Honeywell

SERIES 60 (LEVEL 6)

TYPE DCM9106
HIGH-LEVEL DATA LINK CONTROL
ADAPTER MANUAL

Honeywell

SERIES 60 (LEVEL 6)

TYPE DCM9106
HIGH-LEVEL DATA LINK CONTROL
ADAPTER MANUAL

Doc. No. 71010977-200 Order No. FN12, Rev. 1

RECORD OF REVISIONS

REVISION	DATE	AUTHORITY	AFFECTED PAGES
-100	Jan. 1977	Interim Issue	--
-200	June 1977	BLC061762 BLC061866	Entire Manual

Hardware Publications, M&TO, Billerica, MA 01821 File No.: 4S:3

977/FN12

CONTENTS

CONTENTS

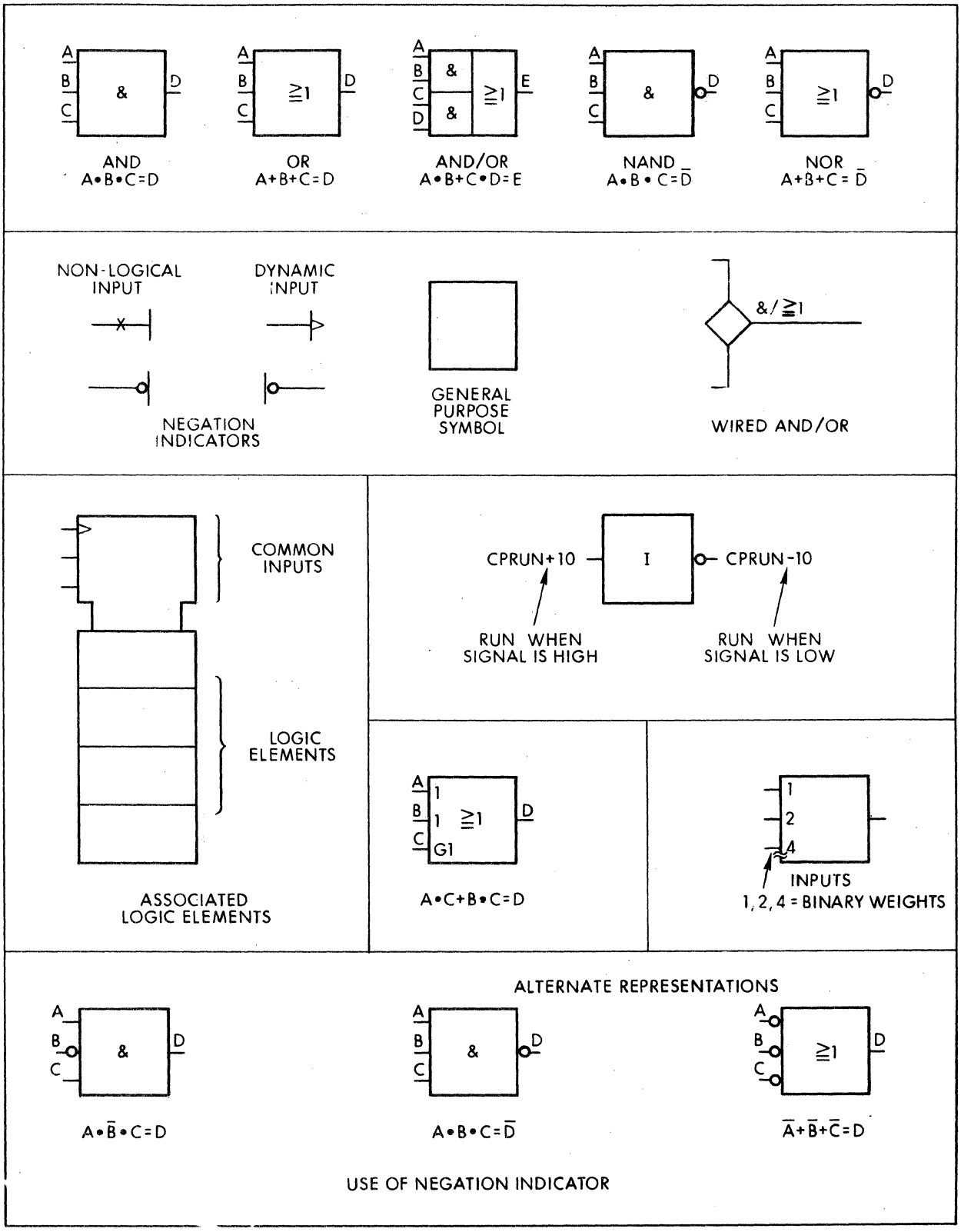
CONTENTS

ILLUSTRATIONS

TABLES

TABLES

LOGIC SYMBOLOGY



I
INTRODUCTION

1.1 SCOPE AND PURPOSE OF THE DOCUMENT

This product manual describes the hardware and firmware of the Type DCM9106 High-Level Data Link Control (HDLC) Adapter.

The theory of operation for the HDLC, presented in Section II, provides a description of the functional hardware elements and their application and analyzes the operation of these elements at the detailed level presented in the logic block diagrams. These are found in the Type DCM9106 HDLC Adapter Reference Manual, Order No. FN13. Programming and medium information is presented to aid in understanding the HDLC adapter hardware description. To obtain programming details, refer to the Level 6 Minicomputer Handbook, Order No. AS22 or the Peripherals Handbook, Order No. AT04.

A description of the HDLC adapter firmware is presented in Section III; a description of and flow charts for the HDLC adapter are also included in Section III of this manual.

1.2 ADAPTER GENERAL DESCRIPTION

The Type DCM9106 High-Level Data Link Control (HDLC) Adapter, one of a series of Communications-Pacs, is a solid-state module used with a Multiline Communications Controller (MLC) to control a single communication line in the Model 6/34, 6/36, or 6/43 configuration of the Series 60 Level 6 computer system. The HDLC subsystem, whose attachment configuration is illustrated in Figure 1-1, can transmit to or receive data from a remote piece of Data Terminal Equipment (DTE).

The HDLC adapter consists of Dual In-Line Packages (DIPs) mounted on a single-size Series 60 Level 6 board (BD2DLC) utilizing printed wiring assembly (PWA) techniques. The HDLC adapter is mounted on the MLC package using its two 25-pin in-line connectors.

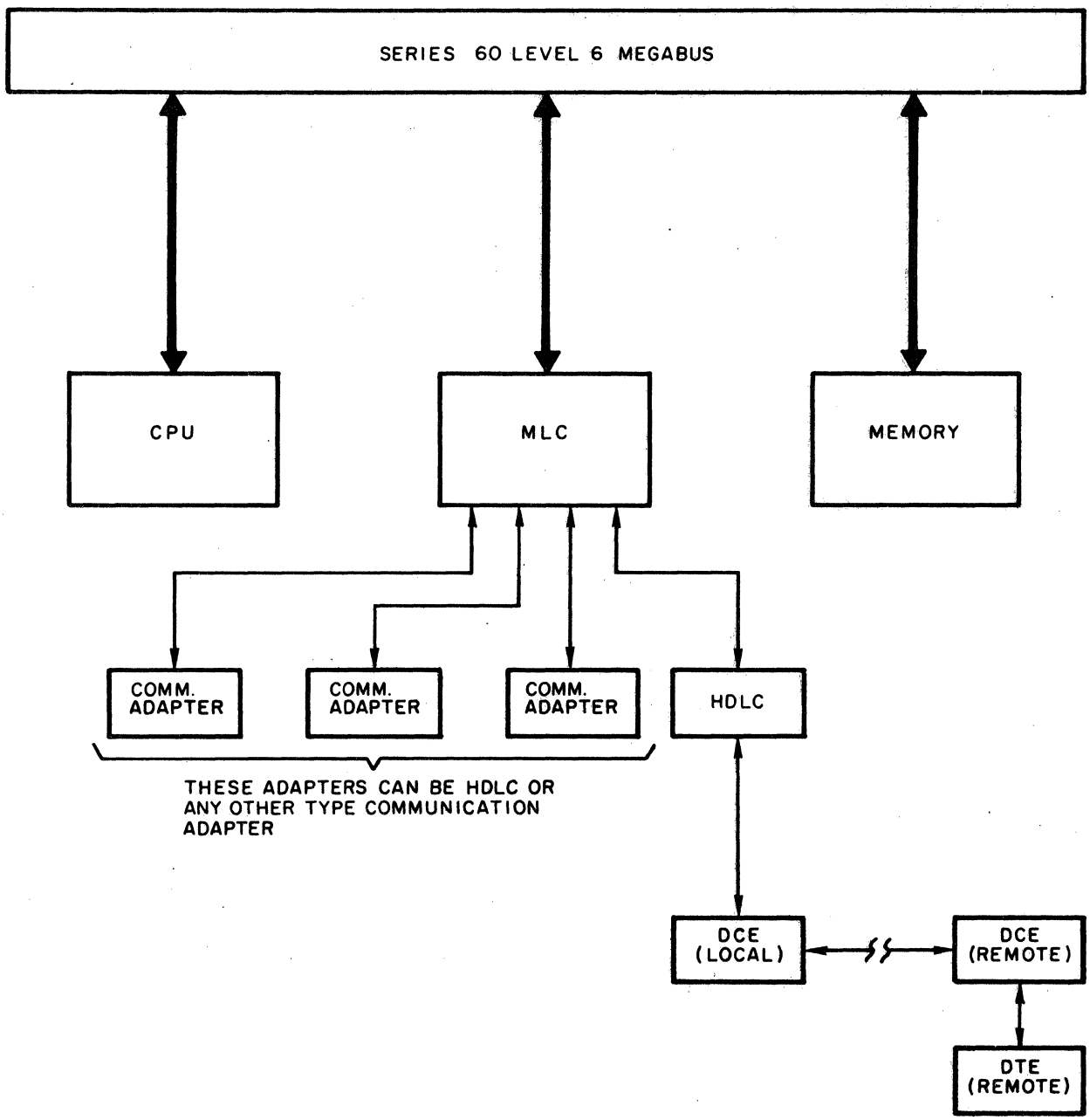


Figure 1-1 HDLC System Block Diagram

1.3 FUNCTIONAL DESCRIPTION

The HDLC adapter contains the communication-line-specific hardware and firmware for control of a single synchronous clocked data communication line. The HDLC provides for independent transmit and receive of data, with individual transmit and receive character sizes and either half-duplex or full-duplex operation. Data characters of five bits through eight bits can be transmitted or received, and this byte size is dynamically controlled by the Channel Control Programs (CCPs) stored in the MLC random access memory (refer to subsection 2.1). The format of the data for transmit and receive operations is in frames and is discussed in subsection 2.4.1 of this manual.

The HDLC adapter hardware is configured to handle a four-bit data structure rather than the nominal eight-bit data structure. The only area in which the HDLC is required to deal with eight-bit data structures is at the HDLC/MLC data line interface (see Figure 1-2). During discussions of the HDLC adapter hardware, any four-bit data structure is referred to as a "byte," and any eight-bit data structure (at the interface level) is called a "word".

1.3.1 Multiline Communications Controller (MLC)

The MLC is a microprogrammed communication line control unit which supports via the HDLC adapter a synchronous clocked data communication line. The majority of the firmware portion of the MLC is generalized to facilitate its application as a control element for various types of communication line adapters. The MLC performs the following general purpose control functions:

- Execution of Series 60 Level 6 Megabus* network sequences
- Command decoding
- Status and control register storage
- Data information multiplexing to adapters
- Execution of CCPs to accomplish transmit or receive operations
- Generation of data verification information
- Checking of data verification information.

1.3.2 HDLC Adapter

The HDLC adapter supports a single synchronous clocked data communication line through the application of the firmware and hardware located on the adapter. The HDLC adapter performs the following communication-line-specific functions:

- Controls communication line interface dialogues
- Generates various types of control sequences for transmission

*Trademark of Honeywell Information Systems Inc.

- Recognizes various types of control sequences on receive
- Supplies and deletes zeros as necessary to provide data field transparency (refer to subsection 2.4.1)
- Supplies data and status information to the MLC
- Provides for data to be wrapped from the MLC to the communication line interface and back to the MLC for test purposes.

1.3.3 Attachable Communication Equipment

The type of communication equipment which can be attached to the HDLC must be able to receive and transmit data across an interface (see Figure 1-2) in a bit serial manner in the format illustrated in Figure 2-7. The communication equipment must also supply the timing required for both transmit and receive operations and the necessary control signals for HDLC operation.

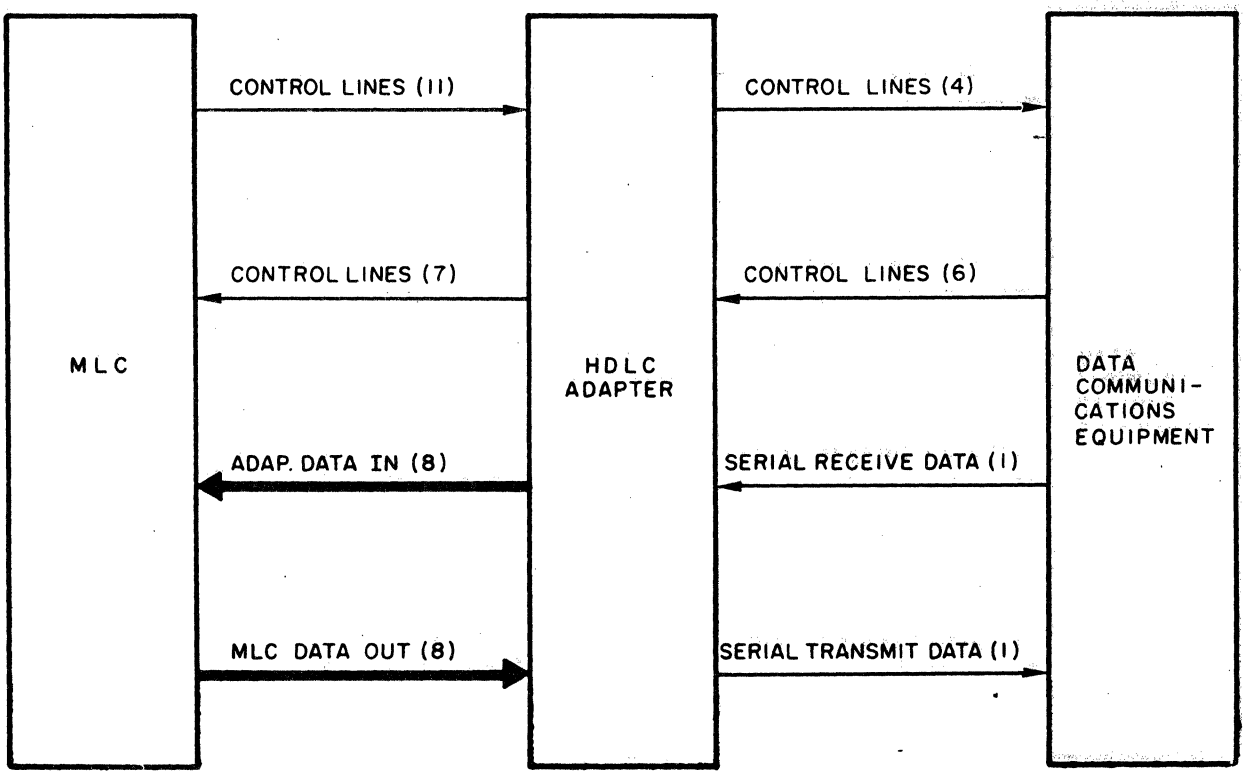
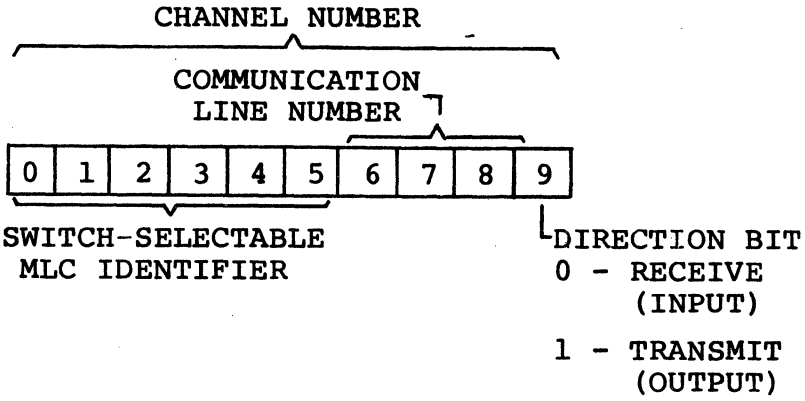


Figure 1-2 MLC/HDLC Adapter/DCE Interfaces

1.4 OPERATIONAL SUMMARY

Each communication line attached to the MLC by way of the HDLC adapter (or other communication adapters) is addressable by software via channel numbers. A line has two channel numbers assigned, differing only in the low-order bit position (called the direction bit). When an I/O Load instruction for a line is accepted, the direction bit of the channel number specifies if this is an input (receive) or output (transmit) data transfer. The following indicates the composition of a channel number, where bits 0 through 5 are assigned at system installation.



Software usability of the communication lines attached to the MLC is such that the channels are, in general, independent of one another. For example, a transmit operation of the HDLC communication line is not dependent on the activity of the other HDLC channel (receive channel) except that the MLC can possibly stall the initiation of a command sequence to one channel while the MLC is busy servicing another channel.

The MLC contains a set of software-loadable random access memory locations in which the Channel Control Programs (CCPs), for controlling the transmit or receive operation on each line, are stored. Commands addressed to a nonbusy HDLC channel are always accepted by the MLC and the CCP stored in the random access memory for that channel is executed under control of the MLC firmware. The CCP is responsible for setting HDLC control information, transferring data to or from the HDLC, and examining adapter status prior to each data transfer. Upon completion of the operation, adapter status is used to update the MLC status for the channel, and the software is informed of the results of the operation.

1.5 REFERENCE DOCUMENTS

The information contained in the following documents will facilitate an understanding of the HDLC adapter and the system of which it is a part.

TITLE	PART NO.	ORDER NO.
Model 34/36 System Manual	71010200-200	FL35
Type MLC9101 MLC Manual	71010230-100	FL48
Type DCM9106 HDLC Adapter Reference Manual	71010405-100	FN87
Circuits Description Reference Manual	71010206-200	FL47
Power System Manual	71010290-200	FL34
Series 60 Level 6 Peripherals Manual	N/A	AT04
Series 60 level 6 Minicomputer Handbook	N/A	AS22
Level 6 Checkout and T&V Manual	N/A	AW94

II THEORY OF OPERATION

The High Level Data Link Control Adapter (HDLC), in conjunction with the MLC, is a software- and firmware-controlled communication line peripheral device adapter. The HDLC, when attached to the MLC, controls input and output (transmit and receive) operations.

The HDLC adapter contains all the communication-line-specific hardware as well as storage for the firmware which is necessary for the functional implementation of data transfer or control sequences of which the Data Communication Equipment (DCE) is capable. The HDLC adapter hardware is divided into five major logic components: MLC control logic, transmit data and control path, microprogram control memory, microprocessor, and receive data and control path (see Figure 2-1), which provides the path for data transfer and allows the control of efficient communication line operations.

2.1 SOFTWARE

Two levels of software are associated with HDLC performance: central processor software operations and Channel Control Program (CCP) software executions.

The central processor software, through the application of the function codes listed in subsection 2.2.1 of the MLC Product Manual, loads the channel control program into the MLC scratch pad memory (refer to Section IV in the MLC product manual listed in Section I). Also utilizing function code capabilities, it is the

initiating factor for communication line operations. Actual performance of the communications line operation starts once the central processor software completes loading the necessary control parameters into the MLC Scratch Pad Memory (SPM). Control of the operation will then result from the execution of the CCP software located in the MLC SPM.

The CCPs are the MLC resident software which allows flexibility in the execution of communication line operations. Information pertaining to CCP instruction type and CCP location is found in Section IV of the MLC Product Manual. Examples of typical CCPs for a transmit and a receive operation are presented in Appendix B and Appendix C, respectively.

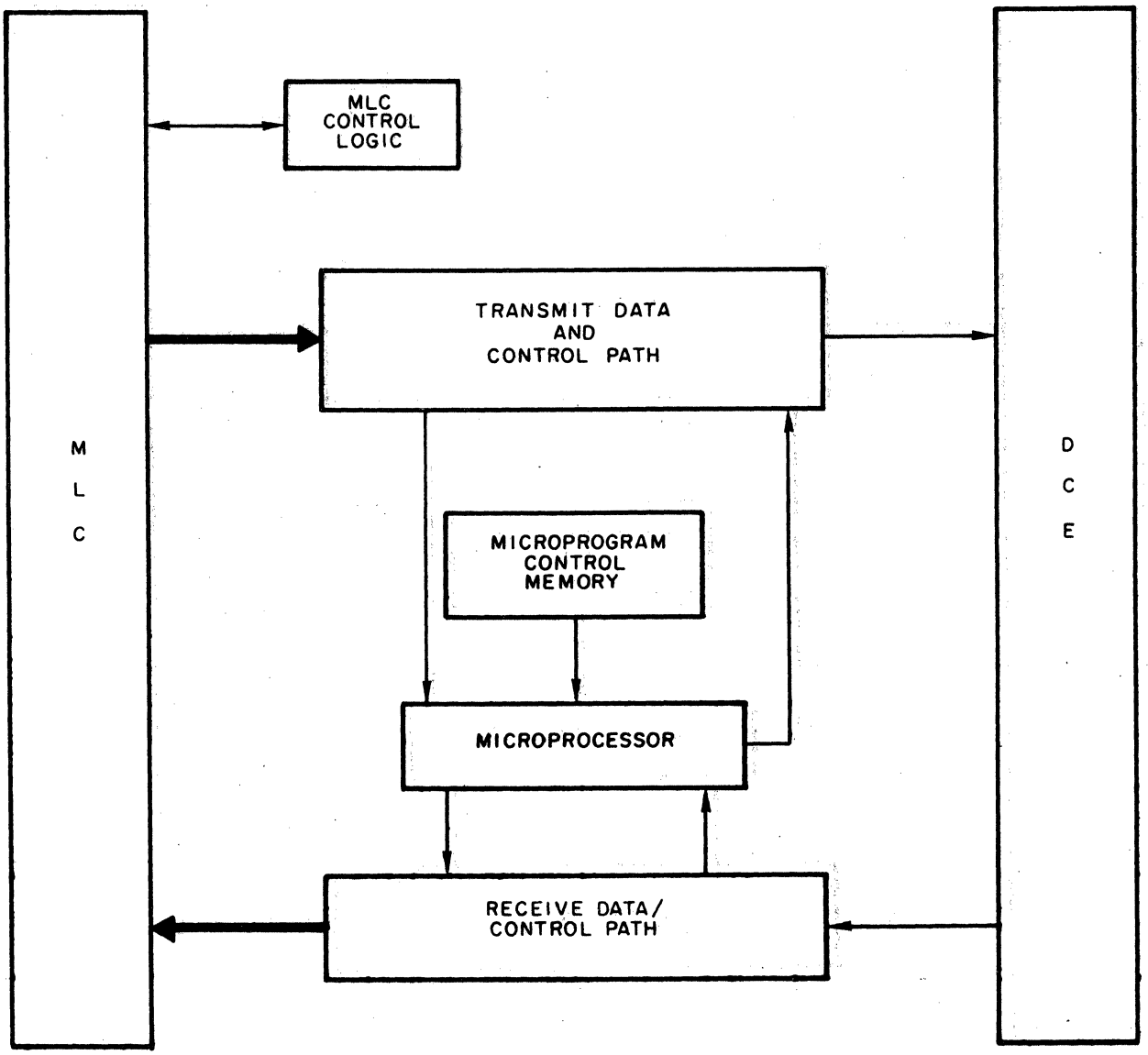


Figure 2-1 HDLC Major Block Diagram

2.2 FIRMWARE

As in the case of the software, there are two levels of firmware associated with HDLC functionality: the generalized firmware and the HDLC-specific firmware. The generalized firmware is an integral part of the MLC microprogram control store and maintains control of the HDLC/MLC interface and also performs execution of CCP instructions. Intermediate flow charts of this firmware and a description of the data control information and firmware utilization are found in Section IV, Theory of Operations-Cycle Flow, of the MLC Product Manual listed in Section I. The HDLC-specific firmware is located in a microprogram control memory on the adapter. It interprets internal events or conditions pertaining to the HDLC and reacts in a prescribed manner (i.e., setting or resetting of hardware functions). Efficient data transfer is also the result of firmware control of hardware elements in the data path. Section III, Theory of Operation-Cycle Flow, of this manual supplies intermediate flow charts of the HDLC-specific firmware routines and descriptions of the control data, routine application, and firmware word structure.

2.3 HARDWARE

As shown in Figure 2-1, the HDLC hardware is organized into five fundamental logic areas. Figure 2-2, an intermediate block diagram of the HDLC adapter, depicts the logical components comprising each of these areas and shows the interconnections with the MLC and the DCE. Although the primary function of the HDLC is to establish efficient data flow between the MLC and the communication line, the adapter performs the following secondary functions of:

- Assisting the MLC in formatting data transmitted to the communication line
- Developing timing and control signals for the communication line
- Generating operational error indicators for MLC interrogation
- Generating transmit and receive channel request interrupts and sending them to the MLC
- Generating status information for sending to the MLC (refer to Table 2-1)
- Reporting DCE status information to the MLC (refer to Table 2-1)
- Serializing data from the MLC for transfer to the communication line
- Deserializing data from the communication line for transfer to the MLC.

The HDLC hardware is configured to handle a four-bit data structure rather than the nominal eight-bit data structure. The only area that the adapter is required to deal with eight-bit data structures is at the HDLC/MLC data line interface. During discussions of the adapter hardware, any four-bit data structure is referred to as a "byte," and any eight-bit data structure (at interface level) is called a "word".

The descriptions in the following subsections are of an overview nature and all pertain to the interfaces and logic blocks depicted in Figure 2-2.

2.3.1 HDLC/MLC Interface

The interface signal lines between the HDLC adapter and the MLC are shown in Figure 2-3. A description of the application of these lines is provided in the MLC Product Manual (see list in Section I). Signal line mnemonics are shown as they are designated by the adapter (certain lines may be assigned different mnemonics within the MLC).

2.3.2 HDLC/DCE Interface

A diagram of the HDLC/DCE interface interconnections is shown in Figure 2-4. This figure identifies the interface lines, their direction, mnemonics, and applications. It represents the lines as seen by the HDLC and not the DCE, where many of the lines have different mnemonics. For a more detailed description concerning the implementation of each signal line, refer to Table 2-2.

This interface provides a link with the DCE and allows the HDLC hardware and firmware the capability of performing the central processor software designated operations. It provides the path necessary to supply the DCE with data, control, and timing pulses; it also supplies the HDLC with timing, data, and status from the DCE.

