction, which has an execution time of 15 microseconds, actually consists of two parts. One part is a context save for the CCP then active. When a channel request interrupt occurs and a CCP is activated, the remainder of the WAIT (a context restore) occurs. (Refer to Section 1 for a definition of channel request interrupt.)

One of the functions of the context restore is to enable the firmware to read the contents of LCT byte 2 (LCT byte 34) and load the character size into the CRC hardware in the MLCP.

BLCT, BLCF Instructions

The use of the BLCT or BLCF instructions to test for end-of-range is mandatory. If one neglects to check for end of range by the CCP, the MLCP will increment the address and decrement the range beyond the end of the buffer. This could result in either the overwrite of other data in main memory on a receive or the transmission of data that should not have been transmitted.

INTR Instruction

INTR presently clears LCT 16 and then sets only bit 0 of LCT 16. The following sequence may be used to preserve the contents of LCT 16.

```
  |
  |
  |
LD 16

INTR

OR 16
  |
  |
  |
```

VALID CCB'S

When a CCB is set up by the main memory program, the Output CCB Control instruction (function code OF) must have bit 1 set (valid bit). When a start I/O[1] is issued for a channel or a LCT byte 8/40 *(data set scan)* starts the CCP, the firmware takes no specific notice of the valid bit except to detect a CCB service error. A CCB service error is deemed to exist if either an LD or an ST is executed and there is no

[1] Recall that this is: Output Channel Control (function code 05) with bit 1 set.

valid CCB. Assuming that the CCB is valid, the firmware resets the valid bit when a GNB instruction is executed.

The CCP instructions BVBT and BVBF enable the CCP to branch on the valid bit condition. Whether or not this instruction is necessary in a specific application depends on the way the program is established.

In one case, a number of CCB's are set up and the last is marked with a last block indicator. The program does BLBT or BLBF at the end of each block and if that is the end, the call is terminated.

In a more sophisticated case the software sets up a number of CCB's and does not mark any of them as last block. The software intent is to set up new CCB's as a response to the CCB interrupts and always keep ahead of the ability of the MLCP to use up CCB's. The advantage of this mode of operation is that it can lead to very high line utilization. A danger is that the main memory program may not keep up. In order to avoid a fatal error, the first instruction following the GNB should be a BVBT or BVBF so that the lack of a valid block can be detected before any damage is done. If no valid block exists, special action could be taken.

DATA SET SCAN

The *data set scan* can be activated by setting bit 0 of LCT byte 8 (LCT byte 40) to 1. When a scan occurs, the firmware ANDs the contents of LR5 with the contents of LCT byte 15 (LCT byte 47), ANDs the contents of LCT byte 14 (LCT byte 46), and compares the results for any change. As a result of the scan, the contents of LR5 replace the previous contents of LCT byte 14 (LCT byte 46). If a change occurred in the designated data set status bit(s), the change is indicated by the setting of bit 3 of LCT byte 17 (LCT byte 49) which is designated as *data set status change.* In addition, several other actions could take place:

- The CCP may be started if bit 3 of LCT 8 (LCT 40) is ON.
- The CPU may be interrupted if bit 2 of LCT 8 (LCT 40) is ON. If bits 2 and 3 are both ON, bit 2 will take precedence.

If the data set scan is being used when a CCP is not active, then it is reasonable to either start a CCP or interrupt the CPU. If a CCP is active, there is no obvious advantage to the data set scan. It may be more convenient to enable the data set scan so the CCP can check data set status in the LCT area as compared to checking LR5 directly. It is not clearly any faster one way or the other. Usually, data set status need only be checked at end of message to ensure that nothing has changed.

It is possible to use the data set scan to start the CCP even if the CCP is active. This is because the scan only occurs when the CCP is not running so no conflict is possible. When the CCP starts, however, there is a problem because the CCP would have to determine why it had been activated (channel request interrupt or data set scan). This decision would slow down the normal character processing loop enough to make this mode undesirable in most cases.

It is also possible to use the data set scan to interrupt the CPU while a CCP is active. This, however, can lead to confusion concerning input status (see "CPU Interrupts").

CPU INTERRUPTS

CPU interrupts are caused by the following three actions in the MLCP:

- Block termination (GNB with I bit[2] set)
- Data set scan
- INTR instruction

[2] The "I" bit is bit 0 of the CCB control byte.

A data set scan interrupt (status change) sets bit 11 of LCT status (specifically, bit 3 of LCT byte 17/49[3]). If in addition a data set scan interrupt were valid, then bit l of LCT byte 16/48 is set.

An INTR instruction sets bit 0 of LCT status (bit 0 of LCT byte 16).

As a response to either of these interrupts, the software could read LCT status directly via the Input LCT Byte instruction (function code IE). When a GNB occurs, LCT status is copied into the CCB status and LCT status is cleared. When a CCB completes, status bit 3 of CCB status is set. If an interrupt was generated on that CCB, bit 1 of CCB status is set. If only CCB's and related interrupts are being used, the Input Next CCB Status instruction (function code lA) and Input CCB Status instruction (function code 18) are used.

It becomes difficult for the software when CCB's are used in conjunction with data set scan interrupts or INTR instructions because of the three choices of input status commands available. In practice, rarely would the various causes of interrupts be used together so that it is generally obvious which I/O command to issue in response to an interrupt. Some of the more typical cases will be described.

CCB's as Cause of CPU Interrupts

If CCB's are the only cause of interrupts, the software response to an interrupt is Input Next CCB Status instruction (function code lA). If that status is complete and bit 1 is set, the cause of the interrupt has been detected. If bit 1 is not set, the operation can be repeated until a CCB is found with bit 1 set. This is the most common usage of interrupts. In this case the assumption is that some CCB's were output by the software with I bits and some without (i.e., in a message spanning several blocks an interrupt would not be required on every block). Refer to the previous discussion of the "get next block" cause of interrupt.

One usage of the INTR instruction in conjunction with CCB's is in the case of a receipt of a long message of unknown length. In this case it is not efficient for the main memory program to set I bits in CCB's, not knowing which was to be the end. For this situation, an alternate method is to output CCB's without I bits and use the INTR instruction followed by a GNB to cause an interrupt based on a CCP-detected event (e.g., EOM).

Data Set Scan as Source of CPU Interrupts

If a data set scan is the only source of interrupts, the software response to an interrupt would be to check LCT status. This would be the case, for example, in an auto answer mode of operation.

Combination of CCP Interrupts and INTR in Debug

Another usage is modifying CCP by the addition of INTR instructions to create breakpoints. In this case, at certain points of significance in the CCP, a branch(es) would be added, pointing to an INTR instruction. Following the INTR, an INZ would be used to freeze activity for software analysis via a dump routine. The software response to an interrupt would be as in the case of a CCB causing a CPU interrupt (see previous discussion) except when the Input Next CCB Status indicated an incomplete CCB. That response would be a signal to the software that this interrupt was not CCB-related but INTR-related. A check of LCT Status would then show bit 1 set.

More complex usage of the three types of interrupts are possible. The major point to be made is that *it can be difficult to determine the cause of the interrupt if they are intermixed too freely in one application.*

DEFERRED INTERRUPT QUEUE

When the MLCP has reached an event which requires a CPU interrupt, that interrupt

[3] If data set status changes and bit 2 of LCT byte 8/40 is set, bit l of LCT byte 16/48 will be set. There will be no termination of the current CCB. When the main memory program executes an Input Status (code 18) or Input Next CCB Status (code lA) instruction, the CCB status will be zero. The LCT status (LCT byte 16/48) should then be read; if bit 1 is set, a data set scan interrupt has occurred.

is sent via the megabus regardless of other prior responses of the CPU to interrupts. This particular interrupt may be accepted (ACK) or rejected (NAK) depending on the CPU level at that time. If an interrupt is rejected, that event is noted by the MLCP and the interrupt is retried when the retry interrupt megabus signal occurs. The MLCP maintains a count of the number of deferred interrupts on a per-channel basis. That count can have a value from 0 to 3. When the retry interrupt megabus signal occurs, that event is noted and deferred interrupts are resubmitted by the firmware in background mode. Firmware in background mode scans the channels in turn and sends an interrupt for each channel that has a non-zero count for deferred interrupts. If the interrupt succeeds, the count is decremented by one. Only a single pass through the channel is made for each retry interrupt megabus signal.

The presumption in the MLCP is that the major source of deferred interrupts is CCB's which have completed but for which the CPU has not yet taken the interrupts.

FORBIDDEN MLCP OPERATIONS

Undefined Op Codes

The MLCP includes no protection for undefined op codes. The op codes which are legal are those in Figure 4-2. Others will cause unpredictable operation with unspecified results.

Undefined Function Codes

The MLCP includes no protection for undefined function codes. Only those function codes listed in Table 2-1 are legal. Others will cause unpredictable operations.

Unprotected MLCP Memory

The MLCP contains no protection of its memory other than those inherent in the MLCP addressing scheme. For example:

- o A JUMP instruction has a range sufficient to allow a CCP to jump to itself (an endless loop), or to a CCB or LCT location.
- o A block mode write can access all of the MLCP memory and can therefore overwrite locations used by other channels if care is not taken.
- o An Output LCT Byte instruction can change any LCT byte, many of which should not be modified by software.
- o An ST instruction can change any LCT location, many of which should not be modified by software.

Reserved Firmware Locations

Those LCT locations specified as firmware working locations contain information which may be of interest to some users.[4] *The information within these locations is subject to change in future implementations.* The result is that software which makes use of any of this information may not be transferrable to other communications products which are otherwise compatible with the MLCP.

Timeouts

There is no practical means of performing a timeout in the MLCP unless that timeout can be related to the normal character stream and accomplished by counting channel request interrupts.

It is possible for the main memory program to timeout if activity stops, or to monitor CCP execution. The latter can be done by means of flags set by the CCP in an LCT work area.

[4] See "LCT Bytes Used by Firmware" in Section 5.

ADDRESSING LIMITS

Implicit in the MLCP architecture are certain addressing limits which subject the CCP to definite restrictions.

CCB Area Only Implicitly Accessible

The CCP cannot access the CCB area. All the required functions which are required are embodied in the MLCP instruction set. In particular,

- o LD (Next Character)
- o ST (Next Character)
- o BVBT
- o BVBF
- o BLCT
- o BLCF
- o BLBT
- o BLBF
- o GNB

are the total complement of instructions relating to the CCB area.

Inability of One Line to Access Another

The CCP may access the entire LCT area for a line. In that sense, the two CCP's of a line may communicate with each other via an LCT location. There is, however, no addressing mode which permits a CCP for one line to access another line in any way. Any line-to-line communication must be by way of the main memory program.

NEED FOR PAD CHARACTERS

On the transmit side of the line, the Communications-Pac has an 8-bit register (LR1) which receives the character from the MLCP and an additional 8-bit shift register between LR1 and the line. The shift register is loaded from LR1 and shifted serially out to the line. There is therefore a delay between the loading of LR1 and the time the last bit of that character clears the Communications-Pac and gets physically onto the line. In the Synchronous Communications-Pac that delay can be as much as 2-1/8 characters. In the Asynchronous Communications-Pac, the delay can be a maximum of 2 characters.

In contrast, the setting or clearing of a bit in LR2 causes that function to occur immediately. One can, for example, output the last character of a message to LR1 and then turn off the transmitter before that character clears the Communications-Pac, thereby truncating the message. The bits in LR2 which cause this kind of problem are Communications-Pac-specific:

- o Asynchronous-Communications-Pac
  - Request to Send
  - Transmitter ON
  - Transmit Mark
  - Transmit Space
- o Synchronous-Communications-Pac
  - Request to Send
  - Transmitter ON

In order to avoid problems when making a change in one of the above at the end of a transmission, the general rule is to follow the last character with pad characters. For the Synchronous Communication-Pac, three pad characters are recommended for the Asynchronous Comunications-Pac two pad characters. The pad characters provide sufficient delay before the output to LR2 so that the real end-of-message may get onto the line before the command to LR2 takes effect. An all ones pad character is recommended. The actual number of pad characters which get onto the line will vary depending on the load on the MLCP at the time.

(Other Communications-Pacs tend to have the same general characteristics.)

TWO-WAY ALTERNATE OPERATION

When operating in Two-Way Alternate mode, the transmitter must be turned off and a request to send dropped at the end of transmission to condition the modem for reception of the message from the other end of the line. The use of pad characters is required in this case as discussed in the preceding paragraphs. In addition, another difficulty relates to some of the earlier modems (201, 202) which tie the transmit and receive lines together so that everything that is being transmitted is being received. In the Asynchronous Communications-Pac, holding off the receiver is not sufficient to prevent problems from this source. The reason is that the pac registers are still accumulating characters when the receiver is off although no channel request interrupts are generated. The Asynchronous Communications-Pac is detecting receive overrun status, however, and will report it as soon as the receiver is turned on. This is analagous to the phantom overrun discussed in "Initialization" (earlier in Appendix A) and the solution is the same. After turning off the transmitter but before turning on the receiver, execute an IN LR1 and discard the results. This clears the overrun bit.

Another difficulty is the fact that the received message may include some of the pad characters output at the end of the transmit operation. Defining a SOM character and having the CCP discard everything until that point is one solution. The other solution is to have the CCP check all received characters and discard all pads up to the first non-pad character.

Refer to Appendix C for further detail on the Asynchronous Communications-Pac receive overrun condition.

ERROR HANDLING

The following subjects, all of which relate directly or indirectly to error handling, are described:

- o Conditions under which MLCP will issue a NAK
- o LCT status bytes
- o CCB status bytes
- o Block mode read
- o Abnormal CCP Termination

Conditions Under Which MLCP Will Issue a NAK

Under the following conditions, the MLCP will issue a NAK in response to an input/output instruction from the main memory program:

- o An input/output instruction is issued before MLCP initialization the result of a recent IO (Output MLCP Control) instruction is complete.[5]
- o An IO (Output CCB Control) instruction has moved the "load" CCB pointer to the CCB one beyond the "status" CCB and an IOLD (Output CCB Address and Range) instruction is attempted before an IO (Input Next CCB Status) instruction is executed. (The MLCP will issue a NAK in response to the IOLD (Output CCB Address and Range) instruction.)
- o After CCB list initialization, an IO (Input Next CCB Status) instruction is attempted before the first CCB is set up.
- o An IO (Input Next CCB Status) instruction has moved the "status" CCB pointer to the CCB one behind the "load" CCB and another IO (Input Next CCB Status) instruction is attempted before the "load" CCB is set up.

[5] Exception: If a *second* IO (Output MLCP Control) instruction is issued to initialize the MLCP while initialization is still in progress as the result of a recent such instruction, the MLCP will not issue a NAK; instead, execution of the first instruction is terminated and execution of the second instruction begins.

After the MLCP issues a NAK, processing in the main memory program continues with the next sequential instruction. The main memory program can use a BIOF (Branch if Input/Output Indicator False) instruction to branch back to the input/output instruction that caused the NAK. However, only a limited number of attempts should be made to reissue this input/output instruction since a closed (two-instruction) loop between the input/output instruction and a BICF instruction would be endless if a NAK were always returned.

LCT Status Bytes

While a given CCB is active, MLCP firmware uses two bytes of the related channel's LCT to accumulate certain status and error information. LCT bytes 16 and 17 are used for a receive channel and LCT bytes 48 and 49 are used for a transmit channel. The format of these bytes is shown in the diagram below. A detailed description of each significant bit position appears under "LCT Bytes 16/48 and 17/49 LCT Status" in Section 5.

	0	1	2	3	4	5	6	7
LCT BYTE 16/48 STATUS BYTE 1	0 SEE NOTE 1	0 SEE NOTE 2	DATA SERVICE ERROR	0	CCB SERVICE ERROR	FOR PROGRAMMING USE	FOR PROGRAMMING USE	0
LCT BYTE 17/49 STATUS BYTE 2	0	DATA CHECK ERROR	0	DATA SET OR COMMUNICATIONS-PAC STATUS CHANGE	CORRECTED MEMORY ERROR	0	0	0

NOTES: 1. This bit is set by the firmware when the CCP issues the INTR instruction.
2. This bit is set when the data set status change causes an interrupt.

MLCP firmware can update the LCT status bytes throughout the time a given CCB is active. (The MLCP firmware sets a bit when the related condition is detected; while the CCB is active. MLCP firmware never resets to 0 any bit of the LCT status bytes.) As soon as processing ends relative to the given CCB, the contents of the LCT status bytes are combined with other information and transferred to the CCB status field (described in the following subsection). The contents of LCT byte 16/48 (LCT status byte 1) are among the information moved to byte 7 of the CCB; the contents of LCT byte 17/49 (LCT status byte 2) are among the information moved to byte 6 of the CCB. Immediately after the LCT status bytes are used to update the CCB status field, they are reset to zero, pending activation of the next CCB for the same channel.

While a given CCB is still active, the LCT status bytes can be read by the main memory program through the use of the IO (Input LCT Byte) instruction. During this time, the CCP can read the LCT status bytes through use of format 2 LD (Load) instructions. The CCP can manipulate bits 5 and 6 of LCT byte 16/48 for application-specific purposes; this technique can be used for passing information from the CCP to the main memory program (which can later read these bit positions from the CCB status field); these bit positions are shown shaded in the diagram.

CCB Status Bytes

As soon as processing ends relative to a CCB, its status field (CCB bytes 6 and 7) is updated by MLCP firmware and is said to be "meaningful." The CCB status bytes are written with information transferred from the LCT status bytes combined with other information. Bit 3 of CCB byte 7 is set to 1 to indicate that the CCB status bytes are meaningful. Bit 0 of CCB byte 7 will be set to 1 if the CCP issued the INTR instruction. (Refer to the LCT byte 16/48 diagram above.)

The CCB's status field can be read from the main memory program by means of an IO (Input CCB Status) or IO (Input Next CCB Status) instruction.

The format of the CCB status bytes is shown below. Those bit positions that are written with information from the LCT status bytes are shown shaded. Note that CCB status byte 1 is stored *above* CCB status byte 2; this order is the opposite of the order of the status bytes in the LCT. A detailed description of each significant bit position appears under "CCB Status Field" in Section 3.

	0	1	2	3	4	5	6	7
CCB BYTE 6 STATUS BYTE 2	0	DATA CHECK ERROR	RECV CCB NONZERO RANGE RESIDUE XMIT LAST BLOCK	DATA SET OR COMMUNICATIONS-PAC STATUS CHANGE	CORRECTED MEMORY ERROR	INVALID MEMORY ADDRESS	MEGABUS PARITY ERROR	UNCORRECTED MEMORY ERROR
CCB BYTE 7 STATUS BYTE 1	0 SEE NOTE	INTERRUPT MAIN MEMORY PROGRAM	DATA SERVICE ERROR	CCB STATUS COMPLETE	CCB SERVICE ERROR	FOR PROGRAMMING USE	FOR PROGRAMMING USE	0

NOTE: This bit is set by the firmware when the CCP issues the INTR instruction. (Refer to the LCT byte 16/48 description above.)

Block Mode Read

A block mode read is a convenient means of reading any portion of MLCP RAM into the main memory program. The block mode read is particularly useful as a debugging tool.

For more information relative to the block mode read, see "Block Mode Write" in Section 7. A block mode read is the direct counterpart of a block mode write except for the following differences:

- o A block mode read is performed over a *receive* channel.
- o A different bit is set in the control word used with the IO (Output Channel Control) instruction. See Section 2.
- o There are no restrictions on the MLCP RAM area that can be included in a block mode read.

The following diagram can be consulted when you are deciding the starting RAM address and range for a block mode read (or a block mode write). The RAM address is a hexadecimal byte address.

CHANNEL NUMBER	RAM ADDRESS	00000000000000001111111111111111 0123456789ABCDEF0123456789ABCDEF				Add to H3 and H4 of RAM Address
0	0000	LCT Bytes				
1	0020	LCT Bytes				
2	0040	LCT Bytes				
3	0060	LCT Bytes				
4	0080	LCT Bytes				
5	00A0	LCT Bytes				
6	00C0	LCT Bytes				
7	00E0	LCT Bytes				LCT Area
8	0100	LCT Bytes				
9	0120	LCT Bytes				
10	0140	LCT Bytes				
11	0160	LCT Bytes				
12	0180	LCT Bytes				
13	01A0	LCT Bytes				
14	01C0	LCT Bytes				
15	01E0	LCT Bytes				
	0200 ↓ 0DFF					CCP Area
0	0E00	CCB 0	CCB 1	CCB 2	CCB 3	
1	0E20	CCB 0	CCB 1	CCB 2	CCB 3	
2	0E40	CCB 0	CCB 1	CCB 2	CCB 3	
3	0E60	CCB 0	CCB 1	CCB 2	CCB 3	
4	0E80	CCB 0	CCB 1	CCB 2	CCB 3	
5	0EA0	CCB 0	CCB 1	CCB 2	CCB 3	
6	0EC0	CCB 0	CCB 1	CCB 2	CCB 3	
7	0EE0	CCB 0	CCB 1	CCB 2	CCB 3	CCB Area
8	0F00	CCB 0	CCB 1	CCB 2	CCB 3	
9	0F20	CCB 0	CCB 1	CCB 2	CCB 3	
10	0F40	CCB 0	CCB 1	CCB 2	CCB 3	
11	0F60	CCB 0	CCB 1	CCB 2	CCB 3	
12	0F80	CCB 0	CCB 1	CCB 2	CCB 3	
13	0FA0	CCB 0	CCB 1	CCB 2	CCB 3	
14	0FC0	CCB 0	CCB 1	CCB 2	CCB 3	
15	0FE0	CCB 0	CCB 1	CCB 2	CCB 3	

NOTE: The status of the CCB used by the block mode read operation is not valid (contains snapshot address of the MLCP RAM).

Abnormal CCP Termination

Error recovery of any abnormal CCP termination should include a CCB list reset or channel initialize to both channels of a line pair.

APPENDIX B

COMMUNICATIONS-PACS ATTACHABLE TO MLCP

MLCP-related main memory programs and communications control programs (CCP's) must be developed in a hardware configuration that supports GCOS/BES, GCOS/BES2, or GCOS 6/MDT program development.

Communications processing requires a 6/30 or 6/40 Model central processor, appropriate peripheral equipment, and local communications equipment selected from the list shown in Table B-1.

TABLE B-1. COMMUNICATIONS-PACS ATTACHABLE TO MLCP

Type Number	Description
DCM9101	Asynchronous Communications-Pac, two lines with 30-foot (9.1-m) data set cables (19.2K bps maximum)
DCM9102	Asynchronous Communications-Pac, one line with 30-foot (9.1-m) data set cables (19.2K bps maximum)
DCM9103	Synchronous Communications-Pac, two lines with 30-foot (9.1-m) data set cables (19.2K bps maximum)
DCM9104	Synchronous Communications-Pac, one line with 30-foot (9.1-m) data set cable (19.2K bps maximum)
DCM9105	Synchronous Broadband Communications-Pac, one line with 30-foot (9.1-m) data set cable, Bell 301/303-compatible (72K bps maximum)
DCM9106	Synchronous HDLC Communications-Pac, one line with 30-foot (9.1-m) data set cable (10.8K bps maximum)
DCM9108	Synchronous Broadband Communications-Pac, one line with 30-foot (9.1-m) data set cable, CCITT-V35-compatible (72K bps maximum)
DCM9109	Synchronous Communications-Pac, one line, with 30-foot (9.1-m) data set cable, MIL188C-compatible (10.8K bps maximum)
DCM9110	Communications-Pac, Auto Call Feature, with 30-foot (9.1-m) cable to attach to two Auto Call Units
DCM9111	Communications-Pac, one line for current loop connection, with 30-foot (9.1-m) cable
DCM9114	Communications-Pac, two lines for current loop connection, with 30-foot (9.1-cm) cables
DCF6927	Universal Modem bypass, one line (4-wire cable), capable of asynchronous or synchronous operation over a distance of 2,500 feet, can be connected to any terminal with an EIA RS-232-C interface and can operate at bit rates up to 19,200 bps. Refer to Appendixes C and D of this manual for programming this device.
W18-0001C	1-foot (30.5-cm) direct connect cable, synchronous or asynchronous

APPENDIX C

ASYNCHRONOUS LINE COMMUNICATIONS-PACS

An Asynchronous Line Communications-Pac (Type DCM9101 or DCM9102) provides an interface between the MLCP and one or two completely independent asynchronous communications lines.[1] For each line, the Asynchronous Line Communications-Pac provides the following services:

- o Serial/parallel data conversion for asynchronous bit-serial data transfers
- o Character synchronization by use of framing bits
- o Control of data sets
- o Monitoring of data set status

Each communications line comprises a receive channel and a transmit channel and is thus capable of half-duplex or full-duplex data communications operations. Each line has an independently configurable speed (up to 19,200 bits per second) and an independently configurable data character size (from 5 to 8 bits with no parity, or 6 to 8 bits including parity); each channel of a line uses the configured line speed and data character size. The Asynchronous Line Communications-Pac supports Basic Mode ASCII and similarly formatted control procedures.

The following data communications equipment and data terminal equipment is supported through an EIA RS232C interface:

- o Bell System 103 or equivalent
- o Bell System 113 or equivalent
- o Bell System 202 or equivalent

Figure C-1 illustrates the Asynchronous Line Communications-Pac's interface position between the MLCP and asynchronous communications lines. (Note that the MLCP can connect any combination of Asynchronous Line Communications-Pacs and Synchronous Line Communications-Pacs up to a total of four.)

LINE REGISTERS

The programming interface to the Asynchronous Line Communications-Pac is achieved through its line registers. These line registers are illustrated in Figure C-2.

As indicated in Figure C-2, each communications line is serviced by a different set of line registers. Each channel of a line has a dedicated set of registers and also shares four registers with the other channel of the same line.

Before data transfer operations can begin over a channel, the CCP must load line registers 6, 4, and 2 using OUT (Output) instructions. Line register 2 must be loaded last.

- o Line register 6 is loaded with data character configuration information. This information is obtained from LCT byte 2/34.

[1] Throughout this appendix, descriptions are based on an Asynchronous Line Communications-Pac that services *two* lines. The *single*-line version of the Asynchronous Line Communications-Pac is identical except for the number of lines it services. Note that reverse channels are per the 202C interface.

- Line register 4 must be loaded with the line speed. Normally, line register 4 is loaded with information obtained from a byte in the LCT programming work area (must be loaded by Receive channel).
- Line register 2 is loaded with data set control and Communications-Pac control information. This information is obtained from LCT byte 20.