Table 2-1 HDLC/DCE Status Reporting Format

SOURCE	MNEMONIC	MLC INTERFACE DATA BIT	DESCRIPTION
DCE	DSRZZZ+	LADAT0+	Data Set Ready
DCE	CTSZZZ+	LADAT1+	Clear to Send
DCE	CDZZZZ+	LADAT2+	Carrier Detected
DCE	RIZZZZ+	LADAT3+	Ring Indicator
HDLC Firmware	RCF	LADAT4+	Receive End of Frame
HDLC Firmware	RCAS	LADAT5+	Receive Abort/Idle Link State
HDLC Firmware	RO	LADAT6+	Receive Overrun
HDLC Firmware	TU	LADAT7+	Transmit Underrun

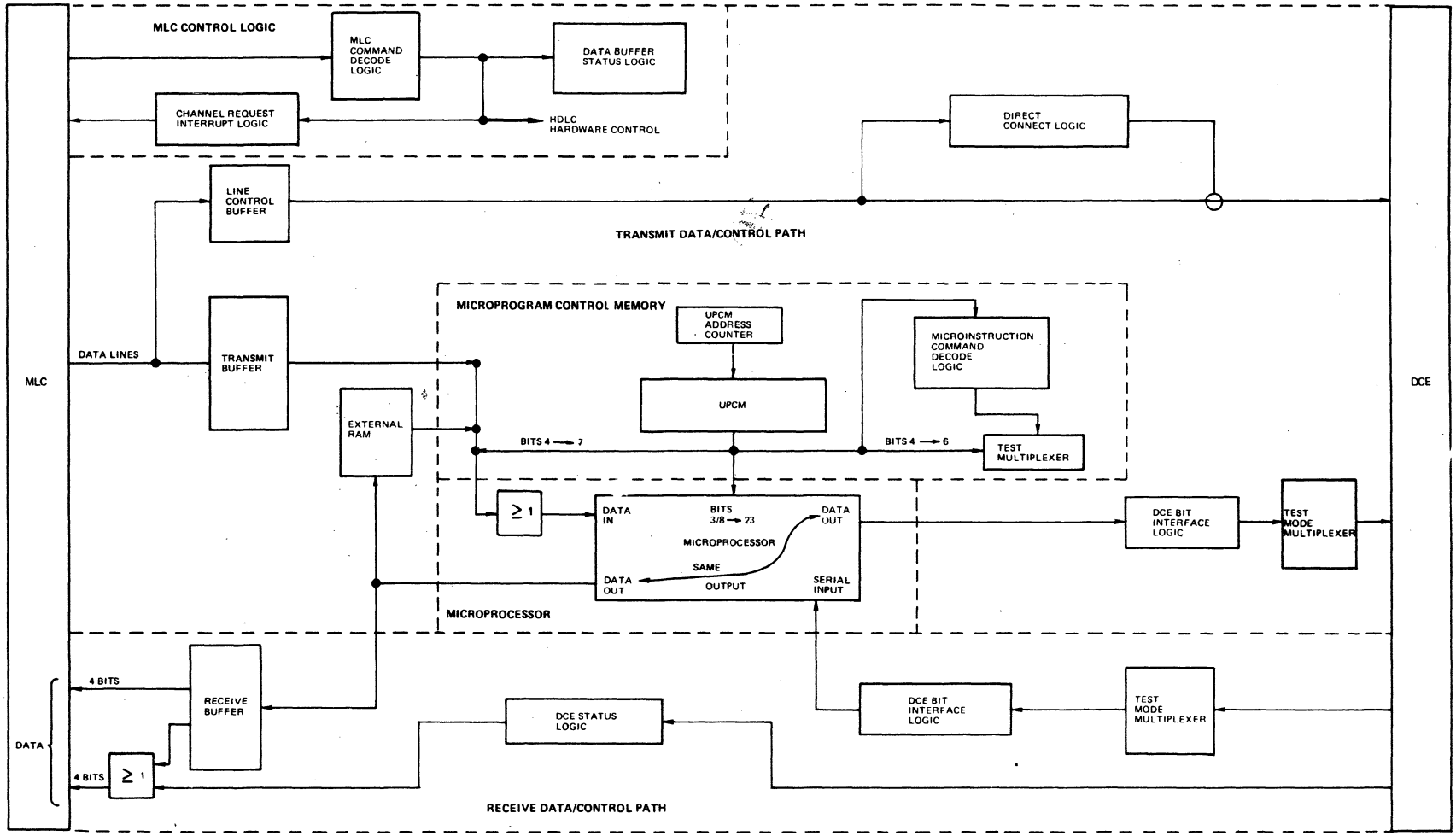


Figure 2-2 HDLC Adapter Intermediate Block Diagram

HONEYWELL PROPRIETARY AND CONFIDENTIAL

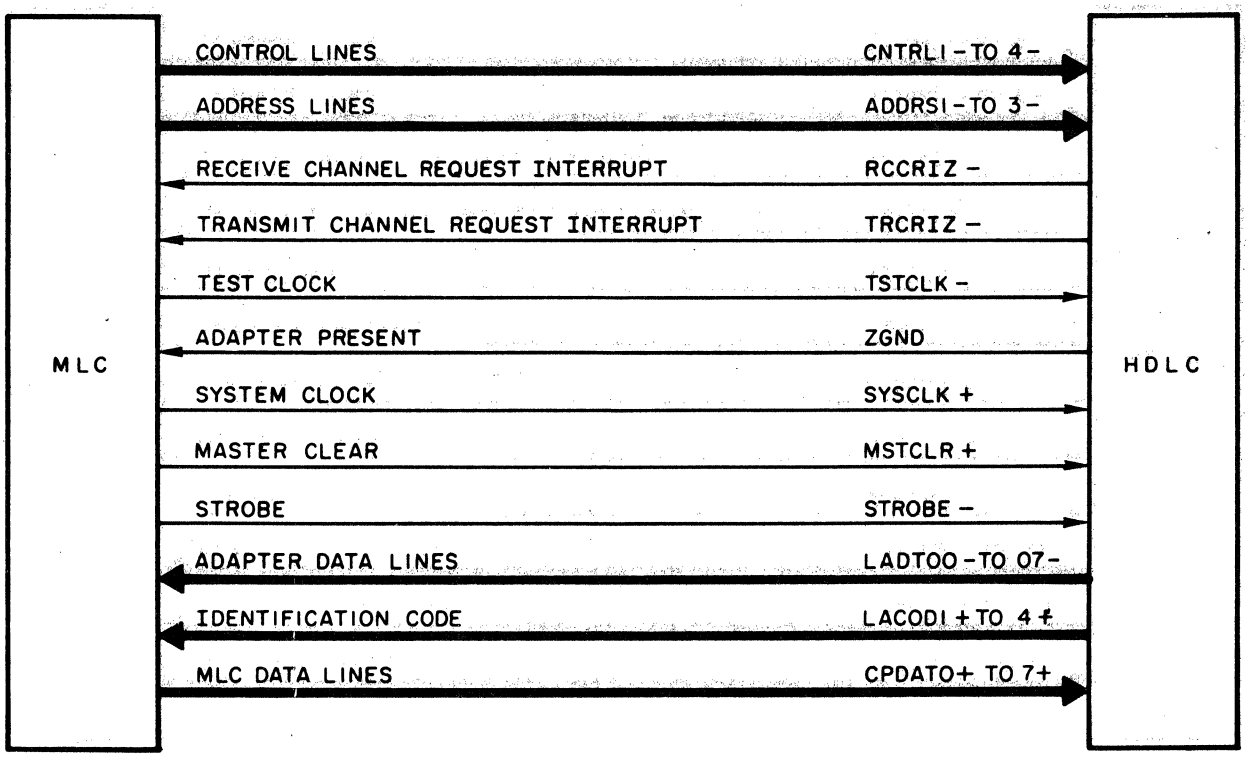


Figure 2-3 HDLC/MLC Interface

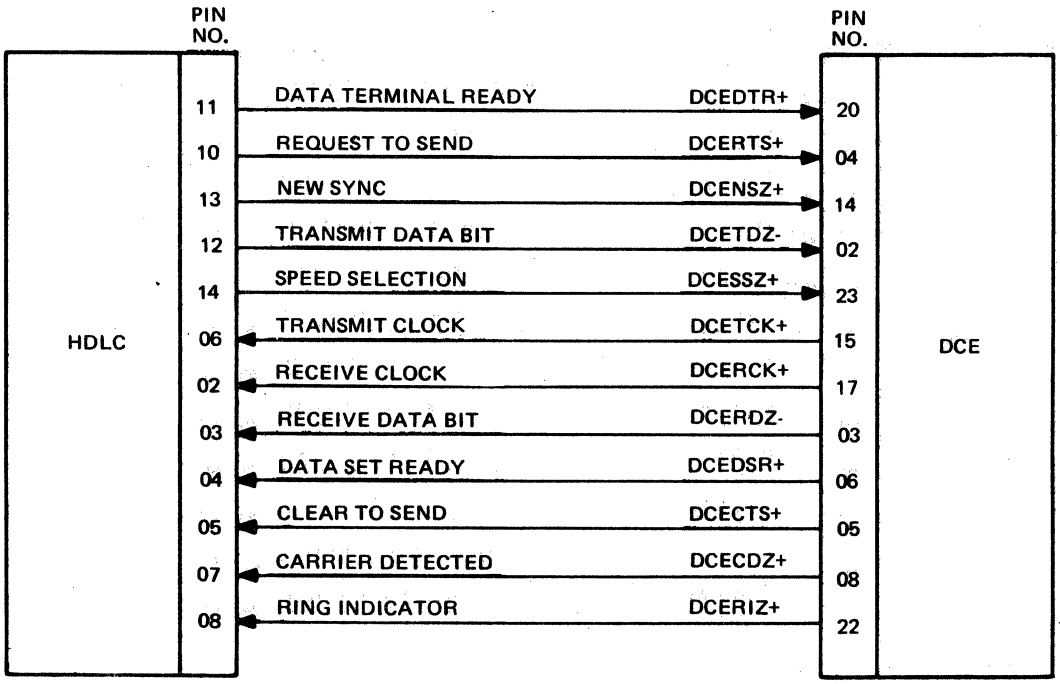


Figure 2-4 HDLC/DCE Interface

Table 2-2 HDLC/DCE Interface (Sheet 1 of 2)

TERM/MNEMONIC	DESCRIPTION
	OUTPUT LINES
Data Terminal Ready (DCEDTR+)	This signal occurs when bit 0 of the line control buffer (see Figure 2-13) is set or reset. This signal indicates to the DCE that the HDLC is ready to transmit or receive.
Request to Send (DCERTS+)	This signal occurs when bit 1 of the line control buffer (see Figure 2-13) is set or reset. This signal indicates to the DCE that the HDLC is ready to transmit.
New Sync (DCENSZ+)	This signal occurs when bit 2 of the line control buffer (see Figure 2-13) is set or reset and indicates a request from the HDLC to the DCE for a new sync signal during normal operation. In the direct connect mode, this signal is the result of bit 4 of the line control buffer and the test clock. It is then used as a clock signal.
Transmit Data (DCETDZ-)	This line is the serial data line from the HDLC to the DCE.
Speed Select (DCESSZ+)	During normal operation, this signal occurs when bit 3 of the line control buffer (see Figure 2-13) is set or reset and indicates to the DCE for it to select a specified rate of transfer during normal operation. In the direct connect mode, this signal is the result of bit 4 of the line control buffer and the test clock. It is then used as a clock for the transmitter.
	INPUT LINES
Transmit Clock (DCETCK+)	This signal (provided from the DCE) supplies the HDLC with the timing element necessary to perform transmit operations.
Receive Clock (DCERCK+)	This signal (provided from the DCE) supplies the HDLC with the timing element necessary to perform receive operations.
Receive Data (DCERDE-)	This line is the serial data line from the DCE to the HDLC.

Table 2-2 HDLC/DCE Interface (Sheet 2 of 2)

TERM/MNEMONIC	DESCRIPTION
	INPUT LINES
Data Set Ready (DCEDSR+)	This signal (from the DCE to the HDLC) indicates that the DCE is connected and available for operation.
Clear to Send (DCECTS+)	This signal is a response by the DCE resulting from the HDLC's request to send, if the DCE can accommodate the requested transmit operation.
Carrier Detected (DCECDZ+)	This signal (from the DCE to the HDLC) indicates that the basic carrier frequency of the communications line is present.
Ring Indicator (DCERIZ+)	This signal (from the DCE) indicates to the HDLC that a ringing condition is being received on the communications line which is directed to the HDLC.

2.3.3 MLC Control Logic

All data, control, configuration, and status information is transferred between the HDLC and the MLC via the interface data in and data out lines. This process is accomplished under control of the MLC, by the MLC Command Decoder which generates control signals as the result of the MLC strobe, address, and control lines (refer to Table 2-3). The outputs of this command decoder, within the MCL control logic, are distributed to the Data Buffer Status logic and to the Channel Request Interrupt logic. External to the MLC control logic, the outputs allow the reading, writing, and resetting of various hardware elements.

The Data Buffer Status logic portion of the MLC control logic is enabled as the result of the HDLC firmware command decoding and supplies signals to the firmware for interrogation. These signals indicate whether the transmit and receive data buffers are full or empty and are reset by the appropriate decode from the MLC command decode logic.

From a functional standpoint, the Channel Request Interrupt logic is also considered part of the MLC control logic. The requests are enabled as the result of HDLC firmware command decoding and when the HDLC requires servicing, and are sent to the MLC. When the MLC responds with the appropriate command, the requests are reset by outputs of the MLC command decode logic.

Table 2-3 MLC Address and Control Line Decode

ADDRESS LINES			CONTROL LINES				STROBE	MNEMONIC	COMMAND DESCRIPTION
1-	2-	3-	1-	2-	3-	4-			
L	H	L	L	L	X	X	Y	WRTRCB-	Write Transmit Control Byte
L	H	L	H	L	X	X	Y	WRLCBT-	Write Transmit Line Control Buffer
H	H	L	H	L	X	X	Y	WRLCBR-	Write Receive Line Control Buffer
H	H	L	L	L	X	X	Y	WRRCCB-	Write Receive Control Byte
L	H	L	H	H	X	X	Y	WRTRDB-	Write Transmit Data Buffer – Reset Data Buffer Empty (TRDBEZ-)
L	H	L	L	H	X	H	X	RDTRST-	Read Transmit Status
H	H	L	L	H	X	H	X	RDRCST-	Read Receive status
L	H	L	L	H	X	L	X	RDTRST-	Read Transmit Status – Reset Transmit Service Request (TRSRQZ+)
H	H	L	L	H	X	L	X	RDRCST-	Read Receive Status – Reset Receive Service Request (RRCSRQ-)
H	H	L	H	H	X	X	N	RDRCDB-	Read Receive Data Buffer
H	H	L	H	H	X	X	Y	RDRCDB-	Read Receive Data Buffer – Reset Data Buffer Full (RCDBFZ-)

L = Low
H = High
X = No effect
Y = Yes
N = No

2.3.4 Transmit Data/Control Path

The transmit data/control path is a composite of six basic fundamental logic areas; the Transmit Buffer, the Line Control Buffer, the External Random Access Memory (RAM), the Direct Connect Logic, a portion of the DCE Bit Interface Logic, and a portion of the Test Mode Multiplexer.

The transmit buffer (see Figure 2-5) has four word-size (8-bit) locations which provide storage for the transmit control byte, the receive control byte, and the transmit data word. Only two locations are employed for storage of this information, leaving two other locations unused at this time. Information is written into the transmit buffer from the MLC data lines by the decode of the appropriate MLC command. The four-bit wide output is enabled when the firmware executes the proper microinstructions.

The Line Control Buffer (LCB) is eight bits wide and is loaded from the MLC data lines when a write line control buffer command is decoded from the MLC. Four outputs of the LCB are sent directly to the DCE interface as control signals, two outputs indicate the mode of operation (direct connect or test mode), and the remaining two specify transmit and receive on or off.

The sixteen byte size location external RAM serves as an extension to the internal RAM located in the microprocessor (refer to subsection 2.3.7 and 2.5.1). The external RAM (see Figure 3-2 for topology) is controlled by the HDLC firmware, and information read from it or written into it from the microprocessor is the result of a firmware command decode.

The direct connect logic uses the direct connect output of the line control buffer in conjunction with the test clock supplied by the MLC to develop a direct connect mode clock. This direct connect mode clock is sent to the DCE on the new sync and speed select interface lines.

The DCE bit interface logic portion of the transmit data/ control path consists of the logic required to transfer the least significant bit of the microprocessor data output to the test mode multiplexer, an action controlled by the HDLC firmware and the transmit clock logic. The DCE bit interface logic also includes a transmit data ready indicator, for firmware testing, which is set at the same clock time that new data is being supplied to the test mode multiplexer.

The transmit portion of the test mode multiplexer supplies data to the DCE interface and transmit clock timing to the DCE bit interface logic. The multiplexer selects one of two inputs for the transmit data, the data from the DCE bit interface logic or a ground level if the HDLC is in test mode. To supply the transmit clock for the DCE bit interface logic, the test mode multiplexer selects the transmit clock supplied by the DCE under normal operations or the test clock from the MLC during test mode operations.

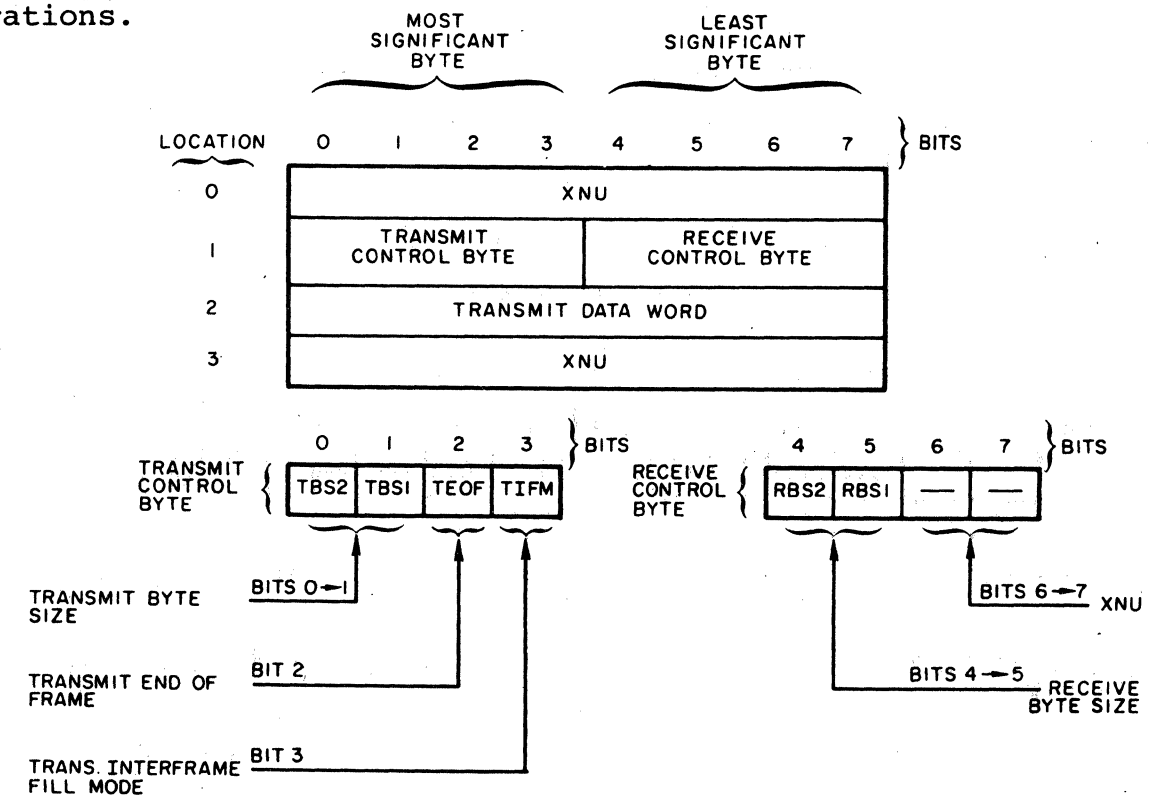


Figure 2-5 Transmit Buffer Topology and Byte Definition

2.3.5 Receive Data/Control Path

Four basic hardware logic elements are implemented to create the receive data/control path: a portion of the Test Mode Multiplexer, a portion of the DCE Interface logic, the DCE Status logic, and the Receive Buffer.

The receive portion of the test mode multiplexer supplies the DCE bit interface logic with receive clock timing and data. The receive timing supplied by the test mode multiplexer is either the result of the receive clock from the DCE during normal operation or the test clock sent by the MLC while in test mode. To provide data for the DCE bit interface logic, the test mode multiplexer selects one of two inputs: the data from the DCE or the transmit data from the transmit portion of the test mode multiplexer if the HDLC is in test mode.

The receive portion of the DCE bit interface logic within the receive data/control path consists of the logic necessary to transfer the receive data bit from the test mode multiplexer to the serial input of the microprocessor (refer to subsections 2.3.7 and 2.5.1). This action is accomplished under control of the HDLC firmware and the receive clock supplied by the test mode multiplexer. The DCE bit interface also consists of a receive data bit ready indicator, which is set at the same time the new receive data is being supplied to the microprocessor input, and is used for firmware examination.

The DCE status logic passes four bits of DCE status for transfer to the MLC. When a read status command is decoded in the MLC control logic, the four status signals (refer to Table 2-1) from the DCE are provided on bits 0 to 3 of the MLC/HDLC interface data lines.

The receive data/control path receives buffer (see Figure 2-6) is four word size (8 bit) locations which provide storage for the receive data word and the firmware generated frame status byte (refer to Table 2-1). Only one and one-half locations are utilized for storage of this information, leaving two and one-half locations unused at this time. Bytes of information are written into the receive buffer from the four-bit wide data output of the microprocessor when the appropriate HDLC firmware command is executed. The output of the receive buffer is enabled onto the HDLC/MLC interface lines when the correct MLC command is decoded by the MLC control logic (refer to Table 2-3). When a read receive data buffer command is received, the complete data word is reflected on the interface lines, whereas, in the case of a read status command, interface data lines 4 to 7 will reflect the receive buffer status byte, and lines 0 to 3 will contain the status from the DCE status logic (refer to Table 2-1).

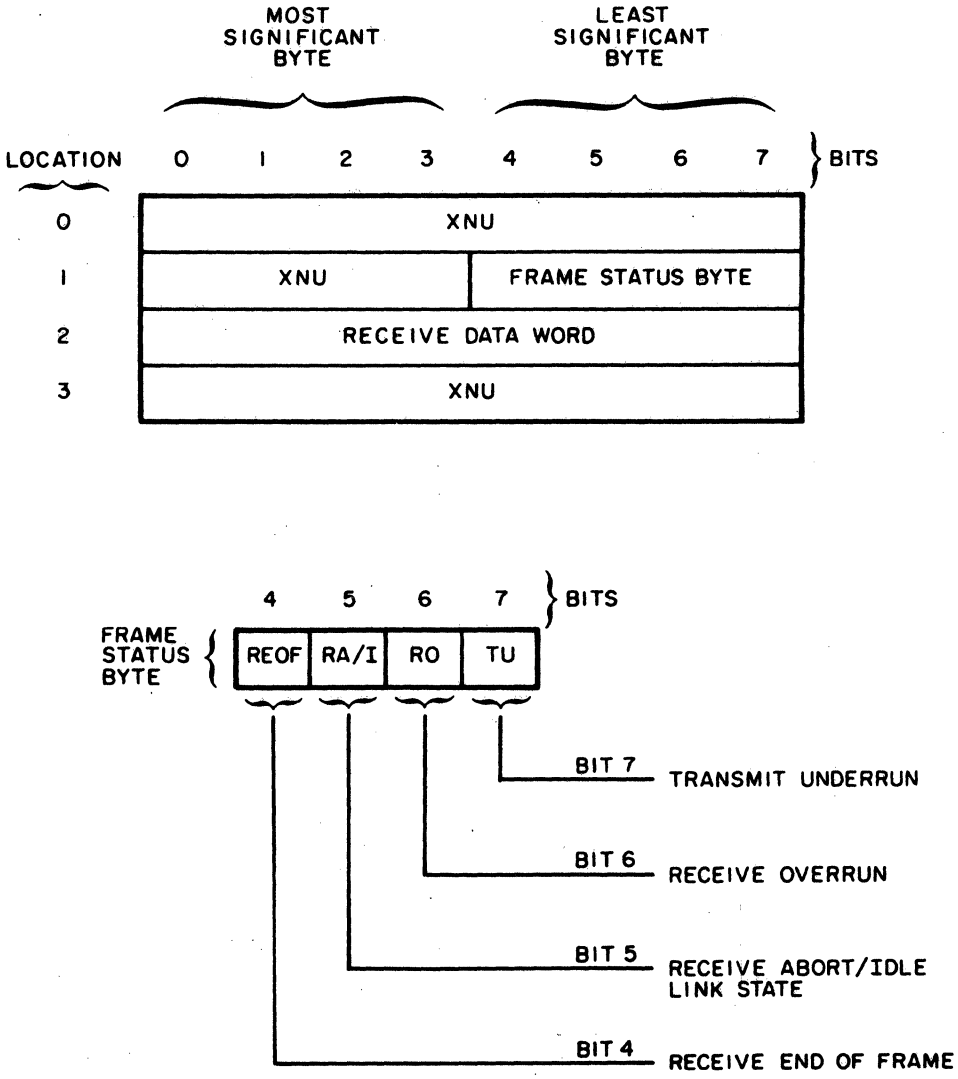


Figure 2-6 Receive Buffer Topology and Byte Definition

2.3.6 Microprogram Control Memory Logic

The Microprogram Control Memory Address Counter, the Microprogram Control Memory (UPCM), the Microinstruction Command Decode logic, and the Test Multiplexer are the four functional areas involved in making up the Microprogram Control Memory logic.

The ten-bit UPCM address counter specifies the location within the UPCM that contains the firmware command (refer to Section III of this manual for specific command information) presently being executed. The address counter increments at the start of each MLC clock cycle or is parallel loaded with ten of the outputs of the UPCM firmware command for a branch operation decode from the microinstruction command decode logic.

The firmware, sequences of microinstructions, for the HDLC is stored in the UPCM, which is arranged into 1024 twenty-eight bit locations. The address of the UPCM is provided by the UPCM address counter and the outputs of the addressed location are always read with the exception of bits 04 to 07. These four outputs are the firmware data inputs to the microprocessor and must be inhibited for certain types of firmware commands (Write External RAM, Read Transmit Buffer, and Read External RAM).

The microinstruction command decode logic is comprised of two major areas, the UPCM command decoder and the UPCM subcommand decoder. The UPCM command decoder evaluates bits 0 to 2 of the UPCM outputs to determine which of the eight possible instruction types the firmware is performing (see Section IV). When the command decoder detects an op code of either 0 (Subcommand instruction) or 7 (Branch and Subcommand instruction), the subcommand decoder is enabled (when bit 7 is Zero). The subcommand decoder then generates one of the eight available strobes (refer to Table 2-4), depending upon the decode of bits 4, 5, and 6 of the firmware command.

The test multiplexer portion of the UPCM logic provides the firmware with the capability of testing various hardware or operational conditions throughout the HDLC. When an op code of 3 (Test and Modify Next Instruction) is detected by the command decoder, the test multiplexer is enabled. Bits 4, 5, and 6 of the firmware command are then analyzed to select the test condition (refer to Table 2-5) and if the condition is true, the next firmware command is bypassed.

Table 2-4 UPCM Subcommand Decode

UPCM BITS					
4+	5+	6+	7+	MNEMONIC	SUBCOMMAND DESCRIPTION
L	L	L	L	STRDBE-	Set Transmit Data Buffer Empty
L	L	H	L	SRCDBF-	Set Receive Data Buffer Full
L	H	L	L	STRSRQ-	Set Transmit Service Request
L	H	H	L	SRCSRQ-	Set Receive Service Request
H	L	L	L	RSTRRY-	Reset Transmitter Ready
H	L	H	L	RSRCRY-	Reset Receiver Ready
H	H	L	L	WRTRBB-	Write Transmit Bit Buffer
H	H	H	L	TSTSYN-	Test SYNC

H = High
L = Low

Table 2-5 UPCM Test Conditions

UPCM BITS			MNEMONIC	TEST DESCRIPTION
4+	5+	6+		
L	L	L	TRDBEZ-	Transmit Data Buffer Empty
L	L	H	RCDBFZ-	Receive Data Buffer Full
L	H	L	TRSRQZ+	Transmit Service Request
L	H	H	RCSRQZ+	Receive Service Request
H	L	L	TRRDYZ+	Transmitter Ready
H	L	H	RCRDYZ+	Receiver Ready
H	H	L	ALUEZZ-	ALU Result Equal Zero
H	H	H	CTSVTH+	An "OR" of Clear to Send and Test Mode

H = High
L = Low

2.3.7 Microprocessor

All activity within the HDLC centers around the processing capabilities of the firmware-controlled microprocessor. The microprocessor supplies the HDLC with a sixteen location byte wide internal RAM, a byte wide shift register, and a byte wide ALU. The addressing within the microprocessor of its various components and the internal data paths supplied (see Figure 2-10) for data transfer allows the HDLC to serialize and deserialize data, store operational information, and perform logical operations on data and information bytes.

Data provided to the microprocessor can come from the UPCM (bits 4 to 7), from either byte of the transmit buffer (see Figure 2-5), or from the external RAM.

The firmware control of the microprocessor is a result of bits 8 to 23 of the firmware command. These bits determine the internal RAM read or write address, the function the ALU is to perform, the source of the ALU operands, and the destination of the ALU result.

2.4 FRAME STRUCTURE/OPERATIONAL CHARACTERISTICS/OPERATIONS

2.4.1 Frame Structure

The High-Level Data Link Control Adapter (HDLC) format is a bit-oriented data communications line control procedure. In this type format all serial-by-bit data transfers between the HDLC and the DCE are in frames. The overall format of the frame type utilized by the HDLC is shown in Figure 2-7 and a description of each field is supplied in the following paragraphs.

Flag Sequence

All valid frames begin and end with the flag sequence which is used for frame synchronization. This sequence consists of an eight-bit field which is formatted with a leading Zero bit followed by six One bits and a closing Zero bit (01111110). Continuous flag sequences can be transmitted between frames, or the closing flag sequence for one frame can be the opening flag sequence for the subsequent frame in the event that contiguous frames are being transmitted.

The HDLC transmitter automatically generates and sends complete flag sequences under three circumstances:

- To initiate a frame once the MLC has provided the first byte of data (D_1)
- To terminate the frame once the MLC has issued the end of frame
- To create fill when interframe fill mode specifies flag sequences and data for the next frame in not available.

The HDLC receiver continuously examines the input data stream on a bit-by-bit basis searching for a flag sequence to achieve synchronization. After detection of a flag sequence, it and any additional flag sequences are discarded with the first non-flag, non-abort (refer to subsection 2.4.2) sequence recognized as the first data byte of the text field for transfer to the MLC. When another flag sequence is detected, it is recognized as the closing flag sequence and indicates to the HDLC firmware that the preceding sixteen bits received were the FCS words. At this point the last data word and the two FCS words which are stored in the internal RAM are transferred to the MLC. The subsequent frame is processed in an equivalent manner; that is, all interframe flag sequences are discarded while waiting for the first non-flag, non-abort sequence.

Text Field

The text field includes all the bits between the opening flag sequence and the FCS field. Basically, this field can include any number or sequence of bits. However, since data is transferred in parallel-word (eight bits) format between the HDLC and the MLC, the text field is represented as being composed of data bytes (D_1, D_2,...D_n).

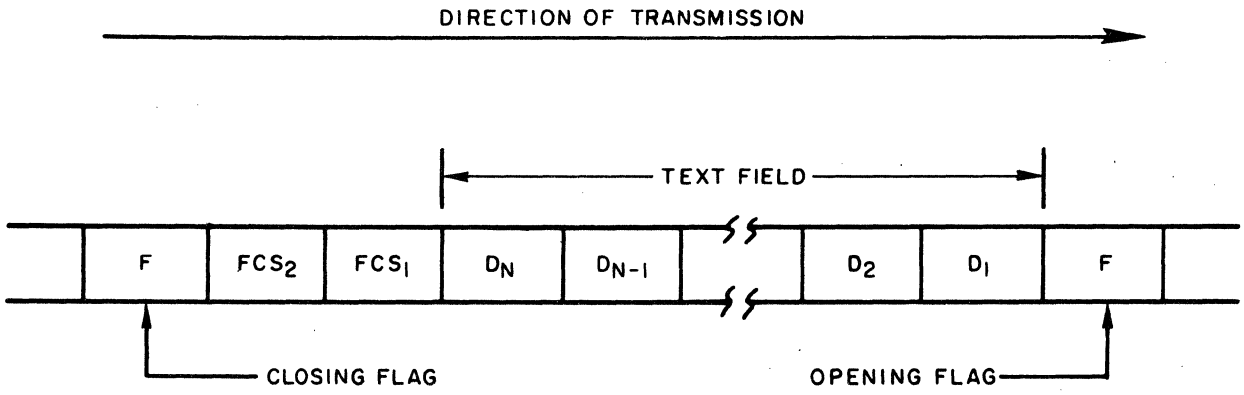
Frame Check Sequence (FCS) Field

A sixteen-bit double word FCS field is included as part of each frame to ensure the integrity of the text. Generation of the FCS field on transmit and verification using it on receive are functions of the MLC firmware and hardware.

The FCS field is unrecognizable from the text field on transmit operations with the exception that the MLC sets the end of frame bit (see Figure 2-5) in the transmit control byte prior to the MLC transferring the second FCS word (FCS_2) to the HDLC. This results in the appending of the closing flag to the frame after the sending of FCS_2 by the HDLC transmitter.

The HDLC receiver cannot distinguish the FCS field from the text until the closing flag is received. Once the closing flag is recognized, processing of the frame is completed in the following manner:

- The last data byte is right justified, if required, and it and FCS_1 and FCS_2 are moved from the microprocessor's internal RAM to the HDLC external RAM for temporary storage. This releases the microprocessor for processing of the next frame.
- The receiver closing flag bit (refer to Table 2-1) is set in the HDLC status word and the last data byte is transferred to the MLC.
- The FCS_1 word is transferred to the MLC.
- The FCS_2 word is transferred to the MLC.



F = FLAG SEQUENCE
D_X = DATA BYTES
FCS_1 = FIRST 8 BITS OF THE FRAME CHECK SEQUENCE
FCS_2 = SECOND 8 BITS OF THE FRAME CHECK SEQUENCE

Figure 2-7 Frame Format

2.4.2 Operational Characteristics

During and between operations performed by the HDLC, it is necessary to execute certain procedures or recognize specific sequences. This is necessary in order to maintain a meaningful dialogue with the DCE and is accomplished through the application of the procedures discussed in the following paragraphs.

Transparency

To prevent unwanted flag and abort sequences from occurring between the opening and closing flag sequences, a Zero bit insertion procedure is used, thereby providing complete text field and FCS field transparency. The Zero bit insertion process adds a Zero bit after every five contiguous One bits during a transmit and deletes the Zero bit after five contiguous One bits while receiving. This procedure applies to all the data between the opening and closing flag sequences, including the last five bits of the FCS field.

During transmit operations, the HDLC firmware continually examines the outgoing bit stream between the opening and closing flags, and whenever a series of five consecutive One bits is detected, a Zero bit is inserted prior to continuing with the next actual data bit (Zero or One).

When receiving, the HDLC firmware continually examines the incoming bit stream, commencing with the first actual data bit received after the opening flag. Five consecutive One bits followed by a Zero (111110XX) in the text or FCS field are recognized as being the result of Zero insertion during the transmit; the Zero bit is deleted (111111XX) by the adapter. A stream of five One bits followed by a One bit (six Ones) is recognized as the closing flag if the seventh bit is a Zero (1111110), or as an abort sequence if the next bit is a One (1111111).

Abort Sequence

The abort sequence is the procedure by which a station in the process of transmitting a frame, terminates the transfer in an abnormal manner, causing the receive station to ignore the message. The abort sequence is generated by transmitting a minimum of seven and a maximum of fourteen contiguous One bits with no Zero insertion. The reception of fifteen or more One bits by a receiver is interpreted as an abort sequence and the idle link state.

The HDLC transmit automatically generates and sends an eight-bit (11111111) abort sequence upon detection of an underrun condition during the transmission of a frame.

The HDLC receive firmware continually examines the receive data stream after the opening flag in search of an abort sequence (seven consecutive one bits). The HDLC can react in two ways to the abort sequence, depending upon the current position within the data stream:

- Provided no data transfers to the MLC have occurred for this frame, due to fewer than 25 bits of data between the opening flag sequence and the abort or because this frame overran the previous frame, the firmware reinitiates the examination of the data stream for a flag sequence or an idle link state.
- If data transfers to the MLC have occurred for this frame, the firmware sets abort in the status word (refer to Table 2-1), initiates a service request, and returns to examining the data stream for a flag sequence or an idle link state.

Interframe Time Fill

Between frames the HDLC transmitter automatically generates and sends complete eight-bit flag sequences (11111101) or abort sequences (11111111) as determined by the interframe fill mode indicator located in the control word of the transmit buffer (see Figure 2-5). Interframe fill continues until the MLC provides data for the subsequent frame.

Flag or abort sequences received from the DCE are always discarded; therefore, no data transfer to the MLC will occur during interframe time fill sequences. When flags are received as interframe time fill, synchronization is maintained, and the HDLC receiver continually searches for the first non-flag, non-abort sequence (first data character). Where the line is in the marking state (all-One continuous abort sequence), synchronization is lost and the HDLC receiver scans for a flag sequence. Should the marking state continue for 15 or more consecutive One bits, the HDLC interprets this as the idle link state.

Idle Link State

The communication line is in the idle link state when a continuous One bit state (marking) presides for a minimum of 15 bit times. The communication line is in the active state when a frame, an abort sequence, or a flag interframe time fill sequence is occurring.

The MLC causes the idle link state to be entered by setting the interframe fill mode bit in the transmit control word (see Figure 2-5) to specify an abort sequence and then not providing data before 15 bit times have elapsed. Since interrupts are generated at word boundaries between frames, the MLC determines how many aborts have been transmitted, and hence whether the idle link state has been entered, by counting the service requests.

The HDLC receive firmware continuously examines the input data stream searching for 15 consecutive one bits. Upon detection of the idle link state, the HDLC sets abort/idle in the status word (refer to Table 2-1) and generates an interrupt to the MLC. After an idle link state service request is made, the HDLC receiver does not initiate any further idle link state interrupts until another frame is being processed and at least one transfer to the MLC has occurred.

Invalid Frame

In general, an invalid frame is one which is not bounded on each extremity by a flag sequence (i.e., terminated by an abort sequence) or one which contains fewer than 32 bits of data and FCS, not counting the Zero bits inserted for transparency, between flag sequences. The HDLC discards any frames which fit the description of an invalid frame.

The HDLC can transmit either type of invalid frame: an abort sequence due to an underrun condition or a short frame for any of the following reasons:

- The MLC sets end of frame in the transmit control byte (see Figure 2-5) prior to transferring four words to the HDLC.
- The MLC provides at least four words of data but the byte size is too small.
- The MLC generates a Master Clear signal to the HDLC.

An invalid frame can also occur on a transmit if the DCE control lines (from the line control buffer, described in subsection 2.3.4) are changed incorrectly.

Invalid frames can occur on a receive operation as the result of detecting an abort sequence. A description of the abort sequence is found in subsection 2.4.2. Invalid frames while receiving also result from short frames with 24 bits or fewer frames. These frames are discarded by the HDLC with no notification from the MLC. Those frames with from 25 to 31 bits must be discarded by the MLC, since the HDLC does not provide enough storage for this number of bits, and should cause an FCS error to be detected by the MLC.

2.4.3 Operations

Transmit Operation (Figure 2-6)

After subsystem power-up when master clear is received from the MLC, an abort sequence of eight Ones is generated and loaded into the shift register (locations within the external RAM). If Clear to Send from the DCE is set, Test Mode is reset, and Transmit On is set, a channel request interrupt is generated to notify the MLC that the data buffer is available for the first word of the frame.

During the time prior to the first frame or between frames, flag or abort sequences as determined by the interframe fill mode of the control byte (see Figure 2-5) are generated, loaded into the shift register at each byte boundary, and right shifted into the bit buffer with each transmit clock received from the DCE. At the completion of each eight-bit sequence, the data buffer is examined to see if the first character of the frame is available.

If no data is present, another fill character is loaded into the shift register and processed in the same way as previously described. After this fill character is processed, if the previous channel request interrupt has been reset by the MLC, another channel request interrupt is generated to request the data.

If data was supplied for the first channel request interrupt, the character is transferred from the data buffer to the shift register and is right shifted with each transmit clock from the DCE. If an abort fill was in effect, a flag is loaded into the shift register and transmitted prior to the processing of the data.

The data in the shift register and DCE bit buffer is shifted at each transmit clock cycle, and Zeros are inserted for transparency as required. Each shift is counted (except inserted Zeros) by decrementing the byte size counter in the internal RAM (refer to Section III). When the counter equals Zero, the entire character has been shifted out, and the shift register is ready for the next word of data.

At each word boundary, the data buffer is examined to determine whether new data has been provided. If new data is available, it is transferred into the shift register, the byte size counter is restored, and another channel request interrupt is generated to request data from the MLC. All byte size changes are made by the MLC after the channel request interrupt is generated by the HDLC and prior to transferring data to the HDLC. This ensures correct detection of an underrun condition if one were to occur.

When the shift register is empty and if no data is provided by the MLC, an underrun condition has occurred, and an eight-bit abort sequence is generated and loaded into the shift register. After the late data has been provided and/or the channel request interrupt has been processed, underrun is set in the status byte of the receive buffer (see Figure 2-6), and a channel request interrupt is generated to notify the MLC of the underrun condition. After the eight clock cycles required to transmit the abort sequence, a normal interframe fill is initiated (refer to subsection 2.4.2).

After each character is transmitted, the end-of-frame bit in the transmit control byte (see Figure 2-5), which is set by the MLC prior to supplying the final data character, is examined by the HDLC to determine whether it was the last character of the frame. Once the last character is transmitted, the closing flag is generated, loaded into the shift register, and transmitted a bit at a time for each transmit clock cycle. At this point, the interframe time fill is generated (refer to subsection 2.4.2) or, in the event that new data has been supplied by the MLC, the closing flag of this frame is the opening flag for the new frame.

Receive Operation (see Figure 2-9)

After subsystem power-up, when a Master Clear signal is received from the MLC, the receive firmware starts the serial

shifting of data into the receive shift register at a rate determined by the DCE-provided receive clock. This is one in search of the idle link state (15 contiguous One bits) or a flag sequence. If the idle link state is detected, the abort/idle link state is set in the status word and, assuming Receive On is set, a channel request interrupt is generated to inform the MLC of the receive status. No further data processing is attempted until an opening flag sequence is detected by the receiver.

Once a flag sequence is received, frame synchronization is achieved, and the input data is examined every eight bits in search of a non-flag or non-abort sequence. The first such sequence is the initial character of the frame. If an abort sequence is received, frame synchronization is lost, and the input data is again examined on a bit-by-bit basis for a flag sequence or an idle link state. When continuous flag sequences are received, synchronization is maintained and the flags are discarded.

Each receive clock pulse causes the receive data to be shifted one bit position down the receive shift register. The inserted Zero deletion process removes Zeros which were inserted for transparency during the transmit operation. During this input operation before the Zeros are deleted, testing is performed for flag or abort sequences. When either sequence is detected prior to 24 shifts of the receive shift register, the frame is considered invalid and the scan for a new frame is initiated. After 24 shifts the first character is assembled in the data register, and the contents are transferred to the receive buffer. Once the data is in the receive buffer, a channel request interrupt is generated, causing the MLC to examine the status word and input the first data character.

With each transition of the receive clock, the entire 24-bit shift register is shifted one bit position unless an inserted Zero is being deleted. To delete an inserted Zero, only the flag register is shifted, causing the Zero in the right-most bit position to be discarded. The number of data bits shifted into the receive shift register, since the last character was transferred to the MLC, is compared with the byte size specified in the receive control byte between each shift. When the two values are equal, a complete character is in the data register. The data character is right justified if necessary, and if the receive buffer is empty, the data is loaded into it, and a channel request interrupt is made to send the data to the MLC. The MLC has one byte size time to input the data and prevent overrun. If the MLC has not removed previously the data word from the receive buffer, an overrun has occurred, and no additional channel request interrupts to transfer data are made for this frame.

As the data shifting, assembling, and transfer process continues, the input data stream is monitored for a flag or abort sequence. The detection of either sequence indicates the end of frame and can occur at any time, regardless of character boundaries.

When a frame is terminated by an abort sequence, end of frame, closing abort/idle link state, and overrun, if applicable, are set in the status word, and a channel request interrupt is generated. If a previous channel request interrupt has not been reset by the MLC, this final transfer is deferred until the MLC completes processing of the original channel request interrupt. This same procedure is performed in the event that an overrun frame is terminated by a flag sequence.

To complete the processing of a normal frame (no overrun or abort sequence), the following sequence of events occurs:

- The final 24 bits of the frame (last data character and two FCS words) are transferred to the history registers for temporary storage to free the receive register for the subsequent frame.
- The data character is moved from the history register into the receive buffer, end of frame is set in the status word, and a channel request interrupt is generated.
- Once the MLC has processed the previous channel request interrupt, the first word of the FCS sequence is transferred from the history register to the receive buffer, and a channel request interrupt is generated.
- Finally, after the previous channel request interrupt is processed by the MLC, the second word of the FCS is moved from the history registers to the receive buffers and the last channel request interrupt for this frame is generated.

After the channel request interrupt for the last FCS word is reset by the MLC, the HDLC resets the status word, and processing of this frame is complete.

While the information in the history registers is being transferred to the MLC at a rate determined by the MLC, it is possible that the next frame is being shifted into the receive shift register. Should the first data byte be assembled before the transfers from the history registers are complete, an overrun occurs for the second frame, and no data transfers to the MLC for that frame are performed. The MLC is informed of the status of the second frame once the last transfer from the history registers is complete and the second frame completed.

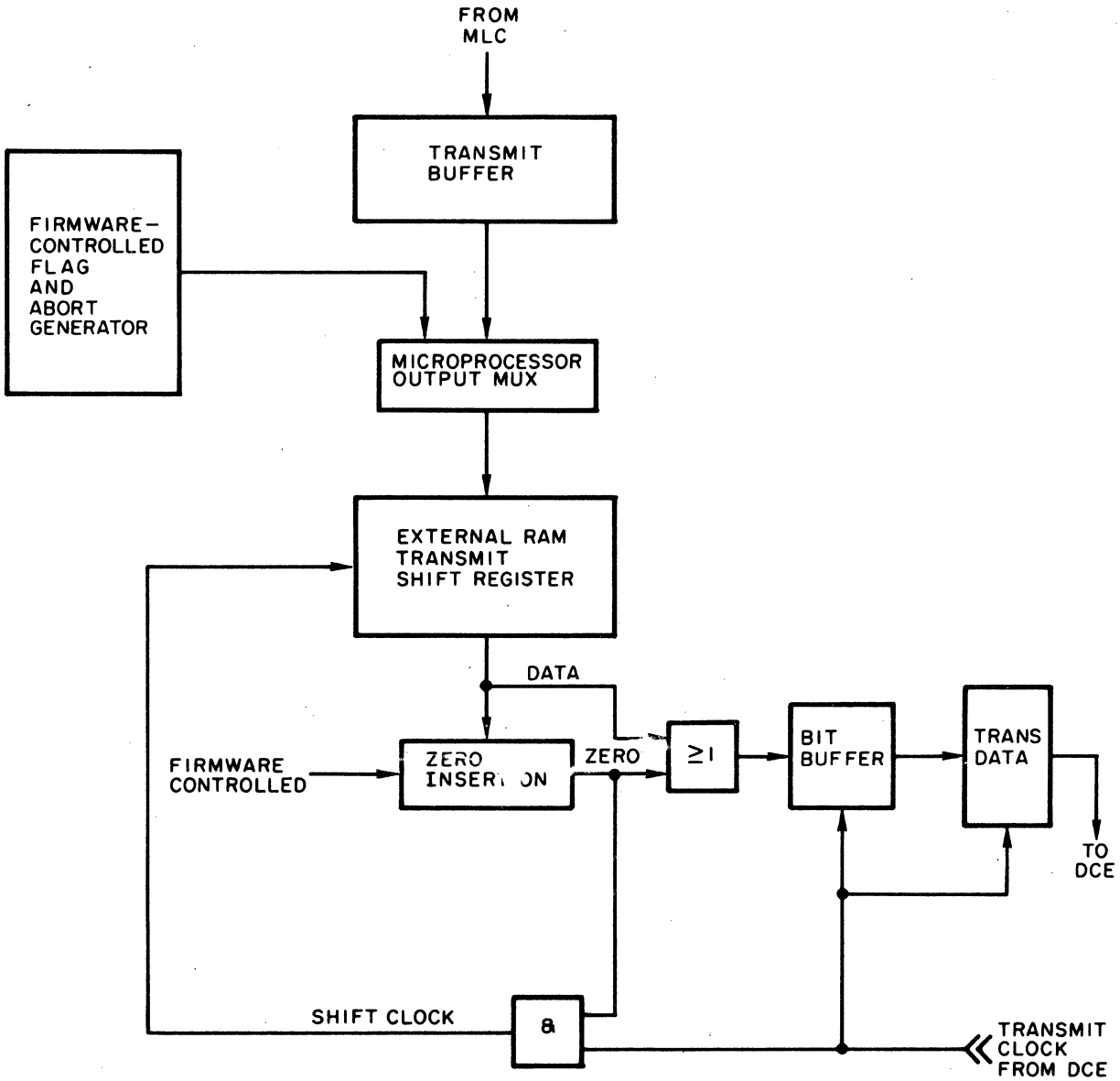


Figure 2-8 Logical Transmit Data Flow

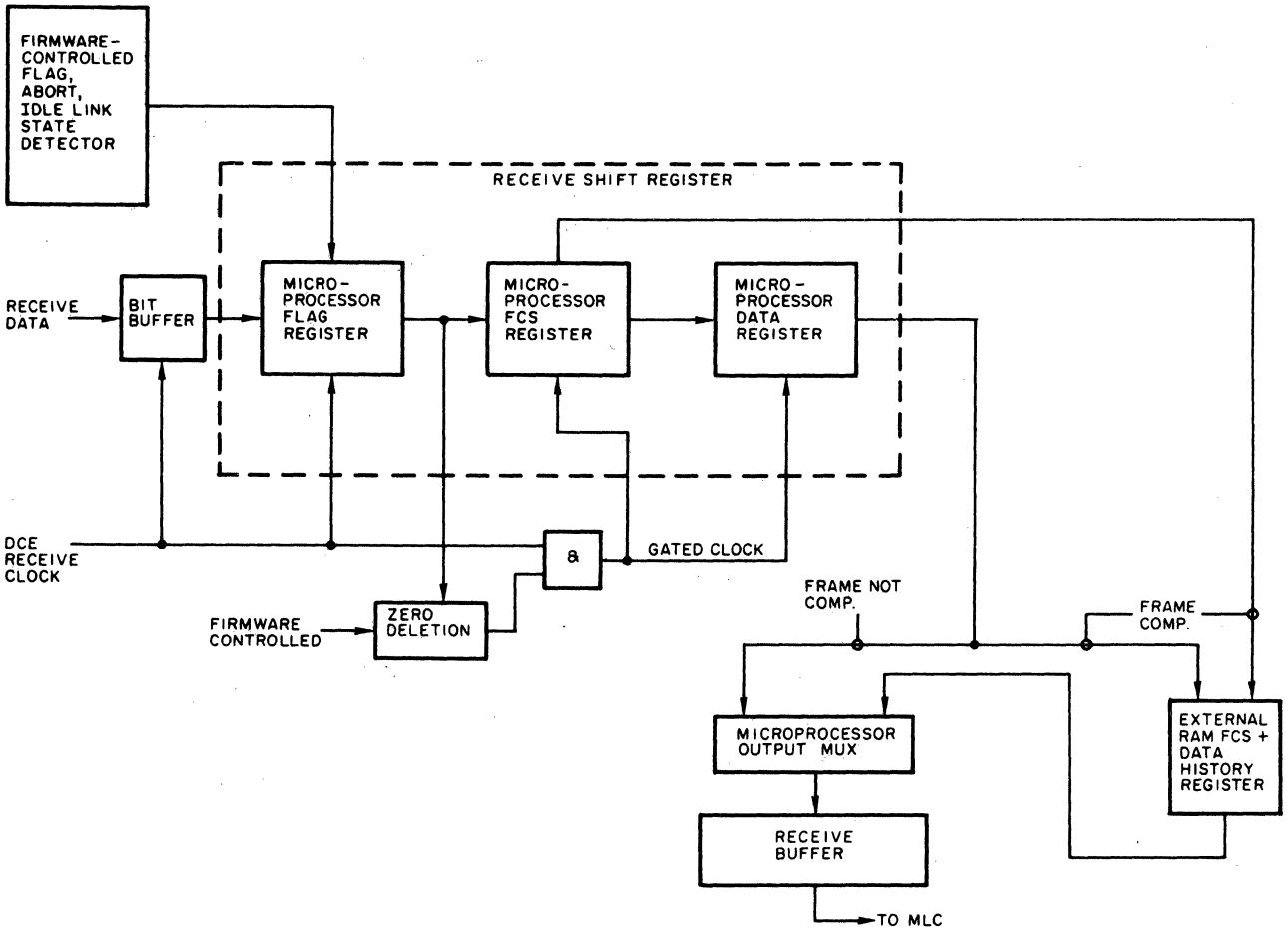


Figure 2-9 Logical Receive Data Flow

2.5 INTERMEDIATE HARDWARE DESCRIPTION

Figure 2-2 is an intermediate level block diagram of the HDLC adapter logic. This illustration is designed to guide the user through the detailed logic block diagrams (LBDs) contained in the HDLC Adapter Reference Manual, and is the overview diagram on which the remainder of the text contained in this section is based.

The transmit and receive operations "commonly shared logic areas", MLC Control logic, Microprogram Control Memory logic, and the Microprocessor logic, are each discussed from a functional point of view. The Transmit Data/Control Path and the Receive Data/Control Path are discussed separately and each from an application standpoint (transmit/receive operation). As each path is described, references are made to the appropriate "commonly shared logic area", as required, to complete the HDLC functional description.

2.5.1 Microprocessor Logic (see Figure 2-10)

The firmware-controlled microprocessor contains the elements necessary for all parallel to serial and serial to parallel data conversions. It also contains the storage necessary for data, FCS words, the flag word, counters for shifts and zero deletion, and receive and storage register status (refer to subsection 3.2 for storage application).

The microprocessor has two data inputs: a byte-wide (four-bit) input (UPIDB0+ to 3+) for parallel data and a single bit input (S1 on the "Q" register selector) for serial data (RCDATZ+). Access to the byte wide data input is from the left half of the transmit buffer (INBFR0+ to 3+), the right half of the transmit buffer (INBFR4+ to 7+), the external RAM (SPMEM0- to 3-), and a constant field (UPCM04+ to 7+) from the output of the microprogram control memory. The serial input (S1) is supplied by the receive data logic from the output of the data driver (see Figure 2-14).

The microprocessor data output (UPODB0+ to 3+) is a byte-wide path which receives its information through the output data selector.

Control of all the elements in the microprocessor is provided by firmware through the use of sixteen bits supplied within the firmware command. The firmware control field (UPCM08+ to 23+) is divided into five individual elements: internal RAM "A" address, internal RAM "B" address, ALU function control field, ALU source operand field, and the ALU result destination.

Data in any of the 16 byte locations of the internal RAM can be read as controlled by the "A" address (UPCM08+ to 11+) input to the microprocessor. Simultaneously, data in any of the 16 byte locations of the internal RAM as defined by the "B" address (UPCM12+ to 15+) can be read. The data from each of the addressed

locations is latched into its respective "A" or "B" latch. The same address can be provided to both the "A" and "B" address fields in which case the identical data is latched into both the "A" and the "B" latches.

When writing into the internal RAM, the new data is always written into the byte location specified by the "B" address field. The new data to be written is selected by the RAM data select three-input multiplexer. This configuration allows the ALU data output (F0 to F3) to be shifted left or right one bit position using an external shift bit input (S3, S4) or to be written directly into the RAM with no shift. The selection of the appropriate data input is determined by the configuration of the firmware ALU destination field (UPCM22+ and 23+) (refer to Table 2-6).

The ALU can perform three arithmetic and five logic operations on the two byte inputs (R0 to R3 and S0 to S3). The "R" input is selected by a two-input multiplexer, while the "S" input is selected by a three-input multiplexer, with the proper inputs selected by the ALU source operand field (UPCM19+ to 21+) (refer to Table 2-7). Both multiplexers also have an inhibit capability which prevents any data selection, effectively supplying a "zero" operand to the ALU. The three arithmetic and five logic functions of which the ALU is capable are a result of the coding in the ALU function control field (UPCM16+ to 18+) listed in Table 2-8. The ALU data output (F0 to F3) is routed to various elements throughout the microprocessor. It can be used as the parallel data output of the microprocessor, it can be stored in the RAM as previously described, and it can be stored in the "Q" register as determined by the ALU destination control field decode shown in Table 2-6.

The "Q" register (Q0 to Q3) is a single byte storage element whose input is provided by a three-input "Q" register selector (D0 to D3). In the no-shift mode, the selector enters the ALU data (F0 to F3) directly into the "Q" register (refer to Table 2-6). In either the right shift or the left shift mode, the selector provides the data output from the "Q" register appropriately aligned, using S1 or S2 as the external shift input.

The final logic area of the microprocessor is a two-input output-data selector which is one byte wide (Y0 and Y1) and supplies parallel data to the HDLC (UPODB0+ to 3+). This selector provides either the "A" latch data (A0 to A3) or the ALU output (F0 to F3) to the microprocessor output bus. The selection is a function of the ALU destination field decode (refer to Table 2-6).

Appendix A contains a logic block diagram of the Type 2901 microprocessor chip used in the HDLC. The pin numbers, input/output mnemonics, and the chip specification identifiers are illustrated to aid understanding of the internal chip versus the external HDLC logic relationship.

HONEYWELL PROPRIETARY AND CONFIDENTIAL

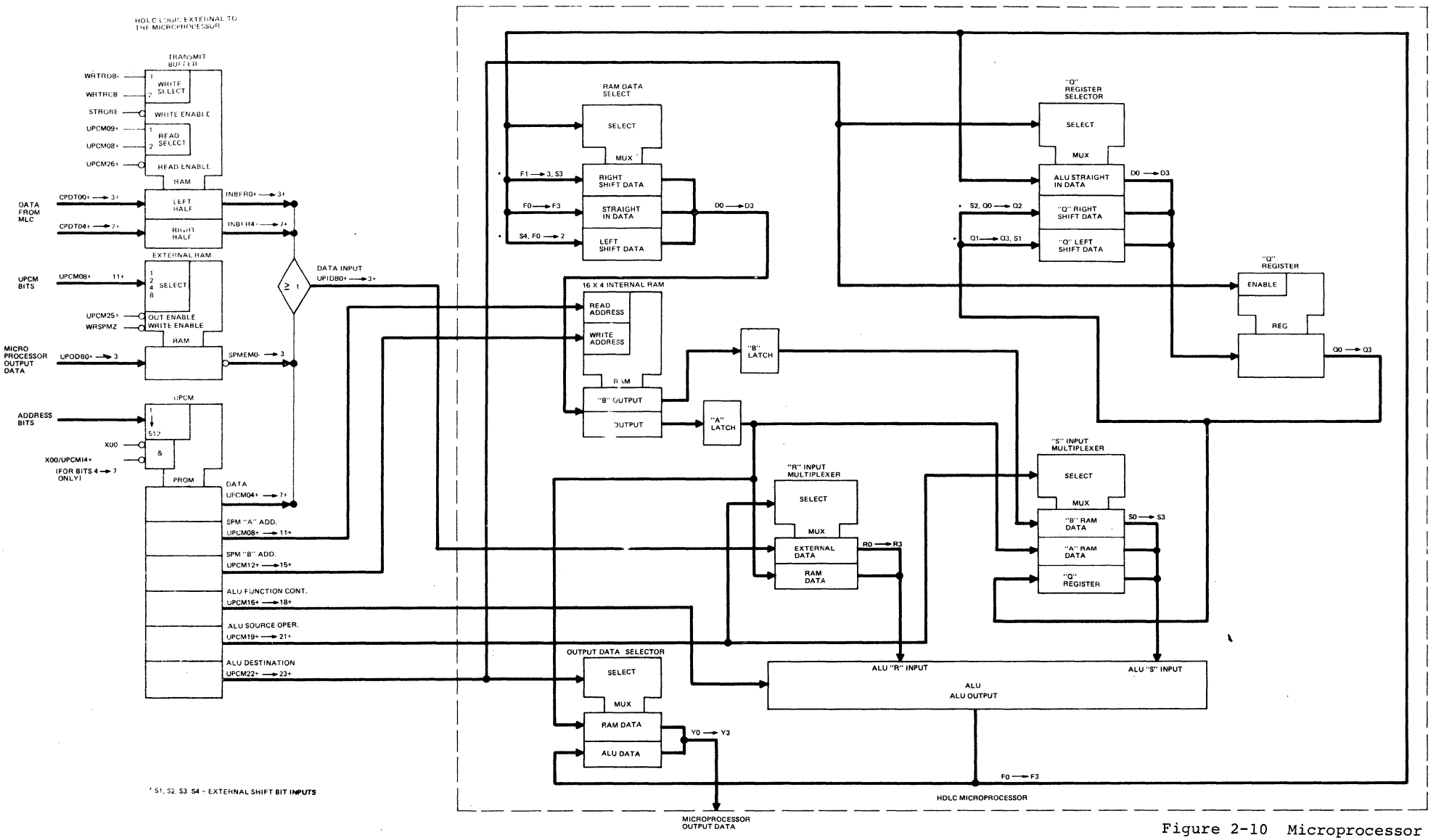


Figure 2-10 Microprocessor
Funtional Block Diagram

HONEYWELL PROPRIETARY AND CONFIDENTIAL

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table 2-6 Microprocessor ALU Destination Control Field Decode

UPCM BITS		OCTAL CODE	RAM FUNCTION		"Q" REGISTER FUNCTION			MICROPROCESSOR DATA OUTPUT
					DATA INPUT			
22	23		SHIFT	WR. RAM		SHIFT	WR. "Q"	
0	0	0	—	No	ALU	None	Yes	ALU
0	1	1	None	Yes	—	—	No	RAM "A" Latch
1	0	2	Right	Yes	"Q"	Right	Yes	ALU
1	1	3	Left	Yes	"Q"	Left	Yes	ALU

Table 2-7 Microprocessor ALU Source Operand Field Decode

UPCM BITS			OCTAL CODE	ALU SOURCE OPERANDS	
				"R" MUX	"S" MUX
19	20	21			
0	0	0	0	"A" Latch	"Q" Register
0	0	1	1	"A" Latch	"B" Latch
0	1	0	2	Zero	"Q" Register
0	1	1	3	Zero	"B" Latch
1	0	0	4	Zero	"A" Latch
1	0	1	5	Parallel Data Input	"A" Latch
1	1	0	6	Parallel Data Input	"Q" Register
1	1	1	7	Parallel Data Input	Zero

Table 2-8 Microprocessor ALU Function Control Field Decode

UPCM BITS			OCTAL CODE	ALU FUNCTION
16	17	18		
L	L	L	0	R plus S
L	L	H	1	S minus R
L	H	L	2	R minus S
L	H	H	3	R OR S
H	L	L	4	R AND S
H	L	H	5	R AND S
H	H	L	6	R XOR S
H	H	H	7	R XNOR S

2.5.2 MLC Command Decode and Control Logic (see Figure 2-11)

The MLC command decode and control logic allows for the transfer between the HDLC and the MLC of all data, configuration, control, and status information. This is accomplished via the decode of the MLC address lines (ADDRS1- to 3-) and control lines (CNTRL1- to 4-) in conjunction with the strobe (STROBE-) (refer to Table 2-3).