*

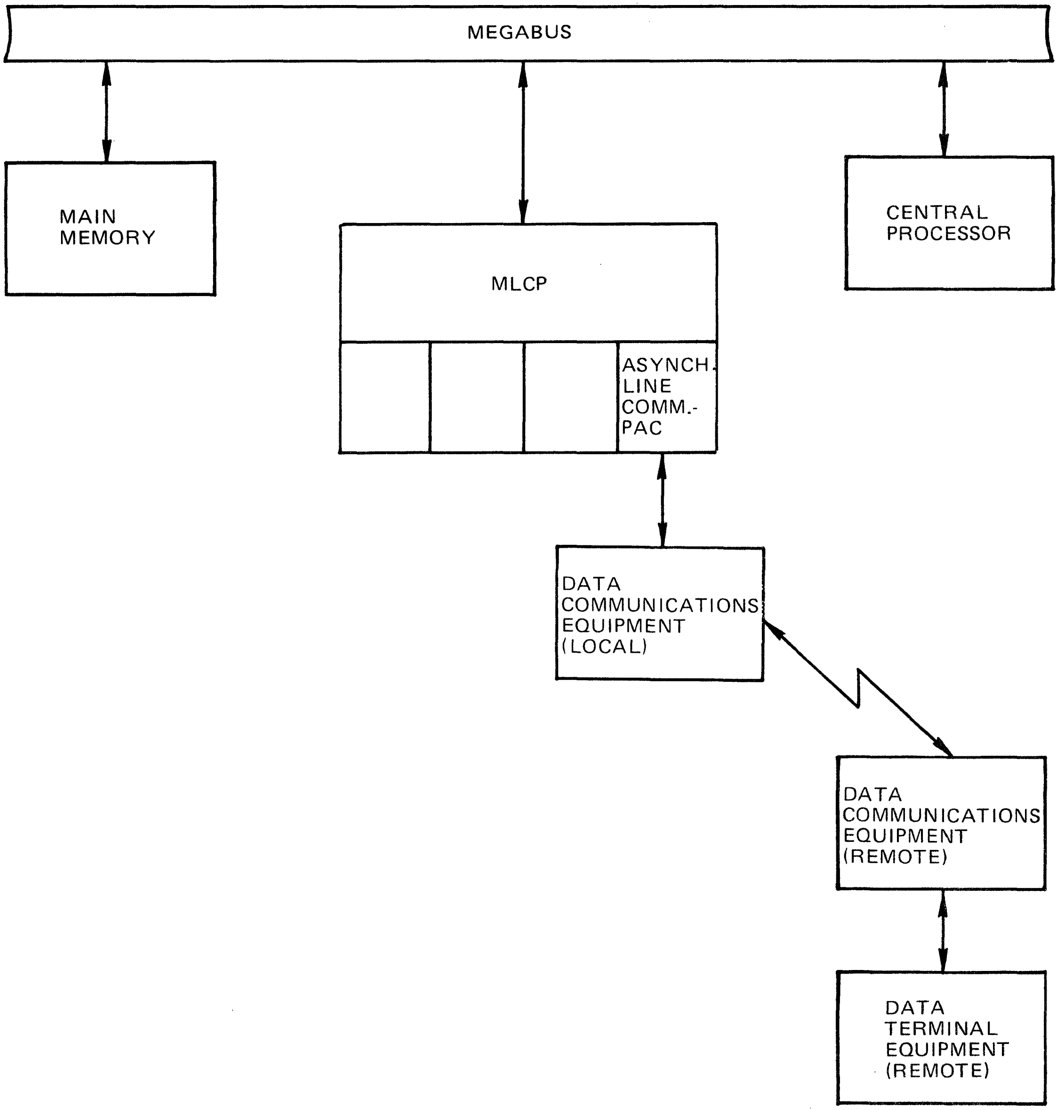

MEGABUS
MAIN MEMORY
MLCP
CENTRAL PROCESSOR
ASYNCH. LINE COMM.-PAC
DATA COMMUNICATIONS EQUIPMENT (LOCAL)
DATA COMMUNICATIONS EQUIPMENT (REMOTE)
DATA TERMINAL EQUIPMENT (REMOTE)

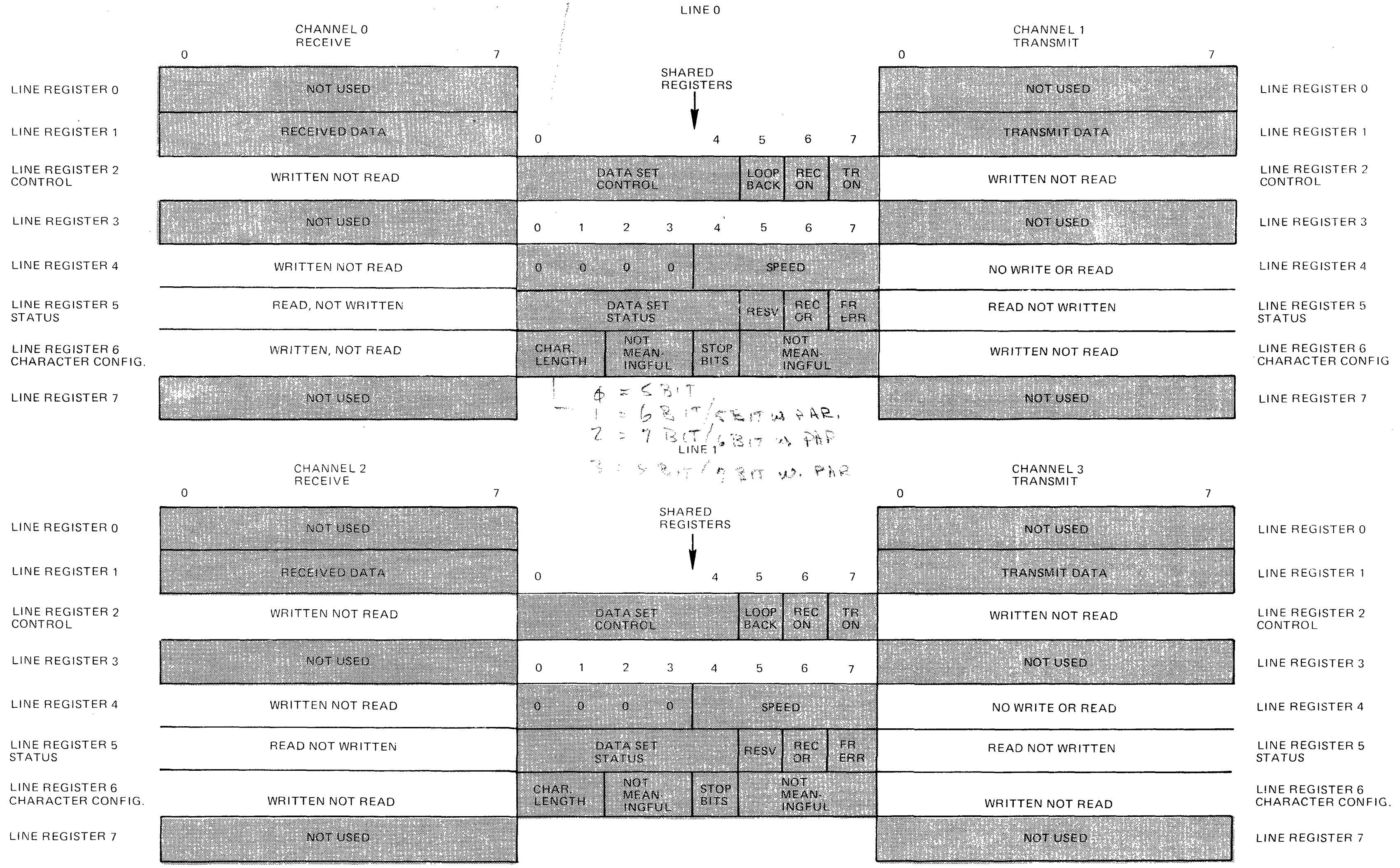
Figure C-1. Interface Provided by Asynchronous Line Communications-Pac

Once data transfer operations have been enabled, a receive CCP uses an RECV (Receive) or IN LR1 instruction to obtain a data character from a receive channel's line register 1 following a channel request interrupt for that channel. Similarly, a transmit CCP uses a SEND (Send) or OUT LR1 instruction to load a data character into a transmit channel's line register 1 following a channel request interrupt for *that* channel.

During processing, a CCP may use IN (Input) instructions to read the contents of individual line registers. OUT (Output) instructions may be used to modify the contents of line registers during processing, but care must be taken to ensure that any such modification of line register contents does not disrupt processing.

The main memory program can read the contents of line register 5 by executing an IO (Input Data Set Status) instruction. The main memory program cannot directly read the contents of any other line register nor can it directly modify the contents of *any* line register.

BIT Φ = DTR
1 RTS



LINE 0

SHARED REGISTERS

	CHANNEL 0 RECEIVE (0–7)	0	1	2	3	4	5	6	7	CHANNEL 1 TRANSMIT (0–7)	
LINE REGISTER 0	NOT USED									NOT USED	LINE REGISTER 0
LINE REGISTER 1	RECEIVED DATA									TRANSMIT DATA	LINE REGISTER 1
LINE REGISTER 2 CONTROL	WRITTEN NOT READ	DATA SET CONTROL					LOOP BACK	REC ON	TR ON	WRITTEN NOT READ	LINE REGISTER 2 CONTROL
LINE REGISTER 3	NOT USED									NOT USED	LINE REGISTER 3
LINE REGISTER 4	WRITTEN NOT READ	0	0	0	0	SPEED				NO WRITE OR READ	LINE REGISTER 4
LINE REGISTER 5 STATUS	READ, NOT WRITTEN	DATA SET STATUS					RESV	REC OR	FR ERR	READ NOT WRITTEN	LINE REGISTER 5 STATUS
LINE REGISTER 6 CHARACTER CONFIG.	WRITTEN, NOT READ	CHAR. LENGTH		NOT MEANINGFUL		STOP BITS	NOT MEANINGFUL			WRITTEN NOT READ	LINE REGISTER 6 CHARACTER CONFIG
LINE REGISTER 7	NOT USED									NOT USED	LINE REGISTER 7

Φ = 5 BIT
1 = 6 BIT/5 BIT w PAR.
2 = 7 BIT/6 BIT w PAR
3 = 8 BIT/7 BIT w. PAR

LINE 1

SHARED REGISTERS

	CHANNEL 2 RECEIVE (0–7)	0	1	2	3	4	5	6	7	CHANNEL 3 TRANSMIT (0–7)	
LINE REGISTER 0	NOT USED									NOT USED	LINE REGISTER 0
LINE REGISTER 1	RECEIVED DATA									TRANSMIT DATA	LINE REGISTER 1
LINE REGISTER 2 CONTROL	WRITTEN NOT READ	DATA SET CONTROL					LOOP BACK	REC ON	TR ON	WRITTEN NOT READ	LINE REGISTER 2 CONTROL
LINE REGISTER 3	NOT USED									NOT USED	LINE REGISTER 3
LINE REGISTER 4	WRITTEN NOT READ	0	0	0	0	SPEED				NO WRITE OR READ	LINE REGISTER 4
LINE REGISTER 5 STATUS	READ NOT WRITTEN	DATA SET STATUS					RESV	REC OR	FR ERR	READ NOT WRITTEN	LINE REGISTER 5 STATUS
LINE REGISTER 6 CHARACTER CONFIG.	WRITTEN NOT READ	CHAR. LENGTH		NOT MEANINGFUL		STOP BITS	NOT MEANINGFUL			WRITTEN NOT READ	LINE REGISTER 6 CHARACTER CONFIG.
LINE REGISTER 7	NOT USED									NOT USED	LINE REGISTER 7

Figure C-2. Registers of Asynchronous Line Communications-Pac

By appropriate settings of bits in LCT bytes 8/40 and 15/47, you can cause MLCP firmware to (1) scan for data set or Communications-Pac status changes reflected in line register 5 and (2) take related action(s) as directed by LCT bytes 8/40 and 15/47.

The following subsections provide more detailed information about the individual line registers.

Line Registers for Receive Channel

As shown in Figure C-2, each receive channel has eight line registers, four of which are shared with the transmit channel of the same line.

	0	7
LINE REGISTER 1	RECEIVED DATA	

The length of each data character in line register 1 is governed by the settings of bits 0 and 1 of line register 6. If the length is fewer than eight bits, the data character is right-justified and the leftmost bits are zero-filled. In all cases, bit 7 is the first bit received over the channel.

Line Register 2–Receive/Transmit Channel

	0	1	2	3	4	5	6	7
	DATA SET CONTROL					COMMUNICATIONS-PAC CONTROL		
LINE REGISTER 2	DATA TERMINAL READY B	REQUEST TO SEND B	SECONDARY CHANNEL TRANSMIT B	TRANSMIT SPACE T	TRANSMIT MARK T	LOOP-BACK TEST B	RECEIVE ON R	TRANSMIT ON T

Line register 2 is shared by the receive channel and the transmit channel of the same line. Its contents affect the operations of both channels. If either channel's CCP modifies the contents of line register 2, both channels' operations are affected. (In the diagram, the letter B, R, or T in the lower right-hand corner of each bit position indicates whether the bit pertains to *both* receive and transmit operations, to *receive* operations only, or to *transmit* operations only.)

Line register 2 is loaded by the CCP with information obtained from LCT byte 20. * The significance of each bit position is described under "LCT Byte 20–Data Set and Communications-Pac Control" in Section 5.

Line Register 4–Receive/Transmit Channel

	0	1	2	3	4	5	6	7
LINE REGISTER 4	0	0	0	0	SPEED			

Line register 4 is shared by the receive channel and the transmit channel of the same line. Its contents affect the operations of both channels.

Line register 4 can be loaded only by the receive channel's CCP. This register is normally loaded with information obtained from a byte of the LCT programming work area for the receive channel.

Table C-1 indicates the line speeds achieved by the settings of bits 4 through 7 of line register 4.

TABLE C-1. CONFIGURABLE LINE SPEEDS FOR ASYNCHRONOUS LINE COMMUNICATIONS-PAC

	Line Speed (Bits per Second)	
Settings for Bits 4-7 of Line Register 4	Device Type 2108	Device Type 2118
0000	No input/output	50
0001	50	75
0010	75	110
0011	110	134.5
0100	134.5	150
0101	150	200
0110	300	300
0111	600	600
1000	900	1,050
1001	1,200	1,200
1010	1,800	1,800
1011	2,400	2,000
1100	3,600	2,400
1101	4,800	4,800
1110	7,200	9,600
1111	9,600	19,200

Line Register 5—Receive/Transmit Channel

	0	1	2	3	4	5	6	7
	DATA SET STATUS					COMMUNICATIONS-PAC STATUS		
LINE REGISTER 5	DATA SET READY	CLEAR TO SEND	CARRIER DETECTOR	RING INDICATOR	SECONDARY CHANNEL RECEIVE	RESERVED	RECEIVE OVERRUN	FRAMING ERROR

Line register 5 is shared by the receive channel and the transmit channel of the same line. Its contents reflect data set status and Communications-Pac status. Line register 5 can be read by the main memory program through use of an IO (Input Data Set Status) instruction.

If bit 0 of LCT byte 8 (or LCT byte 40) is set to 1, the entire contents of line register 5 will be written to LCT byte 14 (or LCT byte 46) whenever a data set or Communications-Pac status change occurs (*provided* that, in LCT byte 15 (or LCT byte 47), there is a 1 in the bit position that corresponds to the status change). Subsequent action(s) depend on the settings of bits 1, 2, and 3 of LCT byte 8 (or LCT byte 40).

The significance of each bit position of line register 5 is described under "LCT Byte 14/46—Data Set and Communications-Pac Status" in Section 5.

Line Register 6—Receive/Transmit Channel

	0	1	2	3	4	5	6	7
LINE REGISTER 6	CHARACTER LENGTH		NOT MEANINGFUL		STOP BITS	NOT MEANINGFUL		

Line register 6 is shared by the receive channel and the transmit channel of the same line. Its contents affect the operations of both channels. If either channel's CCP modifies the contents of line register 6, both channels' operations are affected.

Line register 6 is loaded by the CCP with information obtained from LCT byte 2 (or LCT byte 34). (Note that the parity and cyclic redundancy check information in LCT byte 2/34 is not meaningful to the Asynchronous Line Communications-Pac.) The significance of each bit position is described under "LCT Byte 2/34–Character Configuration" in Section 5.

Line Registers for Transmit Channel

As shown in Figure C-2, each transmit channel has eight line registers, four of which are shared with the receive channel of the same line.

Line Register 1–Transmit Channel

	0 — 7
LINE REGISTER 1	TRANSMIT DATA

The length of each data character in line register 1 is governed by the settings of bits 0 and 1 of line register 6. If the length is fewer than eight bits, the leading bits must be specified as zeros in order to generate parity correctly (when the data character is transferred to line register 1); in this case, the data character is right-justified in line register 1 and the leading zeros are not included in transmissions.

If parity is to be generated, the parity bit (the leftmost bit of the defined character length) must be specified as a zero (when the data character is transferred to line register 1); the MLCP generates the correct parity during the transfer to line register 1. The parity bit is included in transmissions.

In all cases, bit 7 is the first bit of the data character to be transmitted over the channel.

Line Register 2–Receive/Transmit Channel

	0	1	2	3	4	5	6	7
	DATA SET CONTROL					COMMUNICATIONS-PAC CONTROL		
LINE REGISTER 2	DATA TERMINAL READY B	REQUEST TO SEND B	SECONDARY CHANNEL TRANSMIT B	TRANSMIT SPACE T	TRANSMIT MARK T	LOOP-BACK TEST B	RECEIVE ON R	TRANSMIT ON T

Line register 2 is shared by the transmit channel and the receive channel of the same line. Its contents affect the operations of both channels. If either channel's CCP modifies the contents of line register 2, both channels' operations are affected. (In the diagram, the letter B, R, or T in the lower right-hand corner of each bit position indicates whether the bit pertains to *both* receive and transmit operations, to *receive* operations only, or to *transmit* operations only.)

Line register 2 is loaded by the CCP with information obtained from LCT byte 20. The significance of each bit position is described under "LCT Byte 20–Data Set and Communications-Pac Control" in Section 5.

Line Register 4–Receive/Transmit Channel

	0	1	2	3	4	5	6	7
LINE REGISTER 4	0	0	0	0	SPEED			

Line register 4 is shared by the transmit channel and the receive channel of the same line. Its contents affect the operations of both channels.

Line register 4 can be loaded only by the receive channel's CCP. See the description of line register 4 under "Line Registers for Receive Channel," earlier in this section.

Line Register 5–Receive/Transmit Channel

	0	1	2	3	4	5	6	7
	DATA SET STATUS					COMMUNICATIONS-PAC STATUS		
LINE REGISTER 5	DATA SET READY	CLEAR TO SEND	CARRIER DETECTOR	RING INDICATOR	SECONDARY CHANNEL RECEIVE	RESERVED	RECEIVE OVERRUN	FRAMING ERROR

Line register 5 is shared by the transmit channel and the receive channel of the same line. Its contents reflect data set status and Communications-Pac status. Line register 5 can be read by the main memory program through use of an IO (Input Data Set Status) instruction.

If bit 0 of LCT byte 40 (or LCT byte 8) is set to 1, the entire contents of line register 5 will be written to LCT byte 46 (or LCT byte 14) whenever a data set or Communications-Pac status change occurs (*provided* that, in LCT byte 47 (or LCT byte 15), there is a 1 in the bit position that corresponds to the status change). Subsequent action(s) depend on the settings of bits 1, 2, and 3 of LCT byte 40 (or LCT byte 8).

The significance of each bit position of line register 5 is described under "LCT Byte 14/46–Data Set and Communications-Pac Status" in Section 5.

Line Register 6–Receive/Transmit Channel

	0	1	2	3	4	5	6	7
LINE REGISTER 6	CHARACTER LENGTH		NOT MEANINGFUL		STOP BITS	NOT MEANINGFUL		

Line register 6 is shared by the transmit channel and the receive channel of the same line. Its contents affect the operations of both channels. If either channel's CCP modifies the contents of line register 6, both channels' operations are affected.

Line register 6 is loaded by the CCP with information obtained from LCT byte 34 (or LCT byte 2). (Note that the parity and cyclic redundancy check information in LCT byte 2/34 is not meaningful to the Asynchronous Line Communications-Pac.) The significance of each bit position is described under "LCT Byte 2/34–Character Configuration" in Section 5.

PROGRAMMING CONSIDERATIONS

The paragraphs that follow present detailed programming requirements for the Asynchronous Communications-Pac.

Data Transfers

Within the Asynchronous Line Communications-Pac, each data transfer is channel-specific and involves one (of the four) line register 1 and a "shift" buffer associated exclusively with that specific line register 1. The shift buffer is a hardware register controlled internally by the Communications-Pac hardware.

During receive operations, bits of a data character are serially received in a shift buffer.[2] After reception of an entire character (of the length specified by bits 0 and 1 of line register 6), the accumulated contents of the shift buffer are transferred to the receive channel's line register 1 and a channel request interrupt is generated. (As soon as the shift buffer's contents have been transferred to line register 1, the buffer is available for the reception of the first bit of the next incoming character.) When the receive channel's CCP is started in response to the channel request interrupt, it can issue an RECV (Receive) or an IN LR1 instruction to transfer the data character from line register 1 into the MLCP's R-register.

During transmit operations, the Asynchronous Line Communications-Pac generates a channel request interrupt as soon as the previous contents of the transmit channel's line register 1 have been transferred to the associated shift buffer for serial transmission. When the transmit channel's CCP is started in response to the channel request interrupt, it can load the MLCP's R-register with a data character and then issue a SEND (Send) instruction to transfer that character from the R-register to line register 1. The data character will remain in line register 1 until each bit of the preceding data character has been serially transmitted from the associated shift buffer. (The number of bits transmitted is governed by the character length specified in bits 0 and 1 of line register 6.) When the preceding data character has been transmitted, the data character in line register 1 is transferred to the associated shift buffer and a channel request interrupt is generated.

NOTE: Two pad characters are required before resetting Request to Send or Transmit ON.

Channel Request Interrupts

The Asynchronous Line Communications-Pac generates a channel request interrupt (on a channel-specific basis) whenever a data transfer between the MLCP and the Communications-Pac is possible. For a receive channel, the channel request interrupt occurs when the receive channel's line register 1 has been loaded with an incoming data character. For a transmit channel, the channel request interrupt occurs when the transmit channel's line register 1 is "empty" and can be loaded with the next data character for transmission.[3]

Table 1-2 indicates the MLCP's priorities for servicing channel request interrupts. A CCP will not be required to service successive channel request interrupts faster than the character rate for that channel.

Channel request interrupts will not be enabled for a receive channel unless bit 6 (receive ON) is set to 1 in line register 2. Channel request interrupts will not be enabled for a transmit channel unless bit 1 (request to send) and bit 7 (transmit ON) of line register 2 are set to 1; in addition, bit 1 (clear to send) of line register 5 must be set to 1 by the Communications-Pac.

[2] The occurrence of a "start bit" on the communications channel initiates reception of the bits of a data character. The last bit of the data character is followed by one or more "stop bits." The start bit and the stop bit(s) (known as framing bits) are not received in the shift buffer.

[3] If a data character is not provided in response to a channel request interrupt, the transmit channel will hold data output in a mark (logical 1) condition when it is time to transmit the data character not provided.

Transmit Space/Mark

A transmit channel can be directed to hold data output in a space (logical 0) or mark (logical 1) condition if bit 3 (transmit space) or bit 4 (transmit mark) of line register 2 is set to 1. (These conditions are mutually exclusive.) For additional information, see "LCT Byte 20–Data Set and Communications-Pac Control" in Section 5. *

NOTES: 1. Default is the transmit mark and the CCP need not set it.
2. Whenever a transition occurs between normal transmission and mark or space, it must be preceded by two null characters or truncation will occur.

"Loop-Back" Test

A "loop-back" test is performed if bit 5 (loop-back test) of line register 2 is set to 1. (In addition, bit 1 (request to send) and bit 7 (transmit ON) of line register 2 must be set to 1.) In this case, data characters sent to the transmit channel's line register 1 will be "looped back" to the receive channel's line register 1 without being transmitted. The transmit channel will be held in a mark (logical 1) condition during a "loop-back" test and no external data will be received.

The "loop-back" test is possible regardless of whether the communications line is connected to an Asynchronous Line Communications-Pac by means of data communications equipment or whether the communications line is direct-connected to the Asynchronous Line Communications-Pac. The transfer occurs at the line speed indicated by the settings of bits 4 through 7 of line register 4 for the line.

Receive/Transmit ON

Channel request interrupts are not enabled for a *receive* channel unless bit 6 (receive ON) of line register 2 is set to 1. Channel request interrupts are not enabled for a *transmit* channel unless bit 7 (transmit ON) of line register 2 is set to 1.[4]

Line Speed

The speed of each line of an Asynchronous Line Communications-Pac is governed by the settings of bits 4 through 7 of that line's line register 4. Each channel of the line operates at the indicated speed. This speed is used in all situations regarding the line:

- Line connected to Asynchronous Line Communications-Pac by means of data communications equipment
- Line direct-connected to Asynchronous Line Communications-Pac
- Line operating in "loop-back" test mode

Clear to Send

Bit 1 (clear to send) of line register 5 must be set to 1 in order for channel request interrupts to be enabled for a transmit channel.[5]

Receive Overrun

Because of the nature of the asynchronous line adapter, it is capable of receiving even when the receiver is off. There is a danger that any time the receiver is turned on there may be a character or a character and a receive overrun error already stored from a previous operation. A channel initialize does not clear these conditions, but the MLCP general initialize will.

A receive overrun occurs when a data character in a receive channel's line register 1 is overwritten with another incoming data character because the CCP did not respond quickly enough to the channel request interrupt generated for the overwritten data

[4] In, addition, for a transmit channel, bit 1 (request to send) of line register 2 must be set to 1. Moreover, bit 1 (clear to send) of line register 5 must be set to 1 by the Communications-Pac.