Three outputs of the MLC command decoder are sent to the transmit buffer to enable the writing of the correct location. When a Write Transmit Control Byte command is decoded (WRTRCB-), the left half of the control word location (see Figure 2-5) is enabled, and the information on the four MSBs of the MLC data lines (CPDT00+ to 03+) is written into the buffer at STROBE- time. When a Write Receive Control Byte command is decoded (WRRCCB-), the right half of the control word location is enabled, and the information on the four LSBs of the MLC data lines (CPDT04+ to 07+) is written into the buffer at STROBE- time. The last decode which writes into the transmit buffer is a Write Transmit Data Buffer (WRTRDB-) command. This command enables both left and right bytes of the data location word, and the data on the MLC data lines (CPDT00+ to 07+) is written into the buffer at STROBE- time. WRTRDB-, in addition to writing into the transmit buffer, resets the Transmit Buffer Empty (TRDBEZ-) flip-flop. TRDBEZ- is tested by the firmware to determine if another word of data is available for transmission to the DCE.

Two more of the MLC command decodes are used to enable the writing of the line control buffer. These two commands, Write Line Control Buffer for Transmit (WRLCBT-) and Write Line Control Buffer for Receive (WRLCBR-), are gated together to produce the Write Line Control Buffer (WRLCBZ-) command if either command is decoded. WRLCBZ- is then ANDed with STROBE- to generate Write Line Control Buffer Strobe (WRLCBS+), which causes the data on MLC data lines 0 to 5 (CPDT00+ to 05+) to be loaded into the line control buffer and bits 6 and 7 to be loaded into the transmit on (TRONZZ+) and the receive on (RCONZZ-) flip-flops (both are part of the line control buffer).

Each of the remaining three commands causes an input of data to the MLC and can result in setting or resetting of certain control functions. The Read Receive Data Buffer (RDRCDB-) command enables both bytes (left and right) of the data word location onto the MLC input data lines (LADAT0+ to 7+). This command (RDRCDB-) also resets the Receive Data Buffer Full (RCDBFZ-) flip-flop. RCDBFZ- is tested by the HDLC firmware to determine if another word of data has been transferred to the MLC.

The Read Transmit Status (RDTRST-) and the Read Receive Status (RDRCST-) commands are gated together to develop the status allow (Read Status, RDSTAT-) if either command is decoded. This results in enabling the status location (see Figure 2-6) in the right byte position (RDOBRH-) of the receive buffer and the enabling of the four DCE status drivers (see Figure 2-14) to be sent to the MLC via the input data lines (LADAT00+ to 07+). The two status read commands (RDTRST-, RDRCST-), in conjunction with control line 4 (CNTRL4-), also develop resets (RSTRSR+, RSRCSR+) for their respective service request flip-flops (TRSRQZ+, RCSRQZ+).

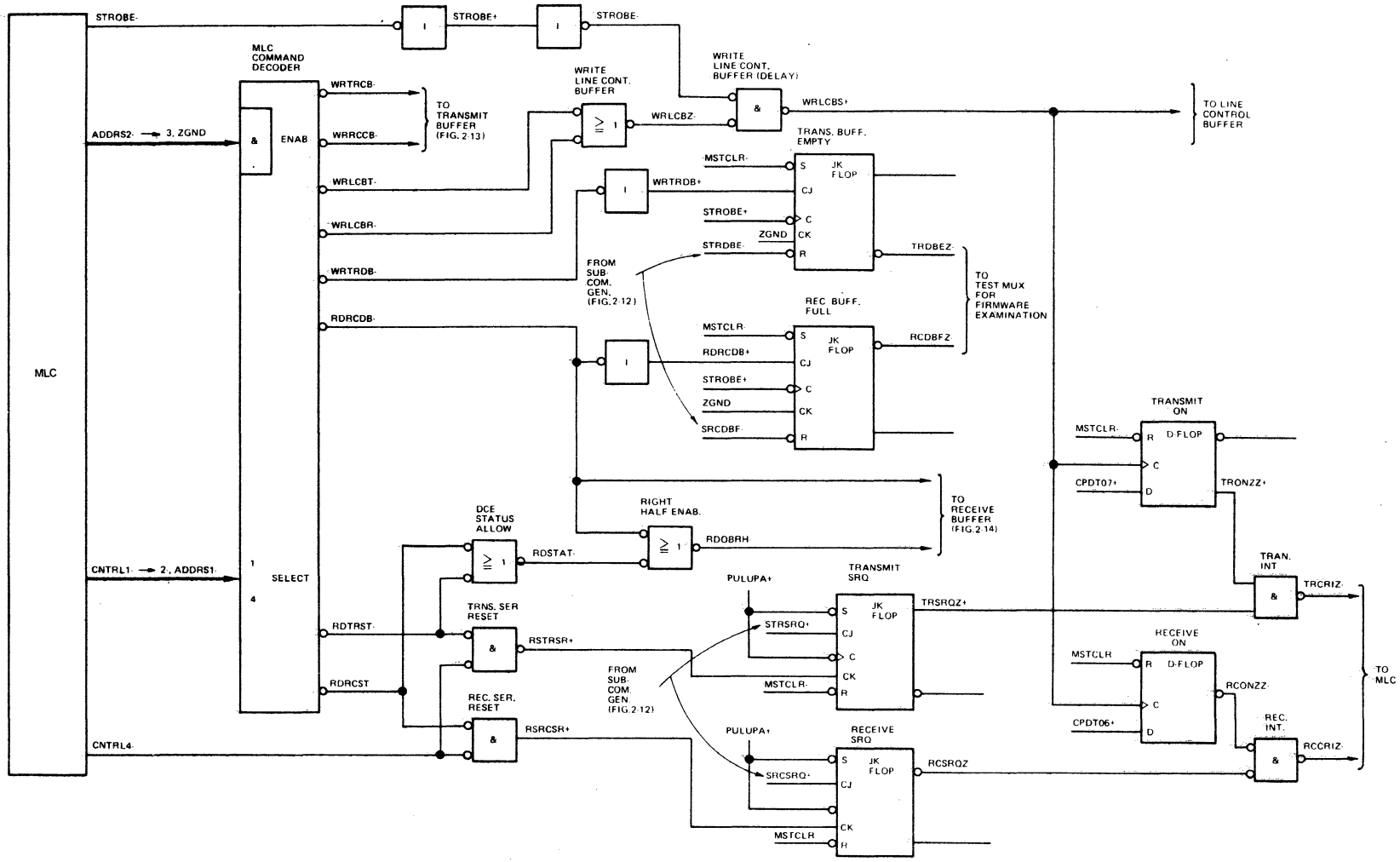


Figure 2-11 MLC Command Decode and Control Signal Generation

HONEYWELL PROPRIETARY AND CONFIDENTIAL

2.5.3 Microprogram Control Memory Logic (see Figure 2-12)

The Microprogram Control Memory (UPCM) logic contains the storage and logic necessary for firmware to perform efficient control of data transfers.

The UPCM address counter (UPAC00+ to 09+) is a 10-bit up counter which is reset to zero when a Master Clear (MSTCLR-) signal is received from the MLC (power up or software initialize). The counter then increments once for each clock cycle (SYSCLK-) pulse from the MLC. This process continues until a firmware Branch command (UPINS7-) is encountered during the firmware execution. At this time the address counter is parallel loaded with 10 bits (UPCM10+ to 19+) of the firmware branch command.

The UPCM address counter outputs (UPAC00+ to 09+) select one of the 1024 locations of the UPCM storage and the 28 outputs (UPCM00+ to 27+) of the UPCM become valid. The outputs of the UPCM are always enabled as a result of the ground (ZGND) on the enable input of the UPCM chips. The exception to this is outputs 4 to 7 (UPCM04+ to 07+) of the UPCM, which are inhibited (UPCMI4+) for a firmware Read Transmit Buffer command (UPINS5-) or a firmware Read/Write External RAM (RAM enable UPCM25+ low).

UPCM outputs 0 to 2 (UPCM00+ to 02+) are the select bits on the UPCM command decoder which determine which of the eight firmware instruction types (UPINS0- to 7-) are to be performed (refer to Section III of this manual for instruction definitions). The UPCM output bits 3 and 8 to 23 (UPCM03+ and UPCM08+ to 23+) are used by the microprocessor as described in subsection 2.5.1, or if a firmware branch is being performed, bits 10 through 19 (UPCM10+ to 19+) are the branch address. Bits 4 through 7 of the UPCM output, after "ORing" (UPIDB0+ to 3+), generate subcommands (refer to Table 2-4) which are strobes that set or reset various hardware elements throughout the HDLC during firmware Subcommand (UPINS0-) or Branch and Subcommand (UPINS7-) instructions. These bits (UPIDB0+ to 3+) are also the constant input to the microprocessor (see Figure 2-10). UPCM output bit 24 (UPCM24+) enables the receive data (see Figure 2-14) with bit 24 (UPCM25+) supplying the enable for the external RAM. The final two outputs (UPCM26+ and 27+) are the output enables for the two halves of the transmit buffer.

The last element of the UPCM logic is the test multiplexer, which selects one of the eight available test conditions (refer to Table 2-5) by using the UPCM outputs 4 to 6 (UPIDB0+ to 2+) during a firmware test and modify instruction (UPINS3-). When the condition selected is true, the test multiplexer (SMODUI+) sets the test flip-flop (MODUIN+) at system clock (SYSCLK-) time from the MLC. Once the test flip-flop sets (MODUIP+), it inhibits the UPCM command decoder (enable high), effectively bypassing the subsequent firmware instruction.

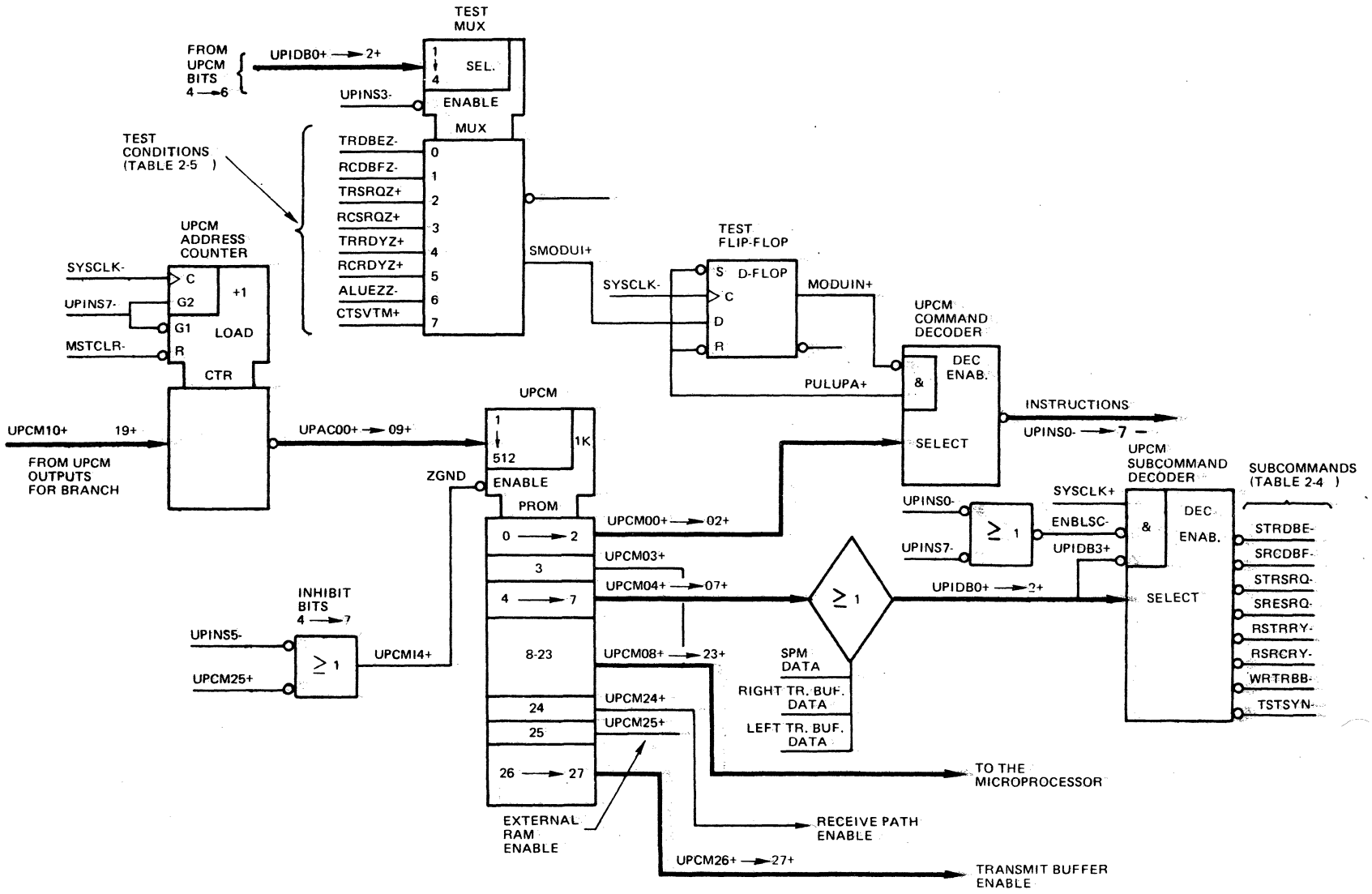


Figure 2-12 UPCM/UPCM Command Decode/Test Multiplexer

HONEYWELL PROPRIETARY AND CONFIDENTIAL

2.5.4 Transmit Logic (see Figure 2-13)

This subsection describes the HDLC adapter hardware implemented when a transmit operation is performed on the attached communication line. The data transfer path, as well as the control logic and timing required to execute a transmit operation, are discussed. The discussion of firmware is minimal, as its control is explained in subsection 2.4.3. The transmit hardware is reset by a Master Clear (MSTCLR-) signal or by firmware clearing all registers to zero when the system is powered up, a software initialize function code is received by the MLC for this channel, or when the CLEAR pushbutton is depressed on the control panel. The central processor software now can initiate a transmit operation.

When the transmit operation is received by the MLC, the MLC initiates processing of the transmit CCP for the appropriate channel. The CCP using WRLCBS+ from the MLC command decode logic (see Figure 2-11) loads the line control buffer from MLC data lines CPDT00+ to 05+ in order to set data terminal ready (DCEDTR-) and request to send (DCERST-), and to set, as appropriate, new sync (DCENSZ-) and speed select (DCESSZ-). These signals are sent to the DCE through the DCE interface drivers to establish data set control. Test mode (TSTMOD-) is reset with the negation of the MLC data bit 5 (CPDT05-), which selects the lower inputs on the Test Mode Multiplexer, establishing the normal data path for a transmit. Also when WRLCBS+ from the MLC command decode logic (see Figure 2-11) is set by the CCP, Transmit On (TRONZZ+) is set with CPDT07+ from the MLC data bus. This allows transmit channel request interrupts (TRCRIZ-) to be generated when the HDLC firmware sets the Set Transmit Service Request (STRSRQ+) subcommand (see Figure 2-11) to request data from the MLC.

The CCP now loads the transmit control byte into the Transmit Buffer by generating WRTRCB- in the MLC command decode logic (see Figure 2-11). This allows the single byte on the MLC data lines 0 to 3 (CPDT00+ to 03+) to be written into the Transmit Buffer at STROBE- time. This sets up the byte size, interframe fill mode (flag or abort), and end of frame control information in the control byte for firmware testing.

If clear to send (DCECTS-) from the DCE (see Figure 2-14) is set, the HDLC firmware generates the set Transmit Service Request (STRSRQ+) subcommand (see Figure 2-12), which sends a channel request interrupt (TRCRIZ-) to the MLC for data. The transmit data buffer empty (TRDBEZ-) indicator (see Figure 2-11) is scanned by HDLC firmware, using the capabilities of the test multiplexer (refer to subsection 2.5.3 and Figure 2-12) to determine when data has been supplied by the MLC.

To send data to the HDLC, the CCP sends a Write Transmit Data Buffer (WRTRDB-) command (see Figure 2-11) to the HDLC. This causes the MLC data lines (CPDT00+ to 07+) to be loaded into the data word location of the transmit buffer at STROBE- time and resets the transmit data buffer empty (see Figure 2-11) indicator. When the firmware determines that the data is available (TRDBEZ-), it transfers the data word to the transmit shift register in the external RAM a byte at a time. This is accomplished by addressing the transmit buffer with microprogram control memory bits 8 and 9 (UPCM08+ and 09+) to select the proper location and enabling first the right byte (UPCM27+) and then the left byte (UPCM26+). The bytes (INBFR0+ to 3+ and INBFR4+ to 7+) are then written into the external RAM by a Write External RAM command (UPINS1-) at SYSCLK- time to the location specified by the firmware command (UPCM08+ to 11+). (This path is through the microprocessor, as explained in subsection 2.5.1.) The firmware then sets the data buffer empty indicator and generates a channel request interrupt to the MLC for the second word of data.

The first word of data is then shifted a byte at a time, using the shifted bit as the serial transmit data. For the first transmit bit, the data word is read from the external RAM (SPMEM0- to 3-) a byte at a time and stored in the internal RAM of the microprocessor (see Figure 2-10) with no shift. The output data of microprocessor bit 3 (UPODB3+), which reflects the LSB of the data word, is then latched into the Bit Buffer (TRBITB+) when a Write Transmit Bit Buffer (WRTRBB-) subcommand is generated by firmware. The data is then written back into the internal RAM. When the next transmit clock (TRCLKZ+) from the DCE is received, the first data bit in the Bit Buffer (TRBITB+) is latched into the Transmit Data flip-flop (TRDATA-) and passed through the Test Mode Multiplexer (DCETDZ+) to the DCE interface drivers (DCETDZ-). TRCLKZ+ also sets the Transmitter Ready (TRRDYZ+) indicator, which is used for firmware testing (refer to Figure 2-12) to indicate that the bit buffer (TRBITB+) is ready for the next data bit.

The internal RAM data word is again read and passed through the microprocessor components only this time with a single bit shift to the right. This puts the second data bit on the microprocessor output data lines bit three (UPODB3+) which is written into the Bit Buffer when a Write Transmit Bit Buffer (WRTRBB-) subcommand is performed. The data in the bit buffer is then transferred to the DCE on the subsequent transmit clock (TRCLKZ+). This process of shifting bits to the DCE and inputting data from the MLC continues until the CCP sets the transmit end of frame and interframe fill mode in the transmit control byte, as previously described, and sends the final FCS word to the HDLC.

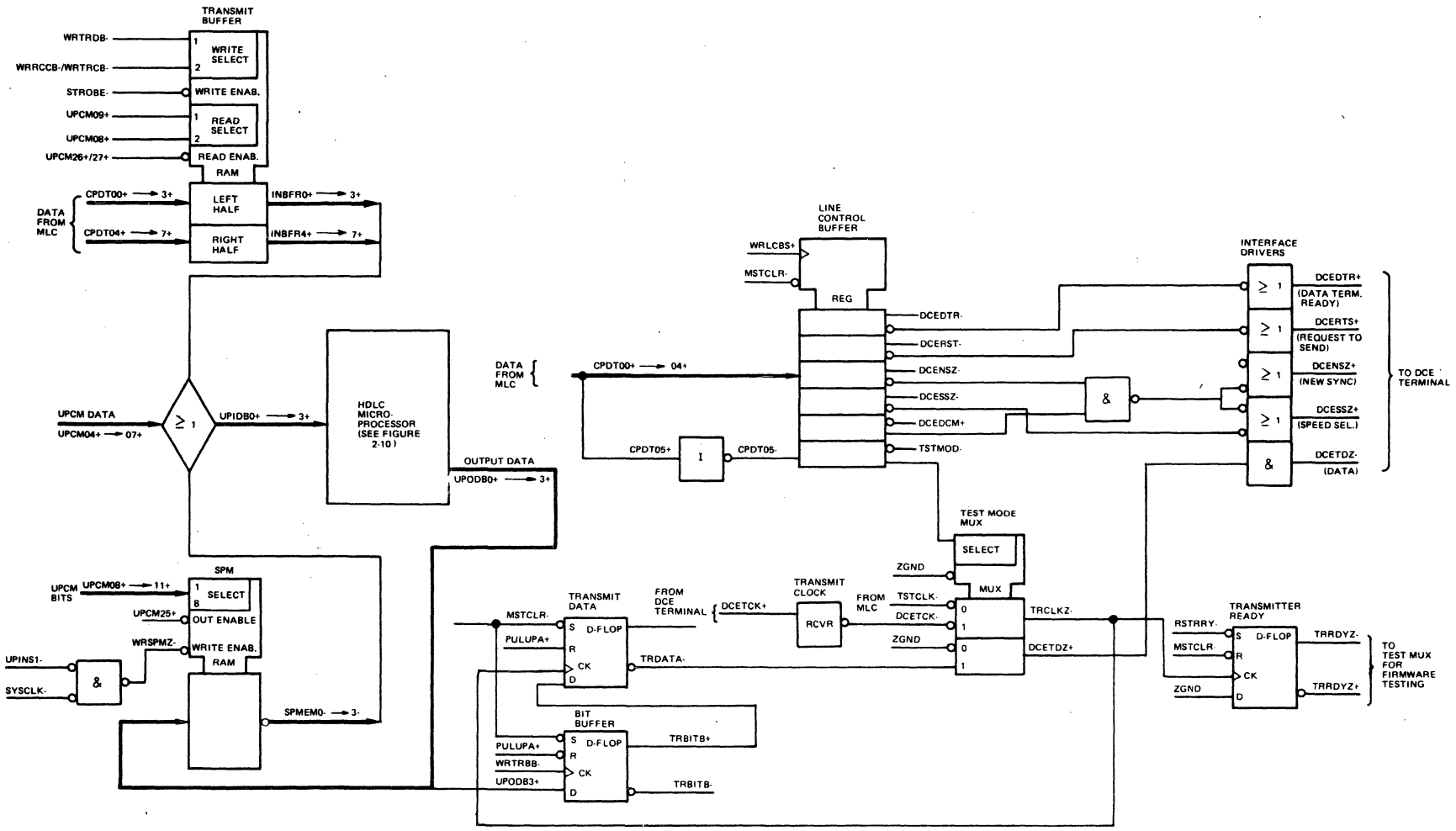


Figure 2-13 Transmit Data and Control Paths

HONEYWELL PROPRIETARY AND CONFIDENTIAL

2.5.5 Receive Logic (see Figure 2-14)

This subsection describes the HDLC adapter hardware implemented when a receive operation is performed on the attached communication line. The data transfer path, as well as the control logic and timing required to execute a receive operation, are discussed. The discussion of firmware is minimal as its function during the receive is explained in subsection 2.4.3.

The receive hardware is reset by a Master Clear (MSTCLR-) signal and by firmware clearing all registers to zero when the system is powered-up, a software initialize function is received by the MLC for this channel, or when the CLEAR pushbutton is depressed on the control panel. The central processor software now can set up the MLC for a receive operation.

The MLC initiates processing of a receive operation by execution of the CCP for the appropriate channel. The CCP, using WRLCBS+ from the command decode logic (see Figure 2-11), loads the line control buffer (see Figure 2-13) from MLC data lines CPDT00+ to 05+ in order to set data terminal ready (DCEDTR-), reset request to send (DCERST-), and to set or reset appropriately new sync (DCENSZ-) and speed select (DCESSZ-). These signals are sent to the DCE through the DCE interface drivers (see Figure 2-13) to establish data set control. Test mode (TSTMOD-) is reset with the negation of the MLC data bit 5 (CPDT05+), which selects the lower inputs on the Test Mode Multiplexer (see Figure 2-13). These establish the normal data path for a receive operation. Also, when WRLCBS+ from the MLC command decode logic is set by the CCP, Receive On (RCONZZ-) is set with CPDT06+ from the MLC data lines. This allows receive channel request interrupts (RCCRIZ-) to be generated when the HDLC firmware issues a Set Receive Service Request (SRCSRQ-) subcommand (see Figure 2-11) to send a data word to the MLC.

The CCP now loads the receive control byte into the Transmit Buffer (see Figure 2-13) by generating WRRCCB- in the MLC command decode logic (see Figure 2-11). This allows the single byte on MLC data lines 4 to 7 (CPDT04+ to 07+) to be written into the Transmit Buffer at STROBE- time. The purpose of loading this byte is to set up the byte size for firmware testing (see Figure 2-5).

When the receive clock (RCCLKZ+) is detected from the DCE (DCERCK+) through the Test Mode Multiplexer, it is inverted (RCCLKZ-). RCCLKZ- causes the data bit from the DCE (DCERDZ+), through the Test Mode Multiplexer (RCDATX+), to be latched into the Receive Data Latch (RCDATY-) and also sets Receive Data Ready (RCRDYZ-) for firmware testing. When the firmware enables the Data Driver (RCDATZ+), using bit 24 (UPCM24+) of the firmware command, the data bit is shifted into the microprocessor (refer to subsection 2.5.1 and Figure 2-10). This shifting into the microprocessor continues until three words of data are accumulated in the microprocessor storage element. The firmware then transfers one word a byte at a time from the microprocessor into the Receive Data Buffer. This is accomplished by enabling the writing of the

buffer (WROBZZ-) with a Write Receive Buffer (UPINS2-) firmware command at SYSCLK- time and selecting the proper location with command bits 8 and 10 and 11 (UPCM08+, UPCM10+ and 11+). This addressing scheme causes the data (UPODB0+ to 3+) to be written into two byte locations simultaneously, but the extra location is not utilized, and no harm is done (refer to Figure 2-6).

The firmware now generates a channel request interrupt (RCCRIZ-) with the Set Receive Service Request (SRCSRQ-) subcommand (see Figure 2-12). When the CCP services the request, it reads the status from the HDLC by sending a Read Receive Status (RDRCST-) to the adapter (see Figure 2-11). When the CCP reads the status, it enables the right byte side of the receive buffer (RDOBRH-) and selects the firmware-generated status location with RDSTAT- (see Figure 2-11). RDSTAT- also enables the four outputs of the status drivers. The right byte of the Receive Buffer and four outputs of the status drivers are sent to the MLC as a complete status word on the input data lines (LADAT0+ to 7+).

The CCP then reads the data word from the Receive Buffer by sending a Read Receive Data Buffer (RDRCDB-), which enables the buffer and selects the data word location. The output of this full word location is then transferred to the MLC via the input data lines (LADAT0+ to 7+).

This receive and transfer process continues until a flag is recognized by the HDLC firmware. The firmware then transfers the three words of information (data and FCS) to the MLC with closing flag set in the status word which accompanies the final transfer for the FCS word.

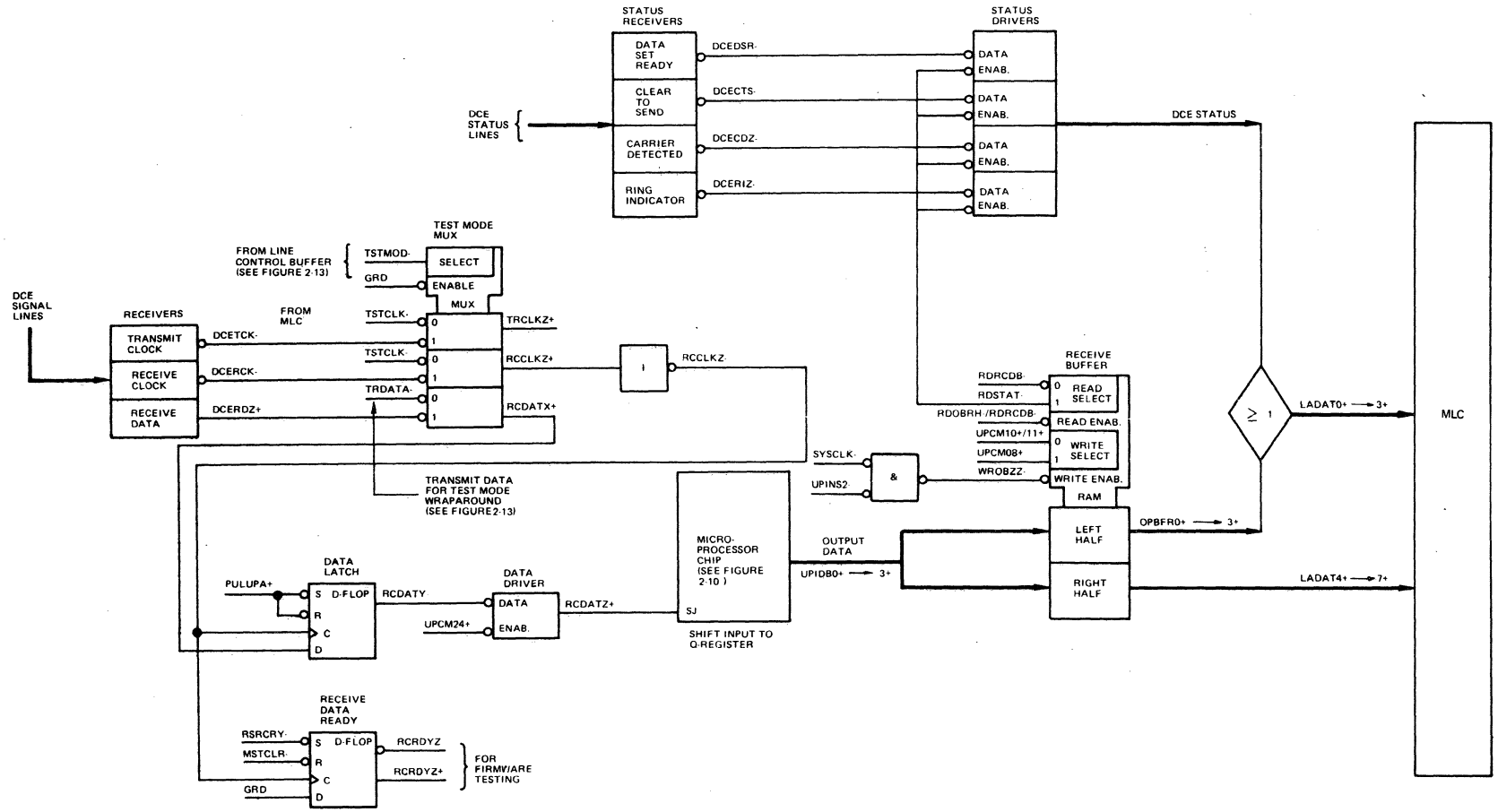


Figure 2-14 Receive Data, Status, and Control Paths

HONEYWELL PROPRIETARY AND CONFIDENTIAL

III
THEORY OF OPERATION - CYCLE FLOW

3.1 ADAPTER FIRMWARE

The HDLC firmware is comprised of routines which are formed from sequences of various types of firmware commands (see subsection 3.4) resident within the HDLC read-only microprogram control memory. This control memory contains up to 1024 locations, with each location containing a 28-bit firmware word called a firmware command. A decoded firmware command results in the specific action of various hardware elements.

HDLC firmware includes two basic types of routines: those which handle the transfer of information to the attached communications device (transmit routines) and those which handle reception of data from the attached communications device (receive routines). Firmware described herein is based on file revision level 1.0.

3.2 INTERNAL RAM

HDLC firmware has access to a 16-word by 4-bit register file called the internal RAM (i.e., the RAM is internal to the microprocessor). The internal RAM contains eight 4-bit receive shift registers for reception of data from the communications device, two work registers, plus registers for storage of receive flags and counter and firmware routine state control information. Internal RAM locations are identified in Figure 3-1 and described in Table 3-1.

REGISTER HEX	BIT POSITION 0	1	2	3		
F	RFR00	RFR.LH			FLAG REGISTER	RECEIVE SHIFT REGISTERS
E	RFR.RH			RFR08		
D	RFCSR.LQ				FCS REGISTER	
C	RFCSR.LCQ					
B	RFRSR.RCQ					
A	RFCSR.RQ					
9	RBR.LH				BYTE REGISTER	
8	RBR.RH					
7	RCDBFFLG	RILS	ABRTRCVD	FLGRCVD		
6	R5C					
5	RBSC					
4	0	RSRCSTATE				
3	0	RFCSTATE				
2	ROSD	(RFU)	RDHBT	RILSFLG		
1	F1				WORK REGISTERS	
0	F0					

RH = Right Hand 4 bits
LH = Left Hand 4 bits
LQ = Left Hand Quarter
LCQ = Left Hand Center Quarter
RCQ = Right Hand Center Quarter
RQ = Right Hand Quarter

Figure 3-1 Internal RAM Register Assignment

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table 3-1 Internal RAM Register File Assignments (Sheet 1 of 2)

Register Location (Hex)	Bit(s)	Mnemonic	Description
0	0 - 3	F0	Work Register 0
1	0 - 3	F1	Work Register 1
2	0	ROSD	Receive Overrun Status Delayed. Stores closing flag condition when second frame ends before status transfer for first frame is completed.
	1	RFU	Reserved for future use.
	2	RDHBT	Receive Data has Been Transferred. Indicates that data (for frame currently being received) has been transferred to MLC.
	3	RILSFLG	Receive Idle Link State Flag. Indicates receipt of fifteen contiguous One bits.
3	0	-	Must Be Zero
	1 - 3	RFCSTATE	Receive Flush Control State. Indicates the state of the receive flush control routine as follows: 000 - OFF state 001 - RESET STATUS state 010 - SECOND FCS state 011 - FIRST FCS state 100 - LAST DATA state 101 - ABRT&OVRN state 110 - ABRT state 111 - FLG&OVRN state
4	0	-	Must be Zero
	1 - 3	RSRCSTATE	Receive Shift Register Control State. Indicates state of receive shift register control routine as follows: 000 - TEXT state 001 - WAIT-2 state 010 - WAIT-1 state 011 - OVRN state 100 - ABRT state 101 - INSYNC state 110 - OUTASYNS state 111 - FIRST state

Table 3-1 Internal RAM Register File Assignments (Sheet 2 of 2)

Register Location (Hex)	Bit(s)	Mnemonic	Description
5	0 - 3	RBSC	Receive Byte Size Counter. Keeps track of shifting in receive shift register.
6	0 - 3	R5C	Receive 5 Counter. (Zero Deletion Counter) Counts contiguous One bits for zero deletion control (actually counts from B(hex) to 0(hex)).
7	0	RCDBFFLG	Receive Data Buffer Full Flag. Indicates that the receive data buffer is full.
	1	RILS	Receive Idle Link State. Indicates that the idle link state status has been reported to the MLC.
	2	ABRTRCVD	Abort Received. Indicates presence of an abort sequence in receive flag register.
	3	FLGRCVD	Flag Received. Indicates presence of a flag sequence in receive register.
8	0 - 3	RBR(RH)	Receive Byte Register (Right half). Part of receive shift register. Characters of less than 8 bits are right-justified here.
9	0 - 3	RBR(LH)	Receive Byte Register (Left Half). Part of receive shift register.
A	0 - 3	RFCSR(RQ)	Receive Frame Check Sequence Register. Four, 4-bit bytes that comprise the frame check sequence register.
B	0 - 3	RFCSR(RCQ)	
C	0 - 3	RFCSR(LCQ)	
D	0 - 3	RFCSR(LQ)	
E	0 - 3	RFR(RH)	Receive Flag Register (Right Half). Data stream is examined here for flag and abort sequences.
F	0 - 3	RFR(LH)	Receive Flag Register (Left Half). See above.

3.3 EXTERNAL RAM

HDLC firmware also has access to an external RAM. The external RAM is a 16-word by 4-bit read/write memory that is an extension of the internal RAM. The external RAM contains the status register, receive history registers, transmit shift register, and also contains storage for transmit flags. External RAM locations can be written into from the microprocessor via the Write External RAM (WER) firmware command, and can transfer information to the microprocessor via the Read External RAM (RER) firmware command. External RAM register locations are identified in Figure 3-2 and described in Table 3-2.

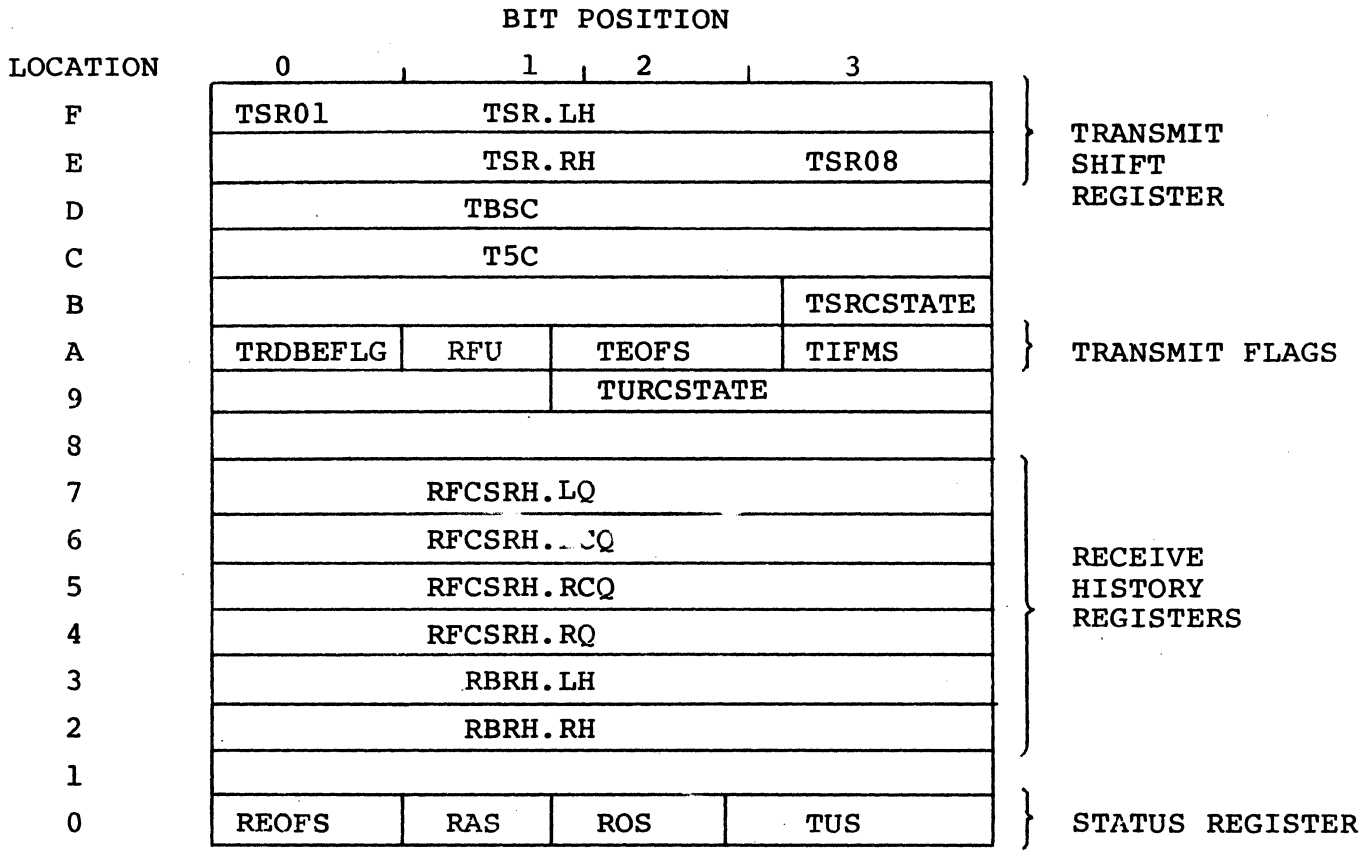


Figure 3-2 External RAM Register Assignment

Table 3-2 External RAM Register Assignments (Sheet 1 of 2)

Register Location (Hex)	Bits	Mnemonic	Description
0	0	REOFS	Receive End of Frame Status. Indicates that a frame was ended by a flag sequence.
	1	RAS	Receive Closing Abort Status. Indicates that a frame was ended by an abort sequence or an idle state link.
	2	ROS	Receive Overrun Status. Indicates that data was lost during frame.
	3	TUS	Transmit Underrun Status. Indicates that underrun has occurred.
1	0 - 3	-	Not Used
2	0 - 3	RBRH(RH)	Receive Byte Register History (Right Half). Part of receive history register. Provides temporary storage for last data byte at end of frame until transferred by MLC.
3	0 - 3	RBRH(LH)	Receive Byte Register History (Left Half). See above.
4	0 - 3	RFCSRH(RQ)	Receive Frame Check Sequence Register History. Sixteen bits that provide temporary storage of the frame check sequence (FCS).
5	0 - 3	RFCSRH(RCQ)	
6	0 - 3	RFCSRH(LCQ)	
7	0 - 3	RFCSRH(LQ)	
8	0 - 3	-	Not Used
9	0 - 1	-	Not Used
	2 - 3	TURCSTATE	Transmit Underrun Report Control State. Indicates the state of the Transmit Underrun Report Control routine as follows: 00 - OFF State 01 - RESET UNDERRUN State 10 - REPORT UNDERRUN State 11 - RFU
A	0	TRDBEFLG	Transmit Data Buffer Empty Flag. Indicates that the transmit data buffer is empty.

Table 3-2 External RAM Register Assignments (Sheet 2 of 2)

Register Location (Hex)	Bits	Mnemonic	Description
A (Cont.)	1	TUSTR	Transmit Underrun Status Transfer Request. Indicates that an underrun status transfer request has been made.
	2	TEOFS	Transmit End of Frame Synchronized. An internal copy of TEOM.
	3	TIFMS	Transmit Interframe Fill Mode Synchronized. An internal copy of TIFM, which specifies whether to send Flag or Abort sequences between frames: 0 = Abort; 1 = Flag
B	0 - 2	-	Must be Zero.
	3	TSRCSTATE	Transmit Shift Register Control State. Indicates state of transmit shift register control routine as follows: 0 = TEXT state; 1 = IDLE state
C	0 - 3	T5C	Transmit 4 (Zero Insertion) Counter. Counts contiguous One bits for zero insertion control. Actually counts from B(hex) to 0(hex).
D	0 - 3	TBSC	Transmit Byte Size Counter. Keeps track of shifting in transmit shift register.
E	0 - 3	TSR(RH)	Transmit Shift Register (Right Half). Register in which transmit byte shifting is done. Rightmost bit is TSR08, which is loaded into the transmit bit buffer.
F	0 - 3	TSR(LH)	Transmit Shift Register (Left Half) See above.

3.4 FIRMWARE COMMANDS AND MICROINSTRUCTIONS

A microinstruction is defined as any bit of a 28-bit word in the microprogram control memory. A micro-operation (micro-op) is defined as a combination of two or more microinstructions. A microinstruction bit can be decoded by itself or in conjunction with other microinstruction bits (i.e., as a micro-op) to cause a specified hardware action. Taken together, the 28 microinstruction bits comprise a firmware command. A combination of a number of firmware commands utilized to perform a particular function is known as a microprogram, or firmware routine.

HDLC firmware has nine types of firmware commands, each of which is subdivided into various fields (micro-ops) to perform specific operations, such as internal or external RAM addressing, transmit or receive buffer addressing, ALU function control, etc. Each firmware command is identified by an op-code in bits 0, 1 and 2 of the firmware command. Table 3-3 contains the formats of the HDLC firmware commands; Tables 3-4 through 3-12 provide definitions of the firmware command fields.

3.4.1 Subcommand Command (SC)

Subcommand commands are used to set and reset hardware control registers, to load the transmit bit buffer, and to generate a test sync pulse. Specific subcommands are a decode of the subcommand field as shown in Table 3-4.

3.4.2 Write External RAM Command (WER)

This command causes the output of the microprocessor to be written into the external RAM location specified by the RAM address (RA) field. Also, the logic or arithmetic function specified by the FC field is performed using the operands specified by the SOC field.

3.4.3 Write Output Buffer Command (WOB)

This command causes the output of the microprocessor to be written in the output (receive) buffer locations specified by the Output Buffer (OB) field. This command is used to transfer information (data) received from the communications equipment, receive status, or transmit status to the output (receive) buffer via the internal RAM.

3.4.4 Test and Modify Next Command (TAMN)

This command enables the microprogram to examine miscellaneous hardware status conditions within the adapter, with the result being saved for the next command (normally a Branch command). Microprogram control memory bits 4, 5, and 6 determine the function to be tested as shown in Table 3-5. If the test result is true and the next command is a Branch command, the branch will not be made.

Table 3-3 Microinstruction Formats

MNEMONIC	NAME	MICROPROGRAM CONTROL MEMORY BIT UPCMnn																							
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
SC	Subcommand	0	0	0	C	SUB				0		FAA		FBA				FC			SOC			DC			1	1	1
WER	Write External RAM	0	0	1	C	X	X	X	X	RA				FBA				FC			SOC			DC			0	1	1
WOB	Write Output Buffer	0	1	0	C	K				OB				FBA				FC			SOC			DC			1	1	1
TAMN	Test and Modify Next	0	1	1	C	TST				FAA				FBA				FC			SOC			DC			1	1	1
MMP	Modify Microprocessor	1	0	0	C	K				FAA				FBA				FC			SOC			DC			1	1	1
RIB	Read Input Buffer	1	0	1	C	X	X	X	X	IB		0	0	FBA				FC			SOC			DC			1	IB	
RER	Read External RAM	1	1	0	C	X	X	X	X	RA				FBA				FC			SOC			DC			0	1	1
BR	Branch	1	1	1	0	0	0	0	1	0	0	Branch Address									0	0	0	0	1		1	1	1
BRSC	Branch and Subcommand	1	1	1	0	SUB				0	0	Branch Address									0	0	0	0	1		1	1	1

0 = Low; 1 = High; X = Don't Care

C = Carry In
K = Constant/Mask
SUB = Subcommand Select
FAA = File Read Address
FBA = File Write Address
RA = (Scratch Pad) RAM Address
IB = Input Buffer
OB = Output Buffer
FC = ALU Function Control
SOC = ALU Source Operand Control
DC = ALU Destination Control

HONEYWELL PROPRIETARY AND CONFIDENTIAL

Table 3-4 Subcommand Selection (SUB Field)

UPCMnn+00 04	05	06	STROBE	FUNCTION
0	0	0	STRDBE-00	Sets Transmit Data Buffer Empty F/F
0	0	1	SRCDBF-00	Sets Receive Data Buffer Full F/F
0	1	0	STRSRQ-00	Sets Transmit Service Request F/F
0	1	1	SRCSRQ-00	Sets Receive Service Request F/F
1	0	0	RSTRRY-00	Resets Transmit Ready F/F
1	0	1	RSRCRY-00	Resets Receive Ready F/F
1	1	0	WRTRBB-00	Writes Transmit Bit Buffer F/F
1	1	1	TSTSYN-00	Generates Test Sync Pulse

Table 3-5 Test and Modify Next Selection (TST Field)

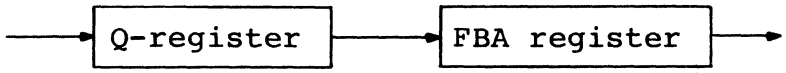
UPCMnn+00 04	05	06	FUNCTION TESTED	MODIFY NEXT INSTRUCTION IF:	EXECUTE NEXT INSTRUCTION IF:
0	0	0	TRDBEZ-00	Transmit Data Buffer Full	Transmit Data Buffer Empty
0	0	1	RCDBFZ-00	Receive Data Buffer Empty	Receive Data Buffer Full
0	1	0	TRSRQZ+00	Transmit Service Request Pending	No Transmit Service Request Pending
0	1	1	RCSRQZ+00	Receive Service Request Pending	No Receive Service Request Pending
1	0	0	TRRDYZ+00	Time to Generate New Transmit Data Bit	Not Time to Generate New Transmit Data Bit
1	0	1	RCRDYZ+00	Time to Process New Receive Data Bit	Not Time to Process New Receive Data Bit
1	1	0	ALUEZZ-00	ALU Output Does Not Equal Zero	ALU Output Equals Zero
1	1	1	CTSVTM+00	Clear to Send or Test Mode True	Clear to Send and Test Mode Both False

3.4.5 Modify Microprocessor Command (MMP)

This command causes the arithmetic or logic function specified by the Function Control (FC) field to be performed using the operands specified by the Source Operand Control (SOC) field, with the result shifted (if applicable) and stored in the Q-register and/or file register as determined by the DC field. A detailed list of arithmetic and logic operations is provided in the firmware file.

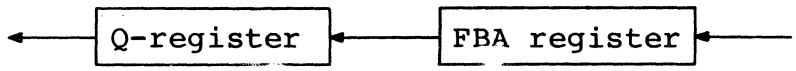
Arithmetic and logic functions are performed by the microprocessor Arithmetic Logic Unit (ALU) on registers in the internal RAM, and the constant field (K). The K-field is frequently used as a mask for the testing, setting or resetting of individual bits in the internal RAM.

The resultant ALU output and the Q-register can be shifted left or right, if desired, under control of the Destination Control (DC) field. When a right shift is performed, the Q-register and the FBA register are aligned as follows:



The right-most bit (low order bit) of the Q-register is shifted into the left-most bit position of the internal RAM register specified by the FBA address, and the received data bit is shifted into the left-most bit of the Q-register. The right-most bit of the FBA register is lost.

When a left shift is performed, the Q-register and the FBA register are aligned as follows:



The left shift is performed in a manner similar to the right-shift operation, with the left-most bit of the Q-register being lost and with an unknown value being shifted into the right-most position of the FBA register.

3.4.6 Read Input Buffer Command (RIB)

This command causes the content of the input (transmit) buffer location specified by the Input Buffer address (IB) field to be applied to the microprocessor input data bus. The input (transmit) buffer is a register file used to store the transmit data byte and transmit and receive control words, and is written directly by the MLC. As shown in Table 3-11, the IB field consists of bits 8, 9, 26, and 27 of the microprogram control memory. These four bits address the input (transmit) buffer.

3.4.7 Read External RAM Command (RER)

This command causes the content of the external RAM location specified by the RAM Address (RA) field to be applied to the microprocessor input data bus. The logic or arithmetic function specified by the FC field is performed using the operands specified by the SOC field, with the result shifted, if applicable, and stored in the Q-register and/or file register as determined by the DC field.

3.4.8 Branch Command (BR)

This command causes an unconditional branch to the microprogram control memory location specified by the branch address field. If a branch command is preceded by a Test and Modify Next command whose test result is true, the branch command is skipped.

3.4.9 Branch and Subcommand Command (BRSC)

This command causes the hardware function specified by the Subcommand field to be executed, followed by an unconditional branch to the microprocessor control memory location specified by the branch address field. This command is skipped if preceded by a Test and Modify Next command whose test result is true.

Table 3-6 Scratch Pad Register Selection (RA Field)

UPCMnn+00				SCRATCH PAD REGISTER
08	09	10	11	
0	0	0	0	R0
0	0	0	1	R1
0	0	1	0	R2
0	0	1	1	R3
0	1	0	0	R4
0	1	0	1	R5
0	1	1	0	R6
0	1	1	1	R7
1	0	0	0	R8
1	0	0	1	R9
1	0	1	0	RA
1	0	1	1	RB
1	1	0	0	RC
1	1	0	1	RD
1	1	1	0	RE
1	1	1	1	RF

Table 3-7 Microprocessor Register File Selection (FAA and FBA Fields)

UPCMnn+00				MICROPROCESSOR REGISTER
08	09	10	11	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	A
1	0	1	1	B
1	1	0	0	C
1	1	0	1	D
1	1	1	0	E
1	1	1	1	F

Table 3-8 ALU Source Operand Control (SOC Field)

UPCMnn+00			OCTAL CODE	ALU SOURCE OPERANDS	
19	20	21		R	S
0	0	0	0	A	Q
0	0	1	1	A	B
0	1	0	2	0	Q
0	1	1	3	0	B
1	0	0	4	0	A
1	0	1	5	D	A
1	1	0	6	D	Q
1	1	1	7	D	0

Table 3-9 ALU Function Control (FC Field)

UPCMnn+00			OCTAL CODE	ALU FUNCTION
16	17	18		
0	0	0	0	R plus S
0	0	1	1	S minus R
0	1	0	2	R minus S
0	1	1	3	R or S
1	0	0	4	R and S
1	0	1	5	$\overline{R}$ and S
1	1	0	6	R Xor S
1	1	1	7	R Xnor S

Table 3-10 Receive Buffer Register Selection (OB Field)

UPCMnn+00				HEX	OUTPUT (RECEIVE) BUFFER REGISTER NAME
08	09	10	11		
0	0	0	1	1	Write Output Status
1	0	0	1	9	Write Receive Data (Left Half)
1	0	1	0	A	Write Receive Data (Right Half)

Table 3-11 Transmit Buffer Register Selection (IB Field)

UPCMnn+00				HEX	INPUT (TRANSMIT) BUFFER REGISTER NAME
08	09	26	27		
0	1	0	1	5	Read Transmit Control Word
0	1	1	0	6	Read Receive Control Word
1	0	0	1	9	Read Transmit Data (Left Half)
1	0	1	0	A	Read Transmit Data (Right Half)

Read Enable for Input Buffer

0 = Read Enabled
1 = Read Not Enabled

Input Buffer Address

Table 3-12 ALU Destination Control (DC Field)

UPCMnn+00		Octal Code	RAM Register Function		Q-Register Function			Y Output
$22(I_8)$	$23(I_7)$		Shift	Load	Shifter Input	Shift	Load	
0	0	0	-	No	ALU	None	Yes	ALU
0	1	1	None	Yes	-	-	No	A
1	0	2	Right	Yes	Q	Right	Yes	ALU
1	1	3	Left	Yes	Q	Left	Yes	ALU

Table 3-13 Generation of Fill Characters in TRSRC IDLE State Subroutine

TURCSTATE	TRSRQ	TRDBEFLG	TRDBE	TIFMS	Comments
Not OFF	-	-	-	-	Send Flag or Abort
-	Set	-	-	-	Send Flag or Abort
-	-	Set	-	-	Send Flag or Abort
OFF	Reset	Reset	Set	-	Set TRDBEFLG to cause TRDRC routine to generate data service request
					Send Flag or Abort
OFF	Reset	Reset	Reset	Set	Ready to start next frame.
					Flag was just sent, so send first data byte.
OFF	Reset	Reset	Reset	Reset	Ready to start next frame.
					Abort was just sent, so send starting flag

3.5 FIRMWARE CYCLE FLOW

As shown in Figure 3-3, HDLC firmware consists of an initialize routine (INITZ), three transmit routines (TRSCR, TRDRC, and TRURC), and three receive routines (RCSRC, RCDRC, and RCFCO). A master clear from the MLC causes initialization of the HDLC. Once initialized, HDLC firmware sequentially executes transmit and receive routines in a continuous loop as shown in Figure 3-3. The only interaction with MLC firmware is the issuance of receive data service requests (to send data or report status to the MLC) and transmit data service requests (to request more data from the MLC and to report status). Once a service request is set, firmware execution continues with the next routine. The only common point between the transmit and receive routine is the sharing of a common status register in the external RAM.

The three transmit routines (and associated subroutines) handle the transfer of data received from the MLC from the transmit buffer to the attached communications device. This data is first transferred from the transmit buffer into the Transmit Shift Register

(TSR) in the external RAM, together with firmware-generated flag or abort sequences. The content of the TSR is then serialized for transfer to the communications device. The major functions performed by the three transmit routines are:

1. Generate flag and abort sequences for interframe time fill
2. Serialize data for transfer to the communications device
3. Insert zeros where required for frame transparency
4. Detect and report underrun conditions
5. Recognize and report byte boundaries between frames.

The three receive routines and associated subroutines handle the transfer of data from the attached communications device to the output (receive) buffer, from which it is transferred to the MLC under MLC firmware control. The major functions performed by these three routines are:

1. Assemble serial receive data in the receive shift registers in the internal RAM
2. Monitor the assembled data for the presence of flag, abort, or idle link state sequences
3. Differentiate between opening flags, closing flags, and interframe flags
4. Delete flag, abort, and idle link state sequences
5. Delete inserted zeros
6. Delete invalid frames of fewer than 25 bits without interrupting the MLC
7. Transfer data bytes from the receive shift registers to the receive buffer
8. Detect and report overrun conditions and inhibit data transfer following overrun
9. Generate service requests for data transfer, status transfer, end-of-frame, and idle link states
10. Synchronize service requests
11. Assemble frame check sequences (FCS) as two 8-bit bytes
12. Accommodates 5-, 6-, 7-, or 8-bit bytes (right justifies 5-, 6-, and 7-bit bytes).

For further details of firmware operations, refer to Figures 3-6 through 3-21, which are intermediate flow charts of each routine and subroutine. Refer to the firmware listing if the precise microinstruction used to execute an operation must be known. For a complete description of symbology and nomenclature used in the flow chart, see the Multiline Communications Processor or Controller Manual listed in Section I.