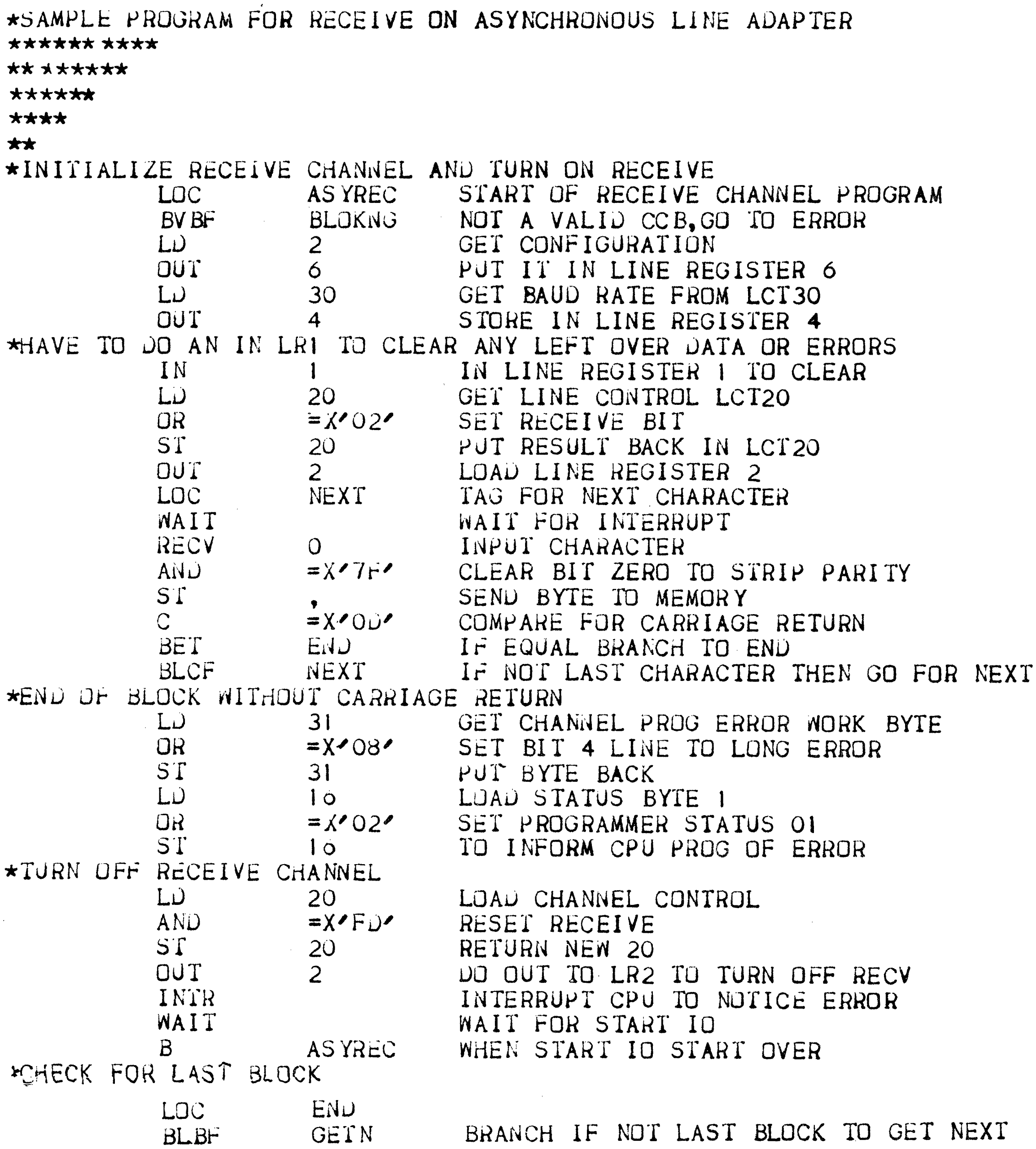
[5] In, addition, bit 1 (request to send) and bit 7 (transmit ON) of line register 2 must be set to 1.

character. A receive overrun condition causes bit 6 (receive overrun) of line register 5 to be set to 1. This bit is reset to 0 (by the Communications-Pac) when the CCP next receives a data character before it is overwritten.

The receive overrun condition also causes bit 2 (data service error) of LCT byte 16 to be set to 1 whenever a RECV instruction is executed. When the contents of LCT bytes 16 to 17 are transferred to the CCB status bytes at the end of processing relative to a CCB, this bit setting permits end-of-block processing to detect and respond to the receive overrun condition.

Refer to the sample segement of a channel program shown in Figure C-3.

This is the correct sequence to flush the receiver. The IN LR1 transfers the leftover character from the adapter, then the receiver is turned on to allow interrupts and the WAIT turns off the channel program until a new character is received. The new character clears the overrun error and gives an interrupt, and the channel program resumes processing in the normal manner.



```
*SAMPLE PROGRAM FOR RECEIVE ON ASYNCHRONOUS LINE ADAPTER
****** ****
** ******
******
****
**
*INITIALIZE RECEIVE CHANNEL AND TURN ON RECEIVE
          LOC       ASYREC    START OF RECEIVE CHANNEL PROGRAM
          BVBF      BLOKNG    NOT A VALID CCB,GO TO ERROR
          LD        2         GET CONFIGURATION
          OUT       6         PUT IT IN LINE REGISTER 6
          LD        30        GET BAUD RATE FROM LCT30
          OUT       4         STORE IN LINE REGISTER 4
*HAVE TO DO AN IN LR1 TO CLEAR ANY LEFT OVER DATA OR ERRORS
          IN        1         IN LINE REGISTER 1 TO CLEAR
          LD        20        GET LINE CONTROL LCT20
          OR        =X'02'    SET RECEIVE BIT
          ST        20        PUT RESULT BACK IN LCT20
          OUT       2         LOAD LINE REGISTER 2
          LOC       NEXT      TAG FOR NEXT CHARACTER
          WAIT                WAIT FOR INTERRUPT
          RECV      0         INPUT CHARACTER
          AND       =X'7F'    CLEAR BIT ZERO TO STRIP PARITY
          ST        ,         SEND BYTE TO MEMORY
          C         =X'0D'    COMPARE FOR CARRIAGE RETURN
          BET       END       IF EQUAL BRANCH TO END
          BLCF      NEXT      IF NOT LAST CHARACTER THEN GO FOR NEXT
*END OF BLOCK WITHOUT CARRIAGE RETURN
          LD        31        GET CHANNEL PROG ERROR WORK BYTE
          OR        =X'08'    SET BIT 4 LINE TO LONG ERROR
          ST        31        PUT BYTE BACK
          LD        16        LOAD STATUS BYTE 1
          OR        =X'02'    SET PROGRAMMER STATUS 01
          ST        16        TO INFORM CPU PROG OF ERROR
*TURN OFF RECEIVE CHANNEL
          LD        20        LOAD CHANNEL CONTROL
          AND       =X'FD'    RESET RECEIVE
          ST        20        RETURN NEW 20
          OUT       2         DO OUT TO LR2 TO TURN OFF RECV
          INTR                INTERRUPT CPU TO NOTICE ERROR
          WAIT                WAIT FOR START IO
          B         ASYREC    WHEN START IO START OVER
*CHECK FOR LAST BLOCK

          LOC       END
          BLBF      GETN      BRANCH IF NOT LAST BLOCK TO GET NEXT
```

Figure C-3. Sample Program for Receive on Asynchronous Communications-Pac

```
*THIS WAS THE LAST BLOCK TURN OFF RECV,INTERRUPT WILL COME FROM GNB TO CPU
          LD       20         LOAD LINE CONTROL LCT20
          AND      =X'FD'     TURN OFF RECV BIT
          ST       20         PUT NEW 20 BACK
          OUT      2          TURN OFF RECEIVER
          GNB                 GET NEXT BLOCK TO UPDATE STATUS
          WAIT                WAIT UNTILL START IO
          B        ASYREC     START OVER WHEN START IO
*NOT LAST BLOCK ,GET NEXT BLOCK
          LOC      GETN
          GNB                 GET NEXT BLOCK TO UP DATE STATUS
          BVBT     NEXT       BRANCH IF NEXT CCB IS VALID
*NEXT BLOCK IS NOT VALID,SIGNAL CPU OF ERROR
          LOC      BLOKNG
          LD       20         LOAD LINE CONTROL 20
          AND      =X'FD'     RESET RECEIVE BIT
          ST       20         RESTORE NEW  CONTROL TO 20
          OUT      2          TURN OFF RECEIVER
          LD       31         LOAD ERROR WORK BYTE LCT31
          OR       =X'04'     SET BLOCK NOT VALID BIT
          ST       31         PUT IT IN WORK LCT31
          LD       16         LOAD STATUS WORD 1
          OR       =X'02'     SET PROGRAMER.STATUS BIT 6
          ST       16         THIS WILL INFORM CPU PROG OF ERROR INTERRUPT
*FORCE A CPU INTERRUPT
          INTR                SEND INTERRUPT TO CPU
          WAIT                WAIT FOR START IO
          B        ASYREC     START OVER WHEN START IO
**********
********
******
****

*THIS IS A SAMPLE RECEIVE CHANNEL PROGRAM THAT DOES NOT ECHO DATA BACK TO
*THE TERMINAL. TO DO THIS THE TRANSMITTER WOULD HAVE TO BE CONTROLLED BY
*THE RECEIVE CCP TO TRANSMIT EACH CHARACTER BACK TO THE OPERATOR.
```

Figure C-3 (cont). Sample Program for Receive on Asynchronous Communications-Pac

Framing Error

A framing error occurs when the Asynchronous Line Communications-Pac detects a missing stop bit for a data character. (The expected number of stop bits is indicated by the setting of bit 4 of line register 6.) A framing error causes bit 7 (framing error) of line register 5 to be set to 1. If a framing error is reported, the preceding data bits should be analyzed to ascertain whether a line break has occurred.

A framing error is detected on the receive channel but is stored in a transmit error bit. If the transmitter is turned on before another receive byte is input, then the transmit will show a data service error in the status which is unrelated to the transmit.

If a framing error occurred and the character was accepted through the use of the RECV command, then the condition is reflected in bit 2 of LCT status byte 1 (LCT 16), a data service error. The data service error indication on transmit status (LCT/CCB Status Byte 1) should always be ignored.

It is also possible for the transmit channel program of an asynchronous line to reset data service error so that it never appears in the CCB status.

The normal receive break detection checks line register 5 bit 7 after every channel request interrupt. If set, the character just received is checked by the CCP to see if it has a value of zero. If zero, the CCP should assume that it is the break key on the terminal. If not zero, then the Communications-Pac has detected a valid framing error which should be reported as a line error in the normal manner.

Character Length

The length of each data character to be received or transmitted is specified by the settings of bits 0 and 1 in line register 6. This character length includes parity (5-bit mode cannot have parity). The same character length applies to both channels of a line.

Line register 6 is loaded with information obtained from LCT byte 2/34. During data transfer operations, the character length specified in line register 6 must match the character length specified in LCT byte 2/34. For additional information, see "LCT Byte 2/34—Character Configuration" in Section 5.

Parity

The type of parity to be generated or checked by the MLCP for each data character (as an option) is indicated by the setting of bit 3 of LCT byte 2/34. The Asynchronous Line Communications-Pac neither generates nor checks parity; thus the setting of bit 3 in line register 6 (which is loaded with information obtained from LCT byte 2/34) is not meaningful.

Stop Bits

The number of stop bits associated with each data character is specified by the setting of bit 4 in line register 6. Line register 6 is loaded with information obtained from LCT byte 2/34. For additional information, see "LCT Byte 2/34—Character Configuration" in Section 5.

Cyclic Redundancy Check

The cyclic redundancy check polynomial to be used by the MLCP (as an option) is indicated by the settings of bits 5 and 6 of LCT byte 2/34. The Asynchronous Line Communications-Pac does not perform cyclic redundancy checking; thus the settings of bits 5 and 6 in line register 6 (which is loaded with information obtained from LCT byte 2/34) are not meaningful.

Master Clear

Execution of an IO (Output MLCP Control) instruction by the main memory program causes (among other operations) a master clear of each Communications-Pac. A master clear causes each Communications-Pac to be unconditionally placed in a quiescent state; each line register 2 of each Communications-Pac reset to zero and all channel request interrupts are inhibited.

Line/Channel Number Assignment

Within an individual Asynchronous Line Communications-Pac, lines and channels are numbered as shown below.[6]

Line Number	*Channel Number*	*Direction*
0	0	Receive
0	1	Transmit
1	2	Receive
1	3	Transmit

Externally, the line and channel numbers of each Communications-Pac conform to the MLCP-relative line and channel numbering system shown in Table 6-1. The MLCP-relative channel numbers are used by input/output instructions from the main memory program.

[6] These, line and channel numbers are based on an Asynchronous Line Communications-Pac that services *two* lines. A *single*-line Asynchronous Line Communications-Pac has only channels 0 and 1 of line 0.

Device Identification Number

The device identification numbers for the Asynchronous Line Communications-Pacs are given below. This device identification number can be obtained by a main memory program through use of an IO (Input Device Identification Number) instruction. Use of this instruction allows the main memory program to verify that a given communications channel is serviced by the appropriate type of Communications-Pac.

Type	*Device Identification*
DCM9101/DCM9102 to 9600 bps	2108
DCM9101/DCM9102 to 19200 bps	2118

PHYSICAL INTERFACE TO DATA COMMUNICATIONS EQUIPMENT AND DATA TERMINAL EQUIPMENT

The Asynchronous Line Communications-Pac provides a separate, identical physical interface for each communications line. This interface is designed to permit connection of data communications equipment or data terminal equipment having a standard EIA RS-232-C interface. The Asynchronous Line Communications-Pac's physical interface is also suitable for data communications equipment or data terminal equipment having a CCITT V24 interface–*provided* the equipment's electrical circuitry is compatible with the Communications-Pac.

The Asynchronous Line Communications-Pac's physical interface comprises 10 signals plus ground per line. See Table C-2. This interface permits connection of the most frequently used data communications equipment and data terminal equipment that supports line speeds of up to 19,200 bits per second. Note, however, that not all features of all such equipment are supported.

TABLE C-2. PHYSICAL INTERFACE OF ASYNCHRONOUS LINE COMMUNICATIONS-PAC

Pin No.	To/From DCE/DTE	Function	EIA RS-232-C	CCITT V24	Bit Positions LR2	Bit Positions LR5
2	T	Transmitted Data	BA	103		
3	F	Received Data	BB	104		
4	T	Request to Send	CA	105	1	
5	F	Clear to Send	CB	106		1
6	F	Data Set Ready	CC	107		0
7	-	Signal Ground	AB	102		
8	F	Received Line Signal Detector	CF	109		2
11	T	Secondary Transmitted Data	SBA	118	2	
12	F	Secondary Received Data	SBB	119		4
20	T	Data Terminal Ready	CD	108	0	
22	F	Ring Indicator	CE	125		3

APPENDIX D

SYNCHRONOUS LINE COMMUNICATIONS-PACS

A Synchronous Line Communications-Pac (Type DCM9103 or DCM9104) provides an interface between the MLCP and one or two completely independent synchronous communications lines.[1] For each line, the Synchronous Line Communications-Pac provides the following services:

- o Serial/parallel data conversion for synchronous bit-serial data transfers
- o Character synchronization by use of a synchronization character such as the ASCII SYN
- o Control of data sets
- o Monitoring of data set status

Each communications line comprises a receive channel and a transmit channel and is thus capable of half-duplex or full-duplex data communications operations. Each line has a clocked, independently configurable speed (up to 20,000 bits per second)[2] and an independently configurable data character size (from five to eight bits–including parity, if used); each channel of a line uses the configured line speed and data character size. The Synchronous Line Communications-Pac supports BSC, Basic Mode ASCII, and similarly formatted control procedures.