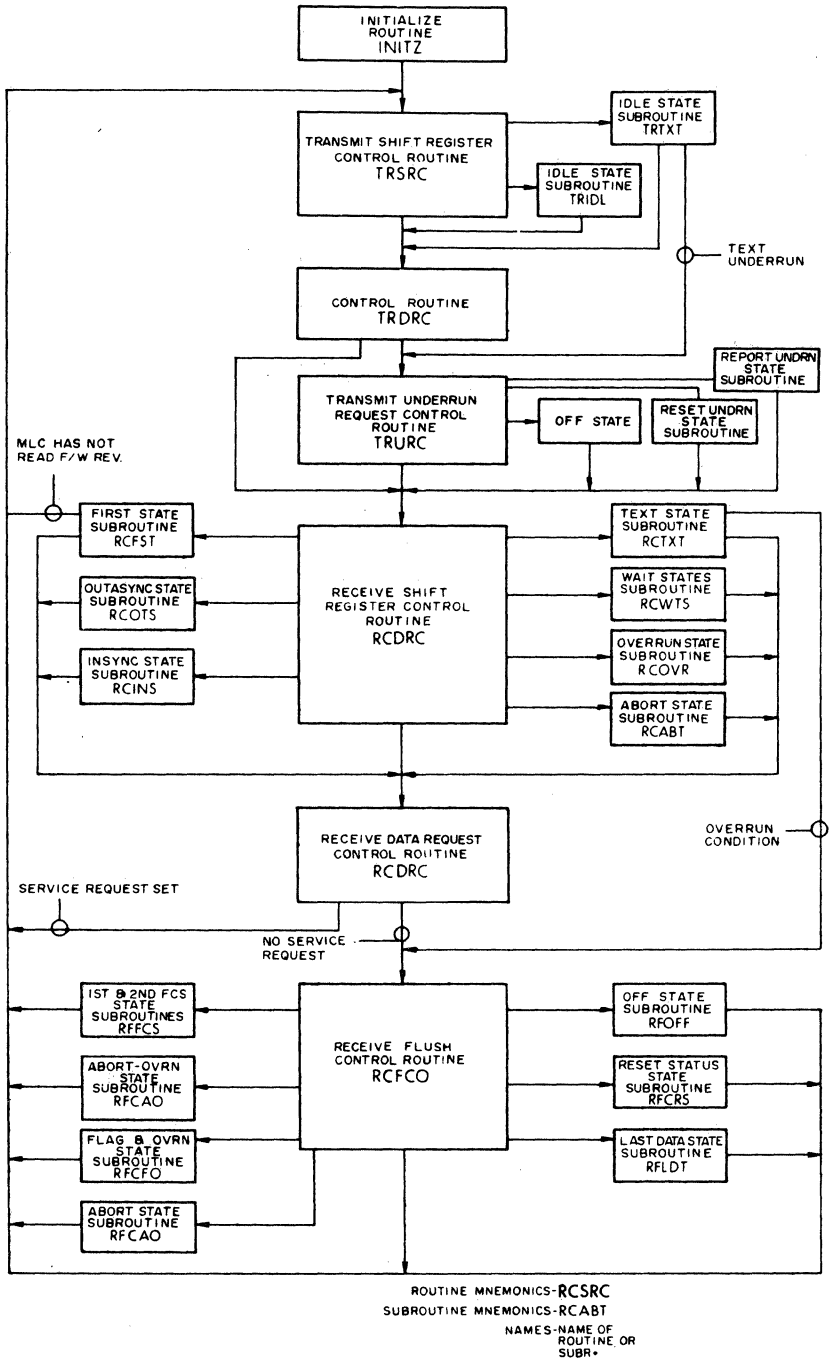
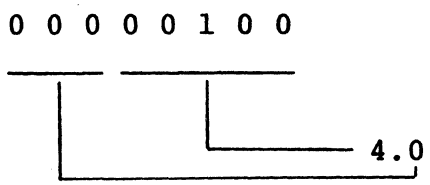


Figure 3-3 HDLC Firmware Overview

3.5.1 Initialize Routine

The Initialize (INITZ) routine sets HDLC hardware elements to the conditions necessary to begin firmware execution. This routine loads the firmware revision level into the left and right-half bytes of the output (receive) data buffer as shown in Figure 3-4. The right-most five bits (3-7) constitute the whole-number portion of

the firmware revision level, while the left-most three bits (0-2) constitute the decimal portion. Thus, a typical firmware revision level is expressed as:



Following initialization, the MLC must read this firmware revision level information before firmware can enter the receive routines.

As shown in Figure 3-6, the Initialize routine:

1. Initializes all counters
2. Resets all flags
3. Resets status in both the status register and output status buffer
4. Sets all firmware routines to their initial state
5. Loads an initial abort sequence in the transmit shift register.

Once initialization is complete, the HDLC firmware loops continuously through the six transmit and receive routines as shown in Figure 3-3.

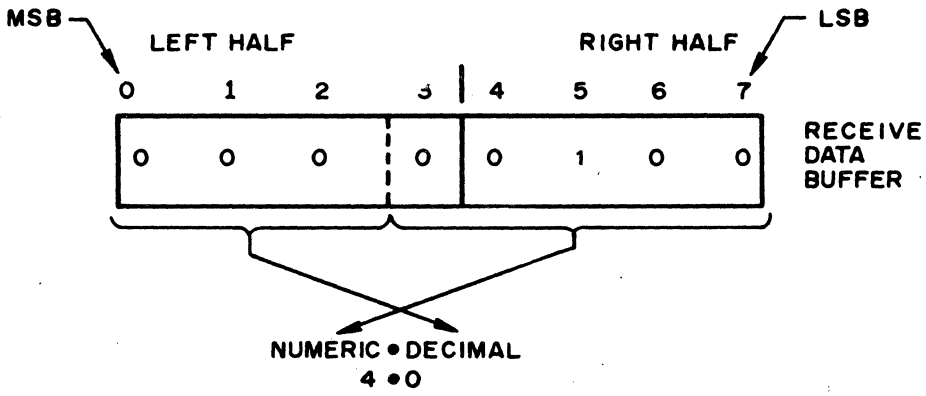


Figure 3-4 Firmware Revision Level in Output (Receive) Buffer

3.5.2 Transmit Shift Register Control Routine

The Transmit Shift Register Control (TRSRC) routine consists of an enter sequence plus two subroutines (called state subroutines) for the transfer of information from the input (transmit) buffer to the attached communications equipment. TRSRC is the first routine entered following initialization, in order to transmit the initial abort sequence that was placed in the transmit shift register during the initialize routine. The TRSRC routine can also be entered from the RCSRC routine. This entry forces firmware to loop only through the transmit routines immediately following initialization until the MLC has read the firmware revision information placed in the output (receive) buffer during the initialize routine. This loop prevents data from being read from the attached communications equipment until after the MLC has read this information.

The TRSRC routine is also entered from the RCDRC routine when a service request is set, and from the RCFCO routine when no service request is set, or when an overrun condition is being reported.

3.5.2.1 TRSRC Enter Sequence (Figure 3-7)

At each pass of the firmware, the Transmit Ready flip-flop is tested to see if the transmit bit buffer is ready to accept the next bit from the transmit shift register for transmission to the attached communications equipment. If the transmit buffer is not ready, an exit is made to the TRDRC routine. If it is ready, a branch is made to the TRSRC state subroutine indicated by the setting of the TSRCSTATE bit in the external RAM.

The TRSRC routine is always in the IDLE state immediately following initialization and in either the IDLE or the TEXT state at other times.

3.5.2.2 TRSRC IDLE State Subroutine (Figure 3-8)

The TRSRC IDLE state handles the transfer of flag and abort sequences to the attached communications equipment. Zero insertion, used in the TRSRC TEXT state subroutine to prevent erroneous flag or abort sequences, is not performed during the IDLE state subroutine.

During the initialize routine the IDLE state is set and an abort sequence is loaded into the transmit shift register. IDLE state is therefore the first subroutine entered following initialization, and is used to send the abort sequence to the attached communications equipment. During other firmware passes, the IDLE state is entered from the TRSRC TEXT state subroutine under the following conditions:

1. The last byte of a frame has been transmitted, a closing flag has been placed in the transmit shift register, and the transmit byte size counter has been set to 8;
2. A text underrun condition has been detected, an abort sequence has been placed in the transmit shift register, and the transmit byte size counter has been set to 8.

During each of eight firmware passes, one bit of the flag or abort sequence is sent to the attached communications equipment via the transmit bit buffer and the transmit byte size counter is decremented. After the eighth bit has been transmitted (i.e., after TBSC = 0), the transmit byte size counter is reset to 8 (for the next fill character or the first byte of the next frame, which is always 8 bits), and firmware checks the state of the TRURC routine. The subsequent action taken by firmware depends on whether or not the TRURC routine is in the OFF state, on whether or not a service request is pending, and on the relative states of TRDBEFLG, TRDBE, and TIFMS as summarized in Table 3-13. Firmware determines whether to send a flag or abort sequence where interframe fill characters are required, and loads the appropriate character in the transmit shift register.

3.5.2.3 TRSRC TEXT State Subroutine (Figure 3-9)

The TEXT state subroutine handles serialization and transmission of data bytes to the attached communications equipment. Related functions of this subroutine are:

1. Insertion of zeros where required for frame transparency
2. Recognition of byte boundaries
3. Recognition of end-of-frame
4. Generation of (closing) flags for frame terminations
5. Detection of underrun condition
6. Generation of abort sequence for underrun condition.

TEXT state subroutine processing begins by checking to see if a zero is to be inserted during this firmware pass (zeros are inserted after every fifth contiguous One bit for frame transparency). If five contiguous One bits have just been transmitted to the attached communications equipment, a zero is inserted into the transmit bit buffer. The next legitimate data bit is left in position 8 (TSR08) of the transmit shift register (to be transmitted during the next firmware pass through this subroutine). Firmware then goes on to see if a byte boundary has been reached.

If zero insertion is not required, the transmit byte size counter is decremented and the next data bit is placed in the transmit bit buffer. At this time the determination is made as to whether or not a zero is to be inserted during the next firmware pass through the TEXT state subroutine. If the data bit just placed in the transmit bit buffer is a zero (i.e., if TSE08 = 0), zero insertion will not be required on the next firmware pass. The zero insertion counter is initialized (to begin again the counting of contiguous One bits) and the subroutine goes on to see if a byte boundary has been reached.

If TSR08 = 1, the zero insertion counter is incremented and then checked to see if it is equal to zero (reflecting that five contiguous One bits have been transmitted). If the counter equals zero, the bit just transmitted was the fifth contiguous One bit, and zero insertion must be performed during the next firmware pass through this subroutine. Processing is now complete for this firmware pass, and an exit is made to the TRDRC routine.

Next, a check is made to see if the last legitimate data bit transmitted was the last bit of a byte. If it was not, the transmit shift register is shifted right to bring the next data bit into position TSR08. An exit is then made to the TRDRC routine with the next data bit in TSR08.

If the last legitimate data bit transmitted was the last bit of a byte, the end-of-frame flag is checked to see if the byte just transmitted was the last byte of a frame. If it was, the TRSRC routine is set to the IDLE state, a closing flag sequence is placed in the transmit shift register, the transmit byte size counter is set to 8, and an exit is made to the TRDRC routine. This flag sequence will be transmitted to the communications equipment during subsequent firmware passes through the TRSRC IDLE state subroutine.

If the byte just transmitted was not the last byte of a frame, firmware checks the input (transmit) buffer to see if it contains the next transmit byte. If it does not, an underrun condition has occurred because the MLC has failed to maintain the correct transfer rate. In this case the appropriate flags and the status register are updated, an abort sequence is loaded into the transmit shift register, and the TRURC routine is set to the REPORT UNDRN state. An exit is then made to the TRURC routine to report the underrun condition to the MLC. The TRSRC routine is set to the IDLE state, which is employed during subsequent firmware passes to transmit the abort sequence to the attached communications equipment.

If the next byte is ready, as indicated by both TRDBE and TRDBE FLG being reset, it is moved from the input (transmit) buffer to the transmit shift register. The Transmit Data Buffer Empty Flag (TRDBE) is set for use by the TRDRC routine to request more data from the MLC. After copying TEOF in TEOFS, and TBS in TBSC, the TRSRC routine is left in the TEXT state, and an exit is made to the TRDRC routine. This byte is transmitted, one bit at a time, during subsequent firmware passes through the TRSRC TEXT state subroutine.

3.5.3 Transmit Data Request Control Routine (Figure 3-10)

The Transmit Data Request Control (TRDRC) routine generates a transmit service request for more data from the MLC if the TRDBE flag is set, no service request is pending, and clear to send is true. If TRDBE is not set, if a transmit service request is pending, or if clear to send is not true, the request is not made, and an exit is made to the TRURC routine.

3.5.4 Transmit Underrun Report Control Routine (Figure 3-11)

The Transmit Underrun Report Control (TRURC) routine reports underrun status to the MLC or resets already reported underrun status. On entry, firmware checks to see if a transmit service request is pending. If true, an exit is made to the RCSRC routine. If no transmit service request is pending, firmware examines the TURCSTATE bits to determine the state of the TRURC routine and branches to execute the appropriate subroutine. If no underrun condition exists,

the routine is in the OFF state and an exit is made to the RCSRC routine to allow firmware to continue.

The TRURC routine is in the REPORT UNDRN state if an underrun detected during the TRSAC TEXT state subroutine must be reported to the MLC. In this case the Transmit Underrun Status (TUS) bit is set, the status register is copied to the output (receive) buffer, and a transmit service request is set as an indication to the MLC that it must read status. The TRURC routine is set to the RESET UNDRN state to allow this underrun status to be reset during the next firmware pass after the transmit service request has been reset (i.e., after the MLC has read the underrun status). An exit is then made to the RCSRC routine.

The TRURC routine is set to the RESET UNDRN state during the REPORT OVRN state subroutine. When TRURC is entered the firmware checks to see if the MLC has read the previously reported underrun status. If the MLC has read status, the transmit service request will be reset, and the RESET UNDRN state subroutine is entered. The TUS bit is reset, and the TRURC routine is set to the OFF state, indicating that the status of a detected underrun condition has been read by the MLC. Next, the current status is copied to the output (receive) buffer, the transmit data buffer empty flip-flop is set, and an exit is made to the RCSRC routine.

3.5.5 Receive Shift Register Control Routine

The Receive Shift Register Control (RCSRC) routine consists of an enter sequence plus eight subroutines (called state subroutines) for recognition and handling of a variety of receive operation conditions. This routine handles the transfer of received data and frame check sequences from the bit buffer to the output (receive) buffer for subsequent transfer to the MLC. Other related functions include deletion of inserted Zeros, detection and deletion of flag, abort and idle link state sequences, overrun detection, and end-of-frame detection.

The RCSRC routine is entered from the Transmit Data Request routine when that routine has just issued a transmit data service request, or from the Transmit Underrun Request Control routine when the MLC has failed to deliver more data to the input (transmit) buffer in time.

The RCSRC routine normally exits to the Receive Data Request Control (RCDRC) routine to determine whether or not data is ready for transfer to the MLC. When the RCSRC routine detects an overrun condition, the data is not to be transferred, and the exit is to the Received Flush Control (RCFCO) routine for further analysis and for reporting of the overrun condition.

Figure 3-5 illustrates the relationships among the eight RCSRC state subroutines. Once the RCSRC routine is entered the enter sequence branches to the state subroutine identified by the bit configuration of the RSRCSTATE register in the internal RAM (set during the previous pass through the firmware loop). Each RCSRC state subroutine performs particular functions as described in the following subparagraphs, and is capable of setting the RCSRC and RCFCO

routines to other states as necessary. The state of the RCSRC routine when exited is the state subroutine that will be executed during the next pass through the firmware loop.

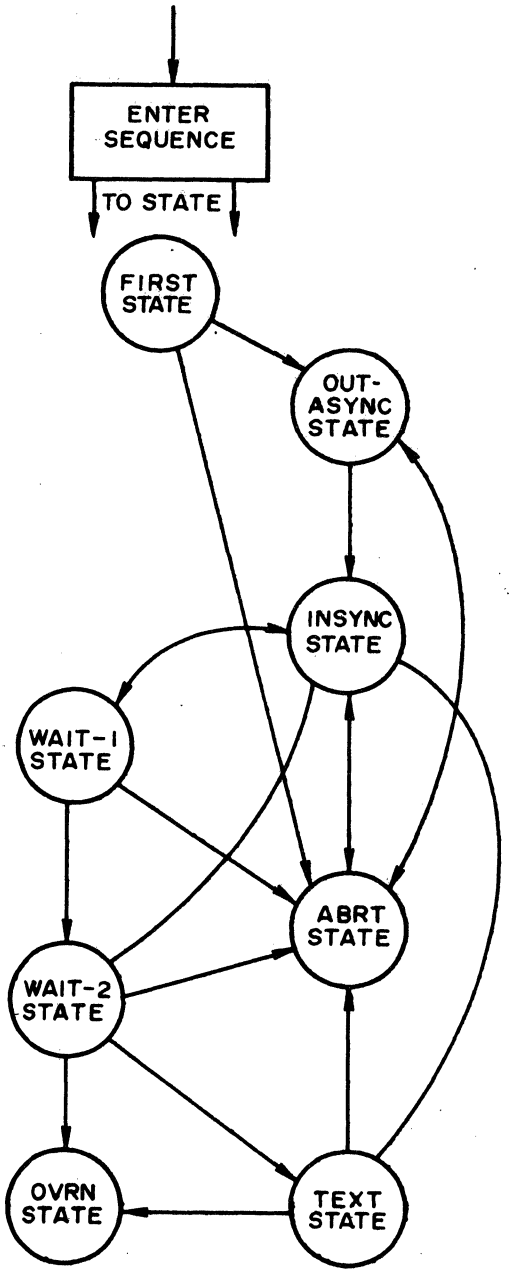


Figure 3-5 RCSRC State Subroutine Relationships

3.5.5.1 RCSRC Routine Enter Sequence
(Figure 3-12)

At each pass of the firmware, the receiver ready flip-flop RCRDYZ+ is tested to see if the next data bit has been shifted into the bit buffer. If reset, the next data bit is not ready, and no further processing is required. Therefore, an exit is made to the RCDRC routine to determine whether or not a previously received data byte is ready for transfer to the MLC. If the flip-flop is set, it and the Flag Received (FLGRCVD) and Abort Received (ABRTRCVD) flag bits are reset at the start of processing for the next bit.

Next, the zero deletion counter (R5C) is checked to see if an inserted zero is to be deleted. (The zero deletion counter is used and maintained during the WAIT-1, WAIT-2, and TEXT state subroutines. The zero deletion counter is updated during each firmware pass, and the decision on whether or not to delete a zero is also made during each firmware pass. Thus, when a zero is to be deleted, the indication is set in the counter during one firmware pass, and the actual deletion occurs during the next firmware pass.) If zero deletion is required, it is accomplished by shifting only the Flag Register (RFR) in the internal RAM. The right-most bit, the inserted zero, is thereby dropped while the next bit is being loaded into the left-most bit of the flag register. If zero deletion is not required, all eight receive shift registers are shifted.

Once the new bit has been loaded into the flag register, the flag register is examined to determine whether it contains data, or a flag or abort sequence. The Flag Received flag (FLGRCVD) or Abort Received flag (ABRTRCVD) is set if appropriate. These flags are tested at the beginning of each RCSRC state subroutine.

Finally, the current state of the Receive Shift Register Control routine is tested by examination of the RSRCSTATE register in the internal RAM, and a branch is made to the appropriate state subroutine.

3.5.5.2 RCSRC FIRST State Subroutine
(Figure 3-13)

The FIRST state subroutine is entered only after initialization, and is used to determine whether or not the MLC has read the firmware revision information that was loaded into the output (receive) data buffer during the initialization routine. This is accomplished by checking to see if the output (receive) data buffer is full. If it is full, the MLC has not yet read the firmware revision information and an exit is made to the TRSRC routine (effectively bypassing the receive routines until all firmware revision information has been read by the MLC).

Once the MLC has read the firmware revision information, the receive routines are included in the firmware loop. During each pass of firmware, one data bit is loaded into the flag register in the internal RAM, and the Receive Byte Size Counter (RBSC) is incremented by one. After the eighth bit has been loaded (i.e., RBSC = 8), the counter is reset to zero, and the flag register is

checked to see if it contains an abort sequence. If it does, the RCSRC routine is set to the ABORT state and an exit is made to the RCDRC routine. If the flag register does not contain an abort sequence, the RCSRC routine is set to the OUTASYNC state, and an exit is made to the RCDRC routine.

3.5.5.3 RCSRC OUTASYNC State Subroutine (Figure 3-14)

The OUTASYNC state can be set during either the FIRST state or the ABRT state subroutine. After resetting the Receive Byte Size Counter (RBSC) to zero, the flag (FLGRCVD) and Abort (ABRTRCVD) received flags are checked. If FLGRCVD is set, firmware sets the RCSRC routine to the INSYNC state and then exits to the RCDRC routine. If ABRTRCVD is set, firmware sets the RCSRC routine to the ABRT state and then exits to the RCDRC routine. RCSRC remains in the OUTASYNC state if neither flag is set.

3.5.5.4 RCSRC ABRT State Subroutine (Figure 3-15)

The ABRT state is set during any other RSCRC state subroutine whenever that subroutine finds the Abort Received (ABRTRCVD) flag set (indicating receipt of an abort sequence in the flag register). The initial function of the ABRT state subroutine is to monitor the Receive Byte Size Counter (RBSC) until it indicates receipt of eight new bits in the flag register (i.e., when the counter reaches eight). On each firmware pass, the RBSC counter is incremented and then tested. If it is not equal to a count of eight, an exit is made to the RCDRC routine, and firmware continues its loop, loads the next receive bit in the flag register, and returns to the RCSRC ABRT state subroutine.

After eight firmware passes the abort sequence (seven ones) has been shifted out of the flag register and eight new bits have been loaded. Next, the ABRTRCVD flag is checked to see if another abort sequence has been received. If set, the low order bit of the flag register (RFR08) is checked. If RFR08 is a one, then the flag register contains eight ones. These eight ones and the seven ones of the previous abort sequence constitute fifteen ones, or an idle link state sequence. The Receive Idle Link State (RILSFLG) flag is, therefore, set to alert the Receive Flush Control (RCFCO) routine of the idle link state condition, and an exit is made to the RCDRC routine.

If the new bits form a flag sequence, the RCSRC routine is set to the INSYNC state, and an exit is made to the RCDRC routine. If the new bits form neither a flag nor an abort sequence, the RCSRC routine is set to the OUTASYNC state, and an exit is made to the RCDRC routine.

3.5.5.5 RCSRC INSYNC State Subroutine
(Figure 3-16)

The INSYNC state is set during any other RCSRC subroutine (except FIRST) whenever that subroutine finds the Flag Received (FLGRCVD) flag set (indicating receipt of a flag sequence in the flag register). If FLGRCVD is set when this subroutine is first entered, the Receive Byte Size Counter (RBSC) is reset to zero (to allow it to count another eight bits as they are received and loaded into the flag register), the RCSRC routine is left in the INSYNC state, and an exit is made to the RCDRC routine. The firmware continues in its loop, the FLGRCVD flag is reset, and eight more bits are received. After each bit is received, firmware returns in its loop to the INSYNC state, increments the receive byte size counter, and then tests it to see if eight more bits have been loaded into the flag register. If eight bits have not been loaded, an exit is made to the RCDRC routine to allow firmware to continue.

Although the INSYNC state subroutine checks the FLGRCVD and ABRTRCVD flags during each firmware pass, these flags cannot be found set until after the eighth bit has been loaded into the flag register. If a flag or abort sequence is received, the appropriate flag will be set by the RCSRC routine enter sequence. If either flag is found set at this time, the RCSRC routine is set to the appropriate state (i.e., either set to ABRT state or left in INSYNC state), and an exit is made to the RCDRC routine. If neither flag is found set, it is an indication that the first byte of a new frame is now in the flag register. The receive byte size counter and zero deletion counter are both set to zero, the RCSRC routine is set to the WAIT-1 state, and an exit is made to the RCDRC routine to allow firmware to continue in its loop.

3.5.5.6 RCSRC WAIT-1 State Subroutine
(Figure 3-17)

The WAIT-1 state is set during the RCSRC INSYNC state subroutine and is entered during the subsequent firmware pass with the first byte of the frame in the flag register. At each firmware pass, processing begins by testing the Flag Received (FLGRCVD) and Abort Received (ABRTRCVD) flags. Normally, neither flag is set, but in the case of a false start or aborted frame, one of the flags can be set. If either flag is set, the Receive Byte Size Counter (RBSC) is set to zero, and the RCSRC routine is set to either the INSYNC or ABRT state.

Next, a test is made to determine whether or not a zero was just deleted. If not, the zero deletion control counter is updated (as a function of the new bit just shifted into RFCSR), and the byte size counter RBSC is incremented.

If a zero was just deleted or if a zero is not going to be deleted in the next cycle, the byte size counter is tested to see if 16 shifts have been made (RBSC=16). If not, processing is complete for the cycle, and the firmware exits to the RCDRC routine. If true, the first byte has been completely shifted into the FCS register, and firmware sets the RCSRC routine to the WAIT-2 state and then exits to the RCDRC routine.

3.5.5.7 RCSRC WAIT-2 State Subroutine (Figure 3-17)

The WAIT-2 state is set during the WAIT-1 state subroutine after the first byte has been loaded into the right half of the FCS register (RFCSR). At each firmware pass, processing begins by testing the Flag Received (FLGRCVD) and Abort Received (ABRTRCVD) flags. Normally, neither flag is set, but in the case of a false start or aborted frame, one of the flags can be set. If either flag is set, the Receive Byte Size Counter (RBSC) is set to zero, and the RCSRC routine is set to either the INSYNC or ABRT state.

Next, a test is made to determine whether or not a zero was just deleted. If not, the zero deletion control counter is updated (as a function of the new bit just shifted into RFCSR), and the byte size counter (RBSC) is incremented.

If a zero was just deleted or if a zero is not going to be deleted in the next cycle, the byte size counter is tested to see if eight shifts have been made (RBSC=8). If not, processing is complete for this cycle, and an exit is made to the RCDRC routine. If true, the first byte has been completely shifted into the Receive Byte Register (RBR) in the internal RAM.

The first byte of the frame is now ready to be sent to the MLC. However, a test is made first to be sure that the previous frame has been completely processed, as indicated by the Receive Flush Control (RCFCO) routine being in the OFF state. If the RCFCO routine is not in the OFF state, an overrun condition has occurred. When this happens the RCSRC routine is set to the OVRN state, and an exit is made to the RCDRC routine.

If the RCFCO routine is in the OFF state, the first byte is transferred to the output (receive) data buffer. Before exiting to the RCDRC routine, the following flags are set:

1. RCDBFFLG - Receive Data Buffer Full Flag is set for use in the RCDRC routine.
2. RILS - Receive Idle Link State flag is reset, since the next interrupt will not be for the idle link state.

Finally, the RCSRC routine is set to the TEXT state, and the exit is made to the RCDRC routine.

3.5.5.8 RCSRC TEXT State Subroutine (Figure 3-18)

The TEXT state is set during the RCSRC WAIT-2 state subroutine, with the second byte of the frame in the right half of the receive history register (RFCSRH) and the byte register empty. As in other RCSRC state subroutines, FLGRCVD and ABRTRCVD are checked at the beginning of each cycle. However, here these conditions indicate the end of a valid frame, whereas in other states they do not. In the case of a flag (FLGRCVD), the start of the next frame could be indicated. When the TEXT state subroutine is entered, the information just received can be data (i.e., not an end of frame), a closing flag, or a closing abort sequence. Each of these conditions is described in the following subsections.

3.5.5.8.1 Not End of Frame in Text State

This sequence occurs each cycle when neither a flag nor an abort is detected. Initial zero deletion processing is as described in subsection 3.5.5.7. If this cycle is not completed by the zero deletion control, the byte size counter is incremented, and a test is made to see if a zero is to be deleted in the next cycle. If not, the byte size counter is compared with the byte size specified in the input (transmit) buffer read receive control word. If unequal, processing for this cycle is complete except for a comparison with the maximum byte size (which is eight). If RBSC equals eight, it indicates an overrun condition (i.e., that the MLC has reduced the byte size after the HDLC has already passed the new value). The RCSRC routine is set to the OVRN state, and an exit is made to the Receive Flush Control routine.

If the byte size counter equals the byte size specified in the control word, the new byte which is in the byte register (RBR) can be sent to the MLC. First a check is made to see if the previous byte has been taken by the MLC, as indicated by the Receive Data Buffer Full flip-flop (RCDBFZ-) and Receive Data Buffer Full flag (RCDBFFLG) both being reset. If either one is set, the RCSRC routine is set to the OVRN state.

If the previous byte has been taken by the MLC, the following functions are performed.

1. The byte in the byte register (RBR in internal RAM) is right-justified if it is less than 8 bits (the left bits are set to zero).
2. The byte is moved to the output (receive) data buffer (RDB).
3. The buffer full flag is set for use in the Receive Data Request routine.
4. The byte size counter is reset to zero.

Processing of the not-end-of-frame condition is now complete, and an exit is made to the RCDRC routine.

3.5.5.8.2 Closing Flag Received in TEXT State

This sequence occurs when, upon entry, the TEXT state subroutine finds that a flag sequence has just been received from the communications device as an indication of the end of a frame. First, the zero deletion counter is checked. If a zero was not just deleted, the byte size counter is incremented since a new bit was just shifted into the FCS and byte registers. If a zero was just deleted, the byte size counter is not decremented (since the FCS and byte registers were just shifted).

Next, the byte size counter is incremented by eight, in anticipation of a possible right-justification procedure. If the byte size count now equals sixteen (zero) right-justification is not necessary. If the byte size is less than sixteen (zero), the byte register is right-justified. Finally, the last data byte and both halves of the frame check sequence (FCS) are saved in the receive history registers in the external RAM. After setting the RCFCO routine to the LAST DATA state and the RCSRC routine to the INSYNC state, an exit is made to the RCDRC routine.

3.5.5.8.3 Closing Abort Received in TEXT State

This sequence occurs when, upon entry, the TEXT state subroutine finds that an abort sequence has just been received from the communications equipment as an indication of the end of a frame. In this case it is not necessary to save the last data byte and the two FCS bytes, as these are discarded when a closing abort terminates a frame. In this sequence, the Receive Data Buffer Full flag (RCDBFFLG) is reset, the byte size counter is set to zero, and both the Flush Control (RCFCO) and Receive Shift Register Control (RCSRC) routines are set to the ABRT state. An exit is then made to the flush control routine to load status into the output (receive) buffer for subsequent reporting to the MLC.

3.5.5.9 RCSRC OVRN State Subroutine (Figure 3-19)

The OVRN state can be set during the WAIT-2 or TEXT state subroutines, where an overrun condition is detected during a receive operation. Processing begins by checking FLGRCVD and ABRTRCVD. If the present data is neither a flag nor an abort sequence, no further processing is necessary and an exit is made to the RCDRC routine.

If ABRTRCVD is set and data has not been sent to the MLC, no status report is required, and an exit is made to the RDCRC routine.

If ABRTRCVD is set and data has been sent to the MLC, the idle link state flag (RILS) is reset, the RCFCO routine is set to the ABRT&OVRN state, and an exit is made to the RCDRC routine.

If FLGRCVD is set, the idle link state flag (RILS) is reset. The state of the Flush Control routine is then checked to see if it is in the OFF state. RCFCO is OFF if OVRN was entered from the TEXT state. If OVRN was entered from the WAIT-2 state, RCFCO may or may not be in the OFF state. If the RCFCO routine is in the OFF state, it is now set to the FLAG&OVRN state, and an exit is made to the RCDRC routine. If RCFCO is not in the OFF state, RILSFLG is reset, and ROSD is set to notify RCFCO routine that an entire frame has been missed.

For all of the above conditions the appropriate status register bits are set later, during the Flush Control routine.

3.5.6 Receive Data Request Control Routine (Figure 3-20)

This routine is employed to determine whether or not data is ready for transfer to the MLC and, if it is, to generate a service request to initiate the transfer. At each pass of the firmware the Receive Data Buffer Full flag (RCDBFFLG) is checked to see if the RCSRC routine has loaded a byte into the output (receive) data buffer. If data is ready, the receive service request is checked to determine whether or not the previous transfer has been completed. If it has, the Receive Data Buffer Full flip-flop (SRCDBF) is set, the buffer full flag is reset, and a receive service request is made.

3.5.7 Receive Flush Control Routine (Figure 3-21)

The Receive Flush Control Routine (RCFCO) consists of an enter sequence plus eight subroutines (called state subroutines) for the reporting of status and the transfer of the last data byte and the Frame Check Sequence (FCS) to the MLC. Information to be sent to the MLC is placed in the output (receive) buffer for subsequent transfer to the MLC. When reporting status, this routine waits until after the MLC has read status from the output (receive) buffer and then resets status in both the status register (in the external RAM) and the output (receive) buffer.

3.5.7.1 RCFCO Enter Sequence

At each pass of the firmware, the receive service request flip-flop (RCSRQZ+) is first checked. If set, there is no reason to proceed further in this routine because the MLC has not yet read the previous status or data. If no service request is pending, the RFCSTATE bits are examined to determine the state of the Receive Flush Control routine, and a branch is made to the appropriate subroutine.

3.5.7.2 RCFCO OFF State Sequence

This subroutine is entered during the first firmware pass following initialization and from the RCFCO Reset Status state subroutine. The purpose of this subroutine is to see if the Receive

Shift Register Control routine has made an idle link state interrupt request, and if it has, and if the last service (interrupt) request was not an idle link state interrupt, to make one.

First, the Receive Idle Link State flag (RILSFLG) is checked. If set, Idle Link State (RILS) is checked to see if the last interrupt was an idle link state interrupt. If it was not, RILS is set, Abort Status (RAS) is set to indicate the Idle Link state, and RAS is set in the status register. The status register, which also contains other transmit status information, is then copied into the output (receive) status buffer. A Receive Service Request (RCSRQZ+) is then made and the RCFCO routine is set to the Reset Status State to await completion of the status transfer.

3.5.7.3 RCFCO Last Data State Subroutine

The RCSRC routine changes the RCFCO state from the OFF to Last Data when receipt of a closing flag terminates a non-overrun frame. The Last Data, First FCS, and Second FCS state subroutines then transfer the last data byte, the two FCS bytes, and closing status (Flag, no overrun, no abort) to the MLC. When this subroutine is entered, the last data byte has been placed in the Receive History Register (RBRH) and the FCS in the Receive History Register (RFCSRH) in the external RAM.

Normally, whenever the Last Data state is entered (i.e., when the Receive Service Request (RSCRQZ+ is reset), all data up to the last data byte will have been completely transferred to the MLC. However, there is a period between the time that the third-from-last byte is transferred (SRCDBF and RCSRQZ reset) and the time that the receive data request control routine is entered to set RCSRQZ again for the second-from-last transfer, when the previous transfer may not have been finished. Thus, the receive data buffer full flag (RCDBFFLG) is also checked for completeness.

When ready, the last data byte is moved from the history register in the external RAM to the Receive Data Buffer (RDB) in the internal RAM, and REOFS is set in the status register. Status is copied to the output (receive) buffer, and a service request is made. The RCFCO routine is then set to the First FCS state and the flush control routine is exited. The First FCS state subroutine will not be entered until the Receive Service Request (RCSRQZ) is reset, indicating completion of this transfer.

3.5.7.4 RCFCO First FCS State Subroutine

Processing in this subsection begins after the last data byte has been transferred by the MLC, as indicated by the RCFCO routine being set to the First FCS state and the service request being reset. The first (right) half of the Frame Check Sequence (FCS) is moved from the receive history register to the output (receive) data buffer. (The first time that the MLC finds the closing flag set, it knows that it must input the first half of the FCS and switch back to 8-bit mode for FCS processing. Also, the specified byte size for the HDLC adapter is switched to eight bits in preparation for the next frame.) Finally, a service request is set, RCFCO routine is

set to the Second FCS state, and an exit is made to the TRSRC routine. Processing of the RCFCO Second FCS state subroutine will not begin until the receive service request has been reset by the MLC (indicating that the MLC has transferred the first half of the FCS).

3.5.7.5 RCFCO Second FCS State Subroutine

Processing in this subroutine begins after the first half of the FCS has been transferred to the MLC, as indicated by the RCFCO routine being in the Second FCS state and the service request being reset.

The second (left) half of the FCS is moved from the receive history register to the output (receive) data buffer.

The service request is set, and the RCFCO routine is set to the Reset Status state. An exit is then made to the TRSRC routine. Processing of the RCFCO Reset Status state subroutine will not begin until the service request has been reset by the MLC as an indication that it has transferred the second half of the FCS.

3.5.7.6 RCFCO Reset Status State Subroutine

The Reset Status subroutine is the last RCFCO subroutine for the processing of any frame. Its function is to reset the status register, thereby eliminating the need for the RCDRC routine to examine these status bits, and then to see if a following frame has already ended (an overrun condition) and to process it if it has. This RCFCO state can be set in one of the following manners:

1. During the RCFCO OFF state subroutine, when an idle link state interrupt is taking place.
2. During the RCFCO Second FCS state subroutine, when the last transfer of a normal frame is taking place.
3. During the RCFCO ABRT&OVRN, ABRT, or FLAG&OVRN status subroutine, when an abort and/or overrun status transfer is taking place.

Processing depends on whether or not the Receive Overrun Status Delayed (ROSD) has been set by the RCSRC routine. If not, then overrun of the next frame has not occurred. The RCFCO routine is set to the OFF state, status is copied to the output (receive) buffer, and the RCFCO OFF state subroutine is entered to check for an idle link state request.

If ROSD is set, another status transfer must be made. The overrun status bit (ROS) is set, and the status register is copied to the output (receive) buffer. ROSD is reset, and a service request is made to send status to the MLC. An exit is then made to the TRSRC routine. The Flush Control routine remains in the Reset Status state. The Reset Status state subroutine is executed again during the firmware pass following the transfer of status to the MLC.

3.5.7.7 RCFCO ABRT&OVRN State Subroutine

The ABRT&OVRN state is set during the RCSRC routine when an overrun frame is ended with an abort sequence. Processing begins by setting REOFS, RAS, and ROS in the status register, after which the status register is copied to the output (receive) buffer for subsequent transfer to the MLC. The RCFCO routine is set to the Reset Status state, and a service request is made to send status to the MLC. Processing of the Reset Status state subroutine takes place on the first firmware pass after the MLC has transferred the status

3.5.7.8 RCFCO FLG&OVRN State Subroutine

The FLG&OVRN state is set during the RCSRC routine when an overrun frame is ended with a flag sequence. Processing is the same as for the ABRT&OVRN subroutine, except that different status bits (REOFS, ROS) are set.

3.5.7.9 RCFCO Abort State Subroutine

The RCFCO Abort state is set during the RCSRC TEXT state subroutine when a frame that has not overrun is terminated by an abort sequence. Processing is the same as for the ABRT&OVRN state subroutine, except that different status bits (REOFS, RAS, ROS) are set.