The following data communications equipment and data terminal equipment is supported through an EIA RS-232-C interface:

- o Bell System 201A, B, C, or equivalent
- o Bell System 203A, B, or equivalent
- o Bell System 208A, B, or equivalent
- o Bell System 209 or equivalent

Figure D-1 illustrates the Synchronous Line Communications-Pac's interface position between the MLCP and synchronous communications lines. (Note that the MLCP can connect any combination of Synchronous Line Communications-Pacs and Asynchronous Line Communications-Pacs up to a total of four.)

LINE REGISTERS

The programming interface to the Synchronous Line Communications-Pac is achieved through its line registers. These line registers are illustrated in Figure D-2.

As indicated in Figure D-2, each communications line is serviced by a different set of line registers. Each channel of a line has a dedicated set of registers and also shares three registers with the other channel of the same line.

Before data transfer operations can begin over a channel, the CCP must load line registers 6, 4, and 2 using OUT (Output) instructions. Line register 2 must be loaded last.

- o Line register 6 is loaded with data character configuration information. This information is obtained from LCT byte 2/34, which must be loaded first.

[1] Throughout this appendix, descriptions are based on a Synchronous Line Communications-Pac that services *two* lines. The *single*-line version of the Synchronous Line Communications-Pac is identical except for the number of lines it services.

[2] Each line's speed is governed by the associated data set or by the MLCP's fixed-rate clock. Details appear under "Data Transfer Clocks," later in this appendix.

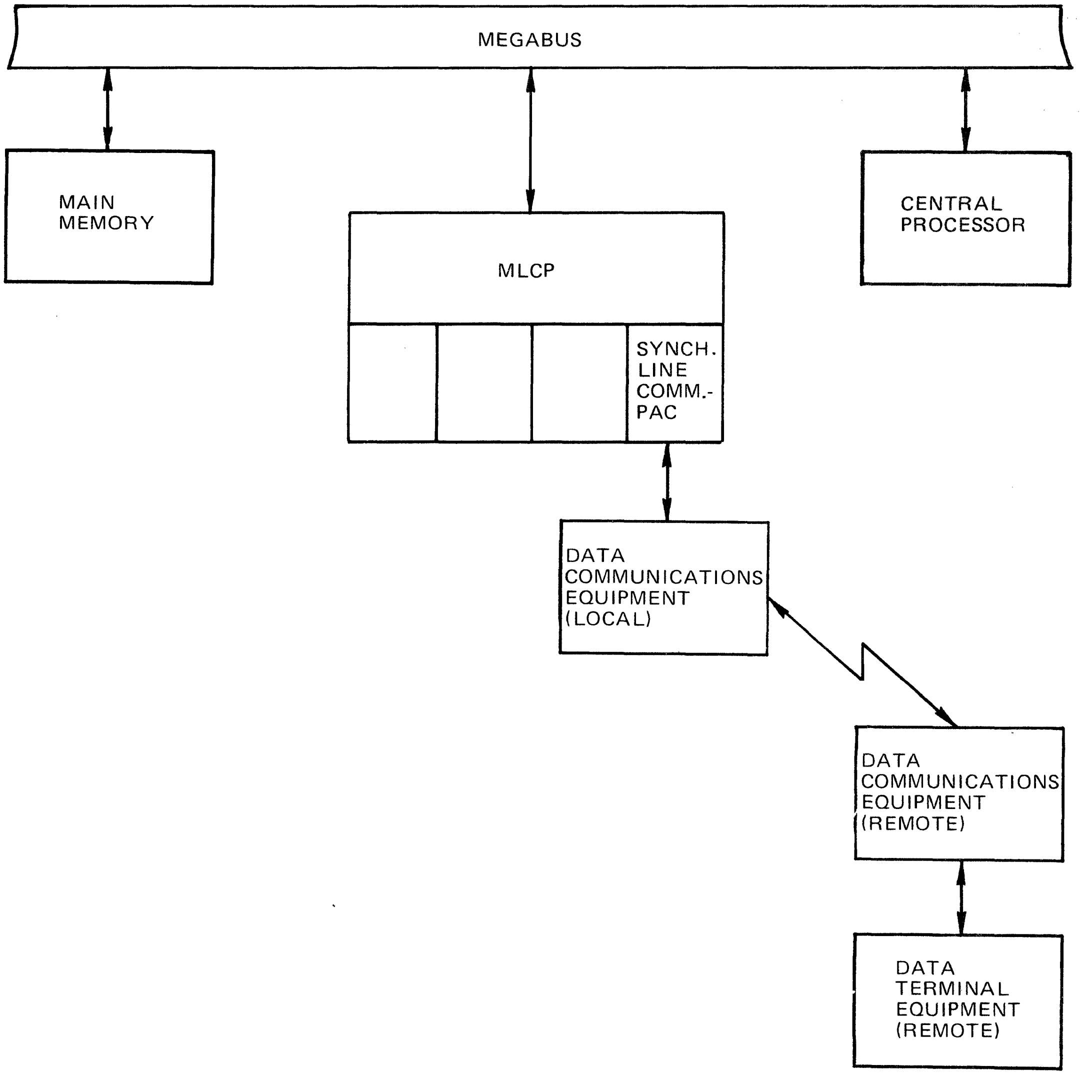

MEGABUS
MAIN MEMORY
MLCP
CENTRAL PROCESSOR
SYNCH. LINE COMM.-PAC
DATA COMMUNICATIONS EQUIPMENT (LOCAL)
DATA COMMUNICATIONS EQUIPMENT (REMOTE)
DATA TERMINAL EQUIPMENT (REMOTE)

Figure D-1. Interface Provided by Synchronous Line Communications-Pac

- o Line register 4 must be loaded with a synchronization character (for a receive channel) or a transmit fill character (for a transmit channel). Normally, line register 4 is loaded with information obtained from a byte in the LCT programming work area.
- o Line register 2 is loaded with data set control and Communications-Pac control information. This information is obtained from LCT byte 20. *

Once data transfer operations have been enabled, a receive CCP uses an RECV (Receive) instruction to obtain a data character from a receive channel's line register 1 following a channel request interrupt for that channel. Similarly, a transmit CCP uses a SEND (Send) instruction to load a data character into a transmit channel's line register 1 following a channel request interrupt for *that* channel.

During processing, a CCP may use IN (Input) instructions to read the contents of individual line registers. OUT (Output) instructions may be used to modify the contents of line registers during processing, but care must be taken to ensure that any such modification of line register contents does not disrupt processing.

The main memory program can read the contents of line register 5 by executing an IO (Input Data Set Status) instruction. The main memory program cannot directly read the contents of any other line register nor can it directly modify the contents of *any* line register.

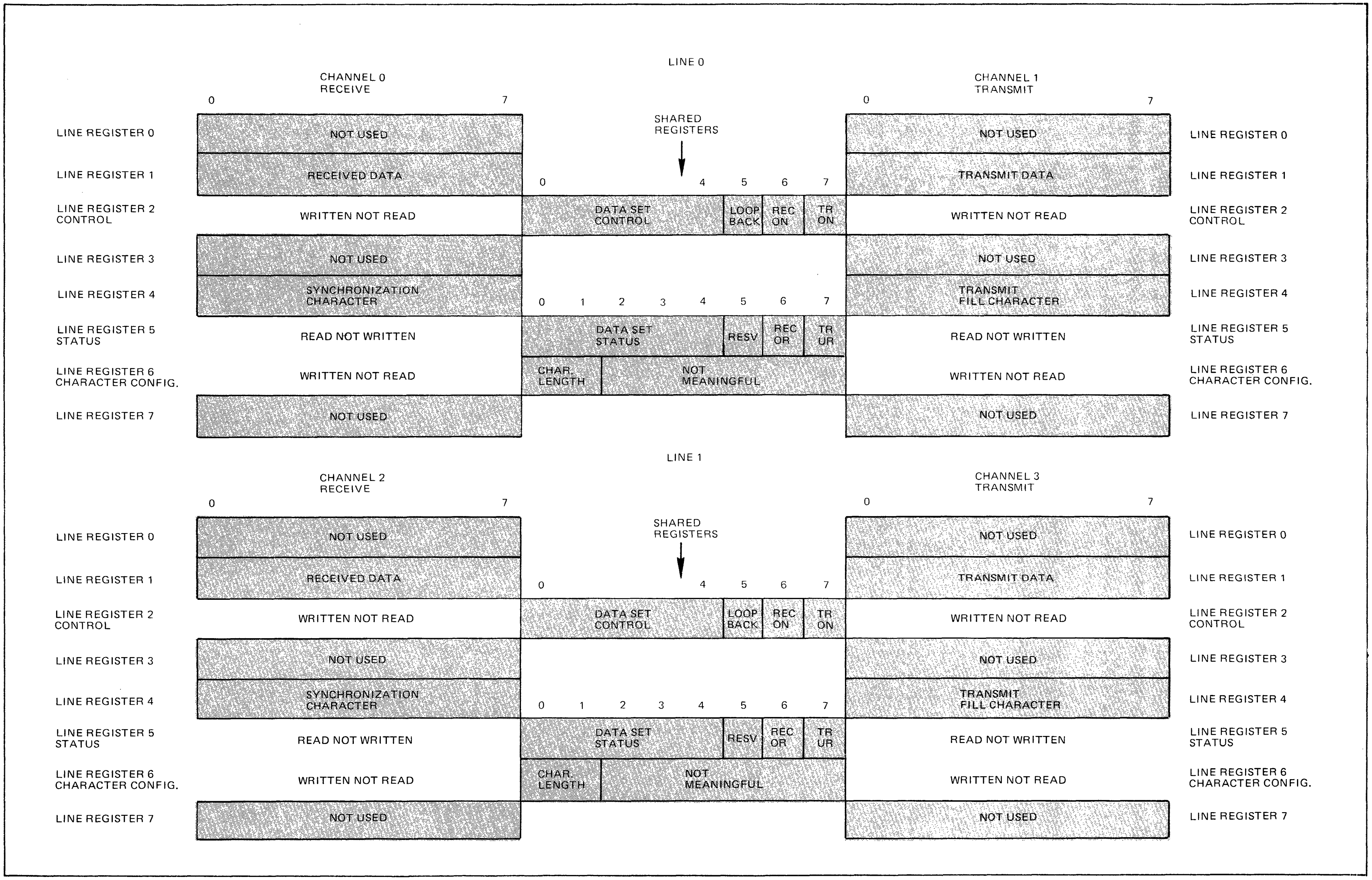


Figure D-2. Registers of Synchronous Line Communications-Pac

By appropriate settings of bits in LCT bytes 8/40 and 15/47, you can cause MLCP firmware to (1) scan for data set or Communications-Pac status changes reflected in line register 5 and (2) take related action(s) as directed by LCT bytes 8/40 and 15/47.

The following subsections provide more detailed information about the individual line registers.

Line Registers for Receive Channel

As shown in Figure D-2, each receive channel has eight line registers, three of which are shared with the transmit channel of the same line.

Line Register 1–Receive Channel

	0	7
LINE REGISTER 1	RECEIVED DATA	

The length of each data character in line register 1 is governed by the settings of bits 0 and 1 of line register 6. If the length is fewer than eight bits, the data character is right-justified and the leftmost bits are zero-filled. In all cases, bit 7 is the first bit received over the channel.

Line Register 2–Receive/Transmit Channel

	0	1	2	3	4	5	6	7
	DATA SET CONTROL					COMMUNICATIONS-PAC CONTROL		
LINE REGISTER 2	DATA TERMINAL READY B	REQUEST TO SEND B	NEW SYNCH B	SPEED SELECT B	DIRECT CONNECT B	LOOP-BACK TEST B	RECEIVE ON R	TRANSMIT ON T

Line register 2 is shared by the receive channel and the transmit channel of the same line. Its contents affect the operations of both channels. If either channel's CCP modifies the contents of line register 2, both channels' operations are affected. (In the diagram, the letter B, R, or T in the lower right-hand corner of each bit position indicates whether the bit pertains to *both* receive and transmit operations, to *receive* operations only, or to *transmit* operations only.)

Line register 2 is loaded by the CCP with information obtained from LCT byte 20. The significance of each bit position is described under "LCT Byte 20–Data Set and Communications-Pac Control" in Section 5.

Line Register 4–Receive Channel

	0	7
LINE REGISTER 4	SYNCHRONIZATION CHARACTER	

Line register 4 must contain a receive synchronization character (normally the ASCII SYN character–16_{16}) to be used to establish synchronization of incoming data characters. Line register 4 is normally loaded with information obtained from a byte of the LCT programming work area for the receive channel.

If the defined character length is fewer than eight bits, the leading bits of the synchronization character must be specified as 1's as this character is loaded into the receive channel's line register 4. If the parity must be odd or even, it must be calculated by the CCP or main memory program for the synchronization character used. The parity would be the most significant bit of the character length specified and does not include the leading fill bits. Note that 5-bit mode cannot have parity.

If the parity of received data characters is to be checked, the parity bit of the synchronization character (the leftmost bit of the defined character length) must contain the proper value as this character is loaded into the receive channel's line register 4.

Line Register 5—Receive/Transmit Channel

	0	1	2	3	4	5	6	7
	DATA SET STATUS					COMMUNICATIONS-PAC STATUS		
LINE REGISTER 5	DATA SET READY	CLEAR TO SEND	CARRIER DETECTOR	RING INDICATOR	RESERVED	RESERVED	RECEIVE OVERRUN	TRANSMIT UNDERRUN

Line register 5 is shared by the receive channel and the transmit channel of the same line. Its contents reflect data set status and Communications-Pac status. Line register 5 can be read by the main memory program through use of an IO (Input Data Set Status) instruction.