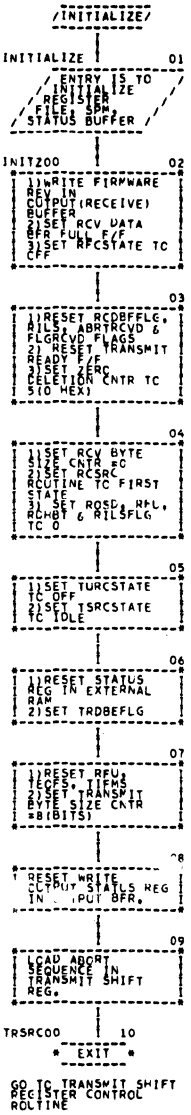


Figure 3-6 Initialize Routine

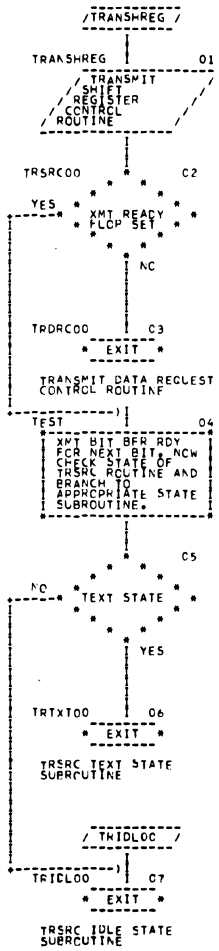


Figure 3-7 TRSRC Enter Sequence

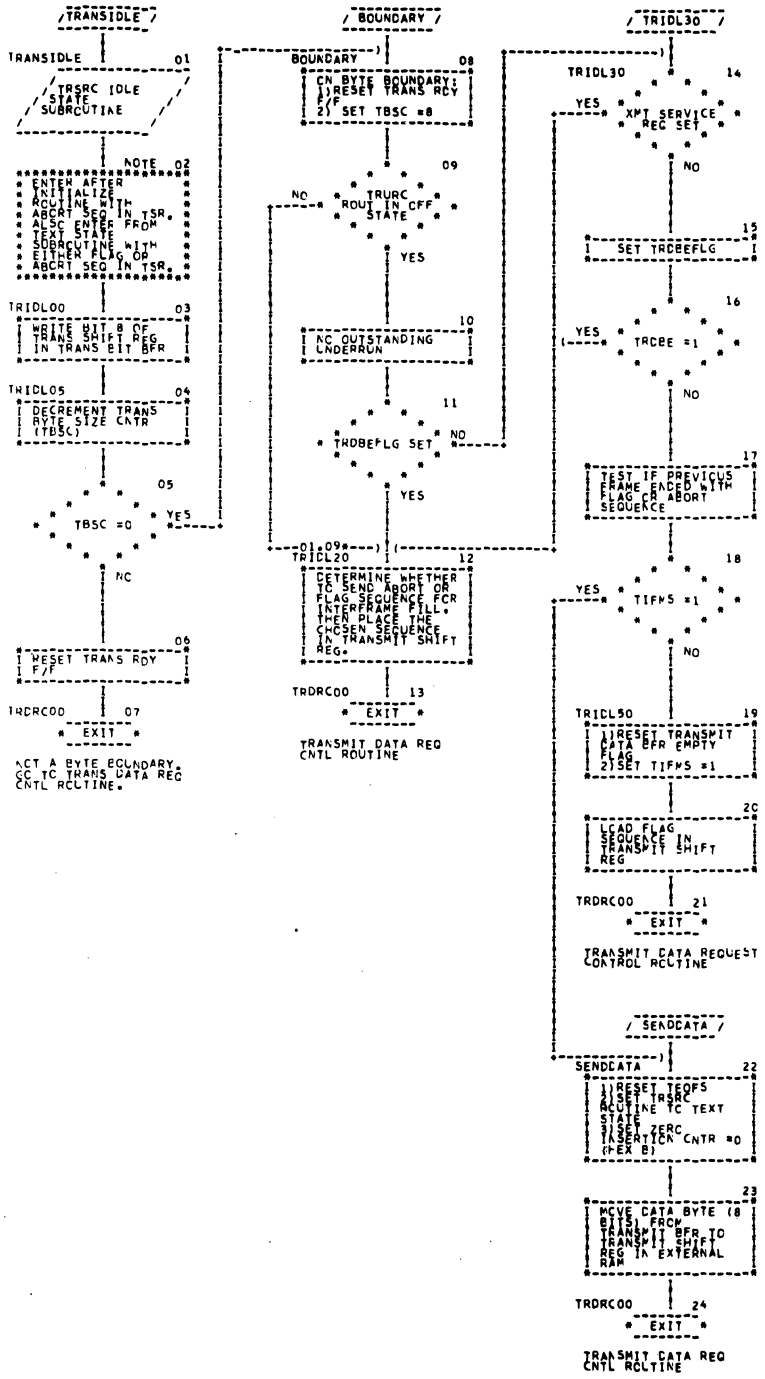


Figure 3-8 TRSRC Idle State Subroutine

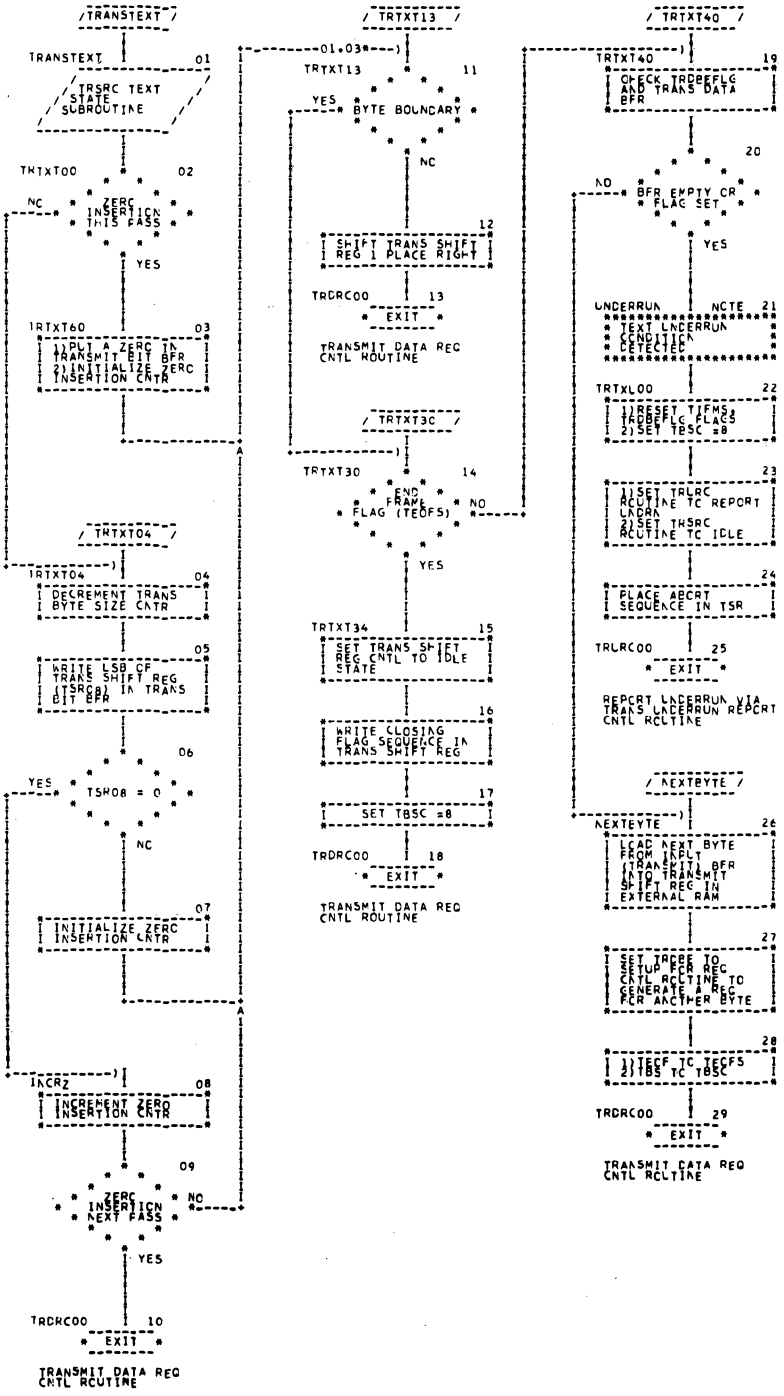


Figure 3-9 TRSRC TEXT State Subroutine

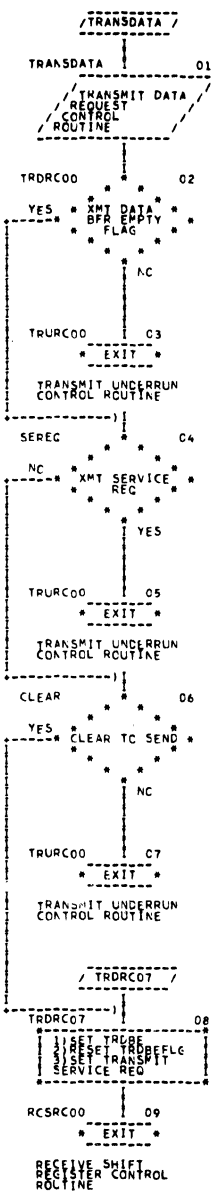


Figure 3-10 TRDRC Routine

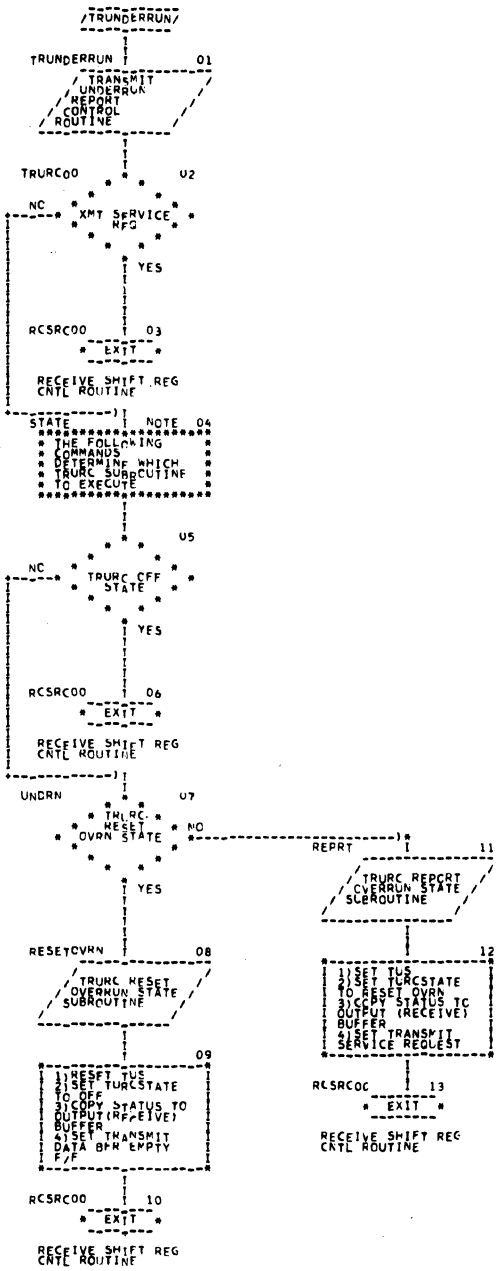


Figure 3-11 TRURC Routine

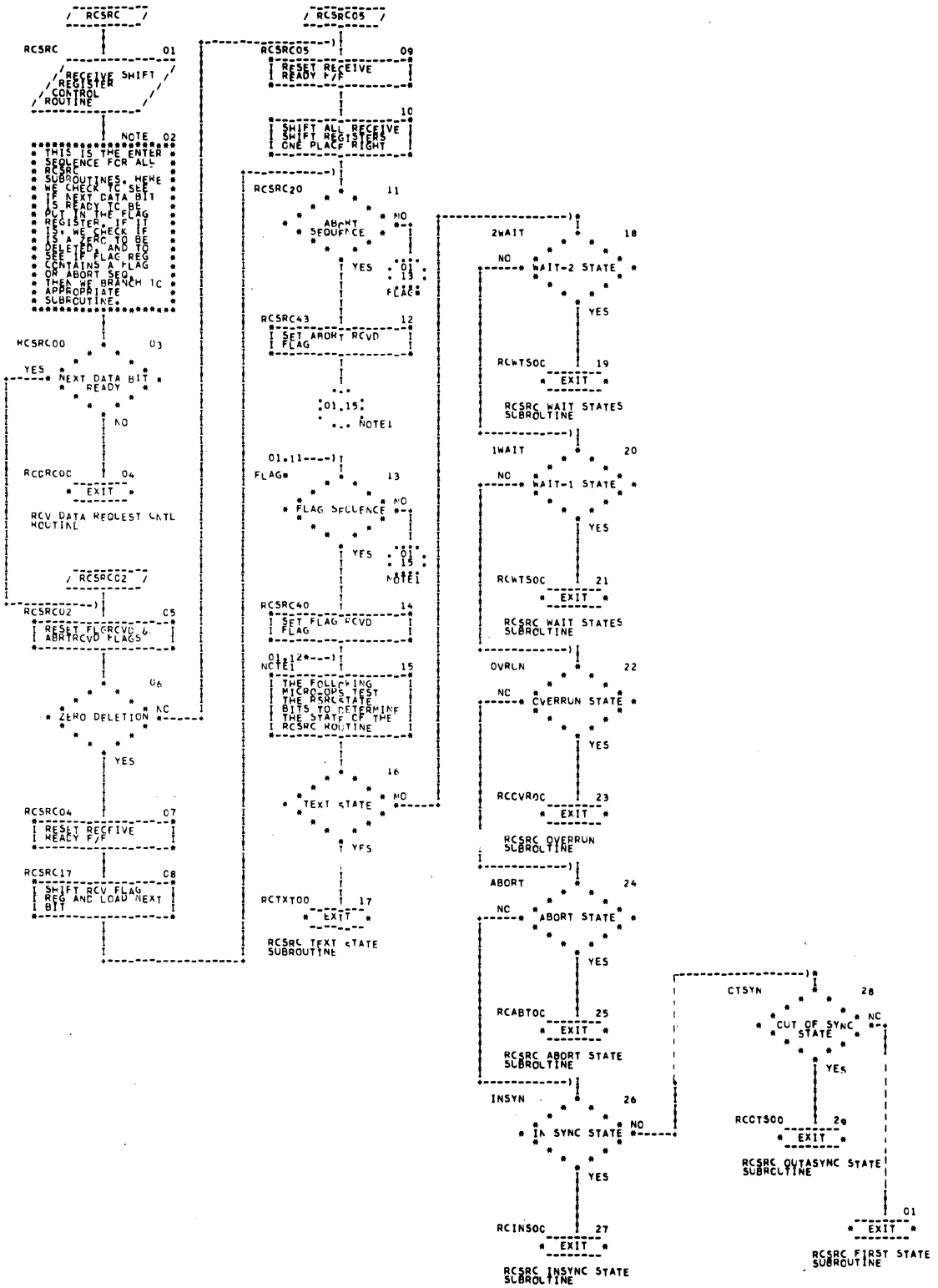


Figure 3-12 RCSRC Routine Enter Sequence

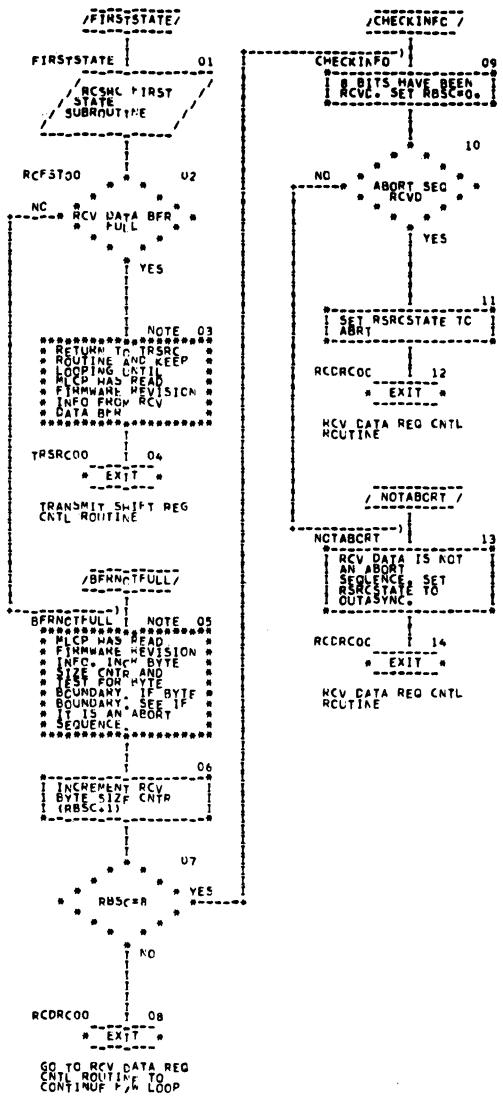


Figure 3-13 RCSRC FIRST State Subroutine

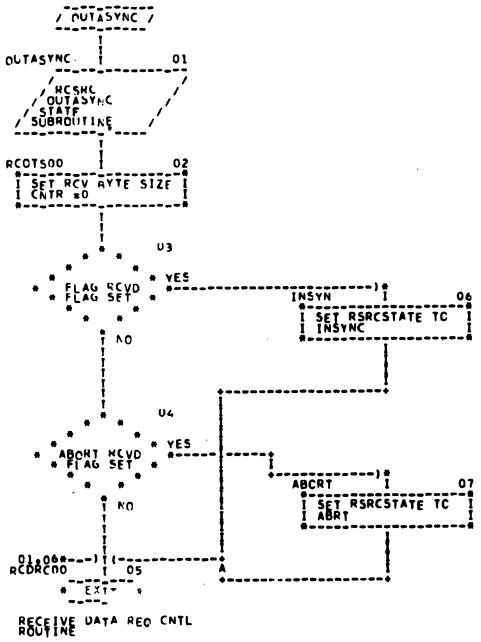


Figure 3-14 RCSRC OUTASYNC State Subroutine

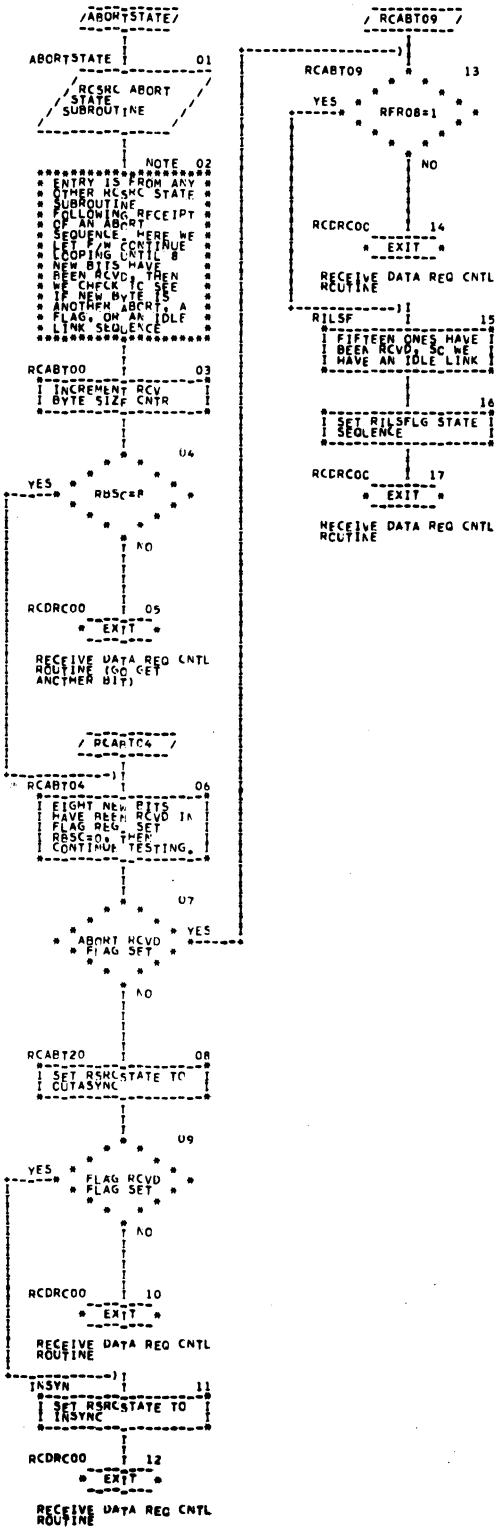


Figure 3-15 RCSRC ABRT State Subroutine

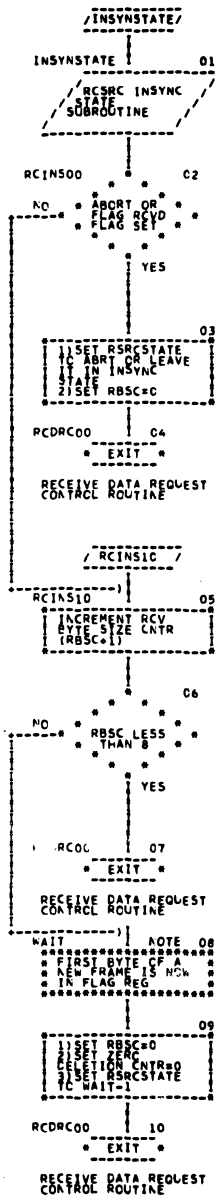


Figure 3-16 RCSRC INSYNC State Subroutine

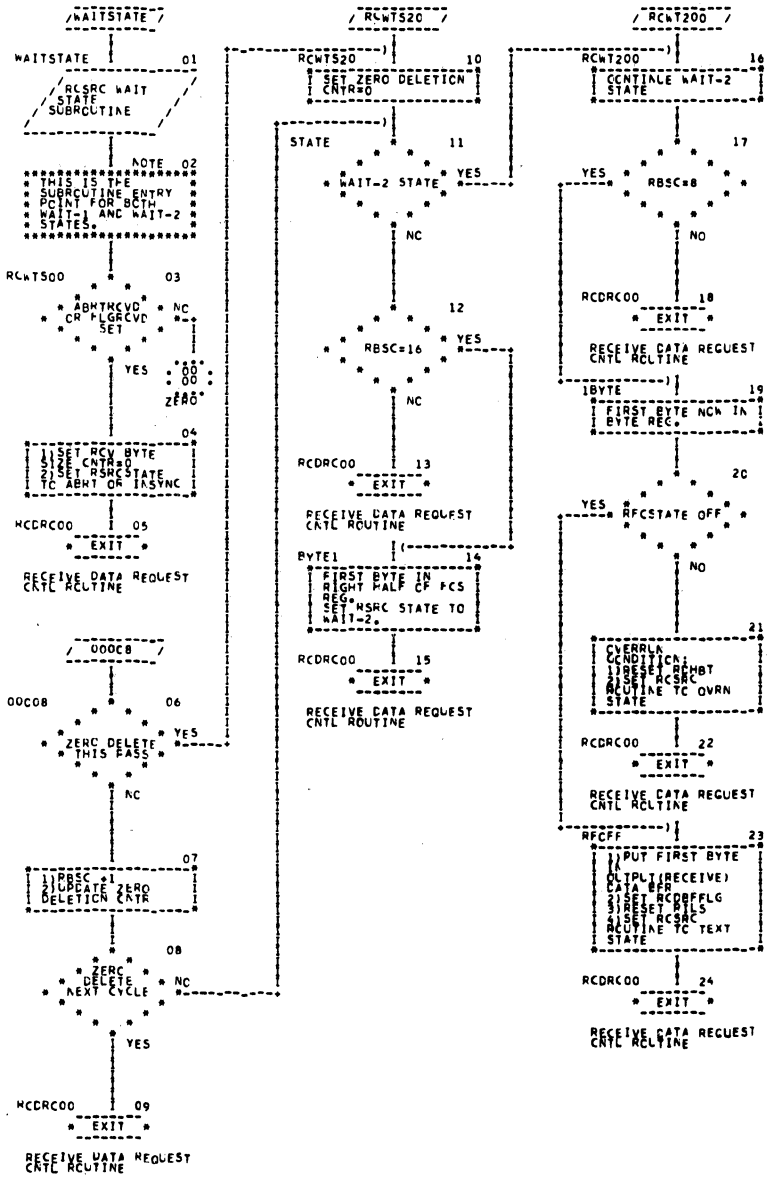


Figure 3-17 RCSRC WAIT States Subroutine

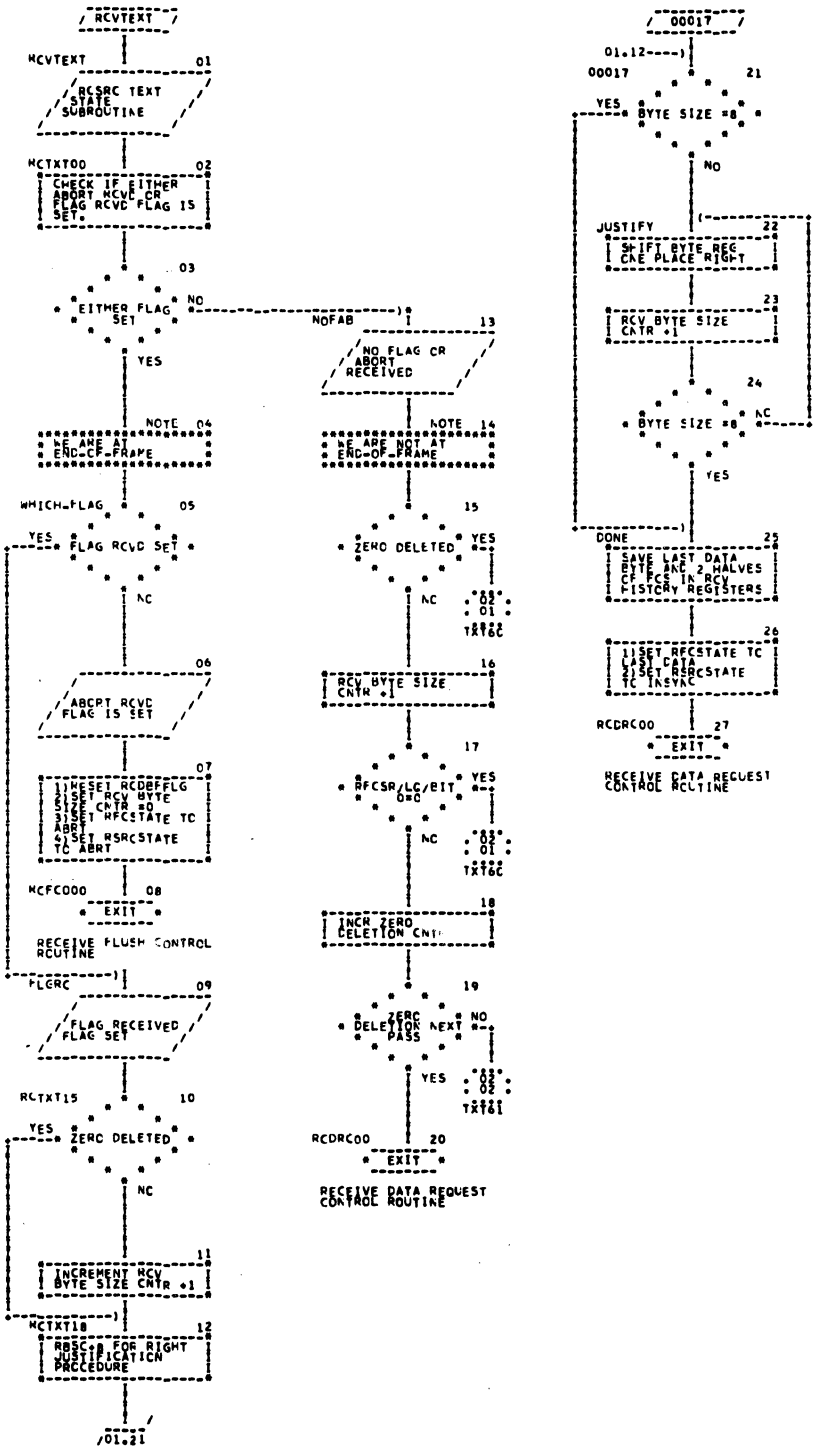


Figure 3-18 RCSRC TEXT State Subroutine (Sheet 1 of 2)

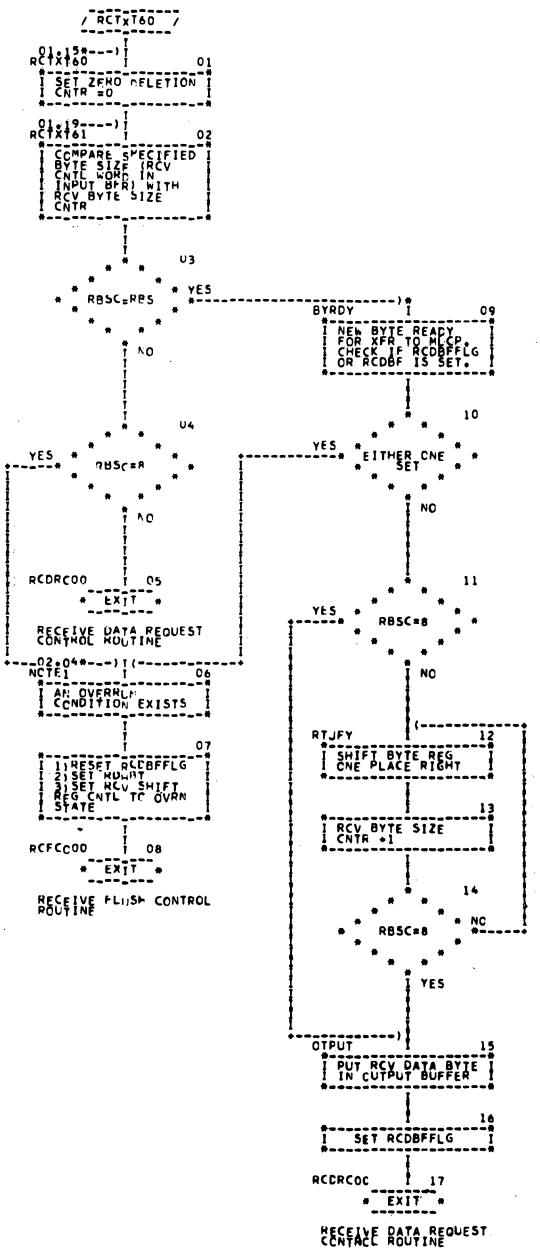


Figure 3-18 RCSRC TEXT State Subroutine
(Sheet 2 of 2)

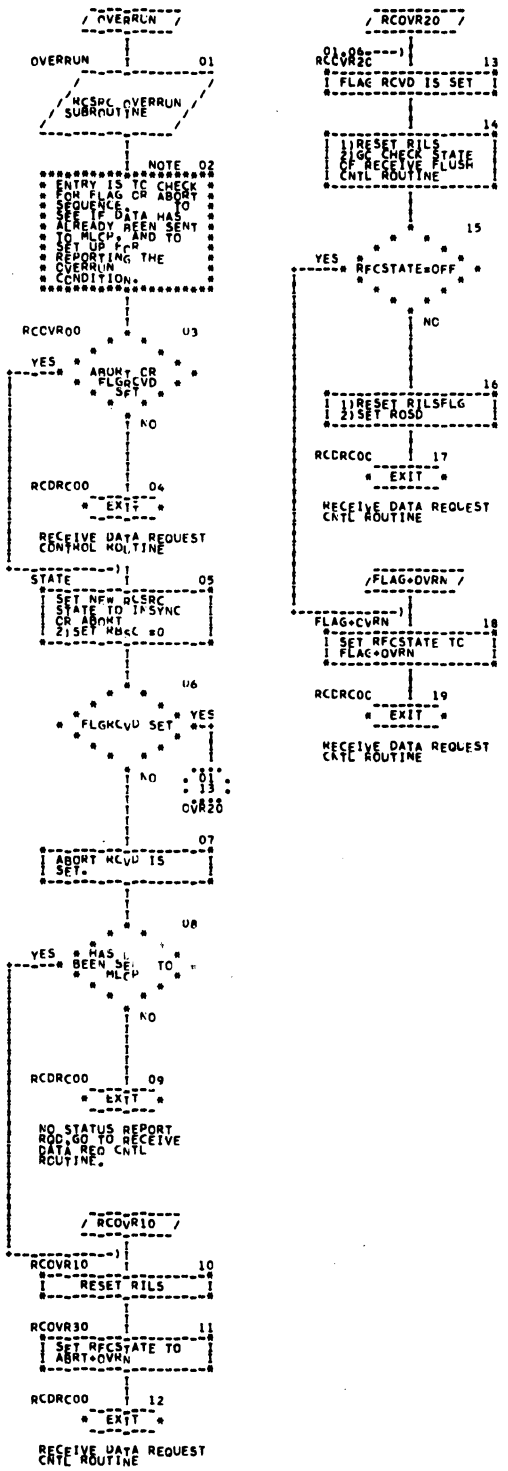


Figure 3-19 RCSRC OVRN State Subroutine

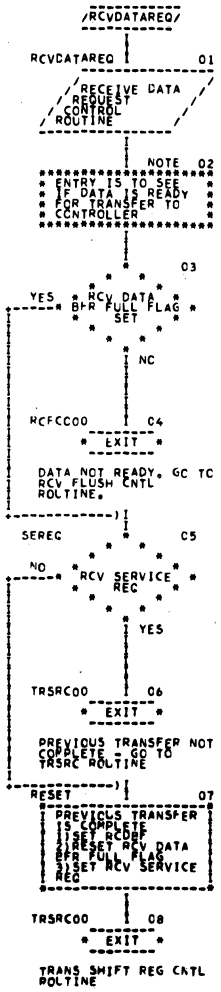


Figure 3-20 RCDRC Routine

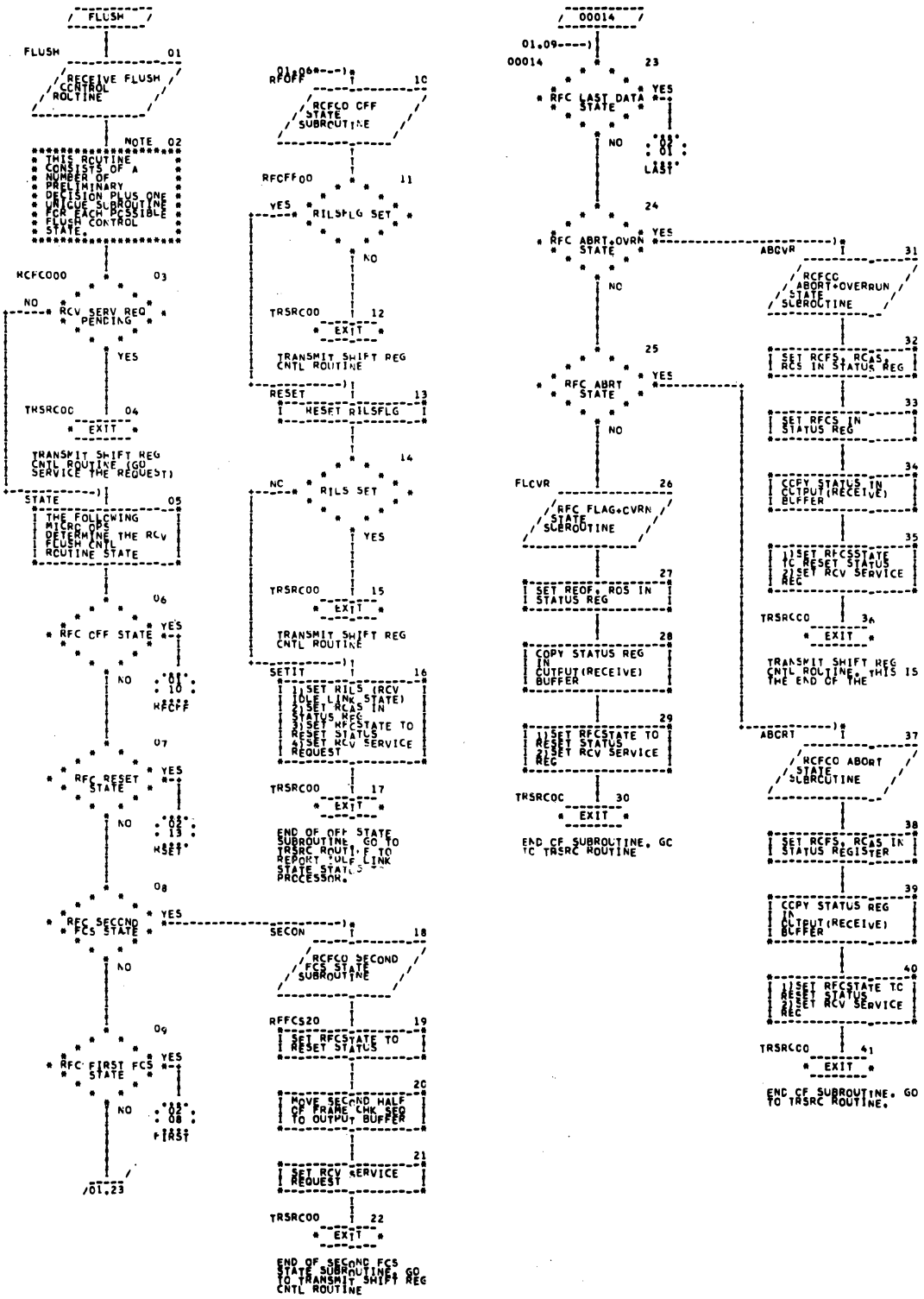


Figure 3-21 RCFCO Routine
(Sheet 1 of 2)

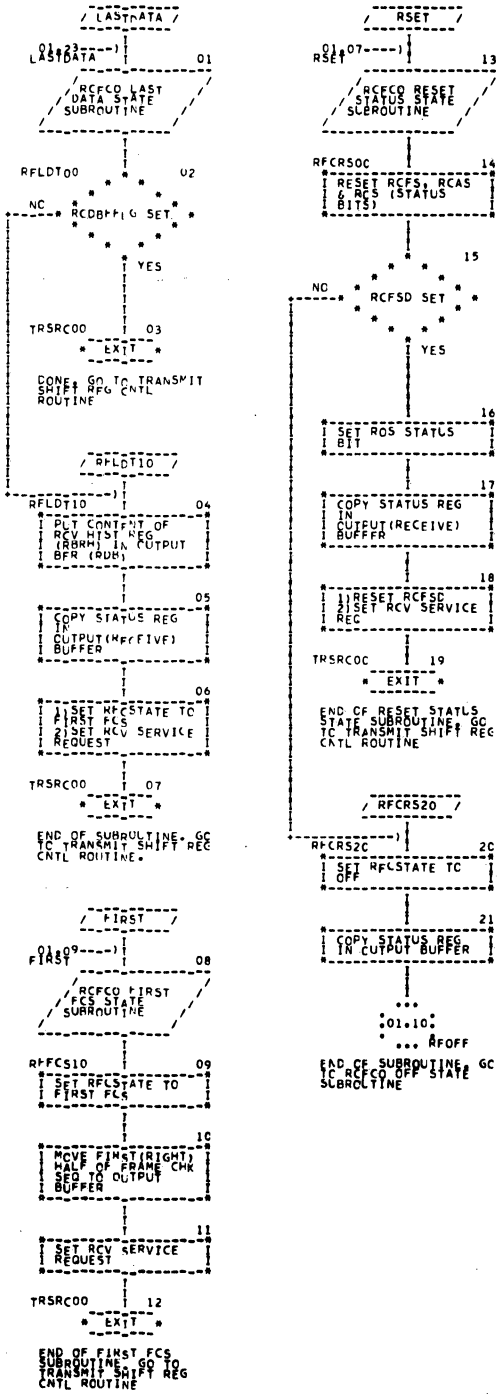
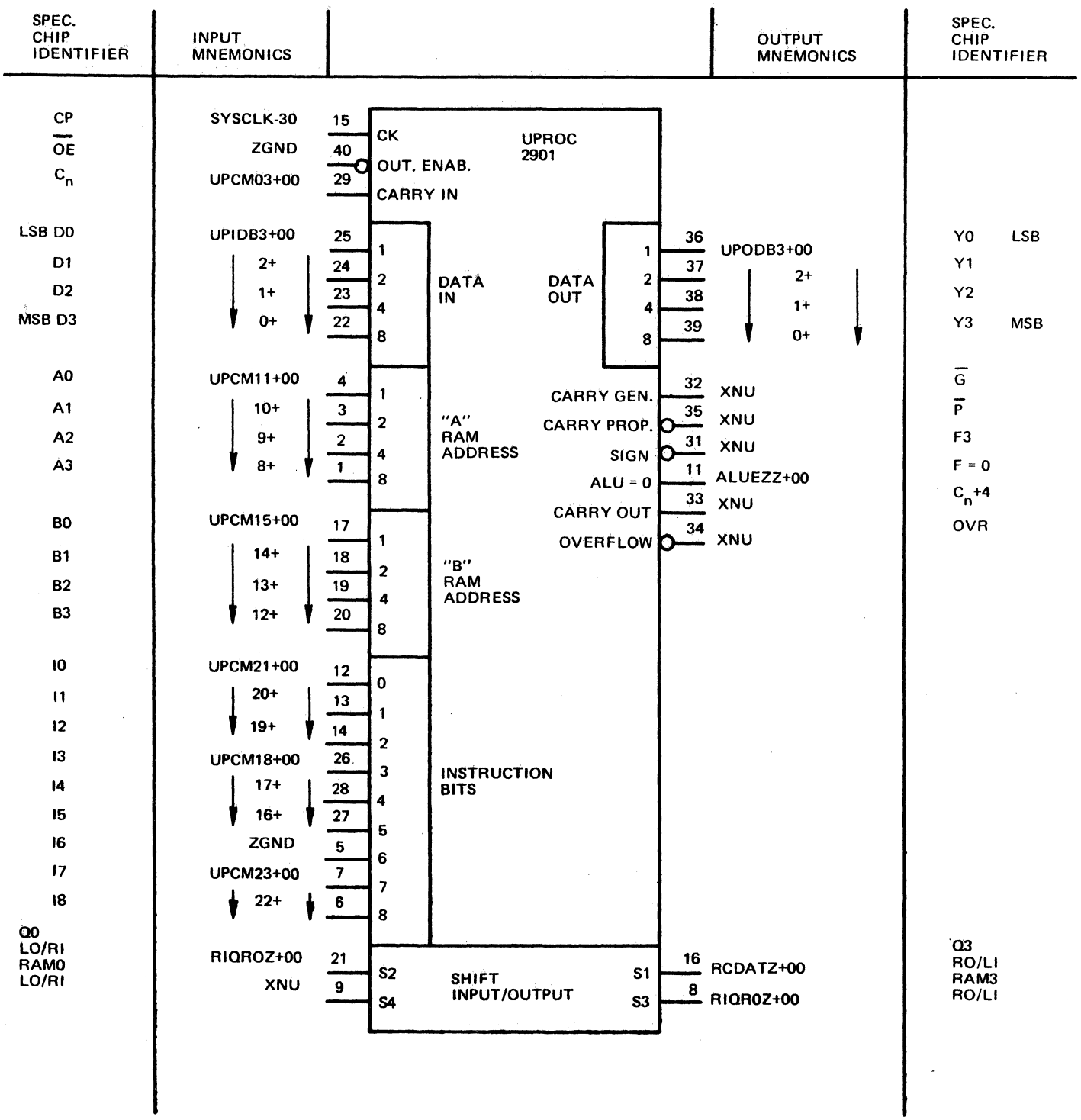


Figure 3-21 RCFCO Routine
(Sheet 2 of 2)

Appendix A

Type 2901 Microprocessor Chip Structure


SPEC. CHIP IDENTIFIER
INPUT MNEMONICS
OUTPUT MNEMONICS
SPEC. CHIP IDENTIFIER
UPROC 2901
CP
OE
Cn
SYSCLK-30
ZGND
UPCM03+00
CK
OUT. ENAB.
CARRY IN
LSB D0
D1
D2
MSB D3
UPIDB3+00
2+
1+
0+
DATA IN
DATA OUT
UPODB3+00
Y0 LSB
Y1
Y2
Y3 MSB
A0
A1
A2
A3
UPCM11+00
10+
9+
8+
"A" RAM ADDRESS
B0
B1
B2
B3
UPCM15+00
14+
13+
12+
"B" RAM ADDRESS
I0
I1
I2
I3
I4
I5
I6
I7
I8
UPCM21+00
20+
19+
UPCM18+00
17+
16+
ZGND
UPCM23+00
22+
INSTRUCTION BITS
CARRY GEN.
CARRY PROP.
SIGN
ALU = 0
CARRY OUT
OVERFLOW
XNU
ALUEZZ+00
G
P
F3
F = 0
Cn+4
OVR
Q0
LO/RI
RAM0
LO/RI
RIQROZ+00
XNU
S2
S4
SHIFT INPUT/OUTPUT
S1
S3
RCDATZ+00
RIQR0Z+00
Q3
RO/LI
RAM3
RO/LI

Appendix B

Installation

Installation information for the DCM9106 High-Level Data Link Control Adapter is contained in the Level 6 Model 34/36 System manual Order No. FL35. Specific information on cabling and cabling hookups is contained in this Appendix.

The adapter is cabled in accordance with the system configuration, which may specify one of three possible hookups.

In most system configurations, the adapter is connected to the Data Communications Equipment (DCE) or to a remote Communications-Pac via modems and a telephone line. Figure B-1 illustrates a typical communications configuration and identifies the type of cable used and the connector type, i.e., male (M).

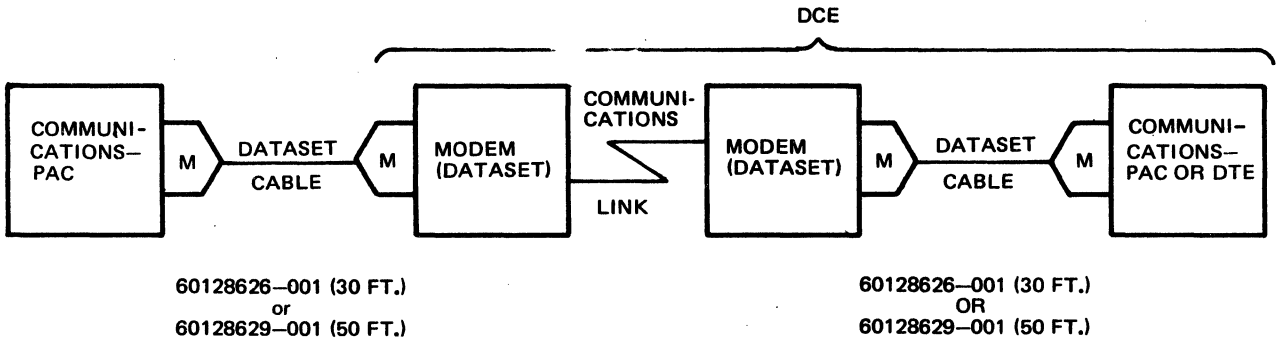
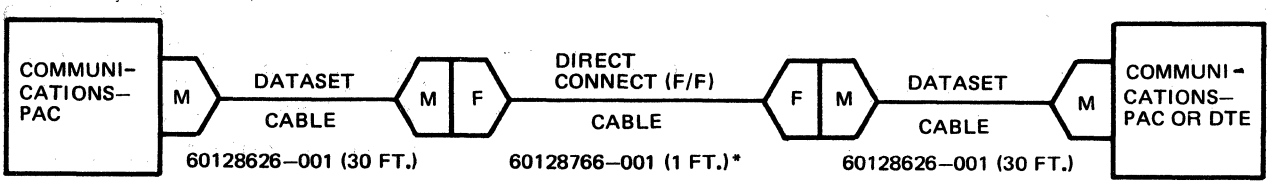


Figure B-1 Typical Modern Configuration

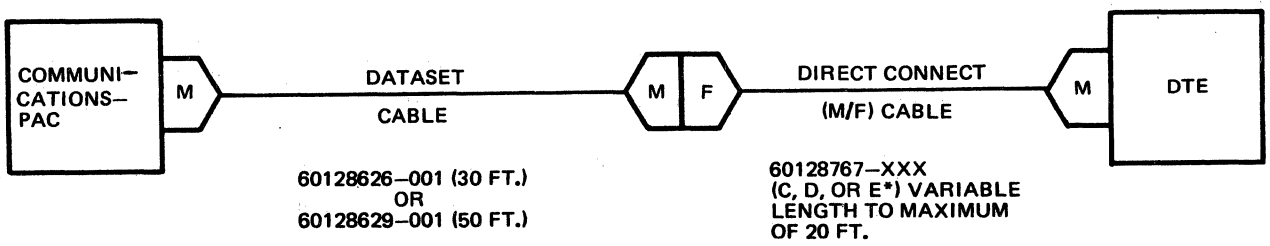
The system can also be configured for a direct connect application. If the two pieces of equipment are close to each other (less than 61 feet apart for RS232C applications), a short jumper cable (direct connect female to female) replaces the modem/telephone line combination. Figure B-2 shows the cabling requirements for this configuration, (i.e., two dataset cables with male (M) connectors and one direct connect with two female (F) connectors).



*CONNECTORS ASSEMBLED TO BOTH ENDS

Figure B-2 Direct Connect Female/Female

For direct connect applications, where the adapter is connected directly to another Communications-Pac or to Data Terminal Equipment without the use of other data communications equipment, the use of a direct connect cable (male to female) and a dataset cable is required. One cable cannot be used for this application due to signal inconsistencies at the connector pinouts. Figure B-3 shows the cable requirements for this configuration, (i.e., one dataset cable with male connectors (M) and one direct connect cable with a male connector (M) and a female (F) connector). The total combined length of the two cables must not exceed 61 feet.



*C – CONNECTORS ASSEMBLED TO BOTH ENDS
D – DISASSEMBLED WITH TWO CONNECTORS SHIPPED UNATTACHED
E – ONE CONNECTOR ASSEMBLED TO CABLE (PADDLE BOARD SIDE)
THE OTHER CONNECTOR SHIPPED BUT NOT ATTACHED.

Figure B-3 Direct Connect Male/Female

Appendix C

T&V Wraparound Tests

Two wraparound tests can be performed on the HDLC adapter through the use of a Test and Verification program.

The first test, DCMS2-Mode A, performs an internal wraparound of the adapter, checking the integrity to transmit and receive data properly. Serial data is sent out of the transmitter and returned directly back to the receiver. The received data must match the transmitted data to verify integrity.

The second test, DCMS2-Mode C, performs an external wraparound of the DCE interface and cable. To perform this external wraparound test, the DCE connector is terminated by an EIA jumper at the DCE end of the cable. This connector is wired as indicated in the following table.

For information pertaining to the sequence of steps required to run these programs, refer to the Level 6 System Checkout and T&V Manual, Order No. AW94. For cabling information, refer to Appendix B of this manual and to the Model 34/36 System Manual Order No. FL35.

Table C-1 EIA Connector Jumpers For DCMS2-Mode C Loop Test

EIA CONNECTOR PIN NUMBER TO BE JUMPERED	INTERFACE SIGNAL NAMES
02 to 03	Transmit Data (02) to Receive Data (03)
06 to 20	Data Set Ready (06) to Terminal Data (20)
04 to 05 to 08	Request to Send (04) to Clear to Send (05) to Carrier Detect (08)
14 to 15	New Sync (14) to Transmit Clock (15)
17 to 22 to 23 to 25	Receive Clock (17) to Ring Indicator (22) to Speed Select (23) and optionally to Reverse Channel Receiver (25)

SSTD HARDWARE PUBLICATIONS
USER COMMENTS FORM

DOCUMENT TITLE: _____

PART NO.: _____

ORDER NO.: _____

ERRORS:

HOW DO YOU USE THIS DOCUMENT?

THEORY ☐

MAINTENANCE ☐

TROUBLESHOOTING ☐

OTHER: _____

_____

DOES THIS MANUAL SATISFY YOUR REQUIREMENTS?

YES ☐ NO ☐

IF NOT, PLEASE EXPLAIN _____

_____

_____

FROM: NAME _____ DATE _____

TITLE _____

COMPANY _____

ADDRESS _____

PLEASE FOLD AND TAPE–
NOTE: U. S. Postal Service will not deliver stapled forms

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 39531 WALTHAM, MA02154

POSTAGE WILL BE PAID BY ADDRESSEE

HONEYWELL INFORMATION SYSTEMS
200 SMITH STREET
WALTHAM, MA 02154

ATTN: PUBLICATIONS, MS872A

CUT ALONG LINE

FOLD ALONG LINE

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 the U.K.: Great West Road, Brentford, Middlesex TW8 9DH
In Australia: 124 Walker Street, North Sydney, N.S.W. 2060
In Mexico: Avenida Nuevo Leon 250, Mexico 11, D.F.

25646, Printed in U.S.A.

FN12, Rev. 1