If bit 0 of LCT byte 8 (or LCT byte 40) is set to 1, the entire contents of line register 5 will be written to LCT byte 14 (or LCT byte 46) whenever a data set or Communications-Pac status change occurs (*provided* that, in LCT byte 15 (or LCT byte 47), there is a 1 in the bit position that corresponds to the status change). Subsequent action(s) depend on the settings of bits 1, 2, and 3 of LCT byte 9 (or LCT byte 40).

The significance of each bit position of line register 5 is described under "LCT Byte 14/46—Data Set and Communications-Pac Status" in Section 5.

Line Register 6—Receive/Transmit Channel

	0	1	2					7
LINE REGISTER 6	CHARACTER LENGTH		NOT MEANINGFUL					

Line register 6 is shared by the receive channel and the transmit channel of the same line. Its contents affect the operations of both channels. If either channel's CCP modifies the contents of line register 6, both channels' operations are affected.

Line register 6 is loaded by the CCP with information obtained from LCT byte 2 (or LCT byte 34). (Note that the parity and cyclic redundancy check information in LCT byte 2/34 is not meaningful to the Synchronous Line Communications-Pac.) The significance of each bit position is described under "LCT Byte 2/34—Character Configuration" in Section 5.

Line Registers for Transmit Channel

As shown in Figure D-2, each transmit channel has eight line registers, three of which are shared with the receive channel of the same line.

Line Register 1—Transmit Channel

	0 … 7
LINE REGISTER 1	TRANSMIT DATA

The length of each data character in line register 1 is governed by the settings of bits 0 and 1 of line register 6. If the length is fewer than eight bits, the leading bits must be specified as zeros (when the data character is transferred to line register 1); in this case, the data character is right-justified in line register 1 and the leading zeros are not included in transmissions.

If parity is to be generated, the parity bit (the leftmost bit of the defined character length) and the fill bits if needed must be specified as a zero (when the data character is transferred to line register 1); the MLCP generates the correct parity during the transfer to line register 1. The parity bit is inserted in the leftmost bits of the defined character length and is included in transmissions.

In all cases, bit 7 is the first bit of the data character to be transmitted over the channel.

Line Register 2—Receive/Transmit Channel

	0	1	2	3	4	5	6	7
	DATA SET CONTROL					COMMUNICATIONS-PAC CONTROL		
LINE REGISTER 2	DATA TERMINAL READY B	REQUEST TO SEND B	NEW SYNCH B	SPEED SELECT B	DIRECT CONNECT B	LOOP-BACK TEST B	RECEIVE ON R	TRANSMIT ON T

Line register 2 is shared by the transmit channel and the receive channel of the same line. Its contents affect the operations of both channels. If either channel's CCP modifies the contents of line register 2, both channels' operations are affected. (In the diagram, the letter B, R, or T in the lower right-hand corner of each bit position indicates whether the bit pertains to *both* receive and transmit operations, to *receive* operations only, or to *transmit* operations only.)

Line register 2 is loaded by the CCP with information obtained from LCT byte 20. The significance of each bit position is described under "LCT byte 20—Data Set and Communications-Pac Control" in Section 5.

Line Register 4—Transmit Channel

	0 … 7
LINE REGISTER 4	TRANSMIT FILL CHARACTER

Line register 4 must contain a transmit fill character (normally the ASCII SYN character—16_{16}) to be used as idle time fill or in case of a transmit underrun. Line register 4 is normally loaded with information obtained from a byte of the LCT programming work area for the transmit channel.

If the defined character length is fewer than eight bits, the leading bits of the transmit fill character must be specified as 1's as this character is loaded into the transmit channel's line register 4.

If parity is to be generated for transmitted data characters, the parity bit of the transmit fill character (the leftmost bit of the defined character length) must contain the proper value as this character is loaded into the transmit channel's line register 4. Note that 5-bit mode cannot have parity.

Line Register 5–Receive/Transmit Channel

	0	1	2	3	4	5	6	7
	DATA SET STATUS					COMMUNICATIONS-PAC STATUS		
LINE REGISTER 5	DATA SET READY	CLEAR TO SEND	CARRIER DETECTOR	RING INDICATOR	RESERVED	RESERVED	RECEIVE OVERRUN	TRANSMIT UNDERRUN

Line register 5 is shared by the transmit channel and the receive channel of the same line. Its contents reflect data set and Communications-Pac status. Line register 5 can be read by the main memory program through use of an IO (Input Data Set Status) instruction.

If bit 0 of LCT byte 40 (or LCT byte 8) is set to 1, the entire contents of line register 5 will be written to LCT byte 46 (or LCT byte 14) whenever a data set or Communications-Pac status change occurs (*provided* that, in LCT byte 47 (or LCT byte 15), there is a 1 in the bit position that corresponds to the status change). Subsequent action(s) depend on the settings of bits 1, 2, and 3 of LCT byte 40 (or LCT byte 8).

The significance of each bit position of line register 5 is described under "LCT Byte 14/46–Data Set and Communications-Pac Status" in Section 5.

Line Register 6–Receive/Transmit Channel

	0	1	2	7
LINE REGISTER 6	CHARACTER LENGTH		NOT MEANINGFUL	

Line register 6 is shared by the transmit channel and the receive channel of the same line. Its contents affect the operations of both channels. If either channel's CCP modifies the contents of line register 6, both channels' operations are affected.

Line register 6 is loaded by the CCP with information obtained from LCT byte 34 (or LCT byte 2). (Note that the parity and cyclic redundancy check information in LCT byte 2/34 is not meaningful to the Synchronous Line Communications-Pac.) The significance of each bit position is described under "LCT Byte 2/34–Character Configuration" in Section 5.

PROGRAMMING CONSIDERATIONS

The paragraphs that follow present detailed programming requirements for the Synchronous Communications-Pac.

Data Transfers

Within the Synchronous Line Communications-Pac, each data transfer is channel-specific and involves one (of the four) line register 1 and a "shift" buffer associated exclusively with that specific line register 1.

During receive operations, bits of a data character are serially received in a shift buffer. After reception of an entire character (of the length specified by bits 0 and 1 of line register 6), the accumulated contents of the shift buffer are transferred to the receive channel's line register 1 and a channel request interrupt is generated. (As soon as the shift buffer's contents have been transferred to line register 1, the buffer is available for the reception of the first bit of the next incoming character.) When the receive channel's CCP is started in response to the channel request interrupt, it can issue an RECV (Receive) instruction to transfer the data character from line register 1 into the MLCP's R-register.

During transmit operations, the Synchronous Line Communications-Pac generates a channel request interrupt as soon as the previous contents of the transmit channel's line register 1 have been transferred to the associated shift buffer for serial transmission. When the transmit channel's CCP is started in response to the channel request interrupt, it can load the MLCP's R-register with a data character and then issue a SEND (Send) instruction to transfer that character from the R-register to line register 1. The data character will remain in line register 1 until each bit of the preceding data character has been serially transmitted from the associated shift buffer. (The number of bits transmitted is governed by the character length specified in bits 0 and 1 of line register 6.) When the preceding data character has been transmitted, the data character in line register 1 is transferred to the associated shift buffer and a channel request interrupt is generated. Three pad characters are required before resetting Request to Send (line register 2, bit 1) or Transmit On (line register 2, bit 7).

Channel Request Interrupts

The Synchronous Line Communications-Pac generates a channel request interrupt (on a channel-specific basis) whenever a data transfer between the MLCP and the Communications-Pac is possible. For a receive channel, the channel request interrupt occurs when the receive channel's line register 1 has been loaded with an incoming data character. For a transmit channel, the channel request interrupt occurs when the transmit channel's line register 1 is "empty" and can be loaded with the next data character for transmission.

Table 1-2 indicates the MLCP's priorities for servicing channel request interrupts. A CCP will not be required to service successive channel request interrupts faster than the character rate for that channel.

Channel request interrupts will not be enabled for a receive channel unless bit 6 (receive ON) is set to 1 in line register 2. Channel request interrupts will not be enabled for a transmit channel unless bit 1 (request to send) and bit 7 (transmit ON) of line register 2 are set to 1; in addition, bit 1 (clear to send) of line register 5 must be set to 1 by the Communications-Pac.

Speed Selection

During "normal" operations (i.e., communications line connected to Synchronous Line Communications-Pac by means of data communications equipment and no "loop-back" of transmitted data), the data transfer rate for a line is governed by a clock in the data set. If the data set clock permits data transfers at either of two speeds, bit 3 of line register 2 can be used to indicate which of the two speeds is desired. In all cases, each channel of a line uses the indicated data transfer rate.

Direct Connect

If a communications line is direct-connected to a Synchronous Line Communications-Pac (i.e., local data communications equipment is not used as an interface), bit 4 of line register 2 must be set to 1. In this case, the data transfer rate for the line is governed by the MLCP's fixed-rate clock (see Table 6-3). Each channel of the line uses the indicated data transfer rate. If this bit is not on, an external timing source is being used.

"Loop-Back" Test

A "loop-back" test is performed if bit 5 (loop-back test) of line register 2 is set to 1. (In addition, bit 1 (request to send) and bit 7 (transmit ON) of line register 2 must be set to 1.) In this case, data characters sent to the transmit channel's line register 1 will be "looped back" to the receive channel's line register 1 without being transmitted. The transmit channel will be held in a mark (logical 1) condition during a "loop-back" test and no external data will be received.

The "loop-back" test is possible regardless of whether the communications line is connected to a Synchronous Line Communications-Pac by means of data communications equipment or whether the communications line is direct-connected to the Synchronous Line Communications-Pac. The data transfer rate is governed by the MLCP's fixed-rate clock (see Table 6-3).

Receive/Transmit ON

Channel request interrupts are not enabled for a *receive* channel unless bit 6 (receive ON) of line register 2 is set to 1. Channel request interrupts are not enabled for a *transmit* channel unless bit 7 (transmit ON) of line register 2 is set to 1.[3]

Receive Character Synchronization

Whenever bit 6 (receive ON) of line register 2 changes from 0 to 1, the Synchronous Line Communications-Pac begins searching for a synchronization character in the shift buffer associated with the receive channel's line register 1. As each bit is received over the communications line, the Synchronous Line Communications-Pac compares the accumulated character in the shift buffer with the contents of line register 4 (which contains the user-provided receive synchronization character). When the character (bit pattern) in the shift buffer matches the character in line register 4, the character in the shift buffer (i.e., the synchronization character) is loaded into the receive channel's line register 1 and a channel request interrupt is generated. (No channel request interrupts will have been generated while the Synchronous Line Communications-Pac was searching for the synchronization character.) Once character synchronization is achieved, the synchronization character and, in turn, each subsequent data character is transferred to line register 1 causing a channel request interrupt to be generated.

Bit 6 (receive ON) of line register 2 changes from 0 to 1 when line register 2 is loaded with a suitable value after being initialized (to zero) by an IO (Output Channel Control) or IO (Output MLCP Control) instruction from the main memory program. This bit is also changed from 0 to 1 when an SFS (Search for Synchronization) instruction is executed by the CCP. In this latter case, the SFS instruction should be followed by a WAIT (Wait) instruction; the CCP will be suspended until detection of a synchronization character causes a channel request interrupt to be generated for this channel. To avoid a false synchronization, the user should evaluate the next character. If it is a synchronization character, then it can be assumed that the line is synchronized and normal data processing can begin. If, however, the next character is not a synchronization character, the user should assume a false synchronization and restart the search for synchronization.

Clear to Send

Bit 1 (clear to send) of line register 5 must be set to 1 in order for channel request interrupts to be enabled for a transmit channel.[4] This bit position is maintained by the Communications-Pac.

Receive Overrun

A receive overrun occurs when a data character in a receive channel's line register 1 is overwritten with another incoming data character because the CCP did not respond quickly enough to the channel request interrupt generated for the overwritten data character. A receive overrun condition causes bit 6 (receive overrun) of line register 5 to be set to 1. This bit is reset to 0 (by the Communications-Pac) when the CCP next receives a data character before it is overwritten.

[3] In addition, for a transmit channel, bit 1 (request to send) of line register 2 must be set to 1. Moreover, bit 1 (clear to send) of line register 5 must be set to 1 by the Communications-Pac.

[4] In addition, bit 1 (request to send) and bit 7 (transmit ON) of line register 2 must be set to 1.

The receive overrun condition also causes bit 2 (data service error) of LCT byte 16 to be set to 1. When the contents of LCT bytes 16 and 17 are transferred to the CCB status bytes at the end of processing relative to a CCB, this bit setting permits end-of-block processing to detect and respond to the receive overrun condition.

Transmit Underrun

When the synchronous transmitter is turned on, a transmit underrun error may occur causing a transmit data service error in the CCB status. A transmit underrun occurs when a CCP does not load the transmit channel's line register 1 quickly enough in response to a channel request interrupt. In this case, bit 7 (transmit underrun) is set to 1 in line register 5 and a transmit fill character (from the transmit channel's line register 4) is transferred to the shift buffer associated with the transmit channel's line register 1. Bit 7 of line register 5 is reset to 0 (by the Communications-Pac) when the CCP next loads the transmit channel's line register 1 before a second channel request interrupt is generated.

The transmit underrun condition also causes bit 2 (data service error) of LCT byte 48 to be set to 1. When the contents of LCT bytes 48 and 49 are transferred to the CCB status bytes at the end of processing relative to a CCB, this bit setting permits end-of-block processing to detect and respond to the transmit underrun condition.

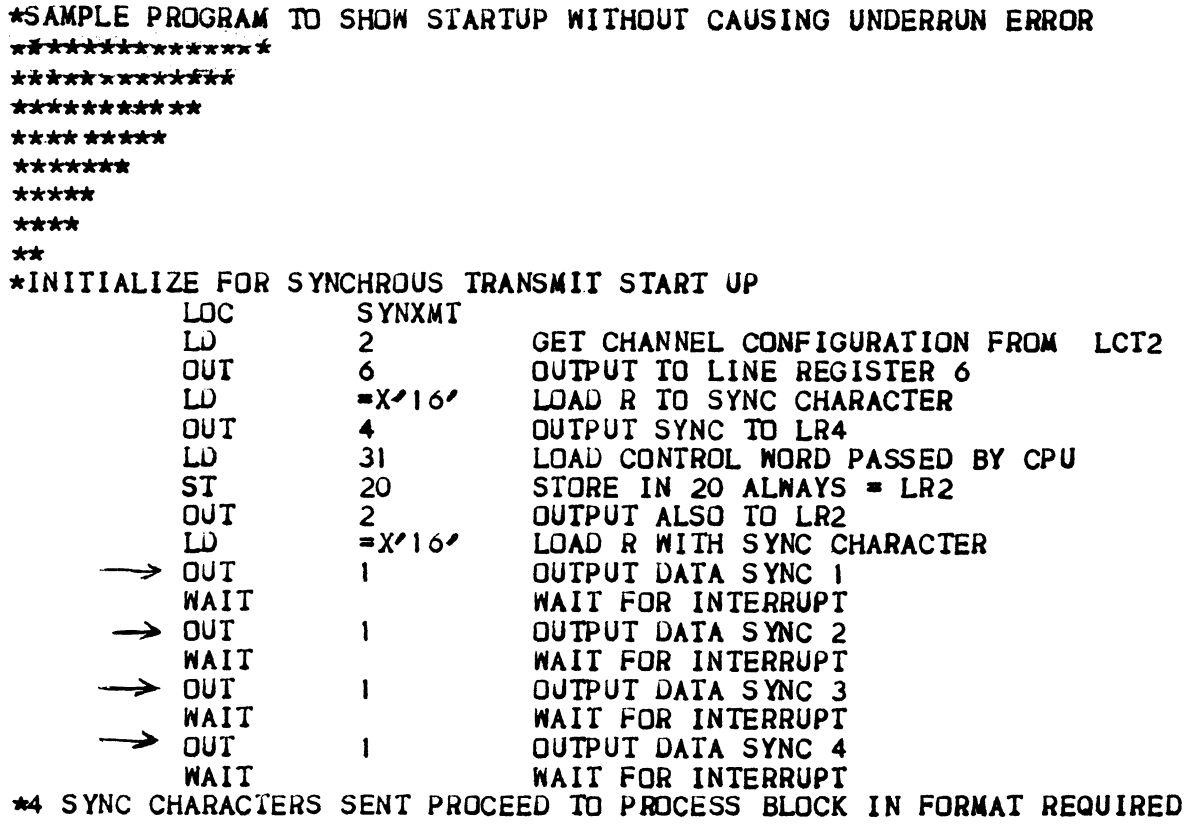
If a specific sequence of transmit fill characters is required in the event of a transmit underrun, it is the responsibility of the CCP to respond to a transmit underrun as soon as it is detected and then provide the desired sequence of characters. (The first transmit fill character will be obtained from the transmit channel's line register 4.)

Setting bit 7 of line register 2 causes an indication of a transmit underrun in line register 5 bit 7; the use of the SEND instruction at this time causes bit 2 of transmit LCT status bit 1 (LCT 48) to be set indicating data service error. To avoid the latter, use one of the following methods.

Method 1: Since most synchronous messages start with three or four synchronization characters and since lead characters are not normally included in CRC, these characters could be sent using the OUT instruction.
Method 2: After transmitting three or four synchronization characters using the SEND command, read LCT byte 48 (transmit status byte 1), reset bit 2, and then write the result to LCT byte 48.

Refer to the portion of a sample channel program shown in Figure D-3.

The (OUT 1) command does not check for error conditions and therefore does not set the data service error status bit. The underrun error is cleared when the second OUT 1 is executed because the transmit shift register and the holding register now have valid data. The CRC is not calculated on synchronization characters and this technique does not disrupt the error checking.



```
*SAMPLE PROGRAM TO SHOW STARTUP WITHOUT CAUSING UNDERRUN ERROR
************** *
*************
*********** **
**** *****
*******
*****
****
**
*INITIALIZE FOR SYNCHROUS TRANSMIT START UP
          LOC       SYNXMT
          LD        2           GET CHANNEL CONFIGURATION FROM  LCT2
          OUT       6           OUTPUT TO LINE REGISTER 6
          LD        =X'16'      LOAD R TO SYNC CHARACTER
          OUT       4           OUTPUT SYNC TO LR4
          LD        31          LOAD CONTROL WORD PASSED BY CPU
          ST        20          STORE IN 20 ALWAYS = LR2
          OUT       2           OUTPUT ALSO TO LR2
          LD        =X'16'      LOAD R WITH SYNC CHARACTER
      --> OUT       1           OUTPUT DATA SYNC 1
          WAIT                  WAIT FOR INTERRUPT
      --> OUT       1           OUTPUT DATA SYNC 2
          WAIT                  WAIT FOR INTERRUPT
      --> OUT       1           OUTPUT DATA SYNC 3
          WAIT                  WAIT FOR INTERRUPT
      --> OUT       1           OUTPUT DATA SYNC 4
          WAIT                  WAIT FOR INTERRUPT
*4 SYNC CHARACTERS SENT PROCEED TO PROCESS BLOCK IN FORMAT REQUIRED
```

Figure D-3. Sample Program Showing Startup Without Causing Underrun Error

Character Length

The length of each data character to be received or transmitted is specified by the settings of bits 0 and 1 in line register 6. This character length includes parity (if in 6-, 7-, or 8- bit mode, no parity in 5-bit mode). The same character length applies to both channels of a line.

Line register 6 is loaded with information obtained from LCT byte 2/34. During data transfer operations, the character length specified in line register 6 must match the character length specified in LCT byte 2/34. For additional information, see "LCT Byte 2/34–Character Configuration" in Section 5. Note that only 6-bit and 8-bit modes are supported by the cyclic redundancy checking.

Parity

The type of parity to be generated or checked by the MLCP for each data character (as an option) is indicated by the setting of bit 3 of LCT byte 2/34. The Synchronous Line Communications-Pac neither generates nor checks parity; thus the setting of bit 3 in line register 6 (which is loaded with information obtained from LCT byte 2/34) is not meaningful.

Cyclic Redundancy Check

The cyclic redundancy check polynomial to be used by the MLCP (as an option) is indicated by the settings of bits 5 and 6 of LCT byte 2/34. The Synchronous Line Communications-Pac does not perform cyclic redundancy checking; thus the settings of bits 5 and 6 in line register 6 (which is loaded with information obtained from LCT byte 2/34) are not meaningful.

Data Transfer Clocks

Each line may use one of two clock sources: one external (e.g., modem), one internal (the MLCP clock common to all lines).

* The data set clock is used if the communications line is connected to the Synchronous Line Communications-Pac by means of data communications equipment and if data transfers are taking place in "normal" (i.e., not "loop-back" test) mode. (If the data set clock permits data transfers at either of two speeds, bit 3 of line register 2 can be used to indicate which of the two speeds is desired.)

The MLCP's fixed-rate clock is used if the communications line is direct-connected to the Synchronous Line Communications-Pac and/or if data transfers are taking place in "loop-back" mode (see bits 4 and 5 of line register 2). The possible settings of the MLCP's fixed-rate clock are shown in Table 6-3.

There is only one fixed rate clock per MLCP, which forces all lines that are directly connected to run at the selected speed of the fixed-rate clock.

Master Clear

Execution of an IO (Output MLCP Control) instruction by the main memory program causes (among other operations) a master clear of each Communications-Pac. A master clear causes each Communications-Pac to be unconditionally placed in a quiescent state; each line register 2 of each Communications-Pac is reset to zero and all channel request interrupts are inhibited.

Line/Channel Number Assignment

Within an individual Synchronous Line Communications-Pac, lines and channels are numbered as shown below.[5]

Line Number	*Channel Number*	*Direction*
0	0	Receive
0	1	Transmit
1	2	Receive
1	3	Transmit

Externally, the line and channel numbers of each Communications-Pac conform to the MLCP-relative line and channel numbering system shown in Table 6-1. The MLCP-relative channel numbers are used by input/output instructions from the main memory program.

Device Identification Number

The device identification number for a Synchronous Line Communications-Pac is 2158_{16}. This device identification number can be obtained by a main memory program through use of an IO (Input Device Identification Number) instruction. Use of this instruction allows the main memory program to verify that a given communications channel is serviced by the appropriate type of Communications-Pac.

PHYSICAL INTERFACE TO DATA COMMUNICATIONS EQUIPMENT AND DATA TERMINAL EQUIPMENT

The Synchronous Line Communications-Pac provides a separate, identical physical interface for each communications line. This interface is designed to permit connection of data communications equipment or data terminal equipment having a standard EIA RS-232-C interface. Signal timing characteristics are compatible with EIA RS334. The Synchronous Line Communications-Pac's physical interface is also suitable for data communications equipment or data terminal equipment having a CCITT V24 interface—*provided* the equipment's electrical circuitry is compatible with the Communications-Pac.

[5] These line and channel numbers are based on a Synchronous Line Communications-Pac that services *two* lines. A *single*-line Synchronous Line Communications-Pac has only channels 0 and 1 of line 0.

The Synchronous Line Communications-Pac's physical interface comprises 12 signals plus ground per line. See Table D-1. This interface permits connection of the most frequently used medium-speed data communications equipment and data terminal equipment. Note, however, that not all features (e.g., reverse channel) of all such equipment are supported.

TABLE D-1. PHYSICAL INTERFACE OF SYNCHRONOUS LINE COMMUNICATIONS-PAC

Pin No.	To/From DCE/DTE	Function	EIA RS-232-C	CCITT V24	Bit Positions	
					LR2	LR5
2	T	Transmitted Data	BA	103		
3	F	Received Data	BB	104		
4	T	Request to Send	CA	105	1	
5	F	Clear to Send	CB	106		1
6	F	Data Set Ready	CC	107		0
7	-	Signal Ground	AB	102		
8	F	Received Line Signal Detector	CF	109		2
14	T	New Sync	-	-	2[a]	
15	F	Transmitter Signal Element Timing	DB	114		
17	F	Receiver Signal Element Timing	DD	115		
20	T	Data Terminal Ready	CD	108	0	
22	F	Ring Indicator	CE	125		3
23	T	Data Signal Rate Selector	CH	111	3[a]	

[a] Not applicable in direct-connect or "loop-back" test mode.

APPENDIX E
REFERENCE LITERATURE FOR DATA COMMUNICATIONS EQUIPMENT

The following non-Honeywell publications should be referenced as appropriate.

- o EIA (Electronic Industries Association) Standard RS-232-C
- o CCITT (International Consultive Committee for Telephony and Telegraphy) White Book, Volume VIII
- o Data Set 103A Interface Specification—February 1967 (Bell System Technical Reference, PUB41101)
- o Data Set 103A3/103E/103G/103H Interface Specification—October 1973 (Bell System Technical Reference, PUB41102)
- o Data Set 103F Interface Specification—May 1964 (Bell System Technical Reference, PUB41103)
- o Data Set 113A Interface Specification—August 1973 (Bell System Technical Reference, PUB41104)
- o 113-Type Data Station Interface Specification—October 1971 (Bell System Technical Reference PUB41105)
- o Data Sets 201A & B—August 1969 (Bell System Technical Reference, PUB41201)
- o Data Set 201C Interface Specification—April 1973 (Bell System Technical Reference, PUB41210)
- o Data Sets 202C & D Interface Specification—May 1964 (Bell System Technical Reference, PUB41202)
- o Data Sets 202S & T Interface Specification—August 1974 (Bell System Technical Reference, PUB41212)
- o Data Set 203-Type—Revised April 1974 (Bell System Technical Reference, PUB41204)
- o Data Set 208A Interface Specification—November 1973 (Bell System Technical Reference, PUB41209)
- o Data Set 208B Interface Specification—August 1973 (Bell System Technical Reference PUB41211)

COMPUTER GENERATED INDEX

CCB STATUS FIELD AFTER &BLOCK MODE WRITE&. 7-5

CUT ALONG LINE

HONEYWELL INFORMATION SYSTEMS
Technical Publications Remarks Form

TITLE	SERIES 60 (LEVEL 6) MLCP PROGRAMMER'S REFERENCE MANUAL	ORDER NO.	AT97, REV. 1
		DATED	MAY 1977

ERRORS IN PUBLICATION

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Your comments will be promptly investigated by appropriate technical personnel and action will be taken as required. If you require a written reply, check here and furnish complete mailing address below. □

FROM: NAME ____________________ DATE ____________

TITLE ____________________

COMPANY ____________________

ADDRESS ____________________

____

CUT ALONG LINE

PLEASE FOLD AND TAPE –
NOTE: U. S. Postal Service will not deliver stapled forms

FOLD ALONG LINE

FIRST CLASS
PERMIT NO. 39531
WALTHAM, MA
02154

Business Reply Mail
Postage Stamp Not Necessary if Mailed in the United States

Postage Will Be Paid By:

HONEYWELL INFORMATION SYSTEMS
200 SMITH STREET
WALTHAM, MA 02154

ATTENTION: PUBLICATIONS, MS 486

FOLD ALONG LINE

Honeywell

Honeywell

Honeywell Information Systems
In the U.S.A.: 200 Smith Street, MS 486, Waltham, Massachusetts 02154
In Canada: 2025 Sheppard Avenue East, Willowdale, Ontario M2J 1W5
In Mexico: Avenida Nuevo Leon 250, Mexico 11, D.F.

19997, 1.5178, Printed in U.S.A.